



# Taskies

Application web REACT



AOUT 2023

# Sommaire

---

1. Contexte “Laboratoire GSB”
2. Expression du ou des besoins
3. Les objectifs
4. Analyse fonctionnelle

# Contexte “Laboratoire GSB”

---

## Description du laboratoire GSB

### Le secteur d'activité

L'industrie pharmaceutique est un secteur très lucratif dans lequel le mouvement de fusion acquisition est très fort. Les regroupements de laboratoires ces dernières années ont donné naissance à des entités gigantesques au sein desquelles le travail est longtemps resté organisé selon les anciennes structures.

Des déboires divers récents autour de médicaments ou molécules ayant entraîné des complications médicales ont fait s'élever des voix contre une partie de l'activité des laboratoires : la visite médicale, réputée être le lieu d'arrangements entre l'industrie et les praticiens, et tout du moins un terrain d'influence opaque.

### L'entreprise

Le laboratoire Galaxy Swiss Bourdin (GSB) est issu de la fusion entre le géant américain Galaxy (spécialisé dans le secteur des maladies virales dont le SIDA et les hépatites) et le conglomérat européen Swiss Bourdin (travaillant sur des médicaments plus conventionnels), lui même déjà union de trois petits laboratoires .

En 2009, les deux géants pharmaceutiques ont uni leurs forces pour créer un leader de ce secteur industriel. L'entité Galaxy Swiss Bourdin Europe a établi son siège administratif à Paris. Le siège social de la multinationale est situé à Philadelphie, Pennsylvanie, aux Etats-Unis.

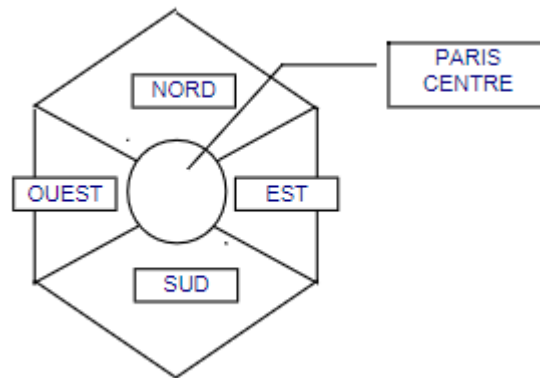
*La France a été choisie comme témoin pour l'amélioration du suivi de l'activité de visite.*

### Réorganisation

Une conséquence de cette fusion, est la recherche d'une optimisation de l'activité du groupe ainsi constitué en réalisant des économies d'échelle dans la production et la distribution des médicaments (en passant par une nécessaire restructuration et vague de licenciement), tout en prenant le meilleur des deux laboratoires sur les produits concurrents.

L'entreprise compte 480 visiteurs médicaux en France métropolitaine (Corse comprise), et 60 dans les départements et territoires d'outre-mer. Les territoires sont répartis en 6 secteurs géographiques (Paris-Centre, Sud, Nord, Ouest, Est, DTOM

Caraïbes-Amériques, DTOM Asie-Afrique). Une vision partielle de cette organisation est présentée ci-dessous.



*Après deux années de réorganisations internes, tant au niveau du personnel que du fonctionnement administratif, l'entreprise GSB souhaite moderniser l'activité de visite médicale.*

## Description du système informatique

### Le système informatique

Sur le site parisien, toutes les fonctions administratives (gestion des ressources humaines, comptabilité, direction, commerciale, etc.) sont présentes. On trouve en outre un service labo- recherche , le service juridique et le service communication.

La salle serveur occupe le 6ème étage du bâtiment et les accès y sont restreints (étage accessible par ascenseur à l'aide d'une clé sécurisée, portes d'accès par escalier munies d'un lecteur de badge, sas d'entrée avec gardien présent 24h/24).

Les serveurs assurent les fonctions de base du réseau (DHCP, DNS, Annuaire et gestion centralisée des environnements) et les fonctions de communication (Intranet, Messagerie, Agenda partagé, etc.). On trouve aussi de nombreuses applications métier (base d'information pharmaceutique, serveurs dédiés à la recherche, base de données des produits du laboratoire, base de données des licences d'exploitation pharmaceutique, etc.) et les fonctions plus génériques de toute entreprise (Progiciel de Gestion Intégré avec ses modules RH, GRC, etc.). Un nombre croissant de serveurs est virtualisé.

Constitué autour de VLAN, le réseau segmente les services de manière à fluidifier le trafic. Les données de l'entreprise sont considérées comme stratégiques et ne peuvent tolérer ni fuite, ni destruction. L'ensemble des informations est répliqué

quotidiennement aux Etats-Unis par un lien dédié. Toutes les fonctions de redondances (RAID, alimentation, lien réseau redondant, Spanning-tree, clustering, etc.) sont mises en œuvre pour assurer une tolérance aux pannes maximale.

## La gestion informatique

La DSI (Direction des Services Informatiques) est une entité importante de la structure européenne qui participe aux choix stratégiques.

Pour Swiss-Bourdin, qui occupait le siège parisien avant la fusion, l'outil informatique et l'utilisation d'outils décisionnels pour améliorer la vision et la planification de l'activité ont toujours fait partie de la politique maison, en particulier pour ce qui concerne la partie recherche, production, communication et juridique. La partie commerciale a été le parent pauvre de cette informatisation, les visiteurs étant vus comme des acteurs distants autonomes. La DSI a convaincu l'entreprise que l'intégration des données fournies par cette partie aura un impact important sur l'ensemble de l'activité.

## L'équipement

L'informatique est fortement répandue sur le site. Chaque employé est équipé d'un poste fixe relié au système central. On dénombre ainsi plus de 350 équipements terminaux et on trouve aussi des stations de travail plus puissantes dans la partie labo-recherche, et une multitude d'ordinateurs portables (personnels de direction, service informatique, services commerciaux, etc).

Les visiteurs médicaux reçoivent une indemnité bisannuelle pour s'équiper en informatique (politique Swiss-Bourdin) ou une dotation en équipement (politique Galaxy). Il n'y a pas à l'heure actuelle d'uniformisation des machines ni du mode de fonctionnement. Chaque employé de l'entreprise a une adresse de messagerie de la forme nomUtilisateur@swiss-galaxy.com. Les anciennes adresses de chaque laboratoire ont été définitivement fermées au 1er janvier 2011.

# Expression du ou des besoins

---

Dans un contexte où l'entreprise souhaite optimiser son activité pour faire face à ses concurrents, le travail d'équipe et la communication au sein de l'entreprise semble primordiale.

De plus, étant donné que les entreprises ont fusionné, l'organisation du travail doit être revue et centralisée pour permettre de faciliter la délégation de tâches et ainsi l'avancement d'un projet.

Il est également pertinent de se pencher sur un outil qui permettrait une meilleure gestion des comptes utilisateurs en fonction des différents pôles.

# Les objectifs

---

## 1. Professionnel

Le but du projet est de créer une application web qui va permettre d'organiser et de gérer des tâches au sein d'une entreprise.

On veut ainsi :

- faciliter la collaboration entre les pôles
- faciliter la collaboration entre les personnes d'un même pôle
- aider à l'organisation du travail
- mieux répartir la charge de travail
- améliorer l'efficacité du travail

## 2. Personnel

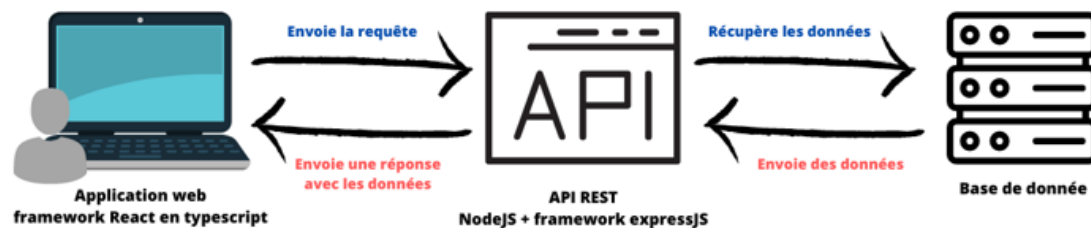
Personnellement, ce projet a aussi pour but de :

- mettre en application les connaissances apprises durant le travail en entreprise dans un autre contexte professionnel
- apprendre à utiliser typescript
- apprendre à utiliser docker
- déploiement d'une application web sur vercel

# Analyse fonctionnelle

---

## Présentation de l'architecture globale de l'AP



Sur ce schéma on a les différents composants de notre architecture.

On a :

- Le client avec l'application web
- L'API REST qui va permettre de faire le lien avec le client et la base de donnée
- La base de données (? utilisation de mongoDB ?)

## Organisation des données

### Organisations des rôles

- **Administrateur (0)**, le rôle le plus élevé, il va créer des utilisateurs, les supprimer, avoir accès à toutes les données

*Le rôle de développeur est particulier car il ressemble à celui de l'administrateur, cependant il n'aura pas toutes les fonctionnalités de ce dernier. Il ne pourra pas avoir accès aux vrais données du site seulement celle de test.*

- **Directeur (1)**, le rôle qui va permettre de voir et d'assigner toutes les tâches aux employés de tous les pôles
- **Le référent (2)**, le rôle qui va permettre de voir et d'assigner toutes les tâches aux employés de son pôle
- **L'employé (3)**, le rôle qui va permettre de voir toutes les tâches du pôle

### Les différents pôles

- Réseau & systèmes
- RH/Compta/Juridique/Secrétariat administratif (Administration)
- Communication & rédaction
- Développement
- Commercial
- Labo-recherche



## Les tâches

Celles-ci seront regroupées par pôles.

Une tâche se compose de :

- un titre
- une personne assignée
- une description
- un/des fichier(s) concerné(s)
- des commentaires

## Organisation du site internet

Maquette générale du site sur Figma :

La vue de l'utilisateur :

<https://www.figma.com/file/yDFWCAnj0cYBtIGHlw229O/AP2---Taskies?type=design&node-id=0%3A1&t=xYBPgZzMm1JtFCVF-1>

La vue de l'admin :

<https://www.figma.com/file/k4oKkQ739b5Py7dupGutpW/AP2---Taskies-admin-view?type=design&t=YWJNDIEgOxKVp9c5-1>

## Organisation du projet

La création d'un profil administrateur est la tâche à réaliser en priorité car c'est celui qui va permettre de créer d'autres utilisateurs.

J'utilise le gestionnaire de paquet 'pnpm' qui est plus efficace que 'yarn' ou 'npm'

## Backend

1. Installations de plusieurs dépendances :

- **cors** : cors est un package qui va permettre de gérer les différentes options du CORS (Cross-Origin Resource Sharing)  
*C'est ce qui nous permet de contrôler quels domaines ont le droit d'envoyer des requêtes à notre API*
- **nodemon** : permet de redémarrer l'API après chaque changement
- **mongoose** : permet de faire la connexion entre une base de données MongoDB et notre API
- **dotenv** : permet d'utiliser les variables d'environnement du fichier .env

- **body-parser** : essentiel à la récupération de données depuis le front, nous permet d'interpréter le corps d'une réponse HTTP
- **bcrypt** : permet de crypter les mots de passe pour plus de sécurité
- **ts-node**: permet d'utiliser nodemon car par défaut nodemon utilise node et ts-node permet de compiler des fichiers javascript
- **sqids** : permet de générer des ids à partir de tableau

```
$ npm install -g ts-node
```

- **@types/express** : permet d'avoir accès au type typescript d'express
- **@types/node** : permet d'avoir accès au type typescript de node
- **@types/cors** : permet d'avoir accès au type typescript de cors
- **@types/bcrypt** : permet d'avoir accès au type typescript de bcrypt

## Schéma de données

### User

```
1 import mongoose from "mongoose";
2 import { Schema } from "mongoose";
3
4 const UserSchema : Schema<any, Model<...>, {...}, {...}, {...}, {...}, DefaultSchemaOptions, = new Schema( definition: {
5   lastname: { type: String, required: true },
6   firstname: { type: String, required: true },
7   email: { type: String, required: true },
8   password: { type: String, required: true },
9   pole: { type: String, required: true },
10  tasks: [{ type: Schema.Types.ObjectId, ref: "Task" }],
11  role: { type: Number, required: true }
12 });
13
14 5+ usages new *
15 export const User : Model = mongoose.model( name: "User", UserSchema)
```

### Task

```
1 import mongoose from 'mongoose';
2 const {Schema} = mongoose;
3
4 const TaskSchema = new Schema({
5   index : {type: Number, required: true},
6   title : {type : String, required : true},
7   status : {type: String, required : true},
8   assignee : {type : String, required : true},
9   desc : {type : String, required : false},
10  comment : {type : String, required : false},
11  pole: {type: String, required: true},
12  files: {type: Array, required: false}
13 });
14
15 5+ usages Kazeynma
16 export const Task : Model<InferSchemaType<...>, ObtainSchemaGeneric<...>, ObtainSchemaGeneric<...>, ObtainSchemaGeneric<...
```

## CodeEmail

```
1 import mongoose from "mongoose";
2 import { Schema } from "mongoose";
3
4 const CodeEmailModel : Schema<any, Model<...>, {...}, {...}, {...}, {...}, DefaultSchemaOptions, = new Schema( definition: {
5   code: {type: String, required: true},
6   active: {type: Boolean, required: true, default: true},
7   user: {type: String, required: true},
8
9 });
10
11 export const CodeEmail : Model = mongoose.model( name: "CodeEmail", CodeEmailModel);
```

## Frontend

### 1. Installations de plusieurs dépendances :

- **react-router-dom** : permet un meilleur contrôle des routes
- **react-hook-form** : permet d'avoir un meilleur contrôle sur les formulaire à l'intérieur de notre application
- **shadcn-ui** : permet de faciliter la création d'interface utilisateur simple et beaux (en coopération avec **Tailwindcss**)
- **Font Awesome Icons** (@fortawesome/fontawesome-svg-core) : librairie svg gratuite parfaite pour les icônes
- **React-beautiful-dnd** : librairie permettant de faire du drag-and-drop (sera utilisé pour la partie des tâches)
- **react-chartsjs-2** : permet de créer des statistiques visuels

## Utilisation de typescript

Apprentissage du langage typescript grâce à la documentation officielle :

<https://www.typescriptlang.org/docs/handbook/typescript-in-5-minutes.html>

### Pourquoi avoir choisi le typescript plutôt qu'à javascript ?

Typescript, comme son nom l'indique, va permettre d'assigner *types* à nos valeurs ce qui permet de gérer plus facilement les erreurs.

C'est un *static type system* qui nous permet de définir les formes et comportements de nos valeurs.

Au lieu de nous retourner *undefined* comme sur javascript, typescript va nous dire où se situe l'erreur. Par exemple, si je veux accéder à une propriété de mon objet `user = {name : "bob", age = 17}` qui n'existe pas comme `user.birthday`, au lieu d'avoir *undefined* on aura une erreur qui nous dit que `user.birthday` n'existe pas.

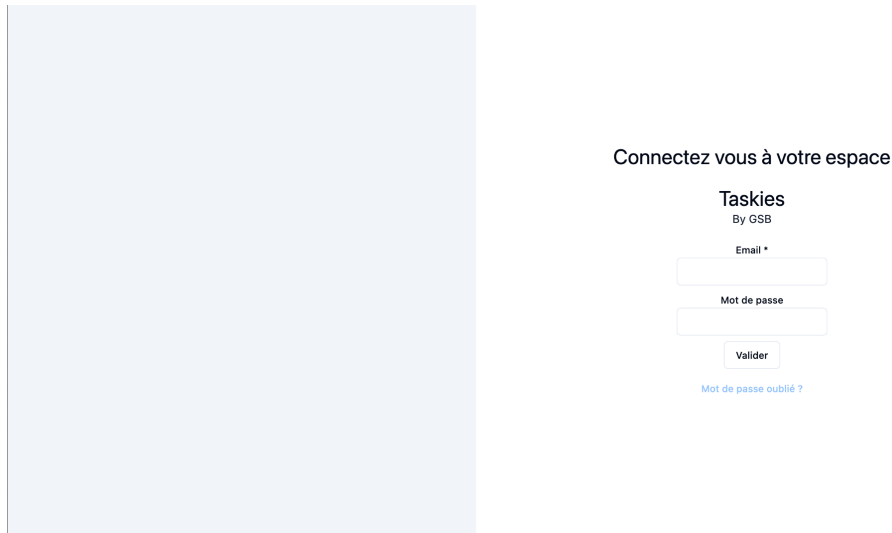
*Il arrive parfois que typescript puisse déterminer le type de la valeur sans qu'on lui donne car le type est évident*

*Exemple : let greetings = "Hello"*

# Fonctionnalités

## Connexion utilisateur

L'utilisateur est une part importante de mon projet. J'ai donc commencé par créer le modèle de l'utilisateur dans l'API



Connectez vous à votre espace

**Taskies**  
By GSB

Email \*

Mot de passe










Valider

[Mot de passe oublié ?](#)

## Création de la page administrateur

1. Création de la page administrateur
  - Permet de gérer les utilisateurs : ajouter, modifier, supprimer

***Page uniquement disponible pour les administrateurs pour gérer les utilisateurs***

ADMIN admin   GSB - Taskies						Administrateur ▾
						 <b>Création d'utilisateur</b>
	Nom	Prénom	Email	Pole	Role	Nombres de tâches assignées
 	ADMIN	admin	ikagnana@gmail.com	Administrateur	Administrateur	0
 	RH	NewSeverineTest	kagnana.lth@ecole-isitech.fr	Administration	Référent	0
 	TEST	TestModif	test.test@tets.fr	Communication & rédaction	Employé	0
 	TEST	ajout	test.test@tets.fr	Commercial	Directeur	0

***Formulaire pour ajouter/modifier un utilisateur***, ce sont des dialog qui s'ouvrent lorsque l'on appuie sur le bouton "Création d'utilisateur" ou sur le crayon dans la colonne de l'utilisateur.

### Création d'un nouvel utilisateur

Nom \*

Prénom \*

Email \*

Mot de passe

Pole

Role

Valider

### Modification de l'utilisateur

Nom \*

Prénom \*

Email \*

Mot de passe

Pole

Role

Valider

## Sécurité au niveau du mot de passe utilisateur

### 1. À la création de l'utilisateur :

Pour assurer la sécurité du mot de passe utilisateur on a créé un bouton qui permet de générer de manière aléatoire un mot de passe. Une fois que l'utilisateur sera créé, on lui enverra un email avec son mot de passe.

### 2. Modification du mot de passe :

#### a. Quand l'utilisateur ne se souvient plus de son mot de passe

Lorsque l'utilisateur ne connaît plus son mot de passe. Il peut cliquer sur "Mot de passe oublié ?" dans la page de connexion. Cette action ouvrira une modale qui lui permettra de renseigner son email.

**Dialog qui permet à l'utilisateur de renseigner son email afin d'avoir un lien pour modifier son mot de passe**

### Demande de réinitialisation du mot de passe

Nous vous enverrons un lien par email pour réinitialiser votre mot de passe.

Envoyer

### Réinitialisation de votre mot de passe

 administrateur GSB <kagnana.ith@ecole-isitech.fr>

 vendredi 29 septembre 2023 11:39:33

 **Delivrabilité**
 Répondre
  Transférer
  Imprimer
  Supprimer
 

Cher utilisateur, suite à la perte de votre mot de passe voici le lien pour réinitialiser votre mot de passe :  
<http://localhost:5173/reset-password/id=L7pnHb31gB>

Lorsqu'il validera, cela enverra une requête pour générer un lien pour modifier son mot de passe. Il recevra le lien par email.

**controllers/Users.tsx**, c'est ici qu'on va générer un code sqids que l'on va sauvegarder en base de données.

```
105 const getCodeEmail = (req: Request, res: Response): void => {
106   User.find({email: req.body.email}) Query
107   .then( onfulfilled: (user: (HydratedDocument) ) : void => {
108     if (user) {
109       //génération d'un id aléatoire
110       const sqids: Sqids = new Sqids( options: {
111         minLength: 10
112       })
113       const id: string = sqids.encode( numbers: [req.body.email.length, Math.floor(Math.random() * 9), Math.floor(Math.random() * 9)])
114
115       const newCode: = new CodeEmail( doc: {code: id, user: req.body.email})
116       newCode.save()
117       transporter.sendMail( mailOptions: {
118         from: "administrateur GSB" <kagnana.ith@ecole-isitech.fr>',
119         to: req.body.email,
120         subject: "Réinitialisation de votre mot de passe",
121         text: "Cher utilisateur, suite à la perte de votre mot de passe voici le lien pour réinitialiser votre mot de passe : " +
122           "http://localhost:5173/reset-password/id=" + id
123       }) Promise<SentMessageInfo>
124       .then(info: SMTPTransport.SentMessageInfo => {
125         res.send("Email bien envoyé.")
126         console.log("Message sendé !")
127       }) Promise<void>
128       .catch((err) => console.log("Message cannot be sendé"))
129     }
130   }) Promise<void>
131   .catch((err): void => {
132     console.log(err)
133     res.status(204).send("Couldn't find the user.")
134   })
135 }
136 }
```



TEST Task  
test.task@tets.fr

b. Quand l'utilisateur est connecté

**Composant de la page “mon compte” avec un bouton qui permet de modifier son mot de passe**

Dans les deux cas l'utilisateur va accéder à la même page.

Je veux modifier mon mot de passe

**Page qui permet à l'utilisateur de modifier son mot de passe**

Réinitialisation du mot de passe

Tasks

Nouveau mot de passe

Validation nouveau mot de passe

Valider

### Code à l'intérieur de la page `ResetPasswordPage.tsx`

```
29  const valideCode = async () : Promise<void> => {
30      if (params.code == undefined) {
31          navigate( to: "/*")
32      } else if (params.code == "token") {
33          console.log("beep")
34          const token : string | null = localStorage.getItem( key: "token")
35          if (token !== undefined) {
36              const user = getUser()
37              setEmail(user.email)
38              setValide( value: true)
39          } else {
40              navigate( to: "/*")
41          }
42      } else{
43          const response = await codeEmailService.getCodeEmailByCode(params.code.slice(3))
44          setValide(response.active)
45          setEmail(response.user)
46          setTimeout( handler: () : void => {
47              console.log("oui")
48              codeEmailService.desactiveCode(response._id)
49          }, timeout: 5000)
50      }
51  }
```

2 comportements différents :

- si le params est égal à "token" : on va récupérer le token de l'utilisateur pour voir s'il est connecté. Si jamais l'utilisateur n'a pas de token, l'utilisateur sera redirigé vers une page d'erreur.
- si le params n'est pas égal à "token" : l'application va envoyer une requête pour récupérer les informations correspondant au code, notamment s'il est actif.
  - Si le code est actif, la page s'affiche
  - Sinon l'utilisateur sera redirigé vers la page de connexion.

**Afin que le lien ne soit pas réutilisable, lorsque l'utilisateur va utiliser le lien, l'application va envoyer une requête pour désactiver le code. L'utilisateur devra alors ne pas recharger la page ou essayer de modifier le mot de passe avec le même lien.**

### Création de la page principale (Tâches)

L'utilisateur :

- A une page d'accueil avec les tâches qui lui sont assigné
- Peut ajouter des tâches :
  - A lui-même
  - A ceux appartenant au même pôle (avec le rôle référent)
  - A tous ceux de l'entreprise (avec le rôle directeur)

## Page qui permet à l'utilisateur de gérer ses ou les tâches.

TEST Task   GSB - Taskies		Développement ▾
<div>Création d'une tâche</div>		
À faire	En cours	Terminées
<div>test déplacement </div> <div>test longue description Le Lorem Ipsum est simplement du faux texte </div>	<div>test important </div> <div>test</div>	<div>test fichiers </div> <div>tester si ajout de fichier possible</div>

## Formulaire pour créer une nouvelle tâche, même formulaire quand il s'agit de modifier une tâche

Modification d'une nouvelle tâche

Assigner à \*  
Task TEST

Pole \*  
Développement

Titre \*  
test fichiers

Description  
tester si ajout de fichier possible

Commentaire

Fichier(s)  
Ajouter un lien

Listes de fichiers  
https://www.sample.org/system#popcorn  
https://sample.info/toothpaste  
http://www.sample.info/sofa#brick

Valider

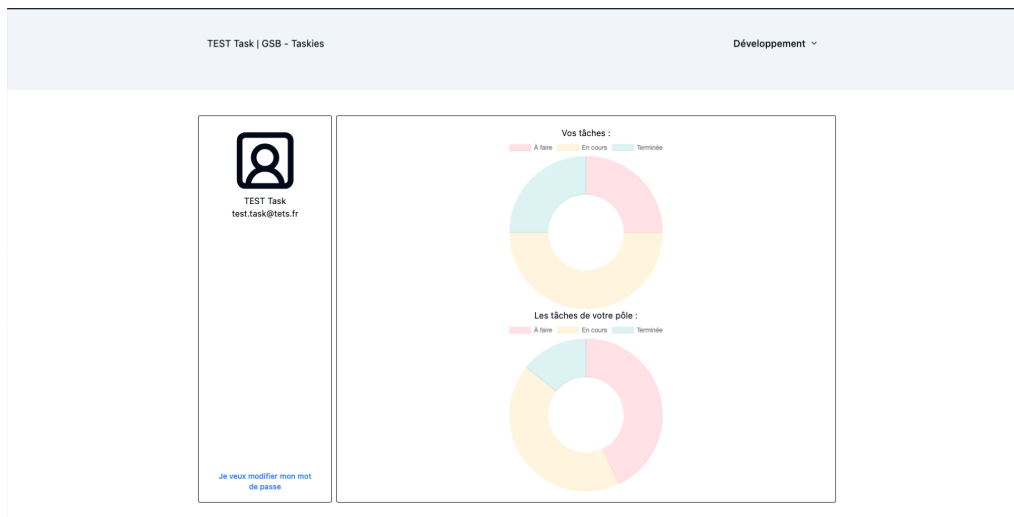
## Création de la page utilisateur

L'utilisateur :

- Peut retrouver ses informations
- Peut retrouver les statistiques sur ses tâches et celle de son pôle
- Modifier son mot de passe (l'url sera alors "/reset-password/token" pour ne pas utiliser les codes enregistrés en base de données)



## Page utilisateur



## Protection des routes sur l'application web

Il est important de protéger les routes. J'ai créé un composant `<ProtectedRoute>` qui permet de filtrer en fonction de si l'utilisateur est connecté ainsi que son rôle.

Les utilisateurs ayant le rôle Directeur (1), Référent (2) ou Employé (3) ne pourront pas accéder aux pages réservées aux administrateurs.

Cependant les utilisateurs possédant les droits Administrateurs peuvent accéder aux pages dédiés aux utilisateurs.

## Protection des routes API

Pour éviter que les données du projet soient accessibles en faisant simplement la requête, on va demander à ce que l'utilisateur soit connecté.

Pour vérifier que l'utilisateur est connecté, on va récupérer son token dans les headers de la requête. Ainsi, si le token est bon, la requête sera faite.

### middleware/verifyAuth.ts

```
4 export const verifyAuth = (req: Request, res: Response, next: NextFunction): void => {[!]  
5   const authHeader: string | undefined = req.headers["authorization"]  
6  
7   if (authHeader) {  
8     const bearer: string[] = authHeader.split(" ")  
9     const token: string = bearer[1]  
10    jwt.verify(token, process.env.SECRET_KEY!, {callback: (err: jwt.VerifyErrors | null, result: string | jwt.JwtPayload | unde...): void => {  
11      if (err) {  
12        res.status(401).send("Vous n'avez pas les accès.")  
13      } else {  
14        next()  
15      }  
16    }  
17  } else {  
18    res.status(401).send("Vous n'avez pas les accès.")  
19  }  
20  
21 }
```

Si l'utilisateur n'est pas connecté, le serveur lui retourne le status 401 avec un message "Vous n'avez pas les accès".

On va aussi protéger des routes qui ne seront accessibles uniquement aux utilisateurs ayant le rôle "admin" (0). Ces routes sont :

- la création d'un utilisateur "<https://test.klemens-galus.fr/users/addUser>"
- la modification d'un utilisateur "<https://test.klemens-galus.fr/users/>"
- la suppression d'un utilisateur "<https://test.klemens-galus.fr/users/>"

Pour chaque route on va demander le paramètre userId pour récupérer le rôle de l'utilisateur.  
[middleware/verifyRole.ts](#)

```
4 export const isAdmin = (req: Request, res: Response, next: NextFunction) : void => {
5     User.findById(req.params.userId) Query<...> & module:mongoose.Schema<any, Mo...
6     .then( onfulfilled: (docs : ... | null ) : void => {
7         if (docs && docs.role === 0) {
8             next()
9         } else {
10             res.status(401).send("Vous n'avez pas l'accès.")
11         }
12     }) Promise<void>
13     .catch((err) : void => {
14         console.log(err)
15     })
16 }
```

## Mise en prod

### Frontend

Le frontend a été mis en production à l'aide de Vercel : <https://ap-2-taskies.vercel.app/>

Vercel est une plateforme pour les développeurs front-end. Elle permet de déployer des sites statiques mais également des applications web, ce qui est notre cas.

La version qui est déployée sur Vercel va se mettre à jour à chaque fois que l'on push notre code sur la branche **main** de notre projet.

### Backend

L'api a été mise en ligne sur un serveur privé (VPS). Pour déployer mon api j'ai utilisé docker.

Pour ce faire, je me suis connectée à la VM via le protocole ssh.

J'ai pull mon répertoire sur github AP2-Taskies-API et j'y ai intégré un Dockerfile qui me permet de build l'image de mon API.

### Dockerfile :

```
FROM node:hydrogen-alpine

WORKDIR /app
COPY package.json ./
RUN npm install
COPY . .
EXPOSE 8080
CMD ["npm", "run", "dev"]
```

- *FROM node* : la base de notre image docker.
  - *hydrogen-alpine* permet de lancer un conteneur avec l'application node à l'intérieur. (hydrogen est la version de node utilisée, alpine est l'OS)
- *WORKDIR* : dossier où tout va se passer
- *COPY* : on va copier le package.json
- *RUN* : pour exécuter une commande, en l'occurrence nous voulons télécharger toutes les dépendances nécessaires à notre API
- *EXPOSE* : pour le port
- *CMD* : c'est la commande qui va être exécutée au premier lancement du conteneur

*Docker va utiliser des librairies de systèmes d'exploitation comme Ubuntu et les lancer sous forme d'un conteneur isolé du système (système de namespace).*

### Commande pour build l'image :

```
$ docker build -t ap2 .
```

### Commande pour lancer mon conteneur :

```
$ docker run -d -p 8081:8080 --name ap2-app ap2
```