ADMIN

DOSSIER DE CONCEPTION

Tableau de gestion des révisions

Classification	☐ Public ☒ Diffusion interne ☐ Confidentiel ☐ Secret	
Référence		
Version	1	
	Nom & Prénom	Service
Propriétaire	D.MULDER	
Rédigé par	D.MULDER	
Validé par		
HISTORIQUES DES MISES A JOUR		
Date	Modifié par	Description du changement
25/02/2025	D.MULDER	Création du document

Sommaire

Introduction

○ Présentation générale de l'application

L'application ADMIN est gérée essentiellement au sein du service support de Fortil Infogérance. Elle centralise et permet de piloter tous les utilisateurs de l'entreprise et des clients partenaires.

○ Objectifs & public cible

Ses principaux objectifs :
- Centralise les informations des utilisateurs.
- Permet d'organiser les fiches par entreprise ou autres critères choisis/filtrés.
- Facilite la mise à jour et le maintien d'informations à jour en temps réel.
- Réduit le temps de recherche des informations de contact.
- Outil plus fluide et plus intuitif = gain de temps.
- Synchronisation simplifiée avec les autres outils connectés à l'application : Oracle, ADP et AD (Active Directory).

Public cible :
- Service support de Fortil infogérance
- Potentiellement certains collaborateurs de Fortil Infogérance

Analyse des besoins

○ Recherche utilisateurs

Les utilisateurs principaux sont les membres de l'équipe support :
⇒ 10 collaborateurs (techniciens hotliner et proximité)
⇒ 2 managers (techniciens système d'information)

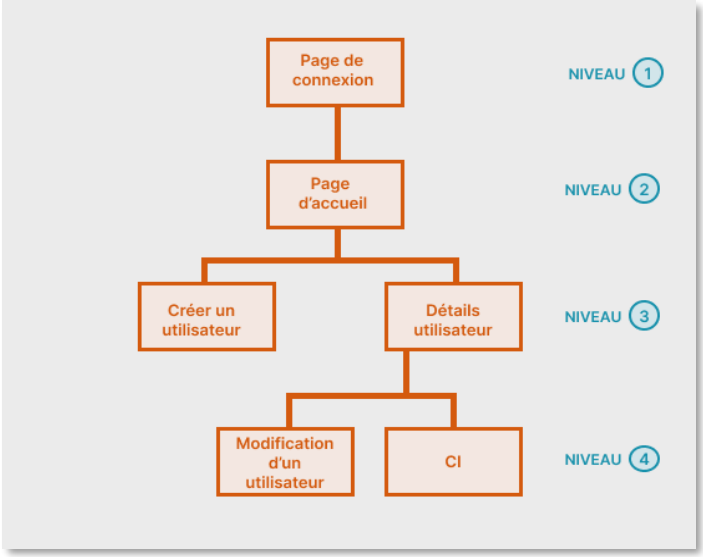
○ Personas et scénario d'utilisation

Personas :
- Techniciens supérieurs : 2 femmes / 10 hommes
- Agés de 23 à 47 ans
- Niveau d'études : Bac à Bac+3
- Ancienneté : de 0 à 13 ans

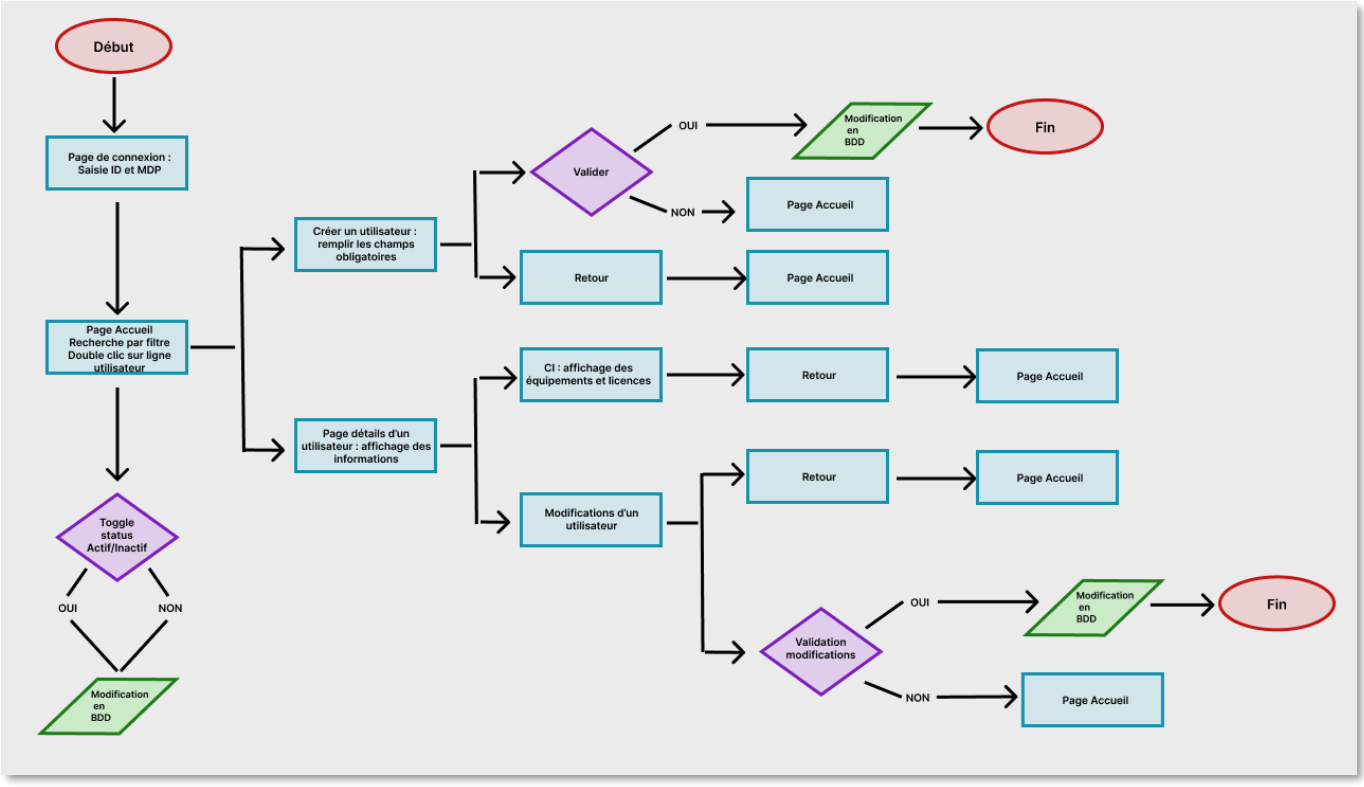
Scénario d'utilisation :
Demande pour une arrivée d'un nouveau collaborateur :
- Création de compte dans l'application ADMIN
- Vérification de la duplication dans l'AD
- Envoi d'un mail au nouveau collaborateur ainsi qu'à son manager

Architecture de l'application

○ Arborescence fonctionnelle



○ Flux de navigation

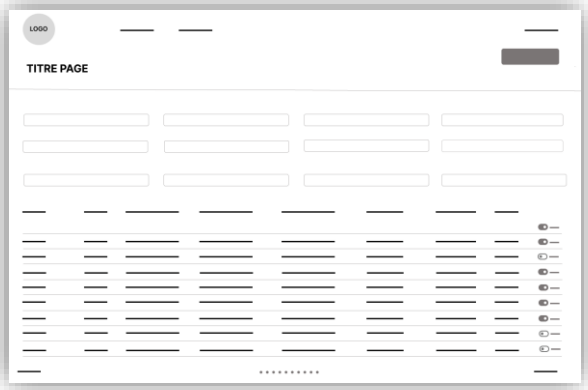


Conception de l'expérience utilisateur (UX)

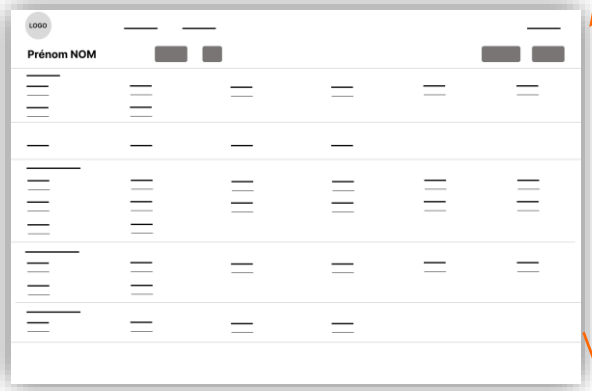
○ Wireframe

Login

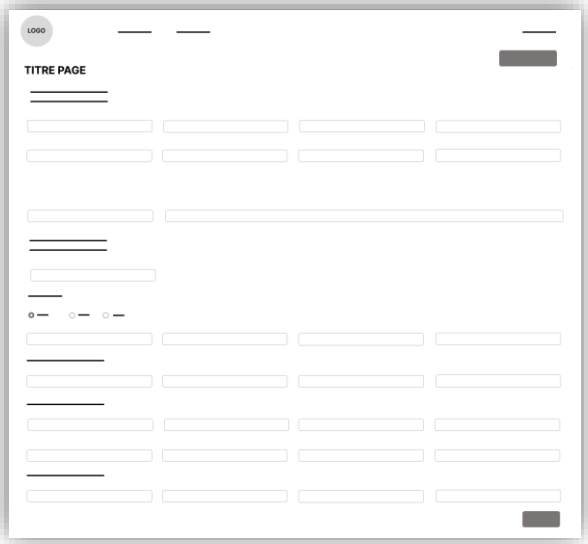
Index



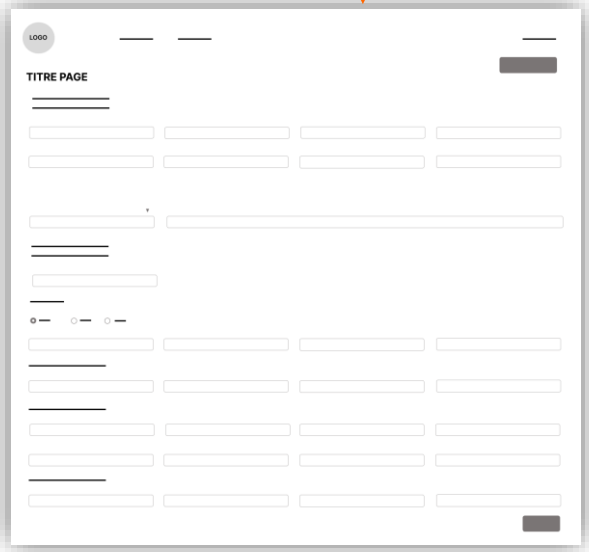
ActorDetail



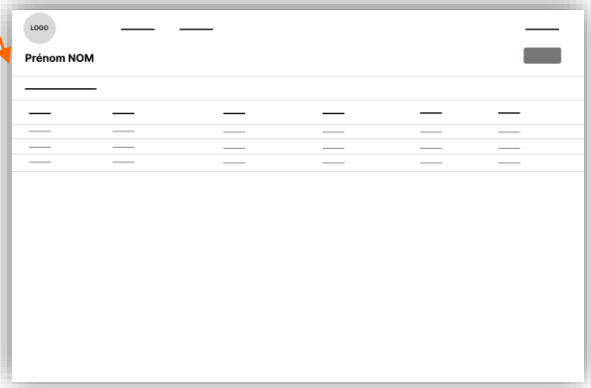
ActorCreateForm



ActorEditForm



ActorCI



○ Parcours utilisateur

•Le rôle de l'application : "ADMIN est utilisée par le service support pour gérer les utilisateurs de chez FORTIL (création, modification, consultation des données)."
•L'objectif principal du parcours : "Optimiser les interactions des agents du support avec l'application pour accomplir leurs tâches rapidement et efficacement."

Voici 2 personas représentant les utilisateurs principaux de l'application :

- **Victoria, agent support débutante**
  - •Objectif : Comprendre rapidement comment utiliser l'application.
  - •Tâche clé : Créer un nouvel utilisateur.
  - •Frustration : Complexité des formulaires ou navigation peu intuitive.
- **Sylvain, responsable support expérimenté**
  - •Objectif : Superviser les modifications effectuées par son équipe.
  - •Tâche clé : Consulter et valider les données utilisateur.
  - •Frustration : Temps perdu à rechercher des informations spécifiques.

Décomposition du parcours utilisateur pour la tâche suivante :
=> <u>Créer un utilisateur (par Victoria)</u>

1. •Action : L'agent se connecte à l'application via un identifiant/mot de passe.
   •Frustration potentielle : Temps de chargement élevé ou échec de connexion.
2. •Action : L'agent navigue sur la page « Search page » et clique sur le bouton "Créer un utilisateur".
   •Frustration potentielle : Navigation peu intuitive.
3. •Action : L'agent saisit les informations nécessaires (Catégorie, Société; nom, tel etc.).
   •Frustration potentielle : Nombreux champs obligatoires ou erreurs non claires.
4. •Action : L'agent valide la création et reçoit une confirmation.
   •Frustration potentielle : Absence de message clair confirmant la réussite.
5. •Action : L'agent revient au tableau de bord « Search page » pour vérifier que l'utilisateur a bien été ajouté.
   •Frustration potentielle : Données non actualisées immédiatement.

Représentation visuelle :

Exemple d'une carte du parcours utilisateur (User journey Map) :

Etape	Action	Emotion/Pensée	Point de friction	Solution proposée
Connexion	Saisie identifiant / mot de passe	"Est-ce que mes identifiants sont corrects ?"	Temps de chargement trop long	Optimiser la vitesse d'accès
Accès à "Créer un utilisateur"	Clic bouton « Créer un utilisateur »	"Où est cette fonctionnalité ?"	Pas assez visible	Déplacer le bouton ou changer la couleur
Remplissage du formulaire	Saisie des données	"Est-ce que j'ai tout rempli ?"	Message erreur trop petit	Ajouter des messages d'erreur plus gros
Validation	Confirmation	"C'est bon, c'est enregistré !"	Pop de confirmation en haut de page	Afficher une confirmation visuelle plus impactante
Retour au tableau	Vérification	"Est-ce que l'utilisateur est là ?"	Données non actualisées immédiatement	Rafraîchissement automatique

Design Visuel (UI)

- Charte graphique (couleurs, typographie, icônes…)

Couleurs

#fff7ed	#ffffff
#ffedd5	#dc2626
#fed7aa	#0f766e
#fb923c	#16a34a
#f97316	#000000
#ff6100	
#c2410c	
#9a3412	
#7c2d12	
#431407	

Icônes

BOUTON RADIO :

TOGGLE :

Fonts

TITRES : sans-serif, bold, 30px

SOUS-TITRES SECTIONS : sans-serif, semibold, 16px

TEXTE : sans-serif, medium, 14px

BOUTONS : sans-serif, medium, 14px

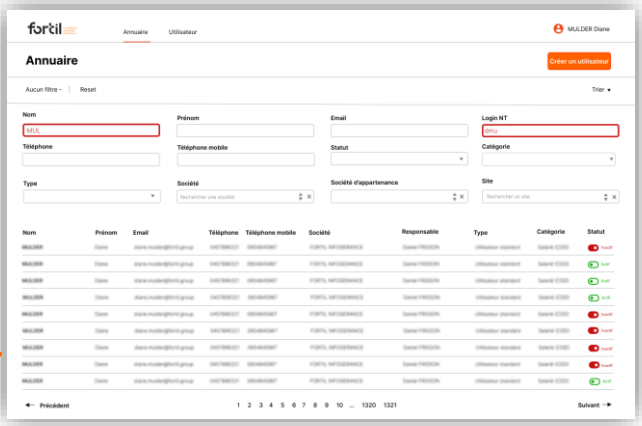
Image & logo

fortil Committed to Sustainable Performance

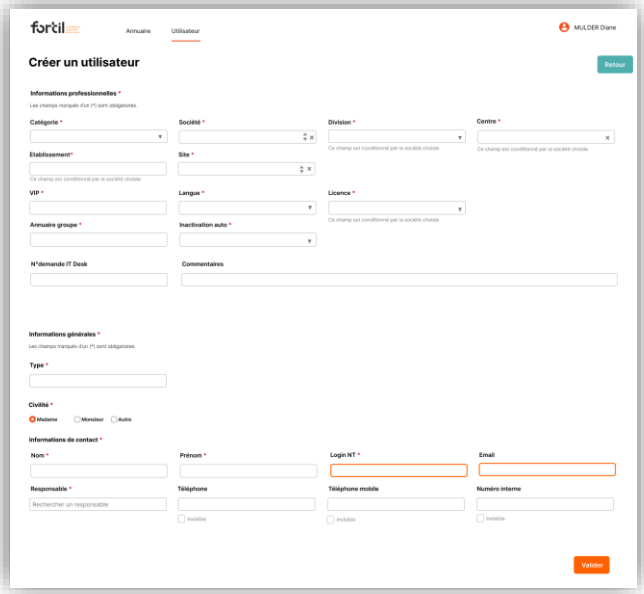
○ Maquette haute fidélité

Login

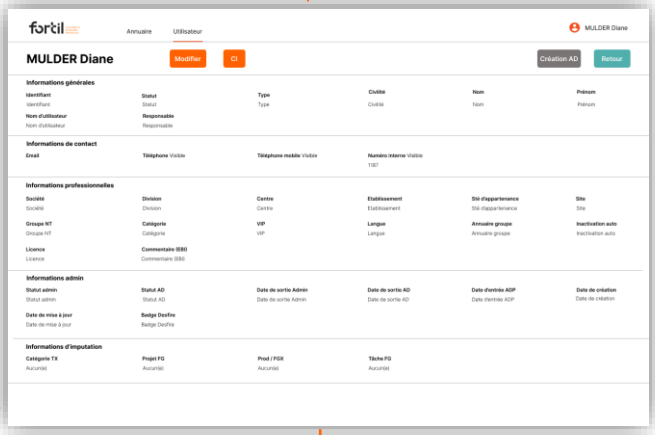
Index



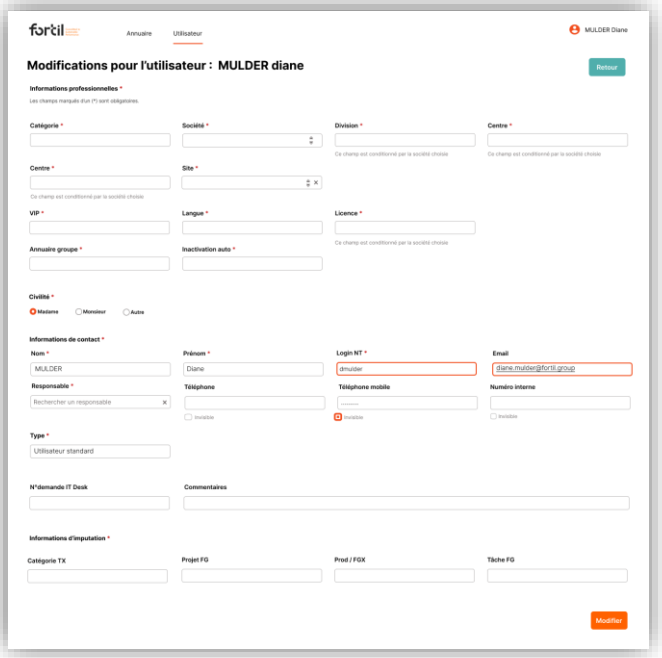
ActorCreateForm



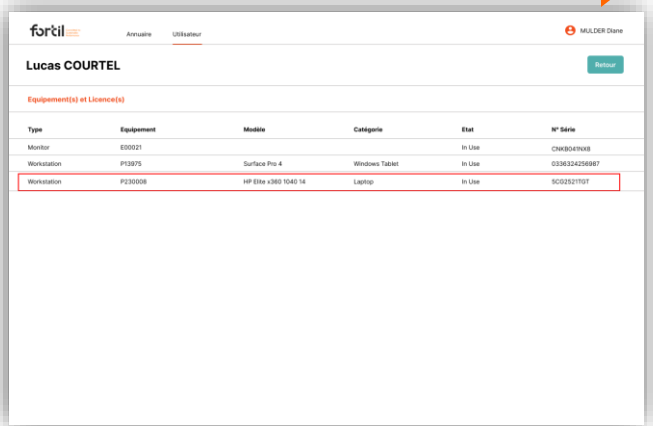
ActorDetail



ActorEditForm



ActorCI



Spécifications fonctionnelles

Description détaillée des fonctionnalités

1. Liste des actions possibles

1.1 Connexion à l'application

Login

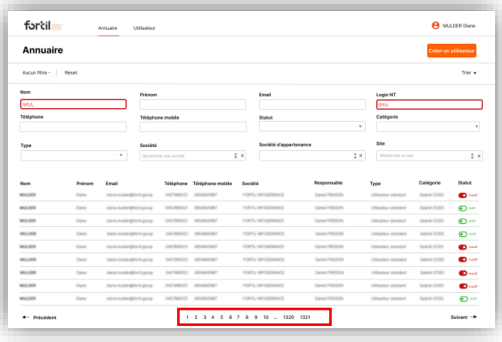
- **Page associée** : Login
- **Description** : Permet aux utilisateurs autorisés (membres du service support et de Fortil infogérance) de se connecter à l'application via ce formulaire.
- **Détails fonctionnels** :

- Authentification par identifiant (Login NT) et mot de passe complexe.
- Redirection vers la page d'accueil après connexion réussie.
- Message d'erreur en cas de tentative de connexion avec
- des identifiants incorrects.

- **Exigences** : Sécurisation de la connexion => réseau Fortil ou FortiToken.

1.2 Rechercher un utilisateur

Index



- **Page associée** : Index
- **Description** : Permet à l'usager de trouver rapidement un utilisateur pour soit consulter sa fiche, modifier ses données ou le rendre inactif / actif.
- **Détails fonctionnels** :

- Recherche par filtre = 12 filtres au choix ou cumulables (Nom / prénom / Email / Login NT / Tel / tel mobile / statut / catégorie / type / Société / Société d'appartenance / site).
- Recherche page par page sur la liste des utilisateurs affichée par ordre alphabétique.

- **Exigences** : fluidité de recherche et rapidité d'affichage des résultats cherchés.

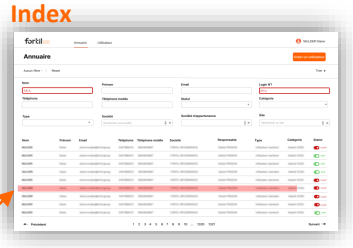
1.3 Consulter les données d'un utilisateur

- **Pages associées** : Index & ActorDetail
- ***Description*** *: Permet au support de visualiser toutes les informations liées à un utilisateur spécifique. Ces données seront affichées sous forme de champs en 5 sections : informations générales / informations professionnelles / informations de contact / AD information / informations d'imputation ORACLE.*
- ***Détails fonctionnels*** *:*

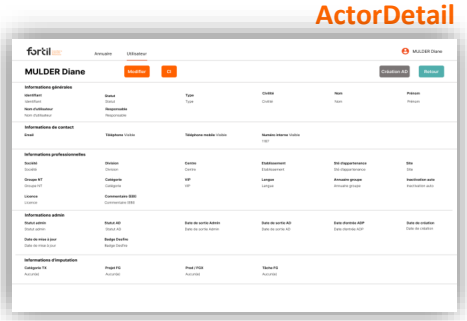
- Sur la page Index, se positionner sur la ligne de l'utilisateur concerné et double cliquer sur cette ligne.
- La page ActorDetail s'affichera avec toutes les informations de l'utilisateur concerné.

- ***Exigences*** *: Recherche rapide pour accéder à la fiche et mise* en page claire et intuitive pour faciliter la consultation

Index



ActorDetail



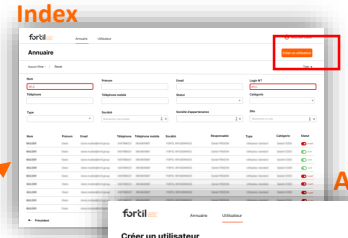
1.4 Créer un utilisateur

- **Pages associées** : Index & ActorCreateForm
- **Description** : Permet d'ajouter un nouvel utilisateur dans la base de donnée ADMIN.
- **Détails fonctionnels** :

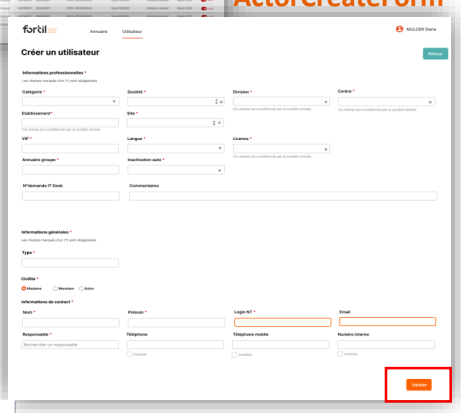
- Sur la page Index : cliquer sur le bouton 'Créer un utilisateur'.
- Formulaire de création avec les champs obligatoires : Catégorie / Société / Division / Centre / Etablissement / Site / VIP / Langue / Licence / Visibilité annuaire / Inactivation auto / Numéro de demande IT Desk / Commentaire / Type / civilité / Nom / Prénom / Responsable / Email.
- Génération d'un identifiant et adresse mail unique par utilisateur dans chaque entreprise.
- Validation des données en cliquant sur le bouton 'Valider'.
- Affichage fenêtre pour confirmation de validation.

- **Exigences** : Vérification que l'adresse mail et Login NT ne soient pas déjà utilisés dans le système et notification (Popup) confirmant la création réussie.

Index



ActorCreateForm

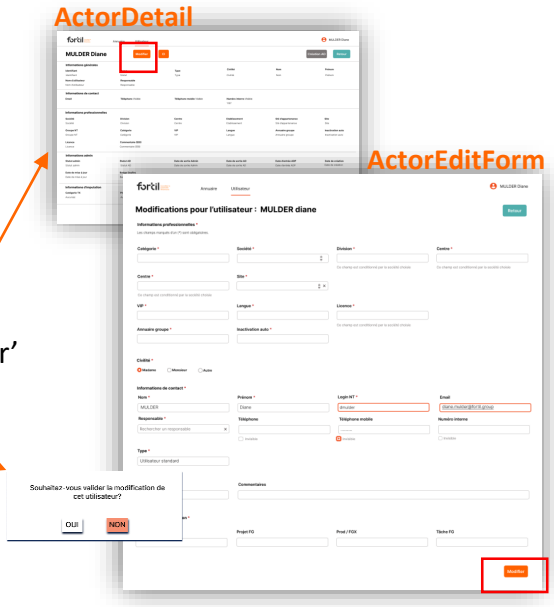


1.5 Modifier un utilisateur

- **Pages associées** : ActorDetail & ActorEditForm
- **Description** : Permet de mettre à jour les informations liées à un utilisateur existant.
- **Détails fonctionnels** :

- Accès à la fiche utilisateur via l'action double clic sur la ligne de celui-ci sur la page Index (cf. 1.3).
- Sur la page ActorDetail, cliquer sur le bouton 'Modifier'.
- Modification possible des champs autorisés.
- Validation des modifications en cliquant sur le bouton 'Valider'
- Affichage fenêtre pour confirmation de validation.

- **Exigences** : Les modifications doivent être validées avant d'être enregistrées dans le système.

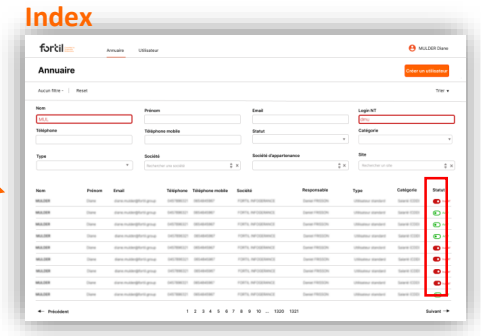


1.6 Desactiver & Activer un utilisateur

- **Page associée** : Index
- **Description** : Permet de désactiver ou activer le profil de l'utilisateur concerné.
- **Détails fonctionnels** :

- Activation : cliquer sur le bouton Toggle pour qu'il apparaisse en vert => utilisateur activé.
- Désactivation : cliquer sur le bouton Toggle pour qu'il apparaisse en rouge => utilisateur désactivé.
- Affichage d'une Popup pour chaque action afin de valider chaque opération.

- **Exigence** : action intuitive et gain de temps.

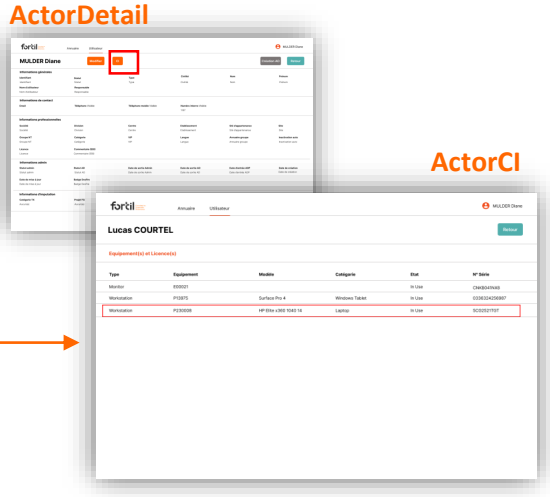


1.7 Consulter les CI d'un utilisateur

- **Pages associées** : ActorDetail & ActorCI
- **Description** : Permet de consulter les équipements et les licences associés à un utilisateur.
- **Détails fonctionnels** :

- Accès à la page des CI en cliquant sur le bouton 'CI' De la page ActorDetail.
- Consultation des données (10 lignes par page).
- Possibilité de changer de page.

- **Exigences** : Présentation claire et espacée pour une meilleure lecture des données.



2. Interactions avec d'autres fonctionnalités

• **Interaction entre gestion des utilisateurs et gestion du matériel :**
- Lorsqu'un utilisateur est créé et que du matériel lui est attribué, l'implémentation se fera directement dans sa fiche sur la page CI.
- Lorsqu'un utilisateur est désactivé, il devient impossible de modifier sa fiche et de lui attribuer du matériel.

- **Interaction entre gestion des utilisateurs et gestion des licences :**

Lorsqu'un nouvel utilisateur est créé, le support peut lui attribuer immédiatement une licence logicielle via sa fiche utilisateur (ActorCreateForm).

- **Interaction entre gestion des CI et rapports (fichiers TXT) :**

Les données sur les CI attribués peuvent être utilisées pour générer des rapports sur l'état du matériel et l'utilisation des licences logicielles.

- **Interaction avec Active Directory :**

Toute modification effectuée sur un profil utilisateur dans ADMIN doit être synchronisée automatiquement avec l'Active Directory pour garantir une cohérence entre les systèmes

3. Exigences non fonctionnelles

- **Performance :**

L'application doit être capable de gérer plusieurs centaines d'utilisateurs simultanément sans ralentissement significatif.

- **Sécurité :**

Toutes les données sensibles doivent être protégées par un chiffrement fort. L'accès aux fonctionnalités critiques doit être limité par rôle.

- **Compatibilité :**

L'application doit être accessible depuis les principaux navigateurs web modernes (Chrome, Edge) et compatible avec l'Active Directory pour la synchronisation.

Interactions & animations

1. Interactions

1.1 Navigations entre les pages

• **Description** : Les utilisateurs naviguent entre les différentes pages (Index, ActorDetail, ActorCreateForm, ActorEditForm, ActorCI) les boutons d'action.
• **Détails**:
- Boutons d'action visibles sur chaque page pour effectuer des tâches spécifiques (exemple : "Créer un utilisateur", "Modifier").
- Ligne cliquable pour accéder directement aux détails d'un utilisateur depuis une liste.
• **Interaction utilisateur** :
- Lorsque l'utilisateur clique sur un lien ou un bouton, la transition vers la nouvelle page est fluide et rapide.
- Les liens non disponibles sont grisés (bouton 'modifier' quand l'utilisateur est inactif.

1.2 Formulaires

• **Description** : Les formulaires sont utilisés pour créer ou modifier des utilisateurs.
• **Details :**
- Champs interactifs avec validation en temps réel (exemple : vérification du format email ou champs obligatoires).
- Boutons « Valider » et « Retour » clairement visibles.
- Indicateurs visuels pour les erreurs (bordure rouge autour des champs invalides ou non remplis quand ils sont obligatoires).
• **Interaction utilisateur :**
- Au clic sur un champ, celui-ci est mis en surbrillance avec une bordure colorée (orange).
- Lorsque le formulaire est soumis avec succès, une notification apparaît sous forme de pop-up (exemple : "Utilisateur créé avec succès").
- Si le formulaire contient des erreurs ou champ vide, une Pop up apparait au champ concerné pour attirer l'attention.

1.3 Recherche & filtres

• **Description** : Les utilisateurs peuvent rechercher ou filtrer les données (utilisateurs, sociétés, divisions…) via des champs dédiés avec ou sans menu déroulant.
• **Details :**
- Champ de recherche dynamique qui affiche les résultats en temps réel au fur et à mesure que l'utilisateur tape.
- Champs de recherche avec menus déroulants.
- **Interaction utilisateur :**
- Lorsqu'un filtre est appliqué, les résultats apparaissent et s'affichent en direct.

1.4 Notifications

• **Description** : Les notifications informent l'utilisateur des actions réussies ou des erreurs rencontrées.

• **Details :**

- Notifications sous forme de pop-ups ou toasts affichées dans le coin supérieur droit de l'écran.
- Messages colorés selon le type d'information :

Vert pour succès (exemple : "Modification enregistrée avec succès"). Impossible de supprimer cet utilisateur

Rouge/orange pour avertissement (exemple : "Toute donnée non enregistrée sera perdue. Souhaitez-vous quitter cette page?").

- **Interaction utilisateur :**
- Les notifications apparaissent avec une animation de glissement depuis le haut ou le côté.
- Elles disparaissent automatiquement après quelques secondes.

1.5 Actions sur les listes

• **Description** : Les listes d'utilisateurs et CI permettent la consultation rapide des données.

• **Details :**

- Pour l'utilisateur sur la page Index, chaque ligne est cliquable pour accéder aux détails.
- Pour les CI, les lignes ne sont pas cliquables, c'est juste de l'affichage.

• **Interaction utilisateur :**

- Clic sur une action : transition rapide vers la page correspondante.

2. Animations

2.1 Survol & clics

• **Flèche souris changeant de forme au survol :**

- Sur chaque champ, elle devient curseur.
- Sur chaque élément cliquable, elle se met en forme de main.

• **Clic sur un bouton :**

- Animation légère au survol qui rend le bouton plus foncé.

2.2 Validation des formulaires

• **La flèche de la souris change de forme au survol :**

- Sur chaque champ, elle devient curseur.
- Sur chaque élément cliquable, elle se met en forme de main.

• **Clic sur un bouton :**

- Animation légère au survol qui rend le bouton plus foncé pour montrer qu'il est cliquable.

2.3 Notifications

- Apparition des notifications (Pop up) => Effet et affichage rapide en haut, à droite de l'écran.
- Disparition progressive après quelques secondes

3. Objectifs des interactions et animations

Les interactions et animations doivent répondre aux objectifs suivants :

1. Améliorer l'expérience utilisateur en rendant l'application intuitive et agréable à utiliser.
2. Fournir des retours visuels immédiats pour chaque action effectuée par l'utilisateur.
3. Réduire les erreurs grâce à des validations claires et des messages explicites.
4. Maintenir une cohérence visuelle dans toute l'application.

Contraintes techniques

1.Technologies utilisées

1.1 Environnement de Développement

- **IDE :** JetBrains Gateway avec WSL pour un environnement de développement sous Linux.
- **Gestion des conteneurs :** Docker pour isoler les services et simplifier le déploiement.
- **Orchestration des conteneurs :** Laravel Sail, une interface en ligne de commande légère pour interagir avec l'environnement de développement Docker par défaut de Laravel.
- **Coordination des services :** Docker Compose via Laravel Sail pour orchestrer les conteneurs (application Laravel, SQLServeur).
- **Langages de programmation :**
  Backend : PHP avec le framework Laravel.
  Frontend : React et TypeScript.
- **Styling :** Tailwind CSS pour un design moderne et réactif.
- **Base de données :** Container dans Docker SQLServeur.
- **Serveur web :** Traefik & VITE
- **Format d'échange de données :** JSON pour les API RESTful.
- **Gestion de version :** GitLab pour suivre les modifications du code.

1.2 Fonctionnalités de Laravel Sail

- Démarrage rapide de l'environnement de développement avec la commande 'sail up'.
- Permet de faire des commandes sans rentrer directement dans le container Docker : Composer, Artisan et npm/yarn via Sail.
- Possibilité d'ajouter des services supplémentaires avec la commande 'sail:add'.

1.3 Workflow de développement avec Sail

- Utilisation de 'sail' pour exécuter les commandes (requêtes php) dans le conteneur.
- Gestion des dépendances via 'sail composer' pour Composer et 'sail npm' pour npm.
- Accès au shell du conteneur avec sail shell pour le débogage et la maintenance.

Cette configuration avec Laravel Sail (en local) permet d'interagir plus facilement avec Docker.

1.4 Architecture technique

- **Frontend :**
  Créé avec React et TypeScript pour une interface utilisateur dynamique et robuste.
  Intégration de Tailwind CSS pour des composants visuels réactifs et personnalisables.
- **Backend :**
  Laravel comme framework principal pour gérer les routes, la logique métier et les interactions avec la base de données.
  Utilisation d'Eloquent ORM pour simplifier les requêtes SQL.
- **Base de données :**
  SQLServeur conteneurisé dans Docker avec des volumes pour éviter la perte de données entre redémarrages. Les volumes permettent de stocker les données de manière persistante, même si le conteneur est arrêté, supprimé ou recréé.
- **Serveur web :**
  Traefik configuré comme reverse proxy pour servir l'application Laravel.

2. Considérations de Sécurité

2.1 Protections des données

- **Chiffrement des données sensibles :**

- Transmission sécurisée via HTTPS grâce à un certificat SSL/TLS sur Nginx.

- **Fichiers .env sécurisés :**

Les variables sensibles (comme DB_PASSWORD ou APP_KEY) sont stockées dans un fichier .env non versionné dans Git.

2.2 Authentification et Autorisation

- Mise en place d'une authentification sécurisée avec le LDAP (Lightweight Directory Access Protocol).
- Gestion des rôles et permissions.

2.3 Sécurité applicative

- Validation des entrées utilisateur côté backend avec Laravel Request Validation :
  - Utilisation de la méthode validate() sur l'objet $request.
  - Création de classes de Form Request pour des validations plus complexes
- Protection contre les attaques courantes :
  Prévention des injections SQL via Eloquent ORM.
  Protection contre les attaques XSS (Cross-Site Scripting) via l'échappement automatique des données dans Blade ou React.
  Protection CSRF (Cross-Script Request Forgery) activée par défaut dans Laravel.

3. Considérations de Performances

3.1 Optimisation Backend

• Utilisation du cache intégré de Laravel pour accélérer les requêtes fréquentes.
• Pagination sur toutes les listes longues (utilisateurs, CI) pour limiter la charge serveur.

3.2 Optimisation Frontend

• Chargement différé des composants React non critiques.
• Minification du code JavaScript/TypeScript et CSS lors du build final.

3.3 Optimisation Base de Données

• Réduction du nombre de requêtes grâce à Eloquent Relationships (with()).

4. Workflow de développement

4.1 Gestion des branches

•Utilisation de GitLab pour gérer les branches.
•**Branche principale (main)** : Contient le code en production.
•**Branche de développement (develop)** : Utilisée pour intégrer et recetter les nouvelles fonctionnalités avant leur déploiement en production.
•**Branche de fonctionnalités** : Créées à partir de develop pour chaque nouvelle fonctionnalité ou correction de bug.

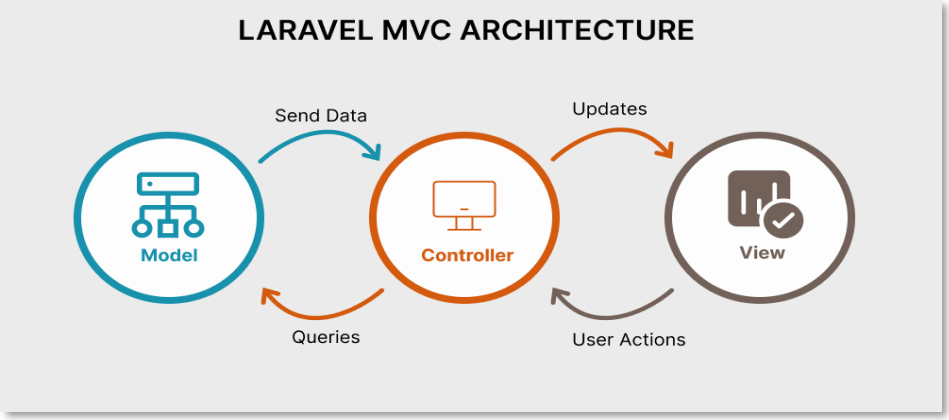
4.2 Optimisation Frontend

1.Création d'une branche de fonctionnalité : À partir de develop, pour travailler sur une nouvelle fonctionnalité.
2.Développement et tests locaux : Utilisation de Docker pour simuler l'environnement de production.
3.Commit et push sur la branche de fonctionnalité : Une fois la fonctionnalité terminée et testée localement.
4.Merge Request : Création d'une demande de fusion vers develop pour revoir le code et valider les tests.
5.Tests et validation : Exécution des tests automatisés et validation manuelle par un autre développeur.
6.Fusion dans develop : Une fois validée, la branche de fonctionnalité est fusionnée dans develop.
7.Déploiement en recette : Pour tester l'intégration de toutes les fonctionnalités dans un environnement proche de la production.
8.Déploiement en production : Une fois les tests en recettes réussis, le code est déployé en production via la branche principale.

Spécifications techniques

•Architecture générale

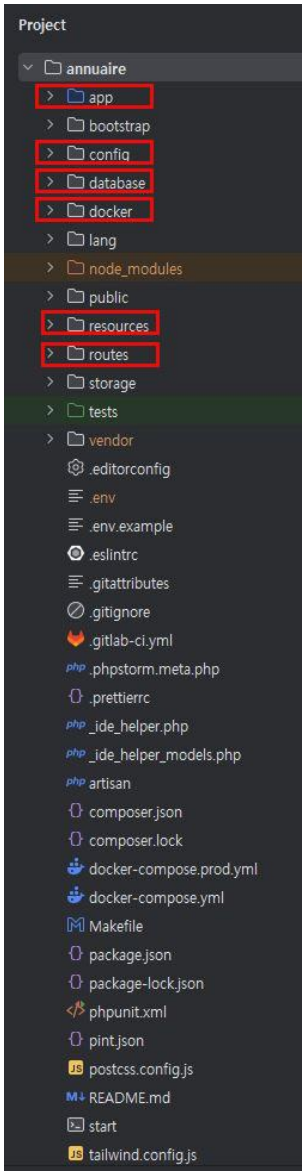
Schéma d'architecture



Arborescence du code

Dossiers principaux du projet :

- app/
- config/
- database/
- docker/
- resources/
- routes/
- storage/



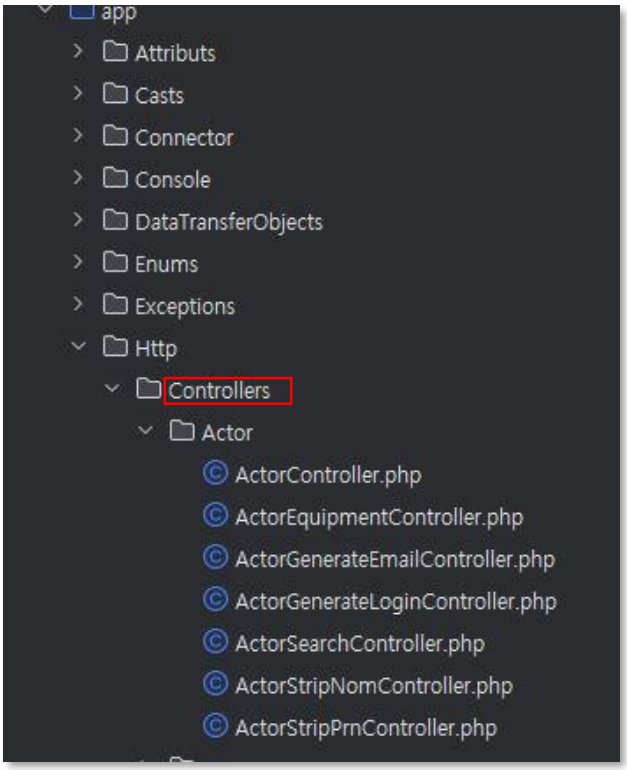
Dossiers clés du projet :

- ### app/Http/Controllers/

Les contrôleurs servent à :
- C'est en quelque sorte l'intermédiaire entre l'utilisateur, le modèle et la vue.
- Organiser le code de l'application en séparant les responsabilités.
- Répondre aux différentes routes de l'application.

Structure type :
- Suit souvent une convention de nommage RESTful pour ses méthodes (index, create, store, show, edit, update, destroy).

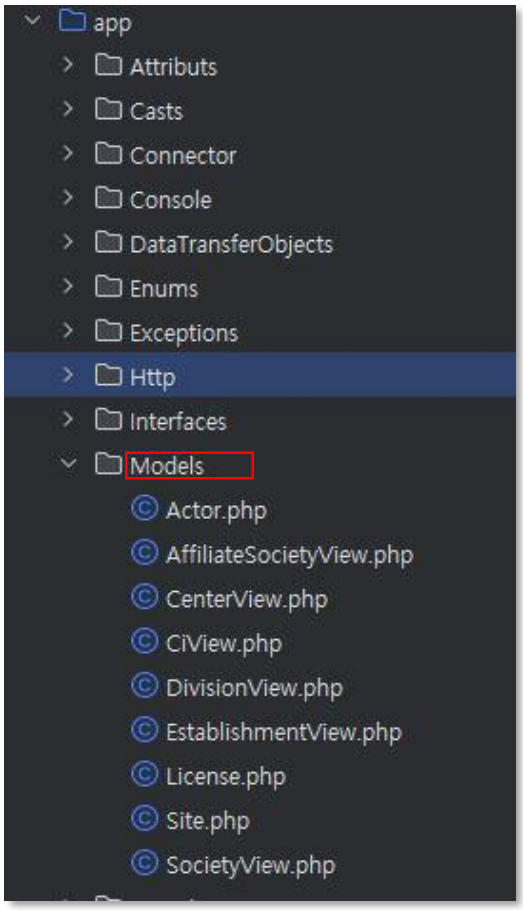
Utilisation dans les routes :
- Les contrôleurs sont utilisés dans le fichier de routes (web.php) pour définir le comportement de l'application en réponse à des requêtes HTTP spécifiques.



- ### app/Models

Les modèles servent à :
- Représenter les tables de la BDD sous forme d'objets manipulables.
- Définir la structure des tables et les relations entre elles.
- Effectuer des opérations CRUD sur les données.

Structure type :
Un Model Laravel est une classe PHP qui :
- Définit des propriétés et des méthodes pour interagir avec la BDD.

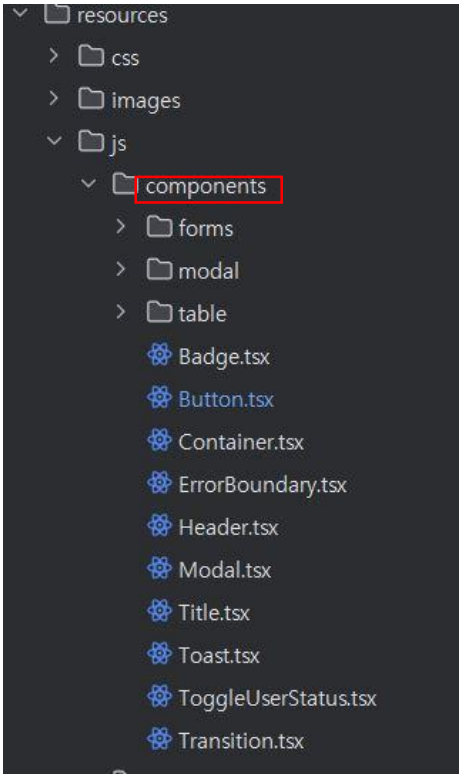


• resources/js/components

Les composants servent à :
- **Réutilisation** => Encapsuler des parties d'interface utilisateur (UI) ou logique de vue qui peuvent être utilisées dans plusieurs endroits de l'application.
- **Modularité** => Simplifier la gestion et la maintenance du code en divisant les vues en petits blocs indépendants.
- **Clarté** => faciliter la lecture et la structure des vues.

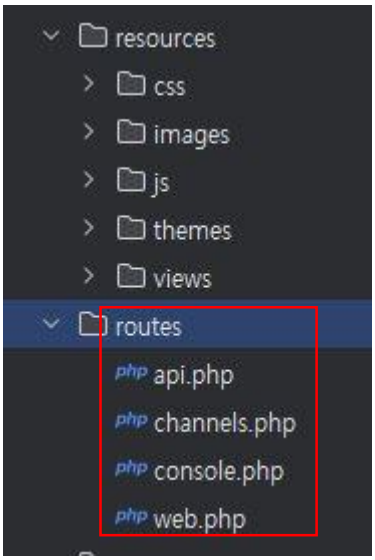
Structure type :
- **React avec fichiers TypeScript.**

Utilisation des composants :
- Dans les vues
- Passage de données
- Personnalisation



• Routes/

Les fichiers sont :
- **web.php** => pour les routes accessibles via le navigateur web.
- **api.php** => pour les routes d'API.
- Les routes servent à :
  - Définir les URL accessibles dans l'application.
  - Associer ces URL à des contrôleurs ou des actions spécifiques.
  - Mise en place des middlewares dans lesquels on regroupe des routes pour réaliser des actions de contrôle communes à toutes ces routes.
  - Nommer les routes pour une référence facile dans le code.



•Mappage de colonnes

Définition explicite de l'association des champs du code associés aux colonnes de la base de données:

Nom des champs	Noms colonnes Base de données
Catégorie	stu_salarie
Société	cod_sct
Division	cod_div_appa
Centre	cod_cte_appa
Etablissement	cod_etb
Société d'appartenance	l_sct_appa
Site	trigramme
VIP	vip
Langue	lng
Licence	lic_o365
Visibilité annuaire	f_annuaire
Inactivation auto	f_inact
Numéro de demande IT Desk	cmt
Commentaire (EBI)	comment_ebi
Type	uo_ad
Civilité	qualite
Nom	nom_act
Prénom	prn_act
Responsable	id_manager
Nom d'utilisateur	login_nt
Email	e_mail_notes
Numéro de téléphone	num_tel
Téléphone mobile	num_port
Numéro interne	num_tel_interne
Statut AD	stu_ad
Date de sortie Admin	date_sortie_admin
Date de sortie AD	date_sortie_ad
Date d'entrée ADP	dte_entree
Date de création	dte_cre
Date prévision AD	dte_prevision_ad
Date de mise à jour	dte_maj
Badge Desfire	desfire
Catégorie TX	categorie_tx
Projet FG	projet_fg
Prod / FGX	Productif
Tâche FG	tache_fg

ENVIRONNEMENT DE RECETTE & TESTS CLIENT

- Plan de tests client
- Processus d'itération et d'amélioration

Prévu en S16

DEPLOIEMENT & MAINTENANCE

- Stratégie de lancement
- Plan de mise à jour et d'évolution

Prévu en S18