

***Parcours informatique Niveau 2***

**RAPPORT DU TRAVAIL PROJET N°1**

Développemnent Web: Mon Miam Miam

25 octobre 2024

Ucac-icam

Zeduc-sp@ce



# **GROUPE N°2**

* **FOKO KENGNE Aurelle Lydienne (Chef projet)**
* **YACOUB BASSI Sounoumougou**
* **MAWAMBA TOWA Maëva**
* **NDTOUNGOU NZAMBI Willem Alex**
* **AZEMKOUO TEZANOU Tony**
* **TSAPI TIOMELA Elie Junior**

**SOMMAIRE**

[**GROUPE N°2** 2](#_Toc180738236)

[**INTRODUCTION** 4](#_Toc180738237)

[**I-** **GESTION DU PROJET** 5](#_Toc180738238)

[**A-** **Planification des ressources** 5](#_Toc180738239)

[**a)** **Ressources matérielles** 5](#_Toc180738240)

[**b)** **Ressources logicielles** 5](#_Toc180738241)

[**c)** **Ressources humaines** 6](#_Toc180738242)

[**B-** **Conception et planification** 6](#_Toc180738243)

[**a-** **Planification des tâches** 6](#_Toc180738244)

[**b-** **Diagrammes** 9](#_Toc180738245)

[**1-** Diagramme de séquence 9](#_Toc180738246)

[**2-** **Diagramme de cas d’utilisation** 10](#_Toc180738247)

[**3-** **Diagramme flux de données** 11](#_Toc180738248)

[**C-** **Schéma conceptuel et relationnel** 12](#_Toc180738249)

[**1.** **Schéma conceptuel** 13](#_Toc180738250)

[**2.** **Schéma relationnelle de données** 13](#_Toc180738251)

[**D-** **Conception de la maquette de l’application web** 14](#_Toc180738252)

[**II-** **DEVELOPPEMENT DE L’APLLICATION** 15](#_Toc180738253)

[**A-** **Choix des langages à utiliser** 15](#_Toc180738254)

[B- **Réalisation de l’application : fonctionnement et fonction clés** 15](#_Toc180738255)

[**CONCLUSION** 31](#_Toc180738256)

[**REFERENCES** 33](#_Toc180738257)

# **INTRODUCTION**

Le restaurant ZeDuc@Space, véritable institution pour les étudiants de l'UCAC-ICAM, est confronté à un succès grandissant. Face à un afflux de commandes et désireux d'offrir une expérience client toujours plus personnalisée, le propriétaire, Mr Miam Miam, a décidé d'innover.

L'objectif est clair : créer une application mobile qui transformera la façon dont les étudiants commandent et consomment au restaurant. En proposant un service de commande en ligne intuitif, des programmes de fidélité attractifs et des fonctionnalités ludiques, l'application vise à renforcer le lien entre le restaurant et sa clientèle, tout en optimisant la gestion des commandes et des opérations...

1. **GESTION DU PROJET**

**A- Planification des ressources**

1. **Ressources matérielles**

Au cours de la conception du code (Réalisation de l’agenda) les ressources matérielles principales étaient l’ordinateur portable et les routeurs. L’ordinateur portable(laptop) qui était notre outil de travail principal et indispensable pour la réalisation du projet et les routeurs utilisés pour nous fournir une connexion internet pour effectuer au mieux nos recherches et favoriser la communication.

1. **Ressources logicielles**

Nous avons eu recours aux différents logiciels tel que :

* Pour la documentation : **Microsoft** **Word, Power Point ;**
* Pour les recherches : **You Tube, Google, Chrome ;**
* Pour le planning : **Jira ;**
* Réalisation du code**: Visual Studiocode, Php Storm ;**

1. **Ressources humaines**

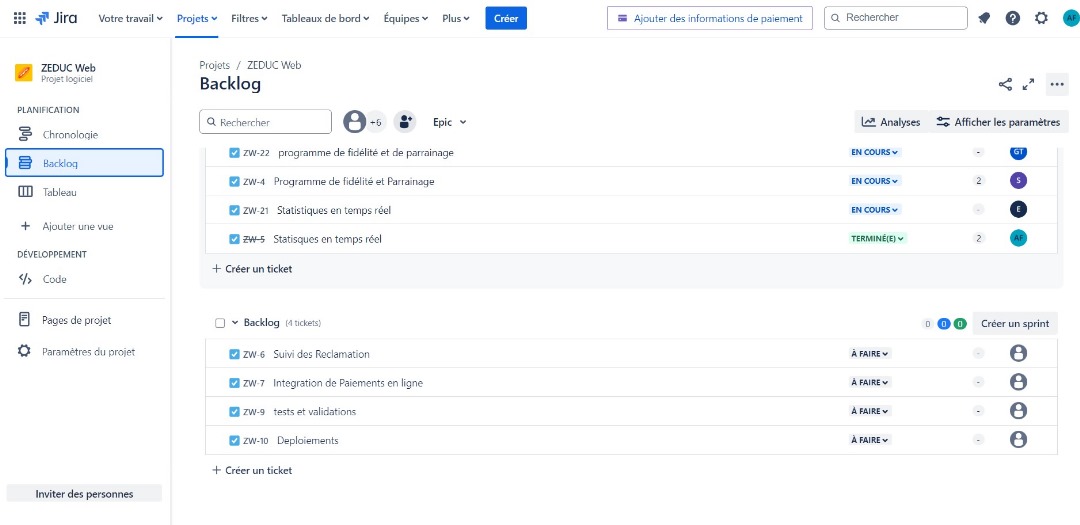
Notre équipe est composée de six (06) membres notamment :

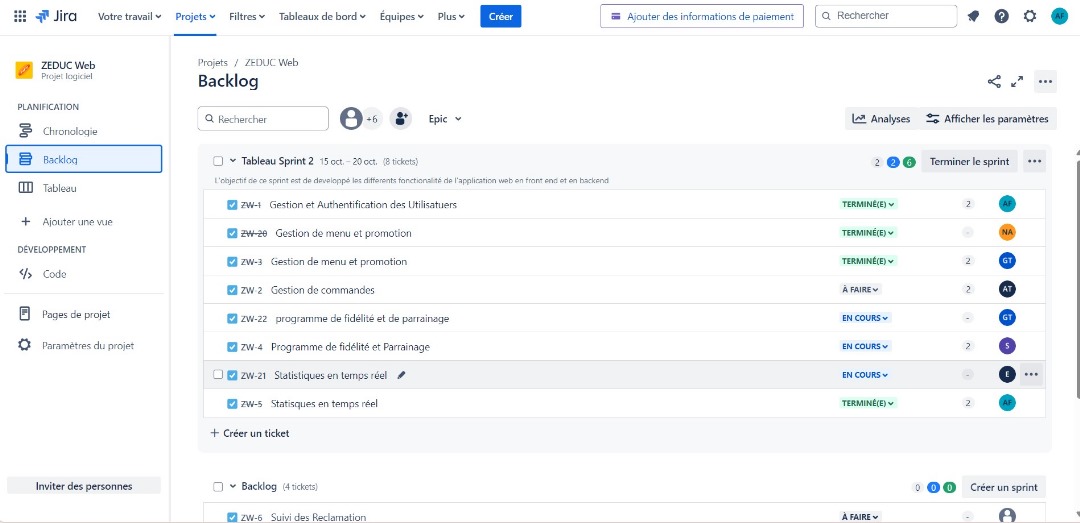
* **FOKO KENGNE Aurelle Lydienne (Chef projet)**
* **YACOUB BASSI Sounoumougou**
* **MAWAMBA TOWA Maëva**
* **NDTOUNGOU NZAMBI Willem Alex**
* **AZEMKOUO TEZANOU Tony**
* **TSAPI TIOMELA Elie Junior**

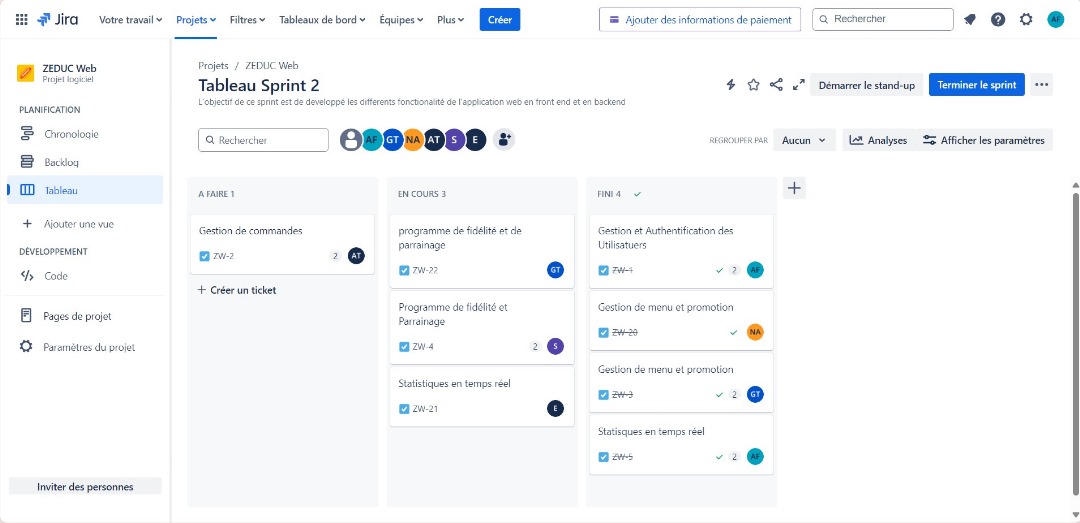
## **B-** **Conception et planification**

1. **Planification des tâches**

Après ouverture du projet, nous avons réaliser Le projet Jira est un projet de gestion de projet et de suivi des problèmes qui est utilisé par de nombreuses entreprises pour gérer efficacement leur travail. Il s'agit d'une plateforme de collaboration qui permet aux équipes de travailler ensemble en temps réel, de suivre les progrès et de résoudre les problèmes de manière centralisée. Dans ce projet, nous allons explorer la façon dont Jira peut être utilisé pour gérer différents types de projets, de la gestion de développement de logiciels à la gestion de projets marketing. Nous allons également examiner les fonctionnalités clés de Jira, telles que la création de tâches, le suivi des problèmes, l'affectation de tâches et la génération de rapports.

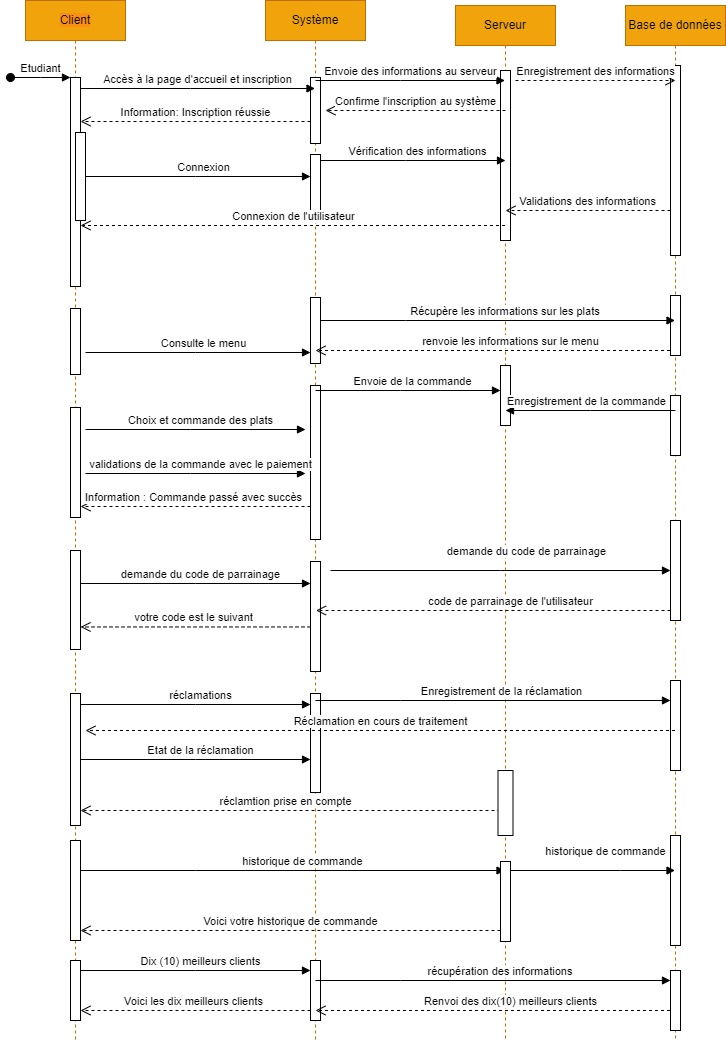
****

****

****

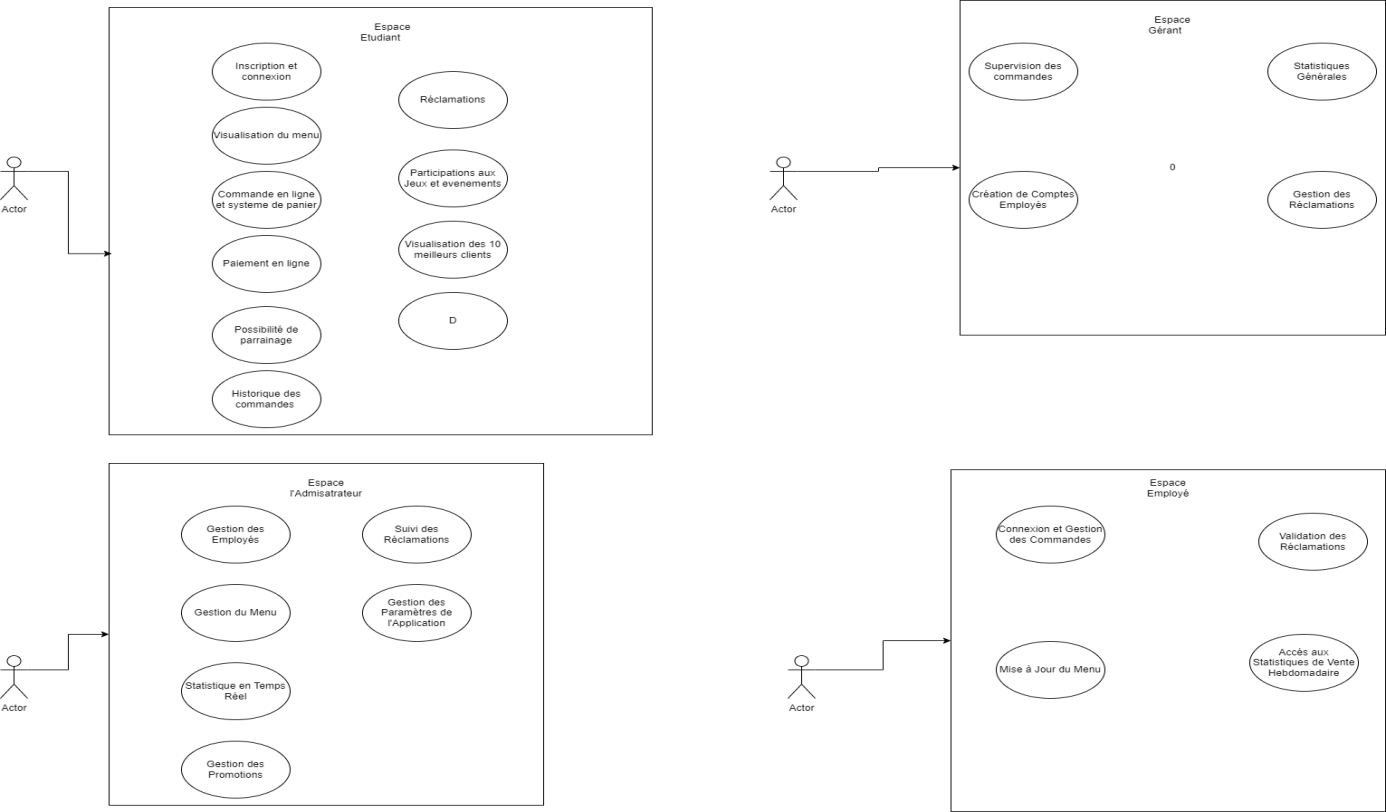
1. **Diagrammes**
2. Diagramme de séquence

Un diagramme de séquence est un type de diagramme utilisé dans la modélisation UML (Unified Modeling Language) pour représenter comment les objets interagissent dans le temps. Nous avons fait le diagramme de séquence pour visualiser et comprendre les interactions temporelles dans le système, facilitant ainsi le développement et la communication entre les équipes.



1. **Diagramme de cas d’utilisation**

Un diagramme de classe est un outil de modélisation utilisé dans la conception de systèmes orientés objet. Nous avons utilisé le diagramme de classe représenter les classes de notre application ainsi que leurs relations comme l’indique l’image ci-après



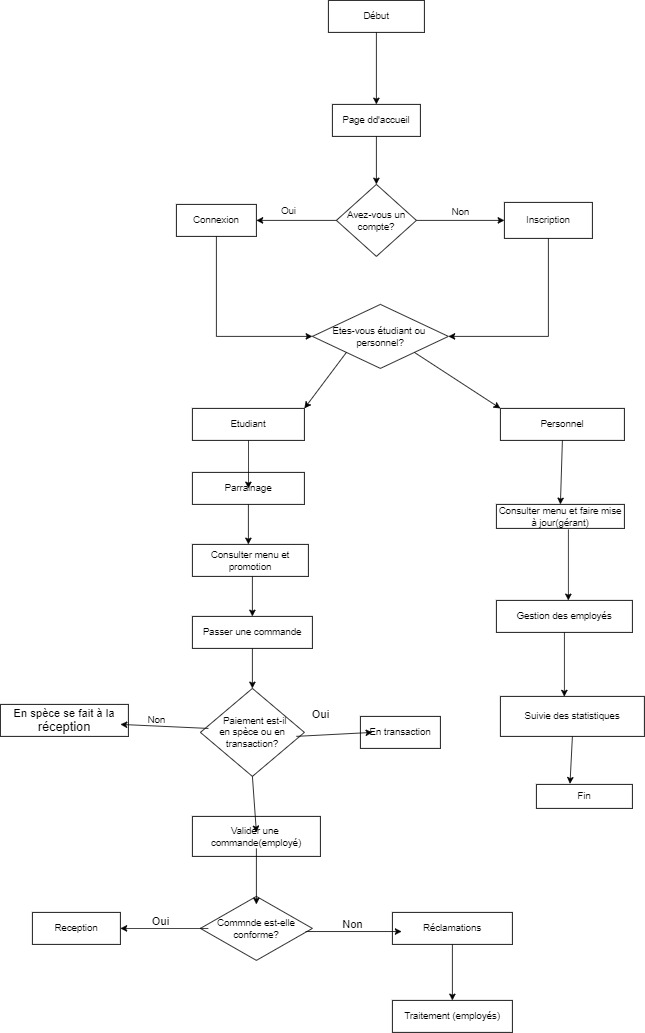
1. **Diagramme flux de données**

Un diagramme de flux de données (DFD) est un outil de modélisation utilisé pour représenter le flux d'informations dans un système. Il montre comment les données se déplacent à travers différents processus, stockages et entités externe. L’Object d’utilisation du diagramme flux de données est de :

-Visualiser la circulation des données dans un système.

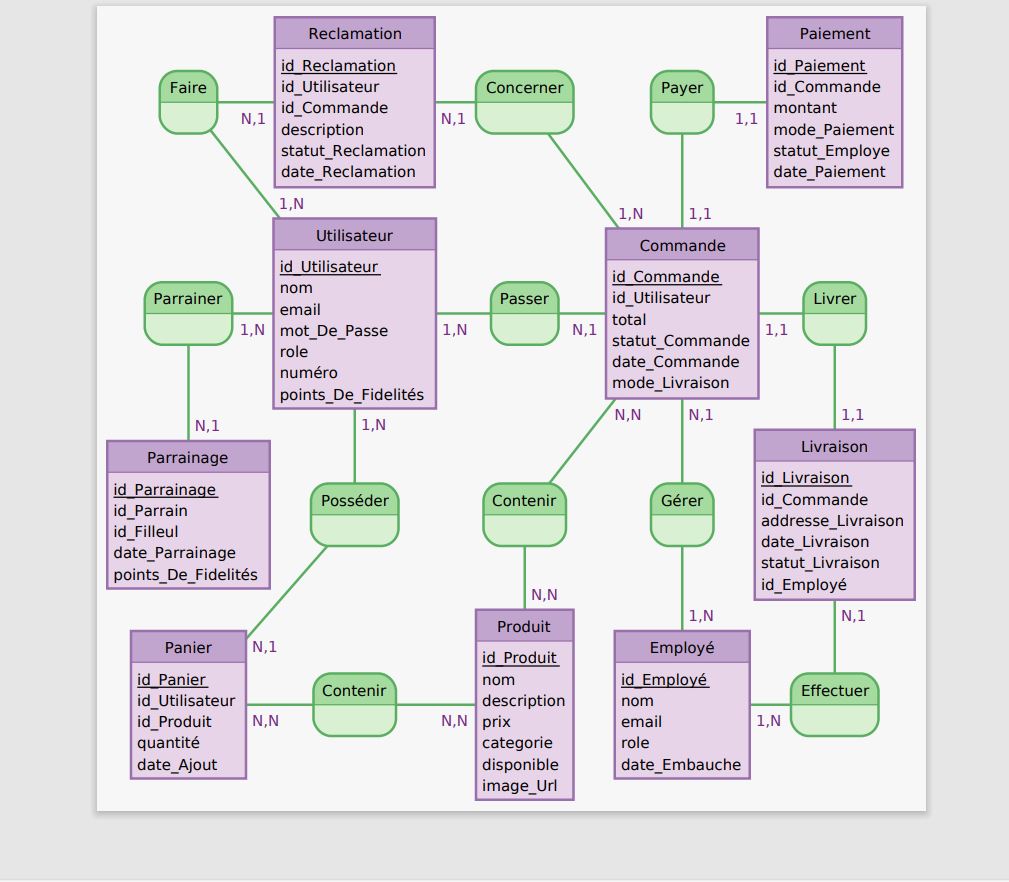
-Identifier les sources et destinations des données.

-Comprendre les processus qui transforment les données.



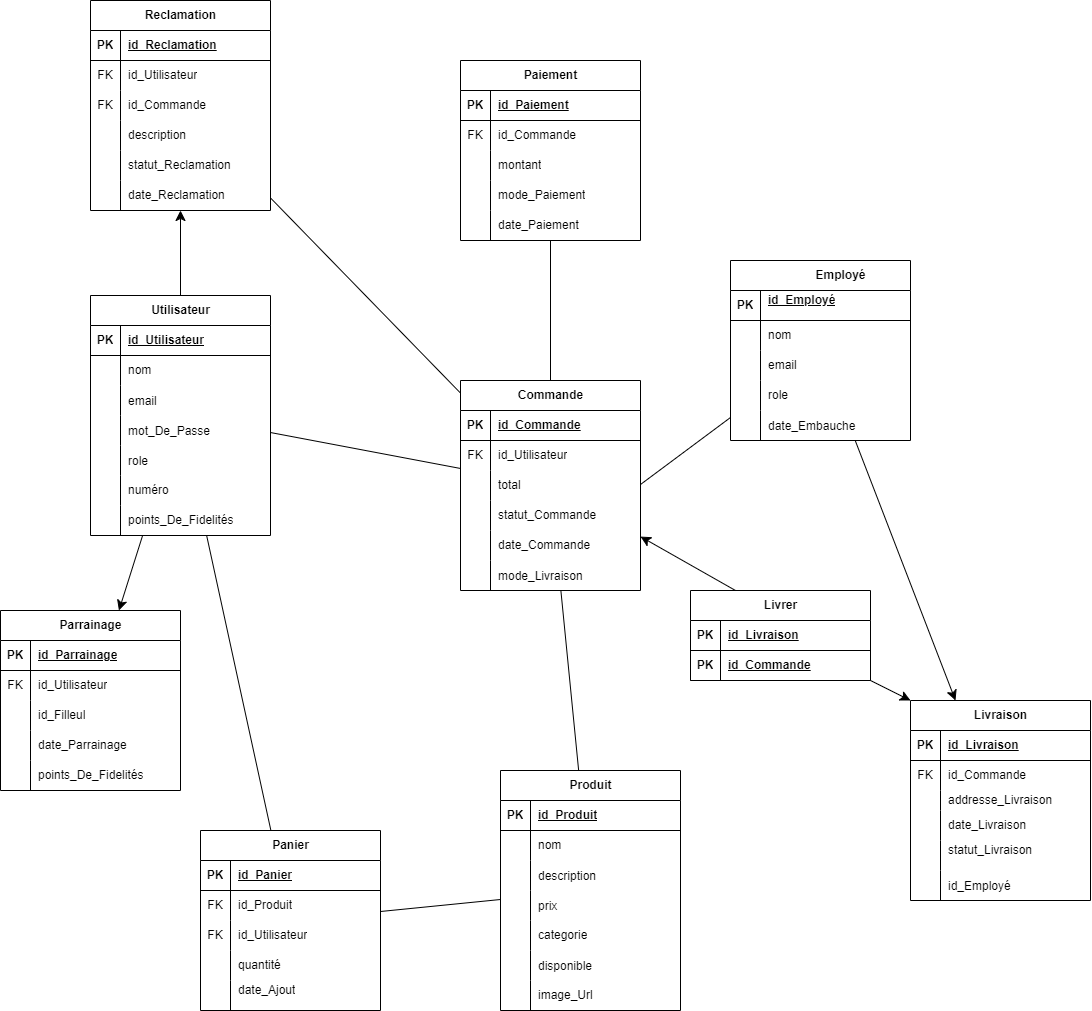
1. **Schéma conceptuel et relationnel**
   * + 1. **Schéma conceptuel**

Pour la réalisation de notre base de données, nous avons préalablement réalisé un schéma conceptuel pour visualiser les différentes tables de la base de données**.**

****

* + - 1. **Schéma relationnelle de données**

Un schéma relationnel de données est une représentation graphique de la structure d'une base de données relationnelle. Il montre les tables de la base de données, les relations entre les tables, les clés primaires et étrangères, les attributs de chaque table et les contraintes d'intégrité de données. Afin d’interpréter les différentes relations entre nos tables, nous avons réalisé un modèle logique de données qui est le suivant :

****

1. **Conception de la maquette de l’application web**

Pour avoir un aperçu de la stucture de nos pages web, nous avons réaliser des maquettes des différentes pages qui vont constituer notre site. La maquette étant réalisé sur Figma et contenant les pages : **Accueil, connexion, inscription, menu, gestion des employés, gestion des commandes, parrainage….** Voici le lien de la maquette Figma pour avoir un aperçu de nos pages**:** [**Figma**](https://www.figma.com/design/fjLSZqoXJtJsOJp7DhUPBQ/ZEDUC-SP%40CE-2?node-id=0-1&node-type=canvas&t=GP9jIYCNVbYeswHs-0)**.**

1. **DEVELOPPEMENT DE L’APLLICATION**
2. **Choix des langages à utiliser**

Pour la réalisation de notre projet zeduc\_web, nous avons eu beaucoup de mal à choisir les différents langages à utiliser; cependant, nous avons utilisé les langages suivant :

* **Le framework Laravel :** pour la réalisation de notre application côté serveur notamment le backend, tout en utilisant le **serveur XAMPP** pour l’utilisation du serveur Apache et de la base de données **Mysql** via l’interface graphique de **Phpmyadmin;**
* **Les langages html, css et les bibliothèques Bootstrap :** pour la réalisation de l’application côté front-end; les bibliothèques Bootstraps nous on permis de rendre notre site responsive et adaptatif sur tous les écrans.
* **Le php :** utilisé du côté backend au niveau de la liason entre le front-end et le backend.

1. **Réalisation de l’application : fonctionnement et fonction clés**

Après avoir réaliser les maquettes du Figma, nous sommes passés à la réalisation du frontend en utilisant la structure du DOM. Le DOM (Document Object Model) est une représentation en mémoire d'un document HTML ou XML. Il permet de manipuler et de modifier dynamiquement la structure et le contenu d'une page web en utilisant des langages de script tels que JavaScript. Le DOM organise les éléments de la page dans une structure hiérarchique en forme d'arbre, avec le document HTML ou XML en tant que nœud racine. Ainsi, voici un aperçu de nos différentes pages :

* **Menu**
* **Sélection des éléments DOM:**
* Le code commence par sélectionner les éléments HTML nécessaires pour interagir avec le JavaScript, tels que les champs de recherche, les menus déroulants, les cartes de produits, la fenêtre modale, le panier et les boutons.
* **Fonctions de filtrage:**
* Les fonctions filterBySearch, filterByCategory et filterByPrice sont utilisées pour filtrer les produits en fonction des critères sélectionnés par l'utilisateur. Elles parcourent les cartes de produits et cachent ou affichent celles qui correspondent aux critères de filtrage.
* **Fenêtre modale:**
* La fonction openPopup ouvre la fenêtre modale en affichant les détails du produit sélectionné.
* La fonction closePopup ferme la fenêtre modale.
* **Panier:**
* Le code utilise un tableau cart pour stocker les produits ajoutés au panier.
* La fonction addToCart ajoute un produit au panier ou met à jour la quantité si le produit existe déjà.
* La fonction displayCartItems affiche le contenu du panier dans un élément HTML.
* La fonction removeFromCart supprime un produit du panier.
* **Événements:**
* Le code attache des écouteurs d'événements aux différents éléments HTML pour gérer les interactions de l'utilisateur :
  + - Lorsque l'utilisateur tape dans la barre de recherche, la fonction filterBySearch est appelée.
    - Lorsque l'utilisateur sélectionne une catégorie, la fonction filterByCategory est appelée.
    - Lorsque l'utilisateur modifie le prix maximum, la fonction filterByPrice est appelée.
    - Lorsque l'utilisateur clique sur une carte de produit, la fonction openPopup est appelée.
    - Lorsque l'utilisateur clique sur le bouton "Ajouter au panier", la fonction addToCart est appelée.
    - Lorsque l'utilisateur clique sur le bouton "Retirer" d'un produit dans le panier, la fonction removeFromCart est appelée.

* **Quantité :**
* Le code permet aux utilisateurs de sélectionner la quantité de chaque produit à ajouter au panier.
  + Les boutons d'incrémentation et de décrémentation mettent à jour la valeur de l'entrée de quantité.
  + Lorsque le bouton "Ajouter au panier" est cliqué, la quantité sélectionnée est utilisée pour ajouter le produit au panier.
* **Paiement**
* **Attachement de l'événement submit:**
  + Le code sélectionne le formulaire HTML à l'aide de document.querySelector('form').
  + Il ajoute un écouteur d'événement submit à ce formulaire. Cet écouteur est une fonction qui sera exécutée lorsque l'utilisateur soumet le formulaire en cliquant sur un bouton "Envoyer".
* **Prévention de la soumission par défaut:**
  + event.preventDefault(); empêche le comportement par défaut du navigateur lors de la soumission d'un formulaire (qui est généralement de recharger la page). Cela nous permet de gérer nous-mêmes la soumission du formulaire.
* **Récupération des valeurs des champs:**
  + Le code récupère les valeurs des champs studentName, contact et location en utilisant document.getElementById et la propriété value. La méthode trim() est utilisée pour supprimer les espaces blancs en début et en fin de chaîne.
* **Vérification des champs obligatoires:**
  + Le code vérifie si les champs studentName, contact et location sont vides. S'il manque une information, un message d'alerte est affiché et la soumission du formulaire est annulée.
* **Vérification de la méthode de paiement:**
  + Le code vérifie si une méthode de paiement a été sélectionnée en utilisant document.querySelector('input[name="paymentMethod"]:checked'). Si aucune option n'est cochée, un message d'alerte est affiché.
* **Affichage d'un message de confirmation:**
  + Si toutes les vérifications sont passées, un message de confirmation est affiché à l'utilisateur, indiquant le nom de l'étudiant et la méthode de paiement sélectionnée.
* **Réclamation Étudiant**
* **Écouteur d'événement submit:**
  + Le code sélectionne le formulaire HTML avec l'ID 'projet' et ajoute un écouteur d'événement submit. Cela signifie que lorsque l'utilisateur clique sur le bouton de soumission du formulaire, la fonction associée à cet écouteur sera exécutée.
* **Empêcher le rechargement de la page:**
  + event.preventDefault(); empêche le comportement par défaut du navigateur lors de la soumission d'un formulaire (qui est de recharger la page). Cela nous permet de gérer nous-mêmes la soumission du formulaire.
* **Récupération des valeurs des champs:**
  + Les valeurs des différents champs du formulaire sont récupérées à l'aide de document.getElementById et stockées dans des variables.
* **Récupération des valeurs des cases à cocher:**
  + Les valeurs des cases à cocher sélectionnées sont récupérées à l'aide de document.querySelectorAll et stockées dans un tableau.
* **Affichage des données dans la console:**
  + Pour des raisons de débogage, les valeurs récupérées sont affichées dans la console du navigateur. Cela permet de vérifier si les données sont correctement récupérées.
* **Affichage d'un message de confirmation:**
  + Un message d'alerte est affiché à l'utilisateur pour confirmer la soumission du formulaire et afficher un récapitulatif des données.
* **Page de réclamation**
* **Sélection du tableau:**
  + let table = document.getElementById("reclamationsTable").getElementsByTagName('tbody')[0]; sélectionne le tableau HTML avec l'ID 'reclamationsTable' et récupère son élément <tbody>. Cela nous permet d'ajouter de nouvelles lignes au tableau.
* **Fonction addReclamation():**
  + Cette fonction est appelée lorsqu'on souhaite ajouter une nouvelle réclamation.
  + Elle récupère les valeurs des différents champs du formulaire d'ajout de réclamation.
  + Elle crée une nouvelle ligne dans le tableau en utilisant table.insertRow().
  + Elle insère les valeurs récupérées dans les cellules de la nouvelle ligne.
  + Elle ajoute des boutons de modification et de suppression à la nouvelle ligne.
  + Enfin, elle réinitialise le formulaire d'ajout de réclamation.
* **Fonction editReclamation(button):**
  + Cette fonction est appelée lorsqu'on clique sur le bouton "Modifier" d'une réclamation.
  + Elle récupère la ligne du tableau contenant la réclamation à modifier à partir du bouton cliqué.
  + Elle remplit les champs du formulaire d'ajout de réclamation avec les valeurs de la ligne sélectionnée.
  + Elle supprime la ligne existante du tableau pour pouvoir la réinsérer avec les nouvelles valeurs modifiées.
* **Fonction deleteReclamation(button):**
  + Cette fonction est appelée lorsqu'on clique sur le bouton "Supprimer" d'une réclamation.
  + Elle récupère la ligne du tableau contenant la réclamation à supprimer à partir du bouton cliqué.
  + Elle supprime la ligne du tableau en utilisant row.parentNode.removeChild(row).
* **Statistiques**

**Gestion de la barre de navigation:**

* Il sélectionne tous les liens (a) à l'intérieur de la classe navbar.
* Il ajoute un écouteur d'événement click sur chaque lien.
* Lorsqu'un lien est cliqué :
  + La classe active est retirée du lien qui était précédemment actif.
  + La classe active est ajoutée au lien qui a été cliqué.
* Cela permet de mettre en évidence le lien de la section active dans la barre de navigation.

**Ajout de lignes dans les tableaux de statistiques :**

* Il attend que le DOM soit chargé (DOMContentLoaded).
* Il sélectionne les éléments <tbody> des deux premiers tableaux de la section statistics-section.
* Il crée deux boutons fictifs :
  + Un bouton "Ajouter une ligne de vente" qui ajoute une nouvelle ligne au premier tableau.
  + Un bouton "Ajouter une ligne de parrainage" qui ajoute une nouvelle ligne au deuxième tableau.
* Lorsqu'on clique sur un bouton d'ajout :
  + Une nouvelle ligne (<tr>) est créée.
  + Cinq cellules (<td>) vides sont ajoutées à la ligne (pour le tableau des ventes).
  + Trois cellules (<td>) vides sont ajoutées à la ligne (pour le tableau des parrainages).
  + La nouvelle ligne est ajoutée au corps du tableau correspondant.
* Il ajoute ensuite les boutons d'ajout au DOM (l'emplacement doit être défini dans le HTML).

**3. Effet de hover sur les icônes des réseaux sociaux:**

* Il sélectionne tous les liens (a) à l'intérieur de la classe socialIcons.
* Il ajoute deux écouteurs d'événement sur chaque icône :
  + mouseenter: Lorsque la souris survole l'icône, elle est agrandie à 120% de sa taille d'origine avec une transition de 0.2 seconde.
  + mouseleave: Lorsque la souris quitte l'icône, elle revient à sa taille d'origine.
* **Connexion**
* **Écouteur d'événement submit:**
  + Le code sélectionne le formulaire avec l'ID loginForm et ajoute un écouteur d'événement submit. Cela signifie que lorsque l'utilisateur soumet le formulaire, la fonction associée à cet écouteur sera exécutée.
* **Empêcher la soumission par défaut:**
  + event.preventDefault(); empêche le comportement par défaut du navigateur lors de la soumission d'un formulaire (qui est de recharger la page).
* **Récupération des valeurs des champs:**
  + Les valeurs des champs email et password sont récupérées à l'aide de document.getElementById et stockées dans des variables.
* **Requête HTTP POST:**
  + fetch('/connexion', ...) envoie une requête HTTP POST à l'URL /connexion (à adapter selon votre configuration).
  + Les en-têtes de la requête incluent :
    - X-CSRF-TOKEN: Ce token est utilisé pour prévenir les attaques CSRF. Sa valeur est récupérée à partir d'un élément méta dans le HTML.
    - Content-Type: Indique que le corps de la requête est au format application/x-www-form-urlencoded.
  + Le corps de la requête contient les valeurs de email et password.
* **Gestion de la réponse:**
  + Si la requête réussit (status code 2xx), la fonction then est appelée.
    - Si la connexion est réussie, l'utilisateur est redirigé vers le tableau de bord.
  + Si la requête échoue (status code différent de 2xx), la fonction then est également appelée.
    - Le contenu de la réponse est analysé en JSON pour récupérer un éventuel message d'erreur.
    - Le message d'erreur est affiché à l'utilisateur via une boîte d'alerte.
* **Gestion des erreurs:**
  + La clause catch est utilisée pour gérer les erreurs qui pourraient survenir lors de la requête ou de l'analyse de la réponse.
* **Parrainage**

**Écouteur d'événement DOMContentLoaded:**

* Le code attend que le DOM (Document Object Model) soit entièrement chargé avant d'exécuter les instructions. Cela garantit que tous les éléments HTML sont prêts à être manipulés.

**Lien de parrainage dynamique:**

* Un lien de parrainage est défini avec une valeur fixe pour l'exemple. En pratique, ce lien serait dynamique et généré côté serveur en fonction de l'utilisateur connecté.

**Bouton "Partager mon lien":**

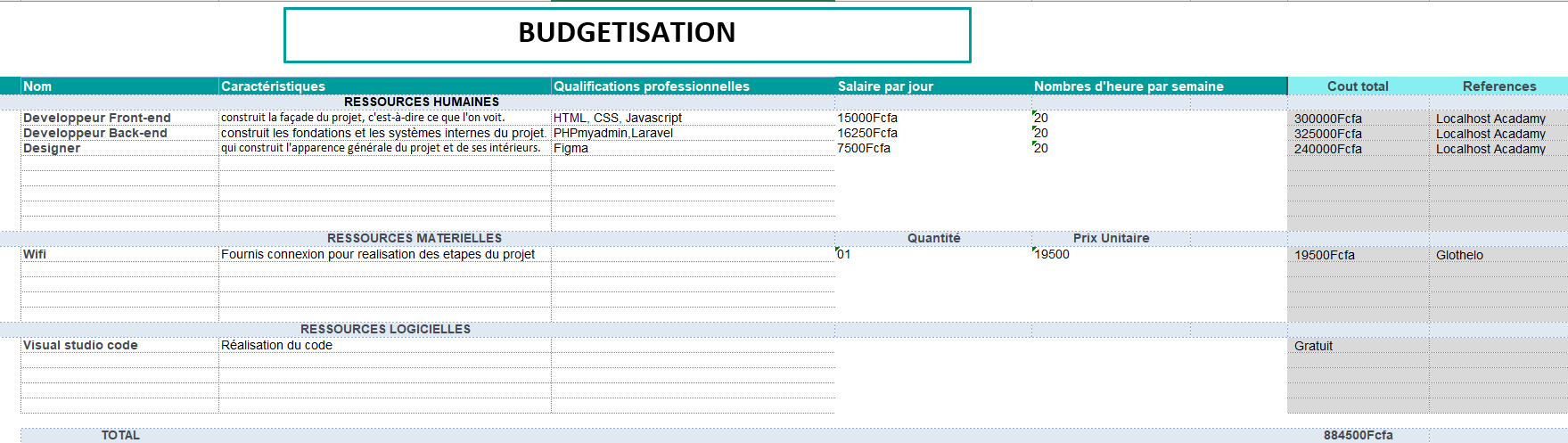
* Lorsque ce bouton est cliqué :
  + Le contenu du lien de parrainage est copié dans le presse-papiers de l'utilisateur à l'aide de navigator.clipboard.writeText().
  + Un message de confirmation s'affiche pour indiquer que le lien a bien été copié.
  + En cas d'erreur, un message est affiché dans la console.

**Points de fidélité:**

* Une variable pointsFidelite est définie avec une valeur fixe pour l'exemple. En pratique, cette valeur serait récupérée depuis le serveur.
* Lorsqu'un utilisateur clique sur le deuxième bouton, une boîte d'alerte s'affiche pour indiquer le nombre de points de fidélité.
* **Inscription**
* **Sélection du formulaire et fonction de validation:**
  + Le code sélectionne le formulaire d'inscription par son ID.
  + La fonction validateForm() vérifie la validité des champs (non vide, format email correct, longueur du mot de passe, etc.).
* **Événement de soumission du formulaire:**
  + Lorsque l'utilisateur soumet le formulaire, la fonction validateForm() est appelée.
  + Si la validation réussit, une requête POST est envoyée au serveur Laravel à l'aide de fetch().
  + Les données du formulaire sont envoyées au format JSON dans le corps de la requête.
  + Le token CSRF est inclus dans les en-têtes pour prévenir les attaques CSRF.
* **Gestion de la réponse du serveur:**
  + Si la requête réussit (code de statut 2xx), le code extrait les données de la réponse JSON et affiche un message de succès.
  + Si la requête échoue (code de statut différent de 2xx), le code extrait les erreurs de validation du serveur et les affiche à l'utilisateur.
* **Home page**
* **Chargement initial et variables:**
  + Le code attend que le DOM soit chargé (événement DOMContentLoaded).
  + Il récupère les éléments HTML nécessaires :
    - cartCount: élément affichant le nombre d'articles dans le panier.
    - totalPrice: élément affichant le prix total du panier.
    - cartItems: élément contenant la liste des articles du panier.
  + Il initialise le panier en essayant de le récupérer du localStorage. Si aucun panier n'existe encore, un tableau vide est initialisé.
* **Fonctions de mise à jour**
  + updateCartCount(): calcule et affiche le nombre total d'articles dans le panier.
  + updateTotalPrice(): calcule et affiche le prix total du panier en additionnant les prix des articles multipliés par leurs quantités respectives.
  + renderCartItems(): affiche la liste des articles du panier dans l'élément cartItems. Il crée des éléments HTML dynamiquement pour chaque article, affichant le nom, le prix, la quantité et des boutons pour modifier ou supprimer l'article.
* **Gestion des articles du panier**
  + saveCart(): sauvegarde le contenu du panier dans le localStorage au format JSON.
  + **Ajout d'un produit au panier**:
    - Gère le clic sur les boutons "Ajouter au panier".
    - Récupère l'ID, le nom et le prix du produit à partir des attributs data-\* du bouton.
    - Vérifie si le produit existe déjà dans le panier.
    - Si oui, on incrémente sa quantité.
    - Sinon, on ajoute un nouvel objet au tableau cart avec les informations du produit et une quantité initiale de 1.
    - Sauvegarde le panier mis à jour et réaffiche la liste des articles et les totaux.
  + **Modification de la quantité**:
    - Gère le clic sur les boutons "+" et "-".
    - Récupère l'index de l'article dans le tableau cart à partir des attributs data-index des boutons.
    - Incrémente ou décrémente la quantité de l'article en fonction du bouton cliqué.
    - Si la quantité atteint 0, l'article est supprimé du panier à l'aide de splice.
    - Sauvegarde le panier mis à jour et réaffiche la liste des articles et les totaux.
  + **Suppression d'un article**:
    - Gère le clic sur le bouton "Supprimer".
    - Récupère l'index de l'article dans le tableau cart à partir de l'attribut data-index du bouton.
    - Supprime l'article du tableau cart à l'aide de splice.
    - Sauvegarde le panier mis à jour et réaffiche la liste des articles et les totaux.
* **Affichage du panier**
  + Gère le clic sur le bouton "Voir le panier".
  + Affiche ou masque le panneau du panier en modifiant la propriété display du CSS.
* **Panneau du panier (seconde partie du code)**
  + Gère le clic sur le bouton "Voir le panier".
  + Empêche le comportement par défaut du bouton (rechargement de la page).
  + Ajoute la classe "open" au panneau du panier pour l'afficher.
  + Gère le clic sur le bouton "Fermer le panier".
  + Retire la classe "open" du panneau du panier pour le masquer.
* **Connexion profil**
* **Écouteur d'événement DOMContentLoaded:**
  + Le code attend que le DOM soit entièrement chargé avant d'exécuter les instructions.
* **Sélection des éléments:**
  + Les éléments HTML nécessaires (bouton de modification, formulaire, bouton de sauvegarde, etc.) sont sélectionnés à l'aide de querySelector.
* **Masquage initial du formulaire:**
  + Le formulaire de profil est initialement masqué.
* **Basculement de la visibilité du formulaire:**
  + Lorsque l'utilisateur clique sur le bouton "Modifier profil", la classe visible est ajoutée ou supprimée du formulaire pour le montrer ou le cacher.
  + Le texte du bouton est également mis à jour en fonction de l'état du formulaire.
* **Validation du formulaire:**
  + Lors de la soumission du formulaire, les données sont récupérées et validées.
  + Une validation basique est effectuée pour vérifier que tous les champs sont remplis et que l'adresse email et le numéro de téléphone ont un format valide.
  + Si la validation réussit, un message de confirmation est affiché (simulant l'enregistrement des données).
* **Fonctions de validation:**
  + Les fonctions validateEmail et validatePhone utilisent des expressions régulières pour vérifier la validité des formats d'email et de numéro de téléphone.
* **Modification de l'avatar:**
  + Un gestionnaire d'événement est attaché à l'avatar pour simuler une fonctionnalité de modification d'avatar.
* **Gestions des menus**
* **Fonctions de gestion des commandes:**
  + deleteOrder(event) : Supprime la ligne de commande correspondante et affiche un toast de confirmation.
  + editOrder(event) : (non implémentée) Probablement prévue pour ouvrir une fenêtre d'édition.
  + createOrder(): Gère la soumission du formulaire de création de commande, ajoute une nouvelle ligne au tableau et affiche un toast de confirmation.
  + handlePagination(event) : (non implémentée) Gère les clics sur les boutons de pagination (à implémenter pour une fonctionnalité de pagination).
  + showToast(message) : Affiche un toast Bootstrap avec le message passé en paramètre.
* **Fonction de recherche améliorée:**
  + searchOrders(): Filtre les lignes du tableau de commandes en fonction d'un terme de recherche et d'une option de filtrage (identifiant de commande ou description).
* **Fonction d'édition des commandes:**
  + showModificationOptions(event): Crée une fenêtre pop-up pour modifier les détails d'une commande (prix, quantité, description).
  + Ajout d'écouteurs d'événements aux boutons "Sauvegarder" et "Annuler" de la fenêtre pop-up pour enregistrer les modifications ou fermer la fenêtre.
* **Association des événements aux éléments:**
  + Ajoute des écouteurs d'événements aux boutons "Supprimer" et "Modifier" (icône crayon) existants dans le tableau des commandes.
  + Ajoute un écouteur d'événement au champ de recherche pour déclencher la fonction searchOrders à chaque saisie de caractère.
  + Ajoute un écouteur d'événement au bouton "Créer une commande" pour ouvrir la modale de création.
  + Ajoute un écouteur d'événement au bouton "Ajouter commande" (à l'intérieur de la modale) pour soumettre le formulaire de création et déclencher la fonction createOrder.
  + Ajoute des écouteurs d'événements aux boutons de pagination (à implémenter pour la gestion de la pagination).

1. **BUDGET**

Le budget représente une composante essentielle de la gestion de tout projet, permettant de planifier, contrôler et optimiser l’ensemble des ressources financières nécessaires à la réalisation des objectifs. Dans le cadre de notre projet, le budget est conçu pour couvrir tous les aspects, depuis le développement technique jusqu’à la maintenance, en passant par la mise en œuvre des fonctionnalités et le déploiement de la solution. Dans cet optique, notre budget est le suivant :



**CONCLUSION**

Le développement d'une application mobile pour ZeDuc@Space représente une opportunité unique de renforcer la position du restaurant sur le marché et de répondre aux attentes d'une clientèle jeune et connectée. En offrant un service de commande en ligne convivial, des fonctionnalités personnalisées et une expérience utilisateur fluide, cette application permettra à ZeDuc@Space de se démarquer de la concurrence et de consolider sa place de référence auprès des étudiants.

# **REFERENCES**

* Lien vers la maquette Figma : [Figma](https://www.figma.com/design/fjLSZqoXJtJsOJp7DhUPBQ/ZEDUC-SP%40CE-2?node-id=0-1&node-type=canvas&t=GP9jIYCNVbYeswHs-0)
* Lien vers notre dépôt github : [Foko860/ZEDUC\_WEB (github.com)](https://github.com/Foko860/ZEDUC_WEB)
* Lien vers la gestion de notre projet dans Jira : <https://2028-team-d1jsrg3v.atlassian.net/jira/software/projects/ZW/boards/3/reports/burnup>
* Ressources fournies :
* <https://marcdezordo.me/les-differences-entre-zoning-wireframe-mockup-et-prototype/>
* http://php.net/manual/fr/
* https://validator.w3.org/
* https://jigsaw.w3.org/css-validator/
* https://jquery.com/
* https://datatables.net/
* https://nodejs.org/en/
* https://www.frugalprototype.com/developpez-propre-api-node-js-express/
* <https://www.getpostman.com/>
* <https://www.atlassian.com/fr/software/jira>
* <https://grafikart.fr/formations/laravel>
* <https://www.youtube.com/watch?v=oGPIks7Ggh8&list=PLDc9bt_00KcJvM4-KJF8OPadbOod_G7Wf>
* <https://www.subdelirium.com/generateur-de-mentions-legales/>
* <https://www.subdelirium.com/generateur-de-mentions-legales/>
* Ressources personnelles :
  + <https://www.youtube.com/watch?v=opugvYy-zhI&t=14s>
  + [Créez des sites web responsives avec Bootstrap 5 - OpenClassrooms](https://openclassrooms.com/fr/courses/7542506-creez-des-sites-web-responsives-avec-bootstrap-5)
  + [Adoptez une architecture MVC en PHP - OpenClassrooms](https://openclassrooms.com/fr/courses/4670706-adoptez-une-architecture-mvc-en-php)