

# Rapport d'alternance

## Développeur Full-Stack

---

Christopher MUFU

Novembre 2020 à Août 2021

Tuteur pédagogique : Jean-Christophe Dubacq

Maîtres d'apprentissage : Laurent Goulenok | Magaye Lopes

Etablissement | Formation : Université Sorbonne Paris Nord

Entreprise d'accueil : CEOS TECH

DUT Informatique

Session 2019-2021

## Remerciements

Avant de débuter mon rapport d'alternance, je tenais tout d'abord à remercier les acteurs qui m'ont permis d'aboutir à une prise de poste en tant qu'apprenti. Je remercie dans un premier temps mon entreprise d'accueil CEOS TECH, qui a eu confiance en mon profil et m'a donné l'opportunité d'intégrer leur projet. Je remercie également les membres de l'Université Sorbonne Paris Nord et de l'IUT de Villetaneuse qui ont énormément contribué à ma recherche d'alternance ainsi qu'à l'amélioration de mes CVs et lettres de motivation.

J'en profite pour remercier personnellement mes maîtres d'apprentissage Laurent Goulenok et Magaye Lopes de leur attention au contenu de mon intégration et de mes missions au sein de l'entreprise. Je remercie mon tuteur pédagogique Jean-Christophe Dubacq de sa disponibilité et son soutien à l'égard de ma personne ainsi qu'à la rédaction de ce rapport. Je souhaiterais aussi remercier plus personnellement les stagiaires Raj Porus Hiruthayaraj, Kelton Semedo, Anne Bequet, Abdelhadi Bayechou et Mustapha Jaridi qui ont su être d'une grande aide à l'évolution de l'entreprise CEOS TECH et qu'ils m'ont également fait beaucoup évoluer dans les différents aspects de mon métier.

---

## Sommaire

Page de remerciements.....	1
Sommaire.....	2
Introduction.....	3
Partie 1 : Présentation de CEOS TECH	
La Société.....	4
L'Équipe.....	4
Partie 2 : Poste et environnement de travail	
Mon poste et mes missions.....	5
Disposition de l'équipe et outils de travail.....	5
Contexte de travail et journée type.....	6
Partie 3 : Chronologies & présentation des projets	
Présentation des projets.....	7
Frise chronologique des projets.....	7
Partie 4 : Missions et tâches	
Présentation de l'application MARKUS.....	8
Missions et tâches du projet MARKUS.....	9
Présentation du site internet O'Lokoso.....	16
Missions et tâches du projet O'Lokoso.....	18
Présentation du site internet Délice-DF5.....	24
Missions et tâches du projet Délice-DF5.....	25
Mes autres projets.....	31
Mes difficultés.....	35
Partie 5 : Bilan d'expérience.....	38
Conclusion.....	39
Indexes.....	40
Liens et références.....	41

## Introduction

Mon nom est Christopher MUFU, je suis étudiant en DUT Informatique au sein de l'Université Sorbonne Paris Nord et apprenti Développeur Full-Stack au sein de la jeune agence informatique CEOS TECH.

À travers ce rapport d'activité en entreprise, je vous détaillerai mon entreprise d'accueil ainsi que le pôle d'activité dont je suis affilié, je préciserai par la suite la composition de l'équipe ainsi que mon environnement de travail, puis après vous avoir décrit au préalable la chronologie de l'ensemble des projets auquel j'ai pu participer, j'entamerai le détail de mes tâches et missions pour chacun de ces différents projets au sein de CEOS TECH, et enfin je fournirai un dernier bilan d'expérience avant de finir par une conclusion.

## Partie 1 : Présentation de CEOS TECH

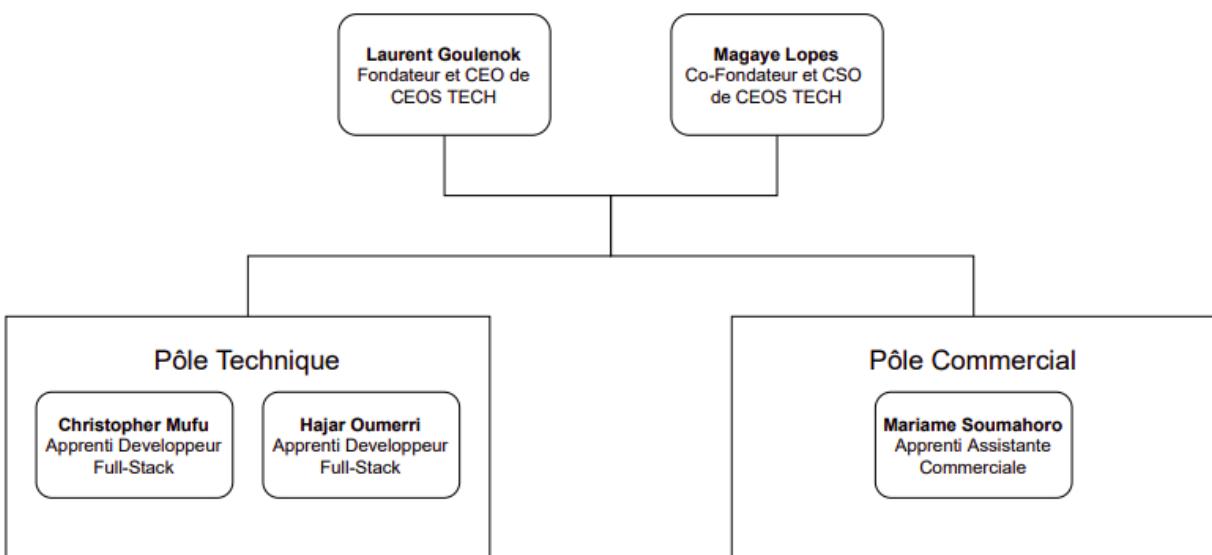
### La Société :

CEOS TECH est une jeune société ayant pour objet la conception, l'édition ainsi que le développement et l'exploitation de logiciels informatiques. Cela comprend les sites internet et applications mobiles, dans tous les domaines d'activités. La société propose également des services en matière de contenus rédactionnels, graphiques, photographiques, audiovisuels ou illustratifs, via le prisme de formations dédiées (formation à rédaction de business plan, formation design, formation no-code). L'entreprise dispose d'un champ d'action situé en France mais elle possède néanmoins l'ambition d'élargir ce champ d'action à l'international.

### L'Équipe :

Concernant la disposition de l'agence, nous sommes une équipe de moins de dix employés disposée en deux pôles. Nous avons le pôle commercial chargé en partie de la communication autour de CEOS TECH ainsi que de ces différents projets et le pôle technique dont je fais partie et ayant pour rôle le développement de logiciels informatiques ou sites internet pour des entreprises clientes mais également pour des projets lancés directement par CEOS TECH.

### Organigramme de l'entreprise :



Au fur et à mesure de son évolution, l'agence a su diversifier ses pôles d'activités en intégrant notamment un pôle spécialisé dans le design d'applications et de sites internet ainsi qu'un pôle dédié au machine learning et l'analyse de données. Néanmoins, l'équipe fluctue régulièrement de par la participation passagère de stagiaires pour nous prêter main forte dans les différents pôles.

Après vous avoir présenté à l'échelle globale la société CEOS TECH, nous changeons de focalisation afin de nous concentrer sur la description de mon poste ainsi que de mon environnement de travail.

## Partie 2 : Poste et environnement de travail

### Mon poste et mes missions :

Comme nous l'avions vu plutôt à travers l'organigramme de la société et mon introduction, j'exerce le poste de Développeur Full Stack chez CEOS TECH, j'entreprends les missions se référant à l'aspect visuel, à interface et à l'expérience utilisateur de l'application, autrement dit le front-end. J'interviens également dans l'aspect interne de l'application avec la récolte, la sécurisation ainsi que la transmission des données générées, autrement dit le back-end.

### Disposition de l'équipe et outils de travail :

Au sein du pôle technique, nous sommes deux alternants affiliés à plein temps aux différents projets de l'entreprise, Hajar Oumerri et moi-même. Nous sommes généralement accompagnés par au moins un stagiaire spécialisé dans les langages de programmation que nous utilisons.

Concernant ces langages de programmation, nous utilisons dans un premier temps le framework React Native pour le front-end destiné aux applications mobiles, il s'agit d'une technologie conçue par Facebook et utilisant le langage Javascript pour le développement d'applications natives<sup>(1)</sup> sur le principe de ReactJS. ReactJS est notamment la seconde technologie que nous utilisons au niveau du front-end, une technologie dédiée cette fois-ci à l'édition de sites internet. Plus précisément, ReactJS est une bibliothèque JavaScript également conçue par Facebook et facilitant la création de sites internet dynamiques grâce à notamment l'utilisation du fameux state<sup>(2)</sup> ou composante d'états en français.

Pour le back-end, nous nous servons du langage Python associé au framework Django, une technologie utilisée à la base pour la conception rapide de sites web, mais dans notre cas nous l'utilisons pour la réalisation de requêtes d'API<sup>(3)</sup> en lien avec l'application. Pour cela nous couplons l'utilisation de Django avec le framework Django REST, une technologie dédiée à l'utilisation de l'API REST et permettant notamment une meilleure optimisation du code.

Au niveau de mes outils de travail, j'utilise pour travailler l'IDE<sup>(4)</sup> Atom, initialement conçu pour faciliter le développement de la partie front-end d'une application, de par ces nombreux packages il est possible de l'utiliser pour une large panel de langages. A noter que lorsque je possède des soucis de performance avec cet IDE je me dirige vers l'IDE GNOME Builder, un IDE fourni par les distributions Linux, utile pour du développement en langage C ou C++ même s'il est totalement possible de coder en d'autres langages. Malgré une coloration syntaxique assez pauvre, il possède néanmoins une bonne fluidité d'utilisation, d'où la raison principale de ce choix d'IDE dans mon cas. Néanmoins je suis arrivé à bon compromis en utilisant l'IDE fourni par Sublime Text qui combine les bons points des deux précédant IDE et que j'utilise désormais activement. Concernant mon matériel informatique, je dispose d'un ordinateur portable qui me sert à la réalisation de mes tâches en entreprise.

## Contexte de travail et journée type :

De par les contraintes liées au Covid-19, j'exerce mon poste en distanciel. Initialement, le travail en distanciel était également prévu avec des jours en présentiel au sein de l'Incub 13 de l'Université Sorbonne Paris Nord. Ainsi je n'ai exercé mon poste que via le prisme du travail en distanciel. Une journée type au sein de CEOS TECH se constitue en trois actes, tout d'abord nous commençons le matin par une réunion des différentes équipes afin de faire un point sur les choses à réaliser pour la journée, survient donc ensuite la réalisation de nos tâches, généralement nous établissons une réunion avec tous les membres de l'équipe technique pour que l'on puisse se coordonner dans nos tâches et par la suite entamons nos missions, nous disposons également d'heure de pause pour déjeuner que nous plaçons selon nos convenances. Enfin, à la fin de la journée nous organisons un dernier point avec l'équipe au complet où nous mettons en avant nos avancées pour la journée, nous détaillons également les difficultés auxquelles nous avons pu faire face ou encore nos tâches inachevées.

En parlant de tâches, je vais désormais vous décrire les principaux projets auxquels j'ai pu participé chez CEOS TECH en précisant leur chronologie.

## Partie 3 : Chronologies & présentation des projets

### Présentation des projets :

À travers mes différentes périodes en entreprise au sein de CEOS TECH, je suis intervenu sur principalement trois de leurs projets. Le premier fut l'application mobile MARKUS dont j'ai participé au développement de la V0 et qui nous a occupés pendant environ 4 mois. Après finalisation de ce premier projet j'ai ensuite contribué à la réalisation du site internet du restaurant africain O'Lokoso, un projet qui nous a occupés pendant environ 2 mois. Enfin, j'ai participé au projet de réalisation de site internet pour le restaurant de type kebab Délice-DF5, un projet qui nous a notamment ouvert la porte à la participation à plusieurs autres projets dont ma participation fut ainsi plus ponctuelle, ce dernier projet nous a occupé pendant environ 3 mois.

### Frise chronologique des projets :



Maintenant que vous avez une idée plus claire de l'agencement chronologique de ces différents projets, je vais désormais axer la focalisation sur chacune d'entre elles en précisant leurs objectifs et enjeux ainsi que ma contribution concrète à travers l'exercice de mes différentes tâches et missions.

## Partie 4 : Missions et tâches

### Présentation de l'application MARKUS :

#### Objectifs du projet :

MARKUS est un logiciel SAAS<sup>(6)</sup> prenant la forme d'applications natives<sup>(1)</sup>, accessibles sur smartphones et tablettes et ayant pour destinataire les professionnels de la restauration. En effet, MARKUS a pour objectif de permettre aux restaurateurs le pilotage de leur activité au quotidien, la surveillance de leurs indicateurs clé de performance ainsi que la possibilité de réaliser des prévisions de chiffre d'affaires en temps réel grâce à l'appui du Big Data et du Machine Learning.

#### Raisons du projet :

Cette idée de projet d'application est venue de Laurent Goulenok, ayant exercé pendant de nombreuses années les métiers de chef cuisinier et de chef de partie dans plusieurs structures de restauration (hôtel restaurant, brasserie, restauration rapide), il a pu être témoin de la difficulté des restaurateurs à prévoir les quantités ou denrées à sortir pour chacun de leur service. Soit les restaurateurs préparaient trop de plats en avance, ce qui résulte d'un gaspillage alimentaire à la fin du service, soit les quantités préparées n'étaient pas assez conséquentes ce qui résultait d'un retard du service face à la forte affluence survenue.

De là, Laurent Goulenok fait le constat de trois problématiques principales. La première est que le gaspillage alimentaire représente une perte importante pour les restaurateurs, la seconde est le manque de valorisation des données dans le secteur de la restauration et la dernière concerne la problématique de la pluralité des applications dédiées à un seul domaine de gestion dont il faut souscrire pour chacun un abonnement spécifique.

#### Fonctionnalités de l'application :

Accompagné par Magaye Lopes, c'est ici que Laurent Goulenok en est venu à la solution MARKUS, une application qui centralise plusieurs domaines de gestion comme :

La gestion des stocks comprenant la réception des marchandises, la création d'un inventaire, un répertoire de fournisseurs où il est possible d'effectuer une commande directement depuis l'application. Ajoutons à cela la présence d'un historique des commandes passées et la possibilité de commander de façon automatisée les mêmes produits et quantités sélectionnées précédemment. Il y a la possibilité de créer ou de partager une fiche technique<sup>(5)</sup>, on retrouve également le répertoire des photos et étiquettes des produits enregistrés pour faire valoir leur traçabilité.

La gestion du personnel étant composée d'un registre du personnel où il est possible de créer et de modifier une fiche salariée ainsi que le répertoire des documents légaux (carte d'identité, carte vitale et contrat de travail pour chaque salarié). Il y a la possibilité de rédiger et d'éditer des contrats de travail (CDD et CDI, avenants), des lettres types (licenciement, convocation) et d'exécuter également des Déclaration préalable à l'embauche (DPAE) via l'aide de l'API<sup>(3)</sup> de l'URSSAF. On peut rajouter à cela les fonctionnalités de création d'un planning salarial avec un historique des plannings passés ainsi que la mise en place d'un émargement à distance des salariés via géolocalisation.

L'application présente comme expliqué plus tôt des fonctionnalités permettant l'analyse prévisionnelle des ventes orientées sur le chiffre d'affaires. Elle comportera diverses KPIs<sup>(7)</sup> pour visualiser l'évolution du chiffre d'affaires selon sa valeur en euros, le nombre de couverts et également sa répartition selon le service fourni (sur place, à emporter ou livraison), là aussi en euros et en nombre de couverts. Il sera possible d'observer le chiffre d'affaires selon le temps de la vente, c'est-à-dire selon la période de la journée (matin, midi, du soir) mais aussi selon le produit vendu, autrement dit la part d'un plat sur le chiffre d'affaires global.

MARKUS propose également des ratios concernant le ticket et le panier moyen qui permettent d'identifier la dépense moyenne de chaque client qui fréquente l'établissement alimentaire selon plusieurs paramètres comme la tranche horaire, la date, le mode de commercialisation et des produits retrouvés dans le ticket.

## Missions et tâches du projet MARKUS :

### État du projet et missions attribuées :

Lorsque j'ai pu être intégré au projet et confronté aux différents codes, j'ai pu remarquer que le code réalisé pour le front-end de l'application semblait plus avancé et complexe de ce qui avait été réalisé pour le back-end. On pouvait observer des différences au niveau des types de variables<sup>(8)</sup> envoyées à partir du front-end vers le serveur et des champs initialisés dans le back-end, des urls d'API dans le front-end qui ne correspondaient pas à celles inscrites dans le back-end. On pouvait trouver dans le back-end des tables ayant aucun champ ou encore dans le front-end retrouver des fonctionnalités qui sont liées à des données inscrites en dur sur un fichier. Ainsi remettre à niveau la structure du back-end avec celui du front-end semblait être une priorité pour l'avancée de l'application.

Cette mission principale s'est divisée en plusieurs tâches majeures faisant chacune référence à une fonctionnalité de l'application, ma première tâche concerne ainsi la gestion des stocks, plus exactement le fait d'activer les requêtes d'APIs pour la section inventaire.

## Première tâche majeure :

Pour réaliser cette tâche, je me suis dans un premier temps inspiré des bonnes fonctionnalités implémentées dans le code, c'était le cas par exemple de la fonctionnalité affichant le registre des fournisseurs où il était possible de créer, enregistrer et supprimer un fournisseur. En partant de là, j'ai étudié les éléments nécessaires au bon développement de la requête d'API. Il nous faut tout d'abord créer la table qui va contenir nos données, le framework Django génère un fichier du nom de *models.py* qui est dédié à l'implémentation de table lors de la création d'une nouvelle application.

### Exemple d'implémentation d'une table sur Django

```
102 class Fournisseur(models.Model):
103     nom = models.CharField(max_length = 45, null=True)
104     adresse = models.CharField(max_length = 45, null=True)
105     code_postale = models.CharField(max_length = 5, null=True)
106     ville = models.CharField(max_length = 45, null=True)
107     nom_contact = models.CharField(max_length = 45, null=True)
108     numero_telephone = models.CharField(max_length = 45, null=True)
109     numero_fax = models.CharField(max_length = 45, null=True)
110     adresse_mail = models.EmailField()
111     delai_livraison = models.IntegerField()
112     jour_livraison = models.CharField(max_length = 255, null=True)
```

Après conception de la table et de ces champs, il nous faut définir l'affichage des données dans le fichier généré par Django *views.py*. Dans ce fichier nous devons inscrire les fonctions qui vont être nécessaires aux requêtes HTTP de récupération (GET), d'ajout (POST) et de suppression (DELETE) de la donnée.

### Exemple d'implémentation d'une vue sur Django

```
12 class ReceptionMarchandiseResponse(generics.GenericAPIView):
13     """
14         Class-based view
15         handle http request comming from /reception_marchandise/ url
16     """
17     __produit_instance = ProduitEnStock()
18
19     def post(self, request, pk_restaurant):
20         try:
21             data = request.data
22             if data:
23                 if self.__produit_instance.create_produitEnStock(data, pk_restaurant):
24                     return create_HttpResponse({'success': True}, 201)
25                     return create_HttpResponse({'error':'Invalid payload'}, 406)
26
27                 return create_HttpResponse({'error':'No payload'}, 400)
28             except:
29                 print('--ReceptionMarchandiseResponse post')
30                 traceback.print_exc()
31             return create_HttpResponse({'error':'Server error'}, 500)
32
```

Enfin, il nous faut ajouter l'url qui donnera accès à la vue implémentée, ainsi dans le fichier *urls.py* il nous suffit de définir l'url ainsi que la vue concernée.

Exemple d'implémentations d'urls sur Django

```

7  urlpatterns = [
8      #path('inventaire/<int:pk_restaurant>', ReceptionMarchandiseResponse.as_view() ),
9      path('inventaire/<int:pk_restaurant>', InventaireResponse.as_view() ),
10     path('inventaire/<int:pk_restaurant>/<int:id_produitEnStock>', InventaireResponse.as_view() ),
11
12     path('inventaire/produitenstock/<int:id_produitEnStock>/delete/', InventaireResponse.as_view() ),
13     path('inventaire/produitenstock/<int:id_produitEnStock>/update/', InventaireResponse.as_view() ),
14     path('fichetechnique/<int:pk_restaurant>', FicheTechniqueResponse.as_view() ),
15     path('fournisseur/', FournisseurResponse.as_view() ),
16     path('fournisseur/<int:id_fournisseur>/delete/', FournisseurResponse.as_view() ),
17     path('fournisseur/<int:id_fournisseur>/update/', FournisseurResponse.as_view() ),
18 ]

```

À la suite de cela nous nous redirigeons vers le front-end où il nous faut définir les fonctions<sup>(9)</sup> donnant accès aux urls des différentes requêtes d'APIs, nous utilisons pour cela la méthode JavaScript *fetch()* avec comme paramètres l'url que l'on souhaite et également les spécificités concernant la requête à utiliser et le format d'affichage de la donnée reçue ou transmise, dans notre cas nous utilisons la donnée au format JSON.

De cette manière, j'ai pu mettre en place les requêtes d'APIs pour la section Inventaire de la partie STOCK. Néanmoins j'ai été confronté à une certaine problématique concernant la modification d'un élément de l'inventaire. En effet, avec la seule utilisation de Django, il semble difficile d'avoir une requête pour actualiser le contenu de la base de données car les requêtes prises en compte sont seulement la récupération (GET), la création (POST) et la suppression (DELETE) d'une donnée. Il nous fallait donc réfléchir à une sorte de requête POST qui écrase une donnée existante, un travail qui allait donc nous demander de la recherche et du temps car cela était encore inconnu pour nous.

Cependant, grâce à l'aide fournie d'un développeur back-end en stage du nom de Porus Hiruthayaraj et étant spécialisé dans l'utilisation de Django, nous avons pu faire la connaissance de Django Rest Framework. Ce framework à l'avantage de proposer des vues déjà configurées en termes de requêtes d'APIs autorisés, de cette manière nous ne sommes plus dans l'obligation de créer des fonctions pour chacune des requêtes utilisées, il nous suffit désormais de juste sélectionner les groupes de requêtes proposées par Django Rest Framework. Le framework implémentent notamment de nouvelles requêtes comme le PUT servant à la modification des données d'une table ce qui a éliminé notre problème. En ayant eu la connaissance de cet outil grâce à l'aide du stagiaire Porus Hiruthayaraj, j'ai pu continuer à avancer sur ma mission principale qui concerne l'établissement des requêtes APIs pour l'ensemble des fonctionnalités présentes sur l'application.

Je me suis ainsi dirigé vers la partie gestion du personnel avec les requêtes d'APIs concernant le registre du personnel à définir. Django Rest Framework m'a permis d'avancer bien plus vite sur la tâche qu'auparavant en plus de rendre toutes les requêtes fonctionnelles. De là, j'ai inséré la requête PUT pour les fonctionnalités présentes dans la partie gestion des stocks, c'est-à-dire pour la liste des inventaires ainsi que le registre des

fournisseurs. Il ne restait à ce niveau-là que la fonctionnalité concernant la liste des fiches techniques à implémenter, élément qui a constitué ma deuxième tâche majeure à l'avancée de la première version de l'application.

Exemple d'une implémentation optimisée avec Django Rest Framework

```
class FournisseurList(generics.ListCreateAPIView):
    queryset = Fournisseur.objects.all()
    serializer_class = FournisseurSerializer
    filter_backends = (filters.OrderingFilter, DjangoFilterBackend)

    def filter_queryset(self, queryset):
        queryset = super(FournisseurList, self).filter_queryset(queryset)
        return queryset.order_by('nom')

class FournisseurDetail(generics.RetrieveUpdateDestroyAPIView):
    queryset = Fournisseur.objects.all()
    serializer_class = FournisseurSerializer
```

Exemple d'une ancienne implémentation (sans Django Rest Framework)

```
class FournisseurResponse(generics.GenericAPIView):
    """
        Class-based view
        handle http request comming from /fournisseur/ url
    """

    __fournisseur_instance = Fournisseur()

    def get(self, request):
        """
            retrieve data from the Fournisseur table
            calls: readFournisseur
        """
        try:
            json_data = self.__fournisseur_instance.readFournisseur(pk=None)
            if json_data:
                return create_HttpResponse(json_data, 200)

            return create_HttpResponse({'error': "Server error"}, 500)
        except:
            print(" Erreur! FournisseurResponse.get")
            traceback.print_exc()
            return create_HttpResponse({'error': "Server error"}, 500)

    def post(self, request):
        """
            transmit data from the Fournisseur table
        """
        try:
            data = request.data
            if data:
```

```

        if self.__fournisseur_instance.createFournisseur(data):
            return create_HttpResponse({'success': True}, 201)
        return create_HttpResponse({'error':"Invalid payload"}, 406)

    return create_HttpResponse({'error':"No payload"}, 400)

except:
    print('--Fournisseur post')
    traceback.print_exc()
    return HttpResponse({'error': "Server error"}, 500)

def delete(self,request, id_fournisseur):
    fournisseur_objet = Fournisseur.objects.get(id=id_fournisseur)
    fournisseur_objet.delete()

    return HttpResponse(json.dumps({'message':"Delete Fournisseur avec succès"}), 200)

```

## Deuxième tâche majeure :

En effet, l'implémentation de cette fonctionnalité m'a pris au total plus de temps que pour les autres fonctionnalités de la partie gestion des stocks du fait de la problématique qu'il n'y avait aucune cohérence entre la table des fiches techniques présentes dans le back-end et celle définie au niveau du front-end, d'autant plus que certaines pages dans le front-end ne semblaient pas fonctionner.

Ainsi pour mener à bien cette tâche j'ai essayé de concilier les bons points du back-end et front-end tout en adaptant les éléments qui me semblaient mal réalisés. Mes modifications se sont plus orientées vers le front-end étant donné que le back-end suivait la logique d'un diagramme de classe qui lui était dédié et qu'il avait été également entièrement modifié pour convenir à la disposition que procure Django Rest Framework. J'ai donc modifié les différentes variables demandées lors de la création d'une fiche technique dans le front-end pour qu'ils maintiennent une certaine cohérence avec le back-end. J'ai également entretenu le fonctionnement des pages contenant la liste et le détail d'une fiche technique. Après tout cela, je suis revenu sur le back-end pour à la fin insérer l'ensemble des appels d'APIs.

Ma tâche était quasiment faite mais néanmoins un problème restait présent, et ce problème concernait les ingrédients de la fiche technique. En effet, le front-end à besoin de garder en mémoire l'ensemble des informations de l'ingrédient car il était prévu pour la suite, la version V1, d'implémenter avec ces informations dans différents calculs pour déterminer le coût matière de chaque ingrédient.

La première solution pour y remédier était d'utiliser dans le back-end le champ ingrédients comme le lien vers une autre table dédiée aux ingrédients. Ici le problème réside dans le fait que cette disposition nous donne accès seulement à l'id de l'ingrédient dans la fiche technique. Avec cette méthode il nous faudra prévoir une requête GET pour chaque ingrédient présent dans la fiche technique et sachant que lors du déploiement du back-end de l'application, le prix du déploiement va fluctuer selon le nombre de requêtes transmises, cette première solution ne semble pas très optimisée. D'autant plus qu'implémenter tout cela

reviendrait également à modifier le détail de l'ingrédient dans la fiche technique qui prendrait la forme d'une flatlist, une fonctionnalité qui prend un certain temps à être implémenté et qui aurait donc pour conséquence de retarder le projet.

La seconde solution plus simple et respectueuse des délais consiste à définir dans le back-end le champ ingrédients comme un champ texte mais par contre dans le front-end les ingrédients seront définis comme une liste. Ainsi pour convenir à ce que demande le back-end, il nous suffirait d'établir une fonction qui aurait pour rôle de convertir l'ensemble des ingrédients ainsi que leurs informations sous forme d'une chaîne de caractère. Cette solution néanmoins possède elle aussi un problème dans le sens où il nous serait impossible d'avoir accès aux informations de chaque ingrédient après sa création car après récupération du champ via un GET, la donnée transmise serait une chaîne de caractère que nous ne pourrons tout simplement pas exploiter.

Nous avons néanmoins fait le choix avec l'accord de nos supérieurs de s'en tenir pour l'instant à la deuxième solution en attendant de voir comment nous pouvons optimiser notre première solution qui nous semblait meilleure.

J'ai donc poursuivi la suite de mes tâches pour le déploiement de la première version de l'application MARKUS, il ne me restait qu'à établir les requêtes d'APIs pour la partie Mon Compte, élément qui va ainsi constituer ma dernière tâche majeure.

### Troisième tâche majeure :

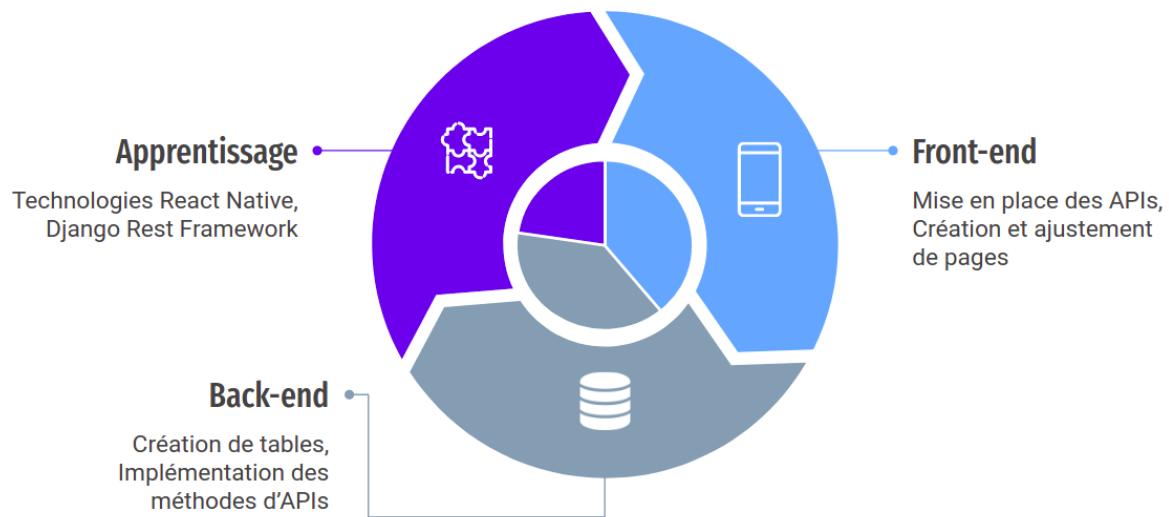
La partie Mon Compte avait pour objectif d'être composée de l'ensemble des informations concernant le profil de l'utilisateur, son entreprise ainsi que son établissement. Ces trois parties constituaient chacune une table à part entière dans les modèles du back-end. Ainsi il nous fallait organiser les requêtes d'APIs pour ces trois tables à chaque fois. Comme expliqué précédemment, il nous faut éviter de générer un trop grand nombre de requêtes d'APIs car c'est justement ce facteur qui risque d'augmenter le prix du déploiement. Dans cette idée, Porus Hiruthayaraj, Développeur Back-end en stage chez MARKUS à cet instant nous a encore une fois renseigné sur un moyen d'optimiser notre nombre de requêtes avec le modèle en nids.

Avec Django Rest Framework, le cheminement de la création des requêtes APIs semble avoir légèrement changé, en effet avec auparavant la seule l'utilisation de Django nous avions une disposition qui se constituait en la création des tables dans le fichier *models.py* dans un premier temps, puis dans un deuxième la configuration des différentes requêtes utilisées dans le fichier *views.py* et enfin l'implémentation de l'url donnant accès à la donnée dans le fichier *urls.py*. Désormais vient se rajouter le fichier *serializers.py* qui se place entre la création de la table et la configuration des requêtes d'APIs et qui va nous servir à définir l'affichage des champs de chaque table. De là il nous est possible par exemple de retirer l'affichage d'un champ de la table ou alors lorsque notre table possède un champ faisant référence à une autre table, et bien il nous est possible ainsi d'afficher également son contenu. Et avec cette méthode nous pouvons réduire à un le nombre de requêtes nécessaires pour avoir accès à l'ensemble des informations. À noter qu'il s'agit d'une

méthode qui pourrait fixer notre problème concernant les ingrédients de la fiche technique. Ainsi avec cette restructuration j'ai pu remplir ma dernière tâche majeure pour le projet d'application MARKUS.

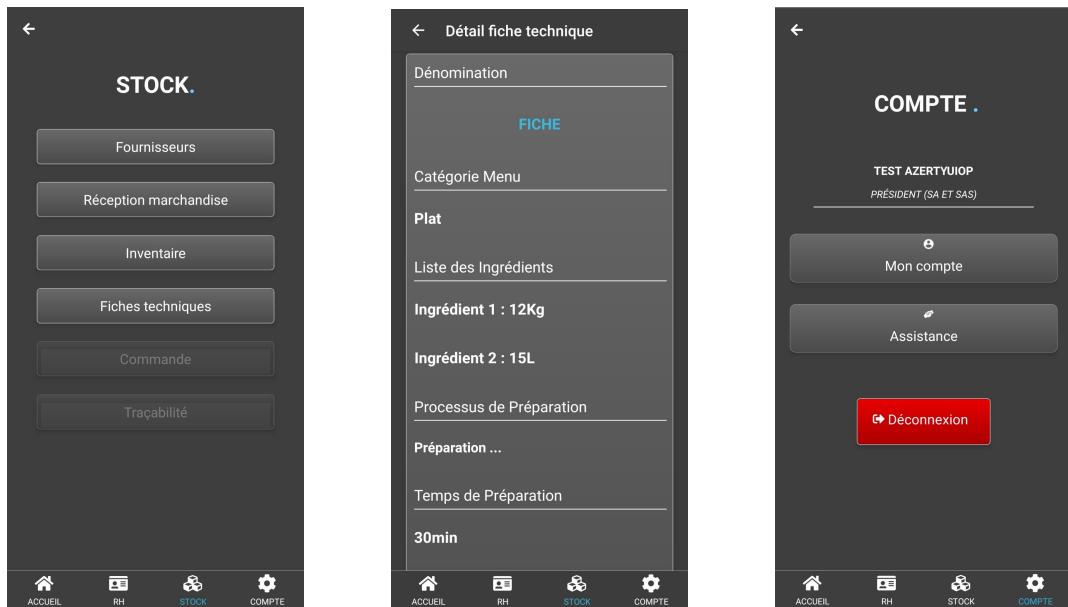
Le reste de mes missions après cela consistait à la finalisation de la partie Gestion du Personnel avec le pré affichage des données de l'entreprise lors de la rédaction de lettre type où de contrat de travail, également la gestion des stocks avec la mise en place de filtre soit par ordre alphabétique soit par ordre de création pour tous les registres.

### Schéma récapitulatif des missions du projet MARKUS :



### Captures d'écran du projet MARKUS :

Les captures d'écran montrent l'interface de l'application MARKUS. La première capture montre la page d'accès avec champs pour l'email et le mot de passe, et des boutons pour se connecter ou s'inscrire. Les deux autres captures montrent des écrans de gestion RH, avec des listes de fonctionnalités comme le registre du personnel, les contrats de travail, les lettres types, le planning, la déclaration préalable à l'embauche, les feuilles de présence et l'émargement. La troisième capture montre une page de convocation avec champs pour le nom du salarié, la dénomination sociale, l'adresse du siège social, le code postal du siège social, la ville du siège social, le lieu de la convocation, la date d'entretien et la date de signature.



## Présentation du site internet O'Lokoso :

### Objectifs du projet :

Le site internet O'Lokoso a pour responsabilité d'être la vitrine du restaurant, elle a pour rôle d'illustrer les produits phares du moment, de présenter les membres en cuisine, leurs valeurs ainsi que bien évidemment l'ensemble des informations concernant le restaurant. Le site est divisé en deux parties principales, une zone cliente ainsi qu'une zone administrateur réservée aux membres du restaurant.

Pour la partie cliente, le site internet O'Lokoso offre la possibilité de prendre une commande directement au sein de son site internet, d'avoir accès à l'ensemble des produits disponibles de la carte et enfin de pouvoir contacter le restaurateur à travers un formulaire de contact.

Pour la partie administrateur, il est possible de moduler la disponibilité du restaurant et des différents produits de la carte, d'accéder aux fichiers contacts des clients, d'observer en temps réel les commandes clientes prises sur le site et enfin d'avoir la possibilité d'organiser un suivi des commandes réalisées de sa prise en compte jusqu'à la remise au client.

## Raisons du projet :

Ce projet intervient suite à une première tentative du restaurateur de O'Lokoso à réaliser son propre site internet. Cette première tentative fut peu concluante pour le restaurateur étant donné que l'organisme embauché à la réalisation de ce premier projet a fourni une production inachevée et non-fonctionnelle au client restaurateur, ainsi le projet fut laissé à l'abandon par celui-ci jusqu'à maintenant. C'est à partir d'ici que CEOS TECH a proposé au client restaurateur de reprendre en main son projet en partant de zéro tout en incluant les nouveaux objectifs mentionnés ci-dessus.

## Fonctionnalités du site internet :

Comme vu plus tôt, le site internet de O'Lokoso se divise en deux parties, une partie se référant à la zone cliente du site ainsi qu'une autre se référant à la zone administrateur. Ces deux parties comportent différentes fonctionnalités disposées de la façon suivantes :

Pour la partie cliente, celle-ci est composée dans un premier temps d'une page principale construite façon one page<sup>(10)</sup> et comportant plusieurs éléments tels qu'un carrousel des produits phares du moment, une section de présentation de l'équipe en cuisine ainsi que des valeurs du restaurant, une zone d'informations concernant le restaurant, un formulaire pour prestation traiteur ou bien une réservation brunch.

Dans un deuxième temps, on retrouve la page de commande d'un produit composée de l'ensemble de la carte et des produits disponibles du restaurant. Tous les produits y sont répertoriés selon leur catégorie de produits (entrées, plats, desserts, boissons, etc...), elles même agencées dans un menu navigable. On retrouve également la possibilité de composer son propre menu à partir des produits disponibles du restaurant.

Nous avons ensuite la page du panier composée d'un récapitulatif des commandes prises sur le site. Il est possible à partir de cette page de définir la méthode de livraison (à emporter ou en livraison), de confirmer le contenu du panier et de procéder au paiement par carte bancaire. Enfin, il nous reste la page de galerie qui répertorie les photos en provenance du compte instagram de O'Lokoso.

Pour la zone administrateur réservée au restaurateur et à ses membres, l'accès au dashboard s'effectue après édition d'un url spécifique et inscription d'un login d'accès créé et fourni par CEOS TECH.

Après que cela a bien été effectué, l'utilisateur dispose de l'accès à la zone administrateur qui est constitué dans un premier temps des onglets nouvelles commandes, commandes en cours et historique des commandes permettant l'acquisition et le suivi des commandes reçus sur le site. Concrètement, l'onglet nouvelle commande sert à signifier l'administrateur de la réception d'une nouvelle commande, celle-ci génère un bip sonore qui ne s'arrête qu'après confirmation de la prise en compte de la commande. L'onglet commande en cours

quant à lui répertorie les commandes confirmées qui sont donc à réaliser et à fournir au client. Enfin, l'onglet historique des commandes affiche l'ensemble des commandes complétées.

Dans un deuxième temps vient l'onglet suivi de l'activité mettant en avant le chiffre d'affaires généré pour ce jour avec également la possibilité de consulter les jours précédents.

Nous avons ensuite l'onglet disponibilité des plats permettant de configurer la présence ou non des différents produits de la carte, cette configuration impacte l'affichage de la partie cliente car il ne sera pas possible de commander un produit configuré comme non disponible dans la zone administrateur. À noter qu'il est également possible de configurer la disponibilité du restaurant sur le même principe que l'onglet disponibilité des plats, cette optionalité se révèle directement au niveau du menu où figure l'ensemble des onglets.

Enfin, nous avons l'onglet fichiers contacts offrant une visibilité aux messages des clients issus du formulaire de contact. Ainsi sont les fonctionnalités imposées par ce premier projet de site internet, je vais donc par la suite vous préciser l'ensemble de mes missions.

## Missions et tâches du projet O'Lokoso :

### État du projet et missions attribuées :

Nous avons débuté le projet sur la base de code établi d'un ancien stagiaire qui avait traité le cas d'un site internet pour un restaurant virtuel. Au niveau du front-end, les fonctionnalités liées à la partie cliente et administrateur du site internet étaient quasiment toutes initialisées. Côté back-end, l'ensemble des tables essentielles au projet avait été construites sur une base de données MySQL n'utilisant pas Django REST Framework.

Ainsi en partant de cette base de code ma ligne directrice au sein de ce projet fut l'optimisation et la finalisation de la partie back-end ainsi que la finalisation des fonctionnalités et du design de la partie administrateur du site internet O'Lokoso.

Cette mission principale s'est organisée autour de trois tâches majeures, la première consiste en la finalisation du back-end du site O'Lokoso sous Django Rest Framework afin de pouvoir par la suite remplir l'ensemble des données de la carte du restaurant.

### Première tâche majeure :

Pour réaliser cette première tâche, il a fallu dans un premier temps repérer les éléments à garder, à modifier ou à enlever au sein de la production de l'ancien stagiaire côté back-end. En effet, vu que ce back-end a été réalisé à l'origine pour un autre restaurant, nous aurons forcément des tables ou requêtes d'API à enlever ou à modifier. Après avoir effectué les modifications nécessaires au projet O'Lokoso, je me suis concentré sur les tables allant servir à l'affichage des produits de la carte du restaurant. Étant en coopération lors de ce

projet avec la stagiaire Anne Bequet ayant une appétence significative au niveau du front-end et de la technologie ReactJS, nous nous sommes donc mis d'accord sur les différentes tables et champs à initialiser.

Ainsi pour la partie impliquant la carte du restaurant nous avons utilisé deux tables, une première qui servira à la définition d'une catégorie de produit (entrée, plats, desserts, etc) et une seconde nécessaire à la définition d'un produit.

La table catégorie est composée de deux champs outre sa clé qui est un numéro unique d'identification, on retrouve le nom de la catégorie de produit ainsi que sa description. Pour la table produit nous avons défini trois catégories de champs. Nous avons les champs communs pour l'ensemble des produits du restaurant, cela englobe le nom du produit et sa description, la catégorie du produit qui est une clé étrangère faisant référence à la table produit, l'image du produit et son prix ainsi que sa disponibilité étant un booléen (vrai ou faux).

Nous avons ensuite les champs spécifiques constitués des booléens suivants, *sur\_grill*, *au\_menu*, *accompagnement*, *supplement*. L'ensemble de ces champs permettent d'indiquer la caractérisation du produit, par exemple un produit ayant la valeur *true* sur le champ *supplement* veut signifier que le produit en question se caractérise comme un supplément, il s'agit donc d'un produit particulier. Et nous avons enfin un champ spécifique aux suppléments qui n'est plus d'autre que le prix du supplément à entrer seulement si le produit en question est un supplément.

Après initialisation des tables sous Django Rest Framework, il nous fallait remplir les données de la carte, pour cela nous devions tout d'abord déployer le notre back-end sur un serveur distant. Nous avons utilisé à cet effet la plateforme Heroku permettant notamment le déploiement de sites internet développés en langage python. Pour mettre en oeuvre le déploiement, une des solutions est de créer un répertoire Github qui sera liée à notre projet Heroku, il faudra rajouter à ce répertoire quelques dépendances et fichier de configuration supplémentaire tel que le fichier *requirements.txt* ou le fichier *Procfile* afin de parfaire le déroulement du déploiement. Par la suite à l'aide du CLI<sup>(11)</sup> de Heroku nous organisons le déploiement du répertoire Github que nous avons créé vers le répertoire Github de la production déployée, générée par la plateforme Heroku. Une fois que le déploiement bien réalisé, Heroku nous fourni l'url de production de notre projet et ainsi grâce à Django Rest Framework, il nous sera possible selon l'autorisation imposée de remplir nos tables initialisés par les données du restaurant O'Lokoso, élément ayant constitué la dernière étape de ma tâche majeure.

Après remplissage de l'ensemble des produits de la carte, il nous fallait réadapter les informations récupérées et transmises de la commande du client vers la partie administrateur, ce qui forme ainsi la composition de ma deuxième tâche majeure.

## Deuxième tâche majeure :

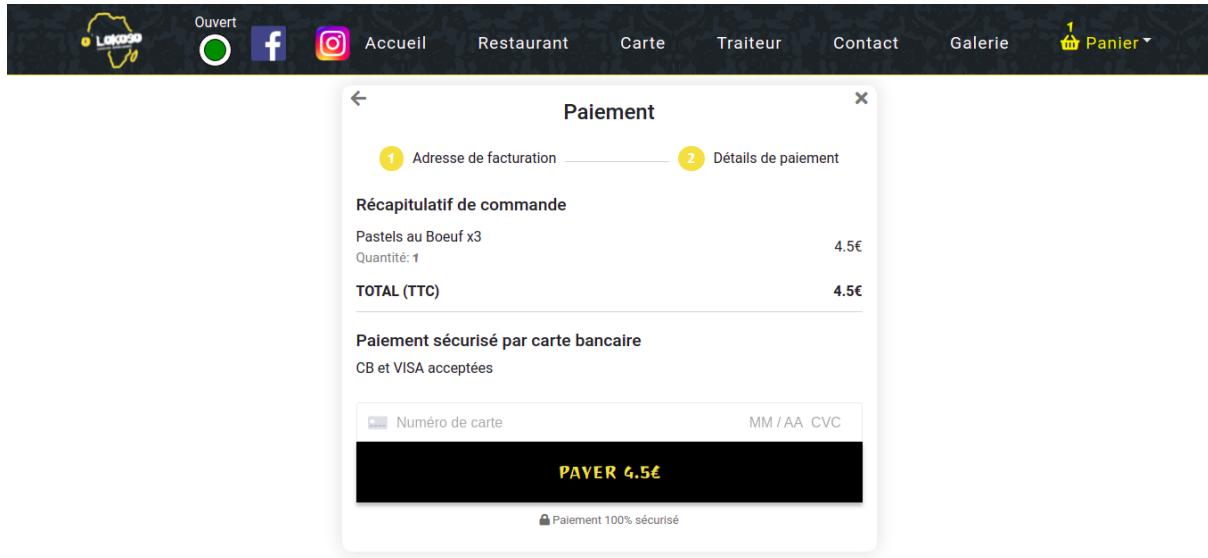
L'objectif pour cette tâche est de remanier la récupération des informations sélectionnées par le client. En effet, selon la catégorie de produit choisie par le client sur le menu de la carte, les informations à relever seront différentes. Certains produits demandent la sélection obligatoire d'un accompagnement au choix, d'autres donnent la possibilité de choisir un supplément et sans oublier la présence des menus composés de plusieurs produits de catégorie produit différentes. Pour tous ces choix de sélection nous devons être en mesure de récolter et d'également stocker les bonnes informations. Pour effectuer cela, plusieurs solutions s'offrent à nous, une première solution est d'utiliser une variable globale<sup>(12)</sup> même s'il semble peu recommandé d'en avoir une pouvant changer de valeur au cours de l'exécution du code.

Une autre solution que nous avons retenue à cette problématique est l'utilisation de reducers, un outil fourni par la librairie Redux de React. Un reducer est une fonction qui à l'aide d'un state (variable d'état) et d'une action (ou fonction associée) donnés en paramètre crée un nouveau state ayant pris en compte l'action fourni. Dans notre cas, nous avons utilisé la fonction *createSlice()* proposé par Redux qui prend comme paramètre un nom, un state initiale ainsi qu'un slice qui n'est d'autre qu'un objet composé de plusieurs reducers. Le slice a pour avantage d'être importable dans n'importe quelle classe et permet, si l'on transfère son contenu dans une constante ou une variable, d'avoir un accès centralisé des informations de la commande du client ainsi qu'à l'ensemble de nos reducers. C'est grâce à cette structure que nous sommes en mesure de fournir un récapitulatif des commandes prises par l'utilisateur au niveau de la page de panier.

Une fois la structure de récupération des informations de commandes mise en place, nous devions revoir l'agencement de sa transmission vers notre serveur distant. Ce processus se déroule lors de la confirmation du paiement de la commande, un processus géré par la librairie Stripe. Stripe est une bibliothèque de React qui propose différents types de formulaire de paiement directement agencable sur notre site internet, elle offre d'autant plus une certaine liberté de customisation afin d'avoir un formulaire plus original ou fidèle à la charte graphique de notre site. Stripe gère également la vérification ainsi que la sécurisation des différents paiements, elle propose notamment un dashboard afin de constater l'ensemble des paiements qui ont eu lieu.

Au niveau du code, l'agencement du formulaire se présente par un composant importé de la librairie Stripe auquel nous définissons une fonction pour récupérer les informations de paiement. Nous créons par la suite un bouton de confirmation qui exécutera lors de son clique la fonction *confirmCardPayment()* issue de Stripe et permettant d'établir la conformité du paiement. Ainsi dans le cas où le paiement est bien conforme nous transmettons les détails de la commande vers le serveur distant via une requête POST.

Exemple de la page de finalisation du paiement



Cette partie de réalisée, il ne me restait désormais qu'à adapter l'affichage des commandes sur la partie administrateur selon les informations transmises depuis le paiement et d'également finaliser l'ensemble des fonctionnalités de la partie administrateur, élément constituant ma dernière tâche au sein du projet O'Lokoso.

### Troisième tâche majeure :

Concernant la partie administrateur, la première chose à réaliser était le système de connexion à la plateforme. Pour ce faire, Django Rest Framework propose un module d'authentification afin de faciliter son implémentation au niveau du back-end. En effet, le module `JWTAuthentication` issu du paquet `rest_framework_simplejwt` permet l'exécution des requêtes d'API après une vérification d'authentification via un token, un élément qui a pour conséquence de donner accès aux données des commandes à un utilisateur s'étant au préalable connecté. Couplé au module `TokenObtainPairView` et `TokenRefreshView` permettant respectivement la génération d'un token valide et d'un token de rafraîchissement pour un utilisateur ayant fourni de bons identifiants pour l'un ainsi que pour la transformation d'un token de rafraîchissement en un token valide, dans le cas où le token valide initial a été expiré pour l'autre. Concrètement, l'authentification s'illustre par une requête POST ayant pour paramètre un JSON des identifiants fourni par l'utilisateur, si ces identifiants sont valides le serveur back-end nous enverra alors les deux tokens nécessaires à nos prochaines requêtes.

Cela effectué je suis intervenu par la suite au niveau des onglets de commandes de la partie administrateur. Les onglets Nouvelles commandes, Commandes en cours et Historique des commandes fonctionnent pour les trois d'une requête GET issue d'une table Commande. Néanmoins chaque onglet utilise un filtre différent qui convient à l'état de suivi de la commande (a été vue, est en cours ou a été livrée). Pour ces onglets, je suis intervenu

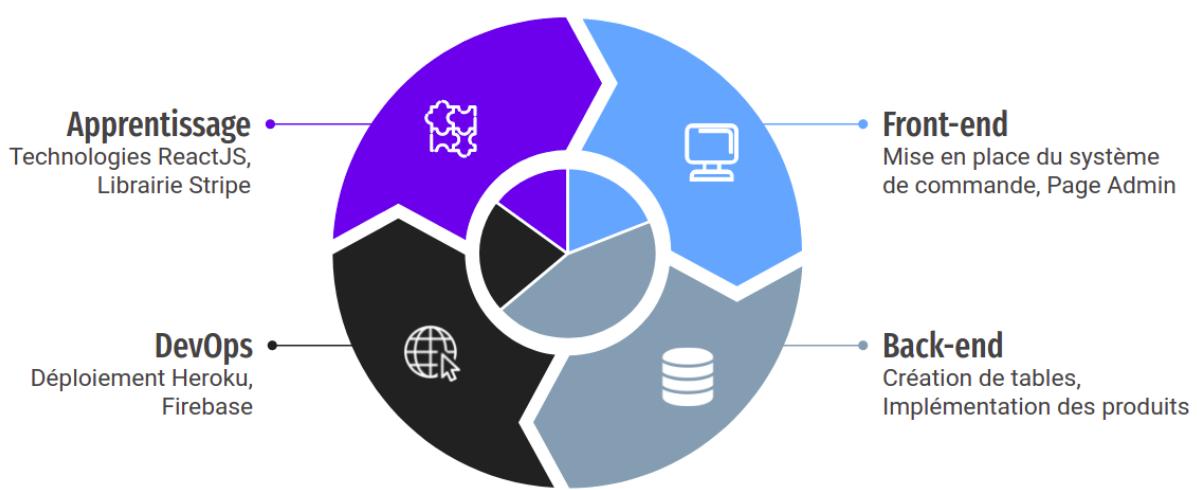
principalement sur la restructuration du design liées aux nouvelles informations des commandes que nous avions modifiées précédemment au niveau de la partie cliente.

Nous en venons ensuite à l'onglet Suivi de l'activité, cet onglet fonctionne d'une requête POST ayant pour paramètre une date de début et de fin au format d'un entier relatif. La valeur numérique de l'entier relatif correspond au nombre de millisecondes écoulées depuis le premier janvier 1970 à minuit d'après le temps universel (UTC). Lorsque le serveur récupère les deux intervalles de date, il va effectuer une somme de l'ensemble des transactions qui ont eu lieu entre les deux intervalles pour enfin envoyer le montant de la somme après exécution de la requête. Ainsi avec cette méthode, nous pouvons récupérer le chiffre d'affaires sur une journée, un mois ou même une année, néanmoins ici nous ne prenons en compte que la date du jour ainsi qu'une date en particulier sélectionnée par l'utilisateur. Pour cet onglet, je suis intervenu sur un problème de conversion des montants de transaction ne s'affichant pas en valeur euros.

Après cela vient l'onglet Disponibilité des plats qui mobilise principalement deux requêtes, la requête GET pour obtenir l'état courant de la disponibilité du produit ainsi que la requête PUT pour modifier la disponibilité d'un produit en particulier. Concernant le design nous utilisons un composant de boutons switch proposé par la librairie Material UI et qui possède l'avantage d'être réactif (ou responsive en prenant le terme anglais). Ainsi cet onglet ayant bien été initialisé il ne m'a demandé aucune modification de ma part.

Enfin nous avons l'onglet Fichier contact utilisant une requête GET pour afficher les messages des clients issus du formulaire de contact. Cet onglet n'ayant pas été implémenté, je me suis donc chargé de sa réalisation.

Schéma récapitulatif des missions du projet O'Lokoso :



Captures d'écrans du projet O'Lokoso :

O'Lokoso est une **ode gustative à l'Afrique de l'Ouest**. Nous vous proposons des plats issus de la culture Africaine avec une touche qui fait toute la **différence !** 🔥🔥🔥

**Commander**

Du lundi au vendredi entre 11h et 14h

**Menu entrée + plat + boisson**

**14€**

**Menu plat + boisson**

Date	Heure	Client	Méthode de vente	Montant TTC
20/08/2021	19h59	A.Josué	livraison	22€
19/08/2021	20h34	D.Jeremy	livraison	27.7€
18/08/2021	11h24	O.Danielle	livraison	14€
13/08/2021	19h15	D.Kaola	livraison	21€
04/08/2021	19h10	B.Dorine	livraison	21€
29/07/2021	20h40	m.morgane	livraison	33€

## Présentation du site internet Délice-DF5 :

### Objectif projet :

Le site internet pour le restaurant de type kebab Délice-DF5 dispose d'objectifs similaires à celui du site internet O'Lokoso. On retrouve pour la partie cliente une présentation de l'ambiance culinaire du restaurant, l'illustration des produits phares du moment, les informations concernant le restaurant, sans oublier la présence de la carte du restaurant avec la possibilité de prendre commande directement sur le site.

Pour la partie administrateur, on reste sur la possibilité de pouvoir moduler la disponibilité du restaurant et des produits de la carte, d'avoir un accès aux commandes clientes et de pouvoir assurer leur suivi. S'ajoute à cela la possibilité d'impression d'un ticket de caisse à partir des informations relevées de la commande clientes.

### Raison du projet :

Ce projet est né de la volonté du gérant du Délice-DF5 souhaitant élargir la visibilité de ces deux restaurants pour pouvoir à terme prétendre à la voie d'une franchise. CEOS TECH est ainsi intervenu pour accompagner le gérant dans cet objectif en lui proposant la réalisation de son site internet ainsi qu'une aide à l'entretien des différents réseaux sociaux de l'enseigne.

### Fonctionnalités projet :

Ainsi pour les fonctionnalités concernant le site internet du Délice-DF5 nous avons :

Pour la partie client, une première page de présentation du restaurant et de ses produits selon une structure One Page. Cette page contient un diaporama de l'accueil du restaurant, un carrousel présentant les produits phares et valeurs du restaurant ainsi qu'une zone d'informations concernant le restaurant.

On retrouve la page de commande contenant les produits de la carte, la page panier qui enregistre les commandes retenues par le client et permet leur finalisation grâce au choix de la méthode de livraison ainsi qu'à la procédure de paiement. Enfin nous avons la page galerie photos du restaurant utilisant les photos issues du compte Instagram du Délice-DF5.

Pour la partie administrateur, nous avons tout comme le site internet O'Lokoso une page de login qui après connexion donne accès aux onglets Nouvelles commandes, Commandes en cours, Historique des commandes, Suivi de l'activité, Disponibilité des plats ainsi que la disponibilité du restaurant directement accessible sur le menu. S'ajoute à cela l'onglet Disponibilité de livraison auquel sa valeur aura un impact au niveau du panier de la partie client qui affichera ou non la possibilité de prendre une commande en livraison.

## Missions et tâches du projet Délice-DF5 :

### État du projet et missions attribuées :

Pour ce nouveau projet, nous sommes partis sur la base de code du projet O'Lokoso, un choix effectué de par la similarité du site internet à celui de O'Lokoso en termes de fonctionnalités et de design. L'avantage de ce choix est qu'il nous dispense d'une relecture de code complète étant donné que nous-même étions en charge du développement du projet O'Lokoso et que le code nous reste ainsi assez familier.

Au niveau de mes missions, celles-ci ont été plus variées que les précédentes. En effet, mes missions se sont organisées en deux axes principaux, un premier axe qui concerne les modifications à apporter au site internet Délice-DF5 afin que celui-ci convienne aux fonctionnalités imposées, puis un deuxième axe faisant intervenir de la maintenance et de l'optimisation de code pour différents projets ainsi que de l'animation de formation sur les technologies Django Rest Framework et ReactJS pour les nouveaux arrivants développeurs sur le projet.

J'ai donc organisé la présentation de ces différentes missions en deux tâches majeures, dont la première relate des modifications effectuées et fonctionnalités ajoutées à la partie cliente du site internet Délice-DF5.

### Première tâche majeure :

Comme pour O'Lokoso, l'utilisation d'un autre projet comme base de code va demander de réaliser certaines modifications pour adapter les fonctionnalités selon le nouveau site internet. Ainsi j'aurais pour une nouvelle fois l'action de modifier les tables impliquées issues du back-end dans l'utilisation de la page cliente de commande côté front-end. Dans le cas du restaurant Délice-DF5, celui-ci propose une large gamme de produits et de catégories de produits pour sa carte. Pour établir cette même cohérence, il m'a fallu donc rajouter plus de tables et plus de champs pour les données.

Nous avons toujours les tables Produit et Catégorie produit cependant j'ai pu implémenter de nouveaux champs spécifiques côté table produit, des champs de type booléen permettant de distinguer l'appartenance à une certaine catégorie de produits. Cette distinction est nécessaire car elle va permettre de définir au niveau du front-end un certain choix d'interactions pour chaque produit. Concrètement, si j'attribue à un produit le champ *est\_milkshake* à *true*, lorsque ce produit aura été sélectionné par le client dans la page de commande, celui-ci aura droit un choix de suppléments ou d'ingrédients spécifiques à sa catégorie de produit, soit ici la catégorie des milkshakes. Ainsi je définis un champ spécifique pour chaque catégorie de produits existants. J'ai ajouté à cela la table ingrédient, une table composée des champs *nom* et *type\_ingredient* (un pain, une garniture, une viande, une sauce, etc) dont les données constituent les différents choix de customisation du produit sélectionné par le client. Nous avons également la table Supplément, une table

assez similaire à celle des ingrédients au niveau des champs utilisés mais qui se distingue par son champ *prix* permettant d'indiquer le prix du supplément lorsqu'on le sélectionne en complément de son produit ainsi que les champs spécifiques nécessaires au regroupement des différents types de supplément et qui seront affichés pour chaque produit selon leur catégorie de produit respective comme vu précédemment.

Exemple d'affichage des choix de suppléments pour un produit

Supplement(s)	Prix
<input type="checkbox"/> Nutella	(0.80€)
<input checked="" type="checkbox"/> Milka	(0.80€)
<input type="checkbox"/> Sneakers	(0.80€)
<input checked="" type="checkbox"/> KitKat	(0.80€)
<input type="checkbox"/> Oreo	(0.80€)
<input type="checkbox"/> Daim	(0.80€)
<input type="checkbox"/> Lotus Spéculos	(0.80€)
<input checked="" type="checkbox"/> M&M's	(0.80€)
<input type="checkbox"/> Twix	(0.80€)
<input type="checkbox"/> Kinder Bueno	(0.80€)

Ainsi après avoir adapté le back-end et le front-end avec la page de commande sans oublier d'ajuster les pages indirectement concernées comme la page panier ainsi que les onglets de commandes de la partie administrateur, je me suis concentré sur l'implémentation de l'onglet Disponibilité livraison ainsi que de la sélection de la zone de livraison au niveau du panier. L'onglet Disponibilité livraison fonctionne de la même manière que pour la Disponibilité des plats et la disponibilité du restaurant, elle mobilise donc les requêtes GET et PUT lors de son utilisation et elle possède un onglet qui lui est dédié au niveau de la partie administrateur. Cette fonctionnalité a pour impact de modifier la possibilité pour le client de sélectionner le choix "en livraison" lors de la finalisation de sa commande au niveau de la page de panier.

Lorsque la livraison est activée et que le client sélectionne ce choix dans la page panier, la fonctionnalité de sélection de la zone de livraison survient. Cette fonctionnalité sert à définir un montant minimum de commande pour certaines localisations de livraison, cela s'illustre par plusieurs propositions de localisation englobant différentes villes. L'utilisateur devra sélectionner la zone où sa ville est comprise et devra en conséquence respecter le minimum de commande imposé pour cette zone. Au niveau de l'implémentation dans le code, lorsqu'une zone aura été sélectionnée par l'utilisateur et que celui-ci enclenchera la procédure de paiement, une condition sera chargée de vérifier si le montant du panier du client est supérieur ou égale à la zone de livraison sélectionnée.

Exemple de l'affichage de la zone de livraison

The screenshot shows a website header with a logo, a 'Fermé' button, and navigation links for Accueil, Restaurant, Carte, Galerie, and Panier. The main content area has a title 'Minimum de commande selon la zone' and a subtitle 'Délai livraison : 45min/1heure'. It prompts the user to 'Veuillez choisir votre zone de livraison'. Below this, there are three radio buttons for delivery zones:

- Zone 1 (20€ minimum de commande) : Le Blanc-Mesnil, Aulnay-sous-Bois, Le Bourget
- Zone 2 (25€ minimum de commande) : Bondy, Bobigny, Drancy
- Zone 3 (30€ minimum de commande) : Stains, Pierrefitte-sur-Seine, Saint-Denis, Villetteuse, Sarcelles

Below the zones is a checkbox labeled 'J'ai lu et j'accepte les [Conditions Générales de Vente](#)'. At the bottom is a large black 'COMMANDER' button.

Ainsi ont été les missions que j'ai effectuées au sein du projet Délice-DF5, néanmoins ma participation au projet ne s'est pas arrêtée à cela. J'ai pu en effet entreprendre des aspects de maintenance et d'amélioration pour le site internet Délice-DF5 ainsi que pour l'ancien projet du site internet O'Lokoso, un élément constituant ma seconde tâche majeure.

Deuxième tâche majeure :

Le premier axe d'amélioration des sites internet s'est effectué avec l'implémentation du design MARKUS pour la partie admin du site internet O'Lokoso. CEOS TECH souhaitant externaliser l'offre de la partie administrateur par rapport à la production de sites internet a décidé d'y incorporer la solution MARKUS pour cette partie. Ainsi ce nouveau dashboard aura en plus des fonctionnalités déjà implémentés, ceux liés à l'application MARKUS comme la Gestion des stock et la Gestion du personnel. Après adaptation de la partie administrateur, nous en avons profité pour implémenter tout comme le Délice-DF5 la zone de livraison dans la finalisation de la commande.

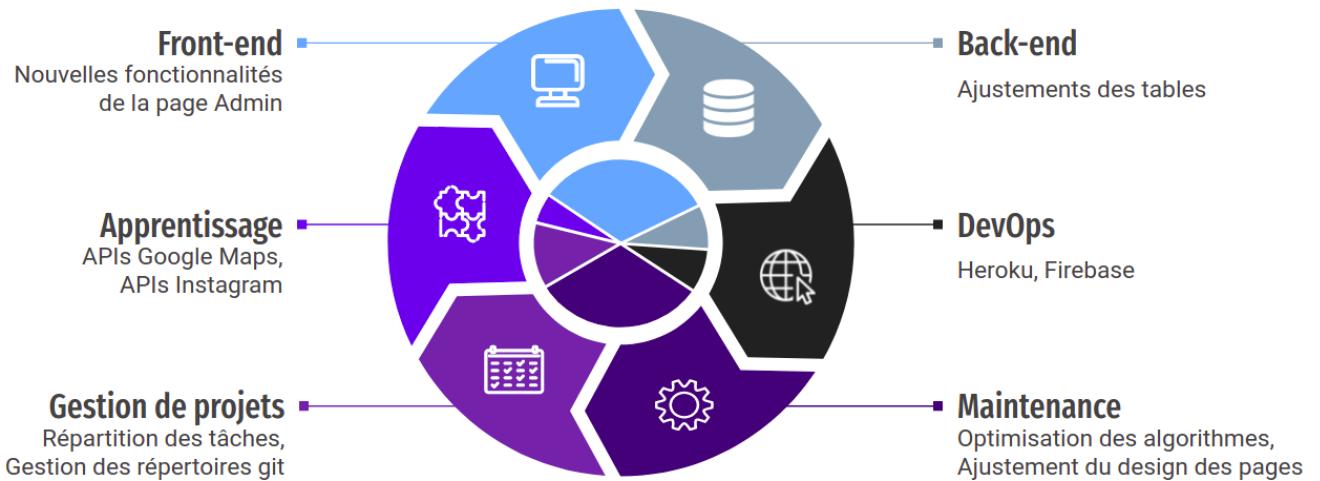
Un autre axe d'amélioration fut l'optimisation de code du choix de commande pour le site Délice-DF5. En effet, les nouveaux membres intégrés au projet ont rencontré des difficultés à pouvoir effectuer des modifications au niveau de cette partie du fait du manque de factorisation de code. L'affichage dédié à chaque catégorie de produit était répertorié dans un fichier en particulier, néanmoins on pouvait retrouver beaucoup de similarité entre ces différents fichiers, ainsi une centralisation de ces informations en un seul fichier était nécessaire. J'ai continué à maintenir et à améliorer d'autres éléments plus mineurs du site internet Délice-DF5 comme l'orthographe, la disposition des éléments du site ou l'agencement des routes.

---

Une part importante à laquelle j'ai participé avec le projet Délice-DF5 fut la formation au code des nouveaux membres ainsi que la gestion du projet associée. En ce qui concerne les formations de code que j'ai animées, celles-ci m'ont permis de consolider mes connaissances au niveau du code et des langages de programmation utilisés. Le fait d'animer une formation code demande de connaître et de pouvoir justifier chaque ligne de code présente dans le projet, il demande aussi d'être capable de tirer son attention sur les fonctionnalités clé du projet et de ne pas s'éparpiller dans des détails parfois inutiles, de garder un œil sur l'état de compréhension de chaque participant en leur posant des questions et enfin d'alterner la vision du code avec celle du résultat associé afin que les participants constatent concrètement de l'utilité du code réalisé.

En ce qui concerne la gestion de projet du Délice-DF5, j'ai pu mener à bien l'attribution des tâches des différents membres sur le projet, des tâches que j'ai essayé d'attribuer selon l'aisance au code de chaque membre et également selon l'importance des tâches en question. En fonction de l'avancée du projet et des difficultés rencontrées, j'adapte les missions et tâches des membres concernés afin que cela ne ralentisse pas trop l'avancée du projet. Il s'agit d'une partie de mes missions que j'ai apprécié expérimenter et qui m'ont permis de développer une vision plus globale du projet.

Schéma récapitulatif des missions du projet Délice-DF5 :



Captures d'écrans du projet O'Lokoso :



Ouvert

Accueil      Restaurant      Carte      Galerie      Panier ▾



Menus sandwichs  
Menu burger  
 Menus tacos  
 Menus sandwichs au four  
 Menus paninis  
 Menus croque  
 Menus crêpes salées



Buffalo  
8€90



Cheese  
4€00



Chicken



Classic

Ouvert

Accueil      Restaurant      Carte      Galerie      Panier ▾

### Détail du panier

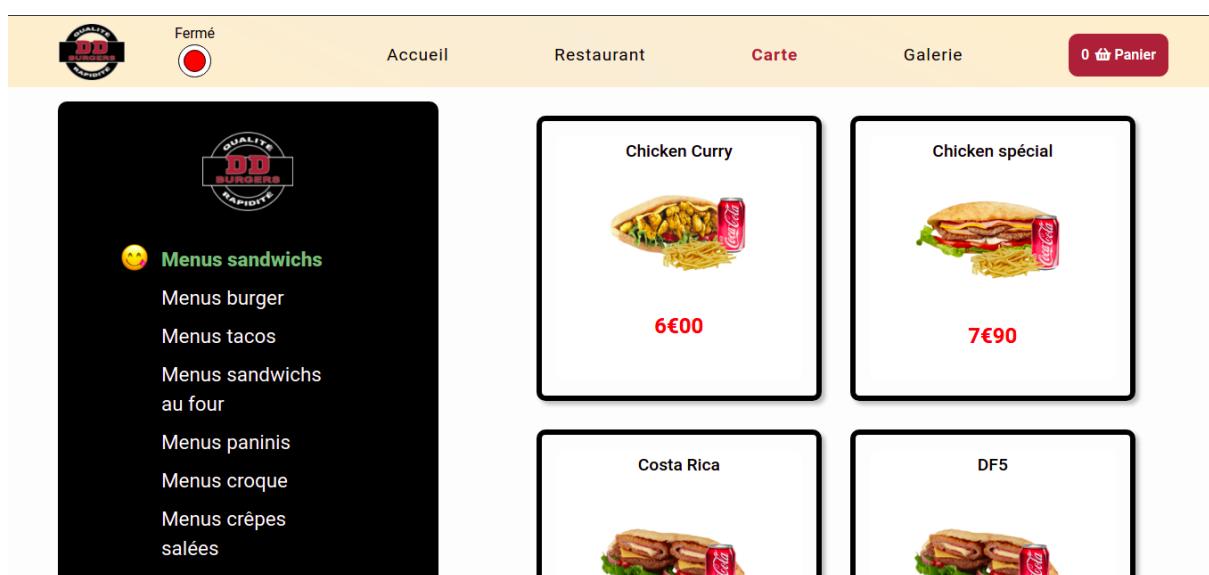
1 article

1x Chicken <small>Garniture Oignons Tomates Sauce Ketchup Biggy burger Supplément 0.5€ Oeuf 0.5€ Boisson Coca-cola Cherry</small>	<span style="border: 1px solid #ccc; padding: 2px;">-</span> <span style="border: 1px solid #ccc; padding: 2px;">1</span> <span style="border: 1px solid #ccc; padding: 2px;">+</span>	<b>6€00</b>	<span style="border: 1px solid #ccc; padding: 2px;">■</span>
		<b>Prix Total (TTC)</b>	<b>6 €</b>

## Mes autres projets :

Mes missions ne se sont pas limitées aux trois projets que je vous ai présentés, j'ai pu participer ponctuellement à des nombreux projets de sites internet, des projets dont je compte vous fournir une présentation synthétique.

Le premier projet auquel j'ai participé ponctuellement durant l'exercice de mes missions est le projet de site internet du Dima-Déllice, un restaurant de type kebab très similaire au restaurant Délice-DF5. Je suis intervenu lors de ce projet pour aider les stagiaires assignés à ce projet à l'agencement de la carte qui possède la particularité d'être identique à celui du Délice-DF5.



Un autre projet auquel j'ai pu participer est celle du site internet de l'Association des Patients de la Maladie de Fabry (APMF), une association chargée de l'accompagnement des patients de la maladie de Fabry\* en leur proposant du soutien, une écoute ainsi que des informations au sujet de la maladie. Pour ce projet je suis principalement intervenu sur la finalisation du design du site avec l'aide des différents stagiaires participant au projet.



#### RENCONTRE INTERNATIONALE DES PATIENTS FABRY

23 juillet 2021

La 6ème Rencontre Internationale des Patients Fabry s'est tenue du 18 au 21 octobre 2007 à Munich - ALLEMAGNE.  
(...)

[En savoir plus](#)



#### 2ÈME RENCONTRE NATIONALE DES PATIENTS FABRY

23 juillet 2021

La 2ème Rencontre Nationale des Patients Fabry s'est tenue le samedi 12 mai 2007 à Reims.  
(...)

[En savoir plus](#)



#### LA CONVENTION AERAS EN DÉTAIL

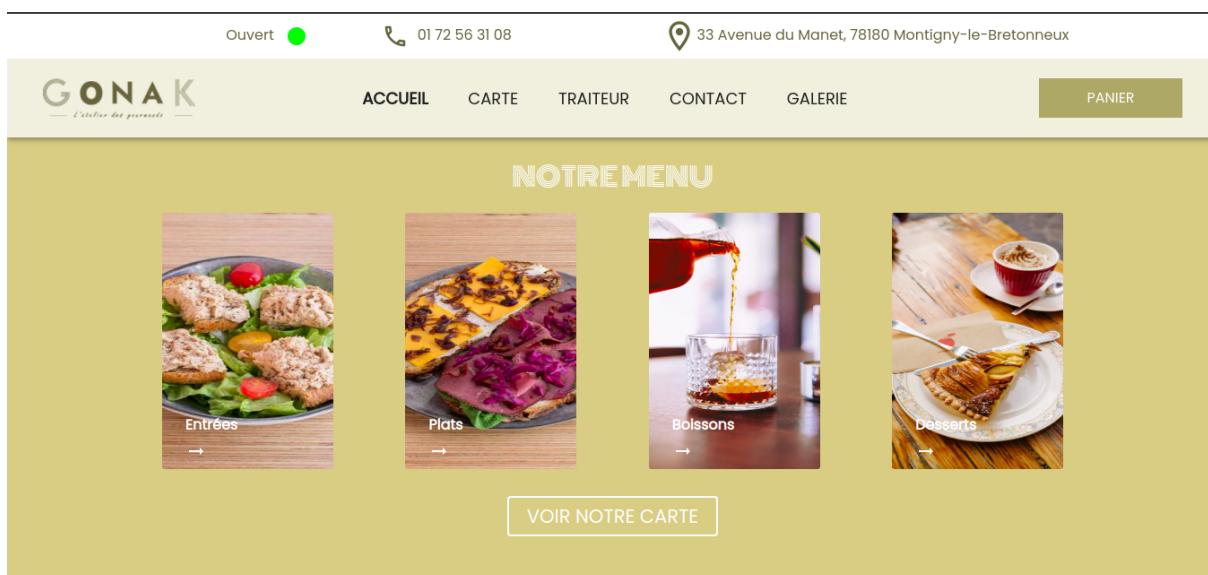
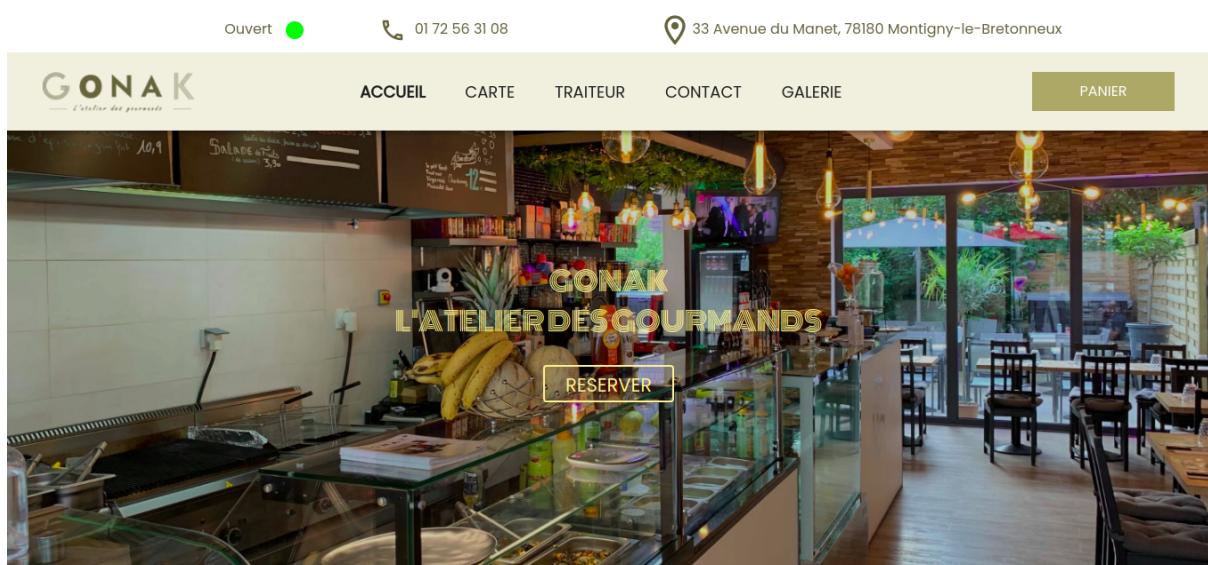
23 juillet 2021

La convention AERAS a pour objet de proposer le plus grand nombre de solutions pour permettre aux personnes ayant ou ayant eu un problème (...)

[En savoir plus](#)



Un projet où ma participation a été plus conséquente est celui du site internet du Gonak, un restaurant semi-gastronomique. Pour ce projet j'ai participé à la création du back-end du restaurant, je suis également intervenu sur le front-end au niveau de l'agencement de la partie administrateur du restaurant, j'ai participé à l'implémentation d'un nouvel onglet, la Gestion des produits permettant l'édition, la modification et la suppression d'un produit de la carte. Il s'agit d'une fonctionnalité qui a été initiée par Abdelhadi Bayechou et Mustapha Jaridi et dont je me suis occupé à étendre leur travail pour la partie administrateur du Gonak. J'ai également contribué à l'agencement de la carte et du panier au niveau de la partie cliente.



Enfin un dernier projet auquel j'ai pu participé est celui du site internet du Mustang, un restaurant type kebab également très similaire au Délice-DF5 et au Dima-Délice. Au sein de ce projet je suis intervenu dans la création du back-end du restaurant à travers une formation dédiée aux nouveaux stagiaires.



A screenshot of the Mustang website showing a product listing for a KEBAB. The product image shows a kebab sandwich with lettuce and tomato, served with fries and a Coca-Cola can. To the right, the product name "KEBAB" is listed with a quantity selector (minus, 1, plus, trash) and a price of "7€60". Below this, a detailed list of ingredients includes Pain, Tortilla, Garniture, Oignons, Salade, Sauce, Youpie, Supplément 1.6€, Mozzarella 0.8€, La Vache qui rit 0.8€, Boisson, and Fanta orange. At the bottom, the total price "7.6 €" is displayed next to a "Prix Total (ttc)" label, and there is a comment input field labeled "Commentaire".

## Mes difficultés :

### Partie applications mobiles :

Pour la partie mobile liée à l'application MARKUS, l'ensemble de ces tâches ne se sont pas faites avec facilité, j'ai en effet pu faire face à de nombreuses difficultés qui se sont le plus situées vers la partie du front-end de l'application. Mise à part les difficultés liées au différents composants, méthodes et librairies proposées par React Native comme avec les flatlists, les props ou encore la navigation. Il s'agit ici de difficultés que j'ai pu résoudre après expérimentations et recherches. Malheureusement j'ai fait face à des difficultés de l'ordre technique qui ont eu pour conséquence de ralentir mon avancée sur l'application.

La première difficulté que j'ai rencontrée s'est située au niveau du code de l'application, il faut savoir que je fais partie de la cinquième génération de développeur pour le cas du front-end et cela peut se ressentir lorsqu'on navigue dans le code.

En effet, on peut observer que chaque page n'utilise pas la même structure de code. Par exemple, nous pouvons trouver différentes manières d'agencement des styles dans le code de l'application, les styles peuvent dans un premier se trouver tout en bas de la disposition des composants comme dans le screen ci-dessous. On peut retrouver également les styles au sein du components créé. Enfin on retrouve les styles dans un fichier à part qui sera ainsi importé et utilisé dans d'autres pages.

Exemple d'agencement de styles

```
<View style={styles.alertFournisseur}>
    <Image source = {this.ico.warning} style={styles.warning}/>
    <Text style={styles.textWarning}>
        Vous n'avez pas encore ajouté de fournisseur.
    </Text>
</View>
```

```

alertFournisseur: {
  marginTop: '5%',
  alignItems: 'center',
},
warning : {
  marginTop: '10%',
  maxWidth: '100%', // à
  minHeight: 61,
  width: 68,
  height: 0,
},
textWarning: {
  marginTop: '2%',
  textAlign: 'center',
  fontSize: 15,
  color: '#fff',
},

```

Autre exemple d'agencement de styles

```

<TextInput //Nom de la marchandise
  style={{
    marginTop: '4%',
    width: '90%',
    height: 55,
    borderWidth : 1.0,
    borderRadius: 8,
    backgroundColor: 'white',
    justifyContent: 'center',
    color: 'black',
    paddingLeft: 35,
  }}
  placeholder='Nom Marchandise'
  value={this.state.nom}
  onChangeText={(value) => this.setState({nom : value})}
/>

```

On peut rajouter à cet exemple la structure des fonctions où l'on peut retrouver des fonctions fléchées<sup>(13)</sup> ainsi que des fonctions plus communes, la structure également des flatlists<sup>(14)</sup> et des items<sup>(15)</sup> qui semblent différer selon les pages. Ainsi le code est très hétérogène dans sa forme et comporte peu de commentaire pour comprendre l'ensemble des fonctionnalités, ce qui résulte au final à un temps d'adaptation plus long. Cela avait été également le cas pour

la partie back-end jusqu'à l'arrivée du stagiaire Porus qui s'est chargé d'optimiser et de configurer le code du back-end pour convenir à Django Rest Framework.

Une autre difficulté que nous avons pu faire face et qui concerne le CLI<sup>(10)</sup> React Native. En effet, son installation dans un premier temps ainsi que son utilisation quotidienne a été le porteur pour nous de soucis techniques. À savoir que nous utilisons le CLI React Native via le support de l'IDE Android Studio, nécessaire pour effectuer nos différents tests sur émulateur mobile et tablette, les erreurs de compilation où sont en cause des fichiers android apparaissent de manière fréquente. Il s'agit d'un élément qui selon le type d'erreur peut nous faire perdre plusieurs heures de travail, ainsi je pense que le basculement vers la réalisation de sites internet a été une bonne solution.

### Partie sites internet :

En ce qui concerne la partie impliquant les différents projets de sites internet, mes difficultés ont été moins nombreuses, cela peut s'expliquer du fait de l'expérience que j'ai pu accumuler des différentes technologies utilisées au niveau de nos projets. Néanmoins un problème assez récurrent a pu intervenir lors de l'implémentation de nos sites internet, il s'agit du problème de structuration de code. En effet, pour pouvoir gagner du temps sur nos projets de sites, nous utilisons souvent comme modèle un de nos anciens projets. Malgré l'avantage incontestable de commencer sur un projet déjà réalisé et fonctionnel pour l'implémentation de notre nouveau projet, des difficultés peuvent survenir à l'intérieur du code au niveau de l'agencement choisi par le projet.

Le projet que nous prenons comme base a pour la plupart du temps un agencement au niveau du dossier de code qui lui est propre, une réutilisation de ce même dossier de code pour un nouveau projet lui impose alors cette structuration à suivre, ainsi dans le cas où une structuration différente est nécessaire au respect du cahier des charges du projets du nouveau projet il devient difficile de restructurer le code sans générer différents bugs ou erreur par la suite. Sans compter que les dossiers et fichiers de code deviennent de moins en moins explicite pour notre projet actuel car ayant été initié à la base pour un autre projet. Il faut alors avoir une connaissance globale de l'ensemble des projets réalisés pour être capable de mener au mieux une restructuration du code.

## Partie 5 : Bilan d'expérience

Ce que je retiens de cette première expérience en entreprise, c'est tout d'abord ma réelle progression en termes d'autonomie. Ayant rejoint le projet MARKUS sans aucun supérieur hiérarchique dans mon pôle d'activité, il m'a fallu par moi-même comprendre le fonctionnement du front-end et du back-end de l'application. Cela se résume en des recherches sur la documentation de chaque technologie afin de comprendre les implémentations qui me posaient problème, le visionnage également de vidéo d'explication pour avoir un point de vue plus concret du fonctionnement de chaque fonctionnalité. J'ai pu notamment beaucoup apprendre des différents stagiaires qui sont intervenus au sein du projet, que ce soit en termes d'outils et de logiciels utilisés ou bien en termes de programmation pure, cela fut également une source d'apprentissage pour moi.

Cette première expérience m'a permis aussi d'approfondir mes compétences en gestion de projet. Le fait d'être en capacité de définir une demande en une tâche informatique, d'essayer de respecter au mieux la deadline malgré les différents problèmes rencontrés et surtout la capacité de se répartir les tâches selon les compétences de chacun sont pour moi l'ensemble des compétences que j'ai pu développer et stimuler à travers les projets de CEOS TECH. J'ai pu remarquer en outre une progression en ma capacité de vulgarisation et de synthèse de mes tâches informatiques, une capacité que je stimule lors de l'ensemble de mes réunions et surtout via le prisme des différentes formations aux technologies utilisées par CEOS TECH dont j'ai été en charge d'animer à l'arrivée de chaque nouveau stagiaire.

Et par-dessus tout, cette première expérience m'a confortée dans l'idée d'orienter ma future carrière professionnelle vers le secteur de la Data et du back-end. Tout au long de mes missions et tâches au sein du projet MARKUS j'ai eu cet attrait à développer le back-end de l'application, j'en ai également beaucoup appris en termes de transmission, de récupération et de sécurisation des données. Ainsi, continuer ma carrière pour parvenir à un poste mêlant la Data est désormais pour moi un réel objectif.

## Conclusion

Pour conclure, je suis extrêmement satisfait de l'accueil proposé par CEOS TECH, malgré une montée en compétence et une montée des enjeux assez rapide, les membres CEOS TECH sont toujours restés à notre écoute et nous ont laissé le temps nécessaire pour nous acclimater à ce nouvel environnement via notamment l'organisation de nombreuses formations pour parfaire nos compétences nouvellement acquises. Cette première expérience chez CEOS TECH a été incontestablement riche en termes de programmation informatique, elle a su m'apporter une vision complète et globale du métier de développeur et elle m'a également permis d'intégrer les codes du monde professionnel.

Nous avons eu le droit de participer à des évènements de team building, ce qui nous a permis de voir l'ensemble de l'équipe dans un cadre moins formel et d'également faire naître une cohésion de groupe au sein de l'entreprise. Ainsi je suis heureux d'avoir contribué à la progression d'une jeune société car grâce à eux j'ai évolué en tant que développeur mais surtout en tant qu'homme.

## Indexes

(1) : Une application native est une application mobile (smartphone et tablette) qui a été développée selon le langage associé à son système d'exploitation (iOS, Android, etc.). Dans le cas de l'application mobile MARKUS, ReactJS permet de convertir notre production Javascript selon les langages natifs utilisés par Android et IOS soit respectivement le Java et l'Objective-C.

(2) : Le state est un objet composé de différentes propriétés (ou props) observables qui contrôlent le comportement du composant courant. Ainsi selon l'état du state pouvant changer au cours de l'utilisation de l'application ou site internet, il est possible de changer intégralement l'affichage du composant courant sans le recharger, ce qui donne justement plus de "réactivité" aux différents outils utilisant cette fonctionnalité de React.

(3) : Une API (Application Programming Interface) est un moyen informatique qui permet à des applications de communiquer entre elles et de s'échanger mutuellement des services ou des données.

(4) : Un IDE (Integrated Development Environment) est un logiciel qui regroupe un ensemble d'outils de développements dédiés aux programmeurs afin qu'ils puissent optimiser leur temps de travail et améliorer leur productivité. Autrement dit, l'IDE facilite la mise en œuvre de projets tels que le développement de logiciels ou d'applications.

(5) : Une fiche technique est une fiche qui détaille la réalisation d'un plat servi en restauration, il s'agit autrement dit d'une recette de cuisine.

(6) : Un logiciel SAAS (software as a service) veut signifier un logiciel applicatif hébergé et exploité en dehors de l'organisation ou de l'entreprise par un tiers. L'outil est ainsi accessible à la demande via un accès Internet.

(7) : Un KPI (Key Performance Indicators) est une fonctionnalité qui concentre tous les indicateurs clé de performance d'une entité précise sous la forme de données et graphiques. Le tableau de bord d'une voiture par exemple est ainsi un regroupement de KPIs.

(8) : Une variable est un élément nommé associé à une certaine information ou valeur que l'on peut manipuler et modifier tout au long du programme.

(9) : Une fonction est un petit programme destiné à être appelé de manière fréquente, elle évite donc la répétition de code dans un programme.

(10) : Un site internet disposant d'une structure one page est un site internet ne disposant que d'une seule page, ainsi toute la navigation du site se joue sur sa verticalité.

(11) : Un CLI est une interface homme-machine dans laquelle la communication entre l'utilisateur et l'ordinateur s'effectue en mode texte via l'utilisation de différentes commandes.

(12) : Une variable globale est généralement une variable déclarée à la racine d'un projet de programmation et dont son utilisation s'étend à l'ensemble des fichiers de programmation présentes sur le projet.

(13) : Une fonction fléchée est une fonction exclusif à l'utilisation de React et sert à pouvoir contenir comme paramètre les propriétés associées au composant courant

(14) : Une flatlist un composant de React qui va nous permettre de nous faciliter le listage des autres composants qui seront à l'intérieur de celui-ci.

(15) : Un item ce cas précis est un composant qui est à l'intérieur d'une flatlist

## Liens et références

Lien du site internet O'Lokoso : <https://olokososite-48316.web.app/>

Lien du site internet Délice-DF5 : <https://delicedf5-blancmesnil-nord.fr/> (nouveau design)

Lien du site internet Dima-Délice : <https://dimadelice.fr/>

Lien du site internet APMF : <https://apmf-2c9c0.web.app/home>

Lien du site internet Gonak : <https://sitegonak.web.app/home>

Lien du site internet Mustang : <https://mustangsite-760db.web.app/>