



X



Rapport de stage

Développement d'un site internet commercial sous ReactJs

Réalisé par :

Rayane CATHELIN

Sous la supervision de :

Sofian BENATIA

Année universitaire 2024-2025

Remerciements

Je tiens à remercier chaleureusement la société IZYDESK de m'avoir accueilli pour réaliser mon stage de fin de deuxième année de BUT Informatique, et plus particulièrement Nordine EL OUACHMI, CEO de l'entreprise, ainsi que Sofian BENATIA, CTO et co-fondateur, pour m'avoir accueilli, guidé, encadré et challengé tout au long de cette expérience.

Je remercie également Simon LASCAUD, développeur Flutter et Chef de Projet, pour sa disponibilité constante, ses nombreux conseils techniques, son regard critique et les échanges réguliers qui m'ont permis de progresser dans la bonne direction.

Plus globalement, je souhaite exprimer ma gratitude à toute l'équipe technique et administrative d'IZYDESK, pour leur bienveillance, leur accueil et l'ambiance de travail stimulante dans laquelle j'ai pu évoluer.

Enfin, je souhaite aussi remercier l'équipe pédagogique de l'IUT de Montpellier, et en particulier mes enseignants de la spécialité Développement, pour m'avoir permis d'acquérir les bases et la rigueur nécessaires pour aborder sereinement ce premier stage en entreprise.

Résumé et Mots-Clés

Ce document est un rapport de stage effectué au sein de l'entreprise IZYDESK et rédigé par un étudiant de l'IUT de Montpellier dans le cadre de la fin de sa deuxième année d'études. Il présente le travail réalisé durant la période de stage, principalement axé sur le développement front-end d'un site de commande en ligne destiné au secteur de la restauration.

Ce rapport se concentre sur les différentes problématiques rencontrées lors de la refactorisation, de l'optimisation et de l'évolution du projet existant. Le travail a porté sur plusieurs aspects :

1. L'amélioration de l'architecture React/Next.js en passant de hooks individuels à une gestion de contexte globale pour limiter la redondance et améliorer les performances.
2. La mise en œuvre de fonctionnalités essentielles au fonctionnement d'un site de e-commerce
3. L'intégration graphique des éléments du site selon des maquettes précises, avec une attention particulière portée au responsive design pour garantir une expérience utilisateur fluide, aussi bien sur desktop que sur mobile.

Ce stage représente une mise en pratique concrète des compétences acquises à l'IUT et une immersion professionnelle enrichissante dans le cadre du développement web moderne.

Mots-clés : IZYDESK, développement web, React, Next.js, Tailwind CSS, composants, API, UX, UI, contexte, hooks, responsive design, optimisation, GitLab, SaaS, front-end, e-commerce

This document is an internship report performed in the company IZYDESK and written by a student of the University Institute of Technology (IUT) of Montpellier during the end of his second year of study. It introduces the work carried out during the internship period, mostly focused on front-end development of an online ordering website designed for the food service industry.

This report addresses several technical and functional aspects encountered during the rework and extension of an existing codebase. The internship mainly involved:

1. Improving the React/Next.js architecture, notably by moving from local hooks to a global context management to avoid redundant API calls and ensure better state handling.

Rapport de stage chez Izydesk

2. Implementing essential features required for a functional e-commerce website, including ordering logic, product display, customer address management, discount codes, and more.
3. Ensuring a high level of user interface and design integration, with strict compliance to design guidelines and a strong emphasis on responsive design for both desktop and mobile experiences.

This internship represents a concrete application of the skills acquired during the student's education and a valuable first immersion in a modern professional development environment.

Keywords: IZYDESK, web development, React, Next.js, Tailwind CSS, components, API, UX, UI, context, hooks, responsive design, optimization, GitLab, SaaS, front-end, e-commerce

Sommaire

Remerciements	2
Résumé et Mots-Clés	3
Sommaire	5
Tables des Figures	6
Glossaire	7
1 - Contexte du stage	9
1.1 - Présentation de l'entreprise Izydesk	9
1.2 – Organisation interne	10
1.3 - Dimension commerciale	11
1.3.1 - Secteur de la restauration	11
1.3.2 - Secteur tertiaire et de la santé	12
2 - Définition du stage	13
2.1 - Enjeux du stage	13
2.2 - Analyse technique	14
2.3 - Cahier des charges	16
2.3.1 - Besoins fonctionnels	16
2.3.2 - Contraintes techniques	22
2.3.3 - Résultats attendus	22
3 - Rapport technique	23
3.1 – Mise en place de l'environnement de travail	23
3.2 – Refactorisation du menu en composants React*	26
3.3 – Afficher uniquement les éléments nécessaires	28
3.4 – De l'utilisation de hooks personnalisés à l'intégration d'un contexte global	29
3.4.1 - Limites de cette approche	30
3.4.2 - Mise en place d'un contexte [2] global	30
3.5 – Personnalisation des pages d'erreur	32
3.6 – Affichage responsive des valeurs nutritionnelles et allergènes	33
3.7 - Conclusion	36
Apprentissages Critiques mobilisés :	36
5 – Conclusion	38
Bilan du travail réalisé	38
Bilan de l'expérience en entreprise	39
Bilan de l'expérience personnelle	39
6 - Annexes	41
Annexe 8 : Diagramme de classe	47
6.1 - Bibliographie	48

Tables des Figures

Figure 1 : Organigramme.....	10
Figure 2 : Structure globale du code vers le début du projet.....	15
Figure 3 : Diagramme des cas d'utilisations.....	17
Figure 3 : Exemple de temps d'exécution avec l'implémentation Next.....	24
Figure 4: Exemple de l'interface de Bruno*.....	25
Figure 5 : Nouveau découpage du menu en plusieurs composants.....	26
Figure 6 : Aperçu de la structure d'un produit avec menu via Bruno*.....	27
Figure 7 : Aperçu de l'attribut isDisplayForPos Dans Bruno*.....	28
Figure 8 : Extrait du code de la page principale montrant le filtrage des produits et categories.....	28
Figure 9 : Hook React* useCompanyDetails().....	29
Figure 10 : endpoints get-company.....	29
Figure 11 : Exemple d'un context global.....	31
Figure 12 : Exemple d'un reducer.....	31
Figure 13 : Récupération des informations d'erreur et entreprises via context [5].....	32
Figure 14 : Contenu affiché pour la page d'erreur 404.....	32
Figure 15 : Boutons de navigation permettant de revenir en arrière ou de retourner à l'accueil.....	33
Figure 16: Affichage non responsive pour mobile.....	33
Figure 17 : Fonctionnement non responsif.....	34
Figure 18 : Exemple de code display mobile.....	34
Figure 19 : Exemple d'implémentation responsive.....	35
Figure 20 : Affichage responsive pour mobile.....	35

Glossaire

API : Interface de Programmation Applicative, permet à deux systèmes de communiquer.

React : Librairie JavaScript pour construire des interfaces utilisateur.

Next.js : Framework basé sur React qui permet de faire du server-side rendering.

SPA : Single Page Application, site web qui se recharge dynamiquement sans rechargelement complet.

SaaS : Software as a Service, modèle où le logiciel est hébergé sur un serveur distant.

TailwindCSS : Framework CSS utilitaire permettant un design rapide et responsive.

Docker : Outil de virtualisation permettant de créer des environnements de développement isolés.

Bruno : Outil de test d'API REST et GraphQL, similaire à Postman. Il permet de structurer, d'organiser et d'exécuter des requêtes HTTP pour interagir avec des APIs. Utilisé chez Izydesk pour tester les appels API disponibles dans les projets, notamment ceux intégrés au site web.

Prop drilling : Terme utilisé en React pour désigner le passage de données d'un composant parent à un composant profondément imbriqué, en transitant par plusieurs niveaux intermédiaires qui n'ont pas nécessairement besoin de ces données. Cette pratique rend le code plus complexe, moins lisible et plus difficile à maintenir.

Rapport de stage chez Izydesk

1 - Contexte du stage

1.1 - Présentation de l'entreprise Izydesk

Spécialisée dans l'édition de logiciels orientés métier, Izydesk est une entreprise française fondée en mars 2024 et implantée à Montpellier. Elle conçoit et développe des solutions numériques combinant interfaces logicielles et dispositifs matériels, dans le but d'accompagner les professionnels dans la digitalisation de leurs opérations et la modernisation de l'expérience utilisateur.

Dès sa création en mars 2024, Izydesk a fait le choix de s'appuyer sur une infrastructure logicielle centralisée, articulée autour d'une plateforme **SaaS*** propriétaire. Celle-ci permet de connecter, configurer et superviser un ensemble de dispositifs (bornes, écrans, caisses, applications mobiles, etc.) depuis une interface unique, sans nécessiter d'installation complexe côté client.

L'entreprise adopte une stratégie de développement centrée sur la maîtrise complète de son socle technologique, lui permettant de proposer des solutions sur mesure, interopérables avec les systèmes existants, et évolutives. Tous les développements sont réalisés en interne, garantissant une expertise technique approfondie et une capacité d'adaptation à des problématiques spécifiques, y compris complexes.

Izydesk place au cœur de son approche l'expérience utilisateur, la fiabilité des outils déployés, et l'optimisation des parcours clients. Cette orientation, combinée à une culture de l'innovation et à une équipe jeune et dynamique, permet à l'entreprise de se positionner comme un acteur émergent dans le secteur des technologies de service.

1.2 – Organisation interne

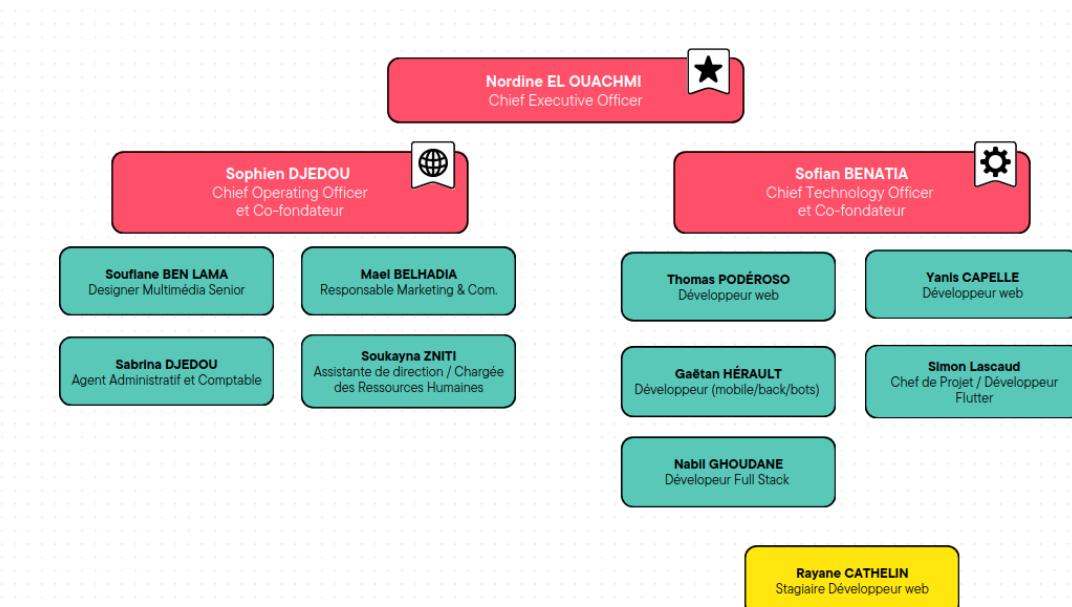
L'entreprise Izydesk est dirigée par Nordine EL OUACHMI, qui occupe la fonction de Chief Executive Officer (CEO). Deux co-fondateurs assurent la structuration des pôles opérationnels : Sophien DJEDOU, Chief Operating Officer (COO), et Sofian BENATIA, Chief Technology Officer (CTO).

L'organisation interne est répartie en trois grands pôles fonctionnels, regroupant à ce jour une quinzaine de collaborateurs (cf. Figure 1) :

- Le pôle administratif et commercial, supervisé par M. DJEDOU, regroupe les fonctions de communication, marketing, comptabilité et ressources humaines.
- Le pôle technique, encadré par M. BENATIA, est en charge du développement des solutions logicielles, de la maintenance de la plateforme SaaS et de la gestion des projets techniques.
- Le pôle direction générale, assumé par M. EL OUACHMI, assure le pilotage stratégique, les orientations de développement et les partenariats.

Au sein de ce schéma, j'ai été intégré à l'équipe technique en tant que stagiaire développeur web, sous la supervision directe de Sofian BENATIA. J'ai également été accompagné tout au long du projet par Simon Lascaud, Project Manager et développeur Flutter, qui m'a aidé à prendre en main l'architecture du site et à corriger certains éléments techniques. J'ai ainsi pu collaborer régulièrement avec les développeurs de l'équipe, notamment sur les aspects liés au front-end (interface utilisateur), à l'intégration des appels API*, et à la structuration des composants dans un environnement React*/Next.js*. Mon positionnement dans l'organigramme est représenté en jaune, à la base de la branche technique.

Figure 1 : Organigramme



1.3 - Dimension commerciale

L'activité commerciale d'Izydesk s'articule autour de la digitalisation des parcours utilisateurs dans différents secteurs professionnels, principalement la restauration, le tertiaire et la santé. L'entreprise conçoit des solutions matérielles (bornes, écrans, caisses, etc.) et logicielles (applications, interfaces, affichage dynamique...), qu'elle regroupe dans un écosystème centralisé et interconnecté via une plateforme **SaaS***.

Ce modèle permet aux clients de gérer l'ensemble de leurs dispositifs (matériel et logiciel) à partir d'un point d'administration unique, accessible en ligne, sans installation locale complexe. Ce socle SaaS est un élément central de l'offre Izydesk : il assure la cohérence des échanges, la synchronisation des données et le paramétrage des différents points de contact avec les utilisateurs finaux.

L'organisation commerciale de l'entreprise repose ainsi sur trois pôles sectoriels, chacun structurant une offre adaptée aux besoins spécifiques du domaine.

1.3.1 - Secteur de la restauration

Izydesk intervient auprès d'acteurs de la restauration de tous formats : restaurants traditionnels, fast-foods, franchises, food-courts, dark kitchens ou food-trucks. Le fonctionnement quotidien de ces structures exige des outils fiables pour gérer un volume important de commandes, souvent sur des temps très courts.



Les solutions proposées incluent :

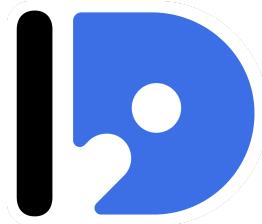
- des bornes de commande libre-service ;
- des caisses connectées (avec écran client intégré) ;
- des écrans KDS (Kitchen Display System) en cuisine ;
- des menus tablettes, pads serveurs pour le service à table ;
- des imprimantes cuisine pour chaque poste de préparation ;
- une application mobile personnalisable (commande, click & collect, fidélité).

Rapport de stage chez Izydesk

L'ensemble est connecté à la plateforme **SaaS*** d'Izydesk, qui permet aux restaurateurs de gérer les menus, produits, disponibilités, prix, et statistiques en temps réel, tout en assurant la synchronisation avec le matériel sur site.

1.3.2 - Secteur tertiaire et de la santé

Dans les bureaux, espaces d'accueil d'entreprises ou de collectivités, les besoins se concentrent sur l'optimisation de l'accueil des visiteurs et la gestion des flux internes. Izydesk propose des solutions destinées à fluidifier ces processus :



- des bornes d'accueil interactives avec identification automatique
- des systèmes d'enregistrement pour visiteurs ou collaborateurs
- des casiers connectés, pour remise sécurisée de documents ou matériels
- des interfaces mobiles de check-in/check-out
- des écrans dynamiques diffusant des informations internes

La plateforme **SaaS*** permet ici aussi une gestion centralisée : configuration des bornes, planification des accès, statistiques d'usage, alertes en cas d'anomalie, etc. Cela permet aux responsables d'accueil de conserver une vue d'ensemble et d'agir rapidement en fonction des besoins.

2 - Définition du stage

2.1 - Enjeux du stage

Le stage que j'ai effectué chez Izydesk présente plusieurs enjeux majeurs, tant sur le plan technique que fonctionnel. L'un des premiers défis consiste à concevoir et développer une interface web intuitive et moderne. Cette interface doit offrir une expérience utilisateur fluide et agréable, en assurant une navigation simple et efficace. Pour cela, il est essentiel de concevoir des pages ergonomiques, adaptées aux besoins des restaurateurs, en prenant en compte les principes de l'UX design et en optimisant la disposition des éléments graphiques.

Un autre aspect crucial de ce projet concerne l'intégration des API* internes d'Izydesk. Cette intégration vise à garantir une communication fluide entre le site de vente en ligne et le système d'information central de l'entreprise. Cela implique la mise en place de requêtes API* efficaces, permettant de récupérer, traiter et envoyer les données nécessaires au bon fonctionnement de la plateforme.

Par ailleurs, l'optimisation des performances est un enjeu essentiel de ce projet. Il est primordial que le site assure un chargement rapide des pages et un traitement efficace des données, afin d'éviter tout ralentissement susceptible d'entraver l'expérience utilisateur. Cela passe par une gestion rigoureuse des ressources, l'utilisation de bibliothèques et de techniques de programmation adaptées ainsi qu'une attention particulière aux détails techniques qui peuvent influencer la performance globale du site.

Enfin, le projet doit être conçu de manière à être compatible avec différents types d'appareils, notamment les ordinateurs, les tablettes et les smartphones. L'enjeu ici est d'assurer une expérience homogène, quel que soit le support utilisé par les restaurateurs pour accéder à la plateforme. Pour cela, l'approche responsive design est essentielle et doit être prise en compte tout au long du développement.

Ce stage constitue une opportunité d'approfondir mes compétences en développement web et en conception de systèmes interactifs. Il me permet d'acquérir une expérience significative dans la réalisation d'un projet concret, en appliquant les connaissances théoriques acquises au cours de ma formation à un contexte professionnel réel. Il s'agit également d'une expérience enrichissante en matière de gestion de projet, de collaboration avec une équipe et d'adaptation aux exigences du monde de l'entreprise.

2.2 - Analyse technique

L'environnement technique de l'entreprise Izydesk repose sur une organisation moderne, structurée autour d'un stack technologique JavaScript, d'un processus de développement collaboratif hébergé sur GitLab, et de pratiques de communication adaptées au travail d'équipe en mode projet.

Le développement des projets s'appuie sur React.js* pour la construction des composants front-end, associé à Next.js, qui permet de bénéficier de fonctionnalités supplémentaires telles que le rendu côté serveur, la génération statique des pages et une meilleure gestion du routage. La mise en page est assurée via TailwindCSS*, qui facilite la conception d'interfaces responsives et cohérentes. L'environnement de développement est soutenu par Docker*, utilisé pour simplifier la configuration locale et garantir la reproductibilité des installations.

Tous les projets de l'entreprise sont centralisés sur une instance GitLab privée. Chaque application dispose de son propre repository, dont l'accès est géré de manière individuelle. Le projet sur lequel j'ai été affecté, intitulé *Izy Order Website*, est hébergé sur cette plateforme. L'organisation du dépôt suit une méthode de gestion de version structurée : la branche main représente la version de production, tandis que la branche dev constitue la version en cours de développement. Les évolutions sont réalisées à partir de branches créées depuis dev, puis soumises à une merge request pour révision. Après validation par Simon Lascaud ou Sofian Benatia, les modifications peuvent être fusionnées dans main en vue d'une mise en production. Cette organisation garantit un suivi rigoureux des évolutions, une traçabilité des modifications, et un contrôle qualité avant déploiement. Les bonnes pratiques de Git sont respectées (commits explicites, revues de code, vérification avant fusion).

La communication au sein de l'équipe repose sur des outils numériques adaptés au travail collaboratif. Slack est utilisé pour les échanges quotidiens, les questions techniques, ou la coordination rapide dans l'équipe. Google Meet est utilisé pour les réunions à distance, en particulier lors des revues de code ou des points d'avancement. Chaque lundi matin, un point d'équipe hebdomadaire est animé par Sofian Benatia. Il permet à chaque membre de présenter l'état de ses tâches, de faire remonter les éventuels blocages, et de clarifier les priorités de la semaine. En complément, des appels ponctuels sont organisés au cours de la semaine avec Simon ou Sofian, permettant d'assurer un accompagnement continu et de débloquer certaines situations complexes.

Lorsque j'ai intégré l'équipe, le projet du site web disposait déjà d'une base de code existante développée en React*. Cependant, cette base était partiellement fonctionnelle et nécessitait une refonte ou une reprise progressive afin d'être pleinement opérationnelle.

Rapport de stage chez Izydesk

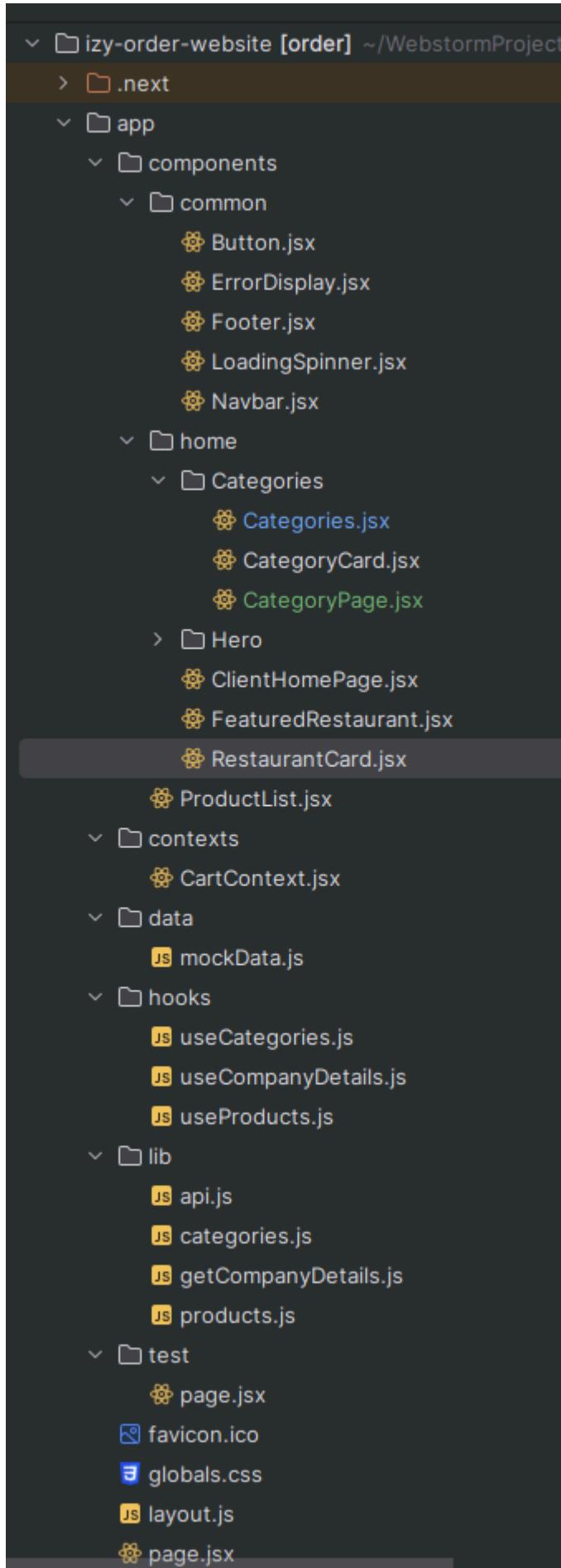


Figure 2 : Structure globale du code vers le début du projet

2.3 - Cahier des charges

Contexte

Le projet correspond à un site de commande en ligne destiné aux clients finaux des restaurateurs. Une base de code existait au démarrage du stage, développée en React* et Next.js, mais elle n'était pas complètement fonctionnelle. L'objectif de la mission était de reprendre cette base existante afin de la rendre opérationnelle et proche d'un niveau de mise en production, en intégrant les composants nécessaires, en corrigeant les dysfonctionnements identifiés et en assurant la cohérence fonctionnelle et visuelle de l'ensemble.

Le cahier des charges à remplir n'étant pas spécialement indiqué par l'entreprise pour la mission qui m'a été assignée, je présente ici celui que j'ai personnellement rédigé pour éclaircir mes objectifs et les contraintes :

Objectif

L'objectif général du stage est de développer une version stable, complète et fluide du site de commande, dans un environnement technique déjà en place, tout en respectant les normes internes de l'entreprise (qualité du code, architecture, intégration API*, bonne gestion des états, composants réutilisables, etc.).

2.3.1 - Besoins fonctionnels

- ✓ Affichage dynamique de tous les produits disponibles via appel à l'API*, conforme aux maquettes de design
- ✓ Mise en page responsive et ergonomique des menus, catégories et articles
- ✓ Intégration d'un système de filtres et de catégories pour la navigation produit (intégration conditionnelle)
- ✓ Gestion complète du panier : ajout, modification, suppression d'articles
- ✓ Synchronisation du panier avec les données issues des API*
- ✓ Implémentation d'un système de commande avec étape de validation
- ✓ Création de formulaires d'adresses client (ajout, édition, suppression)
- ✓ Système de connexion, déconnexion et inscription client
- ✓ Affichage de l'historique des commandes
- ✓ Intégration des commentaires clients sur les menus ou commandes
- ✓ Affichage des valeurs nutritionnelles et allergènes sur les produits
- ✓ Intégration de codes promotionnels (vérification, application de réduction)
- ✓ Affichage conditionnel et gestion des *product addons* (options complémentaires des produits)
- ✓ Traduction du site (internationalisation) avec changement de langue dynamique
- ✓ Affichage cohérent et lisible du panier, conforme aux maquettes de design
- ✓ Gestion des contraintes liées aux types de commande (à emporter, sur place, livraison)
- ✓ Affichage d'alertes ou messages d'erreur en cas d'action invalide (login requis, données incomplètes, etc.)

Rapport de stage chez Izydesk

- ✓ Décomposition des composants conséquent en plusieurs composants plus adapté.



Figure 3 : Diagramme des cas d'utilisations

Gestion du compte

CU01 - S'inscrire

- **Objectif** : Permettre à un nouvel utilisateur de créer un compte
- **Scénario nominal** : L'utilisateur accède au formulaire d'inscription, remplit ses informations personnelles, le système vérifie les données et crée le compte

Rapport de stage chez Izydesk

- **Autres scénarios :** Si les informations sont incomplètes ou incorrectes, le système affiche un message d'erreur et invite l'utilisateur à corriger

CU02 - Se connecter

- **Objectif :** Permettre à un utilisateur existant d'accéder à son compte
- **Scénario nominal :** L'utilisateur saisit son email et mot de passe, le système vérifie et authentifie l'utilisateur
- **Autres scénarios :** Si les identifiants sont incorrects, le système affiche un message d'erreur. Après trois échecs consécutifs, le compte est temporairement bloqué

CU03 - Se déconnecter

- **Objectif :** Permettre à un utilisateur de quitter son compte
- **Scénario nominal :** L'utilisateur clique sur "Déconnexion", le système ferme la session
- **Autres scénarios :** Si la session expire automatiquement, l'utilisateur est redirigé vers la page d'accueil

CU04 - Gérer mes adresses

- **Objectif :** Permettre à l'utilisateur de gérer ses adresses
- **Scénario nominal :** L'utilisateur accède à la section "Mes adresses", ajoute, modifie ou supprime des adresses de livraison et facturation
- **Autres scénarios :** Si les informations d'adresse sont incomplètes, le système demande les données manquantes

Parcours produits

CU05 - Consulter les produits

- **Objectif :** Permettre de voir l'ensemble des produits disponibles
- **Scénario nominal :** L'utilisateur parcourt la liste des produits affichés avec leurs images et prix
- **Autres scénarios :** Si aucun produit n'est disponible, un message approprié est affiché

CU06 - Filtrer par catégorie

- **Objectif :** Permettre de voir uniquement les produits d'une catégorie
- **Scénario nominal :** L'utilisateur sélectionne une catégorie, le système filtre et affiche les produits correspondants
- **Autres scénarios :** Si la catégorie ne contient aucun produit, un message le précise

CU07 - Voir les détails d'un produit

- **Objectif :** Permettre de consulter les informations détaillées d'un produit
- **Scénario nominal :** L'utilisateur clique sur un produit, le système affiche sa description complète

Rapport de stage chez Izydesk

- **Autres scénarios :** Si des personnalisations sont possibles, elles sont présentées à l'utilisateur

CU08 - Consulter les valeurs nutritionnelles

- **Objectif :** Permettre de voir les informations nutritionnelles d'un produit
- **Scénario nominal :** L'utilisateur accède aux informations nutritionnelles depuis la page produit
- **Autres scénarios :** Si les informations ne sont pas disponibles, un message l'indique

CU09 - Consulter les allergènes

- **Objectif :** Permettre de voir les allergènes présents dans un produit
- **Scénario nominal :** L'utilisateur accède aux informations sur les allergènes depuis la page produit
- **Autres scénarios :** Si aucun allergène n'est présent, cette information est clairement indiquée

Gestion du panier

CU10 - Ajouter au panier

- **Objectif :** Permettre d'ajouter un produit au panier
- **Scénario nominal :** L'utilisateur sélectionne un produit et l'ajoute au panier
- **Autres scénarios :** Si le produit a des options obligatoires non sélectionnées, le système demande de les compléter

CU11 - Modifier quantité

- **Objectif :** Permettre de modifier la quantité d'un produit dans le panier
- **Scénario nominal :** L'utilisateur modifie la quantité d'un produit, le système met à jour le panier
- **Autres scénarios :** Si la quantité est mise à zéro, le système propose de supprimer l'article

CU12 - Supprimer du panier

- **Objectif :** Permettre de retirer un produit du panier
- **Scénario nominal :** L'utilisateur supprime un produit, le système met à jour le panier
- **Autres scénarios :** Si le panier devient vide, un message approprié est affiché

CU13 - Voir le panier

- **Objectif :** Permettre de consulter le contenu du panier
- **Scénario nominal :** L'utilisateur accède à son panier pour voir les produits ajoutés et le total
- **Autres scénarios :** Si le panier est vide, un message invite l'utilisateur à ajouter des produits

Rapport de stage chez Izydesk

CU14 - Appliquer un code promo

- **Objectif :** Permettre d'utiliser un code promotionnel
- **Scénario nominal :** L'utilisateur saisit un code promo valide, le système applique la réduction
- **Autres scénarios :** Si le code est invalide ou expiré, un message d'erreur s'affiche

CU15 - Ajouter des options complémentaires

- **Objectif :** Permettre de personnaliser les produits avec des options
- **Scénario nominal :** L'utilisateur sélectionne des options pour un produit avant de l'ajouter au panier
- **Autres scénarios :** Si des options sont incompatibles, le système en informe l'utilisateur

Processus de commande

CU16 - Choisir le type de commande

- **Objectif :** Permettre de sélectionner le mode de commande
- **Scénario nominal :** L'utilisateur choisit entre sur place, à emporter ou livraison
- **Autres scénarios :** Si un mode n'est pas disponible, il n'est pas proposé

CU17 - Sélectionner adresse de livraison

- **Objectif :** Permettre de choisir où livrer la commande
- **Scénario nominal :** L'utilisateur sélectionne une adresse existante ou en ajoute une nouvelle
- **Autres scénarios :** Si aucune adresse n'est enregistrée, le système demande d'en créer une

CU18 - Sélectionner adresse de facturation

- **Objectif :** Permettre de choisir l'adresse de facturation
- **Scénario nominal :** L'utilisateur sélectionne une adresse de facturation
- **Autres scénarios :** Par défaut, l'adresse de livraison est proposée comme adresse de facturation

CU19 - Ajouter un commentaire à la commande

- **Objectif :** Permettre d'ajouter des instructions spéciales
- **Scénario nominal :** L'utilisateur saisit un commentaire pour sa commande
- **Autres scénarios :** Si le commentaire dépasse la limite de caractères, l'utilisateur est informé

CU20 - Valider la commande

- **Objectif :** Permettre de finaliser et soumettre la commande
- **Scénario nominal :** L'utilisateur vérifie le récapitulatif et confirme sa commande

Rapport de stage chez Izydesk

- **Autres scénarios :** Si des informations obligatoires sont manquantes, le système demande de les compléter

Suivi des commandes

CU21 - Consulter l'historique des commandes

- **Objectif :** Permettre de voir l'ensemble des commandes passées
- **Scénario nominal :** L'utilisateur accède à son historique de commandes
- **Autres scénarios :** Si aucune commande n'a été passée, un message approprié est affiché

CU22 - Voir les détails d'une commande

- **Objectif :** Permettre de consulter le détail d'une commande spécifique
- **Scénario nominal :** L'utilisateur sélectionne une commande et accède à ses détails
- **Autres scénarios :** Si la commande est en cours de traitement, son statut est mis en évidence

2.3.2 - Contraintes techniques

- ✓ Utilisation obligatoire de React*, Next.js* et TailwindCSS*
- ✓ Maintenir la structure de l'architecture existante du projet (réutilisation et organisation des composants)
- ✓ Respecter le workflow GitLab de l'entreprise : création de branche depuis dev, merge request vers dev, puis fusion vers main après validation
- ✓ Ne pas modifier la structure ou les spécifications des APIs* internes utilisées
- ✓ Développement dans un environnement Docker* configuré pour le projet
- ✓ Assurer la compatibilité mobile (responsive design sur tous les écrans)
- ✓ Suivre les maquettes graphiques existantes : espacement, tailles, couleurs, agencement
- ✓ Traduire tous les éléments de l'interface utilisateur dans plusieurs langues avec gestion du contexte
- ✓ Éviter les rechargements inutiles : optimiser les appels API*, la gestion des états et le rendu des composants
- ✓ Maintenir un code clair, commenté, relisible et validé par l'équipe technique
- ✓ Intégrer les composants dans une logique modulaire et évolutive
- ✓ Veiller à l'accessibilité et à l'expérience utilisateur sur l'ensemble du parcours
- ✓ Livrer une version testable sans erreurs critiques en fin de stage

2.3.3 - Résultats attendus

- ✓ Site fonctionnel et utilisable sur desktop et mobile
- ✓ Navigation fluide entre les étapes du parcours utilisateur
- ✓ Affichage cohérent des données récupérées via API*
- ✓ Gestion du panier et de la commande
- ✓ Intégration correcte des codes promos, valeurs nutritionnelles et autres informations clients
- ✓ Site suffisamment avancé pour servir de base à une future mise en production

3 - Rapport technique

3.1 – Mise en place de l'environnement de travail

Avant de pouvoir contribuer au développement du site *Izy Order Website*, il a été nécessaire de mettre en place un environnement de travail complet et fonctionnel, compatible avec les outils internes utilisés par l'entreprise. Cette phase, bien qu'elle puisse paraître préliminaire, a représenté un véritable travail d'intégration technique, car elle impliquait à la fois des accès sécurisés, la configuration d'outils spécifiques, la compréhension de la base technique existante et la synchronisation avec les pratiques de l'équipe.

La première étape a consisté à obtenir les accès aux différents outils internes, notamment l'instance GitLab privée de l'entreprise, le SaaS* d'administration développé en interne, ainsi que l'espace de communication Slack. Ces accès sont indispensables pour interagir avec le code source du projet, tester les fonctionnalités via l'API*, et participer à la coordination avec les autres développeurs.

Une fois ces accès obtenus, il a fallu installer l'environnement local de développement. Le projet étant basé sur React* et Next.js*, avec une configuration Docker* spécifique. Cette étape a été relativement longue, notamment en raison de la documentation partielle et de certaines incompatibilités locales que j'ai dû résoudre (dépendances manquantes, ports déjà utilisés, etc.).

La première difficulté technique majeure rencontrée a été liée à la lenteur du conteneur Docker* sur mon poste. Chaque fois que je lançais le projet avec la commande ‘npm run dev’, l’initialisation prenait un temps très long, ce qui ralentissait fortement la productivité et rendait les itérations de développement presque impossibles à court terme. Dans un premier temps, je n’ai pas su identifier précisément l’origine du problème, et il m’était difficile de travailler efficacement.

Ce n’est qu’après plusieurs échanges avec l’équipe, et notamment suite à une refonte partielle de la base du projet effectuée par Simon Lascaud, que la situation s’est améliorée. En effet, l’implémentation progressive de Next.js dans l’architecture du projet a permis d’introduire un système de rendu en continu avec des optimisations liées au rafraîchissement automatique des composants modifiés. Grâce à cette configuration, les modifications apportées au code sont désormais appliquées quasi instantanément, sans nécessiter un redémarrage complet de l’environnement de développement.

Bien que le site n’ait pas été conçu comme une *Single Page Application (SPA)** au sens strict puisque certaines données comme l’historique des commandes ne sont chargées qu’au moment d’accéder aux pages correspondantes il conserve néanmoins une logique de navigation fluide, sans rechargement complet des pages, en exploitant les fonctionnalités de routing client proposées par Next.js.

Rapport de stage chez Izydesk

```
depinfo@PC-depinfo:~/WebstormProjects/izy-order-website$ jobs
depinfo@PC-depinfo:~/WebstormProjects/izy-order-website$ npm run dev

> command-website@0.1.0 dev
> next dev --turbopack

[@sentry/nextjs] WARNING: You are using the Sentry SDK with `next dev --turbo`. The Sentry SDK doesn't yet
urate or incomplete. Production builds without `--turbo` will still fully work. If you are just trying out
veloping locally. Follow this issue for progress on Sentry + Turbopack: https://github.com/getsentry/sentrinment variable)
  ▲ Next.js 15.2.3 (Turbopack)
  - Local:      http://localhost:3000
  - Network:    http://192.168.18.82:3000
  - Environments: .env
  - Experiments (use with caution):
    · clientTraceMetadata

  ✓ Starting...
  ✓ Compiled in 3.5s
  ✓ Ready in 5.6s
```

Figure 3 : Exemple de temps d'exécution avec l'implémentation Next

En parallèle, j'ai dû configurer l'outil Bruno*, utilisé dans l'entreprise pour simuler et tester les appels API*. J'ai importé les collections partagées par l'équipe, puis adapté les requêtes aux paramètres locaux (ports, tokens, routes personnalisées) afin de pouvoir tester les endpoints indispensables : liste des produits, catégories, identification client, ajout d'adresses ou création de commande. Bruno m'a ensuite servi tout au long du projet pour vérifier les retours JSON, identifier les erreurs de communication et ajuster mes intégrations front-end en conséquence.

The screenshot shows the Bruno API documentation interface. On the left, there's a sidebar with a tree view of collections: 'bruno' (selected), 'Collections' (with a dropdown arrow), and several other categories like 'API RESTAURATION', 'MobileApp', etc. Under 'API RESTAURATION', there are sub-collections: 'Addon', 'Authentication', 'AWS', 'Company', 'Customer', 'DeliverySlot', 'Device', 'Integrations', 'Jet', 'Menu', 'MobileApp', 'Company', 'CouponCode', 'Customer', 'DeliverySlot', 'Device', 'Firebase', 'Menu', 'PeriodAndService', 'Product', 'RewardsPoint', 'Sale', and 'Payment'. The main content area is titled 'API RESTAURATION' and contains sections for 'Overview', 'Headers', 'Vars', 'Auth*', 'Script', 'Tests', 'Presets', 'Proxy', 'Client Certificates', and 'Secrets'. The 'Overview' section includes fields for 'Location' (set to '/home/depinfo/WebstormProjects/izy-saas-restauration/assets/bruno/API') and 'Environments' (showing '4 environments configured'). It also lists 'Requests' (179 requests in collection). The right side of the interface has sections for 'Documentation' (welcome message), 'Overview' (high-level overview), 'Best Practices' (list of items), 'Markdown Support' (information about Markdown usage), and a 'Safe Mode' button at the top right.

Rapport de stage chez Izydesk

*Figure 4: Exemple de l'interface de Bruno**

Enfin, j'ai été intégré au Slack de l'entreprise, qui est le principal outil de communication interne dans l'entreprise. C'est via cette plateforme que j'ai échangé avec les développeurs, posé mes premières questions, et suivi les annonces. L'usage de Slack et de Google Meet m'a permis à l'équipe de fonctionner de manière souple, en gardant un lien constant entre les membres. J'ai également profité des points réguliers avec mes encadrants pour m'aligner sur les attentes du projet et résoudre les blocages rencontrés.

3.2 – Refactorisation du menu en composants React*

Une des premières tâches que j'ai eu l'occasion d'effectuer a été d'intervenir sur la page permettant d'afficher un produit configurable (appelé « menu »), afin d'en améliorer la structure et la maintenabilité. Le composant initial était très long et pas décomposé, ce qui rendait sa lecture difficile et limitait la réutilisabilité des éléments. Il m'a donc été demandé de le refactorer en plusieurs composants distincts, suivant les bonnes pratiques de développement React*.

N'ayant jamais travaillé en React* avant ce stage, j'ai dû dans un premier temps me former de manière autonome en consultant la documentation officielle [1] et en effectuant de nombreuses recherches sur les hooks, la gestion des props, le state lifting, ou encore le rendu conditionnel. Ce travail exploratoire m'a permis de mieux comprendre l'approche déclarative de React *et de mettre en pratique le découpage d'un composant complexe en sous-composants spécialisés.

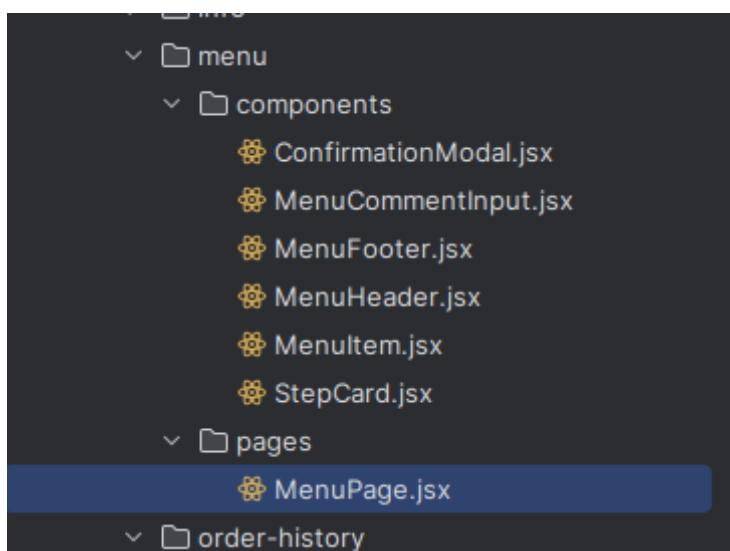


Figure 5 : Nouveau découpage du menu en plusieurs composants

La nouvelle architecture s'organise autour de plusieurs fichiers avec chacun une responsabilité claire :

- **MenuHeader.jsx** gère l'affichage de l'image, du titre, de la description du menu, ainsi que des informations nutritionnelles et allergènes. Il interagit avec le contexte global de l'application pour récupérer les informations de l'entreprise et les catégories.
- **StepCard.jsx** représente une étape de personnalisation du produit (par exemple, choisir un féculent, une sauce, une boisson). Il est chargé d'afficher dynamiquement les options de chaque étape et de gérer l'ouverture/fermeture d'une étape sélectionnée par l'utilisateur.

Rapport de stage chez Izydesk

- **MenuItem.jsx** est utilisé au sein de chaque **StepCard** pour afficher les options ou produits proposés. Il intègre des effets visuels avec Framer Motion, ainsi qu'un affichage conditionnel (quantité, validation, éléments ignorés avec l'attribut skip).
- **MenuFooter.jsx** affiche le récapitulatif du prix total et le bouton pour ajouter le menu au panier. Il est également responsable de la logique de validation (blocage du bouton si certaines étapes ne sont pas complétées).
- **MenuCommentInput.jsx** et **ConfirmationModal.jsx** apportent des fonctionnalités secondaires liées aux commentaires ou à la validation finale du menu.

Ce découpage modulaire m'a permis de mieux organiser le code, de faciliter sa lisibilité, et de rendre chaque partie plus facilement testable et évolutive.

Les données de menu sont récupérées dynamiquement via un appel API* à /get-products-customer-mobile-app, qui fournit la liste des produits disponibles sous forme d'un tableau d'objets. Chaque menu est composé de plusieurs « steps », c'est-à-dire d'étapes de personnalisation, contenant elles-mêmes un tableau de produits ou d'options. Ces données sont ensuite affichées dynamiquement dans StepCard et MenuItem, qui prennent en compte des contraintes comme les limites de sélection (maxItemLimit, isUniqueChoice, etc).

```
,  
{  
  "company": "VNn5VLC4zMvVCNe",  
  "menu": true,  
  "id": 1587,  
  "libelle": "Assiette Poulet",  
  "menuCode": null,  
  "price": "9.55",  
  "priceItc": "10.5",  
  "isActive": true,  
  "isDisplayForPos": false,  
  "isTrending": false,  
  "new": null,  
  "timeSlot": null,  
  "dateSlot": null,  
  "stock": 1,  
  "description": "<p>Brochettes de poulet, riz ou blé, frite:  
/p>",  
  "rewardsPointEarned": 0,  
  "mainProduct": null,  
  "category": {  
    "id": 14,  
    "libelle": "Assiettes"  
  },  
  "steps": [  
    {  
      "rank": 1,  
      "id": 4304,  
      "libelle": "Choisissez votre féculent",  
      "description": "Choisissez-en 1 max.",  
      "isProduct": true,  
      "isOption": false,  
      "isRequired": null,  
      "maxItemLimit": 1,  
      "minItemLimit": 1,  
      "products": [  
        {  
          "id": 3014,  
          "libelle": "Riz",  
          "priceItc": "0.00",  
          "stock": 1,  
          "productIsActive": true,  
          "isActive": true,  
          "isUniqueChoice": null,  
          "skip": false,  
          "isDisplayForPos": true,  
          "timeSlot": null,  
          "category": {  
            "id": 387,  
            "libelle": "Ingrédients et Suppléments"  
          },  
          "ingredients": [],  
          "options": [],  
          "vatRate": {  
            "id": 2,  
            "libelle": "TVA 10%",  
            "value": "10",  
            "rate": 10  
          }  
        }  
      ]  
    }  
  ]  
}
```

Figure 6 : Aperçu de la structure d'un produit avec menu via Bruno*

3.3 – Afficher uniquement les éléments nécessaires

Dans le cadre du développement du site de commande, l'un des enjeux a été de filtrer correctement les produits à afficher à l'utilisateur. En effet, l'appel **/get-products-customer-mobile-app** retourne une liste exhaustive de produits, incluant à la fois les plats destinés à la vente directe et des éléments internes comme des suppléments, des ingrédients ou des composants techniques. les appels API* utilisés tout au long de mon stage, principalement ceux présents dans la section *Mobile App* de notre environnement Bruno*

Parmi ces données, certains produits ne doivent pas être visibles de manière autonome sur l'interface. C'est notamment le cas des produits complémentaires (ex : oignons, salade etc) qui sont liés à une étape de personnalisation d'un plat, mais

```
company : "VNIIQVLC4ZMVVNE",
"menu": false,
"id": 2982,
"libelle": "Oignons",
"rank": null,
"price": "0.45",
"priceTtc": "0.5",
"stock": 1,
"rewardsPointEarned": 0,
"rewardsPointRequired": null,
"isActive": true,
"isDisplayForPos": true,
"isTrending": false,
"new": null,
"productCode": "oignons",
"timeSlot": null,
"dateSlot": null,
"category": {
    "id": 387,
    "libelle": "Ingrédients et Suppléments",
    "subCategoryLibelle": null
},
```

non destinés à être achetés seuls. Ces produits sont identifiables par l'attribut *isDisplayForPos*, qui est défini à *true* lorsqu'ils sont utilisés dans un contexte interne, comme une borne de caisse (*POS – Point of Sale*).

Il est donc nécessaire de filtrer les données côté front-end afin de garantir que seuls les produits pertinents pour la vente directe soient affichés en vente à l'utilisateur.

Figure 7 : Aperçu de l'attribut *isDisplayForPos* Dans Bruno*

J'ai implémenté cette logique de filtrage de manière simple mais efficace dans le code de la page principale du menu. Voici un extrait montrant comment :

```
const filteredProducts = products?.filter(product =>
  product.isDisplayForPos === false &&
  product.libelle.toLowerCase().includes(searchQuery.toLowerCase()) &&
  (!selectedCategory || product.category?.id === selectedCategory)
);

const filteredCategories = categories?.filter(category => category.isDisplayForPos === false);
```

Figure 8 : Extrait du code de la page principale montrant le filtrage des produits et catégories

3.4 – De l'utilisation de hooks personnalisés à l'intégration d'un contexte global

Au début du développement du site, j'ai utilisé des hooks React* pour utiliser les différents calls API* internes proposées par Izydesk. Ces hooks avaient pour but de récupérer des données côté client via des appels asynchrones, tout en gérant l'état de chargement et les erreurs.

Un exemple représentatif de cette approche est le hook `useCompanyDetails`, utilisé pour interroger le backend sur les informations d'une entreprise à partir de son UUID. Voici à quoi ressemblait ce hook :

```
"use client";

import { useState, useEffect } from "react";
import { CompanyAPI } from "@app/API/services/company";

const useCompanyDetails = (UUID) : (...) => { Show usages
  const [data, setData] = useState(initialState: null);
  const [loading : boolean , setLoading] = useState(initialState: true);
  const [error, setError] = useState(initialState: null);

  useEffect( effect: () : void => {
    if (!UUID) return;

    const fetchCompanyDetails = async () : Promise<void> => { Show usages
      try {
        const result = await CompanyAPI.getCompanyDetails(UUID);
        setData(result);
      } catch (err) {
        setError(err);
      } finally {
        setLoading(value: false);
      }
    };
    fetchCompanyDetails();
  }, [deps: [UUID]]);

  return { data, loading, error };
};

export default useCompanyDetails; no usages
```

Figure 9 : Hook React* `useCompanyDetails()`

Ce hook appelait l'endpoint* suivant défini dans notre fichier `endpoints.js`

```
=
  CATEGORIES: "/get-product-categories-customer-mobile-app",
  COMPANY: "/get-company-details-customer-mobile-app",
  ADDONS: "/get-addons-customer-mobile-app",
```

Figure 10 : endpoints `get-company`

Rapport de stage chez Izydesk

Ce call API* renvoie un objet contenant les informations de l'entreprise (nom, logo, couleurs, options, etc.). Ces données étaient utilisées dans plusieurs pages, comme la page d'accueil, le menu, le checkout, etc.

3.4.1 - Limites de cette approche

Même si cette solution fonctionnait, elle présentait plusieurs limites :

- À chaque fois que je voulais accéder aux données de l'entreprise dans un nouveau composant, je devais réutiliser ce hook, ce qui pouvait entraîner plusieurs appels API* inutiles.
- Il fallait souvent faire passer manuellement les données via les props (prop drilling)*, ce qui compliquait la structure des composants.
- Et surtout, cette logique devenait difficile à maintenir dans un projet React* en croissance avec des composants profondément imbriqués.

3.4.2 - Mise en place d'un contexte [2] global

Pour répondre à ces limites, j'ai mis en place un contexte React basé sur un CompanyProvider. Cela m'a permis de centraliser la gestion des données de l'entreprise et de les rendre accessibles à tous les composants sans avoir à les passer en props ou relancer des appels API*.

```
export function CompanyProvider({ children, UUID }) { no usages
  const [state, dispatch] = useReducer(companyReducer, initialState);
  const fetchCompanyDetails = useCallback(callback: async () : Promise<void> => {
    if (!UUID) return;

    try {
      dispatch({ type: 'FETCH_START' });
      const companyData = await CompanyAPI.getCompanyDetails(UUID);
      dispatch({ type: 'FETCH_SUCCESS', payload: companyData });
    } catch (error) {
      dispatch({ type: 'FETCH_ERROR', payload: error.message });
    }
  }, [deps: [UUID]]);
  // Chargement initial
  useEffect(effect: void => {
    if (!state.data) {
      fetchCompanyDetails();
    }
  }, [deps: [UUID]]);

  return (
    <CompanyContext.Provider
      value={{
        data: state.data,
        loading: state.loading,
        error: state.error,
        fetchCompanyDetails
      }}
    >
      {children}
    </CompanyContext.Provider>
  );
}

export function useCompany() { no usages
  const context :null = useContext(CompanyContext);
  if (!context) {
    throw new Error('useCompany must be used within CompanyProvider');
  }
  return context;
}
```

Rapport de stage chez Izydesk

Figure 11 : Exemple d'un context global

Dans cette structure, j'ai utilisé un reducer [6] pour gérer les états loading, data et error de manière centralisée. La fonction fetchCompanyDetails est appelée automatiquement via useEffect dès que le UUID est disponible. Elle effectue l'appel API* une seule fois, puis stocke les données dans le state global du contexte.

Le reducer utilisé dans le contexte sert à centraliser la gestion de l'état lié aux données de l'entreprise. Plutôt que d'utiliser plusieurs appels à useState, le reducer regroupe toutes les données (data, loading, error) dans un seul objet d'état. Ce dernier évolue en fonction des actions qui lui sont envoyées : lorsqu'un appel API* démarre, une action de type FETCH_START active le chargement ; si l'appel réussit, une action FETCH_SUCCESS met à jour les données avec la réponse reçue ; et en cas d'échec, une action FETCH_ERROR permet de stocker l'erreur.

```
const useCompanyDetails = (UUID: ...) => { Show usages
  const [data, setData] = useState(initialState: null);
  const [loading: boolean, setLoading] = useState(initialState: true);
  const [error, setError] = useState(initialState: null);

  useEffect(effect: () : void => {
    if (!UUID) return;

    const fetchCompanyDetails = async () : Promise<void> => { Show us
      try {
        const result = await CompanyAPI.getCompanyDetails(UUID);
        setData(result);
      } catch (err) {
        setError(err);
      } finally {
        setLoading(value: false);
      }
    };
    fetchCompanyDetails();
  }, deps: [UUID]);

  return { data, loading, error };
}
```

Figure 12 : Exemple d'un reducer

3.5 – Personnalisation des pages d’erreur

Dans une logique de finition soignée du site, j’ai également travaillé sur la mise en place de pages d’erreur personnalisées. Ces pages, comme celle illustrée par la Figure 14 ci-dessous, permettent d’informer l’utilisateur lorsqu’une ressource demandée n’est pas disponible (erreur 404) ou lorsqu’un problème serveur survient (erreur 500), tout en maintenant une cohérence graphique avec le reste du site.

L’objectif était d’offrir une expérience utilisateur fluide et esthétique, même en cas d’erreur. Pour cela, j’ai utilisé Tailwind CSS [4] combiné à une architecture de composants modulaires. Le composant ErrorLayout est chargé d’afficher dynamiquement les informations fournies (code, titre, message) tout en stylisant l’arrière-plan avec des cercles animés (générés avec framer-motion) dans les couleurs de l’entreprise. Ces couleurs sont récupérées via le CompanyContext, comme pour le reste du site.

```
j  const router : AppRouterInstance = useRouter();
  const { useSearchParams } = require("next/navigation");
  const searchParams : ReadonlyURLSearchParams = useSearchParams();
  const errorType : string = searchParams.get("error");
  const { data: companyData } = useCompany();
  const primaryColor = companyData?.primaryColor || "#000000";
```

Figure 13 : Récupération des informations d’erreur et entreprises via context [5]

```
default: {
  code: "404",
  title: "Oops ! Page introuvable",
  message:
    "Il semble que vous vous soyez perdu. La page que vous recherchez n'existe pas ou a été déplacée.",
},
```

Figure 14 : Contenu affiché pour la page d’erreur 404

```
<div className="flex flex-col items-center gap-4 w-full">
  <button
    onClick={() : void => router.back()}
    className="px-6 py-3 bg-gray-200 text-gray-800 rounded-xl hover:bg-gray-300 transition-all">
    Retour
  </button>
  <button
    onClick={() : void => router.push( href: "/" )}
    className="px-6 py-3 text-white font-medium rounded-xl transition-colors duration-300 w-full"
    style={{
      backgroundColor: primaryColor,
    }}>
```

Rapport de stage chez Izydesk

Figure 15 : Boutons de navigation permettant de revenir en arrière ou de retourner à l'accueil

3.6 – Affichage responsive des valeurs nutritionnelles et allergènes

Une attention particulière a été portée à l'adaptabilité du site aux différents formats d'écran, notamment pour le composant d'affichage des valeurs nutritionnelles et allergènes, accessibles via un bouton d'information sur chaque produit. À l'origine, ce composant était pensé pour un affichage sur desktop uniquement, avec un simple tooltip positionné au-dessus de la carte produit (voir Figure 16). Cela posait des problèmes majeurs de lisibilité et d'ergonomie sur mobile, où l'espace disponible est plus restreint.

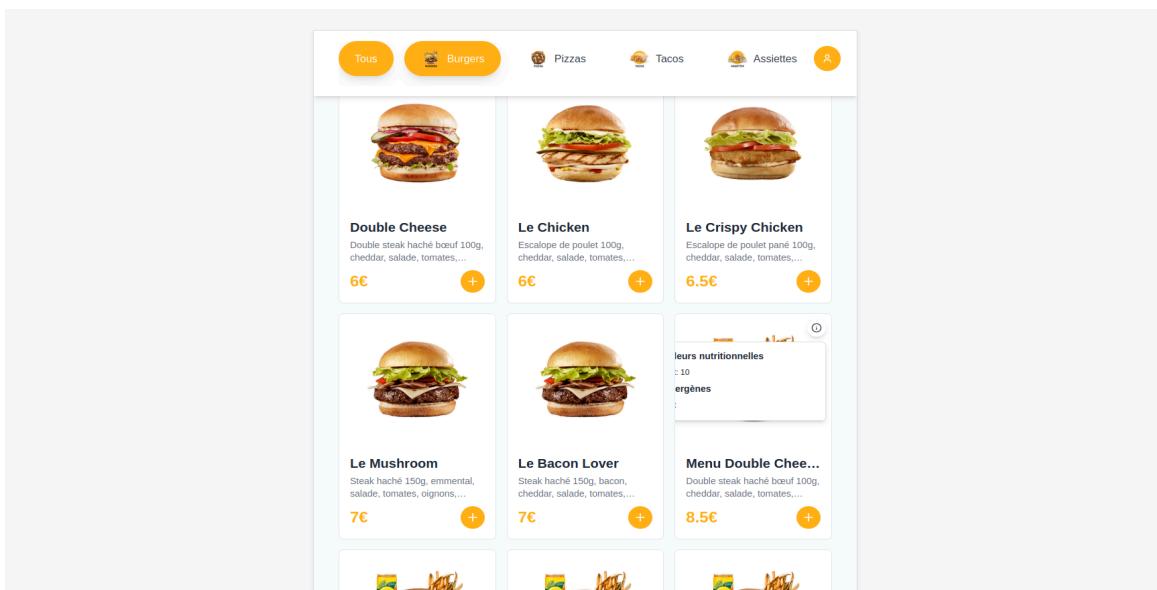


Figure 16: Affichage non responsive pour mobile

Initialement, l'affichage des informations nutritionnelles était géré de manière simple, avec un tooltip statique affiché sous forme de div lorsque l'état isVisible était à true.

Le code se présentait de la manière suivante :

Rapport de stage chez Izydesk

```
return (
  <div className="relative inline-block">
    <div onClick={handleToggle}>{children}</div>
    {isVisible && (
      <div className="absolute bottom-full left-1/2 transform -translate-x-1/2 mb-2 w-[250px] bg-[#f0f0f0] rounded-lg p-4">
        {renderNutritionalValues()}
        {renderAllergens()}
      </div>
    )}
  </div>
);
```

Figure 17 : Fonctionnement non responsive

Pour résoudre cela, j'ai repris complètement le composant. Le premier changement a consisté à détecter dynamiquement si l'utilisateur est sur mobile ou non. Pour cela, j'ai ajouter une variable `isMobile` à l'aide d'un `useEffect()` combiné à un écouteur d'événement `resize` :

```
const SimpleTooltip = ({ children, content, product }) => {
  const [isVisible, setIsVisible] = useState(initialState: false);
  const [isMobile, setIsMobile] = useState(initialState: false);

  useEffect(effect: () => {
    const checkMobileView = () => {
      setIsMobile(value: window.innerWidth < 1340); //640
    };

    // Vérifier initialement
    checkMobileView();

    // Ajouter un écouteur de redimensionnement
    window.addEventListener('resize', checkMobileView);

    // Nettoyer l'écouteur
    return () => window.removeEventListener('resize', checkMobileView);
  }, [deps: []]);
```

Figure 18 : Exemple de code display mobile

Lorsque cette condition est remplie, un panneau est affiché en bas de l'écran avec scroll vertical, fond semi-transparent, et un bouton "Fermer".

```
<div className="relative inline-block">
  {isVisible && product && (
    <>
      <div
        className="fixed inset-0 z-40 bg-black/30"
        onClick={() : void => setIsVisible(value: false)}
      ></div>
      <div
        className="fixed bottom-0 left-0 right-0 z-50 bg-white rounded-t-xl p-4 shadow-2xl"
        style={{
          maxHeight: '80vh',
          overflowY: 'auto'
        }}
      >
        <div className="flex justify-center mb-4">
```

Rapport de stage chez Izydesk

Figure 19 : Exemple d'implémentation responsive

Comme illustré ci-dessous :

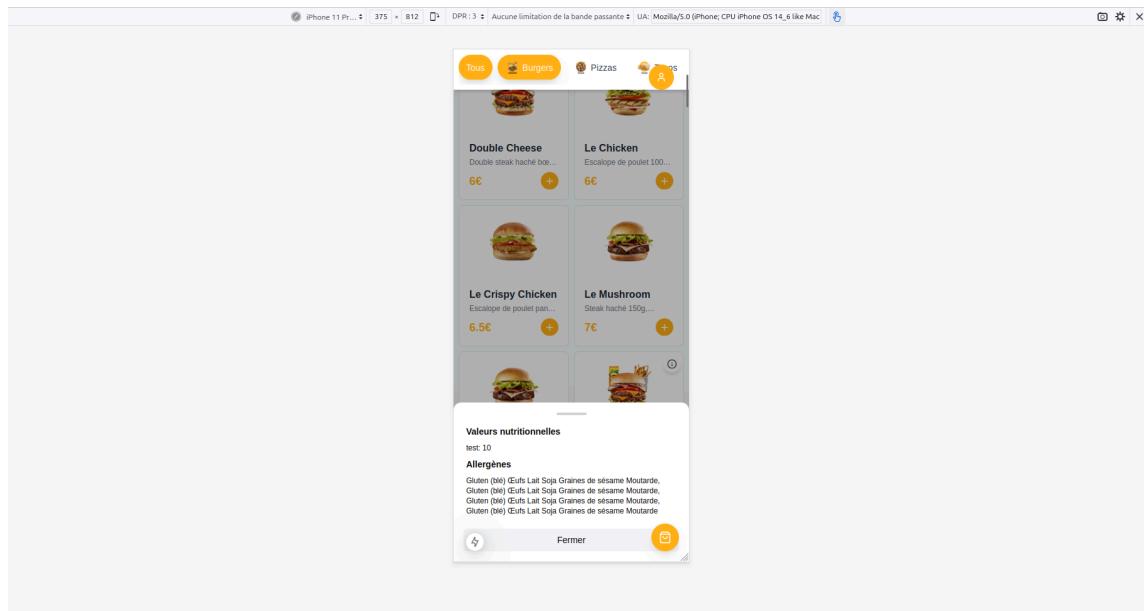


Figure 20 : Affichage responsive pour mobile

3.7 - Conclusion

Pour conclure cette partie, il est important de souligner que ce rapport ne présente qu'un extrait représentatif des nombreuses tâches réalisées tout au long de mon stage. En effet, j'ai travaillé sur une multitude d'éléments, tant au niveau fonctionnel que graphique, que je n'ai pas pu détailler ici de manière exhaustive. Plusieurs autres fonctionnalités ont été implémentées et de nombreux ajustements visuels mineurs ont été effectués marges, tailles, couleurs, interactions, comportements adaptatifs qui, mis bout à bout, ont largement contribué à améliorer l'esthétique globale et l'expérience utilisateur du site.

L'annexe regroupe une partie des notes prises au fil des semaines, comprenant les consignes qui m'ont été transmises, ainsi qu'un exemple de maquette dessinée par Sofian pour illustrer visuellement une de ses attentes. Ces éléments permettent d'avoir un aperçu complémentaire du volume de travail et de la diversité des interventions réalisées, bien que tout n'y figure pas non plus.

Apprentissages Critiques mobilisés :

- 22.01 : Choisir des structures de données complexes adaptées au problème
- 12.01 : Analyser un problème avec méthode (découpage en éléments algorithmiques simples, structure de données...)
- 22.02 : Utiliser des techniques algorithmiques adaptées pour des problèmes complexes (par ex. recherche opérationnelle, méthodes arborescentes, optimisation globale, intelligence artificielle...)
- 31.02 : Faire évoluer une application existante
- 21.03 : Adopter de bonnes pratiques de conception et de programmation
- 31.01 : Choisir et implémenter les architectures adaptées
- 31.03 : Intégrer des solutions dans un environnement de production
- 12.01 : Analyser un problème avec méthode (découpage en éléments algorithmiques simples, structure de données...)
- 13.01 : Identifier les différents composants (matériels et logiciels) d'un système numérique
- 26.04 : Rendre compte de son activité professionnelle

Rapport de stage chez Izydesk

5 – Conclusion

Bilan du travail réalisé

Le stage que j'ai effectué au sein de l'entreprise Izydesk a été une première immersion concrète dans le monde professionnel du développement web. J'ai été intégré dès les premières semaines à un projet réel et structurant pour l'entreprise : la conception et le développement d'un site de commande en ligne pour les clients restaurateurs.

L'objectif initial consistait à reprendre une base de code incomplète et non fonctionnelle, et de la faire évoluer progressivement vers une version opérationnelle et maintenable. Pour cela, j'ai commencé par restructurer l'un des composants les plus importants du projet, à savoir la page de menu, en le divisant en plusieurs composants React réutilisables. J'ai ensuite pris en charge l'intégration progressive de nombreuses fonctionnalités critiques pour l'expérience utilisateur, comme la gestion du panier, la sélection d'addons produits, l'affichage des valeurs nutritionnelles et allergènes, la gestion des adresses, ou encore l'implémentation du système de code promotionnel.

Les différentes fonctionnalités ont été réalisées en respectant les maquettes fournies et les retours réguliers de mes tuteurs techniques. Des améliorations graphiques et ergonomiques ont également été apportées grâce à Tailwind CSS, permettant une interface claire et responsive. Enfin, l'ensemble du travail a été intégré dans l'environnement GitLab de l'entreprise, via un processus rigoureux de merge requests, de validations, et de tests.

Perspectives

Même si le site n'est pas encore totalement finalisé, les bases solides mises en place pendant le stage permettront à l'équipe de poursuivre son développement dans un cadre propre, organisé et scalable. Plusieurs améliorations sont d'ores et déjà prévues à moyen terme, notamment l'intégration avancée des systèmes de fidélité, l'harmonisation des comportements sur mobile, ainsi que l'optimisation des performances du site en production.

Ce stage m'a également permis de mieux cerner la dynamique de projet au sein d'une structure agile, et d'identifier les étapes nécessaires à la mise en œuvre d'une application en conditions réelles. Des pistes ont été évoquées pour que je puisse poursuivre ce projet en alternance l'année prochaine, ce qui témoigne de l'investissement que j'ai pu y apporter, et de la confiance accordée par l'équipe technique.

Bilan des acquis techniques

D'un point de vue technique, ce stage m'a permis d'acquérir et d'approfondir plusieurs compétences fondamentales dans le domaine du développement web. J'ai notamment appris à :

- ✓ Utiliser React de manière modulaire et professionnelle
- ✓ Comprendre et exploiter les capacités de Next.js pour le rendu dynamique
- ✓ Intégrer et tester des appels API* dans une application web avec Bruno*
- ✓ Structurer un projet à l'aide de composants réutilisables et propres
- ✓ Appliquer des styles efficaces et maintenables avec Tailwind CSS
- ✓ Gérer les états complexes d'un panier client et d'un système de commande
- ✓ Mettre en œuvre des pratiques collaboratives via GitLab (branche dev, main, merge request, revue de code)

Ces compétences, apprises en contexte professionnel, viennent renforcer les acquis théoriques obtenus lors de ma formation en BUT Informatique.

Bilan de l'expérience en entreprise

Travailler chez Izydesk m'a permis de découvrir la réalité d'un environnement agile, jeune et innovant. L'organisation du travail est à la fois structurée et souple : j'ai pu participer à des réunions hebdomadaires, échanger librement avec mes encadrants techniques, proposer des solutions, et m'adapter aux retours de l'équipe.

Le fait de travailler sur un projet professionnel en lien direct avec les clients finaux m'a responsabilisé, et m'a appris à soigner à la fois l'expérience utilisateur et la qualité du code. J'ai également pu observer et intégrer la logique de co-construction entre développeurs, dans laquelle les idées circulent, se confrontent, et s'améliorent par des itérations continues.

Bilan de l'expérience personnelle

Ce stage a été pour moi une expérience extrêmement enrichissante, tant sur le plan technique que personnel. J'ai découvert un domaine (le développement web professionnel) qui me motive profondément, et dans lequel je me projette pour la suite de mon parcours. J'ai gagné en autonomie, en rigueur, et en capacité d'adaptation. J'ai aussi pris conscience de l'importance des détails dans une interface, de la fluidité d'un parcours utilisateur, et de la solidité d'une architecture de projet.

L'accueil qui m'a été réservé chez Izydesk, la confiance qu'on m'a accordée, ainsi que les nombreux échanges que j'ai pu avoir avec les développeurs, m'ont donné l'envie de continuer à progresser dans ce métier et de m'investir encore davantage dans des projets d'envergure.

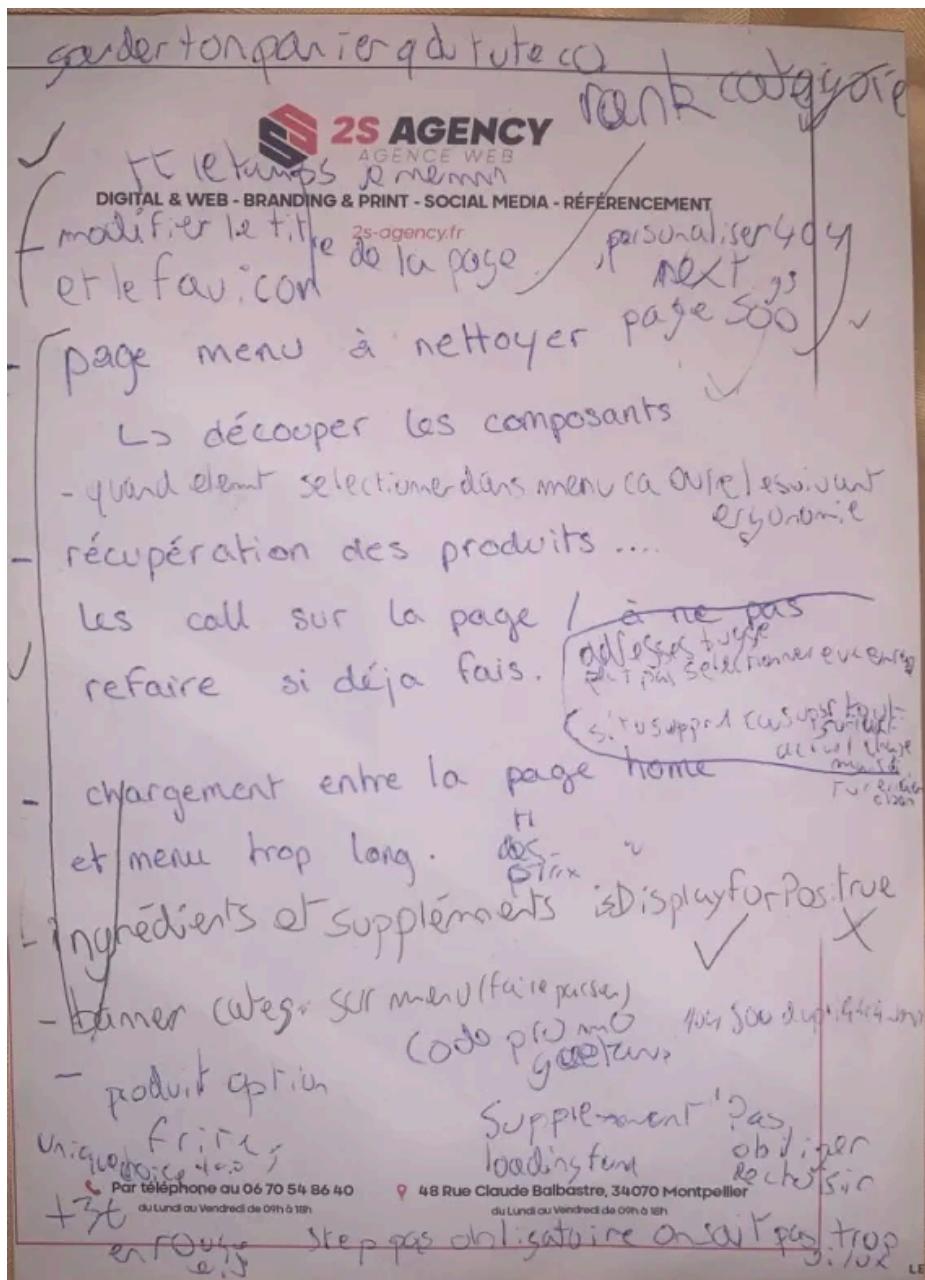
Rapport de stage chez Izydesk

Signatures :

IZYDESK
60 Rue Bernard Grudeau
34080 MONTPELLIER
RCS Montpellier 925 193 781
izydesk.fr

Vu par Sofian BENATIA le : 27/03/2025

6 - Annexes



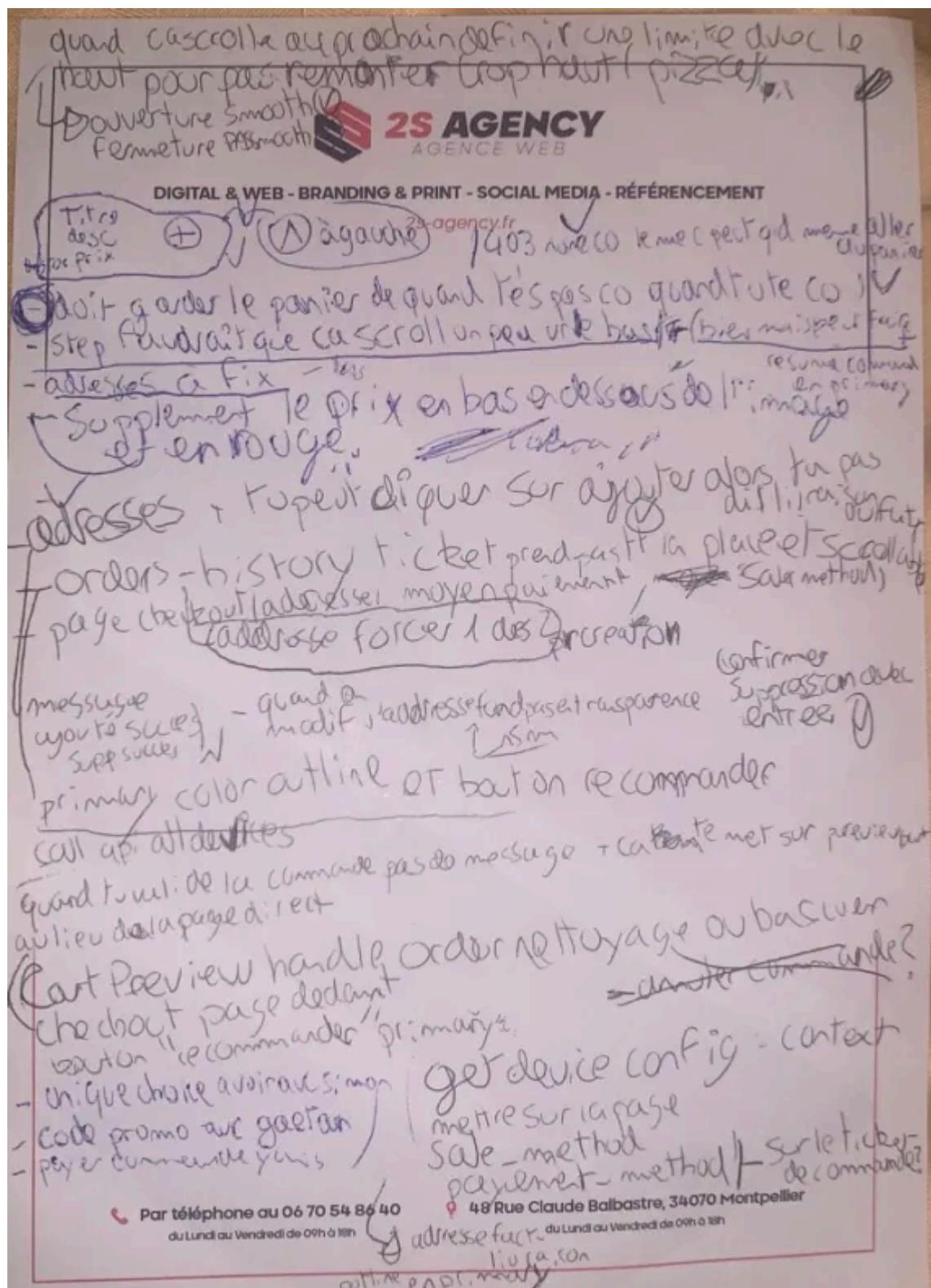
Annexe 1 :

Rapport de stage chez Izydesk

~ faire prix + grand dans
~ combien tout 1082 plats 1090 au
~ dans menu quad LCW complete
~ bouton ajouté au panier pas gradient enlever
~ Tacos ?? → catégorie doit être dans po
~ quand je trie les produits par date
~ Mon connexion mettre msg erreur quand il est correct
~ - order primary color - check out quand cbien
~ order history fix calendar 1-page par NaN
~ adresses ass ~ couleur outline p... Null sur NaN
~ utiliser le log pour accéder à la page checkout
~ 403 bouton décomposition afficher dynamiquement
~ à revue soc mode retard / paiements → Alertes
~ call si période / service ouvert
↳ sinon X de commande
~ modifier dans mes adresses mettre en transparence le fond..
~ titre + photos
~ bouton pr remonter en haut
~ photo trop grosse.
~ traduction Commandes desc bien en bdd
~ titre - checkout quand commandé doit marcher bien
~ trier / faire passer corbeille
~ 2B
~ on peut pas centrer les ancêtres address circuit
~ on doit pas déclencher
~ order history si ticket trop grand qu'on va mesurer
~ bouton change orientation
~ primary color

Annexe 2 :

Rapport de stage chez Izydesk



Annexe 3 :

Rapport de stage chez Izydesk

Annexe 4 :

Rapport de stage chez Izydesk

① Panier quand pas de video → texte nom category null

2S AGENCY
AGENCE WEB

DIGITAL & WEB - BRANDING & PRINT - SOCIAL MEDIA - RÉFÉRENCEMENT
2s-agency.fr

Cave à Fait pour ceux qui vont faire les ingrédients
faire quand il ya plusieurs choix supplément rajouter un + devant pour les goûters

② Détails de la commande (11) enlever badge new / annuler commande address place pas livraison dans order-history allergenes dans menu → logo

✓ rich y qui bouge sur la catég
lundi si on appuie sur ligne / url
sur g des "options" critères ziggy

③ Valider nutri allergene entre description et dans menu que si c'est pas vide
- vente addit ionnelle pas en colonne

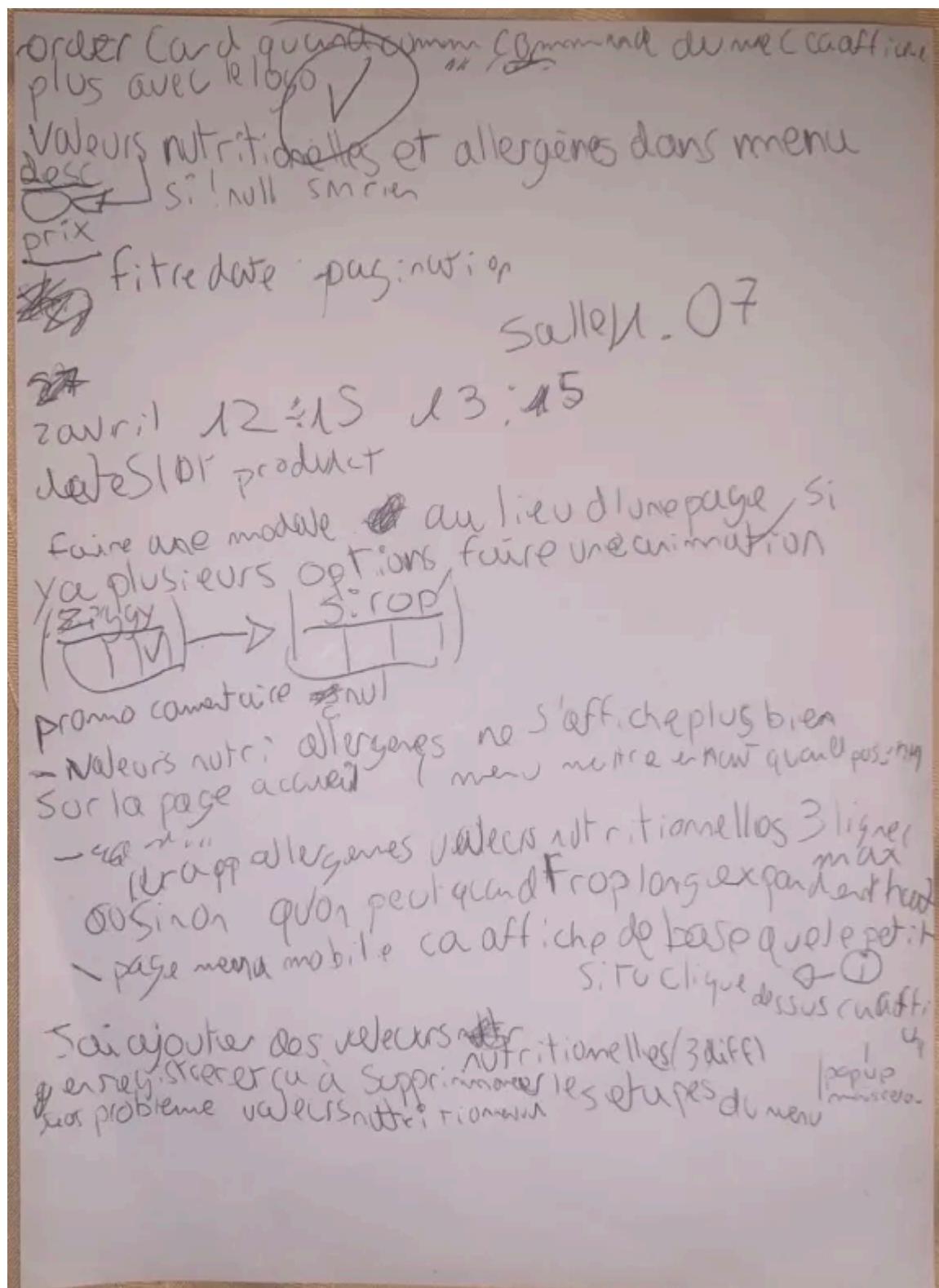
④ add ons généraux que quand mais en grille
- ajouter Panier
- calis avocat concombre chèvre - add ons
- adaptée sur mobile

Par téléphone au 06 70 54 86 40 48 Rue Claude Balbastre, 34070 Montpellier
du Lundi au Vendredi de 09h à 18h du Lundi au Vendredi de 09h à 18h

mes commandes & les dates d'arrivées enlever et j'arrive
⑤ order details Services service enlever

Annexe 5 :

Rapport de stage chez Izydesk



Annexe 6 :

Rapport de stage chez Izydesk

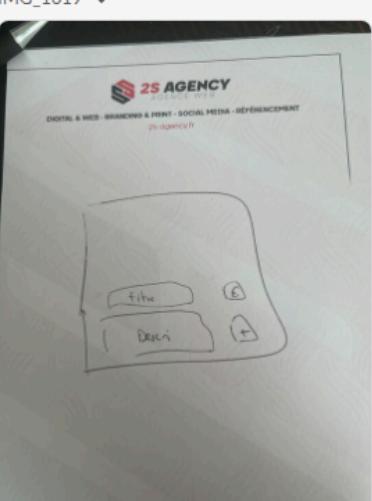
 Sofian BENATIA 16 h 15
<https://meet.google.com/eed-syko-dkk?authuser=1&ijlm=1739546094586&hs=187&adhoc=1>

 [meet.google.com](#)

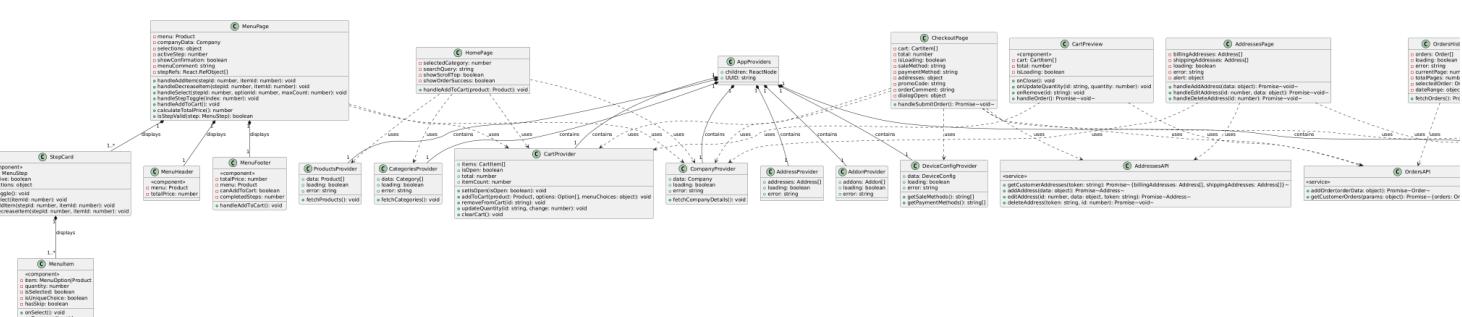
Meet

Real-time meetings by Google. Using your browser, share your video, desktop, and presentations with teammates and customers.

16 h 19 IMG_1619 ▾



Annexe 7 :



Annexe 8 : Diagramme de classe

6.1 - Bibliographie

[1] “Importing and Exporting Components – React,” React Documentation.

<https://fr.react.dev/learn/importing-and-exporting-components>

[2] “Passing Data Deeply with Context – React,” React Documentation.

<https://react.dev/learn/passing-data-deeply-with-context>

[3] “Lucide – Beautiful & consistent icon toolkit,” Lucide. <https://lucide.dev/icons/>

[4] “Learn Tailwind CSS – An Introduction,” Tailwind Labs, 2023.

<https://tailwindcss.com/docs/styling-with-utility-classes>

[5] “Routing – Pages and Layouts,” Next.js Documentation.

<https://nextjs.org/docs/app/building-your-application/routing>

[6] “useReducer – React,” React Documentation.

<https://react.dev/reference/react/useReducer>