

Registro

Accédez aux informations essentielles

de votre entreprise dès la page d'accueil. Retrouvez les données clés de vos employés en un coup d'œil.



• • • • •



Formation de développeur Web, Web mobile
Stage : du 06/01 au 14/03/2025 – CCO (Villefranche de Rouergue - 12200)
Nadine Venet

Table des matières

Contexte du projet :	4
A. CCO :	4
B. Son activité :	4
Infrastructure :	4
Cloud :	5
Développement :	5
C. Le projet :	6
D. Expression des besoins :	6
E. Contraintes du projet :	7
F. Environnement humain et technique :	8
Le front-end :	9
A. Wireframes et maquettes :	9
Maquettes :	9
Wireframes :	11
B. Les écrans utilisateur :	12
Les codes couleur :	13
Les pages et la navigation entre-elles :	13
Responsive :	15
C. Extrait de code partie statique :	16
Company.jsx :	16
Composants utilitaires CCO :	20
ApiHelper.js	20
GenericController	21
D. Extrait de code partie dynamique.	22
Sélection de la société et mise à jour de l'affichage	23
Masquer et afficher le menu sur petit écran :	24
Affichage de données supplémentaires sur le survol d'éléments	25
Accès à la page des salariés	25
E. Les fixtures :	26
F. Tests fonctionnels :	28
Le back-end :	31
A. Analyse UML et définition de MCD :	31
Diagrammes de cas :	31
Modèle Conceptuel de Données :	35
Schéma Workbench :	35
B. Code composant métier	37
C. Code composant d'accès aux données :	38
D. Les tests unitaires :	40
Eléments de sécurité	42
A. Routes sécurisées	42
security.yaml	42
Blocage dans les controllers :	43
Affichage d'élément conditionné aux rôles :	45
Posture réflexive	46
Annexes	47
A. Les tables – Entités – Controller - Fixtures :	47
B. Schéma serialize / deserialize :	47
C. Méthodes des repository communes :	47
D. Extrait du dictionnaire de données :	48
E. Sauvegardes Git :	49
F. MCD et Schéma Workbench plus grands :	49

Résumé

Le stage au sein de la société **CCO** (Villefranche de Rouergue - Aveyron), prolongement de la formation continue en développement Web et Web mobile, avait pour but de mettre en pratique mes compétences théoriques tout en renforçant mes connaissances techniques notamment sur la partie front.

Crée en 2016 par Cédric Couderc, afin d'accompagner les entreprises et collectivités locales, la société CCO étend aujourd'hui son périmètre d'intervention sur plusieurs départements limitrophes. Les clients cibles sont principalement les structures de taille moyenne ayant des besoins informatiques importants (plusieurs postes de travail en réseaux, parfois sur plusieurs sites), mais très souvent pas le service informatique permettant de piloter et maintenir le tout.

CCO opère dans trois domaines principaux : l'**infrastructure** et l'infogérance, incluant la cybersécurité et la sauvegarde, le **cloud** avec des outils collaboratifs et l'hébergement, et puis le **développement**, avec la création d'ERP web plus performants à partir d'outils Excel existants.

Cédric, fondateur de CCO, assure la prospection, la relation client et participe à l'infogérance quotidienne tout en validant les développements lors des phases clés. **Camille**, présente depuis 2022, se consacre entièrement à l'infogérance, intervenant à distance et sur site chez les clients. **Céline**, recrutée en 2023, restructure l'organisation en gérant les tâches administratives, la facturation et le suivi des employés. **David**, arrivé en 2020, supervise les développements, analyse les nouveaux projets, assure leur suivi et prend en charge le développement du back-end. **Nicolas**, développeur en alternance en Master depuis deux ans, excelle dans le front-end avec React et conçoit les maquettes graphiques sur Figma.

C'est dans le cadre du développement que s'est effectué mon stage sur un projet interne de mise à disposition en ligne d'un outil créée par CCO et hébergé sur son propre cloud. L'objectif principal du stage étant de développer une solution permettant le suivi et la dématérialisation du **Registre Unique du Personne** et d'assister le responsable RH dans la gestion des actions clés liées à chaque salarié.

Ma tâche principale était de créer la structure de base (Company, User, Employee, Contract). L'objectif est de préparer le système pour l'ajout futur de modules, en veillant à ce que cette expansion soit facilitée par la conception initiale.

L'environnement technique du projet CCO repose sur des outils modernes et efficaces pour le développement web. **Visual Studio Code** sert d'éditeur principal, offrant des fonctionnalités avancées comme le débogage intégré et la gestion de version Git. Le backend utilise **Symfony**, un framework PHP robuste avec une architecture MVC, tandis que **React** est choisi pour le frontend, permettant la création d'interfaces utilisateur interactives. La gestion des dépendances est assurée par **Composer** (PHP) et **Yarn** (JavaScript), avec **Tailwind CSS** pour la mise en forme. **MySQL Workbench** est utilisé pour la gestion de la base de données, tandis que **Git** et **GitLab** assurent le contrôle de version et la collaboration.

Le mois de janvier a, quasi intégralement, été consacré à la rédaction du dossier d'étude, hybride de spécifications fonctionnelles et un cahier des charges. Il s'est terminé par la création de la structure de base, incluant les entités principales (Company, User, Employee et Contract), 9 tables de données sources (comme Country, TypeContract, Sexe ...) et les tables de jonction.

Le mois de février, a été consacré à la création des tests unitaires avec l'ajout des assertions, des wireframes, des fixtures et des tests fonctionnels. Il finit par le développement de la première page Company.jsx.

Les trois dernières semaines ont été consacrées à la finalisation des fixtures, à la création des deux autres pages principales, Profil et Salarié, ainsi qu'à la mise en place de la navigation, sans oublier la rédaction de ce rapport.

À la veille de la fin du stage, l'équipe CCO chargée du dossier est satisfaite du résultat, elle envisage la reprise du projet sereinement. Personnellement, j'ai considérablement progressé sur la partie front, qui était un point faible, tout en appréciant le contexte de travail et en prévoyant de consacrer le peu de temps restant à compléter les fonctionnalités.

Le projet proposé ainsi que l'équipe, accueillante et compétente, m'ont permis de valoriser au mieux ces 11 semaines de stage. Et donc merci à toute l'équipe CCO pour cette expérience concluante.

Contexte du projet :

Dans le cadre de la formation de développeur Web et Web mobile, j'ai eu l'opportunité de réaliser un stage au sein de la société CCO. Ce stage s'inscrit dans le prolongement de la formation continue en développement Web et Web mobile, afin de mettre en pratique les compétences théoriques acquises tout en me confrontant à des défis professionnels réels. L'objectif principal était de renforcer mes connaissances techniques et de développer des compétences pratiques dans le domaine du développement Web.

A. CCO :

Crée en 2016 par Cédric Couderc, afin d'accompagner les entreprises et collectivités locales, la société **CCO** (basée à Villefranche de Rouergue - Aveyron) étend aujourd'hui son périmètre d'intervention sur plusieurs départements limitrophes.



Les clients cibles sont principalement les structures de taille moyenne ayant des besoins informatiques importants (plusieurs postes de travail en réseaux, parfois sur plusieurs sites), mais très souvent pas de service informatique permettant de piloter et maintenir le tout.

Initialement seul, et après quelques collaborations sans suite, une équipe se forme en 2020 avec l'arrivée de David Bosc-Andrieu. Sa polyvalence, grâce à un passé d'informaticien industriel, ainsi que son appétence pour le développement en général et le développement Web en particulier en font la recrue idoine pour aider au développement de l'activité. Les années suivantes l'équipe s'est renforcée pour être composée en 2025 de 5 personnes.

L'équipe :

Cédric Couderc
Founder et PDG



Cédric assure la prospection et la relation client.

Il participe activement à l'infogérance quotidienne, et valide les développements lors des phases de prospection, de test et de mise en production.



David Bosc-Andrieu
Développeur

David supervise l'ensemble des développements.

Il est responsable de l'analyse des nouveaux projets, du suivi, de la maintenance et de l'évolution des outils développés pour les clients. De plus, il prend en charge le développement backend des projets.

Camille Eugenie
Technicien



Camille depuis son arrivée en 2022, est entièrement dédiée à l'activité d'infogérance. Elle intervient à distance et mais aussi sur site, passant plusieurs demi-journées par semaine chez certains clients.



Céline Fau
Assistante administrative

Céline est arrivée en 2023 pour restructurer l'organisation.

Elle gère l'ensemble des tâches administratives, notamment la relation client, le suivi de la facturation et la réception des appels. De plus, elle est responsable du suivi des salaires et des congés des employés de CCO.

Nicolas Calvelo
Développeur



Nicolas arrivé en France en 2022 (d'origine argentine), il a d'abord travaillé en tant que développeur chez CCO.

En alternance depuis deux ans, il poursuit un Master en Ingénierie du Web. Spécialiste du front-end grâce à sa bonne maîtrise de React, il excelle également dans le design et conçoit les maquettes (avec le logiciel Figma) définissant la charte graphique.

B. Son activité :

CCO opère dans trois domaines principaux

Infrastructure :

La gestion de l'infrastructure informatique, notamment la gestion du parc informatique, est une activité essentielle pour les entreprises modernes. Elle englobe plusieurs aspects clés.

Confiez-nous la gestion de votre parc informatique!

Nous nous engageons dans la réussite de vos projets en vous proposant des solutions personnalisées



Infogérance
Audit, conseil, installation, assistance, maintenance informatique.



Cybersécurité
Etude, sensibilisation et sécurisation de vos outils informatiques.



Vente de matériel
Ordinateurs et périphériques, équipements réseaux.



Sauvegarde et PRA
Sécurisation de vos données et mise en place de Plan de Reprise d'Activité.

Infogérance :

L'infogérance consiste à externaliser la gestion et la maintenance des systèmes d'information à un prestataire spécialisé. Elle permet aux entreprises de se concentrer sur leur cœur de métier tout en bénéficiant d'une expertise technique pointue et d'une surveillance continue de leur infrastructure IT.

L'approche de CCO combine l'infogérance à distance et sur site, offrant un service complet et personnalisé aux clients

Cybersécurité :

La cybersécurité est devenue un enjeu majeur dans la gestion de l'infrastructure informatique. Elle vise à protéger les données, les réseaux et les systèmes contre les menaces numériques croissantes. CCO assiste les entreprises afin de mettre en place des mesures de sécurité adaptées pour réduire les risques d'attaques et de violations de données.

Vente et installation de matériel :

Cette activité comprend la fourniture, l'installation et la configuration d'équipements informatiques adaptés aux besoins spécifiques des entreprises. Cela inclut les ordinateurs, serveurs, périphériques et logiciels nécessaires au bon fonctionnement de l'infrastructure.

Sauvegarde et PRA :

La sauvegarde des données et le Plan de Reprise d'Activité sont des composantes cruciales de la gestion de l'infrastructure. Ces éléments assurent la continuité des opérations en cas d'incident et permettent de restaurer rapidement les systèmes et les données en cas de besoin.

Cloud :

CCO propose une gamme complète de services cloud et informatiques pour répondre aux besoins des entreprises.

Nos solutions à votre service !

Nous nous engageons dans la réussite de vos projets en vous proposant des solutions personnalisées



Outil collaboratif

Nous vous assistons dans la mise en place et aux paramétrages des solutions Microsoft 365.



Sauvegarde Externalisée

Nous proposons des sauvegardes de vos serveurs et poste de travail sur un cloud de proximité.



Hébergement de solutions

Mise en place d'ERP Open Source : intégration et personnalisation.



Hébergement site internet

Fourniture du nom de domaine et paramétrage du serveur pour votre site internet.

Outil collaboratif :

Définition d'une solution cloud sur mesure (Microsoft 365), avec une approche personnalisée permettant de répondre précisément aux exigences spécifiques de chaque entreprise.

Sauvegarde Externalisée :

Avec des services de sauvegarde pour les serveurs et les postes de travail, utilisant un cloud de proximité, cette approche garantit la sécurité des données et une récupération rapide en cas de besoin.

Hébergement de solutions :

Basé sur des services d'intégration et de personnalisation d'ERP Open Source. Cette solution permet aux clients de bénéficier de systèmes de gestion d'entreprise flexibles et adaptés à leurs besoins.

Hébergement site internet :

Grâce à des services complets pour l'hébergement de sites web, incluant l'acquisition de noms de domaine et la configuration des serveurs, cela permet aux clients de lancer et de gérer facilement leur présence en ligne

Développement :

De façon historique, les premiers développements de CCO visaient à réécrire et améliorer des outils, souvent Excel créés par les clients mais devenus limités. Avec l'acquisition de nouvelles compétences en développement web, ces outils ont naturellement évolué vers des ERP web plus performants.

CCO propose désormais des services de développement logiciel sur mesure, adaptés aux besoins spécifiques des entreprises.

CATIONS WEB PAGE POWER AUTOMATE SQL

Votre logiciel, créé avec vous et pour vous

Nous nous engageons dans la réussite de vos projets en vous proposant des solutions personnalisées



Demande
Besoin spécifique ? Fichier Excel dépassé ? Interconnexion difficile?



Analyse/ Etude
Cahier des charges, Analyse des données. Proposition fonctionnelle.



Développement
Agile, précis et adaptatif.



Mise en service
Logiciel adapté à votre métier. Gain de productivité. Réduction des coûts.

Demande

CCO répond aux besoins spécifiques des entreprises, qu'il s'agisse de remplacer un fichier Excel obsolète ou de résoudre des problèmes d'interconnexion entre logiciels existants.

Analyse et Étude

L'équipe élabore un cahier des charges détaillé, analyse les données et propose une solution fonctionnelle adaptée aux exigences du client.

Développement

Grâce à une méthodologie agile, le développement est précis, flexible et adaptatif. Cette approche itérative assure une collaboration étroite avec le client et une capacité à répondre rapidement aux changements.

Mise en Service

Le logiciel déployé est entièrement adapté au métier du client, permettant un gain significatif de productivité et une réduction des coûts. Un support permanent est également assuré pour répondre aux questions et accompagner les évolutions.

C'est dans le cadre de cette dernière activité que s'est effectué mon stage, non pas sur une demande client mais sur un projet interne de mise à disposition en ligne d'un outil créé par CCO et hébergé sur son propre cloud.

C. Le projet :

À la suite d'une collaboration avec une consultante RH et de multiples échanges afin de créer et adapter un outil Excel visant à gérer l'embauche et le suivi du personnel, a germé l'idée d'une solution en ligne qui permettrait d'assister le responsable des ressources humaines dans la gestion des actions clés liées à chaque salarié, comme :

1. Planification et suivi des entretiens
2. Gestion des visites médicales
3. Suivi des formations
4. Gestion des permis spécifiques (comme les CACES)
5. Suivi des accréditations

Ce concept s'inscrit dans la tendance actuelle des solutions RH intégrées, qui visent à centraliser et automatiser les processus RH répétitifs, grâce à une solution plus robuste et évolutive pour la gestion de leurs ressources humaines.



Le logiciel sera en mode SaaS

Hébergement sur le serveur CCO
Le client se connectera sur le serveur de CCO

1 seule Base de Données (BdD).
1 application commune à tous les clients.

Infrastructure-aaS - Platform-aaS - Software-aaS

Sa spécificité, la proximité (hébergement chez CCO) et une cible limitée aux entreprises de moins de 10 salariés (du moins au départ).

D. Expression des besoins :

L'objectif principal du stage sera de développer une solution permettant le suivi et la dématérialisation du **Registre Unique du Personnel**. Dans un second temps, le projet évoluera pour intégrer des modules additionnels assurant le suivi des actions clés RH mentionnées précédemment.

Après une demande d'accès au logiciel, le premier utilisateur de la société reçoit un jeton (CompanyOnboarding) qui lui permet de créer son compte (User) ainsi qu'une fiche société (Company).

Avec la page de connexion, ce sera la seule route qui ne nécessitera pas d'être connecté.

Toutes les autres opérations comme, la création des utilisateurs (personnes habilitées à saisir et/ou consulter les données) et toutes les saisies nécessiteront d'être connecté.

L'accès sera :

- **Multi-utilisateurs** : une société pourra disposer de plusieurs utilisateurs, chacun avec un rôle spécifique (cf. ci-dessous).
- **Multi-sociétés** : un utilisateur pourra être associé à plusieurs sociétés, avec des rôles potentiellement différents dans chacune.
- **Restraint** : chaque client aura uniquement accès aux données des sociétés auxquelles il est rattaché.

Reste ensuite le cas particulier des utilisateurs CCO qui pourront n'être rattachés à aucune société. Ils auront accès à toutes les données afin d'effectuer les tâches de support et d'administration.

Il y aura donc 2 niveaux de rôle :

➤ **Premier niveau :**

Il permet d'autoriser ou interdire toutes les fonctions d'administration et de gestion dédiées à CCO.

- **ADMIN_CCO** : CCO. Aucune restriction. Voit tout et a accès à toutes les données
- **ROLE_GESTION** : CCO. Pourra envoyer le lien au futur client.
 - Activation ou désactivation d'un compte ou d'une société.
 - Pourra consulter toutes les données.
- **ROLE_CLIENT** : non CCO
 - Seules les données liées à la société seront accessibles.
 - Les droits seront définis par le rôle de second niveau, ci-dessous, qui sera rattaché à la société.

➤ **Second niveau :**

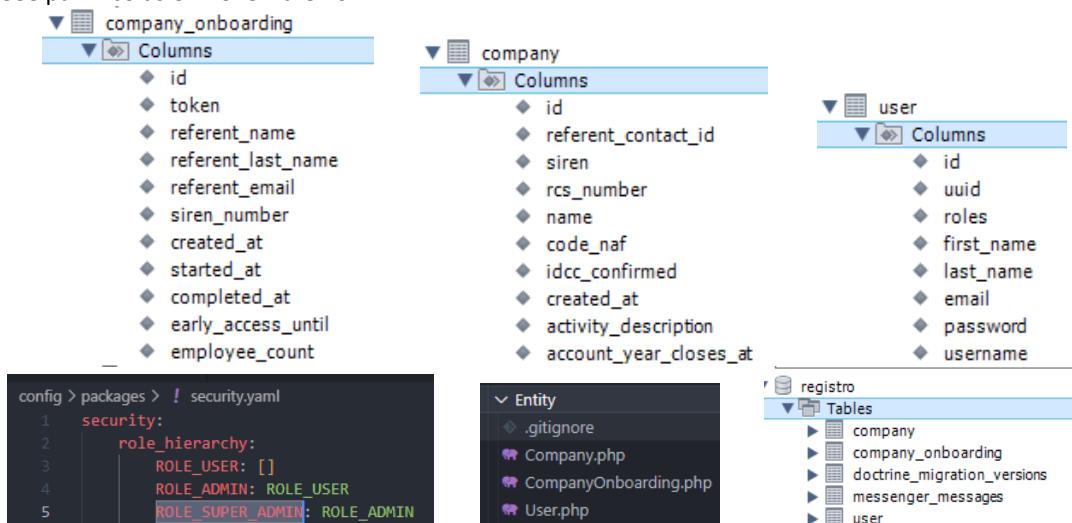
Ces rôles sont liés à l'utilisateur au sein de la société et définissent les droits sur chaque module.

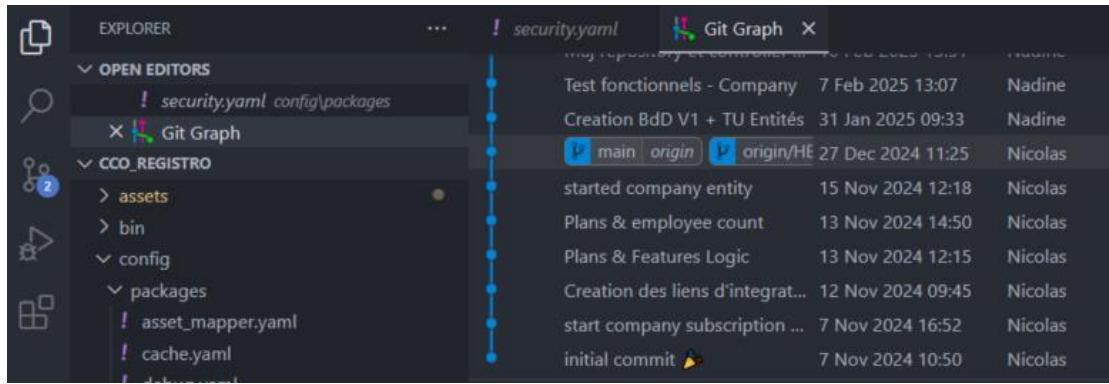
- **ROLE_CLIENT_RESP** : Il reçoit le jeton pour créer une société. Il peut tout faire au sein de la société.
- **ROLE_CLIENT_RH** : seule la création de la société et celle des utilisateurs avec le rôle responsable lui sont interdites.
- **ROLE_CLIENT_COMPTA** : Consultation uniquement.

E. Contraintes du projet :

A mon arrivée le projet était déjà initié depuis fin 2024, mais faute de disponibilité et de temps à consacrer à l'analyse, il était en standby.

Structure créée par Nicolas en novembre 2024 :





Ma tâche principale était de créer la structure de base (Company, User, Employee, Contract) sans délai précis. L'objectif était de préparer le système pour l'ajout futur de modules, en veillant à ce que cette expansion soit facilitée par la conception initiale.

F. Environnement humain et technique :

Humain :

Je vais travailler en étroite relation avec David et Nicolas qui, à mon arrivée, m'ont présenté le stade d'avancement du projet et vont m'apporter une précieuse aide technique tout au long du stage.

Je reste en totale autonomie tout en ayant la possibilité de faire appel à eux à tout moment.

Technique :

L'environnement technique de notre projet sera aligné sur celui mis en place par CCO pour l'ensemble de ses projets, garantissant ainsi une cohérence et une efficacité optimales dans le développement et la gestion du projet.

Éditeur de code :

Visual Studio Code (VS Code) est utilisé comme environnement de développement intégré (IDE) principal pour le codage.

VS Code offre des fonctionnalités avancées telles que :

- Un débogueur intégré.
- Un terminal intégré pour exécuter des commandes.
- Une gestion efficace du contrôle de version avec Git.
- Des extensions pour améliorer la productivité

Framework backend :

Symfony est choisi comme framework PHP pour le développement backend. Il offre :

- Une architecture MVC (Modèle-Vue-Contrôleur).
- Des composants réutilisables.
- Une gestion efficace des dépendances.

Symfony utilise **Doctrine** comme ORM (Object-Relational Mapping) pour interagir avec la base de données, permettant une gestion simplifiée du schéma et des requêtes.

Framework frontend :

React est choisi pour la création des pages et composants frontend. Cette librairie JavaScript permet de construire des interfaces utilisateur interactives et (comme son nom l'indique) réactives.

Gestion des dépendances :

- **Composer** est utilisé pour gérer les dépendances PHP côté backend. Il permet d'installer et de mettre à jour facilement les packages PHP nécessaires au projet.
- **Yarn** est employé pour gérer les dépendances JavaScript côté frontend. Il offre des performances optimisées pour le téléchargement et l'installation des packages.

Framework CSS :

Tailwind CSS est utilisé pour la mise en forme CSS. Ce framework utilitaire permet de :

- Développer rapidement des interfaces utilisateur.
- Personnaliser facilement le design.
- Optimiser les performances avec son approche "utility-first".

Base de données :

MySQL Workbench est utilisé comme outil de gestion et de conception de la base de données. Il offre une interface graphique pour :

- Concevoir et modéliser la structure de la base de données.
- Exécuter des requêtes SQL.
- Administrer le serveur MySQL.

C'est un environnement de développement moderne, flexible et efficace pour la création d'applications web robustes et performantes.

Gestion des versions :

Git est utilisé comme système de contrôle de version distribué. Il offre plusieurs avantages :

- Suivi complet de l'historique des modifications du code
- Facilitation du travail collaboratif
- Possibilité de créer des branches pour développer des fonctionnalités en parallèle
- Capacité à revenir à des versions antérieures du code

GitLab est choisi comme plateforme DevOps intégrée. Elle offre les fonctionnalités suivantes :

- Hébergement de dépôts Git
- Gestion de projet avec tableaux Kanban
- Intégration continue et déploiement continu (CI/CD)
- Suivi des problèmes (issues)
- Wiki pour la documentation du projet

Ces outils s'intègrent parfaitement avec les autres éléments de l'environnement technique déjà mentionnés, offrant ainsi une solution complète pour le développement, la collaboration et le déploiement d'applications web modernes.

Le front-end :

A. Wireframes et maquettes :

La charte graphique ainsi que l'enchaînement de certaines pages avaient déjà été définis par Nicolas qui a réalisé, avec Figma, quelques maquettes permettant de définir les écrans ainsi que l'enchaînement de ceux-ci notamment lors de la création de nouvelles données.

Maquettes:

Utilisation du jeton (l'implémentation de cette partie est largement réalisée aussi).



Ci-contre les maquettes de l'utilisation du jeton qui permettra au client de créer le compte d'accès à l'application

Nous retrouverons le formalisme suivant sur tous les écrans.

- Le quart gauche de la page dédié à l'affichage d'une barre d'avancée de l'action ou d'un menu des actions possibles
- Et le reste de la page contenant les zones de saisie ou l'affichage des données.

Pour les écrans de création, CCO a opté pour une succession de pages respectant des étapes de saisie.

Maquette de la page Société :

	2023	2022	2021	2020
Chiffre d'affaires (€)	456k	352k	293k	180k
Nombre d'employés	58	50	40	30
EBITDA - EBM (€)	120k	26,4k	32,0k	21k
Résultat exploitation (€)	63k	40,7k	28,0k	13,9k
Capital social (€)	1 000,00	1 000,00	1 000,00	1 000,00
Croissance	2023	2022	2021	2020
Taux de croissance du CA (%)	29,4	39,2	61,6	20,8
Taux de marge brute (%)	47,9	43,7	54,8	49,3

Ajout d'un salarié :

Plutôt qu'un formulaire global, on retrouve ici, la saisie des données par étapes.

Pour ma part, après avoir élaboré une première version de la structure des données, j'ai réalisé avec Figma des wireframes afin de proposer l'agencement des écrans des Sociétés, des Utilisateurs et des Salariés.

Wireframes.:

Comme défini dans les maquettes ci-dessus, les pages seront composées d'une partie gauche affichant un menu et d'une partie droite destinée à afficher les données.

J'ai d'abord travaillé sur le tableau de bord et les menus puis organisé un point afin de valider la structure de la base de données et de définir le contenu de chaque page.

Tableau de bord :

Société :

Utilisateur :

Contrats :

Proposition d'adaptation de l'affichage des pages en fonction de la taille des écrans.



Multi société : 2 propositions

Soit une page avec un accès à chaque société, soit plus simplement une liste déroulante contenant toutes les sociétés du User et permettant de switcher de l'une à l'autre.

C'est la seconde option qui a été retenue.

multi Sociétés : 2 options ?



Code couleur : pour identifier le rôle de l'utilisateur connecté.

Identification rapide des rôles par un code couleur ?



B. Les écrans utilisateur :

Comme toutes les routes fonctionnelles sont protégées et exigent une connexion, la page de login s'affiche au démarrage.

Bienvenue

Identifiant	<input type="text" value="responsable@soci.fr"/>
Mot de passe	<input type="password" value="*****"/>
<input checked="" type="checkbox"/> Se souvenir de moi	
Se connecter	

Ou

Se connecter via microsoft

Les codes couleur :

Associés à la vignette du User, ils sont fort heureusement plus sobres.

User

Result Grid			Filter Rows:			Edit:	Export/Import:	Wrap
			first_name	last_name	email	pi		
▶	500	1	["ROLE_SUPER_ADMIN"]	CCO	Dev	support.dev@cco-info.fr	\$2	
	501	2	["ROLE_SUPER_ADMIN"]	Nicolas	Calvelo	nicolas@cco-info.fr	\$2	
	502	3	["ROLE_GESTION"]	Gestion	CCO	gestion@cco-info.fr	\$2	
	503	Soc1_Resp	["ROLE_CLIENT"]	Resp	Soc 1	responsable@soc1.fr	\$2	
	504	Soc1_Rh	["ROLE_CLIENT"]	Rh	Soc 1	rh@soc1.fr	\$2	
	505	Soc2_Resp	["ROLE_CLIENT"]	Resp	Soc 2	responsable@soc2.fr	\$2	
	506	Soc1_Compta	["ROLE_CLIENT"]	Compta	Soc 1	compta@soc1.fr	\$2	
	NUL	NUL	NUL	NUL	NUL	NUL	NUL	

CompUser					
▶	82	187	503	NULL	["ROLE_CLIENT_RESP"]
	83	187	504	NULL	["ROLE_CLIENT_RH"]
	84	187	506	NULL	["ROLE_CLIENT_COMPTA"]
	85	188	505	NULL	["ROLE_CLIENT_RESP"]
	86	188	503	NULL	["ROLE_CLIENT_RH"]
	87	190	503	NULL	["ROLE_CLIENT_RESP"]

CCO		CLIENT		
ADMIN	GESTION	RESPONSABLE	RH	COMPTA

Les pages et la navigation entre-elles :

La fiche société :

Le tableau de bord n'ayant été défini, pour l'instant on arrive sur la page d'accueil de la société

On y retrouve :

- à gauche un menu déroulant qui permet à l'utilisateur connecté de switcher sur les différentes sociétés
- à droite les données de la société avec :
 - sa fiche et, des compteurs
 - en dessous les liste des tous les salariés actifs, avec des données du contrat en cours.
 - en dessous toutes les fiches de poste actives de la société.

The screenshot shows the 'Registro' dashboard for 'Soc 1'. On the left, there's a sidebar with a dropdown for 'Selectionnez votre entreprise:' containing 'Soc 1', 'Soc 2', 'Soc 4', and 'firstName'. The main area displays company information: 'SIREN : 123 456 789', 'Date de création : 19/02/2000', and 'Mois de clôture : 1 (Janvier)'. It also shows '3 Salaris', 'Contrat', and '4 Fiches de poste'. Below this, there are tables for 'Les Salariés' (listing employees with details like name, first name, civility, entry date, contract date, end date, role, start date, hire date, and nationality) and 'Les fiches de poste' (listing job titles with descriptions and responsibilities). A tooltip for the user icon in the top right corner shows options for 'Voir profil' and 'Déconnexion'.

Au survol de la pastille avec les initiales de l'utilisateur connecté, un menu apparaît. Il propose d'accéder au profil de l'utilisateur et de se déconnecter



Le profil utilisateur :

The screenshot shows the Registro application interface. On the left, there's a sidebar with a search bar and a 'SR' logo. The main area has tabs for 'Profil' and 'Votre fiche'. Under 'Votre fiche', it shows 'Responsable Soc 1' with a date of last update. Below that is a section for 'Détail de la fiche' with fields for Nom, Prénom, Email, and UUID. To the right is a table titled 'Sociétés et Rôles' with columns for RESP, RH, and CPT, listing roles assigned to Soc 1, Soc 2, Soc 4, and a placeholder 'firstNameP'.

Très simple pour l'instant, il affiche les données du User et un tableau des rôles par société.

Toujours dans la page société, l'accès à la fiche d'un salarié est possible en effectuant un double clic sur sa ligne dans le tableau.

Les Salariés

Nom	Prénom	Civilité	D'entrée	Date Du contrat	Fin	Re-gistre	Poste	Début	Né(e) le	Natio-nalité	...
Soc 1	Salarié 1	Mme	20/05/2024	21/11/2024		1	Comptable	21/11/2024	01/01/2000	FRA	...
Soc 1	Salarié 2	Mme	01/01/2008	17/05/2010		1	Analyste programmeur	01/09/2011	02/01/1985	FRA	...
Soc 1	Salarié 3	M	23/01/2021	23/01/2021		1	Développeur	23/01/2021	03/01/1990	FRA	...

La fiche du salarié :

The screenshot shows the Registro application interface. On the left, there's a sidebar with a search bar and a 'SR' logo. The main area has tabs for 'Détail' and 'La fiche salarié'. Under 'La fiche salarié', it shows 'Salarié 2' with a date of last update. Below that is a section for 'Détail de la fiche' with fields for Nom, Prénom, Nationalité, and N° Sécurité soc.. To the right is a table titled 'Les contrats' showing three contracts with columns for Code, Reg., Début, Fin, Poste(s), Début-Fin, and Signature.

On y retrouve les données du salarié et des compteurs.

3 Contrats

Ancienneté

15 ans

1 autre période :
8 mois

Et la liste de tous ses contrats avec le détail des postes occupés.

Les contrats

Code	Reg.	Début	Fin	Poste(s)	Début-Fin	Signature
1 CDI	1	17/05/2010		Analyste programmeur Développeur	01/09/2011 - 17/05/2010 - 31/08/2011	17/05/2010
2 Temporaire	1	10/03/2010	16/05/2010	Développeur	10/03/2010 - 16/05/2010	09/02/2010
3 Temporaire	1	01/01/2008	01/09/2008	Développeur	01/01/2008 - 01/09/2008	01/01/2008

Retour vers la page Société :

Dans la fiche du salarié et dans la page profil, un simple lien permet de revenir à l'écran de la société.



Retour vers la fiche société

Ci-dessous l'affichage adaptatif (responsive design) sur écran des salariés

Responsive:

Sur petit écran, le menu de gauche disparait (un bouton permet de l'afficher et le masquer) et les blocs se positionnent les uns en dessous des autres. Les éléments de la fiche salariés se mettent en colonne.

The screenshot shows the application running on a mobile device in portrait mode. The sidebar on the left is collapsed. The main content area displays the employee's details in a single column, followed by the contract table, and then the古enneté information.

Code	Reg.	Début	Fin	Poste(s)	Début-Fin	Signature
1	CDI	17/05/2010	-	Analyste programmeur	01/09/2011 - 17/05/2010	-
2	Temporaire 1	10/03/2010	16/05/2010	Développeur	31/08/2011 - 10/03/2010	-
3	Temporaire 1	01/01/2008	01/09/2008	Développeur	-	09/02/2010 - 01/01/2008

Les contrats

Ancienneté
15 ans
1 autre période : 8 mois

Dès le format tablette paysage, on retrouve l'affichage classique. Il faudra voir pour, surement, masquer certaines colonnes du tableau des contrats.

The screenshot shows the application running on a tablet in landscape mode. The sidebar on the left is visible. The main content area displays the employee's details and contracts in a standard grid layout.

Code	Reg.	Début	Fin	Poste(s)	Début-Fin	Signature
1	CDI	17/05/2010	-	Analyste programmeur	01/09/2011 - 17/05/2010	-
2	Temporaire 1	10/03/2010	16/05/2010	Développeur	31/08/2011 - 10/03/2010	-
3	Temporaire 1	01/01/2008	01/09/2008	Développeur	-	09/02/2010 - 01/01/2008

Les contrats

Ancienneté
15 ans
1 autre période : 8 mois

Format PC.

The screenshot shows the application running on a desktop computer. The sidebar on the left is visible. The main content area displays the employee's details and contracts in a standard grid layout.

Code	Reg.	Début	Fin	Poste(s)	Début-Fin	Signature
1	CDI	17/05/2010	-	Analyste programmeur	01/09/2011 - 17/05/2010	-
2	Temporaire 1	10/03/2010	16/05/2010	Développeur	31/08/2011 - 10/03/2010	-
3	Temporaire 1	01/01/2008	01/09/2008	Développeur	-	09/02/2010 - 01/01/2008

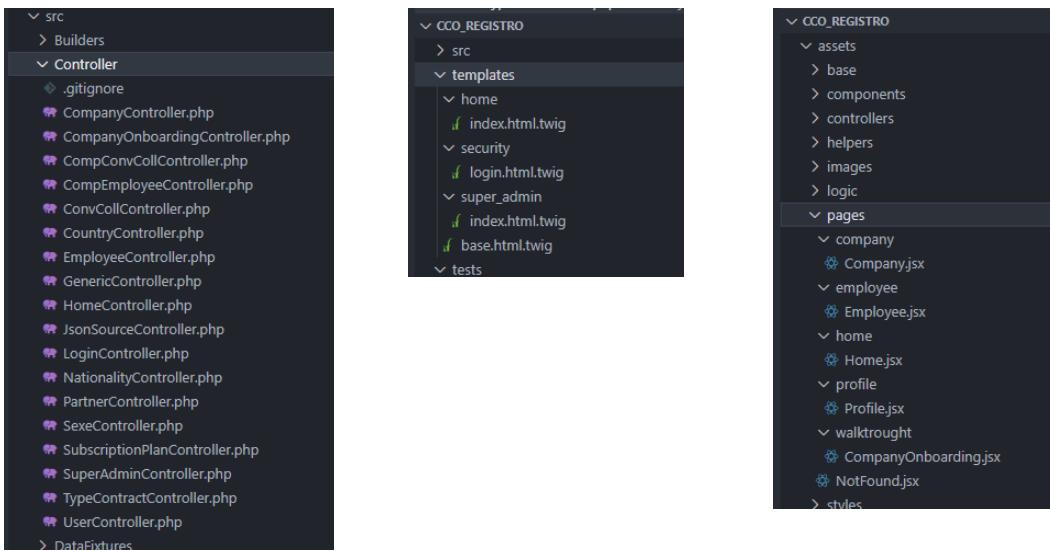
Les contrats

Ancienneté
15 ans
1 autre période : 8 mois

C. Extrait de code partie statique :

CCO a pour habitude de mettre en place des API pour récupérer les données ainsi que pour les modifier. Ces API sont appelées dans les pages javascript et elles permettent de réaliser toutes les opérations.

Et donc, dès la création des contrôleurs, on supprime les fichiers twig associés car inutiles.



Company.jsx :

Dans le controller, il faut créer une route pour chaque action exécutée dans le formulaire (affichage, mise à jour..).

Récupération des infos :

```
assets > pages > company > Company.jsx > ...
1 import { useEffect, useState } from "react";
2 import { methodGet, methodPatch } from "../../helpers/ApiHelper";
3 import Structure from "../../components/navigation/Structure";
4
5 // Accès à La page Employee
6 import { useNavigate } from "react-router-dom";
7
8 // import FloatingActionButton from "../../base/Buttons/FloatingA
```

Import du *hook* React, useState(), qui permet de gérer un état local dans un composant fonctionnel.

```
10 export default function Company() {
11   const [search, setSearch] = useState("");
12   const [userConnect, setUserConnect] = useState();
13   const [compUsers, setCompUsers] = useState();
14   const [compConnect, setCompConnect] = useState();
15
16   const [employees, setEmployees] = useState();
17   const [compteurs, setCompteurs] = useState();
18   const [jobsheets, setJobsheets] = useState();
19
20   // Modification de la cie sélectionnée dans le select
21   const [selectedCompanyId, setSelectedCompanyId] = useState(null);
22 }
```

Il permet la déclaration de variable d'état et de la fonction assurant sa mise à jour.

Par exemple :

[compUsers, setCompUsers]

```
43 useEffect(() => {
44   if (userConnect && userConnect.activeComp) {
45     setSelectedCompanyId(userConnect.activeComp);
46
47     // fetch CompUser
48     methodGet("/company/get_all_companies_uconnect")
49       .then((res) => [
50         //console.log(res);
51         // mettre à jour la variable
52         setCompUsers(res);
53       ])
54       .catch((err) => {
55         // gerer l'erreur           console.log(err);
56         console.log(err.errorMessage);
57       });
58   }
59 }, [userConnect]);
```

La variable d'état est mise à jour par ce code. Le premier argument est le code à exécuter ici methodGet

La méthode GET fait appel à une route définie dans le contrôleur

Ici

```
#Route('/get_all_companies_uconnect',
      name: "get_all_companies_uconnect",
      methods: [Request::METHOD_GET])
```

Et le second qui est tableau de dépendances qui conditionne l'exécution de l'effet.

Ici exécution lorsque **UserConnect** est défini et change

```

src > Controller > CompanyController.php > CompanyController
1  <?php
2
3  namespace App\Controller;
4
5  use App\Entity\Company;
6
7  use App\Repository\CompUserRepository;
8
9
10 #[Route('/api/company')]
11 #[IsGranted('ROLE_USER')]
12 class CompanyController extends GenericController
13 {
14
15     /**
16      * Routes ouvertes à tous mais sécurisées par rapport au User connecté
17      */
18
19     /**
20      * Permet récupérer les company de l'utilisateur connecté
21      */
22
23     #[Route('/get_all_companies_uconnect', name: "get_all_companies_uconnect", methods: [Request::METHOD_GET])]
24     public function getAllCompany(Request $request, CompUserRepository $compUserRepository): JsonResponse
25     {
26
27         /**
28          * @var User $user
29          */
30         $user = $this->getUser();
31
32         if (count($request->query->all()) === 0){
33             $compUsers = $compUserRepository->searchAllCompUser(null, $user->getId());
34             //dump ("sans Disabled");
35
36         } else if ($request->query->has('xorDisabled') && $request->query->get('xorDisabled')){
37             // Tous Les Users inactifs => xorDisabled True : on appelle la méthode avec id du User et xorDisabled
38             $compUsers = $compUserRepository->searchAllCompUser(null, $user->getId(), true, null, null, null);
39             //dump("ICI");
40
41         } else {
42             //dump ("Avec Disabled");
43
44         }
45
46
47         return $this->json($this->serialize($compUsers, ["get_company", "get_user_delete"]));
48     }
49

```

Dans la route du controller

```

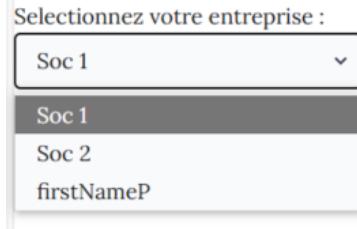
#[Route('/get_all_companies_uconnect',
name: "get_all_companies_uconnect",
methods: [Request::METHOD_GET]])

```

On fait appel à la méthode **searchAllCompUser** du **CompUserRepository** qui retourne la liste de tous les **CompUser** correspondants aux critères

La route renvoie un json qui est exploité dans le code javascript

Ci-contre, le code jsx qui à partir du contenu de la variable **compUsers**, construit la liste déroulante.



```


{compUsers && compUsers.length > 1 ? (
  <select
    className="appearance-none w-full bg-gray-50 border border-gray-300 rounded px-4 py-2 pr-8"
    // Modification de la Cie sélectionnée dans le select :
    onChange={handleCompanyChange}
    value={selectedCompanyId || ""}
  >
    {compUsers !== undefined &&
      compUsers.map((compUser) => (
        <option
          key={compUser.company.id}
          value={compUser.company.id}
        >
          {compUser.company.morale
            ? compUser.company.nameM
            : `${compUser.company.lastNameP} ${compUser.company.firstNameP}`.trim()
          }
        </option>
      ))
    }
  </select>
)


```

On retrouve le même fonctionnement pour l'affichage des données de tous les éléments de la page

	Variable et fonction d'état	Route du contrôleur	Méthode du repository
Fiche société	[compConnect, setCompConnect] methodGet('/company/\${selectedCompanyId})	#[Route('/{id}', name: "get_company_by_id", methods: [Request::METHOD_GET], requirements: ["id" => Requirement::POSITIVE_INT])]	\$company = \$companyRep->find(\$id);
Liste des salariés	[employees, setEmployees] methodGet('/company/search_all_comp_empl')	#[Route('/search_all_comp_empl', name: "comp_search_all_comp_empl", methods: [Request::METHOD_GET])]	\$results = \$compEmplRep->searchAllCompEmployee(\$idCie);
Liste des fiches de poste	[jobsheets, setJobsheets] methodGet('/company/search_all_comp_jobsheet')	#[Route('/search_all_comp_jobsheet', name: "comp_search_all_comp_jobsheet", methods: [Request::METHOD_GET])]	\$results = \$jobsheetRep->searchAllJobsheet(\$idCie);
Les compteurs	[compteurs, setCompteurs] methodGet('/company/compteurs')	#[Route('/compteurs', name: "comp_compteurs", methods: [Request::METHOD_GET])]	\$emplActifs = \$compEmplRep->countAllCompEmployee(\$idCie) \$result = \$jobsheetRep->searchAllJobsheet(\$idCie); \$jobActifs = count(\$result);

Ainsi que pour les compteurs détaillés ci-dessous.

Affichage des compteurs :

Seule particularité ici, on construit une réponse dans le contrôleur, à partir des retours de l'appel à plusieurs méthodes de plusieurs repository.

Ici on appelle 2 méthodes de 2 repository car les données sont issues de 2 entités : **countAllCompEmployee** (EmployeeRepository) et **searchAllJobsheet** (JobsheetRepository)

Chacune est appelée 2 fois avec des paramètres différents afin de récupérer d'abord les enregistrements actifs et ensuite les enregistrements inactifs.

```
193     #[Route('/compteurs', name: "comp_compteurs", methods: [Request::METHOD_GET])]
194     public function compCompteurs(Request $request, CompEmployeeRepository $compEmplRep, JobsheetRepository $jobsheetrep):
195     {
196         /* @var User $user */
197         $user = $this->getUser();
198         $idCie = $user->getActiveComp();
199
200         // Salariés =====
201         // Tous les actifs => Aucun paramètre : on appelle la méthode avec uniquement id de la Company
202         $emplActifs = $compEmplRep->countAllCompEmployee($idCie);
203         // Tous les inactifs => Ajout de $xorDisabled avec le disabled de chaque entité à null
204         $emplInactifs = $compEmplRep->countAllCompEmployee($idCie, null, true, null, null);
205
206         // Jobsheet =====
207         // Toutes les fiches de poste actives
208         $result = $jobsheetrep->searchAllJobsheet($idCie);
209         $jobActifs = count($result);
210         // Toutes les fiches de poste inactives
211         $result = $jobsheetrep->searchAllJobsheet($idCie, null, true, null, null);
212         $jobInactifs = count($result);
```

Ensuite, on construit un tableau associatif qui est renvoyé et récupéré dans la page dans la variable **compteurs**.

```
214     $results = [
215         'emplActifs' => $emplActifs,
216         'emplInactifs' => $emplInactifs,
217         'jobActifs' => $jobActifs,
218         'jobInactifs' => $jobInactifs
219     ];
220
221     return new JsonResponse(json_encode($results));
222 }
```

On exploite cette dernière dans le fichier jsx pour afficher les données

```
<div id="lesChiffres" className="w-1/4 pl-4 border-l border-gray-300" >
  <div id="emplCpt" className="bg-gray-200 rounded-lg py-2 px-4 mt-2" >
    <p id="emplActifs" className="flex items-center justify-center" >
      <svg xmlns="http://www.w3.org/2000/svg" >
        <span className="font-bold mx-2" > {compteurs.emplActifs ?? ""} </span>
        Salarié { (compteurs.emplActifs ?? 0) > 1 ? "s" : ""} </p>
    <p id="emplInactifs" className="text-center italic text-xxs mt-1" >
      {compteurs.emplInactifs > 0 ? (
        <>
        | {compteurs.emplInactifs} inactif{compteurs.emplInactifs > 1 ? "s" : ""}
      ) : null}
    </p>
  </div>

  <div...>
  </div>

  <div id="posteCpt" className="bg-gray-200 rounded-lg py-2 px-4 mt-2" >
    <p id="posteActifs" className="flex items-center justify-center" >
      <svg xmlns="http://www.w3.org/2000/svg" >
        <span className="font-bold mx-2" > {compteurs.jobActifs ?? ""} </span>
        Fiche{ (compteurs.jobActifs ?? 0) > 1 ? "s" : null} de poste
    </p>
    <p id="emplInactifs" className="text-center italic text-xxs mt-1" >
      {compteurs.jobInactifs > 0 ? (
        <>
        | {compteurs.jobInactifs} inactif{compteurs.jobInactifs > 1 ? "s" : null}
      ) : null}
    </p>
  </div>
  <div className="flex mx-2 my-3" ...>
  </div>
</div>
```



La barre de navigation :

Pour cette partie commune à toutes les pages, nous avons créé un composant qui est appelé dans chaque page.



Dans chaque page :

```
assets > pages > company > Company.jsx > Company
1 import { useEffect, useState } from "react";
2 import { methodGet, methodPatch } from "../../helpers/ApiHelper";
3 import Structure from "../../components/navigation/Structure";
```

On importe le composant Structure

```
215
216   return (
217     <Structure
218       compUsers={compUsers}
219       compConnect={compConnect}
220       selectedCompanyId={selectedCompanyId}
221     >
222       <div id="blocPage" className="flex h-full">
223         <div id="blocMenu" className="bg-white w-64 flex-shrink-0">...
224       </div>
225       <div id="blocRight" className="bg-gray-100 w-full p-5 h-full">...
226     </Structure>
227   );
228 }
```

Et on le positionne dans le return de la page en cours

Dans le composant Structure :

```
assets > components > navigation > Structure.jsx > Structure
1 import { useEffect, useState } from "react";
2 import { methodGet } from "../../helpers/ApiHelper";
3
4 // import { useNavigate } from 'react-router-dom';
5 export default function Structure({ compUsers, children, selectedCompanyId }) {
6   const [userConnect, setUserConnect] = useState();
7   const [showMenu, setShowMenu] = useState(false);
8
9   const [borderClass, setBorderClass] = useState("");
10
11  useEffect(() => {
12    ...
13  }, []);
14
15  useEffect(() => {
16    ...
17    [selectedCompanyId, compUsers, userConnect];
18
19    // Afficher une bordure de couleur selon UserComp.rolesN2 de Cie active
20    // Sélectionner la compagnie active
21    const activeCompUser = compUsers?.find(
22      ...
23    );
24
25    // Fonction pour déterminer la classe de bordure
26    const getBorderClass = () => {
27      ...
28    };
29
30    const handleNavigation = (route) => {
31      ...
32    };
33
34    return (
35      <div className="h-screen w-screen overflow-auto">
36        <div className="bg-gray-900 w-full flex justify-between items-center px-10 py-1.5">
37          <div className="flex space-x-10 items-center" ...
38          <div className="flex space-x-8 items-center" ...
39          </div>
40        </div>
41        <div>
42          {children}
43        </div>
44      </div>
45    );
46  }
47
48  return (
49    <div className="h-screen w-screen overflow-auto">
50      <div className="bg-gray-900 w-full flex justify-between items-center px-10 py-1.5">
51        <div className="flex space-x-10 items-center" ...
52        <div className="flex space-x-8 items-center" ...
53        </div>
54      </div>
55      <div>
56        {children}
57      </div>
58    </div>
59  );
60}
```

On met dans le même bloc
La construction du composant

Et le contenu du fichier appelant
{children}

L'expérience de CCO l'ont conduit à créer une bibliothèque personnalisée de fonction API ainsi qu'un contrôleur générique destinés à mutualiser le code récurrent et s'assurer, notamment pour le contrôleur que les contrôles ne soient pas oubliés.

Composants utilitaires CCO :

Ces composants utilitaires encapsulent du code répétitif.

ApiHelper.js

La bibliothèque de fonctions API personnalisées, **ApiHelper.js**, regroupe le code des différents types appels API (GET, POST, PATCH, etc.).

```
assets > Helpers > JS ApiHelper.js > ...
1 import { addToastError } from "../base/Components/Toasts";
2
3 // Fonctions pour les requêtes API de type GET
4 > export function methodGet(path, showError = true, defaultPath = process.env.REACT_APP_API_PATH) { ...
5   }
6
7
8 // Fonctions pour les requêtes API de type POST
9 > export function methodPost(path, params, showError = true, defaultPath = process.env.REACT_APP_API_PATH) { ...
10  }
11
12 // Fonctions pour les requêtes API de type POST avec des données de formulaire
13 > export function methodPostFormData(path, formData, showError = true, defaultPath = process.env.REACT_APP_API_PATH) { ...
14  }
15
16 // Fonctions pour les requêtes API de type PUT
17 > export function methodPut(path, params, showError = true, defaultPath = process.env.REACT_APP_API_PATH) { ...
18  }
19
20 // Fonctions pour les requêtes API de type PATCH
21 > export function methodPatch(path, params, defaultPath = process.env.REACT_APP_API_PATH) { ...
22  }
23
24
25 // Fonctions pour les requêtes API de type DELETE
26 > export function methodDelete(path, showError = true, defaultPath = process.env.REACT_APP_API_PATH) { ...
27  }
28
29 > function getNewHeaders(isJson = true) { ...
30  }
31
32
33 > function fetchApi(path, myInit, showError = true) { ...
34  }
35
36 let pathToAbort = {};
37
38 // permet d'annuler toutes les requêtes en cours
39 > export function abortAllRequests() { ...
40  }
41
42
43 // permet d'annuler une requête spécifique
44 > export function abortRequest(path) { ...
45 }
```

```
3 // Fonctions pour les requêtes API de type GET
4 export function methodGet(path, showError = true,
5   defaultPath = process.env.REACT_APP_API_PATH) {
6   return new Promise((resolve, reject) => {
7     let myInit = getNewHeaders();
8     myInit.method = "GET";
9
10    fetchApi(defaultPath + path, myInit, showError)
11      .then(resolve)
12      .catch(reject);
13  });
14 }
```

```
54 // Fonctions pour les requêtes API de type PATCH
55 export function methodPatch(path, params, defaultPath =
56   process.env.REACT_APP_API_PATH) {
57   return new Promise((resolve, reject) => {
58     let myInit = getNewHeaders();
59     myInit.method = "PATCH";
60     myInit.body = JSON.stringify(params);
61
62     fetchApi(defaultPath + path, myInit)
63       .then(resolve)
64       .catch(reject);
65   });
66 }
```

```
15 // Fonctions pour les requêtes API de type POST
16 export function methodPost(path, params, showError =
17   true, defaultPath = process.env.REACT_APP_API_PATH) {
18   return new Promise((resolve, reject) => {
19     let myInit = getNewHeaders();
20     myInit.method = "POST";
21     myInit.body = JSON.stringify(params);
22   });
23 }
```

```
28 // Fonctions pour les requêtes API de type POST avec des
29 // données de formulaire
30 export function methodPostFormData(path, formData,
31   showError = true, defaultPath = process.env.
32   REACT_APP_API_PATH) {
33   return new Promise((resolve, reject) => {
34     let myInit = getNewHeaders(false);
35     myInit.method = "POST";
36     myInit.body = formData;
37   });
38 }
```

Ainsi que le code de gestion d'une promesse via la fonction **fetchApi**.

```
96 function fetchApi(path, myInit, showError = true) {
97   pathToAbort[path] = myInit.abort;
98
99   return new Promise((resolve, reject) => {
100     fetch(path, myInit)
101       .then(async (res) => {
102         if (!res.ok) {
103           let error = await res.json();
104           error = { ...error, status: res.status };
105           reject(error);
106         } else if (res.headers.get("content-type") ==
107           "application/json") {
108           await res
109             .json()
110             .then((res) => resolve(JSON.parse(res)))
111             .catch(reject);
112         } else if (res.headers.get("content-type") ==
113           "application/pdf") {
114           resolve(res.blob());
115         } else resolve(res);
116       });
117   });
118 }
```

```
96 function fetchApi(path, myInit, showError = true) {
97   pathToAbort[path] = myInit.abort;
98
99   return new Promise((resolve, reject) => {
100     fetch(path, myInit)
101       .then(async (res) => {
102         if (!res.ok) {
103           let error = await res.json();
104           error = { ...error, status: res.status };
105           reject(error);
106         } else if (res.headers.get("content-type") ==
107           "application/json") {
108           await res
109             .json()
110             .then((res) => resolve(JSON.parse(res)))
111             .catch(reject);
112         } else if (res.headers.get("content-type") ==
113           "application/pdf") {
114           resolve(res.blob());
115         } else resolve(res);
116       });
117   });
118 }
```

```

79     function getNewHeaders(isJson = true) {
80         const controller = new AbortController();
81         let myHeaders = new Headers();
82         let myInit = {
83             headers: myHeaders,
84             signal: controller.signal,
85             abort: controller.abort,
86             credentials: "include",
87         };
88         if (isJson) {
89             myHeaders.append("accept", "application/json");
90             myHeaders.append("Content-Type", "application/json");
91         }
92         return myInit;
93     }
94 }

```

Comme pour les bibliothèques natives, il suffit d'importer les fonctions dont on a besoin.

```
2 import { methodGet, methodPatch } from "../../helpers/ApiHelper";
```

Et de les utiliser pour lancer le fetch d'une API selon la méthode attendue par la route.

```
// fetch CompUser
methodGet("/company/get_all_companies_uconnect")
```

La methodGet va interroger une route définie dans le contrôleur et dont le code associé correspond à la demande (voir plus loin § Le back-end / Code composant métier)

```

#[Route('/api/company')]
#[IsGranted('ROLE_USER')]
class CompanyController extends GenericController
{
    /**
     * Routes ouvertes à tous mais sécurisées par rapport au User connecté
     */
    /**
     * Permet recuperer les company de l'utilisateur connecté
     */
    #[Route('/get_all_companies_uconnect', name: "get_all_companies_uconnect", methods: [Request::METHOD_GET])]
    public function getAllCompany(Request $request, CompUserRepository $compUserRepository): JsonResponse
    {

```

GenericController

Un controller générique a été créé pour les mêmes raisons.

```

src > Controller > GenericController.php > GenericController
1 <?php
2
3 namespace App\Controller;
4
5 use App\Exceptions\NoResultException;
6 use App\Exceptions\NotFoundException;
7
8 use Symfony\Component\Validator\Validator\ValidatorInterface;
9 use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
10 use Symfony\Component\HttpFoundation\Exception\BadRequestException;
11
12 /**
13 * Class GenericController
14 * Cette classe permet de centraliser les méthodes communes à tous les controllers
15 *
16 * @package App\Controller
17 */
18 abstract class GenericController extends AbstractController
19

```

Le constructeur utilise l'injection de dépendances pour recevoir des instances d'objets qui implémentent les 3 interfaces dont nous aurons besoin : EntityManagerInterface, SerializerInterface, et ValidatorInterface.

```

23     public function __construct(
24         protected EntityManagerInterface $entityManager,
25         protected SerializerInterface $serializer,
26         protected ValidatorInterface $validator
27     ) {}

```

3 fonctions :

- la première pour sérialiser : cad transformer une instance d'entité en un format json (dans notre cas).

```
29     /**
30      * Permet de serialiser une entité
31      */
32     protected function serialize($data, array $context = [], string $format = 'json'): string
33     {
34         return $this->serializer->serialize($data, $format, ["groups" => $context]);
35     }
```

- une autre pour persister une instance

```
protected function validateAndPersist(mixed $object, bool $flush = false): ?int {
    $errors = $this->validator->validate($object);
    if (count($errors) > 0)
        throw new ValidatorException($errors);

    $this->entityManager->persist($object);
    if ($flush) {
        $this->entityManager->flush();
        return $object->getId();
    }

    return null;
}
```

- et enfin la dernière pour mettre à jour une relation

```
protected function setEntityRelation(mixed &$object, string $fieldName, array $associativeData, string $className): void {
    // On vérifie si la méthode existe
    if (!method_exists($object, 'set' . ucfirst($fieldName)))
        throw new BadRequestException(sprintf("La méthode set%s n'existe pas dans l'entité %s", ucfirst($fieldName), get_class($object)));

    // On vérifie si le type de $associativeData est un tableau
    if (gettype($associativeData) !== "array")
        throw new BadRequestException(sprintf("Le type de \$associativeData n'est pas celui attendu : \"%s\" trouvé au lieu d'un tableau.", gettype($associativeData)));

    // est que le champ existe dans le tableau ?
    if (!array_key_exists($fieldName, $associativeData))
        throw new BadRequestException(sprintf("Le tableau associatif fourni \$associativeData ne contient pas la clé '%s'", $fieldName));

    // est que le champ existe dans le tableau ?
    $function = 'set' . ucfirst($fieldName);
    // est ce que le champ est vide ?
    if (!empty($associativeData[$fieldName])) {
        // On récupère le repository de l'entité
        $repository = $this->entityManager->getRepository($className);
        // Si on a un id, on récupère l'entité
        $response = $repository->find($associativeData[$fieldName]);
        if (empty($response))
            throw new NotFoundException(sprintf("L'entité %s avec l'id %d n'existe pas", $className, $associativeData[$fieldName]));
        // On met à jour
        $object->$function($response);
    } else {
        // On met à null
        $object->$function(null);
    }
}
```

Tous les Controllers n'héritent pas de **AbstractController** mais de la classe abstraite **GenericController** qui elle hérite de **AbstractController**.

```
21 #[Route('/api/company')]
22 #[IsGranted('ROLE_USER')]
23 class CompanyController extends GenericController
24 {
```

D. Extrait de code partie dynamique

Le fait de construire les pages avec React simplifie grandement la gestion dynamique des écrans, grâce à la puissance des variables d'état.

Sélection de la société et mise à jour de l'affichage

Le changement de société dans le select entraîne une mise à jour de l'affichage.

Création d'une variable qui surveille l'état du select.

```
// Modification de la cie sélectionnée dans le select
const [selectedCompanyId, setSelectedCompanyId] = useState(null);
```

Création d'une autre (**compConnect**) qui contient les informations de la société en cours (valeur active du select).

```
const [compConnect, setCompConnect] = useState();
```

Sa valeur est liée à la propriété **activeComp** du User.

```
// Modification de la cie sélectionnée dans le select :
// Initialisation de selectedCompanyId
useEffect(() => {
  if (userConnect && userConnect.activeComp) {
    setSelectedCompanyId(userConnect.activeComp);

    // fetch CompUser
    methodGet("/company/get_all_companies_uconnect")
      .then((res) => {
        //console.log(res);
        // mettre à jour la variable
        setCompUsers(res);
      })
      .catch((err) => {
        // gerer l'erreur
        console.log(err.errorMessage);
      });
  }
}, [userConnect]);

useEffect(() => {
  if (userConnect && selectedCompanyId) {
    // Company en cours
    methodGet(`/company/${selectedCompanyId}`)
      .then((compCours) => {
        //console.log(compCours);
        // mettre à jour la variable
        setCompConnect(compCours);
      })
      .catch((err) => {
        console.error(err);
      });
  }
}, [userConnect, selectedCompanyId]);
```

On positionne l'écouteur d'événement **onChange** sur le select.

```
<div id="selectCie" className="relative">
  {compUsers && compUsers.length > 1 ? (
    <select
      className="appearance-none w-full bg-gray-50 border border-gray-300 py-2 px-3 leading-tight focus:outline-none focus:ring-1 focus:ring-indigo-500"
      // Modification de la cie sélectionnée dans le select :
      onChange={handleCompanyChange}
      value={selectedCompanyId} || ""
    >
      {compUsers !== undefined &&
        compUsers.map((compUser) => (
          <option
            key={compUser.company.id}
            value={compUser.company.id}
          >
            {compUser.company.morale
              ? compUser.company.nameM
              : `${compUser.company.lastNameP} || ${compUser.company.firstNameP} || ${compUser.company.middleNameP}`.trim()
            }
          </option>
        )));
    </select>
  ) : null;
</div>
```

Le changement de société dans le select lance l'exécution de la fonction fléchée **handleCompanyChange** où la valeur de la variable d'état **compConnect** est mise à jour.

```
// Modification de la Cie sélectionnée dans le select
const handleCompanyChange = (event) => {
  const companyId = parseInt(event.target.value, 10);
  setSelectedCompanyId(companyId);

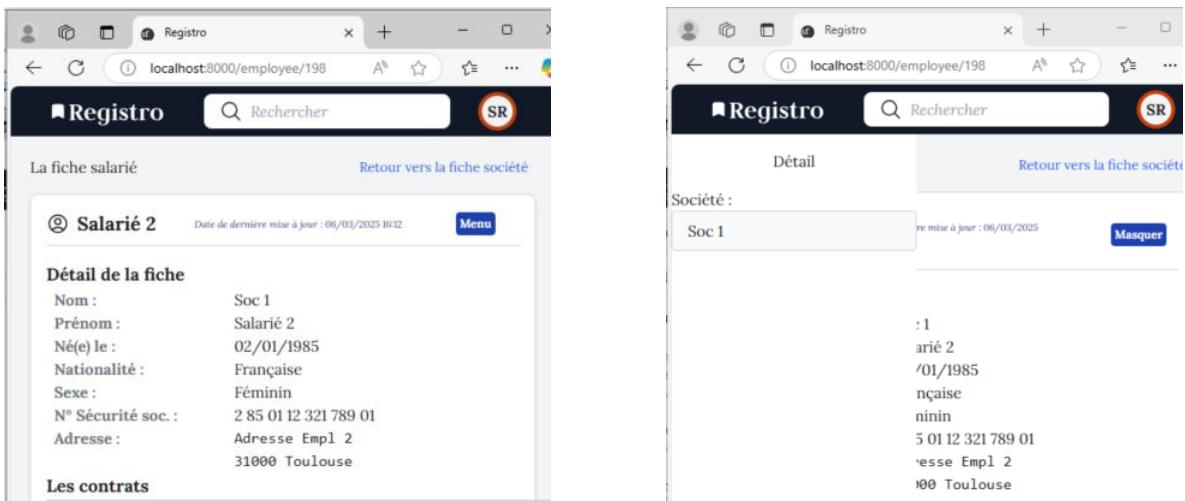
  // Mettre à jour User.activeComp
  if (userConnect) {...}
};
```

Ce qui met à jour l'affichage de la page avec les données de la société sélectionnée.

Masquer et afficher le menu sur petit écran.

Ici associe l'utilisation

- de classes de style dynamique de Tailwind pour déplacer le menu (transform et translate-x-0/full), et
- du code java associé à un bouton qui, sur clic, modifie la valeur de la variable d'état **isMenuOpen**.



La variable **isMenuOpen** est initialisée à false, donc par défaut le menu est masqué.

```
// Masquer la div blocMenu sur petit écran
const [isMenuOpen, setIsMenuOpen] = useState(false);
```

On surveille le clic sur le bouton. Il fait passer **isMenuOpen** de false à true et inversement.

```
/* Bouton pour ouvrir/fermer le menu sur petit écran */
<button>
  // className="lg:hidden absolute top-4 left-4 z-20 bg-blue-500 text-white p-2 rounded"
  // className="lg:hidden mr-3 bg-blue-800 text-white p-1 rounded text-xs font-bold"
  onClick={() => setIsMenuOpen(!isMenuOpen)}
>
  {isMenuOpen ? "Masquer" : " Menu "}
</button>
```

Nota : le bouton est par défaut visible et sur le breakpoint lg il est caché.

Sur la div contenant le menu, on place toutes les classes dynamiques

```
<div id="blocPage" className="flex h-full relative">
  <div id="blocMenu">
    className={`${isMenuOpen ? "bg-white w-64 flex-shrink-0 absolute inset-y-0 left-0 transform" : "-translate-x-full"} transition duration-200 ease-in-out z-10 lg:relative lg:translate-x-0 lg:block`}
  >
    <div className="text-center my-4"> ...
    </div>

    <p>Société :</p>
    <div className="w-full bg-gray-50 border border-gray-300 rounded px-4 py-2"> ...
    </div>
  </div>
```

La position du menu est fixée à gauche et sur toute la hauteur de l'écran (**left-0** et **inset-y-0**).

La classe **transform** active les transformations css de la balise.

Les classes **translate-x-0** affiche le menu et **-translate-x-full** masque le menu. Leur affectation dépend de la valeur de **isMenuOpen**.

Affichage de données supplémentaires sur le survol d'éléments

Sociétés et Rôles : (3)

	RESP	RH	CPT
Soc 1	x		
Soc 2		x	
firstNameP			x
Désactivé (1)			
Soc 4	x		

Sociétés et Rôles : (3)

	RESP	RH	CPT
Soc 1	x		
Soc 2		x	
firstNameP			x
Désactivé (1)			
Soc 4	x		

Dernière mise à jour le : 06/03/2025 16:12:38

Suppression le : 15/03/2025 09:12:00 par CCO Dev

Il est lié à la variable d'état `selectedRowInfo`.

```
// Date de mise à jour, de Suppression et UserDelete
const [selectedRowInfo, setSelectedRowInfo] = useState(null);
```

Sur le survol de la ligne on renseigne `selectedRowInfo` avec les données souhaitées et lorsqu'on quitte le focus on la réinitialise à vide.

```
<tr key={compUser.id}>
  className={ index % 2 === 0 ? "bg-white" : "bg-sky-700" }
  // Affichage des métadonnées de la ligne dans la div sous le tableau
  onMouseEnter={() => setSelectedRowInfo({
    dateUpdate: compUser.dateUpdate,
    dateDelete: compUser.dateDelete,
    userDelete: compUser.UserDelete
  })}
  onMouseLeave={() => setSelectedRowInfo(null)}
```

Ensuite il suffit de conditionner l'affichage des éléments au fait que la variable soit instanciée et non vide.

```
<div className="mt-2 text-sm text-center">
  /* Affichage des métadonnées lors du survol de chaque ligne du tableau ci-dessus */
  {selectedRowInfo ? (
    <>
      <p className="text-xxs italic text-indigo-900" >Dernière mise à jour le : {new Date(selectedRowInfo.dateUpdate).toLocaleString('fr-FR')}

```

Accès à la page des salariés

Lors du double clic sur la ligne du salarié, il suffit de lancer l'url liée à l'affichage de la page avec comme paramètre l'identifiant de salarié.

```
// Accès à la page Employee
import { useNavigate } from "react-router-dom";

// Accès à la page Employee
const navigate = useNavigate();

{employees !== undefined &&
  employees.map((empl, index) => (
    <tr key={empl.employee.id}>
      className={ index % 2 === 0
        ? "bg-white hover:bg-sky-700 hover:text-white"
        : "bg-gray-200 hover:bg-sky-700 hover:text-white"
      }
      // Accès à la page Employee
      onDoubleClick={() => navigate(`employee/${empl.employee.id}`) }
    </tr>
  ))}
}
```

```
14  export default function App() {
15    // document.documentElement.classList.add('dark')
16    // document.body.classList.add('darkbody')
17    return (
18      <Routes>
19        {/* <Route path="/" element={<Home />} /> */}
20        <Route path="/" element={<Company />} />
21        <Route path="/company" element={<Company />} />
22        <Route path="/profile" element={<Profile />} />
23        <Route path="/employee/:idEmpl" element={<Employee />} />
24        <Route path="/demarrer/*" element={<CompanyOnboarding />} />
25        <Route path="*" element={<NotFound />} />
26      </Routes>
27    );
28  }
```

Ces url sont définies dans le fichier App.jsx

E. Les fixtures :

Ce sont des fichiers de création de données.

Plutôt qu'une création en masse, j'ai privilégié des cas qui me paraissent plus représentatifs.

Ex de la création de données dans **User**, **Company** et la table de jonction **CompUser**.

```
src > DataFixtures > UserFixtures.php ...
1  <?php
2
3  namespace App\DataFixtures;
4
5  use App\Entity\User;
6  use Doctrine\Persistence\ObjectManager;
7  use Doctrine\Bundle\FixturesBundle\Fixture;
8  use Symfony\Component\PasswordHasher\Hasher\UserPasswordHasherInterface;
9  use Doctrine\Bundle\FixturesBundle\FixtureGroupInterface;
10
11 class UserFixtures extends Fixture implements FixtureGroupInterface
12 {
13     protected UserPasswordHasherInterface $passwordHasher;
14
15     public function __construct(UserPasswordHasherInterface $passwordHasher)
16     {
17         $this->passwordHasher = $passwordHasher;
18     }
19
20     public function load(ObjectManager $manager): void
21     {
22         // Users avec Role M1 : CCO -----
23         $this->createAndPersist($manager, ['ROLE_SUPER_ADMIN'], 'CCO', 'Dev', 'CCO Dev', 1, 'support.dev@cco-info.fr');
24         $this->createAndPersist($manager, ['ROLE_SUPER_ADMIN'], 'Nicolas', 'Calvelo', 'Calvelo', 2, 'nicolas@cco-info.fr');
25         $this->createAndPersist($manager, ['ROLE_GESTION'], 'Gestion', 'CCO', 'Gestion CCO', 3, 'gestion@cco-info.fr');
26
27         // Users avec Role N1 : CLIENT -----
28         $this->createAndPersist($manager, ['ROLE_CLIENT'], 'Resp', 'Soc 1', 'Responsable Soc 1', 'Soc1_Resp', 'responsable@soc1.fr');
29         $this->createAndPersist($manager, ['ROLE_CLIENT'], 'Rh', 'Soc 1', 'RH Soc 1', 'Soc1_Rh', 'rh@soc1.fr');
30         $this->createAndPersist($manager, ['ROLE_CLIENT'], 'Resp', 'Soc 2', 'Responsable Soc 2', 'Soc2_Resp', 'responsable@soc2.fr');
31         $this->createAndPersist($manager, ['ROLE_CLIENT'], 'Compta', 'Soc 1', 'Comptable Soc 2', 'Soc1_Compta', 'compta@soc1.fr');
32
33         $manager->flush();
34     }
35
36     private function createAndPersist( ObjectManager $manager,
37                                         $role, $firstN, $lastN, $userN, $uuid, $email): void
38     {
39         $newInstance = (new User())
40             ->setRoles($role)
41             ->setFirstName($firstN)
42             ->setLastName($lastN)
43             ->setUsername($userN)
44             ->setUuid($uuid)
45             ->setEmail($email)
46             ->setPassword($this->passwordHasher->hashPassword(new User(), '12345'));
47         $manager->persist($newInstance);
48     }
49
50     // implements FixtureGroupInterface
51     public static function getGroups(): array
52     {
53         return ['user'];
54     }
55 }
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70 }
```

```
src > DataFixtures > CompanyFixtures.php ...
1  <?php
2
3  namespace App\DataFixtures;
4
5  use App\Entity\Company;
6  use App\Repository\UserRepository;
7  use Doctrine\Bundle\FixturesBundle\Fixture;
8  use Doctrine\Common\DataFixtures\DependentFixtureInterface;
9  use Doctrine\Persistence\ObjectManager;
10
11 class CompanyFixtures extends Fixture implements DependentFixtureInterface
12 {
13     public function __construct(protected UserRepository $userRep) {}
14
15     public function load(ObjectManager $manager): void
16     {
17         $compName = "Soc 1";
18         $uuid = str_replace(' ', '', $compName) . '_Resp';
19         $user = $this->userRep->findOneBy(['disabled' => false, 'uuid' =>$uuid]);
20         if (!empty($user))
21             $dateCreate = new \Datetime("2000-02-19");
22         $this->createAndPersist($manager, "123456789", $user, 1, true, $compName, "1234A", "Toutes les activités ", $dateCreate, 1);
23
24         $compName = "Soc 2";
25         $uuid = str_replace(' ', '', $compName) . '_Resp';
26         $user = $this->userRep->findOneBy(['disabled' => false, 'uuid' =>$uuid]);
27         if (!empty($user))
28             $dateCreate = new \Datetime("2005-07-21");
29         $this->createAndPersist($manager, "987456321", $user, 2, true, $compName, "1234C", "Toutes les activités ", $dateCreate, 3);
30
31         $compName= "Soc 4";
32         $uuid = 'Soc_1_Resp';
33         $user = $this->userRep->findOneBy(['disabled' => false, 'uuid' =>$uuid]);
34         if (!empty($user))
35             $dateCreate = new \Datetime("2002-05-25");
36         $this->createAndPersist($manager, "951357852", $user, 2, true, $compName, "1234D", "Toutes les activités ", $dateCreate, 6);
37
38         $manager->flush();
39     }
40
41
42     private function createAndPersist( ObjectManager $manager,
43                                         $siret, $user, $monthClose, $morale, $compName,
44                                         $ape, $activity , \DateTimeInterface $dateCreate, $mClose,
45                                         ?string $sexEP = null, ?string $firstNameP = null): void
46     {
47
48         $lesActivites = $activity . str_replace('Soc', 'Société', $compName);
49         $lesActivites = $lesActivites . (!empty($sexEP) ? ' Physique' : null);
50         $company = (new Company())
51             ->setSiret($siret)
52             ->setReferentContact($user)
53             ->setMonthCloseAccount($monthClose)
54             ->setMorale($morale)
55             ->setCodeApeNaf($ape)
56             ->setActivities($lesActivites)
57             ->setDateCreate($dateCreate)
58             ->setMonthCloseAccount($mClose);
59
60         // Définir soit nameM, soit lastNameP
61         if ($morale)
62             $company->setNameM($compName);
63         else []
64             $company->setLastNameP($compName)
65             ->setFirstNameP($firstNameP)
66             ->setSexEP($sexEP);
67
68         // Ajout de la company dans la collection companies de user
69         $user->addCompany($company);
70
71         $manager->persist($user);
72         $manager->persist($company);
73
74     }
75
76
77
78
79
80
81
82
83
84
85
86     public function getDependencies(): array
87     {
88         return [UserFixtures::class];
89         /***** Les dépendances pour la compagnie ****/
90         // avec ce doctest, on a une erreur de type
91         // avec ce doctest, on a une erreur de type
92     }
93
94
95
96
97
98
99
100 }
```

```

src > DataFixtures > CompUserFixtures.php > ...
1  <?php
2
3  namespace App\DataFixtures;
4
5  use App\Entity\CompUser;
6  use App\Repository\CompanyRepository;
7  use App\Repository\UserRepository;
8  use Doctrine\Bundle\FixturesBundle\Fixture;
9  use Doctrine\Common\DataFixtures\DependentFixtureInterface;
10 use Doctrine\Persistence\ObjectManager;
11
12 class CompUserFixtures extends Fixture implements DependentFixtureInterface
13 {
14     public function __construct(protected UserRepository $userRep,
15                             | | | | | protected CompanyRepository $cieRep) {}
16
17     public function load(ObjectManager $manager): void
18     {
19         $companies = $this->cieRep->findAll();
20         foreach ($companies as $company) {
21             // cette variable permettra de rechercher un User dans le findOneBy
22             if ($company->isMorale()){
23                 $uuid = str_replace(' ', ' ', $company->getNameM());
24             } else {
25                 $uuid = str_replace(' ', ' ', $company->getFirstNameP());
26             }
27
28             // ROLE_CLIENT_RESP -----
29             $user = $this->userRep->findOneBy(['disabled' => false, 'uuid' =>$uuid . '_Resp']);
30             if (!empty($user)){
31                 $this->createAndPersist($manager, $company , $user, ["ROLE_CLIENT_RESP"]);
32             }
33
34             // ROLE_CLIENT_RH -----
35             $user = $this->userRep->findOneBy(['disabled' => false, 'uuid' =>$uuid . '_Rh']);
36             if (!empty($user)){
37                 $this->createAndPersist($manager, $company , $user, ["ROLE_CLIENT_RH"]);
38             }
39
40             // Cas particuliers -----
41             // de la société 2 on lui affecte le User.username = "Soc1_Resp" comme RH
42             if ($company->getNameM() == "Soc 2"){
43                 $user = $this->userRep->findOneBy(['disabled' => false, 'uuid' =>'Soc1_Resp']);
44                 if (!empty($user)){
45                     $this->createAndPersist($manager, $company , $user, ["ROLE_CLIENT_RH"]);
46                 }
47
48             // de la société 4 on lui affecte le User.username = "Soc1_Resp" comme RESP
49             if ($company->getNameM() == "Soc 4"){
50                 $user = $this->userRep->findOneBy(['disabled' => false, 'uuid' =>'Soc1_Resp']);
51                 if (!empty($user)){
52                     $this->createAndPersist($manager, $company , $user, ["ROLE_CLIENT_RESP"]);
53                 }
54
55             }
56
57             $manager->flush();
58         }
59
60         private function createAndPersist( ObjectManager $manager, $company, $user, $role): void
61     {
62         // Création du CompUser
63         $newInstance = ((new CompUser())
64                         | | | | | ->setCompany($company)
65                         | | | | | ->setUser($user)
66                         | | | | | ->setRolesN2($role));
67         $manager->persist($newInstance);
68
69         //dump($newInstance->getUser()->getFirstName() . " - " . $newInstance->getCompany()->getNameM());
70
71         // Ajout du CompUser dans la collection compUsers de User
72         $user->addCompUser($newInstance);
73         if (is_null($user->getActiveComp())){
74             // Maj de User.activeComp => société active du User
75             $user->setActiveComp($company->getId());
76         }
77         $manager->persist($user);
78
79         // Ajout du CompUser dans la collection compUsers de Company
80         $company->addCompUser($newInstance);
81         $manager->persist($company);
82     }
83
84     public function getDependencies(): array
85     {
86         return [UserFixtures::class, CompanyFixtures::class];
87         /**
88          * Note : dans UserFixtures, les User ont été créés avec
89          */
90     }
91
92 }
93
94
95
96
97
98
99
100
101
102

```

Nota :

Lors de la création des Company et des CompUser, il faut avoir des User. C'est le rôle du module des dépendances qui permet de lancer d'abord la création des User, ensuite celle des Company et enfin celle des CompUser.

```

Invité de commandes
C:\Users\Utilisateur\Documents\cco_registro>php C:\Users\Utilisateur\Documents\cco_registro\bin\Console
doctrine:fixtures:load

Careful, database "registro" will be purged. Do you want to continue? (yes/no) [no]:
> yes

> purging database
> loading App\DataFixtures\UserFixtures
> loading App\DataFixtures\ConvCollFixtures
> loading App\DataFixtures\SexeFixtures
> loading App\DataFixtures\NationalityFixtures
> loading App\DataFixtures\CountryFixtures
> loading App\DataFixtures\TypeContractFixtures
> loading App\DataFixtures\CompanyFixtures
> loading App\DataFixtures\EmployeeFixtures
> loading App\DataFixtures\CompUserFixtures
> loading App\DataFixtures\UserDeleteFixtures
> loading App\DataFixtures\JobsheetFixtures
> loading App\DataFixtures\CompConvCollFixtures
> loading App\DataFixtures\CompEmpFixtures
> loading App\DataFixtures\TypePartnerFixtures
> loading App\DataFixtures\PartnerFixtures
> loading App\DataFixtures\ContractFixtures

C:\Users\Utilisateur\Documents\cco_registro>

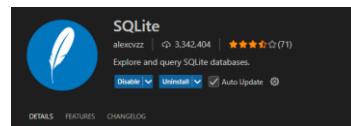
```

F. Tests fonctionnels :

Ces tests visent à simuler des scénarios de navigation, afin de valider les contrôleurs et les formulaires.

Lors de cette phase, les fixtures seront utilisées pour générer automatiquement des données dans les tables nécessaires aux tests.

Ils seront effectués sur une base de données de test utilisant SQLite. Cela nécessite l'installation de l'extension SQLite dans VS Code et la configuration appropriée de Doctrine.



Paramétrage de la base de test :

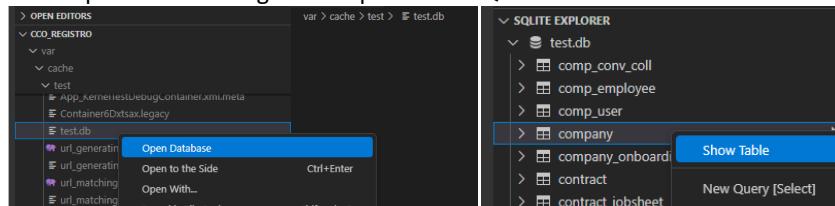
```

config/packages/doctrine.yaml
2/
28 when@test:
29   doctrine:
30     dbal:
31       # "TEST_TOKEN" is typically set by ParaTest
32       dbname_suffix: '_test%env(default::TEST_TOKEN)%'
33       driver: pdo_sqlite
34       path: "%kernel.cache_dir%/test.db"
35       url: null
36       options:
37         default_table_options:
38           charset: utf8mb4
39           collate: utf8mb4_unicode_ci
40         default_schema_options:
41           charset: utf8mb4
42           collate: utf8mb4_unicode_ci
43         override_sqlite_type_mapping:
44           varchar: string
45

```

La base de données se trouve dans le dossier var/cache/test

Le menu contextuel de la base permet l'affichage de l'explorateur SQLite.



Sur chaque table un autre menu permet d'afficher son contenu.

SQL ▾

```

SELECT * FROM company;

```

id	referent_contact_id	user_delete_id	siren	name_m	code_ape_naf	date_create	activités	morale	last_name_p	first_name_p	sexe_p	month
1	4	NULL	123456789	Soc 1	1234A	2000-02-19	Toutes les activités Société 1	1	NULL	NULL	NULL	1
2	6	NULL	456789123	Soc 2	1234B	2005-07-21	Toutes les activités Société 2	1	NULL	NULL	NULL	2
3	4	NULL	987456321	NULL	1234C	2015-04-12	Toutes les activités Société 1 Physique	0	Soc 1	firstNameP	M	3
4	4	NULL	951357852	Soc 4	1234D	2002-05-25	Toutes les activités Société 4	1	NULL	NULL	NULL	6

CompanyTest.php :

```
tests > controller > CompanyTest.php > CompanyTest > setup
  1  <?php
  2
  3  namespace App\Tests\Controller;
  4
  5  use App\DataFixtures\CompanyFixtures;
  6  use App\DataFixtures\UserFixtures;
  7  use App\Repository\CompanyRepository;
  8  use App\Repository\UserRepository;
  9  use Symfony\Component\HttpFoundation\Request;
10  use Symfony\Component\HttpFoundation\Response;
11
12  class CompanyTest extends \App\Tests\WebTestCase
13  {
14      public function setUp(): void
15      {
16          parent::setUp();
17          $this->loadFixtures([UserFixtures::class, CompanyFixtures::class]);
18          $this->user = $this->getRepository(UserRepository::class)->find(1);
19      }
20
21      public function testCreateCompany(): void
22      {
23          $responsesBefore = $this->getRepository(CompanyRepository::class)->findAll();
24          // dump("-----" . $this->user->getUsername() . " - " . implode(", ", $this->user->getRoles()) . "-----");
25          $this->protectedRoute(Request::METHOD_POST, '/api/company', [
26              "siren" => 987654321,
27              "referentContact" => $this->user->getId(),
28              "monthCloseAccount" => 1
29          ], Response::HTTP_CREATED);
30
31          $responsesAfter = $this->getRepository(CompanyRepository::class)->findAll();
32          $this->assertEquals(count($responsesBefore) + 1, count($responsesAfter));
33      }
34
35      public function test GetAll(): void
36      {
37          $responses = $this->protectedRoute(Request::METHOD_GET, '/api/company', [], Response::HTTP_OK);
38
39          $this->assertCount(4, $responses);
40          $this->assertEquals(123456789, $responses[0]->siren);
41          $this->assertEquals(456789123, $responses[1]->siren);
42      }
43  }
```

KO

```
C:\Users\Utilisateur\Documents\cco_registro>php bin/phpunit tests\Controller\CompanyTest.php
PHPUnit 9.6.21 by Sebastian Bergmann and contributors.

Testing App\Tests\Controller\CompanyTest
. [F] 2 / 2 (100%)
Time: 00:01.793, Memory: 46.00 MB
There was 1 failure:
1) App\Tests\Controller\CompanyTest::testGetAll
Failed asserting that actual size 4 matches expected size 2.

C:\Users\Utilisateur\Documents\cco_registro\tests\controller\CompanyTest.php:39
C:\Users\Utilisateur\Documents\cco_registro\vendor\phpunit\phpunit\phpunit:107

FAILURES!
Tests: 2, Assertions: 8, Failures: 1.

Remaining indirect deprecation notices (40)

  40x: Subscribing to onSchemaCreateTable events is deprecated. (AbstractPlatform.php:2191 called by AbstractPlatform.php:2089, https://github.com/doctrine/dbal/issues/5784, package doctrine/dbal)
    40x in CompanyTest::setUp from App\Tests\Controller

C:\Users\Utilisateur\Documents\cco_registro>
```

OK

```
C:\Users\Utilisateur\Documents\cco_registro>php bin/phpunit tests\Controller\CompanyTest.php
PHPUnit 9.6.21 by Sebastian Bergmann and contributors.

Testing App\Tests\Controller\CompanyTest
...
Time: 00:01.818, Memory: 46.00 MB
OK (2 tests, 10 assertions)

Remaining indirect deprecation notices (40)

  40x: Subscribing to onSchemaCreateTable events is deprecated. (AbstractPlatform.php:2191 called by AbstractPlatform.php:2089, https://github.com/doctrine/dbal/issues/5784, package doctrine/dbal)
    40x in CompanyTest::setUp from App\Tests\Controller
```

Ces tests utilisent la classe **Symfony WebTestCase** qui permet de simuler des scénarios de navigation.

Voici ses principales caractéristiques :

1. Elle utilise le composant BrowserKit pour simuler un navigateur web.
2. Elle permet de créer un client HTTP simulé pour effectuer des requêtes vers l'application Symfony.
3. Elle n'utilise pas de vrai navigateur web ni de véritables requêtes HTTP. À la place, elle crée des instances d'objets Request de HttpFoundation et les passe au noyau de Symfony.
4. Elle permet d'inspecter la réponse renvoyée par l'application à travers une API dédiée.
5. Elle offre des méthodes pour interagir avec la page simulée, comme cliquer sur des liens, remplir des formulaires, et effectuer des assertions sur le contenu de la réponse.
6. Elle donne accès au conteneur de services de Symfony, permettant d'interagir avec les différents composants de l'application pendant les tests.

WebTestCase est particulièrement utile pour tester le comportement de l'application sans avoir besoin d'un environnement de navigateur complet, ce qui rend les tests plus rapides et plus faciles à configurer. Cependant, il est important de noter que cette approche ne permet pas de tester les interactions JavaScript ou les problèmes spécifiques aux navigateurs réels.

Là encore CCO a surchargé la classe WebTestCase de Symfony, afin de simplifier le code des tests.

```
tests > WebTestCase.php > WebTestCase
1  <?php
2
3  namespace App\Tests;
4
5  use App\Entity\User;
6  use Liip\TestFixturesBundle\Services\DatabaseToolCollection;
7  use Symfony\Component\HttpFoundation\File\UploadedFile;
8  use Symfony\Component\HttpFoundation\Response;
9
10 class WebTestCase extends \Symfony\Bundle\FrameworkBundle\Test\WebTestCase
11 {
12     // Propriétés protégées
13     protected $client;          // Client HTTP pour simuler la rq
14     protected $databaseTool;    // Outil pour manipuler la BdD
15     /** @var User $user */
16     protected $user;           // Objet User pour les tests d'authentification
17
18     public function setUp(): void
19     {
20         parent::setUp();
21         $this->client = self::createClient();
22         // Configuration de l'outil de BdD
23         $this->databaseTool = static::getContainer()->get(DatabaseToolCollection::class)->get();
24     }
25
26     public function getRepository(string $class)
27     {
28         return self::getContainer()->get($class);
29     }
30
31     //*****
32     // Chargement des données tests dans la BdD
33     public function loadFixtures(array $fixtures): void
34     {
35         $this->databaseTool->loadFixtures($fixtures);
36     }
37 }
```

Comme la quasi-totalité des routes seront protégées, cette fonction a été créée.

```
38     //*****
39     // Méthodes de test des routes protégées (IsGranted("ROLE"))
40     public function protectedRoute(string $method, string $url, array $content = [], int $code = Response::HTTP_OK)
41     : array|\stdClass|null|int
42     {
43         // Tentative d'accès sans authentification et redirection vers le login
44         $this->client->jsonRequest($method, $url, $content);
45         $this->assertResponseRedirects('/login');
46
47         // Accès avec authentification et test du statut
48         $this->client->loginUser($this->user);
49         $this->client->jsonRequest($method, $url, $content);
50         $this->assertResponseStatusCodeSame($code);
51
52         // Si contenu de la réponse non vide, décodage
53         if (!empty($this->client->getResponse()->getContent())) {
54             $response = $this->client->getResponse()->getContent();
55             while (gettype($response) === "string") {
56                 $response = json_decode($response);
57             }
58             return $response;
59         } else {
60             return null;
61         }
62     }
```

Elle teste d'abord l'accès à une route protégée sans authentification, et s'assure que l'utilisateur est redirigé vers /login. Ensuite, elle simule un utilisateur authentifié et effectue à nouveau la requête pour vérifier que l'accès est autorisé et que le code de statut HTTP correspond à celui attendu.

Si la réponse contient des données JSON, celles-ci sont décodées et retournées. Sinon, elle retourne null.

Le back-end :

J'ai commencé par reprendre les différents rôles pour schématiser leur intervention sur les différentes opérations qui seront disponibles dans l'application.

Rappel : nous avons 2 niveaux d'acteurs :

Les Acteurs 1 : ROLE_SUPER_ADMIN, ROLE_GESTION, ROLE_CLIENT

Cas d'utilisation :

- Donner les accès à l'outil (création de l'enregistrement **CompanyOnboarding**).
- Administration de la BdD : créer les rôles et leur affecter des autorisations, conformité RGPD...
- Gestion de certaines tables source : country, typeContract
- Assistance CLIENT.

Les Acteurs 2 (CLIENT) : ROLE_CLIENT_RESP, ROLE_CLIENT_RH, ROLE_CLIENT_COMPTA

Cas d'utilisation :

- Enregistrement de la Société.
- Création des utilisateurs de la société.
- Gestion du **Registre Unique du Personnel** et du suivi des salariés.
 - Création des salariés.
- Suivi des salariés.
 - Suivi des actions liées au salarié : Contrat, Carrière (postes), Visite médicale, Habilitations, Formations, Absences, Salaires.

A. Analyse UML et définition de MCD :

J'ai tout d'abord défini les cas d'utilisation (UML), puis le MCD pour aboutir au schéma de la base d données.

Diagrammes de cas :

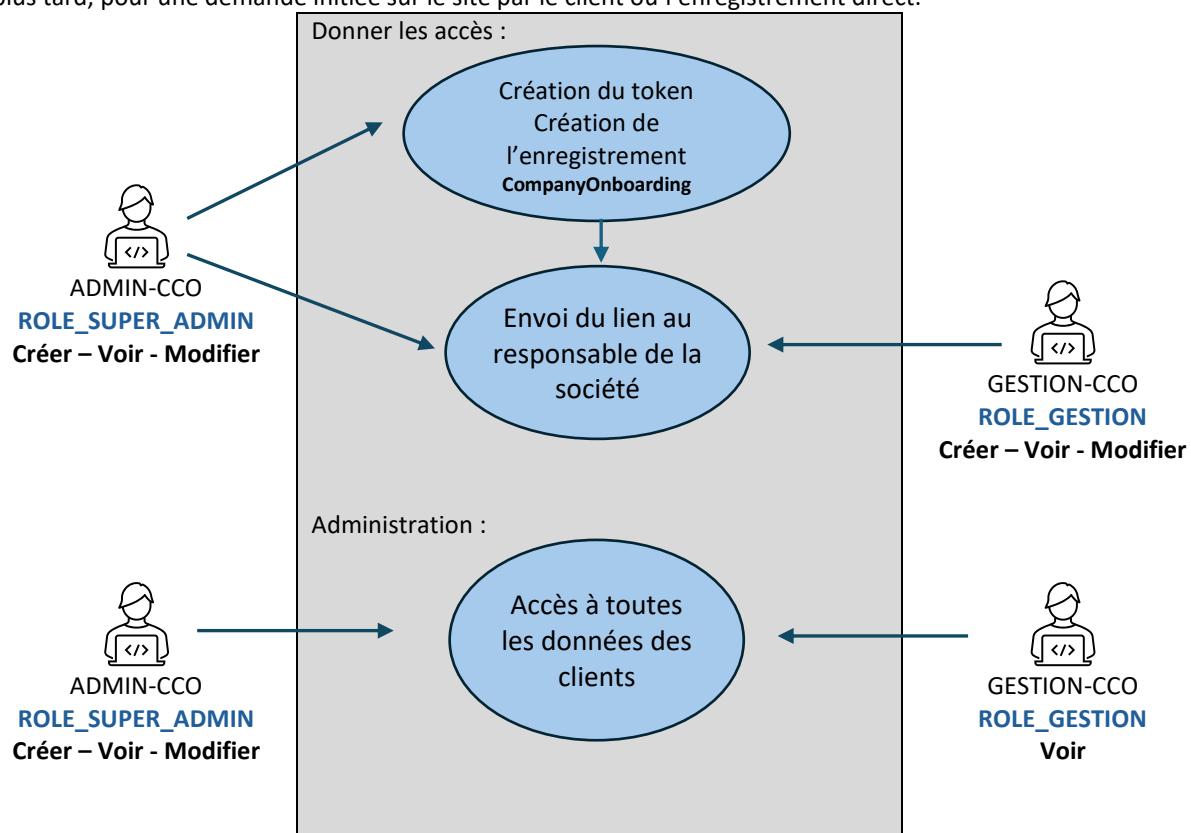
Diagrammes liés aux tâches d'administration CCO :

Toutes les actions liées à l'administration :

- Donner les accès à l'outil.
- Administrer des données.

Pour l'heure la demande de création de lien n'est pas dans le logiciel, elle résulte de la prospection.

A voir plus tard, pour une demande initiée sur le site par le client ou l'enregistrement direct.



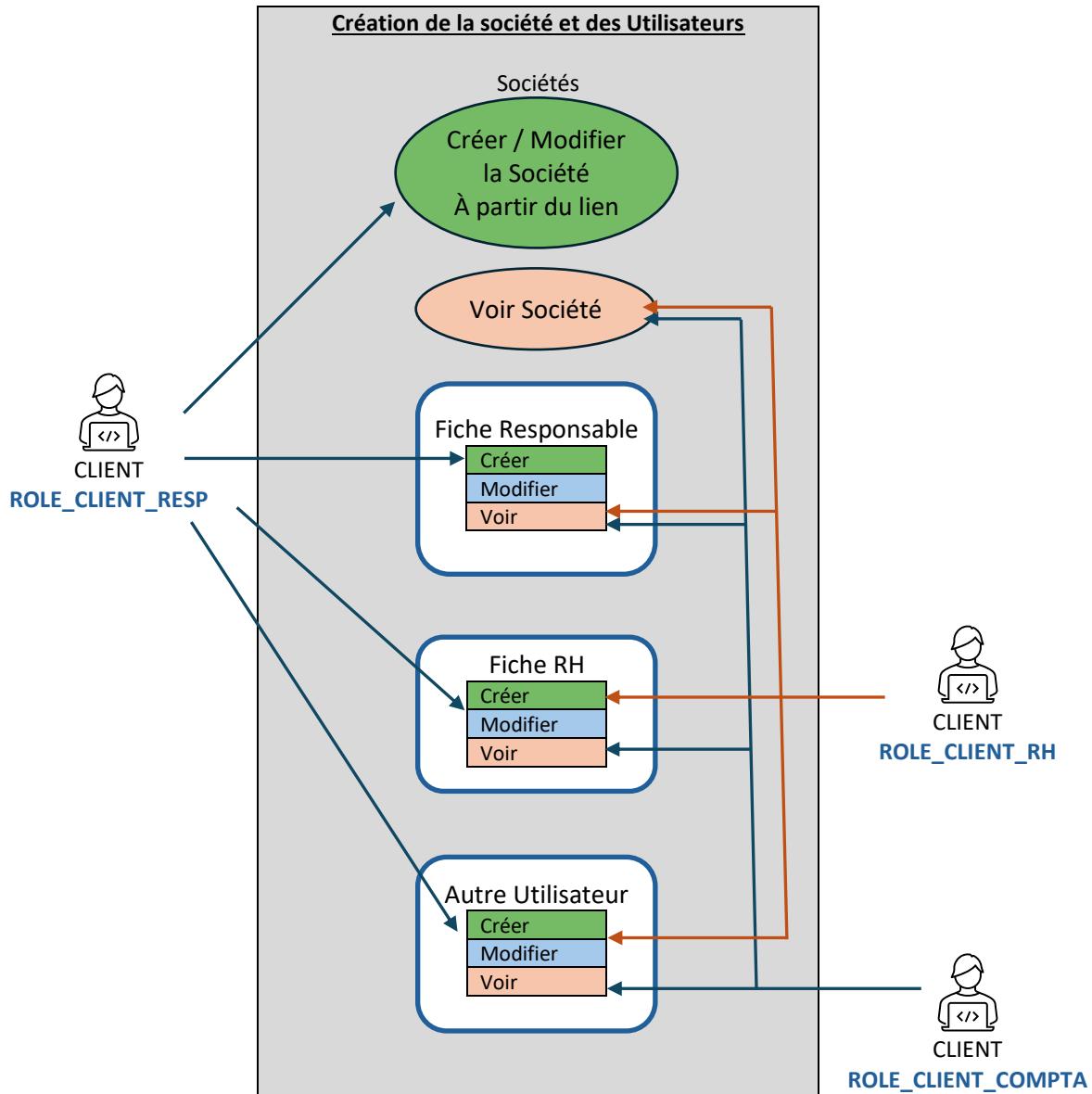
Diagrammes liés aux clients :

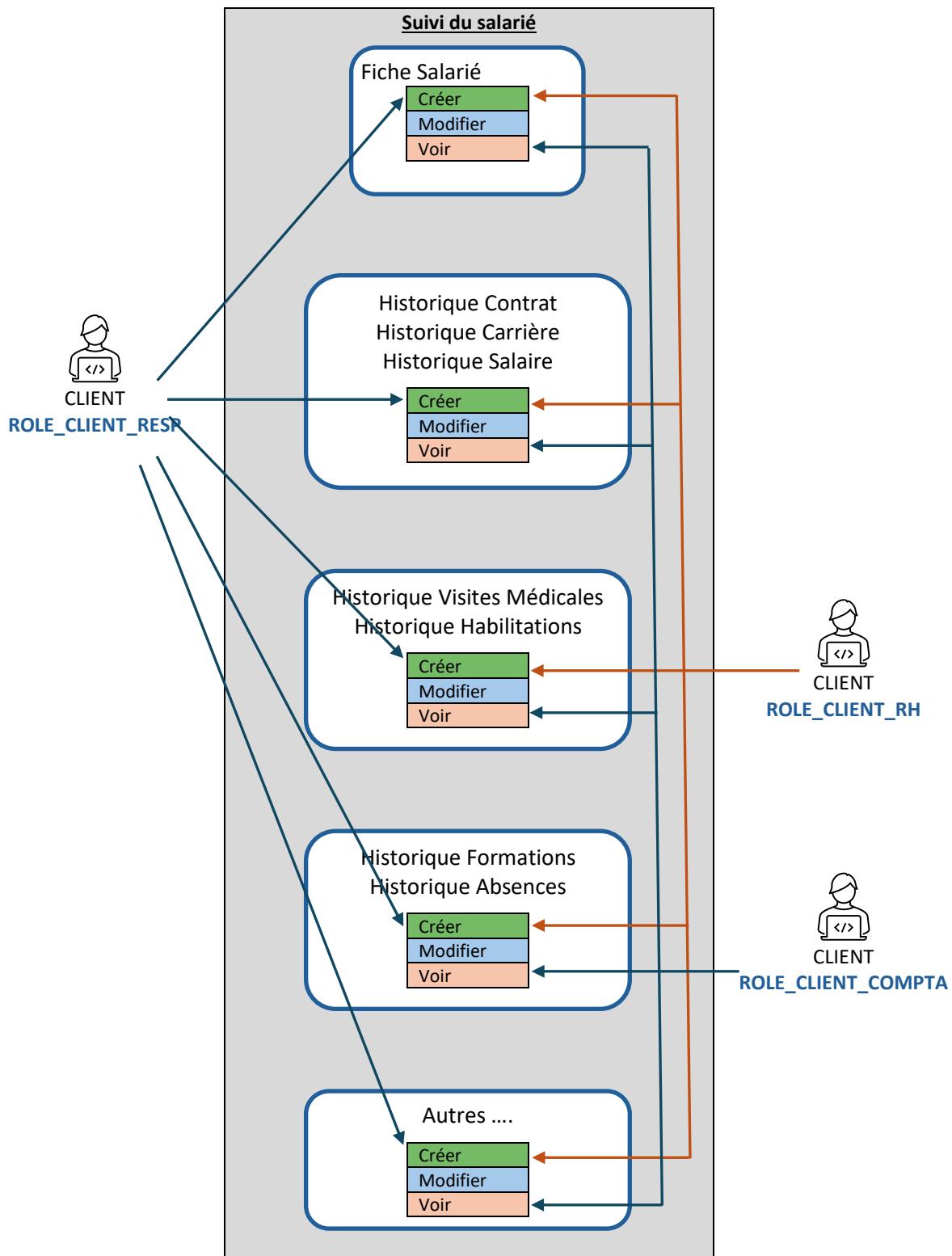
Toutes les actions liées aux attentes du client :

- Créer la société.
- Créer des utilisateurs liés à la société.
- Enregistrer les salariés
- Suivi du salarié

Porte d'entrée : par la société.

Pour les utilisateurs rattachés à plusieurs sociétés, seules les données de la société en cours sont affichées.

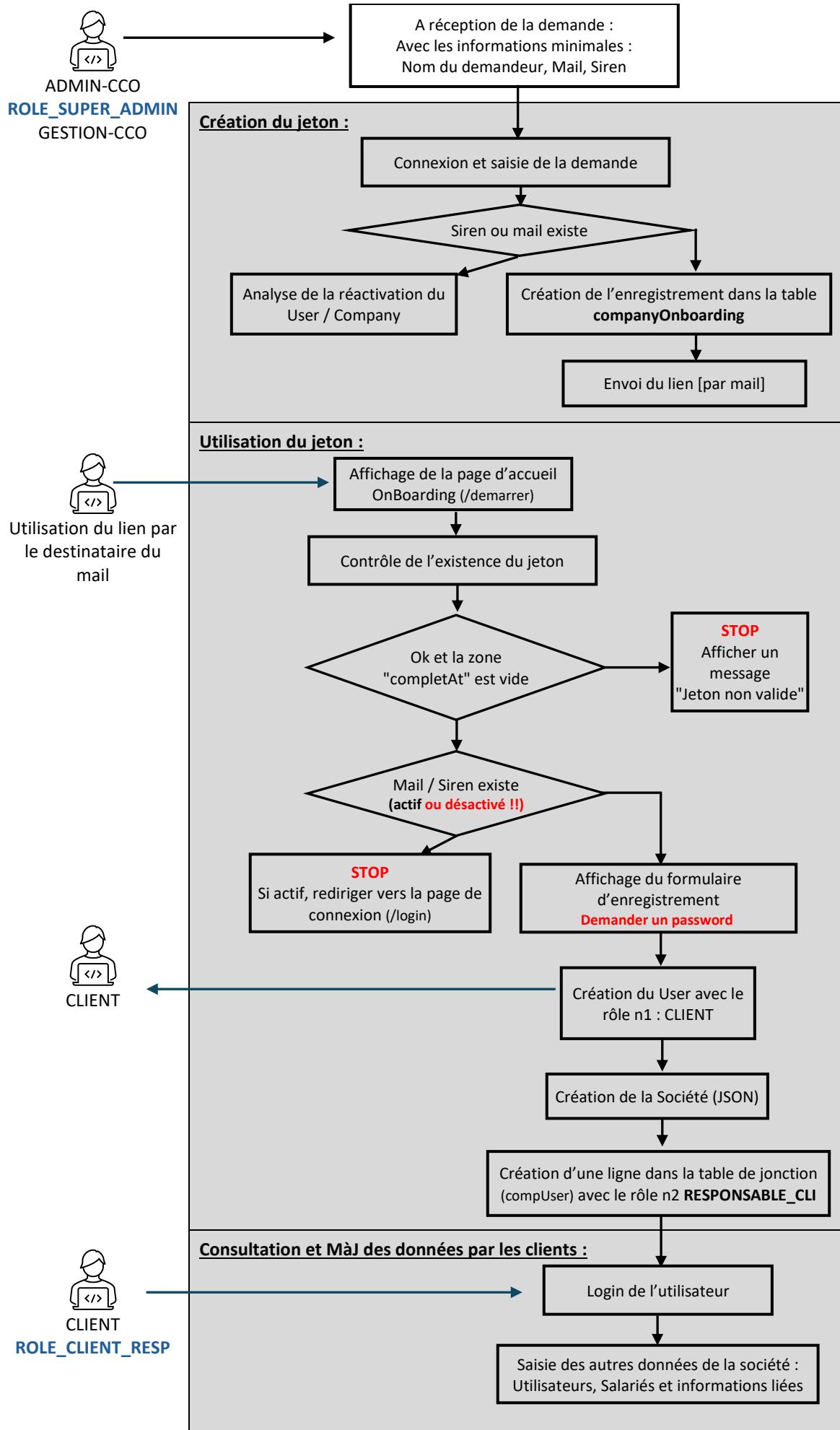




Workflow de création de nouveau compte :

Lors de son utilisation, il y a :

- Contrôle de l'existence du token dans la table CompanyOnboarding.
- Si existe et que la date d'utilisation (completedAt) est vide.
- Création d'un utilisateur avec le rôle (n1) **ROLE_CLIENT**
- Création de la Company.
- Affectation du rôle (n2) **ROLE_CLIENT_RESP** pour la société
- Désactivation du token : avec une date d'utilisation (completedAt).

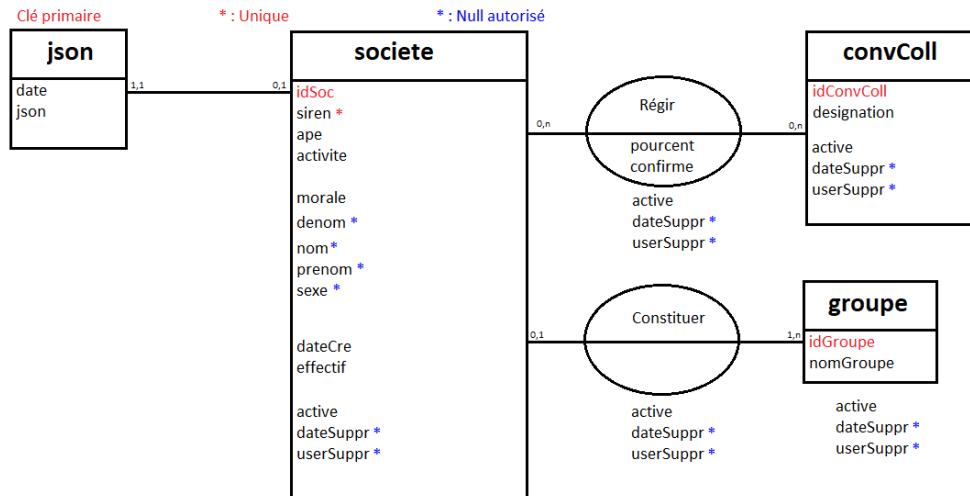


Si le responsable veut créer une autre société, il devra utiliser l'option qui sera proposée dans son tableau de bord ROLE_CLIENT_RESP.

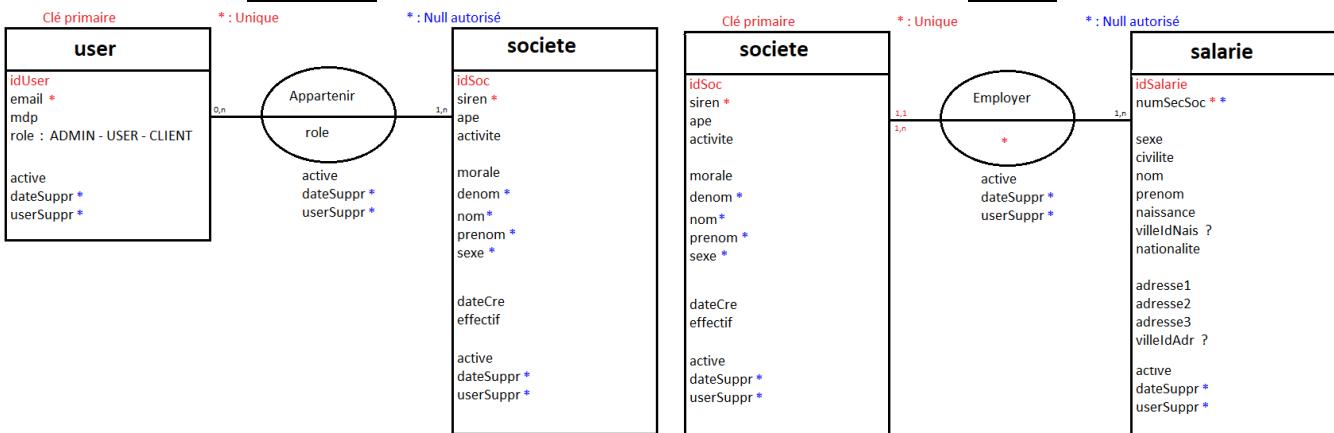
Modèle Conceptuel de Données :

Il s'agit de définir les entités principales et leurs relations avec la cardinalité de chacune.

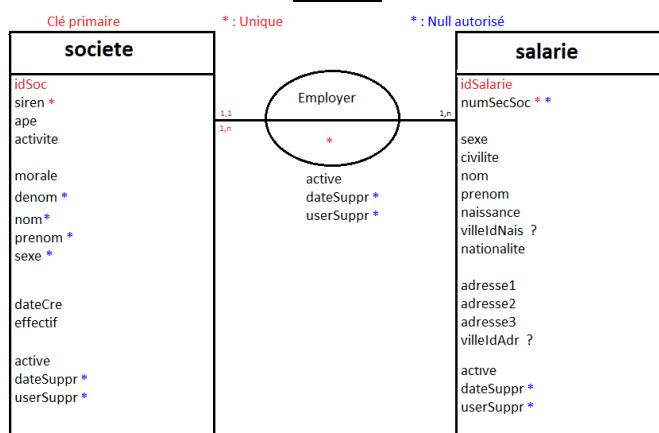
Société :



Utilisateur :



Salarié :



Contrat :

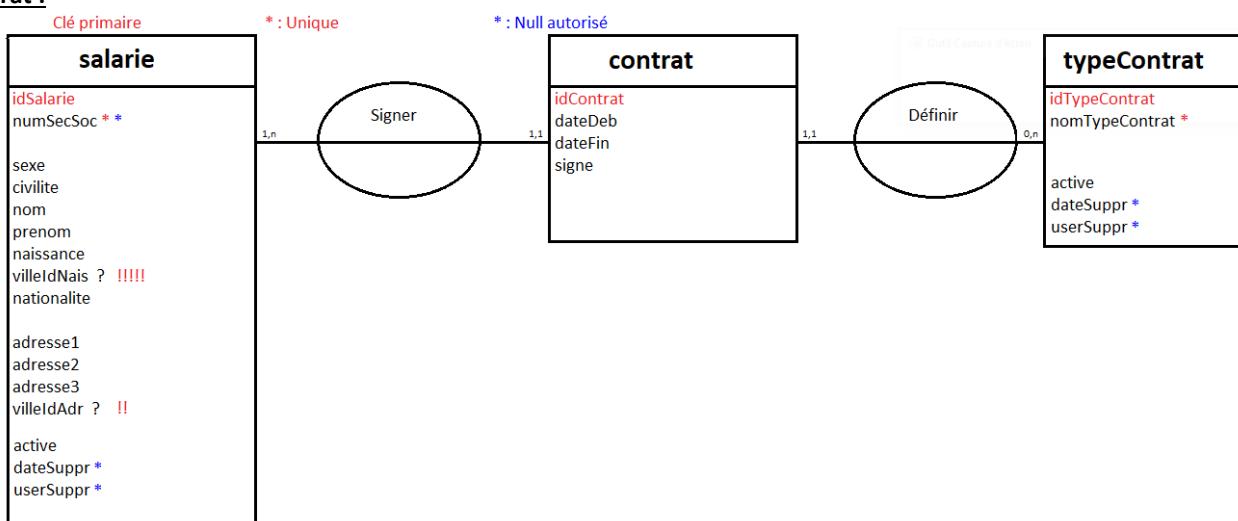
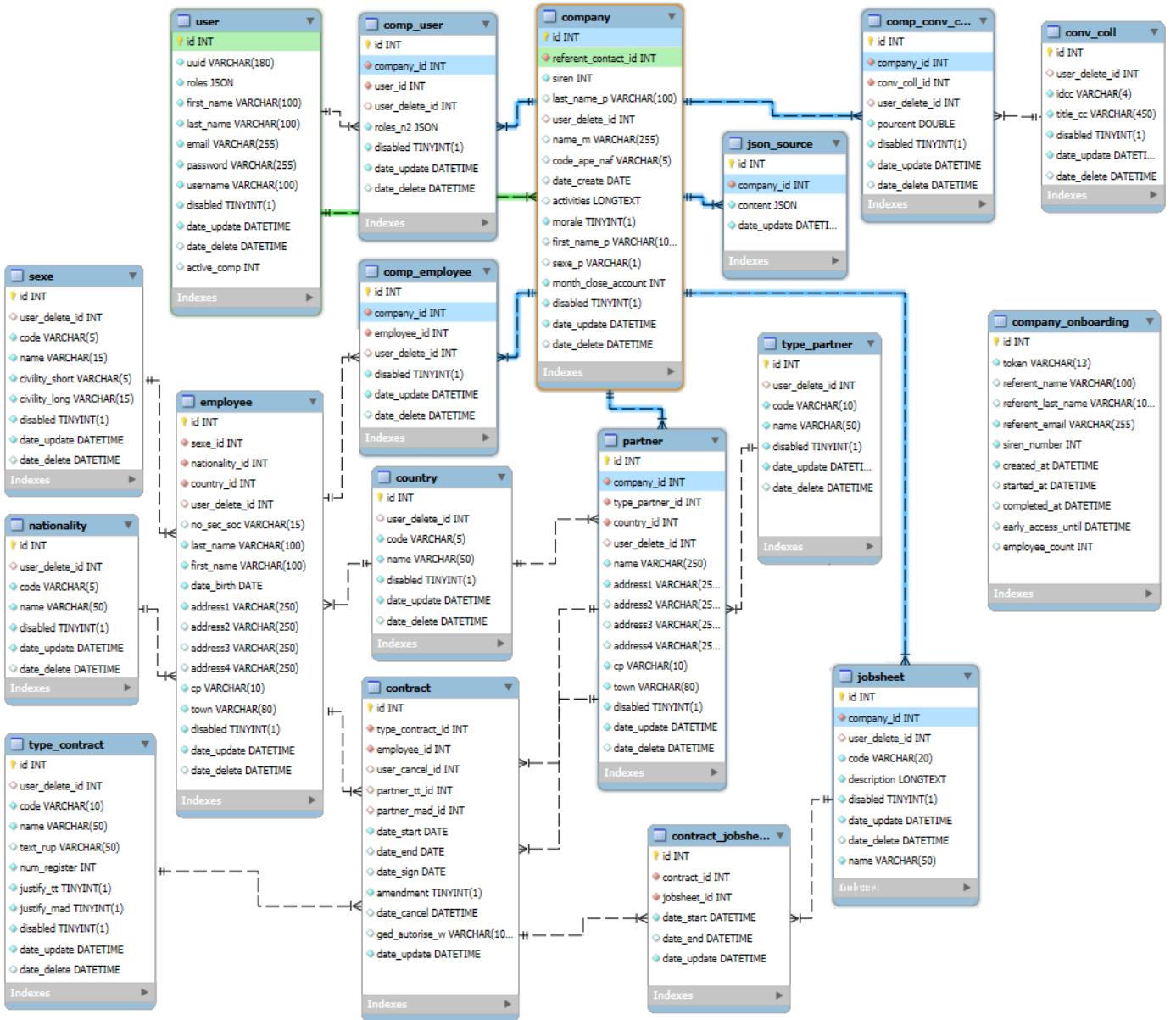
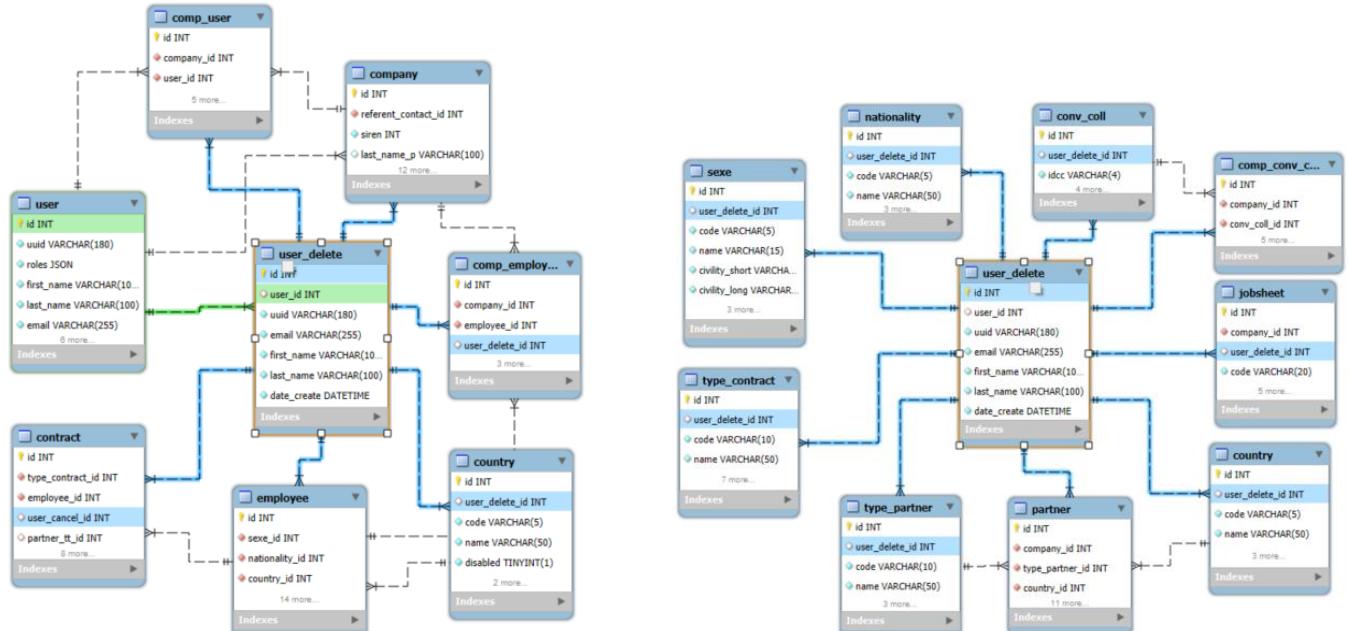


Schéma Workbench :



Liens UserDelete : afin de ne pas alourdir inutilement l'entité User, j'ai décidé de créer une entité UserDelete, liée à User et qui contient les collections traçant l'utilisateur qui a désactivé un enregistrement.



B. Code composant métier

Nous avons vu plus haut dans le frontend, que la construction des pages fait appel à des routes définies dans les contrôleurs.

Ex de CompanyController :

L'accès au contrôleur est conditionné au rôle USER, comme expliqué (un peu plus loin) dans la hiérarchie des rôles définie dans security.yaml, c'est le rôle minimum, dont héritent tous les autres.

Cet attribut (`#[IsGranted('ROLE_USER')]`) permet tout simplement de forcer une connexion pour valider l'accès à la l'url, dans notre cas, `http://localhost:8000/api/company/`.

```
src > Controller > CompanyController.php > CompanyController
 1  <?php
 2
 3  namespace App\Controller;
 4
 5  use App\Entity\Company;
 6  use App\Entity\User;
 7  use App\Repository\CompanyRepository;
 8  use App\Repository\CompEmployeeRepository;
 9
10  #[Route('/api/company')]
11  #[IsGranted('ROLE_USER')]
12  class CompanyController extends GenericController
13  {
14      /**
15      * Routes ouvertes à tous mais sécurisées par rapport au User connecté
16      */
17      /**
18      * Permet récupérer les company de l'utilisateur connecté
19      */
20      #[Route('/get_all_companies_uconnect', name: "get_all_companies_uconnect", methods: [Request::METHOD_GET])]
21      public function getAllCompany(Request $request, CompUserRepository $compUserRepository): JsonResponse
22      {
23          ...
24      }
25
26      /**
27      * Recherche d'une Company sur son id
28      * Appel : /api/company/[Id de la Cie]
29      */
30      #[Route('/{id}', name: "get_company_by_id", methods: [Request::METHOD_GET], requirements: ["id" => Requirement::POSITIVE_INT])
31      public function getOneByIidgetAll(int $id, CompanyRepository $companyRep): JsonResponse
32      {
33          ...
34      }
35
36      /**
37      * Création d'une Cie en BdD
38      * Appel : /api/company/
39      */
40      #[Route('/', name: 'create_company', methods: [Request::METHOD_POST])]
41      #[IsGranted("ROLE_CLIENT_RESP")]
42      public function create(Request $request, SerializerInterface $serializer): JsonResponse
43      {
44          ...
45      }
46
47      /**
48      * Renvoie tous les CompEmployee qui correspondent aux critères facultatifs (disabledEmpl, disabledCe, disabledCie, idEmpl ou idCie)
49      * Appel : /api/company/search_all_comp_empl
50      * ou     /api/company/search_all_comp_empl?disabledEmpl=false&disabledCe=false
51      */
52      #[Route('/search_all_comp_empl', name: "comp_search_all_comp_empl", methods: [Request::METHOD_GET])]
53      public function searchAllCompEmployee(Request $request, CompEmployeeRepository $compEmplRep): JsonResponse
54      {
55          ...
56      }
57
58      /**
59      * Renvoie tous les Jobsheet qui correspondent aux critères facultatifs (disabledEmpl, disabledCe, disabledCie, idEmpl ou idCie)
60      * Appel : /api/company/search_all_comp_jobsheet
61      * ou     /api/company/search_all_comp_jobsheet?disabledJ=true&disabledC=false
62      */
63      #[Route('/search_all_comp_jobsheet', name: "comp_search_all_comp_jobsheet", methods: [Request::METHOD_GET])]
64      public function searchAllJobsheet(Request $request, JobsheetRepository $jobsheetRep): JsonResponse
65      {
66          ...
67      }
68
69      /**
70      * Renvoie les compteurs affichés dans la page Company
71      * CompEmployee actifs et inactifs
72      * JobSheet actifs et inactifs
73      */
74      #[Route('/compteurs', name: "comp_compteurs", methods: [Request::METHOD_GET])]
75      public function compCompteurs(Request $request, CompEmployeeRepository $compEmplRep, JobsheetRepository $jobsheetrep): JsonResponse
76      {
77          ...
78      }
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
```

Toutes les routes ne sont soumises à aucune autre restriction hormis celle de création de la société où l'attribut IsGranted impose le rôle CLIENT_RESP (comme prévu dans la définition des rôles).

```
#[Route('/', name: 'create_company', methods: [Request::METHOD_POST])]
#[IsGranted("ROLE_CLIENT_RESP")]
public function create(Request $request, SerializerInterface $serializer): JsonResponse
```

Ensuite j'ai prévu que chaque fonction associée à une route, puisse être appelée avec ou sans paramètres.

Ex : Récupération des sociétés rattachées à l'utilisateur :

```
*****
 * Routes ouvertes à tous mais sécurisées par rapport au User connecté
 ****
 /**
 * Permet recuperer les company de l'utilisateur connecté
 */
#[Route('/get_all_companies_uconnect', name: "get_all_companies_uconnect", methods: [Request::METHOD_GET])]
public function getAllCompany(Request $request, CompUserRepository $compUserRepository): JsonResponse
{
    /** @var User $user */
    $user = $this->getUser();

    if (count($request->query->all()) === 0){
        $compUsers = $compUserRepository->searchAllCompUser(null, $user->getId());
        //dump ("Sans Disabled");

    } else if( $request->query->has('xorDisabled') && $request->query->get('xorDisabled')){
        // Tous les Users inactifs => $xorDisabled True : on appelle la méthode avec id du User et xorDisabledFalse
        $compUsers = $compUserRepository->searchAllCompUser(null, $user->getId(), true, null, null, null);
        //dump("ICI");
    } else {
        //dump ("Avec Disabled");
        $disabledU = $request->query->has('disabledU')
            ? ($request->query->get('disabledU') === "null" ? null : ($request->query->get('disabledU') === "true"))
            : false;

        $disabledCu = $request->query->has('disabledCu')
            ? ($request->query->get('disabledCu') === "null" ? null : ($request->query->get('disabledCu') === "true"))
            : false;

        $disabledCie = $request->query->has('disabledCie')
            ? ($request->query->get('disabledCie') === "null" ? null : ($request->query->get('disabledCie') === "true"))
            : false;

        $compUsers = $compUserRepository->searchAllCompUser(null, $user->getId(), null, $disabledU, $disabledCu, $disabledCie);
    }

    return $this->json($this->serialize($compUsers, ["get_company", "get_user_delete"]));
}
```

Rappel de l'utilisation dans le code jsx pour la construction de la liste déroulante des company (ici dans paramètres).

```
// fetch CompUser
methodGet("/company/get_all_companies_uconnect")
```

Récupération de la valeur des compteurs :

Ici pas de paramètres sur la route mais des appels successifs à la même méthode avec les valeurs appropriées.

```
/**
 * Renvoie les compteurs affichés dans la page Company
 * CompEmployee actifs et inactifs
 * JobSheet actifs et inactifs
 */
#[Route('/compteurs', name: "comp_compteurs", methods: [Request::METHOD_GET])]
public function compCompteurs(Request $request, CompEmployeeRepository $compEmplRep, JobsheetRepository $jobsheetrep): JsonResponse
{
    /** @var User $user */
    $user = $this->getUser();
    $idCie = $user->getActiveComp();

    // Salariés =====
    // Tous les actifs => Aucun paramètre : on appelle la méthode avec uniquement id de la Company
    $emplActifs = $compEmplRep->countAllCompEmployee($idCie);
    // Tous les inactifs => Ajout de $xorDisabled avec le disabled de chaque entité à null
    $emplInactifs = $compEmplRep->countAllCompEmployee($idCie, null, true, null, null);

    // Jobsheet =====
    // Toutes les fiches de poste actives
    $result = $jobsheetrep->searchAlljobsheet($idCie);
    $jobActifs = count($result);
    // Toutes les fiches de poste inactives
    $result = $jobsheetrep->searchAlljobsheet($idCie, null, true, null, null);
    $jobInactifs = count($result);

    $results = [
        'emplActifs' => $emplActifs,
        'emplInactifs' => $emplInactifs,
        'jobActifs' => $jobActifs,
        'jobInactifs' => $jobInactifs
    ];

    return new JsonResponse(json_encode($results));
}
```

C. Code composant d'accès aux données :

L'utilisation de paramètres dans les contrôleurs, n'est possible que parce ce que la méthode créée dans le repository, prévoit les différents cas.

```

src > Repository > CompUserRepository.php > CompUserRepository
1  <?php
2
3  namespace App\Repository;
4
5  use App\Entity\CompUser;
6  use Doctrine\Bundle\DoctrineBundle\Repository\ServiceEntityRepository;
7  use Doctrine\Persistence\ManagerRegistry;
8
9  /**
10  * @extends ServiceEntityRepository<CompUser>
11  */
12 class CompUserRepository extends ServiceEntityRepository
13 {
14     public function __construct(ManagerRegistry $registry)
15     {
16         parent::__construct($registry, CompUser::class);
17     }
18
19     /**
20      * Récupère les données de CompUser selon les paramètres de désactivation (Tous si paramètre null, ou actif ou désactivé selon le paramètre).
21      *
22      * @param int|null $idUser Filtre sur un utilisateur, (si null = tous)
23      * @param int|null $idCie Filtre sur une société, (si null = toutes)
24      *
25      * @param int|null $xorDisabled permet de sélectionner les CompEmployee inactifs (au moins un des témoins disabled ci-dessous est à true)
26      *
27      * @param bool|null $disabledUser Filtre sur l'état de désactivation de l'utilisateur, ( si null = tous). Par défaut: false
28      * @param bool|null $disabledCu Filtre sur l'état de désactivation de CompUser, (si null = tous). Par défaut: false
29      * @param bool|null $disabledCie Filtre sur l'état de désactivation de la société, (si null = tous). Par défaut: false
30      *
31      * @return CompUser[] Retourne un tableau d'objets CompUser. Le tableau peut être vide si aucun résultat n'est trouvé.
32      */
33     public function searchAllCompUser (?int $idCie = null, ?int $idUser = null,
34                                         ?bool $xorDisabled = null,
35                                         ?bool $disabledUser = false, ?bool $disabledCu = false, ?bool $disabledCie = false) : array
36     {

```

Tous les paramètres sont optionnels ce qui permet d'effectuer différentes recherches en fonction de ceux-ci.
Dans l'exemple nous récupérons les CompUser

- Tous les CompUser :**

- D'une société : \$idCie est renseigné et \$idUser est null,
- D'un utilisateur : \$idCie = null et \$idUser est renseigné
- D'un utilisateur et d'une société : \$idCie et \$idUser sont renseignés,
- De toutes les sociétés et de tous les utilisateurs : \$idCie et \$idUser sont null,

- Actifs ou inactifs :** selon les paramètres \$xorDisabled, \$disabledUser, \$disabledCu) et \$disabledCie
 - Les valeurs par défaut : ont été choisies pour renvoyer les données actives
 - Inactifs : \$ xorDisabled = true,
 - Personnalisés : on teste la valeur de chaque paramètre envoyé.

```

49   $qb = $this->createQueryBuilder('cu')
50       ->select('cu')
51       ->innerJoin('cu.user', 'u')
52       ->innerJoin('cu.company', 'c');

```

Seuls les CompUser sont repris mais il faut joindre les tables User et Company pour pouvoir tester leur flag "disabled" respectifs.

```

// Sélection sur 'id User' =====
if (!empty($idUser)) {
    $qb->andWhere('u.id = :idUser')
        ->setParameter('idUser', $idUser);
}
// et 'id Company'
if (!empty($idCie)) {
    $qb->andWhere('c.id = :idCie')
        ->setParameter('idCie', $idCie);
}

// Sélection sur les flags 'disabled' =====
if (!is_null($xorDisabled) && $xorDisabled){
    $qb->andWhere($qb->expr()->OrX(
        $qb->expr()->eq('u.disabled', 'true'),
        $qb->expr()->eq('cu.disabled', 'true'),
        $qb->expr()->eq('c.disabled', 'true'))
    );
} else {
    if (!is_null($disabledUser) ) {
        $qb->andWhere('u.disabled = :disabledU')
            ->setParameter('disabledU', $disabledUser);
    }
    if (!is_null($disabledCu) ) {
        $qb->andWhere('cu.disabled = :disabledCu')
            ->setParameter('disabledCu', $disabledCu);
    }
    if (!is_null($disabledCie) ) {
        $qb->andWhere('c.disabled = :disabledCie')
            ->setParameter('disabledCie', $disabledCie);
    }
}

```

Construction de la clause WHERE en fonction de la valeur de chaque paramètre.

```

        if(empty($idCie)){
            $qb->orderBy('c.NameM', 'ASC')
                ->orderBy('c.firstNameP', 'ASC')
                ->orderBy('c.firstNameP', 'ASC');
        }

```

```

        if(empty($idUser)){
            $qb->orderBy('u.lastName', 'ASC')
                ->orderBy('u.firstName', 'ASC');
        }

```

On trie les enregistrements.

```

        return $qb->getQuery()->getResult();
    }

```

Renvoi du contenu du select : tous les CompUsers correspondants aux critères
Ce type de recherche a été créé dans tous les repositories.

D. Les tests unitaires :

Ces tests vérifient la validité des données avant la création de l'instances de la classe, bloquant et signalant les erreurs en cas de données invalides.

Ils s'appuient sur des assertions définies dans les entités, parfois au niveau de la classe, mais le plus souvent sur chaque propriété.

Un fichier de test est ensuite créé avec des jeux de données pour vérifier le respect ou le non-respect de chaque assertion.

Etapes de la réalisation des test unitaires :

1. Poser des assertions dans les entités afin de valider la cohérence des données.
2. Créer d'un fichier test par Entité qui teste des ajouts de données

```

ContractTest.php
1 <?php
2
3 namespace App\Tests\Entity;
4
5 use App\Entity\Contract;
6 use App\Entity\Employee;
7 use App\Entity\Partner;
8 use App\Entity\TypeContract;
9 use App\Entity\UserDelete;
10 use DateTime;
11
12 class ContractTest extends \App\Tests\KernelTestCase
13 {
14     // Déclaration des propriétés de la classe
15     private string $Texte50 = "0123456789-0123456789-0123456789-0123456789-123456";
16
17     public function setUp(): void
18     {
19         parent::setUp();
20     }
21
22     //***** Jeux de données INVALIDES *****
23     // Jeux de données INVALIDES -----
24     public function testInvalid(): void
25     {
26         // -----
27         // NULL : typeContract - employee - dateStart
28         // Toutes les zones -----
29         $test = (new Contract())
30             ->setTypeContract(new TypeContract())
31             ->setEmployee(new Employee())
32             ->setDateStart(new DateTime());
33
34         $this->assertHasErrors($test, 3);
35
36         // Longueur max =====
37         // -----
38         $value = str_repeat($this->Texte50, 2) . "A";
39         $test = (new Contract())
40             ->setGedAutoriseW($value)
41             ->setTypeContract(new TypeContract())
42             ->setEmployee(new Employee())
43             ->setDateStart(new DateTime());
44         $this->assertHasErrors($test, 1);
45     }
46 }

EmployeeTest.php
1 //***** Jeux de données VALIDES *****
2 // Jeux de données VALIDES -----
3 public function testValid(): void
4 {
41     // Que les zones Obligatoires =====
42     $test = (new Contract())
43         ->setTypeContract(new TypeContract())
44         ->setEmployee(new Employee())
45         ->setDateStart(new DateTime());
46     $this->assertHasErrors($test, 0);
47
48     // Longueur max =====
49     $value = str_repeat($this->Texte50, 2);
50     //dump(strlen($value));
51     $test = (new Contract())
52         ->setGedAutoriseW($value)
53         ->setTypeContract(new TypeContract())
54         ->setEmployee(new Employee())
55         ->setDateStart(new DateTime());
56     $this->assertHasErrors($test, 0);
57
58     // Toutes les zones -----
59     $test = (new Contract())
60         ->setTypeContract(new TypeContract());
61
62     //dump($test);
63
64     //assertHasErrors($test, 0);
65 }

```

3. De lancer la commande d'exécution de ces tests afin de valider qu'aucune assertion n'est enfreinte.

KO : des assertions ne sont pas respectées	OK : Toutes les assertions sont respectées
<pre> PS C:\Users\Utilisateur\Documents\cco_registro> php bin/phpunit tests\entity\EmployeeTest.php PHPUnit 9.6.21 by Sebastian Bergmann and contributors. Testing C:\Users\Utilisateur\Documents\cco_registro\tests\entity\EmployeeTest.php 30 / 30 (100%) Time: 00:05.214, Memory: 46.00 MB OK (30 tests, 479 assertions) PS C:\Users\Utilisateur\Documents\cco_registro> </pre>	<pre> PS C:\Users\Utilisateur\Documents\cco_registro> php bin/phpunit tests\entity\EmployeeTest.php PHPUnit 9.6.21 by Sebastian Bergmann and contributors. Testing C:\Users\Utilisateur\Documents\cco_registro\tests\entity\EmployeeTest.php 30 / 30 (100%) Time: 00:05.214, Memory: 46.00 MB OK (30 tests, 479 assertions) PS C:\Users\Utilisateur\Documents\cco_registro> </pre>
<pre> Failed asserting that actual size 13 matches expected size 12. C:\Users\Utilisateur\Documents\cco_registro\tests\KernelTestCase.php:38 C:\Users\Utilisateur\Documents\cco_registro\tests\entity\EmployeeTest.php:76 C:\Users\Utilisateur\Documents\cco_registro\vendor\phpunit\phpunit\phpunit:107 FAILURES! Tests: 30, Assertions: 479, Failures: 1. </pre>	<p>Ci-contre on voit qu'à la ligne 76 du fichier EmployeeTest.php, il y a 13 erreurs alors qu'on en attendait seulement 12</p> <p>Une assertion n'est pas respectée.</p>

76 \$this->assertHasErrors(\$test, 12);

Les assertions :

Comme signalé, elles sont définies dans les entités. Elles permettent de contrôler que la donnée fournie a bien le format et la structure attendus.

```
9     use Symfony\Component\Validator\Constraints as Assert;
```

Ces contrôles permettent de s'assurer que les données saisies sont cohérentes et valides avant leur traitement ou leur stockage. Il est possible de contrôler si le champ est vide, sa taille maximale, son format (grâce à une expression régulière, une liste ou un intervalle de valeurs) etc.

Un message d'erreur personnalisé peut être défini. Il s'affiche lorsqu'une assertion n'est pas respectée.

Ci-dessous quelques exemples courants d'assertions posées sur des propriétés.

```
#[ORM\Column(type: Types::INTEGER, options: ["unsigned" => true], unique: true)]
#[Assert\NotBlank(message: "Le 'Siren' ne peut pas être vide")]
#[Assert\Range(
    min: 100000000,
    max: 999999999,
    minMessage: "La valeur doit être supérieure ou égale à {{ limit }}",
    maxMessage: "La valeur doit être inférieure ou égale à {{ limit }}",
    notInRangeMessage: "Le 'Siren' doit être composé de 9 chiffres et ne doit pas commencer par 0"
)]
#[Groups(["get_company"])]
private ?int $siren;

#[ORM\Column(length: 255, unique: true, nullable: true)]
#[Assert\Length(max: 255, maxMessage: "Le 'nom de la société' ne peut pas dépasser {{ limit }} caractères")]
#[Groups(["get_company"])]
private ?string $nameM = null;

#[ORM\Column(length: 5, unique: true, nullable: true)]
#[Assert\Length(
    exactly: 5,
    exactMessage: "Le 'code APE/NAF' doit contenir exactement {{ limit }} caractères"
)]
#[Assert\Regex(
    pattern: '/^1[0-9][0-9]{3}[A-Z]$/',
    message: "Le 'code APE/NAF' doit être composé de 4 chiffres (ne commençant pas par 0) suivis d'une lettre majuscule"
)]
#[Groups(["get_company"])]
private ?string $codeApeNaf = null;

#[ORM\Column]
#[Assert\NotNull(message: "Le 'Mois de clôture' ne peut pas être vide")]
#[Assert\Range(
    min: 0,
    max:12,
    notInRangeMessage: "Vous devez saisir le 'NUMERO du mois' correspondant à la clôture de l'exercice,\n ou 0 (zéro) si vous ne le connaissez pas."
)]
#[Groups(["get_company"])]
private ?int $monthCloseAccount = null;

#[ORM\Column(length: 1, nullable: true)]
#[Assert\Choice(
    choices: ['M', 'F'],
    message: "Le 'Sexe' doit être 'M' ou 'F'."
)]
#[Groups(["get_company"])]
private ?string $sexP = null;

#[ORM\Column(length: 15, nullable: true)]
#[Assert\Regex(
    pattern: "/^1[0-9][0-9]{13}$/,
    message: "Le 'Nº de Sécurité Sociale' doit commencer par 1, 2 ou 3 et avoir 15 chiffres au total"
)]
#[Groups(["get_empl"])]
private ?string $noSecSoc = null;
```

```

#[ORM\Column(length: 255, unique: true)]
#[Assert\NotBlank(message: "La zone 'Email' ne peut pas être vide")]
#[Assert\Length(max: 255, maxMessage: "La zone 'Email' ne peut pas dépasser {{ limit }} caractères")]
#[Assert\Email(message: "Le format de la zone 'Email' est incorrect")]
#[Groups(["get_user", "deserialize"])]
private ?string $email = null;

#[ORM\Column]
#[Assert\Count(
    min: 1,
    minMessage: "Au moins un 'Rôle' au sein de la société doit être affecté à l'utilisateur."
)]
#[Groups(["get_company"])]
private array $rolesN2 = [];

#[ORM\ManyToOne(inversedBy: 'compUsers')]
#[ORM\JoinColumn(nullable: false)]
#[Assert\NotNull(message: "La 'Société' ne peut pas être nulle.")]
#[Groups(["get_company"])]
private ?Company $company = null;

```

Et puis parfois il faut poser les assertions directement sur l'entité. Par exemple ici, pour vérifier l'unicité de valeur dans les zones codeApeNaf et siren de la table Company.

```

#[ORM\Entity(repositoryClass: CompanyRepository::class)]
#[UniqueEntity(
    fields: ['codeApeNaf'],
    message: 'Ce `code APE/NAF` existe déjà. Veuillez en saisir un autre.')
]
#[UniqueEntity(
    fields: ['siren'],
    message: 'Ce `code Siren` existe déjà.')
]
class Company
{

```

Eléments de sécurité

A. Routes sécurisées

Nous avons vu que la sécurisation de l'accès aux routes peut se faire sur le rôle (IsGranted), qui utilise la hiérarchisation des rôles définie dans **security.yaml**.

security.yaml :

```

! security.yaml ×
config > packages > ! security.yaml
  1   security:
  2     role_hierarchy:
  3       # Rôles N1 : User.roles
  4       ROLE_USER: []
  5       ROLE_CLIENT: ROLE_USER
  6       ROLE_GESTION: ROLE_CLIENT
  7       # Rôles N2 : CompUser.rolesN2
  8       ROLE_CLIENT_COMPTA: []
  9       ROLE_CLIENT_RH: ROLE_CLIENT_COMPTA
 10      ROLE_CLIENT_RESP: ROLE_CLIENT_RH
 11      | # Rôles N1 : User.roles - SUPER_ADMIN
 12      ROLE_SUPER_ADMIN: [ROLE_GESTION,ROLE_CLIENT_RESP]
 13      # https://symfony.com/doc/current/security.html#registering-a-new-user-provider

```

GESTION hérite des droits de de **CLIENT** qui hérite de **USER**.

RESP hérite des droits de **RH** qui hérite de de **COMPTA** qui hérite de **USER**.

SUPER_ADMIN hérite de **GESTION** et de **RESP**.

Blocage dans les controllers :

Par rapport au rôle

Avec l'utilisation de IsGranted

```
/**
 * Création d'une Cie en BdD
 * Appel : /api/company/
 */
#[Route('/', name: 'create_company', methods: [Request::METHOD_POST])]
#[IsGranted('ROLE_CLIENT_RESP')]
public function create(Request $request, SerializerInterface $serializer): JsonResponse
{
    // Ajout du dernier paramètre ["get_company"] car sinon => pb de référence circulaire
    $company = $serializer->deserialize($request->getContent(), Company::class, 'json', ['get_company']);
    $this->setEntityRelation($company, "referentContact", json_decode($request->getContent(), true), User::class);

    /** @var int $id */
    $id = $this->validateAndPersist($company, true);

    return $this->json($id, Response::HTTP_CREATED);
}
```

Par rapport à la société en cours :

Dans le code ci-dessous, on commence par récupérer les données de l'utilisateur connecté \$user = \$this->getUser(). Et ensuite, on vérifie que la valeur de sa propriété (activeComp) correspond à l'id envoyé en paramètre.

```
/**
 * Recherche d'une Company sur son id
 * Appel : /api/company/[Id de la Cie]
 */
#[Route('/{id}', name: "get_company_by_id", methods: [Request::METHOD_GET], requirements: ['id' => Requirement::POSITIVE_INT])
public function getOneByIdGetAll(int $id, CompanyRepository $companyRep): JsonResponse
{
    /** @var User $user */
    $user = $this->getUser();

    if ($id !== $user->getActiveComp())
        throw new NoResultException(sprintf("Seules les informations de votre société sont accessibles."));

    $company = $companyRep->find($id);

    return $this->json($this->serialize($company, ['get_company', 'get_convColl', 'get_jsonDate']));
}
```

Ou plus simplement :

Plutôt que d'attendre un paramètre dans la route, on récupère la donnée dans le contrôleur. Ici la société active de l'utilisateur connecté \$user = \$this->getUser().

```
/**
 * Renvoie tous les CompEmployee qui correspondent aux critères facultatifs (disabledEmpl, disabledCe, disabledCie, idEmpl ou idCie)
 * Appel : /api/company/search_all_comp_empl
 * ou      /api/company/search_all_comp_empl?disabledEmpl=false&disabledCe=false
 */
#[Route('/search_all_comp_empl', name: "comp_search_all_comp_empl", methods: [Request::METHOD_GET])]
public function searchAllCompEmployee(Request $request, CompEmployeeRepository $compEmplRep): JsonResponse
{
    /** @var User $user */
    $user = $this->getUser();
    $idCie = $user->getActiveComp();

    if (count($request->query->all()) === 0) {
        // Tous les salariés actifs => Aucun paramètre : on appelle la méthode avec uniquement id de la Company
        $results = $compEmplRep->searchAllCompEmployee($idCie);
    } else if( $request->query->has('xorDisabled') && $request->query->get('xorDisabled')){
        // Tous les Salariés inactifs => xorDisabled True : on appelle la méthode avec id de la Company et sorDisabledFalse
        $results = $compEmplRep->searchAllCompEmployee($idCie, null, true, null, null, null);
    } else {
        // Tous les salariés inactifs => xorDisabled False : on appelle la méthode avec id de la Company et sorDisabledTrue
        $results = $compEmplRep->searchAllCompEmployee($idCie, null, false, null, null, null);
    }
}
```

Cas particulier de la fiche profil :

La route n'est pas protégée car il faut permettre à tous les utilisateurs de modifier leur profil.

```
#Route('/{id<\d+>}', name: "user_update", methods: ["PATCH"])
public function update(Request $request, ?User $user, ?int $id, CompUserRepository $compUserRep): JsonResponse
{
    // 1-On vérifie si l'utilisateur existe
    if (empty($user))
        throw new NoResultException(sprintf("Aucun utilisateur n'a été trouvé avec l'id %d", $id));

    /** @var User $userConnect */
    $userConnect = $this->getUser();
```

On vérifie que la fiche à modifier est bien rattachée à la société en cours.

```
// 2- Bloquer si le UserMaj n'est pas associé à la Company en cours
$companyCours = $userConnect->getActiveComp();
$compUsersMaj = $compUserRep->searchAllCompUser($companyCours, $user->getId());
if (empty($compUsersMaj)) {
    throw new NoResultException(sprintf("il n'y a pas d'utilisateur avec l'id %d", $id));
```

Ensuite, dès lors que le profil demandé n'est pas celui de la personne connectée, on contrôle les droits du demandeur.

Si c'est un Super Admin => feu vert

```
// 3- Traitement supplémentaire si $user est différent de l'utilisateur connecté
// Il faut être soit ROLE_SUPER_ADMIN dans User.roles, soit au minimum ROLE_CLIENT_RH dans CompUser.rolesN2
if ($user->getId() !== $userConnect->getId()) {
    // UserConnect SUPER_ADMIN => Ok
    $ctrlOk = $this->isGranted('ROLE_SUPER_ADMIN') ? true : false;
```

Si c'est un rôle CLIENT, on recherche son rôle au sein de la société

```
if (!$ctrlOk && $this->isGranted('ROLE_CLIENT')) {
    // Récupération du/des rolesN2 du User connecté sur la société en cours
    $compUsersConnect = $compUserRep->searchAllCompUser($companyCours, $userConnect->getId());
```

Si la fiche à modifier est celle d'un responsable, il faut être aussi responsable pour y accéder.

```
// =====
// Seul un profil RESP peut modifier un autre profil RESP => Contrôle spécifique
// User Maj
$majHasRoleResp = false;
// Vérifier si l'un des $compUsersMaj contient 'ROLE_CLIENT_RESP'
foreach ($compUsersMaj as $compUser) {
    if (in_array('ROLE_CLIENT_RESP', $compUser->getRolesN2())) {
        $majHasRoleResp = true;
        break;
    }
}
if ($majHasRoleResp) {      // Le profil à Maj est RESP
    // Vérifier si l'un des $compUsersMaj contient 'ROLE_CLIENT_RESP'
    foreach ($compUsersConnect as $compUser) {
        if (in_array('ROLE_CLIENT_RESP', $compUser->getRolesN2())) {
            $ctrlOk = true;
            break;
        }
    }
}
```

Sinon, on autorise la modification aux RH.

```
// =====
} else {                  // Le profil à Maj n'est pas RESP
    foreach ($compUsersConnect as $compUser) {
        // Utilisation du service RolesN2Voter.php
        if ($this->isGranted('rolesN2', [$compUser, ['ROLE_CLIENT_RH']]))) {
            $ctrlOk = true;
            break;
        }
    }
}
```

Affichage d'élément conditionné aux rôles :

Sociétés et Rôles : (3)

	RESP	RH	CPT
Soc 1	X		
Soc 2		X	
Soc 1 firstNameP			X

Cet utilisateur est Responsable dans la Soc 1, RH dans la Soc 2 et Comptable dans la société firsNameP.

La variable d'état **ajoutEmplOk** permet de trouver le CompUser correspondant à la société actuelle. La fonction fléchée (cu)=> examine chaque élément jusqu'à trouver une correspondance. Dès qu'un élément satisfait aux critères, son contenu est renvoyé et la recherche prend fin.

```
// Affichage du bouton d'ajout d'un salarié
const ajoutEmplOk = () => {
  if (!compUsers || !selectedCompanyId) {
    return false;
  }

  const user = compUsers.find(
    (cu) => cu.company.id === selectedCompanyId &&
      (cu.rolesN2.includes('ROLE_CLIENT_RESP') || cu.rolesN2.includes('ROLE_CLIENT_RH'))
  );

  return !!user; // Retourne true si un utilisateur répondant aux critères spécifiés est trouvé, false sinon
};
```

Ensuite il suffit de conditionner l'affichage de l'élément au fait que **ajoutEmplOk** soit vrai.

```
<th className="px-2 pt-1 font-bold text-left">Natio-</th>
<th rowspan="2" className="text-center">
  ajoutEmplOk() && (
    <p className="text-gray-800 hover:text-blue-600">
      <svg ...>
      </svg>
    </p>
  )
</th>
</tr>
```

Cela nous donne le résultat suivant :

Le bouton permettant d'ajouter un nouveau salarié n'est disponible que pour le responsable et le RH.

The screenshot shows a user interface for managing employees across different companies. At the top, there are three company cards: "Soc 1", "Soc 2", and "firstNameP Soc 1". Each card has a header with a bell icon and a "SR" button. The "SR" button is highlighted in orange for "Soc 1", blue for "Soc 2", and white for "firstNameP Soc 1". Below each header is a table with columns "Nom" and "Prénom". Under "Soc 1", the table shows "Salarié 1". Under "Soc 2", the table shows "RA" and "...". Under "firstNameP Soc 1", the table shows "RA" and "...". To the right of the tables, there is a "+" button.

Posture réflexive.

Mon choix sur la demande de stage auprès de la société CCO était murement réfléchi. J'espérais que mon expérience dans le développement, la maintenance et l'évolution d'ERP serait un plus. Le choix s'est avéré judicieux.

La pertinence du projet proposé par CCO, combinée au soutien de l'équipe et aux ressources mises à disposition, a créé un environnement favorable à mon apprentissage et à ma contribution.

Si j'ai travaillé en autonomie tout au long de l'analyse, j'ai bénéficié d'un soutien précieux lors de la phase de développement, particulièrement pour la partie front-end. L'expertise et la disponibilité de David et Nicolas ont été des atouts majeurs dans cette étape. Cédric a su prendre le temps de faire des points nécessaires afin valider les avancées et réorienter les propositions qui ne correspondaient pas aux attentes. Cette approche m'a permis de progresser efficacement, sans jamais me trouver dans l'impasse par manque de directives.

Malgré l'abandon de la structure initiale du tableau de bord, les fondations posées et les pages créées ont constitué une base solide pour mener à bien mon travail.

Si je devais apporter une seule modification à mon approche, je commencerais le développement du front-end immédiatement après les tests unitaires. Cela aurait probablement évité la création de routes superflues dans les contrôleurs, optimisant ainsi le temps de développement. Cette expérience bien que perfectible, s'est révélée enrichissante et m'a permis d'affiner ma méthodologie pour de futurs projets.

Pour conclure, la qualité du projet proposé, couplée à l'accueil chaleureux et à l'expertise de l'équipe, a considérablement enrichi ces 11 semaines de stage. Cette expérience m'a permis de mettre en pratique mes compétences, tout en bénéficiant d'un environnement propice à l'apprentissage du développement Web.

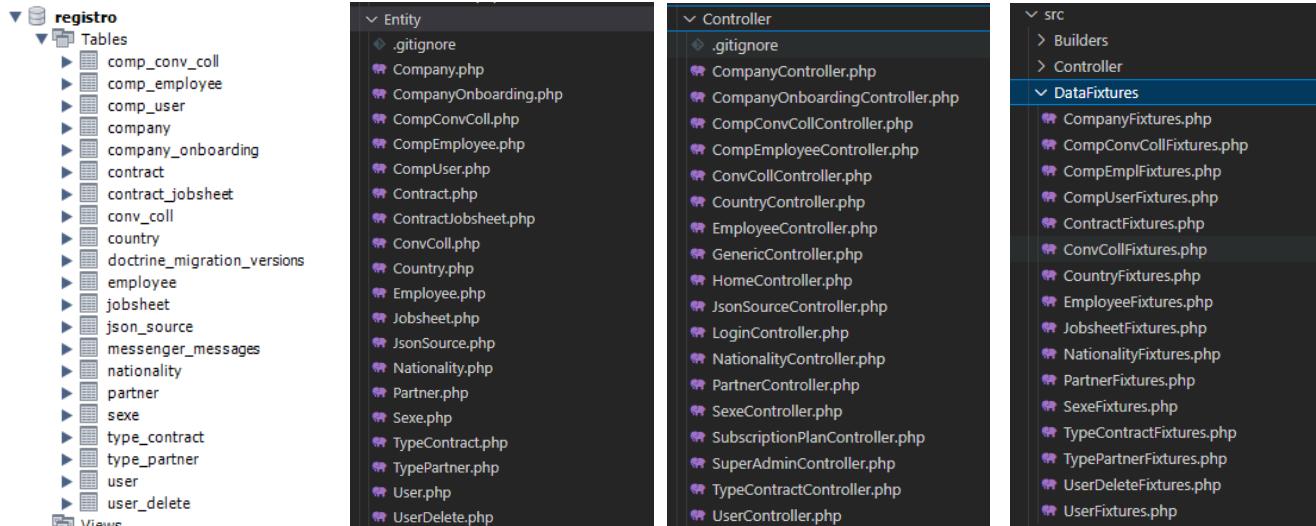
La combinaison de ces facteurs a maximisé la valeur de cette période, la rendant à la fois productive et enrichissante pour mon parcours. De son côté CCO retire une structure de base de données cohérente par rapport à ses attentes et saura amener le développement de l'outil à son terme

Le projet proposé ainsi que l'équipe, accueillante et compétente, nous ont permis de valoriser au mieux ces 11 semaines de stage.

Et donc merci à toute l'équipe CCO pour cette expérience concluante.

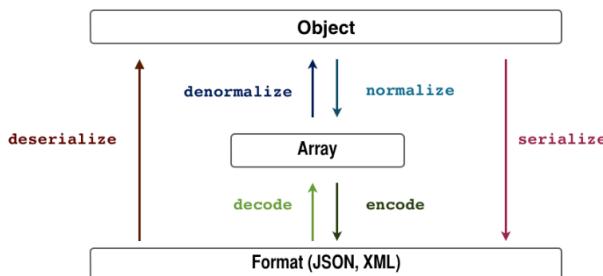
Annexes

A. Les tables – Entités – Controller - Fixtures :



B. Schéma serialize / deserialize :

Schéma tiré de la documentation Symfony



C. Méthodes des repository communes :

```

    CompanyController.php
    ...
    ...$compUsers = $compUserRepository->searchAllCompUser(null, $user->getId());
    ...$compUsers = $compUserRepository->searchAllCompUser(null, $user->getId(), true, null, null);
    ...$compUsers = $compUserRepository->searchAllCompUser(null, $user->getId(), null, $disabledCu, $disabledCie);
    public function searchAllCompUser(Request $request, CompUserRepository $compUserRepository): JsonResponse
    ...
    ...$results = $compUserRepository->searchAllCompUser();
    ...$results = $compUserRepository->searchAllCompUser($idCie, $idUser, null, $disabledUser, $disabledCu, $disabledCie);
    ...$results = $compUserRepository->searchAllCompUser($idCie, $idUser, $disabledUser, $disabledCu, $disabledCie);

    EmployeeController.php
    ...
    ...$compUsers = $compUserRep->searchAllCompUser($user->getActiveComp(), $user->getId());

    UserController.php
    ...
    ...$compUsersMaj = $compUserRep->searchAllCompUser($companyCours, $user->getId());
    ...$compUsersConnect = $compUserRep->searchAllCompUser($companyCours, $userConnect->getId());
    ...$activeCompUser = $compUserRep->searchAllCompUser(null, $user->getId(), null, null, false);
    ...$activeCompUser = $compUserRep->searchAllCompUser(null, $user->getId(), false, false, false);
    public function searchAllCompUser(Request $request, CompUserRepository $compUserRep): JsonResponse
    $results = $compUserRep->searchAllCompUser();
    $results = $compUserRep->searchAllCompUser($idCie, $idUser, null, $disabledUser, $disabledCu, $disabledCie);

    CompUserRepository.php
    ...
    public function searchAllCompUser (?int $idCie = null, ?int $idUser = null)

```

La méthode **searchAllCompUser** du **CompUserRepository** est utilisée dans 3 contrôleur et dans ceux-ci elle est appelée dans plusieurs routes (ex **CompanyController** : 2 routes)

```

    CompanyController.php
    ...
    #[Route('/get_all_companies_uconnect', name: 'get_all_companies_uconnect',
    public function getAllCompany(Request $request, CompUserRepository $compus
    {
        /**
         * @var User $user */
        $user = $this->getUser();

        if (count($request->query->all()) === 0){
            $compUsers = $compUserRepository->searchAllCompUser(null, $user->g
            //dump ("Sans Disabled");
        }

        else if ($request->query->has('xorDisabled') && $request->query->get
        // Tous les Users inactifs => xorDisabled True : on appelle la mè
        $compUsers = $compUserRepository->searchAllCompUser(null, $user->g
        //dump ("XOR");
    }
}

    UserController.php
    ...
    #[Route('/search_all_comp_users', name: "company_search_all_comp_users", me
    #[IsGranted("ROLE_SUPER_ADMIN")]
    public function searchAllCompUser(Request $request, CompUserRepository $com
    {
        // Vérifie si aucun paramètre n'est envoyé
        if (count($request->query->all()) === 0) {
            // Aucun paramètre : on appelle la méthode sans argument
            $results = $compUserRepository->searchAllCompUser();
        } else {
            ...
        }
    }
}

```

D. Extrait du dictionnaire de données :

Company :

Nom	Libellé	Format	Taille	Unique	Null autor.	FK	Complément	Json
<code>id</code>	Identifiant	AI						
<code>referentContact</code>	Id User	INT			x		Identifiant du User qui a créé la société Source : User	
<code>siren</code>	SIREN	INT		x	x		Max = 999999999	x
<code>morale</code>	Personne morale	Booléen		x			Valeur défaut = True	x
<code>nameM</code>	Dénomination de l'entreprise	String	255	x	x		Si personne morale	x
<code>lastnameP</code>	Nom	String	100		x		Si personne physique	x
<code>firtnameP</code>	Prénom	String	100		x		Si personne physique	x
<code>sexeP</code>	Sexe	String	1		x		Si personne physique	x
<code>dateCreate</code>	Date de création	Date			x		<code>"date_creation": "2002-05-16",</code> <code>"date_creation_formate": "16/05/2002",</code> <code>"entreprise_cessee": true,</code> <code>"date cessation": "2002-05-16",</code>	x
<code>codeApeNaf</code>	Code APE / NAF	String	5	x	x		4 chiffres suivis de 1 lettre	x
<code>activities</code>	Domaines d'activité	Texte					Liste, séparateur ; => table à part ? <code>"domaine_activite": "Activités des sièges sociaux ; conseil de gestion ;</code>	x
<code>monthCloseAccount</code>	Mois de clôture des comptes		Int				Valeur entre 1 et 12	
<code>disabled</code>	Désactivé	Booléen					Valeur par défaut False	
<code>dateUpdate</code>	Date création ou de dernière M&J	DateTime						
<code>dateDelete</code>	Date désactivation	DateTime			x			
<code>userDelete</code>	User désactivation	INT			x	x	Source : UserDelete - Ok	

Les conventions collectives :

Nom	Libellé	Format	Taille	Unique	Null autor.	FK	Complément
<code>id</code>	Identifiant	AI					Création automatique dans Symfony
<code>idcc</code>	Num identifiant CC	String	4	x	x		Entre 0001 et 9999
<code>titleCc</code>	Titre	String	450	x	x		
<code>disabled</code>	Désactivé	Booléen					Valeur par défaut False
<code>dateUpdate</code>	Date création ou de dernière M&J	DateTime					
<code>dateDelete</code>	Date désactivation	DateTime			x		
<code>userDelete</code>	User désactivation	INT			x	x	Source : UserDelete - Ok

Table de jonction Company - Conventions

Nom	Libellé	Format	Taille	Unique	Null autor.	FK	Complément	Json
<code>id</code>	Identifiant	AI						
<code>company</code>	Id société	INT			x		Source : Company	x
<code>convColl</code>	Id convention collective	INT			x		Source : ConvColl	x
<code>pourcent</code>	Pourcentage	FLOAT(1)					Entre 0.1 et 100	x
<code>disabled</code>	Désactivé	Booléen					Valeur par défaut False	
<code>dateUpdate</code>	Date création ou de dernière M&J	DateTime						
<code>dateDelete</code>	Date désactivation	DateTime			x			
<code>userDelete</code>	User désactivation	INT			x	x	Source : UserDelete - Ok	

User :

Nom	Libellé	Format	Taille	Unique	Null autor.	FK	Complément
<code>id</code>	Identifiant	AI					
<code>uuid</code>	UUID	String	180				
<code>roles</code>	Rôle de niveau 1	Json					Rôle de niveau 1 (security.yaml)
<code>username</code>	Nom Utilisateur	String	100	x			
<code>lastName</code>	Nom	String	50				
<code>firstName</code>	Prénom	String	50				
<code>email</code>	Adresse mail	String	255				
<code>password</code>	Mot de passe	String	255				
<code>disabled</code>	Désactivé	Booléen					Valeur par défaut false
<code>dateUpdate</code>	Date création ou de dernière M&J	DateTime					
<code>dateDelete</code>	Date désactivation	DateTime			x		
<code>userDelete !!</code>	User désactivation	Num			x	x	Source : UserDelete

Table de jonction User - Company

Nom	Libellé	Format	Taille	Unique	Null autor.	FK	Complément	Json
<code>id</code>	Identifiant	AI						
<code>company</code>	Id société	INT			x		Source : company	x
<code>user</code>	Id User	INT			x		Source : User	x
<code>rolesN2</code>	Rôles CLIENT	json					Rôles de niveau 2 (security.yaml)	x
<code>disabled</code>	Désactivé	Booléen					Valeur par défaut false	
<code>dateUpdate</code>	Date création ou de dernière M&J	DateTime						
<code>dateDelete</code>	Date désactivation	DateTime		x				
<code>userDelete</code>	User désactivation	Num		x	x		Source : UserDelete - Ok	

Employee :

Nom	Libellé	Format	Taille	Unique	Null autor.	FK	Complément
<code>id</code>	Identifiant	AI					
<code>noSecSoc</code>	N° sécurité sociale	String	15	x			
<code>sexe</code>	Sexe	Num			x		Source : Sexe
<code>lastName</code>	Nom	String	100				
<code>firstName</code>	Prenom(s)	String	100				
<code>dateBirth</code>	Né(e) le	Date					
<code>nationality</code>	Nationalité	Num			x		Source : Nationality Si plusieurs nationalités, seule 1 est stockée
<code>address1</code>	Adresse	String	250				
<code>address2</code>	Complément 1	String	250	x			
<code>address3</code>	Complément 2	String	250	x			
<code>address4</code>	Complément 3	String	250	x			
<code>countryId</code>	Id pays	Num			x		Source : Country
<code>cp</code>	CP	String	10				
<code>town</code>	Ville	String	80				
<code>disabled</code>	Désactivé	Booléen					Valeur par défaut False
<code>dateUpdate</code>	Date création ou de dernière M&J	DateTime					
<code>dateDelete</code>	Date désactivation	DateTime		x			
<code>userDelete</code>	User désactivation	INT		x	x		Source : UserDelete - Ok

E. Sauvegardes Git :

Graph	Description	Date	
○	○ Ajout d'un salarié	21 Mar 2025 09:27	Nadine
	Modif de Employee et Company.jsx	15 Mar 2025 10:03	Nadine
	Affichage des données de ContractJobsheet dans Employee et Company.jsx	11 Mar 2025 12:57	Nadine
	Maj Employee.jsx : Correction de l'ancienneté	10 Mar 2025 15:53	Nadine
	Maj Employee.jsx : affichage + Calcul ancienneté	10 Mar 2025 12:59	Nadine
	Maj Company.jsx : compteur et liste Jobsheet	7 Mar 2025 13:45	Nadine
	Ajout entités : Jobsheet et ContractJobSheet + Company liste salariés : affichage des contrats	6 Mar 2025 14:38	Nadine
	Fixtures Maj + Contract + Partner	5 Mar 2025 16:28	Nadine
	Correction des tests unitaires	4 Mar 2025 14:18	Nadine
	Company + Profile + Employee et Navigation entre les pages	4 Mar 2025 11:56	Nadine
	Company.jsx et Profile.jsx	28 Feb 2025 11:57	Nadine
	Revue de code des Fixtures - tests Controller Company Ok	19 Feb 2025 11:51	Nadine
	Toutes les Fixtures Ok (merci Nicolas) - Reste à faire ContractFixtures	18 Feb 2025 15:37	Nadine
	Fixtures Ok : Tables Sources - Company - User - Ko : Employee	17 Feb 2025 17:00	Nadine
	Maj repository et controller User et Company	10 Feb 2025 15:51	Nadine
	Test fonctionnels - Company	7 Feb 2025 13:07	Nadine
	Creation BdD V1 + TU Entités	31 Jan 2025 09:33	Nadine
	○ ○ modifs	27 Dec 2024 11:25	Nicolas
	started company entity	15 Nov 2024 12:18	Nicolas

F. MCD et Schéma Workbench plus grands :

MCD :

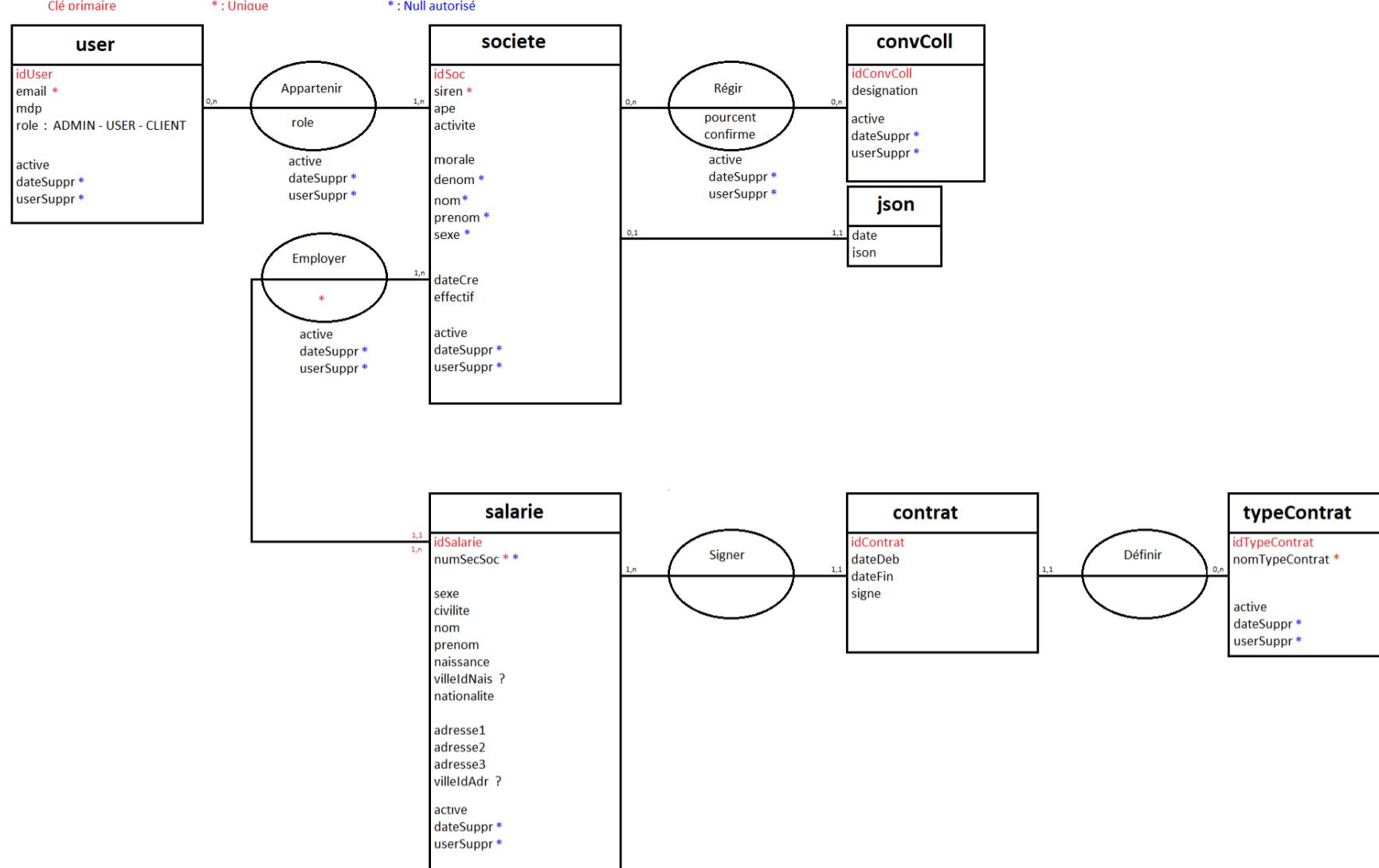


Schéma :

