

---

# Rapport de stage

**PLR  
CONSEIL** X



**FATOUMATA DIABATE**

Stage effectué du 06 Janvier au 07 Février  
2025

---

# Table des Matières

Remerciements

1-Introduction

2-Présentation de l'entreprise

2.1-Domaine d'activité

2.2-Cadre de travail

3 – Mission : Contexte et Objectifs

4-Mise en place de l'environnement de travail et technologies utilisées

5- Les différentes étapes (front-back) pour le développement de la nouvelle fonctionnalité : Affichage et sélection des professionnels les plus sollicités

Partie Front-End

Partie Back-End

6- Intégration du code, Résolution de Conflits et Retours Qualité du code

7-Gestion de projet

8- Compétences techniques acquises

9- Objectifs personnels atteints

10-Conclusion

Glossaire

Annexes

---

---

# Remerciements

Je souhaite exprimer ma profonde reconnaissance à toutes les personnes qui ont contribué à la réussite de mon stage et à l'élaboration de ce rapport.

Tout d'abord, je souhaite remercier chaleureusement mon maître de stage, Paul Larsonneur, fondateur de PLR Conseil, pour son accueil bienveillant, sa confiance et le partage de son expertise. Grâce à son encadrement, j'ai pu acquérir de nouvelles compétences et découvrir de bonnes pratiques dans le monde du développement logiciel.

Je remercie également toute l'équipe de Tauturu Fare pour leur collaboration et leur soutien tout au long de ce stage. Leur professionnalisme et leur engagement ont été d'une grande aide dans la réalisation de mes missions. Leur accompagnement m'a permis de découvrir l'univers de l'aide à domicile tout en évoluant dans un environnement professionnel stimulant.

Enfin, je tiens à exprimer ma gratitude envers les enseignants et responsables pédagogiques de l'EPSI pour la qualité de l'enseignement dispensé et pour leur soutien tout au long de ma formation.

---

# 1.Introduction

Durant ma deuxième année de BTS SIO option SLAM à l'EPSI, j'ai effectué un stage de 5 semaines au sein de l'entreprise **PLR Conseil**, fondée le 1er août 2021 par **Paul Larsonneur**, qui en est également le **directeur et** mon **tuteur de stage**. Cette société, experte dans les domaines du développement logiciel, est en collaboration avec **Tauturu Fare**, une entreprise spécialisée dans les services d'aide à domicile.

Ce rapport a pour objectif de présenter les missions qui m'ont été confiées, les compétences développées ainsi que les défis rencontrés au cours de cette expérience.

## 2.Présentation de l'entreprise

### 2.1.Domaine d'activité

Durant ce stage, j'ai été amené à travailler exclusivement sur des missions directement liées à Tauturu Fare.

Fondée le **13 mars 2024**, **Tauturu Fare** est une entreprise polynésienne qui se spécialise dans **la mise en relation entre travailleurs indépendants et particuliers** à la recherche de prestations à domicile. Actuellement, l'entreprise propose **des services tels que le ménage et le jardinage**, avec l'ambition d'étendre progressivement son offre à d'autres secteurs comme le **bien-être** (massage à domicile par exemple).

L'initiative est née de l'idée de **Heimoana Chuong**, un jeune polynésien ayant initialement suivi un **BTS en aquaculture** avant de se réorienter vers le développement informatique. Animé par un fort attachement à sa culture et à son territoire, il a souhaité créer une plateforme pensée par et pour les polynésiens, mettant en avant des valeurs de **proximité**, de **simplicité** et de **liberté**. Pour concrétiser son projet, il s'est associé à Paul Larssonneur, un développeur expérimenté, afin de donner vie à l'entreprise et à sa plateforme numérique.



**Logo de Tauturu Fare**

---

L'un des principaux atouts de Tauturu Fare est son approche ancrée dans la culture locale et centrée sur l'humain. Contrairement aux plateformes généralistes qui uniformisent leurs services, cette entreprise se distingue par une expérience utilisateur fluide, spécifiquement adaptée aux besoins et aux valeurs de la culture polynésienne. Grâce à l'expertise de Paul, la plateforme a été conçue pour offrir une interface intuitive, permettant une navigation simplifiée et une prise en charge rapide des demandes.

## 2.2.Cadre de travail

Ce stage s'est déroulé du **6 janvier au 7 février 2025**, entièrement en **télétravail**, une organisation qui m'a permis de travailler de manière autonome tout en restant en contact avec mon équipe via **Discord**. Cette configuration m'a offert plusieurs avantages, tels que la réduction du temps de trajet, la possibilité de travailler dans un environnement confortable et familier, et une diminution des interruptions fréquentes rencontrées dans un bureau traditionnel.

Cependant, cette expérience n'a pas été sans défis : il m'a fallu faire preuve de discipline personnelle pour rester concentré, maintenir une communication efficace à distance et gérer la séparation entre vie professionnelle et vie personnelle. Grâce à cette expérience, j'ai pu développer des compétences en gestion du temps, communication à distance, et organisation, en utilisant des outils numériques pour mener à bien les missions qui m'ont été confiées.

### 3.Mission : Contexte et Objectifs

Dans le cadre de mon stage sur le projet Tauturu Fare, j'ai été amené à intervenir sur l'application web, qui permet aux habitants de Polynésie de commander des services d'aide à domicile en toute simplicité.

L'accès à la plateforme est conçu pour être fluide et intuitif. Avant de pouvoir réserver un service, l'utilisateur doit créer un compte ou se connecter à son espace personnel. L'inscription est simplifiée et ne nécessite que quelques informations essentielles comme le nom, l'adresse e-mail et un mot de passe. Une fois inscrit, un e-mail de confirmation est envoyé pour activer le compte. La connexion peut ensuite s'effectuer avec une adresse e-mail et un mot de passe.

Une fois connecté, l'utilisateur peut naviguer sur l'interface principale et commencer le processus de réservation. Celui-ci débute par le choix du service souhaité, parmi ceux proposés par la plateforme. Chaque prestation est accompagnée d'une description détaillée et d'un tarif indicatif afin d'aider le client à faire son choix. Après cette sélection, il doit renseigner l'adresse où l'intervention aura lieu. L'étape suivante consiste à définir la date et l'horaire de la prestation. L'utilisateur choisit le créneau qui lui convient en fonction des créneaux disponibles. L'interface est conçue pour afficher uniquement les plages horaires où des professionnels sont potentiellement disponibles, réduisant ainsi les risques d'annulation. Et pour finir, l'utilisateur choisit un moyen de paiement (carte ou espèce).

Avant mon arrivée, une fois ces étapes complétées, la commande du client était créée. Un e-mail était automatiquement envoyé à l'ensemble des professionnels exerçant le service demandé. Ces derniers avaient ensuite la possibilité d'accepter la prestation selon leur disponibilité. Une fois la commande acceptée par un professionnel, celle-ci passait du statut 'Ouvert' à 'Accepté'.

**Ce mode de fonctionnement présentait certaines limites, notamment en termes d'expérience utilisateur et d'efficacité du système de mise en relation car les clients n'avaient aucun contrôle sur le professionnel sélectionné, et il pouvait arriver que la demande reste sans réponse si aucun prestataire n'était disponible ou intéressé.**

Conscient de ces limites, l'équipe a envisagé d'améliorer ce processus de réservation afin d'optimiser la mise en relation entre les clients et les prestataires. C'est dans ce cadre que j'ai été amené à travailler sur l'intégration d'une nouvelle fonctionnalité qui est ***l'affichage et la sélection des professionnels les plus sollicités*** dans l'application web.

L'objectif de cette nouvelle fonctionnalité est de permettre aux utilisateurs de visualiser et de sélectionner, pendant la création de leur commande, les prestataires qu'il a le plus sollicité auparavant. Cette nouvelle fonctionnalité reposera sur un affichage dynamique des professionnels récurrents. Lorsqu'un utilisateur remplit les informations de réservation (service, durée, horaire et lieu), et après s'être connecté, un nouvel écran devra apparaître. Cet écran devra présenter une grille mettant en avant les professionnels retenus. Ces derniers devront satisfaire plusieurs critères :

- Avoir au préalable traité la commande de l'utilisateur
- Exercer le service souhaité par l'utilisateur
- Le nombre maximal de professionnels affichables est de 3
- Si plus de 3, on prend les 3 premiers qui ont effectué le plus de prestations
- Chaque professionnel est affiché avec son nom, son prénom et sa note moyenne

L'affichage de ces professionnels sera accompagné d'un indicateur de disponibilité, calculé en fonction de l'emploi du temps du professionnel et de ses réservations déjà bookées.

Cette approche permettra ainsi à l'utilisateur de faire un choix éclairé : il pourra

- Soit sélectionner l'un des professionnels affichés et un **email personnalisé** sera envoyé uniquement à ce dernier, l'informant chaleureusement qu'il a été choisi comme favori par l'utilisateur.
- Soit décider de laisser sa demande ouverte en sélectionnant la carte "neutre" s'il ne souhaite pas choisir parmi les suggestions et dans ce cas, un **e-mail collectif** est envoyé, comme dans l'ancien système, à l'ensemble des professionnels exerçant le service demandé.

L'intégration de cette étape va améliorer le processus de réservation en le rendant plus interactif avec l'affichage des professionnels les plus sollicités par l'utilisateur et plus personnalisé avec l'envoi d'un e-mail spécifique dans le cas où celui-ci choisira un professionnel comme favori. Ce qui permettra de réduire les délais de réponse et d'optimiser la prise en charge des commandes, mais également de renforcer la confiance entre les utilisateurs et les prestataires.



## 4. Mise en place de l'environnement de travail et technologies utilisées

La plateforme de réservation de service à domicile de Tauturu Fare intègre plusieurs technologies complémentaires qui assurent sa robustesse et son efficacité. Ainsi, la **partie web** (qui est visible par les utilisateurs) a été développée avec **Angular**, tandis que la **partie backend** repose sur **NestJS**. Par ailleurs, **l'application mobile** est réalisée en Flutter. **MySQL** est utilisé comme base de données relationnelle, et **Sequelize** sert de couche d'abstraction afin de faciliter les interactions avec cette base via des objets JavaScript, plutôt que par l'écriture de requêtes SQL brutes. La conception des interfaces a été réalisée à l'aide de **Figma**, qui permet de créer des designs cohérents et adaptés aux besoins des utilisateurs.

La mise en place de mon environnement de travail a débuté par le clonage du dépôt existant sur GitLab. Comme je ne suis intervenue que sur **l'application web**, j'ai récupéré toutes les données et fichiers liés à celle-ci uniquement.

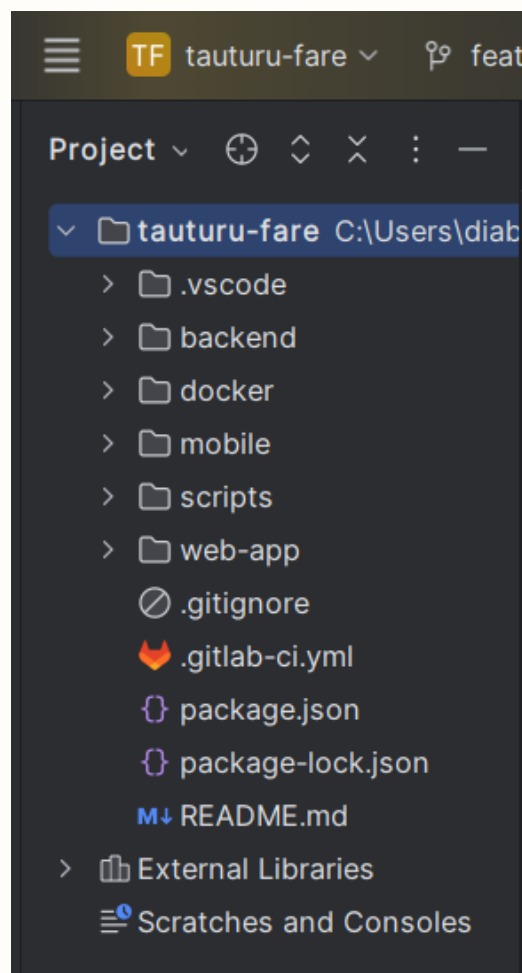
Afin d'assurer le bon fonctionnement du projet, j'ai installé l'ensemble des prérequis nécessaires, notamment **NodeJS, NPM, Angular CLI et NestJS CLI**, ainsi que les outils indispensables pour la gestion de la base de données, tels que **MySQL, le serveur MySQL et MySQL Workbench**.

Ces installations m'ont permis de disposer d'un environnement de développement complet et opérationnel pour travailler à la fois sur le front-end et le back-end de l'application.

Une fois l'environnement configuré, j'ai créé un fichier **.env** à la racine du dossier backend afin d'y stocker toutes les informations sensibles et nécessaires au bon fonctionnement de l'application. Ce fichier contient, entre autres, les informations de connexion à la base de données, la clé de l'API du **backend**, la clé Stripe pour le paiement en ligne, ainsi que les accès à Firebase, qui est utilisé pour l'hébergement, l'envoi de notifications et la remontée d'erreurs.

Par ailleurs, j'ai porté une attention particulière à la migration de la base de données. Dans le fichier `.sequelizerc`, situé à la racine du dossier backend, il a fallu que je procède à un ajustement de la configuration en remplaçant temporairement la date de référence (passant de 2024 à 2023) pour exécuter les commandes de migration et de seed. Après avoir appliqué ces commandes avec **npm run migration:up** et **npm run seed:up**, la configuration a été remise à jour en rétablissant la date initiale (2024) et j'ai exécuté les commandes de migration de nouveau pour finaliser le processus.

Ces étapes de préparation de l'environnement de travail ont constitué une phase essentielle pour me permettre d'appréhender correctement le projet et de me familiariser avec les technologies utilisées. Grâce à ces outils et à leurs configurations, j'ai pu aborder sereinement les missions qui m'ont été confiées et implémenter les tâches qui m'ont été affectées.



**Structure du projet Tauturu Fare**

## 5. Les différentes étapes (front-back) pour le développement de la nouvelle fonctionnalité : Affichage et sélection des professionnels les plus sollicités

Pour l'implémentation de cette nouvelle fonctionnalité, j'ai été amené à toucher à tous les composants de l'application: le front qui est la partie visuelle de l'application, le back qui expose les APIs et la BDD où sont stockées toutes les données.

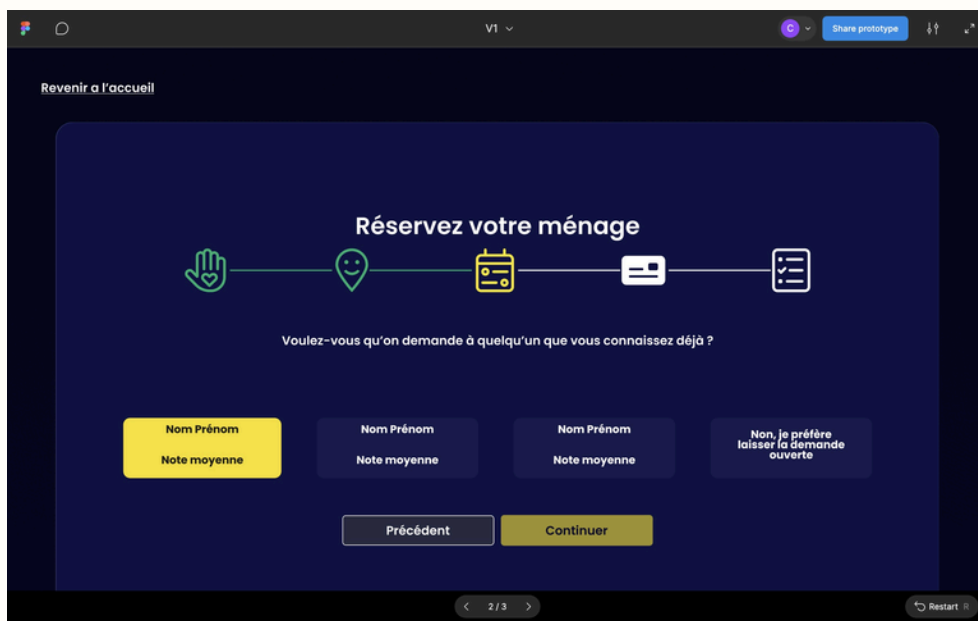
Ci-après, une présentation détaillée de chaque action que j'ai menée dans chacune de ces parties.

### Partie Front-End

#### Étape 1 : Conception du design sur Figma

J'ai débuté par la création d'un design pour la nouvelle étape de proposition de professionnels. Utilisant Figma, j'ai élaboré une interface affichant une grille pouvant présenter jusqu'à trois cartes de professionnels, chacune accompagnée d'un indicateur de disponibilité. J'ai également prévu une carte neutre qui offre l'option « Je ne veux pas choisir de patente ». Des animations au survol et un style de sélection ont été intégrés pour rendre l'expérience utilisateur plus interactive et agréable.

Une fois le design élaboré, j'ai présenté ma maquette à l'équipe. Cette validation a permis de recueillir des retours constructifs et d'apporter les ajustements nécessaires afin de s'assurer que l'interface respecte l'identité visuelle de la plateforme et réponde aux attentes fonctionnelles.



## Design final figma pour l'affichage et la sélection des pros favoris

### Étape 2 : Réordonnancement de l'étape de connexion

Pour intégrer efficacement la nouvelle fonctionnalité, j'ai réorganisé le workflow de réservation. L'étape de connexion a été déplacée pour être positionnée après la saisie des informations de durée, d'heure et de lieu, et avant le récapitulatif final de la commande. Ce réordonnancement garantit que la proposition des professionnels récurrents s'affiche au moment le plus opportun du parcours.




### Étape 3 : Implémentation de la nouvelle étape dans le workflow

J'ai intégré la nouvelle étape dans le processus de réservation. Dès que l'utilisateur renseigne les informations de réservation, un appel est effectué à un nouveau service back-end créé à cet effet pour récupérer la liste des professionnels les plus sollicités. Si des professionnels sont trouvés, l'interface affiche leur carte dans une grille ; sinon, on affiche pas cette page de sélection de professionnels et on garde le même comportement qu'avant, on passe directement à l'étape d'après.

**Réservez votre taille de haies et arbustes**

---

Voulez-vous qu'on demande à quelqu'un que vous connaissez déjà ?

 <b>Marius Bernard</b> Note moyenne : 4	 <b>Toto Tata</b> Note moyenne : 0
 <b>Eric Dupont</b> Note moyenne : 0	Non, je préfère laisser la demande ouverte

Précédent
Suivant

## L'affichage des pros favori pour le service demandé par le client

### Étape 4 : Gestion de la sélection utilisateur




Dans cette étape, j'ai mis en place la logique permettant à l'utilisateur de choisir un professionnel parmi ceux proposés ou de refuser cette sélection via l'option neutre. **Le choix effectué est stocké dans une variable, `providerIdChosenByClient`, afin d'être transmis ultérieurement lors de la création de la commande.**

Lorsqu'un professionnel est sélectionné par l'utilisateur, j'applique une couleur de fond **jaune** à la carte sélectionnée permettant d'indiquer son choix.

**Réservez votre taille de haies et arbustes**

---

Voulez-vous qu'on demande à quelqu'un que vous connaissez déjà ?

 <b>Marius Bernard</b> Note moyenne : 4	 <b>Toto Tata</b> Note moyenne : 0
 <b>Eric Dupont</b> Note moyenne : 0	Non, je préfère laisser la demande ouverte

Précédent
Suivant

## La sélection du pro favori par le client

Par défaut, aucun choix n'est effectué mais j'ai prévu un style lorsque la souris de l'utilisateur survole les différentes grilles, et l'affichage de "Indisponible pour la date et heure demandées" si le professionnel n'est pas disponible. **Voir Annexe 5**

**Réservez votre taille de haies et arbustes**

Voulez-vous qu'on demande à quelqu'un que vous connaissez déjà ?

<b>Marius Bernard</b> Note moyenne : 4	Indisponible pour la date et l'heure demandées
<b>Eric Dupont</b> Note moyenne : 0	Non, je préfère laisser la demande ouverte

Précédent Suivant

Dans le cas où l'utilisateur tente de cliquer sur "Suivant" sans avoir choisi aucune option (un professionnel ou l'option neutre), j'ai mis en place une fonction qui affiche un message d'erreur pour lui indiquer qu'un choix d'option est obligatoire pour pouvoir continuer, ainsi tant qu'il ne fait pas un choix, le bouton "Suivant" reste désactivé. **Voir Annexe 3**

**Réservez votre taille de haies et arbustes**

Voulez-vous qu'on demande à quelqu'un que vous connaissez déjà ?

<b>Marius Bernard</b> Note moyenne : 4	<b>Toto Tata</b> Note moyenne : 0
<b>Eric Dupont</b> Note moyenne : 0	Non, je préfère laisser la demande ouverte

Précédent Suivant

Veuillez sélectionner un professionnel ou choisir de laisser la demande ouverte

## Étape 5 : Passage à l'étape suivante et création de commande

Après que l'utilisateur ait sélectionné (ou non) un professionnel, il valide cette étape pour passer à la création de la commande. Le champ **providerIdChosenByClient** est alors envoyé au back-end via l'API order/create. Si ce champ est renseigné, la commande est créée avec un état spécifique, PROVIDER\_REQUIRED (contrairement à l'état **OPEN** qui était attribué jusqu'à présent), et le prestataire sélectionné est associé à la commande.

## Étape 6 : Adaptation de la page "Demandes de prestations"

La page "**Demandes de prestations**" est une page qui se trouve dans l'espace du professionnel, elle affiche toutes les commandes à l'état OPEN, c'est à dire toutes les commandes qui ont été envoyées à l'ensemble des prestataires offrant le service du fait d'une non sélection d'un professionnel par le client.

J'ai adapté cette page pour que, pour un professionnel connecté, l'API renvoie non seulement les commandes à l'état OPEN mais aussi celles à l'état PROVIDER\_REQUIRED qui lui ont été directement attribuées par des clients qui l'auraient directement choisis.

J'ai appliqué une mise en forme particulière avec une bordure plus épaisse aux commandes à l'état PROVIDER\_REQUIRED pour que le professionnel puisse les différencier des commandes à l'état OPEN.

 <b>KONÉ JEAN</b> K.JEAN@OPMAIL.COM	<b>A venir</b>				
	N°: 100	Commandé le: 29/01/2025	Longueur: 25 m	Adresse: 2 tt, ftt	Prix: 31000 F
	 <b>TAILLE DE HAIES ET ARBUSTES BIMENSUELLE</b> PRÉVU POUR LE 21 FÉVRIER 2025 à 18h30 En attente de confirmation par un professionnel				<b>Annuler</b>
	N°: 101	Commandé le: 29/01/2025	Durée: 01h00	Adresse: 2 tt, ftt	Prix: 2770 F
	 <b>MÉNAGE POUR PERSONNES ÂGÉES PONCTUELLE</b> PRÉVU POUR LE 22 FÉVRIER 2025 à 17h30 En attente de confirmation par un professionnel				<b>Annuler</b>
	N°: 109	Commandé le: 01/02/2025	Durée: 03h30	Adresse: 2 tt, ggg	Prix: 8750 F
	 <b>MÉNAGE PONCTUELLE</b> PRÉVU POUR LE 22 FÉVRIER 2025 à 08h30 ou (09h30, 10h30) En attente de confirmation par un professionnel				<b>Annuler</b>
	N°: 95	Commandé le: 28/01/2025	Longueur: 25 m	Adresse: 2 tt, hh	Prix: 31000 F
	 <b>TAILLE DE HAIES ET ARBUSTES BIMENSUELLE</b> PRÉVU POUR LE 22 FÉVRIER 2025 à 19h00 En attente de confirmation par un professionnel				<b>Annuler</b>
	N°: 132	Commandé le: 07/02/2025	Durée: 02h00	Adresse: 5 tt, gg	Prix: 7420 F
	 <b>ENTRETIEN DU JARDIN BIMENSUELLE</b> PRÉVU POUR LE 22 FÉVRIER 2025 à 09h30 Prestation assurée par: <b>Quocost</b>				<b>Annuler</b>

**Une bordure spécifique sur les commandes où le pro a été choisi comme favori**

## Étape 7 : Adaptation du composant Order Data View

Le composant **Order Data View** est le composant qui s'occupe de l'affichage de la page **Demandes de prestations**.



Enfin, j'ai mis à jour ce composant pour qu'il puisse reconnaître et différencier le nouvel état PROVIDER\_REQUIRED. Il applique désormais une bordure spécifique et un libellé distinct pour ces commandes, facilitant ainsi leur identification par les professionnels.



## Partie Back-End

### Étape 1 : Création d'une migration pour le nouvel état

J'ai débuté par la création d'une migration via Sequelize afin d'ajouter le nouvel état PROVIDER\_REQUIRED dans la table ORDER\_STATE. Pour cela, j'ai créé un dossier spécifique (2025) dans le répertoire des migrations, généré une migration avec la commande **npx sequelize-cli migration:generate --name add-provider-required-to-order-state**, puis appliqué les commandes de migration et de seed pour mettre à jour la base de données. **Voir Annexe 1**

Result Grid					Filter Rows:
	id	name			
▶	1	OPEN			
	2	ASSIGNED			
	3	DONE			
	4	PAID			
	5	ARCHIVED			
	6	CANCELED			
	7	CLOSED			
	8	REFUSED			
	9	EXPIRED			
	12	PROVIDER_REQUIRED			
⊕	NULL	NULL			

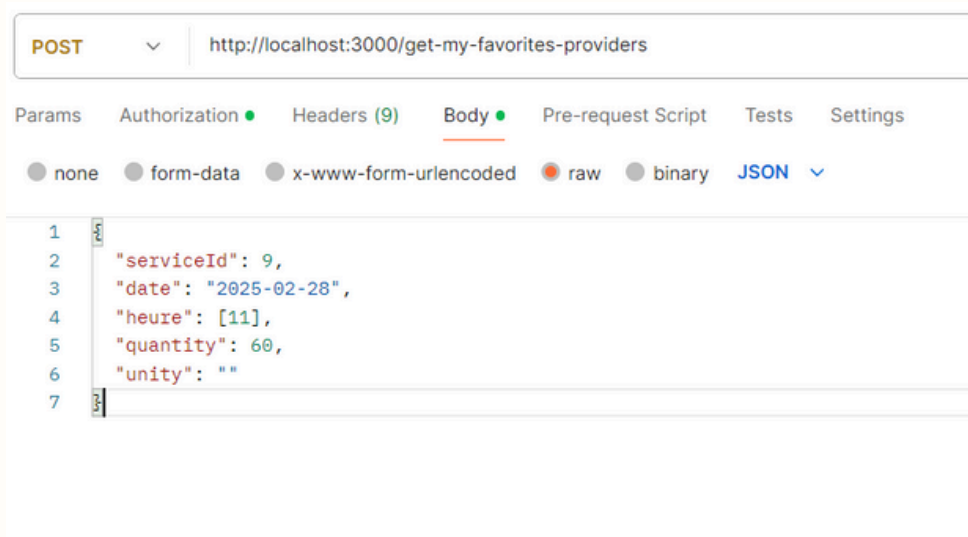
**Nouveau champ (PROVIDER\_REQUIRED) ajouté dans la table qui stocke les états des commandes**

## Étape 2 : Développement du service de récupération des professionnels récurrents

J'ai écrit une fonction dans le service utilisateur qui, à partir d'un ensemble de critères (userId, serviceId, date, heure, quantité et unité), renvoie les trois professionnels les plus sollicités par l'utilisateur pour le macro-service concerné. J'ai aussi écrit une fonction qui calcule la disponibilité de chaque professionnel renvoyé (attribut isAvailable), en vérifiant s'il y a un chevauchement d'horaires ou des blocages liés à la commande.

## Étape 3 : Création de l'appel API pour récupérer les professionnels

Pour exposer ce service au front-end, j'ai intégré l'appel dans un endpoint de l'API. Accessible via une requête POST à la route /get-my-favorites-providers, cet endpoint prend en paramètre un objet définissant les critères de recherche et retourne la liste des professionnels récurrents, avec leur disponibilité. **Voir Annexe 2**



**Point d'entrée pour l'affichage des pro les plus sollicités**

```
[
  {
    "providerId": 4,
    "nom": "Bernard",
    "prenom": "Marius",
    "noteMoyenne": 4,
    "totalOrdersNumber": 17,
    "oldestOrderDate": "2025-01-24",
    "isAvailable": true
  },
  {
    "providerId": 9,
    "nom": "Tata",
    "prenom": "Toto",
    "noteMoyenne": 0,
    "totalOrdersNumber": 14,
    "oldestOrderDate": "2025-01-31",
    "isAvailable": true
  },
  {
    "providerId": 3,
    "nom": "Dupont",
    "prenom": "Eric",
    "noteMoyenne": 0,
    "totalOrdersNumber": 12,
    "oldestOrderDate": "2025-01-25",
    "isAvailable": false
  }
]
```

**La réponse envoyé par l'API**

## Étape 4 : Modification de l'API order/create

J'ai adapté l'API existante de création de commande pour qu'elle accepte un nouveau champ, `providerIdChoosenByClient`, dans le payload. Lorsque ce champ est renseigné, la commande est créée avec l'état `PROVIDER_REQUIRED`, et le prestataire sélectionné est associé à la commande. Cette modification permet une personnalisation accrue de la réservation.

```
▼ Request Payload    view source
Request Payload ...}
  ▶ address: {streetNumber: 2, street: "bb", city: "bb", zipCode: "98735", isle: {id: 2, name: "Bora Bora"}}
  date: "2025-02-19"
  paymentMethodId: 2
  providerIdChoosenByClient: 4
  quantity: 100
  recurrenceId: 2
  serviceId: 9
  ▶ startHoursId: [9]
```

**Ce payload représente une commande créée avec l'état `PROVIDER_REQUIRED` car le client a sélectionné le professionnel qu'il souhaite garder pour sa commande.**

```
▼ Request Payload    view source
▼ {serviceId: 9,...}
  ▶ address: {streetNumber: 2, street: "ff", city: "ff", zipCode: "98735", isle: {id: 2, name: "Bora Bora"}}
  date: "2025-02-21"
  paymentMethodId: 2
  providerIdChoosenByClient: null
  quantity: 200
  recurrenceId: 2
  serviceId: 9
  ▶ startHoursId: [10]
```

**Ce payload représente une commande créée avec l'état `OPEN` car le client n'a pas choisi de prestataire, donc un mail collectif est envoyé à l'ensemble de professionnels qui effectuent le service demandé.**

## Étape 5 : Mise en place d'un template d'e-mail personnalisé

Avant, lors de la création d'une commande par le client, un e-mail collectif était envoyé à tous les professionnels concernés par le service demandé.



### Ancien template de mail collectif envoyé à tous les pro concerné par le service si le client n'a sélectionné aucun pro affiché

J'ai donc j'ai conçu un template d'e-mail personnalisé sur SendGrid. Ce template est utilisé dans le service d'envoi d'e-mails (sendmail.service) lors de la création d'une commande avec le champ providerIdChoosenByClient, afin d'informer chaleureusement le professionnel sélectionné qu'il a été choisi comme favori par le client.

## TAUTURU FARE

---

### Une nouvelle mission pour le service Taille de haies et arbustes, rien que pour vous !

Ia ora na Eric,

Nous sommes ravis de vous informer qu'un client pour lequel vous avez déjà réalisé une prestation souhaite à nouveau faire appel à vos talents !

Jean, a demandé le service Taille de haies et arbustes et a apprécié votre mana et travail et serait ravi que vous interveniez avec une fois de plus.

Merci pour votre expertise, nous sommes fiers de vous compter parmi nos professionnels de confiance !

Pour découvrir cette nouvelle demande, rendez-vous sans attendre sur votre espace pro.

Demandes ouvertes

Maururu,  
L'équipe de support de Tauturu Fare.

---

## Nouveau template de mail personnalisé pour un pro sélectionné en tant que favori par le client

Désormais, lorsque le champ **providerIdChosenByClient** est renseigné, seul le professionnel sélectionné reçoit l'e-mail personnalisé. En revanche, si aucun choix n'a été fait par l'utilisateur, on conserve le fonctionnement précédent et on envoie l'e-mail collectif à tous les prestataires effectuant le service demandé.

---

## **Étape 6 : Adaptation de l'API des commandes ouvertes**

Enfin, j'ai modifié l'API qui retourne les commandes ouvertes pour qu'elle prenne désormais en compte le nouvel état PROVIDER\_REQUIRED. Ainsi, pour un professionnel connecté, l'API renvoie non seulement les commandes en état OPEN, mais aussi celles en état PROVIDER\_REQUIRED qui lui sont associées, avec une présentation spécifique dans l'interface.

L'ensemble de ces développements ont été accompagnés de tests unitaires pour garantir la robustesse de la solution, avec notamment deux tests réalisés sur le service d'envoi d'e-mails et un test sur le endpoint de récupération des professionnels récurrents.

Ces différentes étapes, allant du design initial sur Figma à l'adaptation des API back-end, illustrent le travail complet que j'ai réalisé pour intégrer cette fonctionnalité innovante.

L'intégration de cette nouvelle fonctionnalité a permis d'améliorer le processus de réservation de service d'aide à domicile sur Tauturu Fare. Alors qu'auparavant, la commande se créait avec un envoi massif d'e-mails aux prestataires concernés par le service demandé, le nouveau système offre désormais aux utilisateurs la possibilité de sélectionner directement parmi les professionnels qu'ils ont le plus sollicité, tout en assurant une communication personnalisée grâce à un template d'e-mail spécialement conçu. Cette évolution améliore significativement l'expérience utilisateur et la qualité du service, tout en renforçant la relation de confiance entre les clients et les professionnels.

---

## 6.Intégration du code, Résolution de Conflits et Retours Qualité du code

Après avoir rendu opérationnelle cette nouvelle fonctionnalité, j'ai poussé mon code sur GitLab et assigné la relecture à mon tuteur de stage. Dans cette phase, j'ai rencontré quelques conflits lors du merge vers la branche principale (dev). J'ai alors pris le temps de résoudre minutieusement ces conflits pour assurer une intégration harmonieuse de mes modifications dans l'ensemble du projet, garantissant ainsi la stabilité et la cohérence du code existant.

Mon tuteur de stage m'a ensuite fourni des retours détaillés, mettant en avant plusieurs aspects essentiels : la qualité du code, le respect des bonnes pratiques de développement en entreprise, et la nécessité d'adapter mes contributions aux standards déjà établis dans le projet. Ces observations ont été particulièrement précieuses, me permettant d'affiner mon travail et d'améliorer la lisibilité, la maintenabilité et la cohérence de mon code.

Cette phase d'intégration et de validation m'a fait savoir l'importance cruciale des retours constructifs dans un environnement professionnel. Les retours reçus m'ont non seulement permis d'optimiser mon code, mais également de renforcer ma compréhension des standards de développement en entreprise, favorisant ainsi ma progression en tant que développeur collaboratif.



---

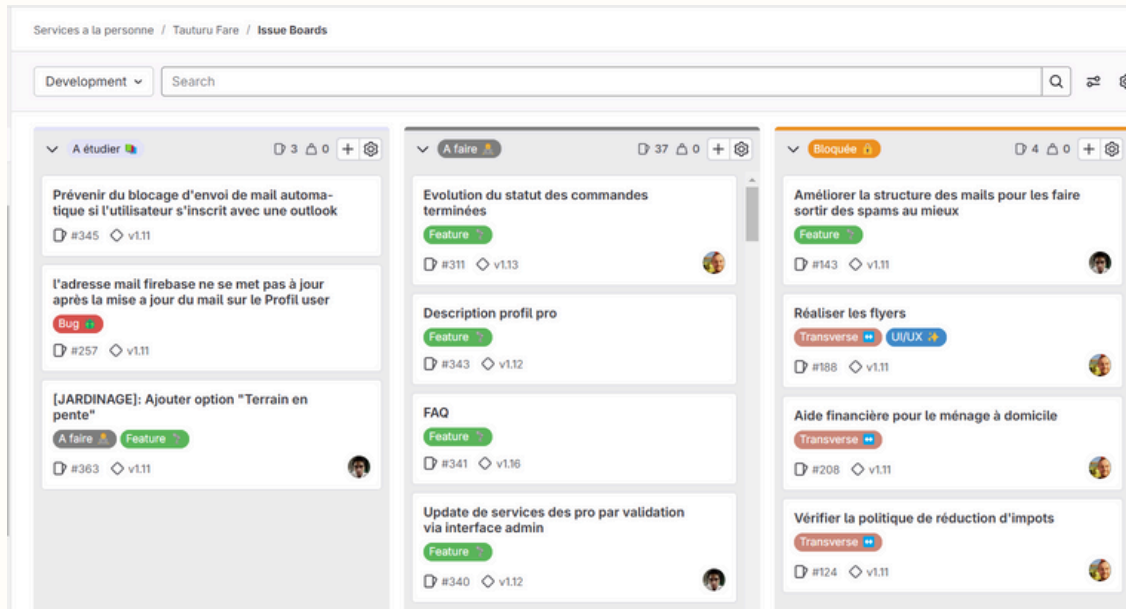
## 7. Gestion de projet

Durant ce stage, chaque jour (du lundi au vendredi), nous tenions des réunions quotidiennes, appelées "daily meetings", via Discord. Ces réunions avaient pour objectif de coordonner l'équipe en faisant le point sur les avancées de chacun, d'identifier les obstacles éventuels et de planifier les actions à venir.

Parallèlement, des réunions hebdomadaires (chaque vendredi) étaient organisées pour prendre du recul sur le travail accompli et discuter des orientations futures du projet. Ces réunions permettaient d'évaluer les progrès réalisés, d'ajuster les priorités et de s'assurer que tous les membres de l'équipe partageaient les objectifs stratégiques de Tauturu Fare.

L'utilisation de Discord pour ces réunions facilitait les échanges, renforçant ainsi la cohésion de l'équipe malgré la distance. De plus, ces réunions m'offraient l'opportunité de m'exprimer librement, de poser des questions concernant ma mission et de proposer mes idées. Cette dynamique participative m'a permis de contribuer activement au projet et de me sentir pleinement intégré au sein de l'équipe.

Pour le suivi des tâches et la gestion du code, nous utilisons GitLab. Cette plateforme centralisait les dépôts de code, facilitait le versioning et permettait une collaboration efficace entre les membres de l'équipe. Les fonctionnalités de suivi des issues et de gestion de projet de GitLab nous aidaient à organiser, prioriser et suivre l'avancement des différentes tâches.



## Organisation et suivi des tâches du projet Tauturu Fare sur GitLab

## 8. Compétences techniques acquises

Au cours de ce stage, j'ai eu l'opportunité d'élargir mes connaissances et d'acquérir de nouvelles compétences techniques, en particulier sur des technologies que je n'avais pas encore eu l'occasion de découvrir en profondeur.

- **Développement Backend avec NestJS**

Ce stage m'a permis de découvrir et d'apprendre **NestJS**, un framework backend basé sur Node.js et TypeScript. J'ai pu comprendre son architecture modulaire et l'importance des **contrôleurs**, **services**, **DTO** dans la gestion des requêtes et du traitement des données. Grâce à cette expérience, j'ai appris à :

- Structurer une application backend de manière propre et évolutive.
- Manipuler des API REST avec NestJS.
- Gérer l'authentification et la validation des données.

- **Développement Frontend avec Angular**

Avant ce stage, je n'avais jamais travaillé avec **Angular**, et cette expérience m'a permis d'acquérir des compétences solides sur ce framework frontend. J'ai appris à :

- Créer et organiser des **composants réutilisables**.
- Utiliser **les services et l'injection de dépendances** pour communiquer avec le backend.
- Optimiser l'interface utilisateur avec **SCSS** pour améliorer l'expérience utilisateur.

- **Intégration HTML / SCSS et optimisation UI/UX**

- J'ai perfectionné mes compétences en HTML et SCSS, notamment pour créer des interfaces modernes et responsives.
- J'ai appris à utiliser les variables SCSS pour maintenir beaucoup de cohérences dans le code.

- **Utilisation de GitLab et gestion des versions**

- J'ai appris à travailler efficacement avec GitLab, en utilisant les branches et en appliquant les bonnes pratiques du versioning.
- J'ai gagné en autonomie sur la résolution des conflits lors des merges et sur la gestion des merge requests avec relecture de code.

- **Approche Agile et gestion de projet**

- Participation aux daily meetings pour suivre l'avancement du projet.
- Réunions hebdomadaires pour planifier les futures évolutions de l'application.

---

## 9.Objectifs personnels atteints

Avant de commencer ce stage, je m'étais fixée plusieurs objectifs afin de maximiser mon apprentissage et de devenir plus autonome en tant que développeuse.

### **Maîtriser de nouvelles technologies (NestJS & Angular)**

Grâce à ce stage, j'ai découvert et appris **deux frameworks majeurs** utilisés en entreprise. NestJS m'a permis de mieux structurer une application backend, et Angular m'a appris à gérer un projet frontend complexe.

### **Développer une fonctionnalité complète en autonomie**

J'ai pu travailler sur une **nouvelle fonctionnalité de sélection des prestataires**, en développant **tant le frontend que le backend**. Cela m'a permis de comprendre l'ensemble du processus de développement et d'apprentissage en entreprise.

### **Améliorer mes compétences en gestion de projet et en travail d'équipe**

Les *daily meetings* et les **revues de code** m'ont aidée à mieux collaborer avec l'équipe et à intégrer les retours de mon tuteur pour améliorer la qualité de mon code.

### **Adopter des bonnes pratiques et structurer mon code**

J'ai appris à écrire un **code plus propre et maintenable**, en respectant les conventions et standards utilisés dans l'entreprise.

---

# 10.Conclusion

Ce stage a été pour moi une expérience extrêmement enrichissante tant sur le plan professionnel que personnel. Il m'a permis d'acquérir des compétences techniques solides, tant en **backend** qu'en **frontend**, tout en me familiarisant avec des outils modernes comme **NestJS** et **Angular**. Au-delà des compétences techniques, ce stage m'a offert une immersion précieuse dans le monde du développement en entreprise, où j'ai pu évoluer dans un environnement agile et collaboratif.

J'ai eu l'opportunité de travailler sur un projet réel, d'interagir avec une équipe compétente et bienveillante, de suivre l'avancement du projet et de participer activement à la planification des futures évolutions de l'application à travers les **daily meetings** et les **réunions hebdomadaires**. Ces échanges réguliers m'ont permis de comprendre les enjeux d'un travail collaboratif efficace et d'adopter une approche méthodique pour résoudre les problèmes.

En outre, les retours constructifs de mon tuteur de stage ont été déterminants dans ma progression. Ils m'ont permis de perfectionner mon code, d'intégrer les bonnes pratiques de développement, et de mieux comprendre les exigences liées à l'adaptation aux standards d'une entreprise.

Enfin, au-delà des compétences techniques, ce stage m'a aussi permis de renforcer mes compétences en travail d'équipe et en communication. L'ambiance collaborative et le soutien constant de mon tuteur ont été des éléments clés qui ont favorisé mon développement professionnel. Cette expérience m'a non seulement permis d'améliorer mes compétences en développement, mais elle m'a également donné une vision plus claire de la réalité du travail en entreprise, me préparant ainsi de manière plus solide à mes futurs projets professionnels.

---

# Glossaire

- **API (Application Programming Interface)** : Interface permettant à deux logiciels de communiquer entre eux.
- **Endpoint** dans une API, est une URL spécifique qui permet aux clients d'accéder à une ressource ou à une fonctionnalité donnée. Il représente un point d'entrée où une requête peut être envoyée pour interagir avec l'API, généralement via des méthodes HTTP (GET, POST, PUT, DELETE)
- **Un service** contient la logique métier de l'application. Il est utilisé par le contrôleur pour effectuer des opérations complexes, interagir avec la base de données ou appeler d'autres services.
- **Un contrôleur** est une composante d'une application qui gère les requêtes entrantes, traite les données si nécessaire (souvent en appelant un service) et retourne une réponse au client. Il sert d'intermédiaire entre le client et la logique métier.
- **DTO** : Elle sert d'intermédiaire entre le Service et le Controller. Les DTO permettent de cacher la structure des entités de la base de données et d'éviter d'exposer trop d'informations.
- **Backend** : Partie d'une application qui gère les données et la logique, souvent invisible pour l'utilisateur.
- **Frontend** : Partie d'une application avec laquelle l'utilisateur interagit directement, incluant l'interface graphique.
- **Angular** : Framework JavaScript open-source utilisé pour le développement d'applications web front-end. Angular permet de créer des applications dynamiques et réactives en utilisant des composants et des services.
- **NestJS** : Framework backend basé sur Node.js et TypeScript, conçu pour créer des applications serveur efficaces et évolutives. Il utilise une architecture modulaire pour faciliter l'organisation du code.

- **Sequelize** : ORM (Object-Relational Mapping) pour Node.js qui facilite l'interaction avec des bases de données SQL, permettant de manipuler des objets JavaScript au lieu d'écrire des requêtes SQL brutes.
- **Figma** : Outil de design collaboratif utilisé pour la conception d'interfaces utilisateur, permet de créer des maquettes interactives et de collaborer en temps réel.
- **Node.js** : Environnement d'exécution JavaScript côté serveur permettant de développer des applications web performantes et évolutives.
- **NPM** : Node Package Manager, un gestionnaire de paquets pour le JavaScript, utilisé pour installer et gérer les dépendances nécessaires au projet.
- **Angular CLI** : Interface en ligne de commande pour Angular, permettant de faciliter le développement d'applications en automatisant des tâches courantes comme la génération de composants ou le lancement de l'application.
- **NestJS CLI** : Interface en ligne de commande pour NestJS, facilitant la gestion du projet et la création de modules, services et contrôleurs.
- **.env** : Fichier de configuration utilisé pour stocker des variables d'environnement sensibles et spécifiques à une application, telles que les clés d'API ou les informations de connexion à la base de données.
- **Firebase** : Plateforme de Google qui fournit une gamme de services backend pour les applications web et mobiles, notamment l'hébergement, l'authentification, l'envoi de notifications push, et le stockage de données en temps réel.
- **Stripe** : Service de paiement en ligne qui permet de traiter les transactions sur des plateformes de commerce électronique. Il fournit des API pour gérer les paiements par carte bancaire.
- **Sequelize CLI** : Outil en ligne de commande utilisé avec Sequelize pour gérer les migrations de base de données, les seeds (données initiales) et les configurations de modèle.

- **MySQL** : Système de gestion de base de données relationnelle (SGBDR) largement utilisé pour stocker et organiser les données dans une application web. Il repose sur le langage SQL pour manipuler les données.
- **Migration de base de données** : Processus consistant à mettre à jour la structure de la base de données, par exemple en ajoutant des tables ou en modifiant des colonnes, tout en conservant les données existantes.
- **Seeding de base de données** : Remplissage initial de la base de données avec des données d'exemple ou des valeurs par défaut, souvent utilisé pour tester l'application après la configuration.
- **GitLab** : Plateforme de gestion de version et de collaboration permettant de suivre les modifications de code, de gérer des projets et d'assurer un flux de travail continu grâce à des outils comme les merge requests et les pipelines CI/CD.
- **.sequelizeerc** : Fichier de configuration de Sequelize utilisé pour personnaliser les paramètres de migration et les chemins des fichiers de modèle ou de migration.
- **HTML (HyperText Markup Language)** : Langage de balisage standard utilisé pour créer la structure de pages web. HTML permet de définir des éléments tels que les titres, les paragraphes, les images, les liens et les tableaux.
- **SCSS (Sassy CSS) : Superset de CSS (Cascading Style Sheets)** qui ajoute des fonctionnalités supplémentaires comme les variables, les mixins et les fonctions. SCSS permet de rendre le code CSS plus modulaire, réutilisable et maintenable, facilitant ainsi la gestion de styles complexes.
- **Workflow** : Aussi appelé flux de travaux ou flux opérationnel, désigne une suite de tâches ou d'opérations qui doivent être réalisées par un individu ou un groupe d'individus selon un ordre spécifique.
- **Payload**: Correspond aux données envoyées dans le corps (body) d'une requête HTTP.



# Annexes

## Annexe 1 : Ajout du nouvel état (PROVIDER\_REQUIRED) dans la table ORDER\_STATE

```
'use strict';

/** @type {import('sequelize-cli').Migration} */
module.exports = {
  async up (queryInterface :QueryInterface , Sequelize :...) :Promise<void> {
    //Insert new order state
    await queryInterface.bulkInsert( {tableName: 'ORDER_STATE',
      records: [{name: 'PROVIDER_REQUIRED'}]
    });
  },
  async down (queryInterface :QueryInterface , Sequelize :...) :Promise<void> {
    //Delete new order state
    await queryInterface.bulkDelete( {tableName: 'ORDER_STATE', identifier: {name: 'PROVIDER_REQUIRED'}};
  }
};
```

## Annexe 2 : API permettant d'exposer les pros récurrents au front

```
@Post('get-my-favorites-providers') Show usages  ⓘ diaba
@Throttle({ default: { limit: 4, ttl: 60000 } })
@UseGuards(FirebaseAuthGuard)
async getMyFavoritesProviders(
  @UserId() userId: string,
  @Body() favoriteProviderCriteria: FavoriteProviderCriteriaDto,
): Promise<ProFavoriteLiteDto[]> {
  return await this.userService.getMyFavoritesProviders(
    userId,
    favoriteProviderCriteria,
  );
}
```

## Annexe 3 : Fonction permettant d'afficher un message d'erreur lorsque le client n'a sélectionné aucune option

```
validateSelection(): boolean { Show usages  ⓘ diaba
  if(this.selectedOptionIndex === undefined) {
    this.toastService.addErrorToast("Veuillez sélectionner un professionnel ou choisir de laisser la demande ouverte");
    return false;
  }
  return true;
}
```

## Annexe 4 : Fonction écrite dans (sendmail.service) pour récupérer dynamiquement des infos du pro choisi et le client pour une meilleure personnalisation du nouveau template de mail

```
async mailToFavoriteChooosenProvider( Show usages  ▴ diaba
  providerInfo: ProviderNameAndServiceAndEmail,
): Promise<void> {
  try {
    const email :string = providerInfo.email;
    const name :string = providerInfo.name;
    const service :string = providerInfo.service;
    const userName :string = providerInfo.userName;

    const msg = {
      to: email,
      from: this.senderMail,
      templateId: 'd-816b5d3662a846fe97b622de4057b9dd',
      dynamic_template_data: {
        subject: `Une nouvelle mission ${service} vous attend !`,
        name,
        service,
        userName,
        url: process.env.AUTHORIZED_URLS.split(',')[0] + '/pro/open-orders',
      },
    };
    await sgMail.send(msg);
    this.logger.log(
      'Envoi de mail pour informer chaleureusement le pro qu il a été choisi par le client',
      'mailToProviderChooosenByClient',
    );
  } catch (error) {
    this.logger.error(
      "Une erreur est survenue lors de l'envoi du mail au pro l'informant qu'il a été choisi par le client",
      error.response,
      'mailToProviderChooosenByClient',
    );
  }
}
```

## Annexe 5 : Bloc de code permettant l'affichage dynamique des pros récurrents avec indice de disponibilité

```
<p>Voulez-vous qu'on demande à quelqu'un que vous connaissez déjà ?</p>

<div class="options">
  <button
    *ngFor="let provider of providers; let i = index"
    class="option"
    [class.selected]="selectedOptionIndex === i"
    [class.unavailable]="!provider.isAvailable"
    (click)="provider.isAvailable && selectOption(i, provider.providerId)"
    [disabled]="!provider.isAvailable"
    (click)="selectOption(i, provider.providerId)"
  >
    <div class="card-content">
      
      <div class="info">
        <h3>{{provider.prenom}} {{provider.nom}}</h3>
        <p>Note moyenne : {{provider.noteMoyenne}}</p>
      </div>
    </div>
    <div *ngIf="!provider.isAvailable" class="overlay">
      Indisponible pour la date et l'heure demandées
    </div>
  </button>
  <button
    class="option"
    [class.selected]="selectedOptionIndex === -1"
    (click)="selectOption(index: -1, providerId: null)"
  >
    <div class="none">Non, je préfère laisser la demande ouverte</div>
  </button>
</div>
```