

Compte rendu de stage

2ème année BTS SIO

Florian Martin

Année 2019-2020

Table des matières

1. Présentation de l'entreprise	1
2. Présentation de la mission	1
2.1. But de l'application	1
2.2. Technologies utilisées	1
2.3. Organisation du travail	2
2.4. Méthodologies et procédés de travail	2
2.5. Prérequis et installation de l'environnement de travail	3
3. Développement de l'application	3
3.1. Initialisation de l'environnement de travail	3
3.2. Développement du frontend de l'application	3
3.3. Création de la base de données.....	22
3.4. Création du backend	25
3.5. Contrôle des données importées.....	32
3.6. Gestion de l'authentification	35
3.6.1. <i>Page de connexion</i>	35
3.6.2. <i>Authentification</i>	38
3.7. Gestion de la sécurité côté backend.....	40
3.7.1. <i>Echange de token</i>	40
3.7.2. <i>Sécurisation des URL de l'API</i>	42
3.8. Gestion de la sécurité côté frontend	43
3.8.1. <i>Echange de token</i>	43
3.8.2. <i>Sécurisation des URL de l'application</i>	44
3.9. Implémentation de la partie admin.....	46
3.9.1. <i>Gestion des droits d'administration</i>	46
3.9.2. <i>Création de l'interface d'administration</i>	46
3.9.3. <i>Implémentation des fonctionnalités côté API</i>	52
4. Mise en industriel de l'application	52
4.1. Mise en industriel du backend	52
4.2. Mise en industriel du frontend.....	56
5. Automatisation du déploiement grâce à une configuration de fichiers	60
6. Documentation	62
7. Gestion de version du projet	66
8. Application annexe	67

1. Présentation de l'entreprise

Michelin est un fabricant de pneumatiques français dont le siège social est à Clermont-Ferrand. Les produits fabriqués sont destinés à tous types de véhicule : automobiles, camions, deux roues, avions, engins de génie civil et agricoles. C'est une multinationale implantée industriellement dans 17 pays.

Pour ce stage de BTS SIO 2ème année option SLAM, j'ai effectué mon stage sur le site de Cataroux de l'entreprise Michelin, à Clermont-Ferrand. Ce stage a été effectué dans le service "Méthode et qualité" qui a pour but de donner des outils et une méthodologie utile aux autres sites Michelin dispersés dans le monde entier. Ce service permet notamment aux autres services Michelin de gérer au mieux le commerce grâce aux différents outils proposés par le service de Cataroux dans lequel je suis.

2. Présentation de la mission

2.1. But de l'application

Michelin possède beaucoup de middlewares qui évoluent au cours du temps. Les versions des middlewares évoluent également, de même que les services proposés par ces derniers, certaines versions devenant obsolètes. Les différentes BS (BS signifiant business service, qui caractérise un quelconque service Michelin dans le monde comme le service recherche et développement français par exemple) utilisent les middlewares qui doivent donc être tenu à jour et évoluer vers les dernières versions afin de ne pas devenir obsolètes.

De ce fait, on m'a demandé de développer une application qui allait permettre aux leaders des différentes BS de Michelin de pouvoir renseigner les versions actuelles de leur middleware, les versions vers lesquelles ils souhaitent migrer et la date prévue de migration. Une application existante était déjà présente mais elle avait été réalisée avec des technologies qui sont devenues obsolètes et de plus cette dernière proposait une interface peu agréable.

2.2. Technologies utilisées

Pour cette mission, je me suis servi, côté frontend, du framework Angular développé par Google, basé sur le langage TypeScript qui est un sur-ensemble du langage JavaScript, associé à l'IDE Visual Studio Code. Du côté du backend, je me suis servi du framework Spring

Boot basé sur le langage Java couplé à l'IDE IntelliJ et du langage d'interrogation de données SQL sous l'IDE MySQL workbench.

Au niveau du versionning, je me suis servi du logiciel de gestion de versions Git en association avec la plateforme GitLab.

Je me suis également servi de l'interpréteur de commande Bash qui permet d'écrire des scripts et qui m'a permis d'interagir avec le serveur Linux de Michelin.

2.3. Organisation du travail

L'application a été réalisée seul. Cependant, toute une équipe de développeur était présente pour moi lorsque j'en avais besoin, lorsque j'avais des questions ou lorsque j'avais besoin d'une petite formation sur certains aspects des différents langages ou outils utilisés.

2.4. Méthodologies et procédés de travail

Michelin possède plusieurs normes, standards et habitudes de travail.

Tout d'abord, la méthode Agile est la méthode adoptée par le service dans lequel j'ai effectué mon stage. La méthode Agile permet de découper un projet en différents modules appelés des "sprint". Un sprint est ensuite découpé en différentes tâches (appelées dans le service où je suis "user story") et ces différentes tâches sont elles-mêmes découpées en autres tâches (appelées "task").

En effet, chaque matin pendant 15 minutes une réunion est organisée. Chaque membre de l'équipe annonce la tâche qu'il a effectué la veille et la tâche qu'il va effectuer dans la journée. Cela permet de faire un bilan chaque jour, de motiver l'équipe et également de savoir qui peut avoir potentiellement besoin d'aide sur une tâche ou même d'annoncer certains problèmes à régler.

De plus, tous les lundis matin et pendant une heure, une réunion est organisée, intitulée "warm up". Cette heure comprend la réunion de 15 minutes exposée précédemment avec en supplément les objectifs à réaliser sur la semaine, le recueil du moral de chaque membre de l'équipe et l'annonce des points positifs et négatifs actuels du projet avec des propositions d'axes d'améliorations.

Également, chaque semaine et durant une heure, une séance est organisée au sein de l'équipe visant à mettre en place une nouvelle "good practice". Par-là, il faut comprendre que cette heure est réservée pour discuter autour d'une nouvelle bonne pratique à mettre en place comme par exemple envoyer un formulaire aux clients 2 mois après la livraison du produit afin de mesurer l'utilisation des services livrés.

Enfin, à chaque fin de sprint, qui dure chez Michelin 4 semaines, une réunion le lundi matin est organisée afin de planifier le sprint suivant, d'assigner les différentes tâches aux différents

développeurs et également de planifier les nouveaux objectifs sur les 4 prochaines semaines. Généralement, en fin de sprint, des démonstrations du projet en cours de développements sont proposés aux supérieurs ou aux clients afin de montrer l'avancement.

2.5. Prérequis et installation de l'environnement de travail

Durant la première semaine de mon stage, on m'a formé sur les différents langages dont j'allais me servir. On m'a également demandé d'installer les différents outils sur mon poste de travail nécessaires au développement.

Lors de cette première semaine, on m'a également présenté le cahier des charges que j'allais devoir respecter ainsi que le but de l'application que j'allais devoir réaliser.

3. Développement de l'application

3.1. Initialisation de l'environnement de travail

Afin de commencer le projet, j'ai tout d'abord dû initialiser un dépôt Git dans le repository du service dans lequel j'ai effectué mon stage. Après cela, j'ai initialisé un projet Angular pour la partie Frontend avec Visual Studio Code puis créé un repository Git en local afin de pouvoir push et pull le dépôt distant. De la même façon, j'ai effectué les mêmes étapes pour mon backend sous l'IDE IntelliJ pour initialiser un projet Spring Boot.

3.2. Développement du frontend de l'application

Chaque mardi, j'avais une réunion avec le chef de projet afin de rendre compte de mon avancement et de réaliser quelques ajustements et modifications de l'existant de l'application dans le but d'éviter l'effet tunnel.

Pour ce qui est de l'interface graphique de l'application, on m'a laissé libre de réaliser mon propre design.

Dans un premier temps, j'ai commencé par réaliser la maquette de l'application, c'est-à-dire que j'ai commencé par créer l'interface graphique de l'application sans implémenter les fonctionnalités derrières les boutons, les appels à l'API ou quoi que ce soit, tout cela dans le but d'avoir un premier visuel de la solution et des retours de mon chef de projet. Une fois que l'interface graphique a été réalisé, j'ai commencé à implémenter les fonctionnalités. La plupart de l'interface graphique de l'application a été réalisé avec Angular Material qui est une librairie externe de composants matériels spécialement développée pour le framework Angular.

Voici l'interface de la page d'accueil lorsqu'un utilisateur est connecté et qu'il peut modifier les valeurs de son inventaire des middlewares :

The screenshot shows a web-based application titled "OBSCO TRACKER". The top navigation bar includes "BSS INVENTORY", "LOG OUT", and several dropdown filters: "Technology", "Current version", "Target version", "BSS", "Application code", "Application instance", "Environment", "Forecast date of the migra...", "Migration completed date", and a "Reset" button.

The main content area is titled "YOUR BSS INVENTORY". It features a table with the following columns:

Technology	BSS	Application code	Application instance	Device name	Current version middleware	Target version middleware	Environment	Number of incoming stream	Number of outgoing stream	Forecast Migration Date	Migration completed date	Update
AD LEGACY	DCSI-BS-RD	AAA	RAAA		EUW	ADR	NONE	0	0	2000-01-01	2000-01-01	<input type="checkbox"/>
AD LEGACY	DCSI-BS-RD	ASQ	CASQ		EUW	ADR	NONE	0	0	2000-01-01	2000-01-01	<input type="checkbox"/>
AD LEGACY	DCSI-BS-RD	BDS	IBDS		EUW	ADR	NONE	0	0	2000-01-01	2000-01-01	<input type="checkbox"/>
AD LEGACY	DCSI-BS-RD	BDS	IBDSCAR0		EUW	ADR	NONE	0	0	2000-01-01	2000-01-01	<input type="checkbox"/>
AD LEGACY	DCSI-BS-RD	BDS	RBDS		EUW	ADR	NONE	0	0	2000-01-01	2000-01-01	<input type="checkbox"/>

Buttons at the top right of the table area include "+ Add values", "Display all BS", "Import", and "Export".

Ci-dessous, une petite partie du code écrit qui correspond à l'équivalent d'un contrôleur dans certain framework. On peut notamment apercevoir la méthode `ngOnInit` qui s'occupe d'effectuer des traitements lorsque le composant qui s'occupe de la vue de la page d'accueil est initialisé. Ici sur cette capture d'écran, on s'occupe de récupérer les technologies, les versions, les valeurs des middlewares et autres dans la base de données afin de pouvoir les afficher à l'utilisateur.

```

144  ngOnInit(): void {
145    this.readOnly = this.authService.readOnly;
146    this.technologyService.getAllCurrVersion().subscribe(
147      | res => this.allCurrVersion = res
148    );
149    this.technologyService.getAllCurrVersion().subscribe(
150      | res => this.currVersion = res
151    );
152    this.technologyService.getAllTargVersion().subscribe(
153      | res => this.targVersion = res
154    );
155    this.technologyService.getAllTargVersion().subscribe(
156      | res => this.allTargVersion = res
157    );
158    this.technologyService.getAllCurrVersionWithTechno().subscribe(
159      | (res: Technology[]) => this.allCurrVersAndTechno = res);
160    this.technologyService.getAllTargVersionWithTechno().subscribe(
161      | (res: Technology[]) => this.allTargVersAndTechno = res);
162    if (!this.readOnly) {
163      this.dataBssService.getValuesByBss(this.authService.userConnected.bss).subscribe(res => {
164        this.dataSource = res;
165        this.dataSource.forEach(
166          element => {
167            this.datePipe.transform(element.migrationCompletedDate, 'yyyy-MM-dd');
168            element.migrationCompletedDate = element.migrationCompletedDate.slice(0, 10);

```

Toujours dans la méthode ngOnInit, on s'occupe d'initialiser la variable qui va contenir les valeurs du tableau en instanciant notamment une classe du framework Angular Material qui permet de faciliter l'implémentation de différentes fonctionnalités telles que la pagination ou la possibilité de trier les valeurs par ordre alphabétique dans le tableau. On aperçoit cela par exemple à la ligne 202 et 203 dans la capture d'écran ci-dessous :

```

195  this.dataSourceSubscription = this.loadDataService.dataSourceObservable.subscribe(
196    (vals) => {
197      if (this.readOnly) {
198        this.dataFormat = new MatTableDataSource(this.allBssDatas);
199      } else {
200        this.dataFormat = new MatTableDataSource(this.dataSource);
201      }
202      this.dataFormat.sort = this.sort;
203      this.dataFormat.paginator = this.paginator;
204

```

Et voici donc la possibilité offerte par Angular d'ajouter une pagination simplement à un tableau avec la possibilité de modifier le nombre d'items afficher par page :

Application instance		Environment		Forecast date of the migra...		Migration completed date				<button>Reset</button>									
<u>YOUR BSS INVENTORY</u>																			
+ Add values <button>Display all BS</button> <button>Import</button> <button>Export</button>																			
Technology	BSS	Application code	Application instance	Device name	Current version middleware	Target version middleware	Environment	Number of incoming stream	Number of outgoing stream	Forecast Migration Date	Migration completed date								
AD LEGACY	DCSI-BS-RD	AAA	RAAA		EUW	ADR	NONE	0	0	2000-01-01	2000-01-01								
AD LEGACY	DCSI-BS-RD	ASQ	CASQ		EUW	ADR	NONE	0	0	2000-01-01	2000-01-01								
AD LEGACY	DCSI-BS-RD	BDS	IBDS		EUW	ADR	NONE	0	0	2000-01-01	2000-01-01								
AD LEGACY	DCSI-BS-RD	BDS	IBDSCARO		EUW	ADR	NONE	0	0	2000-01-01	2000-01-01								
AD LEGACY	DCSI-BS-RD	BDS	RBDS		EUW	ADR	NONE	0	0	2000-01-01	2000-01-01								

+ Add values <button>Display all BS</button> <button>Import</button> <button>Export</button>											
Technology	BSS	Application code	Application instance	Device name	Current version middleware	Target version middleware	Environment	Number of incoming stream	Number of outgoing stream	Forecast Migration Date	Migration completed date
AD LEGACY	DCSI-BS-RD	AAA	RAAA		EUW	ADR	NONE	0	0	2000-01-01	2000-01-01
AD LEGACY	DCSI-BS-RD	ASQ	CASQ		EUW	ADR	NONE	0	0	2000-01-01	2000-01-01
AD LEGACY	DCSI-BS-RD	BDS	IBDS		EUW	ADR	NONE	0	0	2000-01-01	2000-01-01
AD LEGACY	DCSI-BS-RD	BDS	IBDSCARO		EUW	ADR	NONE	0	0	2000-01-01	2000-01-01
AD LEGACY	DCSI-BS-RD	BDS	RBDS		EUW	ADR	NONE	0	0	2000-01-01	2000-01-01

Sur la capture ci-dessous, on peut voir que 50 items sont affichés par page. La fenêtre du navigateur n'étant pas assez grande pour afficher 50 items par page, le header du tableau reste toujours affiché en haut de l'écran lors du scroll afin de savoir en permanence à quoi correspondent les colonnes :

Technology	BSS	Application code	Application instance	Device name	Current version middleware	Target version middleware	Environment	Number of incoming stream	Number of outgoing stream	Forecast Migration Date	Migration completed date	Update	Delete
AD LEGACY	BS-RD	RFI	IRFIEUR2		EUW	ADR	NONE	0	0	2000-01-01	2000-01-01		
AD LEGACY	DCSI-BS-RD	RFI	IRFIGBL0		EUW	ADR	NONE	0	0	2000-01-01	2000-01-01		
AD LEGACY	DCSI-BS-RD	RFI	RRFI0		EUW	ADR	NONE	0	0	2000-01-01	2000-01-01		
AD LEGACY	DCSI-BS-RD	RFI	RRFI1		EUW	ADR	NONE	0	0	2000-01-01	2000-01-01		
AD LEGACY	DCSI-BS-RD	RFI	RRFI10		EUW	ADR	NONE	0	0	2000-01-01	2000-01-01		
AD LEGACY	DCSI-BS-RD	RFI	RRFI11		EUW	ADR	NONE	0	0	2000-01-01	2000-01-01		
AD LEGACY	DCSI-BS-RD	RFI	RRFI12		EUW	ADR	NONE	0	0	2000-01-01	2000-01-01		
AD LEGACY	DCSI-BS-RD	RFI	RRFI2		EUW	ADR	NONE	0	0	2000-01-01	2000-01-01		
AD LEGACY	DCSI-BS-RD	RFI	RRFI3		EUW	ADR	NONE	0	0	2000-01-01	2000-01-01		

Items per page: 101 – 150 of 302 |< < > >|

Dans la capture d'écran suivante, nous nous trouvons dans un service Angular. Un service, dans le framework Angular, est une classe partagée par toute l'application. C'est ici que les méthodes qui servent à toute l'application sont écrites. Ici, par exemple, nous avons deux méthodes qui permettent de récupérer les valeurs des middlewares grâce à un appel au backend avec la méthode HTTP GET. La variable httpClient est une instance du module HttpClient du framework Angular qui permet d'utiliser les différentes méthodes HTTP telles que POST, GET, DELETE ou PUT pour les plus connues. Ce module facilite grandement l'utilisation des requêtes HTTP pour interagir avec des API par exemple. Voici deux exemples :

```

75  getValues() {
76    return this.httpClient.get<DataFormat[]>(` ${this.baseUrl}datas`);
77  }
78
79  getValuesByBss(bss: string) {
80    return this.httpClient.get<DataFormat[]>(` ${this.baseUrl}datas/${bss}`);
81  }

```

Sur ce tableau, j'ai ajouté des filtres, facilitant grandement la possibilité de rechercher une valeur lorsqu'il existe beaucoup de valeurs. Par exemple, sur la capture d'écran suivante, on peut voir que j'ai recherché tous les middlewares se trouvant dans la technologie WMQ, pour la version 7,5 et comprenant dans leur code applicatif la suite de caractère "RWF". On peut ainsi voir dans le tableau qu'uniquement les lignes en relation avec le filtre sont affichés :

The screenshot shows a web-based application interface for managing BSS inventory. At the top, there are several filter inputs: 'WMQ' (selected), '7.5', 'Target version', 'BSS', 'RWF' (highlighted in blue), 'Application instance' (with a clear button), 'Environment' (with a clear button), 'Forecast date of the migra...', 'Migration completed date' (with a clear button), and a 'Reset' button. Below the filters is a title 'YOUR BSS INVENTORY'. Underneath is a table with the following columns: Technology, BSS, Application code, Application instance, Device name, Current version middleware, Target version middleware, Environment, Number of incoming stream, Number of outgoing stream, Forecast Migration Date, Migration completed date, and Update (with a trash icon). The table contains five rows of data for WMQ components.

Technology	BSS	Application code	Application instance	Device name	Current version middleware	Target version middleware	Environment	Number of incoming stream	Number of outgoing stream	Forecast Migration Date	Migration completed date	Update	
WMQ	DCSI-BS-RD	RWF	RALEL01		7.5	9.0	NONE	0	0	2000-01-01	2000-01-01		
WMQ	DCSI-BS-RD	RWF	RARAL01		7.5	9.0	NONE	0	0	2000-01-01	2000-01-01		
WMQ	DCSI-BS-RD	RWF	RCPYL01		7.5	9.0	NONE	0	0	2000-01-01	2000-01-01		
WMQ	DCSI-BS-RD	RWF	RRYOL01		7.5	9.0	NONE	0	0	2000-01-01	2000-01-01		
WMQ	DCSI-BS-RD	RWF	RVITL01		7.5	9.0	NONE	0	0	2000-01-01	2000-01-01		

Ci-dessous des extraits de code ayant permis la possibilité de filtrer. On retrouve notamment le fichier html sur lequel on place des listeners sur le “keyup” des touches (ce qui déclenche une méthode à chaque fois que l’utilisateur relâche une touche du clavier) sur les valeurs entrées dans les balises input des filtres. La seconde capture d’écran correspond à un morceau de code écrit qui permet de filtrer les valeurs du tableau (qui sont contenues dans la variable “dataFormat”) :

```

40      <mat-form-field class="filter-form">
41        <mat-label>Application code</mat-label>
42        <input (keyup)="applyFilter()" matInput [(ngModel)]="appCodeSearch"
43          placeholder="Application code"
44          autocomplete="off">
45        <button mat-button matSuffix mat-icon-button aria-label="Clear"
46          (click)="applyFilter('appCode')">
47          <mat-icon>close</mat-icon>
48        </button>
49      </mat-form-field>

```

```

this.dataFormat.filterPredicate = (data, filter) => {
  let found = true;
  if (found && this.technologySearch) {
    found = found && data.technology.toLowerCase().includes(this.technologySearch.toLowerCase())
  }
  if (found && this.bssSearch) {
    found = found && data.bss.toLowerCase().includes(this.bssSearch.toLowerCase());
  }
  if (found && this.appCodeSearch) {
    found = found && data.appCode.toLowerCase().includes(this.appCodeSearch.toLowerCase());
  }
  if (found && this.appInstanceSearch) {
    found = found && data.appInstance.toLowerCase().includes(this.appInstanceSearch.toLowerCase());
  }
  if (found && this.currVersSearch) {
    found = found && data.currVersion.toLowerCase().includes(this.currVersSearch.toLowerCase());
  }
}

```

Il faut savoir qu'en fonction de la technologie, les versions disponibles ne sont pas équivalentes. Par exemple pour une technologie A il existe les versions 6.1 et 6.2 alors que pour une technologie B il existe les versions 7.1 et 7.2. On m'a donc demandé de prendre en compte ce point-là dans le filtrage des valeurs. En effet, il ne fallait pas rendre possible le fait de choisir la version 7.1 pour la technologie A si cette dernière ne possède pas la version 7.1. En revanche, si aucune technologie n'était choisie, alors toutes les versions devaient être affichées.

Sur la capture d'écran, on peut voir qu'aucune technologie n'est sélectionnée, ainsi, on peut choisir n'importe quelle technologie :

Au clic sur une technologie, la méthode “getVersionsOfTechnology” est appelée avec en paramètre la technologie sélectionnée par l'utilisateur :

```

4      <mat-form-field class="filter-form">
5          <mat-label>Technology</mat-label>
6          <mat-select [(ngModel)]="technologySearch">
7              <mat-option (click)="getVersionsOfTechnology('none')"
8                  technologySearch (click)="applyFilter('noneTechnology')"
9                  None
10             </mat-option>
11             <mat-option (click)="getVersionsOfTechnology(value)" (click)="applyFilter()"
12                 [value]="value" *ngFor="let value of technology">
13                 {{value}}
14             </mat-option>
15         </mat-select>
16     </mat-form-field>

```

Cette méthode est détaillée ci-dessous. Elle s'occupe notamment d'effectuer un appel dans la base de données pour récupérer les versions d'une technologie donnée. En fonction des valeurs rentrées, la liste déroulante pour les versions actuelles et cibles sont mises à jour :

```

718 /**
719  * Return the current and target versions related to a technology
720 */
721 getVersionsOfTechnology(technology) {
722     this.technologyService.getCurrentVersionByTechnology(technology).subscribe(
723         (res) => this.currVersion = res
724     );
725     this.technologyService.getTargetVersionByTechnology(technology).subscribe(
726         (res) => this.targVersion = res
727     );
728
729     if (technology === 'none') {
730         this.technologyService.getAllCurrVersion().subscribe(
731             res => this.currVersion = res
732         );
733         this.technologyService.getAllTargVersion().subscribe(
734             res => this.targVersion = res
735         );
736     }
737 }
738

```

En effet, on peut voir qu'en ayant sélectionné la technologie WMQ, on a cette fois-ci que les versions disponibles pour cette technologie :

The screenshot shows a software interface with a dropdown menu open for 'Target version'. The options listed are 6.5, 7.0, 7.5, 8.0, and 9.0. Below the dropdown, there is a section titled 'R BSS INVENTORY' with a table header containing columns: Target version, Environment, Number of incoming, Number of outgoing, Forecast Migration, Migration completed, and Update. There are also buttons for 'Display all BS', 'Import', and 'Reset'.

Ensuite, il est possible de mettre à jour les lignes présentes dans le tableau en cliquant sur le bouton dans la colonne update. Au clic, une fenêtre s'ouvre avec un formulaire permettant de mettre à jour les valeurs de la ligne sélectionnée :

The screenshot shows an 'Update' dialog box overlaid on a table. The dialog has fields for Technology (AD LEGACY), BSS (DCSI-BS-RD), Application code (AAA), Application instance (RAAA), Device name, Current version (EUW), Forecast migration date (2000-01-01), Target version (ADR), Migration completed date (2000-01-01), and Number of outgoing stream (0). The table below the dialog shows several rows of data with an 'Update' button next to each row.

On peut voir qu'au clic sur le bouton, on appelle la méthode « onEdit » :

```

161      <ng-container *ngIf="!isAllInventory" matColumnDef="update">
162          <th mat-header-cell *matHeaderCellDef>Update</th>
163          <td mat-cell *matCellDef="let row">
164              <button *ngIf="!isAllInventory" mat-icon-button (click)="onEdit(row)">
165                  <mat-icon>launch</mat-icon>
166              </button>
167          </td>
168      </ng-container>

```

Ensuite, il est possible de mettre à jour les lignes présentes dans le tableau en cliquant sur le bouton dans la colonne update. Au clic, une fenêtre s'ouvre avec un formulaire permettant de

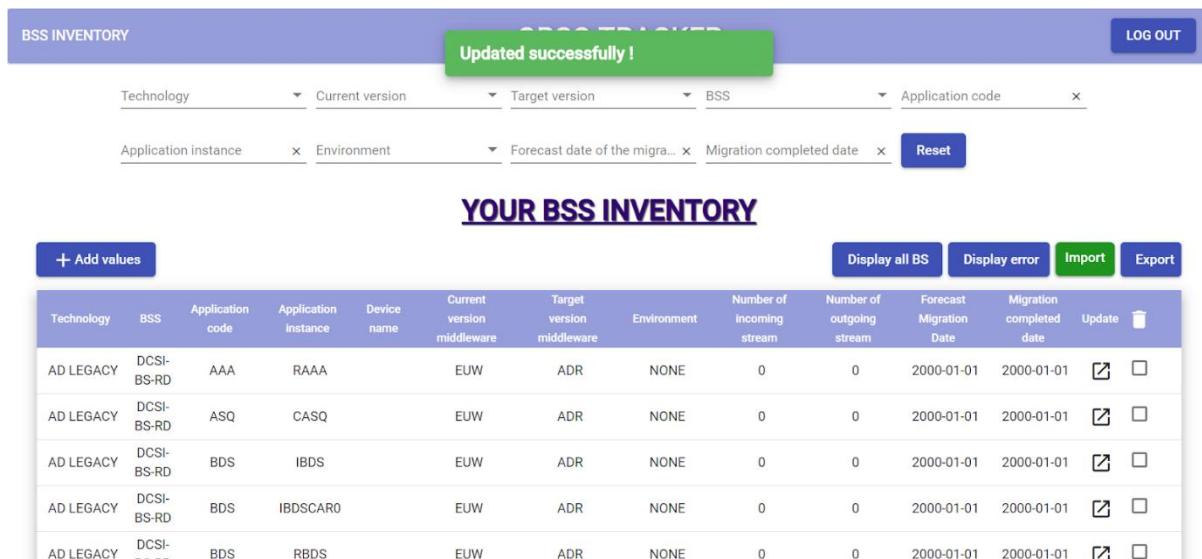
La méthode « onEdit » est présentée juste ci-dessous. Cette méthode s'occupe de récupérer les valeurs de la ligne que l'on souhaite modifier. Une fois les valeurs récupérées, on appelle le composant qui s'occupe d'afficher la pop-up de mise à jour en envoyant les valeurs de la ligne :

```
558  /**
559   * On the click on the edit button, we open the form dialog
560   * that allows to edit the row
561   */
562   onEdit(row) {
563     const dialogConfig = this.dialog.open(EditRowComponent, {
564       data: {
565         values: this.dataSource[this.dataSource.indexOf(row)],
566         ind: this.dataSource.indexOf(row)
567       },
568       width: '50%',
569       autoFocus: true,
```

Enfin, le composant qui affiche la pop-up récupère les valeurs qui lui sont transmises pour les afficher à l'utilisateur. C'est ce que l'on peut voir dans la méthode ngOnInit. On peut également voir entre autre dans cette méthode le service "formService" qui s'occupe d'initialiser le formulaire affiché à l'utilisateur :

```
42   constructor(
43     public dialogRef: MatDialogRef<EditRowComponent>,
44     @Inject(MAT_DIALOG_DATA) public data: any,
45     private formService: FormService,
46     private dialogService: DialogService,
47     private loadDataService: LoadDataService,
48     private technologyService: TechnologyService,
49     private bssService: BssService,
50     private environmentService: EnvironmentService
51   ) { }
52
53   ngOnInit(): void {
54     this.initialAppInstance = this.data.values.appInstance;
55     this.initialAppCode = this.data.values.appCode;
56     this.initialBss = this.data.values.bss;
57     this.initialTechnology = this.data.values.technology;
58     this.editRowForm = this.formService.populateForm(this.data.values);
59     this.bssService.getBss().subscribe(
60       res => {
61         this.bss = res;
62       }
63     );
64 }
```

Lorsque l'utilisateur a terminé de mettre à jour les valeurs, il n'a plus qu'à cliquer sur le bouton update ce qui va mettre à jour la ligne dans la base de données par l'intermédiaire d'une requête POST au back. Si la mise à jour a été effectuée, on affiche une pop-up pour l'indiquer à l'utilisateur en haut au milieu de l'écran :



The screenshot shows a web-based application titled "BSS INVENTORY". At the top, there is a search bar with dropdowns for "Technology", "Current version", "Target version", "BSS", and "Application code", along with buttons for "LOG OUT", "Reset", "Display all BS", "Display error", "Import", and "Export". A green banner at the top center says "Updated successfully!". Below the search bar is a section titled "YOUR BSS INVENTORY" containing a table with the following data:

Technology	BSS	Application code	Application instance	Device name	Current version middleware	Target version middleware	Environment	Number of incoming stream	Number of outgoing stream	Forecast Migration Date	Migration completed date	Update
AD LEGACY	DCSI-BS-RD	AAA	RAAA		EUW	ADR	NONE	0	0	2000-01-01	2000-01-01	<input checked="" type="checkbox"/> <input type="checkbox"/>
AD LEGACY	DCSI-BS-RD	ASQ	CASQ		EUW	ADR	NONE	0	0	2000-01-01	2000-01-01	<input checked="" type="checkbox"/> <input type="checkbox"/>
AD LEGACY	DCSI-BS-RD	BDS	IBDS		EUW	ADR	NONE	0	0	2000-01-01	2000-01-01	<input checked="" type="checkbox"/> <input type="checkbox"/>
AD LEGACY	DCSI-BS-RD	BDS	IBDSCARO		EUW	ADR	NONE	0	0	2000-01-01	2000-01-01	<input checked="" type="checkbox"/> <input type="checkbox"/>
AD LEGACY	DCSI-BS-RD	BDS	RBDS		EUW	ADR	NONE	0	0	2000-01-01	2000-01-01	<input checked="" type="checkbox"/> <input type="checkbox"/>

De la même façon, on peut ajouter une nouvelle ligne dans la base de données en cliquant sur le bouton "Add values" en haut à gauche du tableau. La soumission du formulaire n'est rendue possible que lorsque tous les champs requis sont renseignés, sinon on affiche des messages d'erreur indiquant à l'utilisateur de renseigner correctement les champs. On peut également voir ici par exemple que le bouton "submit" est désactivé dans la première capture d'écran et cliquable dans la seconde capture d'écran quand les valeurs sont valides :

:

The screenshot shows a modal dialog titled "Add new values" overlaid on a main application interface. The modal contains two columns of form fields:

Technology*	BSS*
Application code*	Application instance*
Trigram (XXX format)	
Device name	Environment*
Current version*	Target version*
Forecast of the migration date* (date in yyyy-mm-dd format)	Migration completed date* (date in yyyy-mm-dd format)
Number of incoming stream*	Number of outgoing stream*

Below the modal are two buttons: "Submit" (blue) and "Reset" (red).

This screenshot shows the same modal dialog with different input values filled in:

Technology*	BSS*
WMQ	DCSI-BS-RD
Application code*	Application instance*
TEST	TEST
Trigram (XXX format)	Environment*
Device name	PROD
TEST	Target version*
Current version*	9.0
7.0	Migration completed date*
Forecast of the migration date* 2020-01-01	(date in yyyy-mm-dd format)
(date in yyyy-mm-dd format)	Number of incoming stream*
Number of outgoing stream*	1

Below the modal are two buttons: "Submit" (blue) and "Reset" (red).

Comme on peut le voir dans la balise d'Angular Material "mat-error" ci-dessous, des messages d'erreur sont affichés :

```

8   <form [formGroup]="newRowForm" (ngSubmit)="onSubmitForm()">
9     <mat-grid-list cols="2" rowHeight="200px">
10    <mat-grid-tile rowspan="2" colspan="1">
11      <div class="controles-container">
12        <mat-form-field>
13          <mat-label>Technology*</mat-label>
14          <mat-select matTooltip="Technology used" matTooltipPosition="above" formControlName="technology">
15            <mat-option (click)="getVersionsOfTechnology(value)" [value]="value" *ngFor="let value of technology">
16              {{value}}
17            </mat-option>
18          </mat-select>
19          <mat-error>
20            This field is required
21          </mat-error>
22        </mat-form-field>
23        <mat-form-field>
24          <input matInput formControlName="appCode" matTooltip="The code of the application (XXX)" matTooltipPosition="above" place-
25          <mat-hint>Trigram (XXX format)</mat-hint>
26          <mat-error>
27            This field is required
28          </mat-error>
29        </mat-form-field>

```

Ici, on lie la propriété “disabled” à la ligne à la propriété “invalid” de la propriété newRowForm qui contient le formulaire de la pop-up permettant d’activer ou de désactiver le bouton « Submit » :

```
117    <div class="button-row">
118      <button type="submit" mat-raised-button color="primary"
119        [disabled]="newRowForm.invalid">Submit</button>
120      <button type="reset" mat-raised-button color="warn" (click)="onReset()">Reset</button>
121    </div>
122  </form>
```

De la même façon que pour les filtres, l’utilisateur qui entre une nouvelle ligne ne peut pas renseigner n’importe quelles valeurs. C’est pour cela que certains champs possèdent une liste de valeurs prédéfinies qui sont récupérées dans la base de données. On retrouve notamment le fait que les versions présentes dans les listes sont fonctions de la technologie sélectionnée. Dans les captures d’écran ci-dessous, on voit que les valeurs sont récupérées dans la base de données lors de l’initialisation du composant dans la méthode ngOnInit et qu’en fonction de la technologie que l’on choisit, on récupère uniquement les versions disponibles pour celle-ci :

```
36  constructor(
37    public dialogRef: MatDialogRef<NewRowComponent>,
38    @Inject(MAT_DIALOG_DATA) public data: any,
39    private formService: FormService,
40    private loadDataService: LoadDataService,
41    private dataBssService: DataBssService,
42    private technologyService: TechnologyService,
43    private bssService: BssService,
44    private environmentService: EnvironmentService
45  ) { }
46
47  ngOnInit() {
48    this.newRowForm = this.formService.initForm();
49    this.bssService.getBss().subscribe([
50      res => {
51        this.bss = res;
52      }
53    ]);
54    this.environmentService.getEnvironment().subscribe(
55      res => {
56        this.environment = res;
57      }
58    );
59    this.technologyService.getTechnologies().subscribe(
60      res => {
61        this.technology = res;
```

```

89  /**
90  * Return the current and target versions related to a technology
91  */
92  getVersionsOfTechnology(technology) {
93    this.technologyService.getCurrentVersionByTechnology(technology).subscribe(
94      (res) => this.currVersion = res
95    );
96    this.technologyService.getTargetVersionByTechnology(technology).subscribe(
97      (res) => this.targVersion = res
98    );
99  }
100}
101

```

Voici l'exemple du service que j'ai créé et qui s'occupe par exemple des appels au backend permettant de récupérer les différentes informations en relation avec les technologies. On peut voir que chaque méthode du service s'occupe de récupérer, grâce à la méthode HTTP GET, différentes informations par l'intermédiaire d'un appel au backend :

```

9  export class TechnologyService {
10
11  private baseUrl = environment.baseUrl;
12
13  constructor(private httpClient: HttpClient) { }
14
15  getTechnologies() {
16    return this.httpClient.get<string[]>(`${this.baseUrl}technology`);
17  }
18
19  getTargetVersionByTechnology(technology: string) {
20    return this.httpClient.get<string[]>(` ${this.baseUrl}technology/target/${technology}`);
21  }
22
23  getCurrentVersionByTechnology(technology: string) {
24    return this.httpClient.get<string[]>(` ${this.baseUrl}technology/current/${technology}`);
25  }
26
27  getAllCurrVersionWithTechno() {
28    return this.httpClient.get<Technology[]>(` ${this.baseUrl}technology/current/`);
29  }

```

Également, pour contrôler les valeurs entrées par les utilisateurs, je me suis occupé d'ajouter des validateurs prédéfinis par Angular et même d'en créer moi-même. On peut voir que la date entrée ci-dessous n'est pas au bon format (on attend le format américain yyyy-mm-dd). Ainsi, le formulaire ne peut pas être soumis. De même, les nombres négatifs ne sont pas acceptés. Ainsi, on affiche pour chacun des inputs des messages d'information :

Ici on se trouve dans la méthode « initForm » du service qui s'occupe d'initialiser les formulaires. Chaque ligne représente un champ du formulaire. On retrouve notamment les validateurs. Validators.required signifie par exemple que le champ est requis et qu'il ne peut pas être vide. InputValidator.containsPositiveIntegerNumber est un validateur que j'ai écrit moi-même pour valider que la valeur est un nombre nul ou positif :

```

17  initForm() {
18    this.formGroup = this.formBuilder.group({
19      technology: ['', [Validators.required]],
20      bss: ['', [Validators.required]],
21      appCode: ['', [Validators.required]],
22      appInstance: ['', [Validators.required]],
23      deviceName: ['',],
24      currVersion: ['', [Validators.required]],
25      targVersion: ['', [Validators.required]],
26      environment: ['', [Validators.required]],
27      numbInStream: ['', [Validators.required, InputValidator.containsPositiveIntegerNumber]],
28      numbOutStream: ['', [Validators.required, InputValidator.containsPositiveIntegerNumber]],
29      forecastMigrationDate: ['', [Validators.required, InputValidator.containsValidReverseAmericanDate]],
30      migrationCompletedDate: ['', [InputValidator.containsValidReverseAmericanDateOrNull]]
31    });
32    return this.formGroup;
33  }

```

On retrouve ci-dessous deux validateurs personnalisés que j'ai écrit:

```

71  /**
72   * Check that the control value contains a valid reverse american date format (yyyy-mm-dd)
73   * @param control the control who contains the value to test
74   */
75  static containsValidReverseAmericanDateOrNull(control: FormControl): ValidationResult {
76    const regexpDate = new RegExp(/([12]\d{3}-(0[1-9]|1[0-2])-(0[1-9]|1[0-2]))$/g);
77    if (!regexpDate.test(control.value) && (control.value)) {
78      return {invalidDate: true};
79    }
80  }
81
82 /**
83  * Check that the control value contains a positive integer number (can't be zero)
84  * @param control the control who contains the value to test
85  */
86  static containsPositiveIntegerNumberNotZero(control: FormControl): ValidationResult {
87    const regexpPositiveInteger = new RegExp(/^[-]?[1-9][0-9]*$/g);
88    if (!regexpPositiveInteger.test(control.value)) {
89      return {invalidDate: true};
90    }
91  }
92

```

Enfin, lors de la soumission du formulaire, on appelle la méthode `onSubmitForm` qui s'occupe d'enregistrer dans la base de données les valeurs entrées et de fermer le pop-up en appelant notamment la méthode `postNewValue` présentée dans la seconde capture d'écran :

```

69  onSubmitForm() {
70    this.loadDataService.addRow(this.newRowForm.value);
71    const data = this.dataBssService.postNewValue(this.newRowForm.value).subscribe();
72    this.onClose();
73  }
74
75 /**
76  * Close the form dialog
77  */
78 onClose() {
79   this.dialogRef.close();
80 }
81
82 /**
83  * Clear all the values enter in the form
84  */
85 onReset() {
86   this.newRowForm.reset();
87 }

```

```

15     postnewValue(value) {
16         const newValue = new DataFormat()
17         | value.technology,
18         | value.bss,
19         | value.appCode,
20         | value.appInstance,
21         | value.deviceName,
22         | value.currVersion,
23         | value.targVersion,
24         | value.environment,
25         | value.numbInStream,
26         | value.numbOutStream,
27         | value.forecastMigrationDate,
28         | value.migrationCompletedDate
29     ];
30     if (newValue.migrationCompletedDate === '') {
31         newValue.migrationCompletedDate = '2000-01-01';
32     }
33     return this.httpClient.post<DataFormat>(` ${this.baseUrl}data`, newValue);
34 }

```

Il est également possible dans l'application de supprimer un ou plusieurs enregistrements à la fois en cochant les check box et en cliquant sur le logo de la poubelle dans la dernière colonne du tableau :

Technology	BSS	Application code	Application instance	Device name	Current version middleware	Target version middleware	Environment	Number of incoming stream	Number of outgoing stream	Forecast Migration Date	Migration completed date	Update
AD LEGACY	DCSI-BS-RD	AAA	RAAA		EUW	ADR	NONE	0	0	2000-01-01	2000-01-01	<input checked="" type="checkbox"/>
AD LEGACY	DCSI-BS-RD	ASQ	CASQ		EUW	ADR	NONE	0	0	2000-01-01	2000-01-01	<input checked="" type="checkbox"/>
AD LEGACY	DCSI-BS-RD	BDS	IBDS		EUW	ADR	NONE	0	0	2000-01-01	2000-01-01	<input checked="" type="checkbox"/>
AD LEGACY	DCSI-BS-RD	BDS	IBDSCARO		EUW	ADR	NONE	0	0	2000-01-01	2000-01-01	<input checked="" type="checkbox"/>
AD LEGACY	DCSI-BS-RD	BDS	RBDS		EUW	ADR	NONE	0	0	2000-01-01	2000-01-01	<input type="checkbox"/>

Ici, la méthode « onDelete » est celle qui s'occupe de supprimer des enregistrements en appelant notamment la méthode « deleteRow » qui s'occupe, elle, de supprimer une ligne du tableau de la variable locale. Egalement, la méthode « onDelete » fait appel à la méthode « deleteValue » qui permet d'effectuer une requête HTTP DELETE qui va supprimer les enregistrements dans la base de données :

```

577  /**
578  * On the click on the button delete, we delete the row
579  */
580 onDelete() {
581   if (this.rowToDelete.length !== 0) {
582     this.dialogService.openConfigDialog(this.rowToDelete.length + ' row to delete. Are you sure to del')
583     .afterClosed().subscribe(res => {
584       if (res) {
585         this.rowToDelete.forEach(element => {
586           const dataDeleted = this.dataBssService.deleteValue(
587             element.technology,
588             element.bss,
589             element.appCode,
590             element.appInstance,
591             element.environment
592           ).subscribe();
593           this.loadDataService.deleteRow(this.dataSource.indexOf(element));
594         });
595         this.resetFilter();
596         this.applyFilter();
597         this.notificationService.warn('Deleted successfully !');
598       }
599     });
600   });
}

```

La méthode « deleteValue » s'occupe enfin d'effectuer le DELETE en appelant le back :

```

63  deleteValue[
64    technology: string,
65    bss: string,
66    appCode: string,
67    appInstance: string,
68    env: string
69  ] {
70   return this.httpClient.delete<string>(
71     `${this.baseUrl}data/${technology}/${bss}/${appCode}/${appInstance}/${env}`
72   );
73 }

```

Lorsque la suppression a été réussie, un pop-up d'information s'affiche en haut au milieu de l'écran :

The screenshot shows the BSS Inventory application interface. At the top, there is a navigation bar with 'LOG OUT'. Below it is a search bar with dropdowns for 'Technology', 'Current version', 'Target version', 'BSS', 'Application code', 'Application instance', 'Environment', 'Forecast date of the mig...', and 'Migration completed date'. A 'Reset' button is also present. An orange notification bar in the center says 'Deleted successfully !'. Below the search bar is a section titled 'YOUR BSS INVENTORY' with a table. The table has columns: Technology, BSS, Application code, Application instance, Device name, Current version middleware, Target version middleware, Environment, Number of incoming stream, Number of outgoing stream, Forecast Migration Date, Migration completed date, and Update. There are five rows of data in the table.

Technology	BSS	Application code	Application instance	Device name	Current version middleware	Target version middleware	Environment	Number of incoming stream	Number of outgoing stream	Forecast Migration Date	Migration completed date	Update
AD LEGACY	DCSI-BS-RD	AAA	RAAA		EUW	ADR	NONE	0	0	2000-01-01	2000-01-01	<input type="checkbox"/> <input type="checkbox"/>
AD LEGACY	DCSI-BS-RD	ASQ	CASQ		EUW	ADR	NONE	0	0	2000-01-01	2000-01-01	<input type="checkbox"/> <input type="checkbox"/>
AD LEGACY	DCSI-BS-RD	BDS	IBDS		EUW	ADR	NONE	0	0	2000-01-01	2000-01-01	<input type="checkbox"/> <input type="checkbox"/>
AD LEGACY	DCSI-BS-RD	BDS	IBDSCAR0		EUW	ADR	NONE	0	0	2000-01-01	2000-01-01	<input type="checkbox"/> <input type="checkbox"/>
AD LEGACY	DCSI-BS-RD	BDS	RBDS		EUW	ADR	NONE	0	0	2000-01-01	2000-01-01	<input type="checkbox"/> <input type="checkbox"/>

Ici, un extrait du code qui permet d'afficher la pop-up grâce aux classes MatSnackBarConfig et MatSnackBar d'Angular Material qui effectuent la plupart des traitements à la place du développeur :

```
7  export class NotificationService {
8
9    constructor(public snackBar: MatSnackBar) { }
10
11   /* Config the snackbar */
12   config: MatSnackBarConfig = [
13     duration: 5000,
14     horizontalPosition: 'center',
15     verticalPosition: 'top'
16   ];
17
18   success(msg) {
19     this.config.panelClass = ['notification', 'success'];
20     this.snackBar.open(msg, '', this.config);
21   }
22
23   warn(msg) {
24     this.config.panelClass = ['notification', 'warn'];
25     this.snackBar.open(msg, '', this.config);
26   }
27 }
```

Il fallait donner la possibilité aux leaders des BS de pouvoir importer et exporter les valeurs présentes dans le tableau de l'application. Pour cela, après quelques recherches sur Internet, j'ai trouvé une bibliothèque externe, nommée "xlsx" qui permet de lire un fichier excel importé et également de créer un fichier excel à partir de valeurs de l'application.

Comme on peut le voir dans les captures d'écran précédentes, j'ai donc implémenté des boutons import et export permettant respectivement d'importer un fichier excel et d'exporter un fichier excel.

Voici un extrait de code de la fonction qui s'occupe de convertir les valeurs du tableau (filtrées ou non) au format excel et d'ouvrir un fichier excel avec les valeurs souhaitées. Cela a été possible grâce à la bibliothèque "xlsx" qui propose les méthodes nécessaires :

```

        for (let i = 0; i < this.dataFormat.filteredData.length; i++) {
            this.dataExcelFormat.push([]);
            this.dataExcelFormat[i + 1][0] = this.dataFormat.filteredData[i].technology;
            this.dataExcelFormat[i + 1][1] = this.dataFormat.filteredData[i].bss;
            this.dataExcelFormat[i + 1][2] = this.dataFormat.filteredData[i].appCode;
            this.dataExcelFormat[i + 1][3] = this.dataFormat.filteredData[i].appInstance;
            this.dataExcelFormat[i + 1][4] = this.dataFormat.filteredData[i].deviceName;
            this.dataExcelFormat[i + 1][5] = this.dataFormat.filteredData[i].currVersion;
            this.dataExcelFormat[i + 1][6] = this.dataFormat.filteredData[i].targVersion;
            this.dataExcelFormat[i + 1][7] = this.dataFormat.filteredData[i].environment;
            this.dataExcelFormat[i + 1][8] = this.dataFormat.filteredData[i].numbInStream;
            this.dataExcelFormat[i + 1][9] = this.dataFormat.filteredData[i].numbOutStream;
            this.dataExcelFormat[i + 1][10] = this.dataFormat.filteredData[i].forecastMigrationDate;
            this.dataExcelFormat[i + 1][11] = this.dataFormat.filteredData[i].migrationCompletedDate;
        }

        /* generate worksheet */
        const ws: XLSX.WorkSheet = XLSX.utils.aoa_to_sheet(this.dataExcelFormat);

        /* generate workbook and add the worksheet */
        const wb: XLSX.WorkBook = XLSX.utils.book_new();
        XLSX.utils.book_append_sheet(wb, ws, 'Sheet1');

        /* save to file */
        XLSX.writeFile(wb, this.fileName);
    }

```

Les traitements effectués derrière le bouton import seront détaillés dans une autre partie du compte rendu.

Pour terminer la partie sur l'interface graphique, il est également possible pour l'utilisateur connecté de visualiser l'ensemble des valeurs de l'ensemble des BS en cliquant sur le bouton display all BS. Cette fois-ci, l'utilisateur ne peut ni modifier, ni ajouter, ni supprimer de valeurs. Seulement la visualisation est possible :

Technology	BSS	Application code	Application instance	Device name	Current version middleware	Target version middleware	Environment	Number of incoming stream	Number of outgoing stream	Forecast Migration Date	Migration completed date
AD LEGACY	DCSI-BS-MV	CBB	RCBBGBLO		ADR	ADR	NONE	0	0	2019-04-25	2000-01-01
AD LEGACY	DCSI-BS-MV	FDM	RFDMGBLO		ADGROUP	ADR	NONE	0	0	2000-01-01	2000-01-01
AD LEGACY	DCSI-BS-MV	FDM	RFDMGBL2		EUW	DECOM	NONE	0	0	2019-11-20	2000-01-01
AD LEGACY	DCSI-BS-MV	MCC	RMCCEURO		EUW	DECOM	NONE	0	0	2018-12-18	2000-01-01
AD LEGACY	DCSI-	MKE	RMKE		ADR	ADR	NONE	0	0	2019-04-17	2000-01-01

3.3. Création de la base de données

Après la possibilité d'importer et d'exporter un fichier excel, il a fallu que je crée le schéma de la base de données car aucune base de données n'était à ce jour existante pour récupérer les données entrées dans l'ancienne application. Les données étaient jusqu'à présent récoltées manuellement par exportation d'un fichier excel directement par l'intermédiaire de l'application chaque mois. L'on m'a ainsi fourni le dernier fichier excel comprenant l'intégralité des dernières

données officielles récoltées via l'ancienne application. Ce fichier excel comprenait donc des colonnes tel que la technologie utilisée par le middleware, la version du middleware, l'environnement (production, développement) et d'autres colonnes. C'est sur ce fichier que j'ai dû me baser pour créer le schéma de la base de données.

Voici un exemple de quelques valeurs du fichier excel d'origine qui m'a été fourni :

1	Title	BSS	Appl	Application Inst	CurrentVersion	TargetVersion	MoveToTargetVersion-Date	Item Type	Path
53	ETL	DGSI/BS/SC	A2P	RA2PDP50	8.7	11.5	30/09/2019	Élément	sites/SWIFT/Lists/ObsManagement
54	ETL	DGSI/BS/SC	A2P	RA2PEURO	8.7	11.5	31/05/2019	Élément	sites/SWIFT/Lists/ObsManagement
55	ETL	DGSI/BS/SC	A2P	RA2PMNA0	8.7	11.5	31/05/2019	Élément	sites/SWIFT/Lists/ObsManagement
57	ETL	DGSI/BS/RD	AUG	RAUGEUR1	8.7	11.5	31/10/2018	Élément	sites/SWIFT/Lists/ObsManagement
58	ETL	DGSI/BS/ES	BDG	RBDGEURO	8.7	Decom	06/06/2018	Élément	sites/SWIFT/Lists/ObsManagement
59	ETL	DGSI/BS/PS	BSM	RBSMGBL1	8.7	Decom		Élément	sites/SWIFT/Lists/ObsManagement
60	ETL	DGSI/BS/PS	BSM	RBSMGBL2	8.7	11.5	31/07/2019	Élément	sites/SWIFT/Lists/ObsManagement
61	ETL	DGSI/BS/MV	CBB	RCCBGBL0	8.7	11.5	31/01/2019	Élément	sites/SWIFT/Lists/ObsManagement
62	ETL	DGSI/BS/ES	CHS	RCHSGBL3	8.7	Decom		Élément	sites/SWIFT/Lists/ObsManagement
63	ETL	DGSI/BS/SC	COE	RCOEEURO	8.7	Decom	30/11/2018	Élément	sites/SWIFT/Lists/ObsManagement
65	ETL	DGSI/BS/SC	DOO	RDOOGBL10	8.7	11.5	29/03/2019	Élément	sites/SWIFT/Lists/ObsManagement
67	ETL	DGSI/BS/ES	ESD	RESDNCA0	8.7	Decom		Élément	sites/SWIFT/Lists/ObsManagement
69	ETL	DGSI/BS/SC	FDC	RFDCEURO	8.7	Decom	30/11/2018	Élément	sites/SWIFT/Lists/ObsManagement
71	ETL	NA/SI/MNA/BS/SC	FGI	RFGIADS0	8.7	11.5		Élément	sites/SWIFT/Lists/ObsManagement
72	ETL	DGSI/BS/ES	FLH	RFLHEURO	8.7	8.7	19/12/2019	Élément	sites/SWIFT/Lists/ObsManagement
75	ETL	DGSI/BS/ES	FOP	RFOPEURO	8.7	11.5	14/11/2019	Élément	sites/SWIFT/Lists/ObsManagement
77	ETL	DGSI/BS/ES	FRG	RFRGGBL1	8.7	11.5	30/04/2019	Élément	sites/SWIFT/Lists/ObsManagement
78	ETL	NA/SI/MNA/BS/SC	FST	RFSTPLAO	8.7	11.5		Élément	sites/SWIFT/Lists/ObsManagement

Pour créer cette base de données, j'ai dû demander certaines informations à l'équipe de développeur notamment les clés primaires qui devaient composer les différentes tables. Une fois ces clés primaires obtenues, j'ai réalisé le schéma de la base de données en langage SQL sous le logiciel MySql Workbench. La base de données a dans un premier temps été créée sous phpmyadmin en local afin de pouvoir développer dans un environnement de développement, avant la mise en production.

Voici quelques exemples de scripts SQL que j'ai écrit qui représente chacun la création d'une table. La seconde capture d'écran représente la création d'une association entre la table target_version et la table technology afin de pouvoir récupérer toutes les versions ciblées en fonction de chaque technologie :

```

1  CREATE TABLE `ot_technology` (
2      `name` varchar(255) NOT NULL,
3      PRIMARY KEY (`name`)
4  ) ENGINE=InnoDB DEFAULT CHARSET=utf8
5
6
7  CREATE TABLE `ot_target_version_technology` (
8      `technology` varchar(255) CHARACTER SET utf8 COLLATE utf8_general_ci NOT NULL,
9      `version` varchar(255) NOT NULL,
10     `enable_version` int(10) DEFAULT '1',
11     PRIMARY KEY (`technology`, `version`),
12     KEY `fk_vers_targ` (`version`),
13     CONSTRAINT `fk_tech_targ` FOREIGN KEY (`technology`) REFERENCES `ot_technology` (`name`),
14     CONSTRAINT `fk_vers_targ` FOREIGN KEY (`version`) REFERENCES `ot_target_version` (`version`)
15 ) ENGINE=InnoDB DEFAULT CHARSET=utf8

```

```

1 CREATE TABLE `ot_datas` (
2     `technology` varchar(255) NOT NULL DEFAULT '',
3     `bss` varchar(255) NOT NULL DEFAULT '',
4     `app_code` varchar(100) NOT NULL,
5     `app_instance` varchar(255) NOT NULL,
6     `device_name` varchar(255) DEFAULT '',
7     `environment` varchar(100) NOT NULL DEFAULT '',
8     `current_version` varchar(255) NOT NULL,
9     `target_version` varchar(255) NOT NULL,
10    `forecast_migration_date` date NOT NULL,
11    `migration_completed_date` date DEFAULT '2000-01-01',
12    `number_incoming_flow` int(11) NOT NULL,
13    `number_outgoing_flow` int(11) NOT NULL,
14
15    PRIMARY KEY (`technology`,`bss`,`app_code`,`app_instance`,`environment`),
16    CONSTRAINT `fk_technology` FOREIGN KEY (`technology`) REFERENCES `ot_technology` (`name`),
17    CONSTRAINT `fk_bss` FOREIGN KEY (`bss`) REFERENCES `ot_bss` (`name`),
18    CONSTRAINT `fk_environment` FOREIGN KEY (`environment`) REFERENCES `ot_environment` (`name`),
19    CONSTRAINT `fk_technology` FOREIGN KEY (`technology`) REFERENCES `ot_technology` (`name`)
20 ) ENGINE=InnoDB DEFAULT CHARSET=utf8

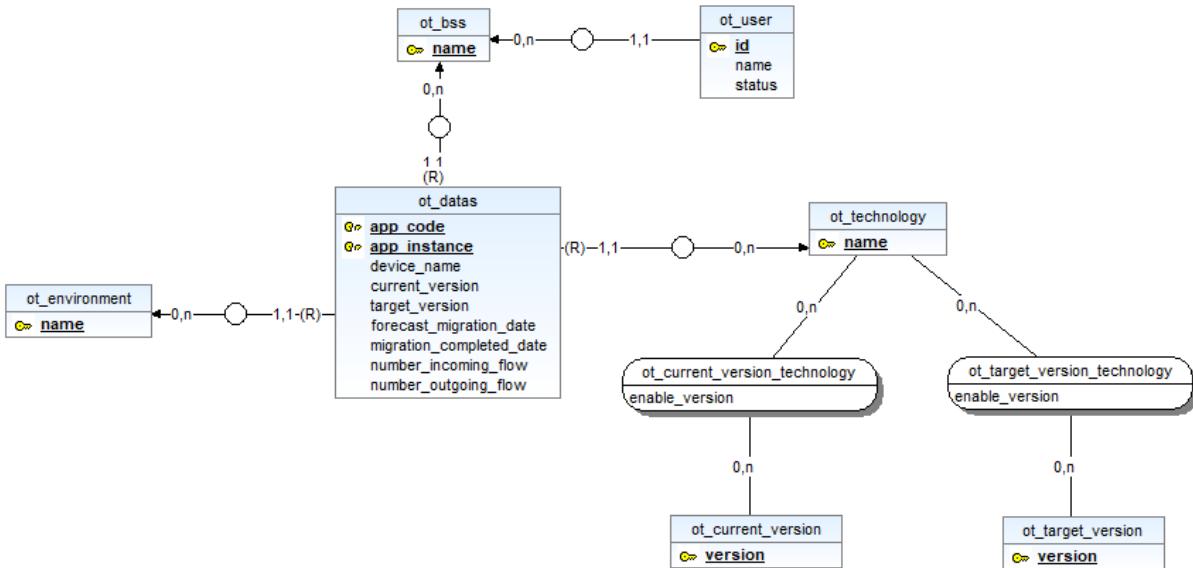
```

```

1 ● CREATE TABLE `ot_user` (
2     `id` int(10) NOT NULL AUTO_INCREMENT,
3     `name` varchar(255) NOT NULL,
4     `status` int(10) NOT NULL,
5     `bss` varchar(255) NOT NULL,
6     PRIMARY KEY (`id`),
7     CONSTRAINT `fk_bss_user` FOREIGN KEY (`bss`) REFERENCES ot_bss (`name`)
8 ) ENGINE=InnoDB AUTO_INCREMENT=7 DEFAULT CHARSET=utf8

```

Voici le modèle conceptuel de données représentant le schéma de la base de données que j'ai créé. On peut voir qu'un utilisateur (référencé dans la table « ot_user ») n'est rattaché qu'à une seule BSS par une clé étrangère. La table « ot_datas » qui contient les données des différents possède une clé primaire composée de 5 champs comme on peut le voir sur le schéma avec notamment 3 liens identifiants. Enfin, on peut voir qu'on retrouve deux associations permettant de référencer les versions actuelles et cibles disponible en fonction des technologies et permettant de savoir si ces versions sont actuellement disponible ou non (grâce au booléen « enable_version ») :



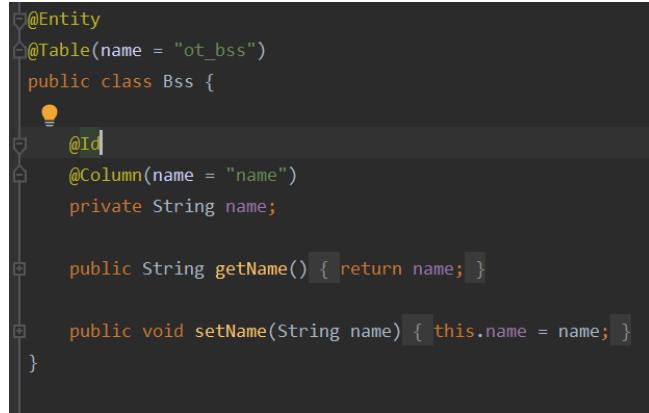
3.4. Création du backend

Une fois la base de données créées, il a fallu la remplir avec les valeurs existantes. Pour cela, j'ai tout d'abord créé mon API côté backend à l'aide du framework Spring boot. Spring boot permet de créer des API rapidement. Lorsque j'ai terminé de créer mon API, il a fallu que je crée la connexion entre mon API et la base de données. Pour cela, Spring boot propose des solutions qui facilite énormément la tâche aux développeurs grâce notamment à JPA (JPA signifiant Java Persistence API) qui est une spécification Java pour la gestion des données relationnelles dans les applications Java et qui permet d'accéder et de conserver des données entre les objets Java et la base de données relationnelle. JPA fournit également une API qui permet d'exécuter des traitements, des requêtes sur les objets Java qui sont traduits en requête SQL sur la base de données. Une fois la dépendance JPA ajouté dans le projet, j'ai ainsi créé toutes les classes Java reflétant les entités de la base de données. A chaque modèle de classe, il a fallu que je crée un repository associé. Un repository lié à une classe permet d'effectuer des opérations sur la table correspondante dans la base de données (notamment les opérations CRUD : Create, Read, Update et Delete). Un repository est une interface devant implémenter l'interface Crudrepository afin d'effectuer les opérations CRUD.

Voici ci-dessous plusieurs captures d'écran des classes Java qui représentent des modèles de table de la base de données.

Quelques explications sur ces captures d'écran tout de même. La capture d'écran représente tout simplement la table “ot_bss”. JPA sait qu'il doit lier cette classe à la table “ot_bss” dans la base de données tout d'abord grâce à l'annotation @Entity qui déclare cette classe comme un modèle correspondant à une table dans la base de données. Ensuite, l'annotation @Table

permet de définir le nom dans la base de données à laquelle cette classe doit correspondre. Ensuite, avec l'annotation `@Id`, on sait que la propriété "name" est une clé primaire dans la base de données. Enfin, avec l'annotation `@Column`, on indique à JPA qu'il doit faire correspondre la propriété qui contient l'annotation `@Column` à colonne "name" dans la base de données :



```

@Entity
@Table(name = "ot_bss")
public class Bss {
    @Id
    @Column(name = "name")
    private String name;

    public String getName() { return name; }

    public void setName(String name) { this.name = name; }
}

```

La seconde capture d'écran ci-dessous contient un cas intéressant. En effet, la table "ot_data" contient une clé composée en tant que clé primaire. En effet, la clé primaire de la table "ot_data" est (technology, bss, app_code, app_instance, environment). Pour indiquer à JPA que cette classe contient une clé primaire composée, on ajoute l'annotation `@IdClass` avec en paramètre le modèle de classe de la clé primaire :



```

@Entity
@Table(name = "ot_data")
@IdClass(DataId.class)
public class Data implements Serializable {

    @Id
    @Column(name = "technology")
    private String technology;

    @Id
    @Column(name = "bss")
    private String bss;

    @Id
    @Column(name = "app_code")
    private String appCode;

    @Id
    @Column(name = "app_instance")
    private String appInstance;
}

```

Et voici ci-dessous le modèle de classe de la clé primaire :

```

public class DataId implements Serializable {

    private String technology;

    private String bss;

    private String appCode;

    private String appInstance;

    private String environment;
}

```

Enfin, voici des extraits de code des repository créés et qui permettent d'effectuer des requêtes SQL et donc d'interagir avec la base de données. On peut voir que les deux interfaces héritent de la classe CrudRepository qui permet de pouvoir se servir de méthodes de bases permettant d'interagir avec la base de données, notamment les méthodes save, update, delete. Ensuite, on peut voir plusieurs méthodes que j'ai écrit avec des requêtes SQL qui leurs sont associées et qui sont exécutées à l'appel des méthodes grâce à l'annotation @Query :

```

@Repository
public interface CurrentVersionRepository extends CrudRepository<CurrentVersion, String> {

    @Query("SELECT c.version FROM CurrentVersion c ORDER BY c.version ASC")
    Iterable<CurrentVersion> getCurrentVersion();
}

```

```

@Repository
public interface CurrentVersionTechnologyRepository extends CrudRepository<CurrentVersionTechnology, String> {

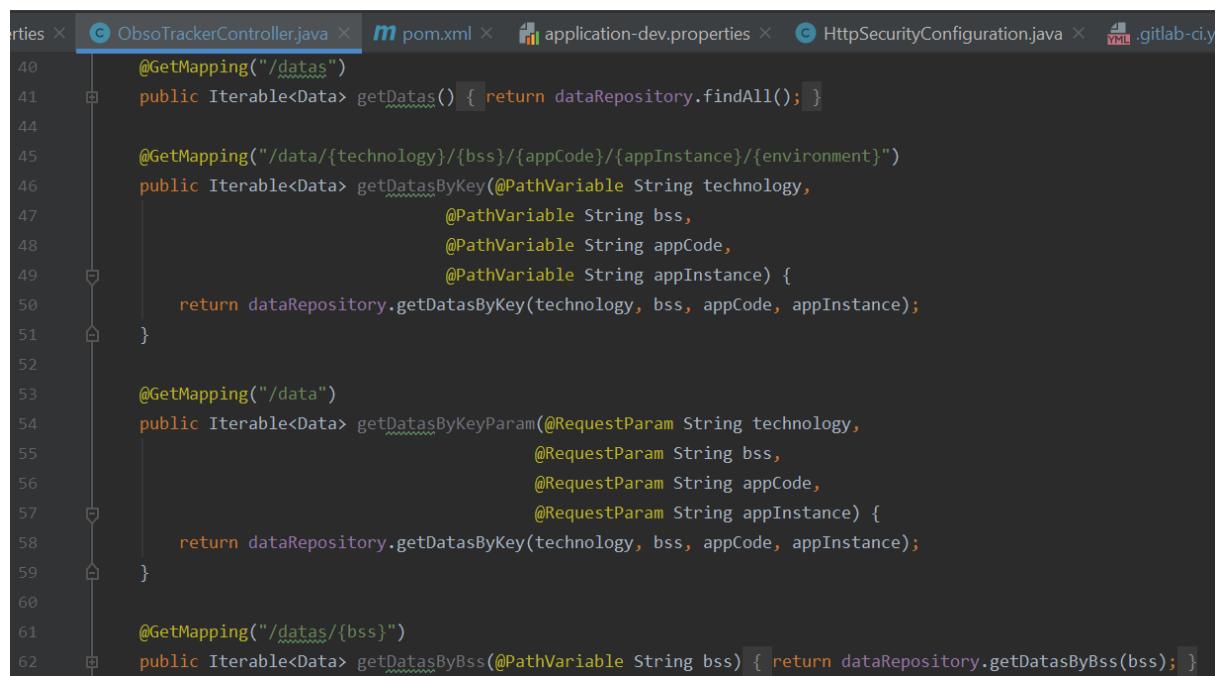
    @Query("SELECT c.version FROM CurrentVersionTechnology c WHERE c.technology = :technology order by version ASC")
    Iterable<CurrentVersionTechnology> getVersionOfTechno(@Param("technology") String technology);

    @Query("SELECT c.technology, c.version FROM CurrentVersionTechnology c order by version ASC")
    Iterable<CurrentVersionTechnology> getAllByAsc();
}

```

Une fois la connexion effectuée entre mon API et la base de données, j'ai créé les URL de mon API qu'il est possible d'appeler avec les verbs correspondant (POST, PUT, DELETE, GET) ainsi que les méthodes correspondantes à ces appels et les résultats à retourner en fonction des appels effectués sur le backend.

Voici quelques exemples des URL exposées par l'API qui est possible d'aller consommer. Sur la première capture d'écran on peut voir plusieurs méthodes qui retournent des valeurs. L'annotation `@GetMapping` permet d'indiquer dans un premier temps que cette méthode ne peut être appelée que lorsque la requête HTTP est un GET et deuxièmement de faire correspondre une URL à cette méthode. Les mots placés entre les accolades correspondent aux paramètres fournis par la requête HTTP dans l'URL qui sont ici récupérés grâce à l'annotation `@PathVariable` qui se charge d'aller récupérer les paramètres de l'URL de la requête. De la même façon, `@RequestParam` récupère les variables dans l'URL mais cette fois-ci les paramètres de l'URL sont fournis sous formes de clé/valeur. La deuxième capture d'écran montre les annotations placées sur la classe qui contient toutes les URL exposées par l'API. L'annotation `@RestController` permet de déclarer cette classe en tant que "controller Restful", c'est-à-dire en tant que contrôleur agissant de façon modulaire, en découplant au maximum les tâches à effectuer et surtout en utilisant le langage HTTP (à savoir les méthodes GET, POST, PUT, DELETE pour les principales). L'annotation `@RequestMapping` quant à elle définit la base de l'URL de l'API. Ainsi on pourrait par exemple appeler cette API en effectuant une requête sur l'adresse `monAdresseUrl/api/v1.0/...` :



```

Properties x ObsoTrackerController.java x pom.xml x application-dev.properties x HttpSecurityConfiguration.java x .gitlab-ci.yml
40   @GetMapping("/datas")
41   public Iterable<Data> getDatas() { return dataRepository.findAll(); }
42
43
44   @GetMapping("/data/{technology}/{bss}/{appCode}/{appInstance}/{environment}")
45   public Iterable<Data> getDatasByKey(@PathVariable String technology,
46                                     @PathVariable String bss,
47                                     @PathVariable String appCode,
48                                     @PathVariable String appInstance) {
49     return dataRepository.getDataByKey(technology, bss, appCode, appInstance);
50   }
51
52
53   @GetMapping("/data")
54   public Iterable<Data> getDatasByKeyParam(@RequestParam String technology,
55                                         @RequestParam String bss,
56                                         @RequestParam String appCode,
57                                         @RequestParam String appInstance) {
58     return dataRepository.getDataByKey(technology, bss, appCode, appInstance);
59   }
60
61   @GetMapping("/datas/{bss}")
62   public Iterable<Data> getDatasByBss(@PathVariable String bss) { return dataRepository.getDatasByBss(bss); }

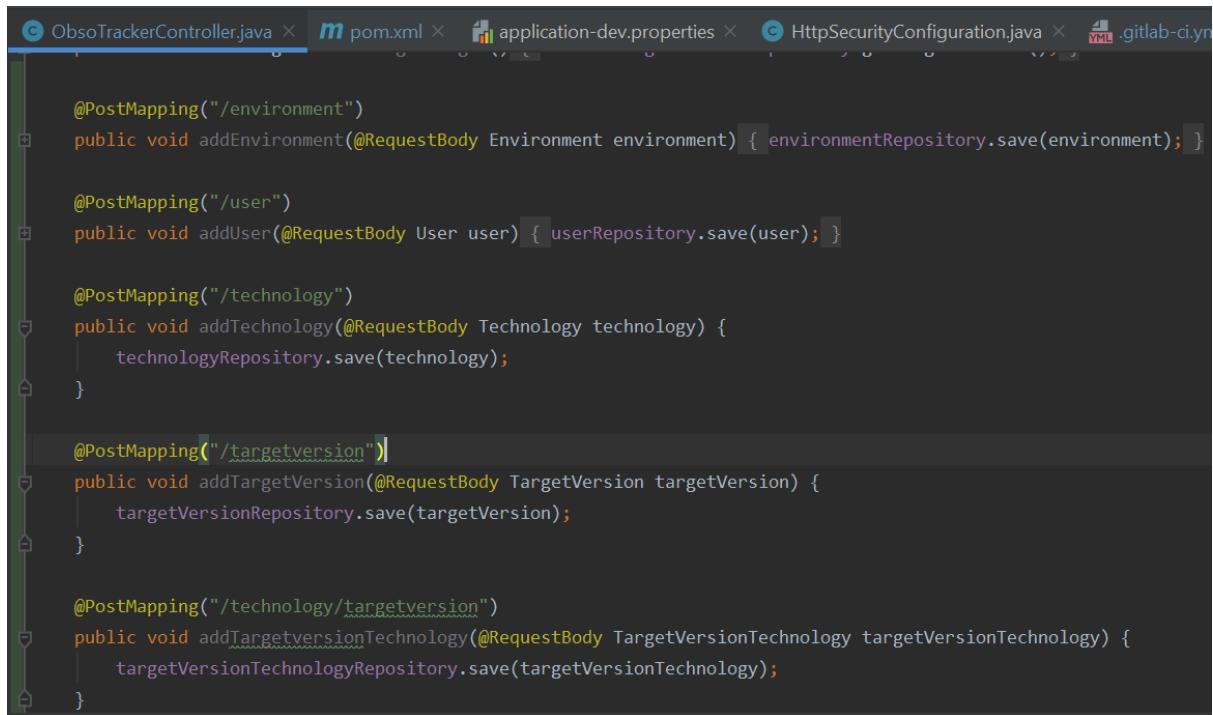
```

```

@RestController
@RequestMapping(path = "api/v1.0")
public class ObsoTrackerController {

```

Voici cette fois-ci une capture d'écran qui permet d'enregistrer des valeurs grâce au verbe HTTP POST et à l'annotation `@PostMapping`. On peut voir que grâce à JPA, l'enregistrement d'une nouvelle valeur dans la base de données est très simple. On peut voir dans la première méthode par exemple que l'on s'occupe tout simplement d'appeler le repository correspondant à la table `environment` et d'appeler sur ce repository la méthode `save` qui prend en paramètre la valeur à ajouter dans la base de données. Derrière la méthode `save`, une requête SQL d'insertion est effectuée. JPA gère tout pour nous :



```
ObsoTrackerController.java X pom.xml X application-dev.properties X HttpSecurityConfiguration.java X .gitlab-ci.yml

@PostMapping("/environment")
public void addEnvironment(@RequestBody Environment environment) { environmentRepository.save(environment); }

@PostMapping("/user")
public void addUser(@RequestBody User user) { userRepository.save(user); }

@PostMapping("/technology")
public void addTechnology(@RequestBody Technology technology) {
    technologyRepository.save(technology);
}

@PostMapping("/targetversion")
public void addTargetVersion(@RequestBody TargetVersion targetVersion) {
    targetVersionRepository.save(targetVersion);
}

@PostMapping("/technology/targetversion")
public void addTargetversionTechnology(@RequestBody TargetVersionTechnology targetVersionTechnology) {
    targetVersionTechnologyRepository.save(targetVersionTechnology);
}
```

Enfin, un dernier exemple illustrant la suppression de valeur dans la base de données grâce à l'annotation `@DeleteMapping`:

```
112     @DeleteMapping("/data/{technology}/{bss}/{appCode}/{appInstance}/{environment}")
113     public void deleteData(
114         @PathVariable String technology,
115         @PathVariable String bss,
116         @PathVariable String appCode,
117         @PathVariable String appInstance,
118         @PathVariable String environment
119     ) {
120         dataRepository.deleteData(
121             technology,
122             bss,
123             appCode,
124             appInstance,
125             environment);
126     }
```

J'ai voulu tester ensuite cette connexion et les relations créées entre mes classes Java et les tables de la base de données d'une part, et d'autre part si les appels aux URL exposés par l'API retournaient les bons résultats. Je me suis servi pour cela de l'application Postman qui permet d'effectuer rapidement et simplement des appels à une API en utilisant les verbs GET, PUT, DELETE, POST et d'obtenir le résultat retourné par l'API.

Voici dans la capture d'écran ci-dessous un exemple d'appel à mon API grâce à l'application Postman. En haut, j'ai sélectionné la méthode POST, et j'effectue cette méthode POST sur l'URL fournie. Dans le corps de la requête, j'envoie des données au format json avec notamment un login et un mot de passe. Dans la fenêtre du bas, j'ai la réponse de l'API qui m'indique que l'authentification a échoué étant donné que je n'ai pas fourni un bon couple login/mot de passe :

The screenshot shows the Postman interface. At the top, a POST request is made to `http://localhost:8080/api/v1.0/authenticate`. The request body is set to `JSON` and contains the following JSON:

```

1 {
2   "login": "TEST",
3   "password": "TEST",
4   "rememberMe": false
5 }

```

The response status is `401 Unauthorized`, with a timestamp of `1584348440819`. The response body is:

```

1 [
2   {
3     "message": "Authentication error, token expired or invalid",
4     "code": 0,
5     "key": "authentication.error",
6     "timestamp": 1584348440819
7   ]

```

Un autre exemple ici. J'utilise la méthode GET pour obtenir les versions actuelles de la technologie “CFT” que j'ai fournie en paramètre. Dans la fenêtre du bas, j'obtiens la réponse de l'API au format JSON :

The screenshot shows a GET request to `http://localhost:8080/api/v1.0/technologie/target/CFT`. The response body is:

```

1 [
2   {
3     "technologie": "CFT",
4     "version": "9.5"
5   }
6 ]

```

Ensuite, pour effectuer des tests, étant donné que ma base de données était à ce moment là encore vide, j'ai créé quelques jeux d'enregistrements permettant les tests. Une fois mes tests réalisés qui se sont avérés concluant, il ne me restait plus qu'à remplir la base de données avec les valeurs existantes dans le fichier excel. Pour cela, j'ai trouvé comme solution d'importer le fichier excel dans ma partie Frontend, d'effectuer un appel à mon API et qui, elle, allait se charger de remplir la base de données, ce qui a fonctionné comme je le souhaitais.

3.5. Contrôle des données importées

Est venu ensuite l'implémentation du contrôle des valeurs importées par l'utilisateur afin de vérifier s'il importe une technologie référencée par Michelin, une version correcte, un environnement correct et encore d'autres valeurs. Pour cela, il a fallu réaliser toutes une batterie de contrôles permettant de vérifier la validité des données. Pour faciliter l'importation aux utilisateurs, j'ai fait le choix d'ouvrir une petite fenêtre d'information indiquant le nombre de lignes correctement importées et le nombre de lignes n'ayant pas été importées avec le numéro de la ligne et la ou les colonnes ayant générées l'échec de l'importation de la ligne. Ainsi, l'utilisateur n'avait qu'à retourner dans son fichier excel et corriger uniquement les lignes comportant des erreurs étant donné qu'il avait le numéro des lignes affiché par la fenêtre ouverte à l'importation.

Cette validation des données a été effectuée après l'implémentation du backend et de la base de données étant donné que le contrôle se fait en fonction des valeurs présentes dans les tables de la base de données.

Voici tout d'abord quelques jeux d'enregistrement que j'avais écrit pour tester le bon fonctionnement du contrôle des valeurs importées par les utilisateurs. La première ligne représente les différentes colonnes. La ligne 2 contient comme erreur la technologie "TEST". La ligne 3 est une ligne qui ne contient pas d'erreur. La ligne 4 contient comme erreur la BSS qui n'est pas, dans le cas de mon test, la BSS de l'utilisateur connecté (un utilisateur ne peut pas importer les valeurs d'une BSS qui n'est pas la sienne). La ligne 5 contient comme erreur la version 20 qui n'existe pas pour la technologie WMQ. Enfin, la ligne 6 contient comme erreur l'environnement "TEST" qui n'est pas un environnement valide dans notre cas.

	A	B	C	D	E	F	G	H	I	J	K	L
1	Technology	BSS	Application Code	Application Instance	Device name	CurrentVersion	TargetVersion	Environment	Number incoming flow	Number outgoing flow	Forecast migration date	Migration completed date
2	TEST	DCSI/BS/RD	TEST	TEST		7	8	Dev	10	10	01/01/2000	01/01/2000
3	WMQ	DCSI/BS/RD	TEST	TEST		7	8	Dev	10	10	01/01/2000	01/01/2000
4	WMQ	DCSI/BS/SC	TEST	TEST		7	8	Dev	10	10	01/01/2000	01/01/2000
5	WMQ	DCSI/BS/RD	TEST	TEST		20	8	Dev	10	10	01/01/2000	01/01/2000
6	WMQ	DCSI/BS/RD	TEST	TEST		7	8	TEST	10	10	01/01/2000	01/01/2000
7												

Voyons ce qu'il se passe lorsqu'on importe ce fichier excel. Comme prévu, je me suis occupé d'afficher le nombre de lignes correctement importées et le nombre de lignes contenant des erreurs avec ensuite pour chaque lignes contenant des erreurs, les détails qui ont provoqué des erreurs :

BSS INVENTORY

ROWS WITH ERROR:

- Row 2 in the excel file: TEST technology is not available in the list. Current version 7 is not available for the TEST technology. Target version 8 is not available for the TEST technology.
- Row 4 in the excel file: DCSI/BS/SC isn't you're BSS.
- Row 5 in the excel file: Current version 20 is not available for the WMQ technology. Target version 8 is not available for the WMQ technology.
- Row 6 in the excel file: Environment TEST doesn't exist.

Technology	Version	Type	Status	Date	Date					
AD LEGACY	DCSI-BS-RD	BDS	IBDSCAR0	EUW	ADR	NONE	0	0	2000-01-01	2000-01-01
AD LEGACY	DCSI-BS-RD	BDS	RBDS	EUW	ADR	NONE	0	0	2000-01-01	2000-01-01

Ci-dessous, on peut notamment voir comment le tableau des erreurs est valorisé au moment de l'import du fichier excel. Chaque ligne du fichier excel est lue. Si une ligne contient une erreur, elle est ajoutée au tableau des erreurs :

```

407      this.errorTab.push('Row ' + (i + 1) + ' in the excel file: ');
408      const indexToChange = this.errorTab.length - 1;
409      if (!isBssOk) {
410        this.errorTab[indexToChange] = this.errorTab[indexToChange] +
411          this.dataExcelFormat[i][1] + ' isn\'t you\'re BSS. ';
412      }
413      if (!isTechnoOk) {
414        this.errorTab[indexToChange] = this.errorTab[indexToChange] +
415          aData.technology + ' technology is not available in the list.';
416      }
417      if (!isCurrVers) {
418        this.errorTab[indexToChange] = this.errorTab[indexToChange] +
419          ' Current version ' + aData.currVersion + ' is not available for the ' +
420          aData.technology + ' technology. ';
421      }
422      if (!isTargVers) {
423        this.errorTab[indexToChange] = this.errorTab[indexToChange] +
424          ' Target version ' + aData.targVersion + ' is not available for the ' +
425          aData.technology + ' technology. ';
426      }
427      if (!isEnvOk) {
428        this.errorTab[indexToChange] = this.errorTab[indexToChange] +
429          ' Environment ' + aData.environment + ' doesn\'t exist. ';
430    }
  
```

Ensuite, comme on peut le voir, on s'occupe d'afficher chaque case du tableau contenant les erreurs :

```

1  <div>
2    <p class="imported-row">Number of rows imported: {{data.nbImportedLines}}</p>
3  </div>
4  <div>
5    <p class="error-row">Number of rows that contains errors: {{data.nbErrors}}</p>
6  </div>
7  <h2>ROWS WITH ERROR: </h2>
8  <div>
9    <p class="error-row" *ngFor="let error of data.errors">
10      {{error}}
11    </p>
12  </div>

```

Enfin, on peut voir quelques-uns des nombreux tests effectués sur les valeurs du fichier excel importé :

```

362      this.technology.forEach(element => {
363        let techno = aData.technology;
364        techno = techno.replace('/', '');
365        techno = techno.replace('-', '');
366        techno = techno.replace(' ', '');
367        let elementChange = element;
368        elementChange = element.replace('/', '');
369        elementChange = elementChange.replace('-', '');
370        elementChange = elementChange.replace(' ', '');
371        if (elementChange.includes(techno) && !isTargVers) {
372          isTechnoOk = true;
373          aData.technology = element;
374          this.allCurVersAndTechno.forEach(ele => {
375            aData.curVersion = aData.curVersion.toString().replace(',', '.');
376            if (aData.technology === ele[0] && ele[1].includes(aData.curVersion.toString()) && !isCurrVers) {
377              aData.curVersion = ele[1];
378              isCurrVers = true;
379              this.allTargVersAndTechno.forEach(elem => {
380                aData.targVersion = aData.targVersion.toString().replace(',', '.');
381                if (aData.technology === elem[0] && elem[1].includes(aData.targVersion.toString()) && !isTargVers) {
382                  aData.targVersion = elem[1];
383                  isTargVers = true;
384                }
385              });
386            });
387          });
388        });

```

Pour faciliter la correction des lignes contenant des erreurs, j'ai donné la possibilité aux utilisateurs de pouvoir visualiser à tout moment la fenêtre affichant les erreurs levées par l'application en cliquant sur le bouton "display error" avec quelques extraits de code dans la seconde et troisième capture d'écran :

YOUR BSS INVENTORY												
Display all BS Display error Import Export												
Technology	BSS	Application code	Application instance	Device name	Current version middleware	Target version middleware	Environment	Number of incoming stream	Number of outgoing stream	Forecast Migration Date	Migration completed date	Update
AD LEGACY	DCSI-BS-RD	AAA	RAAA		EUW	ADR	NONE	0	0	2000-01-01	2000-01-01	<input type="checkbox"/>
AD LEGACY	DCSI-BS-RD	ASQ	CASQ		EUW	ADR	NONE	0	0	2000-01-01	2000-01-01	<input type="checkbox"/>
AD LEGACY	DCSI-BS-RD	BDS	IBDS		EUW	ADR	NONE	0	0	2000-01-01	2000-01-01	<input type="checkbox"/>
AD LEGACY	DCSI-BS-RD	BDS	IBDSCAR0		EUW	ADR	NONE	0	0	2000-01-01	2000-01-01	<input type="checkbox"/>
AD LEGACY	DCSI-BS-RD	BDS	RBDS		EUW	ADR	NONE	0	0	2000-01-01	2000-01-01	<input type="checkbox"/>

```

105 |     <button *ngIf="containError && !isAllInventory && !readOnly" mat-raised-button color="primary"
106 |       class="display-error" (click)="displayError()">
107 |       Display error
108 |     


```

739 | displayError() {
740 | const dialogConfig = this.dialog.open(ErrorImportComponent, {
741 | data: {
742 | errors: this.errorTab,
743 | nbErrors: this.errorTab.length,
744 | nbImportedLines: (this.dataExcelFormat.length - 1 - this.errorTab.length)
745 | },
746 | width: '75%',
747 | height: '90%',
748 | maxWidth: '90%',
749 | maxHeight: '90%',
750 | autoFocus: true,
751 | });
752 | }

```


```

3.6. Gestion de l'authentification

3.6.1. Page de connexion

A partir de là, je pouvais alors travailler avec les tables remplies des véritables données.

La tâche suivante à développer dans l'application a été de créer une page de connexion et les traitements à effectuer sur cette connexion. Dans l'entreprise Michelin, chaque employé possède un identifiant et un mot de passe enregistré dans le LDAP de l'entreprise. A savoir que chaque employé est rattaché à une BS. Sachant que l'application doit permettre aux leaders de chaque BS de pouvoir ajouter ou mettre à jour les versions des middlewares qu'il possède et uniquement ceux qu'il possède et non pas ceux des autres BS, il fallait qu'à la connexion d'un leader sur l'application, uniquement les données qui sont rattachées à sa BS soient générées et non pas les données des autres BS. Ainsi, dans la base de données une table "User" a été créée, comprenant entre autres un champ "bs" qui permet d'associer à chaque leader une BS d'appartenance. Dans un premier temps, pour effectuer des tests, j'ai créé un utilisateur test rattaché à un service. Lors de la connexion, un appel au back est ainsi effectué et vérifie si le couple identifiant/mot de passe est correct. Si la connexion est réussie, uniquement les données correspondantes à son service sont récupérées dans la base de données par l'intermédiaire de l'appel à l'API. Une variable dans la partie frontend est alors valorisée et le tableau affiche ainsi toutes les valeurs correspondantes avec possibilité de modifier ou de supprimer graphiquement les valeurs ce qui modifie ou supprime directement les valeurs dans la base de données par l'intermédiaire, comme d'habitude, d'un appel au backend. Également, cet utilisateur a la possibilité d'importer directement des valeurs dans l'application.

Une autre étape dans la connexion a été d'implémenter le fait qu'un utilisateur Michelin n'étant pas leader d'une BS pouvait tout de même avoir la possibilité de visualiser les valeurs de toutes les BS, mais sans possibilité de modification, d'ajout ou de suppression. Ainsi, dans la table "User" de la base de données, uniquement les deux ou trois leaders de chaque BS sont présents. Lors de la connexion, on vérifie ainsi si l'utilisateur qui se connecte est référencé dans la base : si c'est le cas, alors il a la possibilité de modifier, supprimer ou ajouter des enregistrements, sinon l'utilisateur ne peut que visualiser les données de toutes les BS référencées.

Voici l'exemple de l'interface d'un utilisateur qui ne possède que des droits en lecture :

BSS INVENTORY											OBSO TRACKER		LOG OUT
Technology			Current version		Target version		BSS		Application code			x	
Application instance			x Environment		Forecast date of the migra...		Migration completed date		Reset			x	
ALL BSS INVENTORY													
Technology	BSS	Application code	Application instance	Device name	Current version middleware	Target version middleware	Environment	Number of incoming stream	Number of outgoing stream	Forecast Migration Date	Migration completed date		
AD LEGACY	DCSI-BS-MV	CBB	RCBBGBL0		ADR	ADR	NONE	0	0	2019-04-25	2000-01-01		
AD LEGACY	DCSI-BS-MV	FDM	RFDMGBL0	ADGROUP	ADR	NONE	0	0	0	2000-01-01	2000-01-01		
AD LEGACY	DCSI-BS-MV	FDM	RFDMGBL2	EUW	DECOM	NONE	0	0	0	2019-11-20	2000-01-01		
AD LEGACY	DCSI-BS-MV	MCC	RMCCEURO	EUW	DECOM	NONE	0	0	0	2018-12-18	2000-01-01		
AD LEGACY	DCSI-BS-RD	MKE	RMKE		ADR	ADR	NONE	0	0	2019-04-17	2000-01-01		

Et un autre exemple montrant l'interface d'un utilisateur possédant les droits d'écritures et de suppressions :

BSS INVENTORY											OBSO TRACKER		LOG OUT
Technology			Current version		Target version		BSS		Application code			x	
Application instance			x Environment		Forecast date of the migra...		Migration completed date		Reset			x	
YOUR BSS INVENTORY													
Technology	BSS	Application code	Application instance	Device name	Current version middleware	Target version middleware	Environment	Number of incoming stream	Number of outgoing stream	Forecast Migration Date	Migration completed date	Update	trash
AD LEGACY	DCSI-BS-RD	AAA	RAAA		EUW	ADR	NONE	0	0	2000-01-01	2000-01-01	<input checked="" type="checkbox"/>	<input type="checkbox"/>
AD LEGACY	DCSI-BS-RD	ASQ	CASQ		EUW	ADR	NONE	0	0	2000-01-01	2000-01-01	<input checked="" type="checkbox"/>	<input type="checkbox"/>
AD LEGACY	DCSI-BS-RD	BDS	IBDS		EUW	ADR	NONE	0	0	2000-01-01	2000-01-01	<input checked="" type="checkbox"/>	<input type="checkbox"/>
AD LEGACY	DCSI-BS-RD	BDS	IBDSCAR0		EUW	ADR	NONE	0	0	2000-01-01	2000-01-01	<input checked="" type="checkbox"/>	<input type="checkbox"/>
AD LEGACY	DCSI-BS-RD	BDS	RBDS		EUW	ADR	NONE	0	0	2000-01-01	2000-01-01	<input checked="" type="checkbox"/>	<input type="checkbox"/>

Voici la page de connexion :

A screenshot of a sign-in form. At the top is a green button labeled "SIGN IN". Below it is a white input field labeled "Michelin ID" with the placeholder "Michelin ID". A red border surrounds this field, and below it, a red error message says "This field is required". Below the input field is a white input field labeled "Password" with the placeholder "Test". At the bottom of the form is a grey "Sign in" button and a "Remember Me" checkbox.

Et voici la page de connexion lorsqu'il y a une erreur de connexion :

A screenshot of a sign-in form similar to the one above, but with an error message. The "Michelin ID" field contains "Test" and the "Password" field contains "....". The "Sign in" button is green. Below the "Sign in" button is a pink rectangular box containing the red text "Invalid credentials". The "Remember Me" checkbox is present at the bottom.

Ci-dessous, un extrait de code du code HTML permettant d'afficher cette page de connexion :

```

1 <mat-toolbar>
2   <span class="fill-remaining-space"></span>
3   <div>SIGN IN</div>
4   <span class="fill-remaining-space"></span>
5 </mat-toolbar>
6
7 <form [FormGroup]="authForm" (ngSubmit)="onSignIn()" autocomplete="off">
8   <div class="sign-form mat-elevation-z8">
9     <mat-form-field appearance="outline">
10       <mat-label>Michelin ID</mat-label>
11       <input type="text" matInput formControlName="login" placeholder="Michelin ID">
12       <mat-error>This field is required</mat-error>
13     </mat-form-field>
14     <mat-form-field appearance="outline">
15       <mat-label>Password</mat-label>
16       <input matInput formControlName="password" placeholder="password" type="password">
17       <mat-error>This field is required</mat-error>
18     </mat-form-field>
19     <button class="button-row" type="submit" [disabled]="authForm.invalid" mat-raised-button color="primary">Sign
20     <div *ngIf="isAuthWrong" class="invalid-auth">Invalid credentials</div>
21     <mat-checkbox color="primary" [(ngModel)]="rememberMe" formControlName="rememberMe" name="rememberMe">Remembe
22   </div>
23 </form>

```

3.6.2. Authentification

Une fois que j'avais terminé d'implémenter les fonctionnalités permettant soit de visualiser uniquement les valeurs soit de pouvoir modifier les données en fonction de l'utilisateur connecté qui n'était à ce stade là que des utilisateurs tests, il a fallu que j'implémente, côté backend, l'authentification en allant interroger l'annuaire LDAP de Michelin.

Pour cela, Spring boot propose des bibliothèques et des fonctionnalités permettant de gérer l'interrogation d'un annuaire LDAP. Après avoir ajouté les dépendances nécessaires dans le backend, je me suis occupé d'implémenter les fonctionnalités permettant de récupérer un utilisateur dans le LDAP.

Voici ci-dessous une capture d'écran de la méthode, correspondant à une URL exposée par l'API, qui s'occupe de gérer l'authentification. La plupart des méthodes sont des méthodes rendues disponibles par des bibliothèques du framework Spring Boot. Le code qui s'exécute en arrière-plan est relativement complexe pour pouvoir rentrer dans le détail. Cependant, la logique est la suivante : on récupère le login et le mot de passe fourni dans la requête HTTP en POST. Une fois le couple récupéré, on s'occupe de vérifier si le couple login/mot de passe est correct et est référencé dans l'annuaire LDAP. Si tel est le cas, on crée un token qu'on ajoute dans la réponse HTTP et qu'on renvoie au frontend :

```

50     * Method used to authenticate an user
51     *
52     * @param credentialsDto user object containing user/password
53     * @return the token
54     */
55     @ApiOperation(value = "Authenticate the user and create a new Jwt Token", response = JwtTokenDto.class)
56     @ApiResponses(value = { @ApiResponse(code = 401, response = ApiErrorDto.class, message = "Bad credential"),
57                     @ApiResponse(code = 201, response = JwtTokenDto.class, message = "Token created") })
58     @ResponseStatus(HttpStatus.CREATED)
59     @PostMapping("/authenticate")
60     public ResponseEntity<JwtTokenDto> authorize(
61         @ApiParam(value = "Object containing login and password", required = true) @Valid @RequestBody CredentialsDto credentialsDto)
62         // initialise authentication token with passed login and password
63         UsernamePasswordAuthenticationToken authenticationToken = new UsernamePasswordAuthenticationToken(
64             credentialsDto.getLogin(), credentialsDto.getPassword());
65         // authenticate against ldap
66         Authentication authentication = this.authenticationManager.authenticate(authenticationToken);
67         // set authenticated user into SecurityContextHolder
68         SecurityContextHolder.getContext().setAuthentication(authentication);
69         // generate jwt token
70         String jwt = tokenService.createToken(authentication, credentialsDto.isRememberMe());
71         // construct response
72         HttpHeaders httpHeaders = new HttpHeaders();
73         httpHeaders.add(JWTConfigurer.X_AUTH_TOKEN, jwt);
74         return new ResponseEntity<>(new JwtTokenDto(jwt, JWTConfigurer.BEARER.trim()), httpHeaders,
75             HttpStatus.CREATED);
76     }

```

Voici des extraits de code des classes et méthodes qui s'exécutent pour vérifier que l'authentification est réussie, de vérifier si l'utilisateur qui tente de se connecter existe bien dans le LDAP ou retourner l'utilisateur authentifié. Le code qui suit n'a pas été écrit de A à Z par mes soins. En effet, je me suis inspiré de modes opératoires présents sur le Wiki du GitLab du service Michelin et j'ai adapté pour mon application le contenu des classes. Voici des extraits :

```

53     @Override
54     @Override
55     public Authentication attemptAuthentication(
56         HttpServletRequest req, HttpServletResponse res) throws IOException {
57
58         //Instantiate LoginFilterUserCredentials object from request. json in form {"username":"xxx","password":"yyy"}
59         LoginFilterUserCredentials creds = new ObjectMapper()
60             .readValue(req.getInputStream(), LoginFilterUserCredentials.class);
61
62         //Set LoginFilterUserCredentials into current thread
63         loginFilterUserCredentialsThreadLocal.set(creds);
64
65         // Iterate through authentication providers in order to authenticate/authorize.
66         return getAuthenticationManager().authenticate(
67             new UsernamePasswordAuthenticationToken(
68                 creds.getUsername(),
69                 creds.getPassword(),
70                 Collections.emptyList()
71             );

```

```

90  @Bean
91  public UserDetailsService userDetailsService() {
92      return new LdapUserDetailsMapper() {
93
94          @Override
95          public UserDetails mapUserFromContext(DirContextOperations ctx, String username,
96                                              Collection< GrantedAuthority> authorities) {
97
98              // Using ConnectedUser object even if it doesn't correspond to ldap structure in
99              // ldif file
100             ConnectedUser connectedUser = new ConnectedUser(username, authorities);
101
102             // last name in ldif file found at sn
103             connectedUser.setLastName(ctx.getStringAttribute("sn"));
104             // first name in ldif file found at uid
105             connectedUser.setFirstName(ctx.getStringAttribute("givenName"));
106             // no mail address in ldif
107             connectedUser.setEmail(ctx.getStringAttribute("mail"));
108
109             return connectedUser;
110         }
111     };
112 }

```



```

26 /**
27 * The Class LdapServerAuthenticationConfiguration.
28 */
29 @Configuration
30 @EnableWebSecurity
31 @Order(4)
32 @ConditionalOnProperty(name = "application.authentication.provider", havingValue = "ldap", matchIfMissing = true)
33 public class LdapServerAuthenticationConfiguration extends WebSecurityConfigurerAdapter {
34
35     private final ApplicationProperties applicationProperties;
36
37     private final LdapProperties ldapProperties;
38
39     private final UserDetailsServiceLdapAuthoritiesPopulator userDetailsServiceLdapAuthoritiesPopulator;
40
41     public LdapServerAuthenticationConfiguration(ApplicationProperties applicationProperties,
42                                                 LdapProperties ldapProperties,
43                                                 UserDetailsServiceLdapAuthoritiesPopulator userDetailsServiceLdapAuthoritiesPopulator) {
44
45         this.applicationProperties = applicationProperties;
46         this.ldapProperties = ldapProperties;
47         this.userDetailsServiceLdapAuthoritiesPopulator = userDetailsServiceLdapAuthoritiesPopulator;
48     }
49 }

```

3.7. Gestion de la sécurité côté backend

3.7.1. Echange de token

Si l'authentification LDAP est réussie, c'est-à-dire si l'utilisateur a fourni le bon couple login/mot de passe, alors il faut générer un token.

Pour générer ce token, encore une fois, Spring boot fournit les bibliothèques nécessaires. Après avoir ajouté les dépendances nécessaires et avoir créé les classes et fonctionnalités nécessaires, un token était généré automatiquement lors de l'authentification réussie à l'appel du LDAP.

Ce token permet notamment de pouvoir échanger des requêtes HTTP en toute sécurité entre le frontend et le backend. En effet, le token généré lors de l'authentification dans le LDAP réussie est placé dans le header de chaque réponse HTTP que fournit l'API au frontend. Lors de la première réponse que retourne l'API au frontend, le frontend qui reçoit ainsi cette réponse

récupère le token dans le header de la requête HTTP et le stocke. Ainsi, lors de chaque appel à l'API, les requêtes HTTP envoyées au backend contiennent ce token. A chaque requête HTTP, l'API vérifie ainsi le header de la requête afin de vérifier si le token est présent et valide. Si le token est bien présent et est valide, alors l'API effectue les traitements demandés par le frontend. En revanche, si le token n'est pas présent ou si un token est présent mais n'est pas valide, alors l'API n'effectuera aucun traitement et retournera un code erreur (code par exemple le code 401 unauthorized).

Voici des extraits de code de la création du token lorsque l'authentification est réussie ou de la récupération et de vérification de la validité du token récupéré dans le header. Ici encore, le contenu des méthodes n'a pas été écrit de A à Z par mes propres soins :

```

115  /**
116   * Method used to create a new token
117   *
118   * @param authentication the user authentication
119   * @param databaseCheck indicate if we need to check the token in database
120   * @param issuedAt issued date
121   * @return the jwt token
122   */
123  @
124  private String createToken(Authentication authentication, boolean databaseCheck, Date issuedAt, Date validity) {
125
126      List<String> authorities = authentication.getAuthorities().stream().map(GrantedAuthority::getAuthority)
127          .collect(Collectors.toList());
128
129      if (authentication.getPrincipal() instanceof ConnectedUser) {
130
131          // Used by ldapFile and ldap Authentication
132          ConnectedUser connectedUser = (ConnectedUser) authentication.getPrincipal();
133
134          return createToken(authentication.getName(), connectedUser.getLastName(), connectedUser.getFirstName(),
135              connectedUser.getEmail(), StringUtils.join(authorities, separator: ";"), databaseCheck, issuedAt, validity);
136      } else {
137
138          return createToken(authentication.getName(), lastname: "", firstname: "", mail: "", StringUtils.join(authorities, s
139              issuedAt, validity);
140
49  /**
50   * Method used to resolve token from Authorization header
51   *
52   * @param request the http request object
53   * @return the bearer token resolved
54   */
55  @
56  private static String resolveToken(HttpServletRequest request) {
57      String bearerToken = request.getHeader(JWTConfigurer.AUTHORIZATION_HEADER);
58      if (StringUtils.hasText(bearerToken) && bearerToken.startsWith(JWTConfigurer.BEARER)) {
59          return bearerToken.substring(JWTConfigurer.BEARER_LENGTH, bearerToken.length());
60      }
61  }

```

3.7.2. Sécurisation des URL de l'API

Dans la partie précédente, il a été évoqué le fait que si un token était non valide, alors l'accès à la l'URL de l'API était refusé. Cependant, pour que ce mécanisme fonctionne, il faut que l'API connaisse les URL qui nécessitent un token et donc une authentification. Pour cela, encore une fois, Spring boot fournit tout le nécessaire, notamment une bibliothèque permettant de sécuriser les différentes routes de l'API exposées. Grâce à cette bibliothèque, on peut par exemple indiquer à Spring boot qu'on souhaite autoriser les appels à une URL uniquement avec la méthode HTTP GET et que l'URL ne peut être appelée que si dans le header de la requête un token est présent. Ainsi, je me suis chargé de sécuriser chaque URL exposée par l'API, qui devaient nécessiter une autorisation (à savoir, une authentification au préalable réussie dans le LDAP et comprenant un token dans le header de la requête HTTP). Il n'y avait qu'une seule URL qui ne nécessitait pas d'authentification ni de token au préalable : l'URL de l'API permettant l'authentification.

Voici un extrait du code de la méthode configure de la classe HttpSecurity qui s'occupe de sécuriser des URL. En effet, grâce à cette méthode, les URL exposées par l'API ne peuvent pas être appelées sans une authentification préalable. On peut le voir par exemple à la ligne 73 et 74, on voit que les URL qui commencent par /v1.0/ ont besoin d'une authentification au préalable. Cela est rendu possible grâce à la méthode "authenticated()" à la ligne 74. On peut aussi voir à la ligne 66 que l'URL /login est accessible à tout le monde grâce à la méthode "permitAll()" car en effet le fait de pouvoir se connecter doit être possible à tous. Voici l'extrait de code de la méthode configure :

```
52
53 @Override
54 protected void configure(HttpSecurity http) throws Exception {
55     // Ok to disable csrf when evergreen is stateless and cookies not used for
56     // sessionmanagement
57     http.csrf().disable()
58         // by default uses a Bean by the name of corsConfigurationSource
59         .cors().and()
60         // exceptions returned in json format with custom classes
61         .exceptionHandling().ExceptionHandlingConfigurer<HttpSecurity>
62         // custom error handler returns error in json format
63         .authenticationEntryPoint(apiAuthenticationFailureHandler)
64         .accessDeniedHandler(apiAuthenticationFailureHandler).and().authorizeRequests()
65         // everyone can access
66         .antMatchers("/*").permitAll()
67         .antMatchers("/login").permitAll()
68         // everyone can access /authenticate (POST requests) : See LoginFilter example
69         .antMatchers(RequestMethod.GET, ...antPatterns: "/api/v1.0/authenticate/**").permitAll()
70         .antMatchers(RequestMethod.POST, ...antPatterns: "/api/v1.0/authenticate").permitAll()
71         // everyone can access /api/v1/authenticate with HttpMethod.POST : See
72         // AuthenticationController example
73         // all calls to /api need to be authenticated
74         /*.antMatchers("/*").authenticated()*/.antMatchers("/*").authenticated().and();
```

3.8. Gestion de la sécurité côté frontend

3.8.1. Echange de token

Comme expliqué précédemment, si les identifiants fournis par le frontend étaient valides, alors le backend renvoyait un token. Côté frontend, il fallait stocker ce token et le renvoyer à chaque appel sur le backend en le plaçant dans le header des requêtes HTTP. Pour cela, Angular fournit les bibliothèques nécessaires, notamment la bibliothèque Http qui permet de gérer les requêtes Http en valorisant les headers et le “body” des requêtes.

Voici des extraits de code qui illustrent le fait de remplir le header des requêtes HTTP. On peut voir dans la capture d'écran ci-dessous qu'on place dans le header, pour la clé “Authorization”, le token avec le préfixe Bearer :

```
10  @Injectable()
11  export class CustomHttpInterceptor implements HttpInterceptor {
12
13      constructor(private authService: AuthService, private router: Router) {
14      }
15
16      intercept(request: HttpRequest<any>, next: HttpHandler): Observable<HttpEvent<any>> {
17
18          if (this.authService.isAuthenticated()) {
19
20              const token = AuthService.getToken();
21
22              request = request.clone({
23                  setHeaders: {
24                      Authorization: `Bearer ${AuthService.getToken()}`
25                  }
26              });
27
28          if (token != null) {
29              request = request.clone({ headers: request.headers.set(
30                  ['Authorization', 'Bearer ' + token]
31              });
32      }
```

Voici d'autres extraits de code qui illustrent la récupération du token lorsque l'authentification est réussie. On peut voir que si le backend retourne des données, notamment le token, alors on le stocke pour la session en cours. Également, si l'authentification est réussie, on récupère l'utilisateur connecté dans la base données pour récupérer son statut. En fonction de ce statut, il aura plus ou moins de droits dans l'application :

```

76  login(login: string, password: string, rememberMe: boolean) {
77
78    this.userIdLogin = login;
79    const credentials: ICredentials = {};
80    credentials.login = login;
81    credentials.password = password;
82    credentials.rememberMe = rememberMe;
83
84    this.httpClient.post(`.${this.loginUrl}`, credentials, httpOptions).subscribe((data: IToken) => {
85      localStorage.setItem('loginName', login);
86      this.loginError = false;
87      if (data) [
88        // Store Current User in localStorage to workaround F5 issue
89        AuthService.setToken(data.token)];
90      this.isAuthenticated = true;
91      this.isAuthenticatedWrong = false;
92      this.emitAuth();
93      this.emitAuthWrong();
94      this.httpClient
95        .get<User>(`.${this.baseUrl}user/` + this.userIdLogin)
96        .subscribe(
97          (user: User) => {
98            if (user === null) {
99              this.readOnly = true;
100            } else {
101              this.readOnly = false;

```

```

47  static getToken() {
48    return localStorage.getItem(AuthService.tokenItemName);
49  }
50
51  static setToken(token) {
52    localStorage.setItem(AuthService.tokenItemName, token);
53  }
54
55  static removeToken() {
56    localStorage.removeItem(AuthService.tokenItemName);
57  }
58

```

3.8.2. Sécurisation des URL de l'application

Un point important que j'ai dû également implémenter a été le fait de pouvoir sécuriser les différentes vues de l'application. Autrement dit, si aucune authentification n'a été effectuée, on ne peut accéder à un aucun URL de l'application qui nous redirige vers la page d'authentification. Pour cela, Angular propose des "Guard" sur les différentes routes de l'application. Si la condition d'une guard appliquée à une route n'est pas remplie, alors l'URL n'est pas accessible.

Voici des exemples de codes qui s'occupent de sécuriser les différentes URL de l'application. Ci-dessous, on peut par exemple voir que l'url */inventory est sécurisée par le service AuthGuardService :

```
8  const routes: Routes = [
9    { path: '', canActivate: [AuthGuardService], component: AuthenticationComponent },
10   { path: 'inventory', canActivate: [AuthGuardService], component: UpdateInventoryComponent },
11   { path: 'auth', component: AuthenticationComponent },
12   { path: '**', redirectTo: '' }
13 ];
```

Dans la capture d'écran ci-dessous, on retrouve justement ce service. La méthode canActivate du service s'occupe tout simplement de retourner "true" si l'authentification a au préalable réussi, et false dans le cas contraire. Ainsi, si on tente de naviguer vers l'URL */inventory mais que nous ne sommes pas authentifiés, alors nous seront redirigés vers la page de connexion (grâce à la ligne this.router.navigate(['auth']) qui nous redirige vers la page de connexion):

```
12  export class AuthGuardService {
13
14    constructor(
15      private router: Router,
16      private authService: AuthService
17    ) { }
18
19    canActivate(
20      route: ActivatedRouteSnapshot,
21      state: RouterStateSnapshot
22    ): Observable<boolean> | Promise<boolean> | boolean {
23      if (this.authService.isAuthenticated) {
24        return true;
25      } else {
26        this.router.navigate(['auth']);
27      }
28    }
29  }
```

A noter que la première capture d'écran représente le fichier de routing du framework Angular. On peut notamment voir qu'à chaque route définie correspond un composant. Ainsi, lorsqu'on navigue à une URL, le composant lui correspondant est généré et les composants générés pour la route sur laquelle on se trouvait précédemment sont détruits par Angular. Cela évite de consommer trop de ressources pour le navigateur, augmentant grandement les performances et la fluidité de l'application et rendant une expérience d'utilisateur agréable. On notera donc la grande modularité que permet le framework Angular qui est justement la philosophie du framework: découper au maximum et de façon modulaire les applications et tenter au maximum de créer des modules réutilisables dans l'application développée et même dans d'autres applications.

3.9. *Implémentation de la partie admin*

Arrivé sur la fin de mon stage, les tâches qui m'avaient été demandées de faire étaient terminées. J'ai alors pris l'initiative de développer une interface pour les admins de l'application qui allait notamment permettre d'ajouter des utilisateurs ayant des droits particuliers, d'ajouter des technologies, des versions pour les technologies et autres.

3.9.1. *Gestion des droits d'administration*

Avant tout, je me suis donc occupé, lors de l'authentification, d'afficher l'onglet ADMIN de façon conditionnel en fonction du statut de l'utilisateur qui se connecte. Pour cela, étant donné que la table "User" possède une colonne "status", en fonction du statut de l'utilisateur, l'onglet ADMIN est affiché ou masqué. Cette gestion des droits étant terminée, je me suis ensuite occupé dans un premier temps du design du panel d'administration.

3.9.2. *Création de l'interface d'administration*

Ainsi, j'ai créé une interface pour les admins de l'application qui leur permet d'ajouter des valeurs, d'en supprimer et d'en mettre à jour. De ce fait, par l'intermédiaire d'une interface graphique, la base de données est mise à jour par les admins sans écrire le moindre code SQL et sans aller modifier directement la base de données ce qui permet de faciliter la mise à jour des valeurs de l'application.

Cette administration est particulièrement intéressante étant donné que les versions des différentes technologies évoluent rapidement et donc cela demande une mise à jour régulière des valeurs stockées dans la base de données. Également, les leaders des différentes BS peuvent être amenés à changer, ainsi les tuples de la table "User" sont également amenés à être modifiés. Cela pourra donc se faire directement par l'interface graphique de l'application sans aller changer les valeurs directement dans la base de données.

Voici des captures d'écran de l'interface graphique pour l'administration de l'application, avec, dans la barre de navigation, un onglet "inventory" qui permet de retourner sur l'inventaire en cas de besoin. Ci-dessous, l'interface permettant d'ajouter des valeurs :

Administration of the Obso Tracker

Add values

Add a DevOps or adminMichelin ID* BS of the DevOps*
Michelin ID (FXXXXXX)Status of the user to add*

Add DevOps

Add a technologyTechnology to add*
Technology to add

Add technology

Add an environmentEnvironment to add*
Environment (dev, prod, indus ...)

Add environment

Add a version for a technologyTechnology to add a version*
WMQVersion to add for the selected techn*
9.7

Add version

Sur la capture d'écran suivante, on peut notamment voir qu'à l'ajout d'une valeur, une pop-up est affichée en haut au milieu de l'écran pour indiquer à l'utilisateur que l'ajout a correctement été effectué :

Environment successfully added !

Michelin ID* BS of the DevOps*
Michelin ID (FXXXXXX)Status of the user to add*

Add DevOps

Technology to add

Add technology

Add an environmentEnvironment to add*
This field is required

Add environment

Add a version for a technologyTechnology to add a version* Version to add for the selected techn*
9.7

Add version

Voici également quelques extraits de code qui correspondent aux traitements effectués derrière les différentes fonctionnalités de ce panel d'administration. Ci-dessous, la capture d'écran représente le code HTML :

```
<form [formGroup]="adminForm" (ngSubmit)="onSubmitFormAdmin()">
  <p class="title-devops">Add a DevOps or admin</p>
  <div class="controles-container">
    <mat-form-field class="input-id-michelin">
      <input matInput formControlName="michelinId" matTooltip="The michelin ID DevOps that you
        matTooltipPosition="above" placeholder="Michelin ID">
      <mat-hint>Michelin ID (FXXXXXX)</mat-hint>
      <mat-error>
        This field is required
      </mat-error>
    </mat-form-field>
    <mat-form-field>
      <mat-label>BS of the DevOps*</mat-label>
      <mat-select matTooltip="Code of the BS" matTooltipPosition="above" formControlName="bss">
        <mat-option [value]="" *ngFor="let value of bss">
          {{value}}
        </mat-option>
      </mat-select>
```

Ici, la capture d'écran montre un exemple de comment est gérée la soumission du formulaire d'ajout d'un utilisateur :

```
64  /**
65   * On click on the submit form, post new user entered in the database
66   */
67  onSubmitFormAdmin() {
68    let statut: number;
69    if (this.adminForm.value.status === 'DevOps') {
70      statut = 1;
71    } else {
72      statut = 0;
73    }
74    this.userService.postNewUser({
75      name: this.adminForm.value.michelinId.toUpperCase(),
76      status: statut,
77      bss: this.adminForm.value.bss.toUpperCase()
78    }).subscribe();
79    this.adminForm.reset();
80    this.notificationService.success('User successfully added !');
81  }
```

Enfin, on appelle l'API pour ajouter l'utilisateur saisi dans le formulaire :

```
14  | postNewUser(user) {
15  |   return this.httpClient.post(`${this.baseUrl}user`, user);
16  |
17  | }
```

Ci-dessous cette fois-ci, l'onglet permettant la suppression de valeurs, notamment la suppression des droits que possède un utilisateur de l'application. Ici, on peut voir que l'utilisateur E000421 qui possède des droits d'admin (affiché entre parenthèses) va se voir supprimer ses droits d'admin au clic sur le bouton "remove rights" :

The screenshot shows a modal window titled "Remove rights on a DevOps or admin". At the top, there are three buttons: "Add values", "Delete values", and "Update values". Below these buttons, a dropdown menu labeled "User to delete*" contains the option "E000421 (admin)". At the bottom of the modal is a blue "Remove rights" button.

Enfin, voici le panel d'administration permettant aux admins de l'application de pouvoir mettre à jour un certain nombre de valeurs. On peut notamment modifier les droits d'un utilisateur et activer ou désactiver une version actuelle ou une version cible pour une technologie donnée :

The screenshot shows the "Administration of the Obso Tracker" interface. At the top, there are three buttons: "Add values", "Delete values", and "Update values". The main area is divided into three sections:

- Change rights of a DevOps or admin**: Contains fields for "User to change rights*" and "New wanted status fo...". A button at the bottom is "Update the rights of the user".
- Disable or enable a current version for a technology**: Contains fields for "Select the technolo..." and "Select the version*". A button at the bottom is "Update the status".
- Disable or enable a target version for a technology**: Contains fields for "Select the technolo..." and "Select the version*". A button at the bottom is "Update the status".

Ci-dessous, on peut apercevoir un extrait du code HTML permettant d'afficher le panel d'administration de la mise à jour des valeurs :

```
34  <form [formGroup]="statusOfCurrentVersionForm" (ngSubmit)="onUpdateStatusCurrentVersion()">
35      <div class="title marg-top">Disable or enable a current </div><div class="title align">version for
36      <div class="controles-container">
37          <mat-form-field>
38              <mat-label>Select the technology*</mat-label>
39              <mat-select matTooltip="Select the technology which you want to enable or disable a current
40                  <mat-option (click)="getVersionsOfTechnology(value)" [value]="value" *ngFor="let value
41                      &gt; {{value}}</mat-option>
42                  </mat-select>
43                  <mat-error>
44                      This field is required
45                  </mat-error>
46              </mat-form-field>
47              <mat-form-field>
48                  <mat-label>Select the version*</mat-label>
49                  <mat-select matTooltip="Select the current version that you want to disable or enable. You
50                      <mat-option [value]="value[0]" *ngFor="let value of currVersion">
51                          {{value[0]}}
52                          <span *ngIf="value[1] === 0">(status: disable)</span>
53                          <span *ngIf="value[1] === 1">(status: enable)</span>
54                      </mat-option>
55                  </mat-select>
56                  <mat-error>
57                      This field is required
58                  </mat-error>
59              </mat-form-field>
60              <mat-form-field>
61                  <mat-label>Select the status*</mat-label>
62                  <mat-select matTooltip="Select the status to set for the current version and technology se
```

On peut voir le code qui est exécuté lorsqu'on clique sur le bouton "update the status" dans l'encadré "disable or enable a current version for a technology". La méthode « onUpdateStatusCurrentVersion » s'occupe d'appeler une méthode qui fait un appel à l'API pour mettre à jour la base de données, notamment le statut de la version sélectionnée :

```

72  }
73  onUpdateStatusCurrentVersion() {
74    const stat = this.statusOfCurrentVersionForm.value.status === 'Disable' ? 0 : 1;
75    this.technologyService.updateStatusCurrentVersionTechnology({
76      technology: this.statusOfCurrentVersionForm.value.technology,
77      version: this.statusOfCurrentVersionForm.value.version,
78      enableVersion: stat
79    }).subscribe(
80      () =>
81        this.technologyService.getTechnologies().subscribe(
82          res => [
83            this.technologies = res;
84            this.currVersion = [];
85          ]
86        );
87        this.notificationService.success(
88          this.statusOfCurrentVersionForm.value.version + ' current version for the ' +
89          this.statusOfCurrentVersionForm.value.technology + ' technology successfully ' +
90          this.statusOfCurrentVersionForm.value.status.toLowerCase() + ' !'
91        );
92        this.statusOfCurrentVersionForm.reset();
93    }

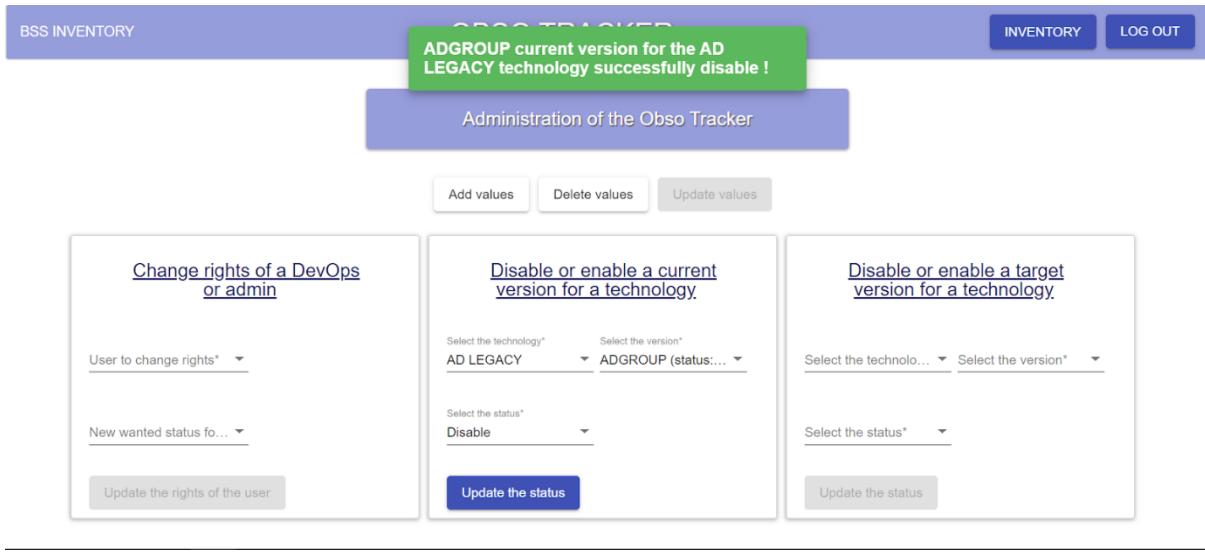
```

Enfin, voici ci-dessous des exemples concrets de mises à jour de valeurs. Sur la première capture d'écran, on peut voir qu'on veut mettre les droits d'admin à l'utilisateur FX30405 qui avait des droits de DevOps auparavant. Sur la seconde capture d'écran, on peut voir qu'on souhaite désactiver la version actuelle "EUW" pour la technologie "AD LEGACY". Enfin, sur la troisième capture d'écran, on peut voir qu'un message d'information indique à l'admin l'action qu'il a choisi d'effectuer :

The image consists of three screenshots of a software application's user interface:

- Screenshot 1: Change rights of a DevOps or admin**
A modal window titled "Change rights of a DevOps or admin". It shows a dropdown menu where "FX30405 (DevOps)" is selected from a list containing "E000421 (admin)". Below the dropdown, there is a text input field with placeholder "New wanted status for th...". A dropdown menu below it shows "Admin" selected from options like "User" and "Admin". At the bottom is a blue button labeled "Update the rights of the user".
- Screenshot 2: Disable or enable a current version for a technology**
A modal window titled "Disable or enable a current version for a technology". It has a dropdown menu for "Select the technology*" showing "AD LEGACY". Another dropdown menu for "Select the status*" shows "Disable" selected from options like "Enable" and "Disable". A list of technologies and their current status is shown in a table:

Technology	Status
ADGROUP	(status: enable)
ADR	(status: disable)
DECOM	(status: enable)
EUW	(status: enable)
- Screenshot 3: Confirmation message**
A simple confirmation message: "Success! The rights have been updated for user FX30405."



3.9.3. Implémentation des fonctionnalités côté API

Côté backend, l'implémentation a été rapide, il m'a suffi d'écrire les URL qu'il est possible d'aller consommer par le frontend avec les traitements nécessaires sur la base de données, notamment l'ajout de valeurs.

4. Mise en industriel de l'application

A ce stade-là de mon stage, les tâches que l'on m'avait demandées d'effectuer étaient terminées. A partir de là, il a fallu que je mette l'application et le backend dans l'environnement industriel avant la mise en production dans un second temps, c'est-à-dire sur le serveur du service dans lequel j'ai effectué mon stage afin que l'application soit accessible par les différents services de chez Michelin.

L'environnement industriel est un environnement de test, se rapprochant de l'environnement de production et permettant d'effectuer un test simulant la mise en production pour s'assurer que tout fonctionne correctement. C'est l'étape préalable à la mise en production permettant de valider la mise en production.

Ainsi, je me suis occupé de mettre en indus (abréviation d'industriel) l'application et le backend.

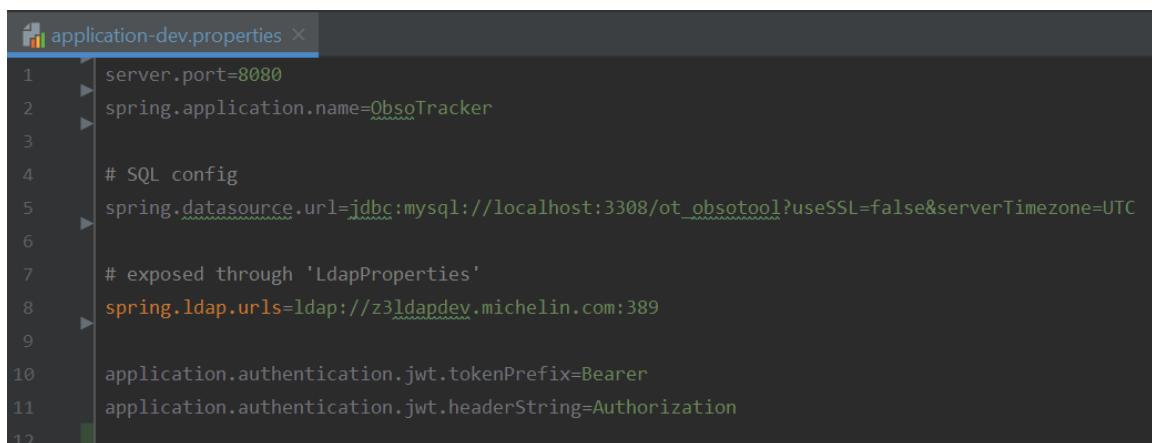
4.1. Mise en industriel du backend

Il faut savoir que la mise en production n'a pas pu être réalisée par mes soins, car avant une mise en production, l'application doit suivre tout un procédé de validation qui dure plusieurs semaines. J'ai en revanche réalisé la mise en indus.

Cependant, la mise en industriel comprend toutes les mêmes étapes détaillées ci-après que la mise en production, exception faite de l'adresse de connexion de la base de données et de l'adresse du serveur qui diffèrent.

Pour placer l'API sur le serveur Linux du service, il a fallu dans un premier temps que je génère un fichier JAR de l'API. Cela est possible avec une commande que propose Maven qui un outil de gestion de production permettant notamment la gestion de l'intégration continue. Une fois ce fichier avec l'extension .jar généré, je me suis occupé de créer le dossier contenant le frontend et le backend sur le serveur Linux. Le dossier du backend contient notamment le fichier .jar généré auparavant, un dossier "config" permettant de gérer les différentes configurations du backend, un dossier "jre" contenant le JRE (Java Runtime Environment) permettant d'exécuter le fichier du backend (le fichier .jar), un dossier "log" permettant de récupérer les différents logs des appels qui est utile notamment pour du débogage. La création des différents dossiers a été effectuée en ligne de commande, notamment avec l'interpréteur de commande Bash étant donné que le serveur et un serveur Linux. Pour ce qui est du transfert du fichier .jar sur le serveur, je me suis servi de WinSCP qui est un client SFTP graphique pour Windows et qui se sert de SSH qui est un protocole de communication sécurisé. WinSCP permet de copier de façon sécurisée des fichiers entre un ordinateur local et un ordinateur distant, ce qui m'a ainsi permis par exemple de copier le fichier .jar, présent sur mon ordinateur local, sur le serveur Linux.

Spring boot propose des fichiers de configuration qui facilite énormément la vie aux développeurs, notamment pour gérer les différents environnements (environnement de développement, de production, d'indus). Ainsi, j'ai créé pour chaque environnement un fichier de configuration en renseignant les différentes bases de données utilisées par chacun des environnements par exemple ou bien les ports d'écoute de l'API. Voici quelques extraits de ces fichiers de configuration. On peut notamment apercevoir sur les captures d'écran les ports d'écoute, les url des bases de données et même les url des annuaires LDAP :



The screenshot shows a code editor window with the file 'application-dev.properties' open. The file contains the following configuration properties:

```
server.port=8080
spring.application.name=ObsoTracker

# SQL config
spring.datasource.url=jdbc:mysql://localhost:3308/ot_obsotool?useSSL=false&serverTimezone=UTC

# exposed through 'LdapProperties'
spring.ldap.urls=ldap://z3ldapdev.michelin.com:389

application.authentication.jwt.tokenPrefix=Bearer
application.authentication.jwt.headerString=Authorization
```

```

application-dev.properties x application-prod.properties x
1 server.port=8083
2
3 spring.application.name=ObsoTracker
4
5 # SQL config
6 spring.datasource.url=jdbc:mysql://icgvgb10.michelin.com:3306/icgvgb13?useSSL=false&serverTimezone=UTC
7
8 ##### Ldap #####
9
10 # exposed through 'LdapProperties'
11 spring.ldap.urls=ldap://z3ldap.michelin.com:636
12
13 application.authentication.jwt.tokenPrefix=Bearer
14 application.authentication.jwt.headerString=Authorization

```

Ainsi, le fichier de configuration application-indus.properties a été placé sur le serveur Linux lors du déploiement en indus pour prendre en compte tout l'environnement indus notamment le LDAP et la base de données d'indus.

Ci-après, voici les commandes m'ayant permis de déployer le backend. La commande « mvn clean package » créée le fichier .jar du backend :

```

Terminal: Local (2) +
Microsoft Windows [version 10.0.16299.1686]
(c) 2017 Microsoft Corporation. Tous droits réservés.

C:\Users\E000421\Desktop\evergreen>mvn clean package

```

Ensuite, on place ce fichier .jar sur le serveur Linux :

```
mv /home/users/E000421/obsotracker-api-0.0.1.jar /tmp/obsotracker|
```

Ensuite on change le propriétaire du fichier .jar pour que tous les membres du groupe "icgvgb1a" puissent avoir des droits sur le fichier :

```
chown icgvgb1a /tmp/obsotracker/obsotracker-api-0.0.1.jar|
```

Il nous reste plus qu'à aller se placer dans le dossier exploit qui contient notamment les trois scripts bash qui permettent de stopper l'API, de la déployer et de la démarrer :

```
cd /busapps/icgv/gb10/obsotracker/exploit|
```

```
./stop_api_ot.sh|
```

```
./deploy_api_ot.sh
```

```
./start_api_ot.sh
```

A partir de là, le backend était déployé en indus et opérationnel.

Voici des extraits des scripts bash que j'ai écrit pour le déploiement du backend. La capture d'écran ci-dessous permet de stopper l'API en allant supprimer le fichier api.pid qui contient l'id du process de l'API ce qui a pour conséquence de stopper le back :

```
#!/bin/bash
#
# Generated during deployment
#
# This script is called by init scripts to stop the obsotracker API application component

# exit status = 0 SUCCESS
# exit status > 0 FAILED
#
RETVAL=0

if [ "`whoami`" != "icgvgbia" ]
then
    echo "ERROR : This script must be launched by @ENV_LOWER_CASE@@APP_CODE_LOWER_CASE@@SCOPE_LOWER_CASE@@APP_INSTANCE@ owner only !"
    exit 100
fi

PID=`cat /busapps/icgv/gb10/obsotracker/run/api.pid`
echo "kill process $PID"
kill $PID
sleep 5
rm -f /busapps/icgv/gb10/obsotracker/run/api.pid
RETVAL=$?

exit $RETVAL
```

La capture d'écran ci-dessous décrit le déploiement du back. On retrouve notamment tout d'abord un case qui récupère le paramètre fourni lors de l'appel du script et qui effectue des traitements en fonction de ce dernier. Ensuite dans le script, on s'occupe de changer les droits sur le fichier de l'api et de remplacer un fichier par un autre :

```

#!/bin/bash
#####
# Script : deploy_api_ot.sh
#
#-----#
# Description : Deploy Obso tracker API
#-----#
# Parameters : * 1 : action
#-----#
# Exit codes : * 0 if OK
# Exit codes : * 1 if error
#####

Action=$1

Usage="Usage: $(basename $0) [clean|download artifactory_path|deploy]"

case ${Action} in
    clean ) rm -Rf /tmp/obsotracker; mkdir /tmp/obsotracker; exit 0 ;;
    download ) wget --no-check-certificate -nv -O /tmp/obsotracker/obsotracker-frontend.jar $2; exit 0 ;;
    deploy ) ;;
    * ) echo ${Usage}; exit 1;;
esac

chown icvgvbla /tmp/snn/obsotracker-api.jar
mv /tmp/obsotracker/*.jar /busapps/icgv/gbl0/obsotracker/obsotracker-frontend.jar
rm -rf /busapps/icgv/gbl0/obsotracker/temp/*

if [ $? -eq 0 ]
then
    echo "Successfully executed script"
    exit 0
else
    # Redirect stdout from echo command to stderr.
    echo "Script exited with error." >&2
    exit 1
fi

```

4.2. *Mise en industriel du frontend*

Pour placer la partie cliente sur le serveur, il a tout d'abord fallu gérer les différents environnements. Pour cela, j'ai créé 3 fichiers distincts permettant de gérer l'environnement de développement, d'indus et de production. Au moment du déploiement en indus ou en prod, il suffit ainsi de générer le bon fichier.

Voici par exemple les fichiers s'occupant respectivement de l'environnement de développement et de production avec, en fonction de l'environnement, les bonnes URL de l'API à consommer :

```
TS environment.prod.ts  TS environment.ts X  {} package.json  update-inven
src > environments > TS environment.ts > [e] environment
● 5
6 export const environment = [
7   production: false,
8   indus: false,
9   dev: true,
10  // URL of development API
11  apiUrl: 'http://localhost:8080/api/v1.0',
12
13  baseUrl: 'http://localhost:8080/api/v1.0/',
14
15  // URL of login
16  loginUrl: 'http://localhost:8080/api/v1.0/authenticate',
17
18  // URL of logout
19  // logoutUrl: 'http://localhost:8080/radar/logout',
20
21  // used by interceptor to add withCredentials or not (controls
22  withCredentials: false,
23
24  VERSION: require('../package.json').version
25
26 ];
```

```
TS environment.prod.ts X  {} package.json  update-inventory.component.html  TS update-inventory.compo
src > environments > TS environment.prod.ts > [e] environment
1 declare var require: any;
2
3 export const environment = [
4   production: true,
5   indus: false,
6   dev: false,
7
8   // URL of development API
9   baseUrl: 'https://rcgvgbl0.michelin.com:8443/obsotracker/api/v1.0/',
10
11   // URL of login
12   loginUrl: 'https://rcgvgbl0.michelin.com:8443/obsotracker/api/v1.0/authenticate',
13
14   // URL of logout
15   // logoutUrl: 'https://rcgvgbl0.michelin.com:443/snnapi/api/v1/logout',
16
17   // used by interceptor to add withCredentials or not (controls use of JSESSIONID)
18   withCredentials: false,
19
20   VERSION: require('../package.json').version
21
22 ];
```

Une fois cette gestion des environnements terminée il a fallu que je build l'application pour obtenir le dossier à placer sur le serveur. Cela s'est fait avec la commande qui suit dans la capture d'écran dans la fenêtre du bas (dans le terminal permettant d'être en ligne de commande). C'est ensuite Angular qui s'occupe de générer le dossier nécessaire à placer sur le serveur :

```

 1  {
 2    "name": "evergreen-frontend",
 3    "version": "0.0.0",
 4    "scripts": [
 5      "ng": "ng",
 6      "start": "ng serve",
 7      "build": "ng build",
 8      "test": "ng test",
 9      "lint": "ng lint",
10      "e2e": "ng e2e",
11      "buildIndus": "ng build --configuration=indus",
12      "buildProd": "ng build --configuration=production"
13    ],
14    "private": true,
15    "dependencies": {
16      "@angular/animations": "~9.0.0",
17      "@angular/cdk": "~9.0.0",
18      "@angular/common": "~9.0.0",
19      "@angular/compiler": "~9.0.0",
20      "@angular/core": "~9.0.0",
21      "@angular/forms": "~9.0.0",
22      "@angular/material": "~9.0.0",

```

OUTPUT TERMINAL DEBUG CONSOLE PROBLEMS 7

```

WARNING in budgets: Exceeded maximum budget for initial-es2015. Budget 2 MB was
B.

WARNING in budgets: Exceeded maximum budget for initial-es5. Budget 2 MB was no
C:\Users\E000421\evergreen-front\evergreen-frontend>npm run-script buildIndus

```

Sur la capture d'écran suivante, on peut notamment voir le fichier de configuration “angular.json” qui permet de choisir le bon fichier d'environnement lors du build et qui s'occupe ici par exemple de remplacer le fichier “environment” par le fichier “environment.prod” si on lance la commande “buildProd” (commande présente dans la capture précédente et qui appelle --configuration=production ce qui va aller choisir le fichier environment.prod):

```

 1  {
 2    "projects": {
 3      "evergreen-frontend": {
 4        "architect": {
 5          "build": {
 6            "configuration": "node_modules/@angular/material/build-styles/indus"
 7            "node_modules/bootstrap/dist/css/bootstrap.css",
 8            "src/styles.scss"
 9          ],
10          "scripts": []
11        },
12        "configurations": [
13          "production": {
14            "fileReplacements": [
15              {
16                "replace": "src/environments/environment.ts",
17                "with": "src/environments/environment.prod.ts"
18              }
19            ],

```

Ensute, je me suis occupé de zipper le contenu du dossier généré par la commande ci-dessus et d'ensuite placer le zip dans mon dossier /home sur le serveur en me servant de commande Linux. Voici les étapes suivantes procédure sur les captures d'écran. D'abord on déplace le dossier zip dans le dossier tmp sur le serveur :

```
mv /home/users/E000421/obsotracker-client.zip/ /tmp/obsotracker|
```

Ensute, on change le propriétaire du dossier zippé pour avoir les droits adéquats sur le dossier, notamment les droits partagés par l'ensemble des développeurs travaillant sur le projet :

```
chown icvgvbl1a /tmp/obsotracker/obsotracker-client.zip|
```

Ensute on se place dans le dossier exploit qui contient les scripts permettant de stopper le client, de la déployer puis de la lancer :

```
cd /busapps/icgv/gb10/obsotracker/exploit|
```

```
./stop_client_ot.sh|
```

```
./deploy_client_ot.sh|
```

```
./start_client_ot.sh|
```

Grâce aux 3 scripts bash ci-dessus, l'application front était ainsi opérationnelle et déployée en indus.

Voici à titre d'exemple quelques extraits des scripts bash appelés ci-dessus. La première capture d'écran s'occupe du déploiement de la partie cliente de l'application. Le script effectue tout d'abord des traitements en fonction du paramètre fourni lors de l'appel du script. Ensute, il s'occupe notamment de dézipper le dossier zip contenant l'application cliente et de déplacer le dossier dézippé dans le bon dossier sur le serveur. Avec la commande "sed", on remplace tous les chaînes de caractères "rcvgvbl0" par "icvgvbl0" dans le fichier main*js. Cela était nécessaire car le fichier main.js contenant les URL de l'API en prod (à savoir l'URL qui commençait par rcvgvbl0). Or nous sommes dans un environnement d'indus et l'URL de l'API d'indus commence par "icvgvbl0", d'où la commande « sed » :

```

#!/bin/bash
#####
# Script : deploy_client_ot.sh
#-----
# Description : Deploy obsotacker client
#-----
# Parameters : * 1 : action
#               * 2 : artifactory_path
#-----
# Exit codes : * 0 if OK
# Exit codes : * 1 if error
#####

Action=$1

Usage="Usage: $(basename $0) [clean|download artifactory_path|deploy]"

case ${Action} in
    clean ) rm -Rf /tmp/obsotacker; mkdir /tmp/obsotacker; exit 0 ;;
    download ) wget --no-check-certificate -nv -O /tmp/obsotacker/obsotacker-frontend.zip $2; exit 0 ;;
    deploy ) ;;
    * ) echo $Usage; exit 1;;
esac

RETVAL=0

rm -rf /busapps/icgv/gbl0/obsotacker/frontend/*
unzip /tmp/obsotacker/evergreen-frontend.zip -d /busapps/icgv/gbl0/obsotacker/frontend/
sed 's/xcvqgb10/icgvqb10/g' /busapps/icgv/gbl0/obsotacker/frontend/main*.js
rm -rf /busapps/icgv/gbl0/obsotacker/temp/*

exit $RETVAL

```

Quant à la seconde capture d'écran ci-dessous, le script s'occupe de démarrer l'application cliente. Il n'y a pas grand-chose dans ce script, la commande httpd start permet tout simplement de démarrer l'application :

```

#!/bin/bash
#####
# Script : start_client_ot.sh
#-----
# Description : Start Obso Tracker frontend client
#-----
# Exit codes : * 0
#####

service httpd start

exit 0

```

5. Automatisation du déploiement grâce à une configuration de fichiers

Rendre possible l'automatisation d'une application est très utile et est un grand gain de temps. On l'a vu avec deux étapes de déploiement du frontend et du backend : c'est assez long et cela demande l'exécution de beaucoup de commandes et de beaucoup d'étapes. C'est ici que la CI/CD va être utile.

Chez Michelin, la CI/CD est très présente. CI/CD signifie “Continuous Integration, Continuous Deployment” traduit par intégration continue et déploiement continu. Pour cela, GitLab propose des méthodes permettant de faire de la CI/CD, notamment grâce au fichier YAML

gitlab-ci.yml. Ce fichier est en fait un fichier de configuration, dans lequel on place des lignes de commande, et qui vont être exécutées par GitLab lors d'un push sur un repo par exemple.

Dans le cadre de mon application, j'ai écrit des fichiers de configuration permettant l'automatisation du déploiement sur le serveur lors d'un push sur la branche master sur le dépôt GitLab de l'application. J'ai ainsi écrit 2 fichiers de configuration : un pour la partie cliente et un pour la partie backend permettant respectivement de déploiement sur le serveur le front ou le back si un push est réalisé sur la branche master du front ou du back.

Voici un extrait du fichier de configuration pour la partie front de l'application. Sur cet extrait par exemple, au moment où l'on push sur GitLab, le script s'occupe, entre autres, de builder l'application et de créer un zip du dossier contenant l'application :

```
1 |   image: docker.artifactory.michelin.com/michelin-tools:1.4-alpine3.10
2 |
3 |   stages:
4 |     - deploy
5 |
6 |   build:
7 |     stage: deploy
8 |     script:
9 |       - ls
10 |       - npm install
11 |       - npm run lint
12 |       - npm run-script buildProd
13 |       - APP_VERSION=$(jq -r ".version" package.json)
14 |       - cd dist
15 |       - cd evergreen-frontend
16 |       - zip -r evergreen-frontend.zip *
```

Voici un autre extrait, cette fois-ci du fichier de configuration de la partie back de l'application. Sur l'extrait suivant, le script s'occupe notamment de créer une connexion ssh, de se connecter avec l'user "icgvgbla" sur la machine Linux (le serveur) et de lancer différents scripts comme le déploiement de l'api, l'arrêt de l'api, et le démarrage de l'API :

```
└─production:
  stage: deploy
  script:
    - echo "Deploy application version ${CI_COMMIT_TAG}"
    - jfrog rt c artifactory --url=https://artifactory.michelin.com --apikey=$ARTIFACTORY_TOKEN
    - eval $(ssh-agent -s)
    - echo "$SSH_PRIVATE_KEY" | tr -d '\r' | ssh-add - > /dev/null
    - ssh $SSH_USER@PROD_SERVER "sudo -nu icgvgbla /busapps/icgv/gb10/obsotracker/exploit/deploy_api_ot.sh clean"
    - ssh $SSH_USER@PROD_SERVER "sudo -nu icgvgbla /busapps/icgv/gb10/obsotracker/exploit/deploy_api_ot.sh download"
    - ssh $SSH_USER@PROD_SERVER "sudo -nu icgvgbla /busapps/icgv/gb10/obsotracker/exploit/stop_api_ot.sh"
    - ssh $SSH_USER@PROD_SERVER "sudo -nu icgvgbla /busapps/icgv/gb10/obsotracker/exploit/deploy_api_ot.sh deploy"
    - ssh $SSH_USER@PROD_SERVER "sudo -nu icgvgbla /busapps/icgv/gb10/obsotracker/exploit/start_api_ot.sh"
```

6. Documentation

Afin de permettre aux futurs développeurs qui vont travailler et faire évoluer l'application que j'ai développé, j'ai réalisé une documentation technique permettant de s'imprégner du projet un peu plus facilement. J'ai également écrit une documentation utilisateur pour faciliter l'utilisation de l'application aux futurs utilisateurs.

J'ai réalisé la documentation grâce au langage markdown. La documentation a été placée dans l'onglet "Wiki" du groupe Michelin sur GitLab. Voici un extrait du document écrit en markdown ainsi que le rendu visuel du document

Voici ci-dessous un extrait de la documentation écrit en langage markdown :

```
## Application Overview

# Link
[https://icgvgb10.michelin.com:8443](https://icgvgb10.michelin.com:8443)

# Description

The application is a tool that allows you to view, manage and follow the obsolescence of the different middleware.

# High-level architecture

The platform needs 3 components :
- the Obso Tracker API
- the Angular Client
- the CGV database

# Zone(s) of usage

The tools is use all other the world.

# Working and non-working hours

The application is working on all zones times.

# Internal and external Support groups

L2: SPOC team
TODO: give the name of the L3 support

# Access to the tools

To login into this tools, you must have **LDAP rights**
```

```

# Database

Mysql database:

- **Indus**

    - host: icgvgbl0.michelin.com
    - port: 3306
    - user: ???
    - password: *****
    - schema: icgvgbl3

- **Prod**

    - host: ???
    - port: 3306
    - user: u_rcgvgbl3_dba
    - password: *****
    - schema: rcgvgbl3

```

```

# How to use Obso Tracker

Log in with your LDAP id and password
![alt text](img/login.JPG "Login page")

## Your BS inventory

When you're authenticate, you're redirected on the home page that display all your BS inventory for your BS if you're a DevOps leader.
![alt text](img/your-inventory.JPG "Your BS inventory")

On the click of the "Add values" button, you can add a new record.
![alt text](img/new-record.JPG "Add new record")

On the click of the update button, in the Update column of the table, you can update the row.
![alt text](img/update-record.JPG "Update a record")

On the click of the delete button in the last column of the table, you can delete all the row selected with the checkbox.
![alt text](img/delete-record.JPG "Delete a record")

If you're aren't a DevOps leader, you just can visualize the records. In the same way, if you're connected as DevOps leader, you can see this view on the click of the "Display all BS" button.

![alt text](img/all-bs.JPG "Display all BS inventory")

On the click of the "import" button, you can choose a excel file to import. On te import, a pop-up display how many row are imported and the potential errors raises.
![alt text](img/information-import.JPG "Display information of the imported excel file")

```

Et voici maintenant des extraits de la documentation technique et utilisateur visuels :

Project overview Repository Issues (0) Jira Merge Requests (0) CI / CD Operations Analytics Wiki Snippets	<h3>Link</h3> <p>Indus: https://icgvgbl0.michelin.com:8443</p> <h3>Description</h3> <p>The application is a tool that allows you to view, manage and follow the obsolescence of the different middlewares.</p> <h3>High-level architecture</h3> <p>The plateform needs 3 components :</p> <ul style="list-style-type: none"> • the Obso Tracker API • the Angular Client • the CGV database <h3>Zone(s) of usage</h3> <p>The tool is used all over the world.</p>
---	---

O Obso-tool-client

- Project overview
- Repository
- Issues 0
- Jira
- Merge Requests 0
- CI / CD
- Operations

Working and non-working hours

The application is working on all zones times.

Internal and external Support groups

L2: SPOC team TODO: give the name of the L3 support

- Project overview
- Repository
- Issues 0
- Jira
- Merge Requests 0
- CI / CD
- Operations
- Analytics
- Wiki
- Snippets

Database

Mysql database:

- Indus

- host: icgvgbl0.michelin.com
- port: 3306
- user: u_icgvgbl3_ot_rw
- password: ****
- schema: icgvgbl3

- Prod

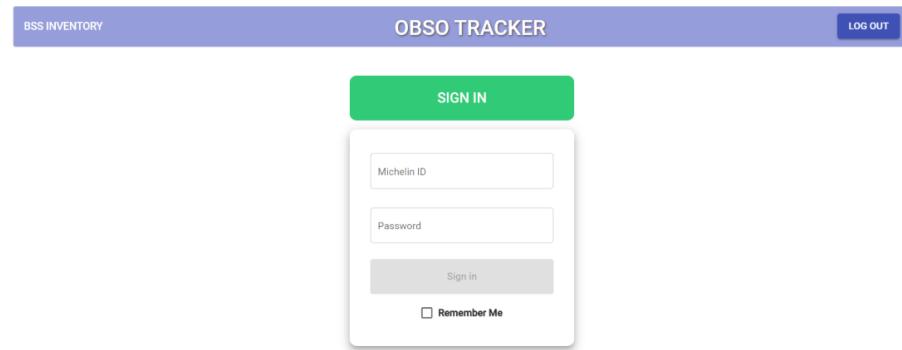
- host: rcgvgbl0.michelin.com
- port: 3306
- user: u_rcgvgbl3_dba
- password: ****
- schema: rcgvgbl3

O Obso-tool-client

- Project overview
- Repository
- Issues 0
- Jira
- Merge Requests 0
- CI / CD
- Operations
- Analytics
- Wiki

How to use Obso Tracker

Log in with your LDAP id and password



O Obso-tool-client

- Project overview
- Repository
- Issues 0
- Jira
- Merge Requests 0
- CI / CD
- Operations
- Analytics
- Wiki**
- Snippets

Your BS inventory

When you're authenticate, you're redirected on the home page that display all your BS inventory for your BS if you're a DevOps leader.

Technology	BSS	Application code	Application instance	Device name	Current version middleware	Target version middleware	Environment	Number of incoming stream	Number of outgoing stream	Forecast Migration Date	Migration completed date	Update
AD LEGACY	DCSI-BS-RD	AAA	RAAA		EUW	ADR	NONE	0	0	2000-01-01	2000-01-01	<input checked="" type="checkbox"/>
AD LEGACY	DCSI-BS-RD	ASQ	CASQ		EUW	ADR	NONE	0	0	2000-01-01	2000-01-01	<input checked="" type="checkbox"/>
AD LEGACY	DCSI-BS-RD	BDS	IBDS		EUW	ADR	NONE	0	0	2000-01-01	2000-01-01	<input checked="" type="checkbox"/>
AD LEGACY	DCSI-BS-RD	BDS	IBDSCARO		EUW	ADR	NONE	0	0	2000-01-01	2000-01-01	<input checked="" type="checkbox"/>
AD LEGACY	DCSI-BS-RD	BDS	RBDS		EUW	ADR	NONE	0	0	2000-01-01	2000-01-01	<input checked="" type="checkbox"/>

Repository

- Issues 0
- Jira
- Merge Requests 0
- CI / CD
- Operations
- Analytics
- Wiki**
- Snippets

On the click of the "Add values" button, you can add a new record.

O Obso-tool-client

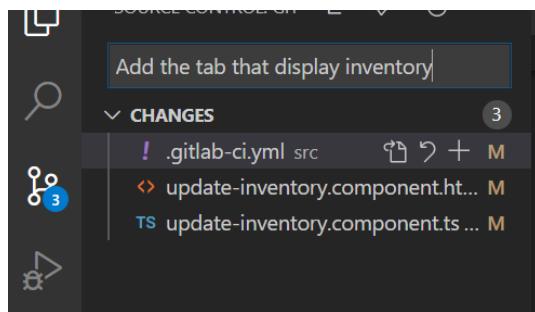
- Project overview
- Repository
- Issues 0
- Jira
- Merge Requests 0
- CI / CD
- Operations
- Analytics

On the click of the "import" button, you can choose a excel file to import. On te import, a pop-up display how many row are imported and the potential errors raises.

7. Gestion de version du projet

Afin de gérer les différentes versions du front et du back, je me suis servi de Git. Je me suis servi de Visual Studio Code pour la partie front de l'application, j'ai donc utilisé des plugins de Visual Studio Code pour pouvoir me servir de Git directement avec l'IDE.

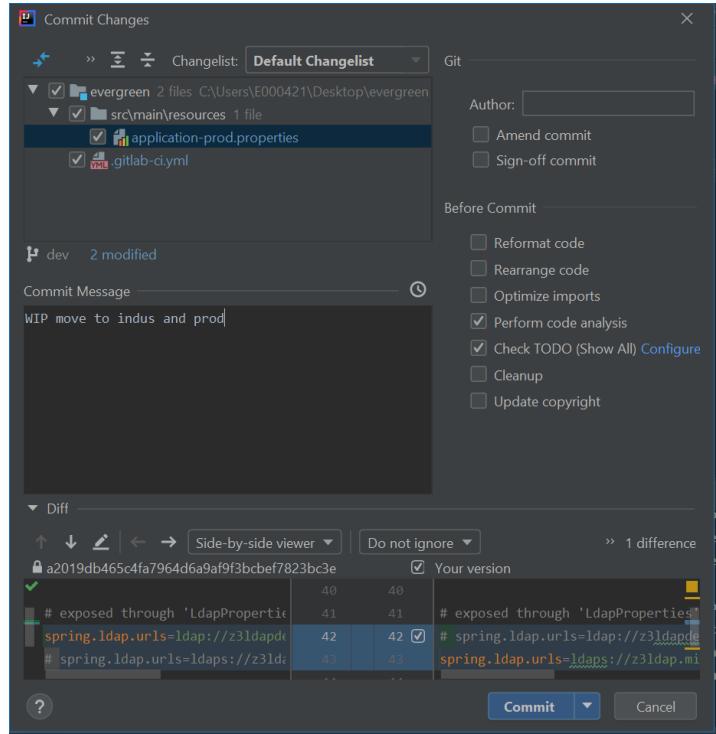
Voici un extrait d'un commit que j'étais sur le point de réaliser avec le message associé et en dessous les 3 fichiers à commiter :



Sur la capture d'écran ci-dessous, voici une liste, grâce à la commande git branch, des différentes branches que j'avais en local pour mon projet :

```
C:\Users\E000421\evergreen-front\evergreen-frontend>git branch
authenthImplementation
dev
* feature-choiceOfEnv
fixImportExcel
master
sortTableFix
teleTravail-branch
```

Pour la partie back de l'application, je me suis servi de l'IDE IntelliJ qui permet également de gérer Git directement à l'intérieur de l'IDE. Voici ci-dessous un extrait d'un commit que j'ai effectué. On peut voir le message associé au commit ainsi que les fichiers à committer dans la fenêtre du haut et dans la fenêtre du bas le delta entre les fichiers d'origine et les lignes que j'ai modifié :



8. Application annexe

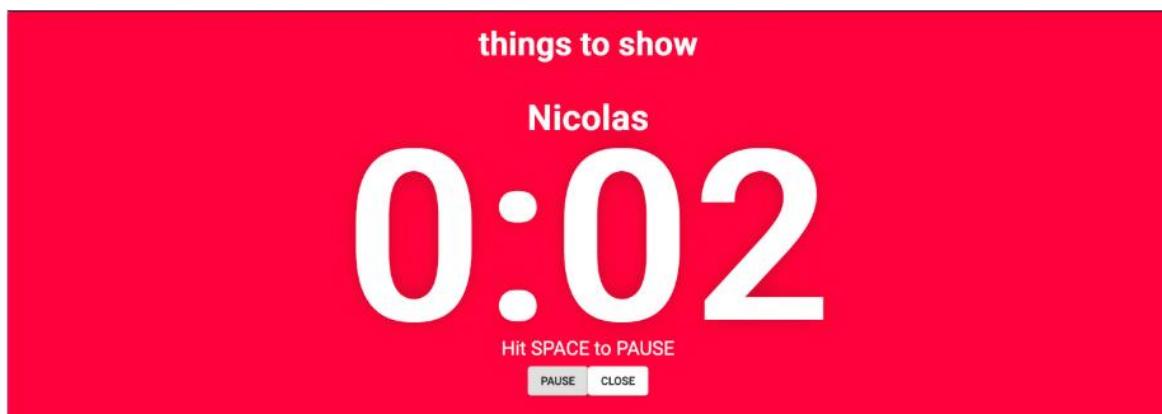
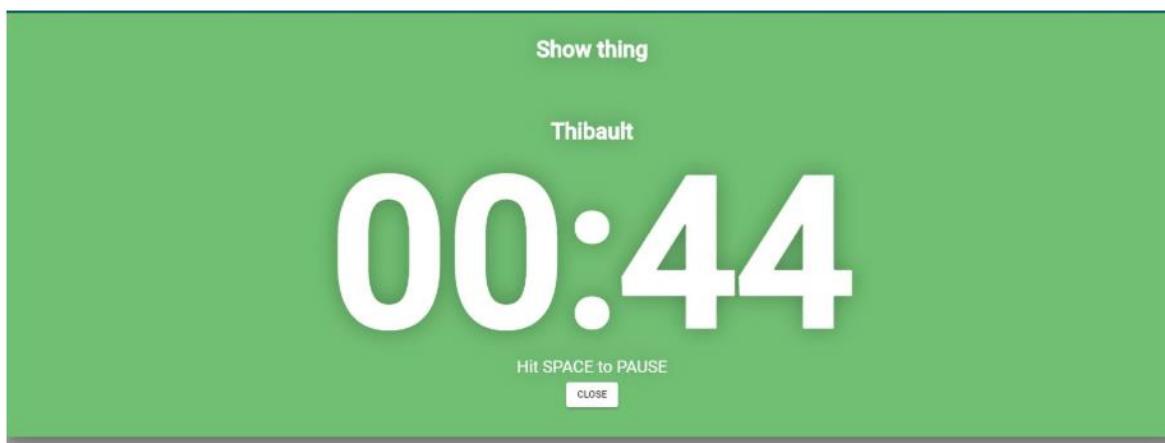
Ma pause de midi durait de 12h à 13h30. Etant donné que je mangeais sur place avec un développeur, nous avions beaucoup de temps libre pendant cette pause. Ainsi, m'étant lié d'amitié avec ce développeur, nous nous sommes mis à développer une petite application pour le service dans lequel j'ai effectué mon stage. Le responsable de service ainsi que les équipes du service dans lequel j'étais ont apprécié l'idée et l'application a été acceptée. Lors d'une réunion, l'existant actuelle de l'application a même été utilisée et tout a correctement fonctionné, les feedbacks ont été positifs.

Le but de l'application était très simple : lors des réunions et des différentes démonstrations aux clients des solutions développées, chaque membre de l'équipe se voit attribuer un certain temps de parole. Pour cela, un des membres de l'équipe se voit attribué à chaque réunion le rôle de « timekeeper », c'est-à-dire que c'est lui qui gère le chrono et qui indique lorsque le temps de parole d'un membre est terminé par exemple. Cette tâche n'étant pas vraiment très passionnante pour un développeur, nous sommes partis de ce constat pour développer une petite application interne au service pour, au préalable d'une réunion, entrer tous les membres de la réunion avec le temps attribué à chacun et pour afficher un compte à rebours en gros sur un écran afin d'indiquer le temps restant.

Ce projet a été réalisé sur GitHub, nous avons donc travaillé en collaboration à deux.

Voici, à titre d'exemple les contributions que j'ai faite sur le projet ainsi que des captures d'écrans de la petite application développée. On peut notamment y voir que l'on peut dans le tableau du bas ajouter un titre, le nom du membre qui va parler ainsi que le temps qui lui sera attribué. On peut ensuite ajouter d'autres membres avec le petit bouton « + ». Enfin, on peut lancer le compte à rebours en appuyant sur le bouton « start » dans la colonne de droite du tableau ce qui lance les compte à rebours (visible sur les 3 captures d'écran qui suivent). A savoir que nous avons choisi de mettre des indicateurs visuels : à 25% du temps restant, le fond s'affiche en rose et lorsque le temps restant est dépassé, le fond s'affiche en rouge (avec un compte à rebours inverse) :

The screenshot shows a web-based application titled "Review Timekeeper tools". At the top, there is a blue header bar with the title. Below it, a large blue button labeled "Start Review" is visible. The main area contains a table with columns for "Title", "Who", and "Time". There is also a column with a "+" sign for adding more rows. The first row has fields for "title" and "speaker". To the right of the table is a "Time" column with a value of "0" and a "Start" button. A small icon of a bee is next to the word "Subjects".





Enfin, voici quelques contributions que j'ai effectué sur le projet :

```
1 + <table mat-table [dataSource]="dataSource" matSort class="mat-elevation-z8">
2 +   <ng-container matColumnDef="title">
3 +     <th mat-header-cell *matHeaderCellDef mat-sort-header>Title</th>
4 +     <td mat-cell *matCellDef="let row">
5 +       <mat-form-field>
6 +         <input matInput type="text" name="title" placeholder="title">
7 +       </mat-form-field>
8 +     </td>
9 +   </ng-container>
10 +  <ng-container matColumnDef="who">
11 +    <th mat-header-cell *matHeaderCellDef mat-sort-header>Who</th>
12 +    <td mat-cell *matCellDef="let row">
13 +      <mat-form-field>
14 +        <input matInput type="text" name="who" placeholder="who">
15 +      </mat-form-field>
16 +    </td>
17 +  </ng-container>
18 +  <ng-container matColumnDef="time">
19 +    <th mat-header-cell *matHeaderCellDef mat-sort-header>Time</th>
20 +    <td mat-cell *matCellDef="let row">
21 +      <mat-form-field>
22 +        <input matInput type="text" name="timeLeft" placeholder="time">
23 +      </mat-form-field>
24 +    </td>
25 +  </ng-container>
26 +  <ng-container matColumnDef="create">
27 +    <th mat-header-cell *matHeaderCellDef>
28 +      <button mat-icon-button>
29 +        <mat-icon (click)="addSubject()">add</mat-icon>
30 +      </button>
31 +    </th>
32 +    <td mat-cell *matCellDef="let row">
33 +      <button mat-raised-button (click)="startTimer(row)">Start</button>
34 +    </td>
35 +  </ng-container>
```

```
86 +  addSubject() {  
87 +    let newSubject = new ReviewSubject();  
88 +    newSubject.title = '';  
89 +    newSubject.who = '';  
90 +    newSubject.time = 0;  
91 +    this.subject.push(newSubject);  
92 +    this.dataSource.data = this.subject;  
93 +  }  
25 +  
26 +  table {  
27 +    width: 100%;  
28 +  }  
29 +  
30 +  ::ng-deep.mat-sort-header-container {  
31 +    padding-left: 17px;  
32 +    justify-content: center;  
33 +  }  
34 +  
35 +  th.mat-header-cell {  
36 +    text-align: center;  
37 +  }
```