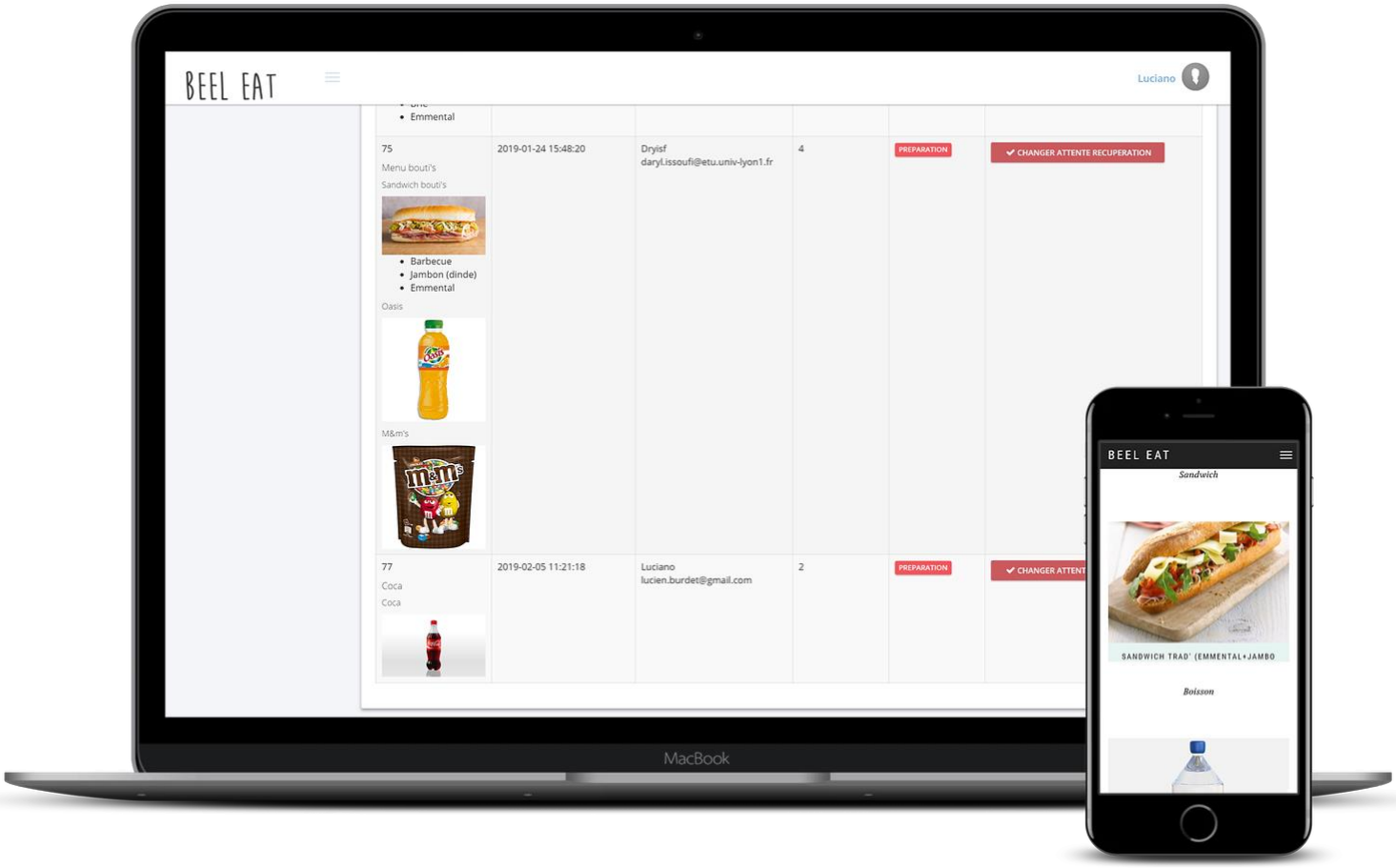


BEEL EAT

GET IT ON Google Play

BeelEAT

HTTPS://BEELEAT.LUCIEN-BRD.COM/

| Rapport de projet tutoré |

Emilio Maldonado
Berson Castillo
Etienne Dahoumane
Lucien Burdet

Tutrice : Aude Joubert
IUT Lyon 1
2019

Table des matières

1. Préface

BeelEAT permet de gérer tout un restaurant. D'une part un côté administrateur/cuisine permet :

- D'automatiser certaines tâches comme la gestion des stocks.
- De gérer les commandes via l'envoi automatique de notifications aux clients sur le statut de leur commande.
- De gérer les menus, produits, ingrédients...

D'autre part, le côté client du site permet de passer des commandes en ligne.

BeelEAT est pour l'instant utilisé par le bureau des élèves de l'IUT génie civil de l'université Lyon 1 pour leur cafétéria. Ce site a su conquérir leur cœur grâce à son interface simple et efficace. De plus, l'envoi de notifications aux clients est un réel avantage et l'application Android BeelEAT est une révolution.

Les technologies utilisées sont le PHP, MYSQL, HTML/CSS, Javascript. Nous avons utilisé le Framework Bootstrap. L'architecture de notre projet est le MVC, modèle vue contrôleur.

Nous gérons notre projet et les versions de notre projet à l'aide de git, voici le lien :

https://github.com/lucianoBrd/BeelEAT

Le site est disponible à cette adresse :

https://beeleat.lucien-brd.com

L'application est disponible sur le Play Store sous le nom de BeelEAT.

2. Les outils utilisés

Afin de mener à bien notre projet, nous avons choisi de le développer une application web. En effet, c'est une plateforme accessible par tous et très populaire.

1) LANGAGE DE PROGRAMMATION

Nous allons coder notre projet en PHP, html, css, javascript et nous aurons aussi besoin d'une base de données.

Pourquoi le PHP ?

C'est un code qui est exécuté sur le serveur directement, ainsi nous pourrons afficher au client des pages web dynamiques. De plus, il est facile de relier une base de données à notre application web en PHP. Donc le PHP présente tous les avantages que nous cherchions :

- Sécurité, le client n'aura aucun moyen d'avoir accès au code PHP
- Polyvalence

Html et Css sont indispensables, ils vont nous permettre de mettre en forme nos pages web et de les structurer.

Javascript va nous permettre de rendre nos pages web plus interactives.

Enfin, une base de données est obligatoire car elle va nous permettre de stocker et manipuler toutes les données du projet (par exemple les utilisateurs).

2) LOGICIEL DE DEVELOPPEMENT

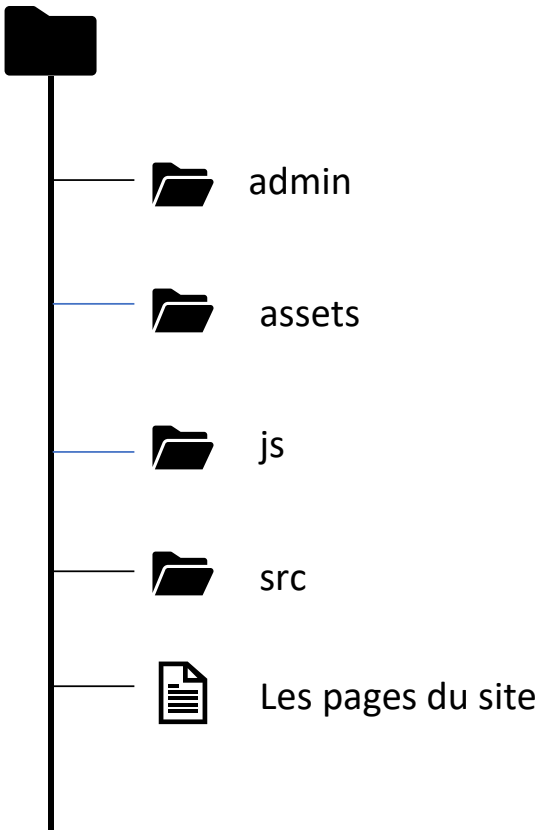
Pour développer notre code nous utilisons un éditeur de texte. Nous avons choisi en particulier atome car il est très simple d'utilisation et puissant. De plus, nous pouvons relier notre git directement depuis atome ce qui est un avantage. En effet, nous utilisons gitLab afin de gérer les différentes versions de notre projet. Git permet un travail en équipe plus optimal.

3. Architecture

Cette partie sera dédié à l'architecture du site. Nous commencerons par exposer l'architecture que nous avions dans les prémices du site, ensuite nous expliquerons les raisons de notre passage au modèle MVC, nous détaillerons comment ce passage s'est fait, enfin nous décrirons notre architecture MVC actuelle, d'une certaine manière nous vous présenterons notre propre « Framework ».

3) ARCHITECTURE DE DEPART

N'ayant pas encore eu le moindre cours de PHP au début du développement notre auto-formation nous permettait d'implémenter correctement des fonctionnalités mais pas de comprendre l'importance de l'architecture d'un site, celle-ci était alors très simple et peu efficace.



admin : l'ensemble des fichiers concernant la partie administrateur du site.

assets : les différents composants graphiques du site (polices, images, js etc…)

js : les scripts javascripts ne concernant pas les parties graphiques (validation, passage de commande etc…)

src : le fichier PHP de connexion à la base de donnée

Comme vous pouvez le constater cette architecture est très simpliste et pas très fonctionnelle, nous n'avions à l'époque pas une notion d'architecture et nous ne comprenions pas les problématiques engendrées par cette architecture de mauvaise qualité.

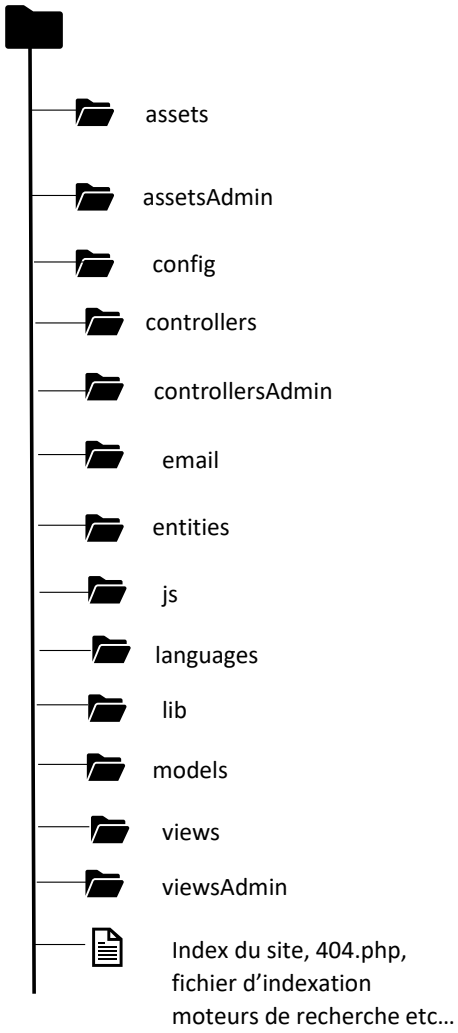
Le passage au modèle MVC était donc nécessaire ce que nous détaillerons dans la prochaine partie.

4) ARCHITECTURE ACTUELLE

En découvrant le modèle MVC et le concept d'architectures de site web, il nous est alors paru évident que celle de notre site était à revoir complétement. Nous avons donc décidé de reprendre l'architecture MVC que nous avions vu en cours en l'adaptant aux particularités de notre site, ce qui donne une architecture unique et très efficace facilitant grandement l'ajout de nouvelles fonctionnalités. En forçant légèrement le trait on pourrait même dire que nous avons créé notre propre Framework pour site de restauration ! Pour revenir aux bienfaits de cette architecture on peut également mentionner le gain de sécurité majeur qu'apporte cette organisation des fichiers et des fonctionnalités.

Le passage au modèle MVC a donc représenté une grosse charge de travail sur le moment mais ce n'est pas du temps perdu puisque le développement s'en est retrouvé facilité et donc accéléré.

Vous trouverez donc dans la page suivante une explication détaillée de l'architecture de notre site.



assets : ce dossier se divise en 6 sous-répertoires :

- css : ce répertoire contient toutes nos feuilles de style css, les css de Bootstrap et un dossier colors.
- fonts : ce répertoire contient les différentes polices utilisées dans notre site.
- images : ce répertoire contient toutes les images utilisées dans notre site, s'y trouve également un répertoire test pour placer les images en test durant le développement.
- js : ce répertoire contient simplement les différents fichiers javascript de notre site.
- lib : ce répertoire contient toutes les librairies, et outils de développement extérieurs.
- scss : ce répertoire contient tous nos fichiers scss servant à enrichir nos fichiers css pour tout le côté graphique du site (principalement la page d'accueil).

assetsAdmin : ce répertoire contient toutes les ressources graphiques de la partie admin de notre site.

config : contient seulement le fichier configuration. PHP dans lequel on retrouve les différentes constantes et les PATHS.

controllers : ce répertoire contient nos différents contrôleurs, ils sont au nombre de 10 dans la partie client du site.

controllersAdmin : contient nos différents contrôleurs de la partie admin du site, ils ont au nombre de 11.

email : contient le script PHP s'occupant de l'envoi de mails, c'est donc là-dedans que se trouvent les modèles des mails envoyés.

entities : ce répertoire contient toutes nos entités, elles représentent les différents objets manipulés dans notre site (commandes, produits, users ...), elles sont au nombre de 10.

js : ce répertoire contient nos deux script js les plus importants, à savoir celui de validation de formulaires et celui s'occupant du passage des commandes.

languages : contient les fichiers de langue du site, pour l'instant notre site n'est qu'en français. Nous aimerions faire une version en espagnol ou en anglais dans le futur.

lib : ce répertoire contient foncBase. PHP notre script gérant les alertes et les messages.

models : contient tous les modèles du site, ce sont ces fichiers qui gèrent notre base de données, ils sont au nombre de 12.

views : contient toutes les vues de notre site côté client, c'est « l'affichage du site », elles sont au nombre de 12.

viewsAdmin : contient toutes les vues de notre site côté admin, c'est à l'identification que l'utilisateur est redirigé vers le côté admin ou « client ».

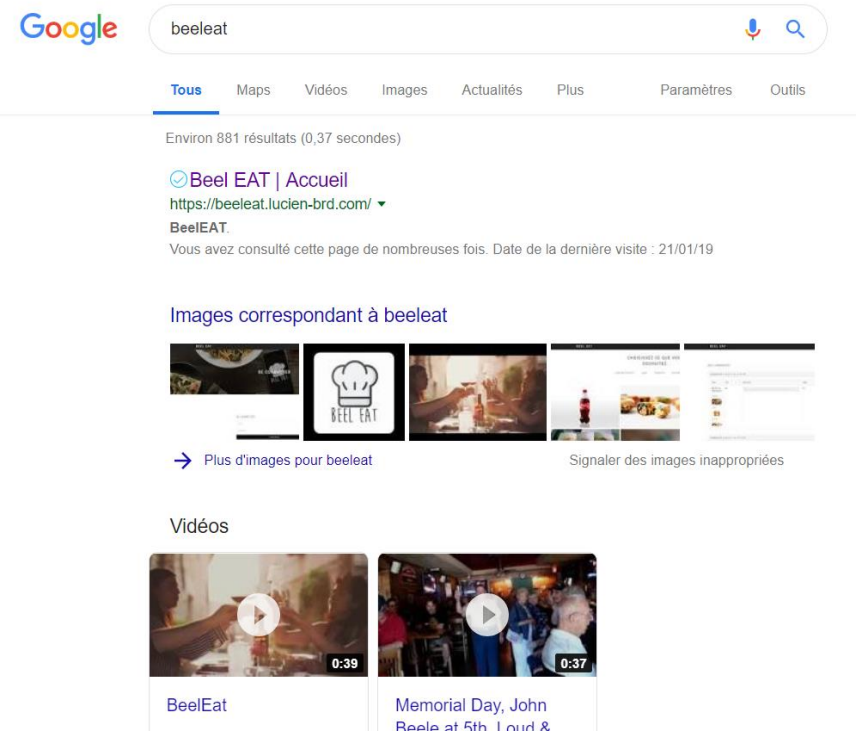
Dans le répertoire racine se trouve donc l'index du site, le 404.php, les fichiers pour le référencement et le script pour le déploiement en continu via git.

5) DEPLOIEMENT CONTINU

Nous avons un script dans notre site qui nous permet de mettre automatiquement à jour le code présent sur le serveur avec les derniers changements (commit) effectués. En sommes, dès qu'un des développeurs envoie les changements effectués sur notre git (push). Git va appeler une url que nous avons configurée qui va exécuter le script sur le serveur et mettre à jour le code de celui-ci (pull).

6) REFERENCEMENT

Nous avons référencé notre site à l'aide des outils de google. Comme vous pouvez le constater, nous sommes la première apparition de recherche :



4. Cahier des charges

Pour réaliser notre cahier des charges, nous avons dû consulter, tout au long du semestre 2 les membres du BDE Génie civil pour savoir ce qu'ils attendaient de notre application web. Nous avons ensuite traduit leurs besoins en un ensemble de fonctions et sous fonctions que nous avons rassemblées sous forme de tableau.

1) CAHIER DES CHARGES PREVISIONNEL

Nom	Description	Contraintes
F1	L'utilisateur sélectionne le menu/sandwich de son choix et va le récupérer au BDE une fois que la commande est prête	L'utilisateur ne peut commander qu'un seul menu ou qu'un seul sandwich par commande
C1	L'utilisateur entre son adresse universitaire et un mot de passe dans une zone de texte pour s'inscrire. Un mail de confirmation est envoyé Pour se connecter l'utilisateur entre son adresse mail et son mot de passe.	Le MDP doit comporter au moins 1 majuscule, 1 minuscule et un chiffre
C2	L'utilisateur aperçoit les menus/sandwichs non disponible de manière grisée	L'utilisateur ne peut pas sélectionner les menus/sandwichs non disponibles
C3	L'utilisateur peut sélectionner le menu/sandwich en cliquant sur les encadrés menus/sandwichs	

C4	A la fin de sa commande l'utilisateur doit voir un récapitulatif de l'intégralité de sa commande	Le récapitulatif n'apparaît qu'après que l'utilisateur ait choisi son dessert
C5	L'utilisateur doit recevoir une notification l'alertant que sa commande est prête	L'alerte doit être visible depuis l'écran de verrouillage
C6	L'utilisateur reçoit une notification s'il n'est pas allé chercher sa commande	La notification est envoyée après 15 minutes de la réalisation de la commande
C7	Les administrateurs peuvent mettre les sandwichs dans l'état disponible ou indisponible	
C8	Les administrateurs mettent à jour les commandes en cliquant sur celles-ci	1er clic → commande en cours 2eme clic → commande prête

C9	Les administrateurs peuvent voir la liste des commandes	

C10	Une notification est envoyée quand les stocks sont bas	La notification est envoyée quand les stocks sont en dessous d'une certaine valeur
C11	Les administrateurs peuvent désactiver les notifications lorsque les stocks sont bas	Les notifications doivent être activées pour pouvoir être désactivées
C12	Les administrateurs peuvent connaître les personnes qui n'ont pas pris leur commande	

DETAILS ET DEROULEMENT DE LA FONCTION F1 (PASSER UNE COMMANDE)

1. L'utilisateur choisit s'il veut un sandwich/panini/assiette ou un menu, en cliquant sur un des encadrés (menu ou sandwich/panini/assiette) qui sont sur la page d'accueil :
2. S'il choisit un sandwich/panini/assiette :
3. Il est redirigé sur une page où il peut choisir les ingrédients de son choix qui se trouvent dans un menu déroulant. Une fois son choix effectué, il clique sur un bouton valider
4. Sinon s'il choisit un menu :
5. Il est redirigé sur une page où il peut choisir de cliquer sur les encadrés sandwich/panini/assiette…etc.
6. Lorsqu'il clique sur un des encadrés, il est redirigé sur une page où il peut choisir les ingrédients se trouvant à <disposition à déterminer> (en haut de la page est affiché le chemin où il se trouve ex : Menu > Panini > Ingrédients). Il clique ensuite sur un bouton suivant qui est en bas à droite.
7. Il est alors dirigé sur une page où il peut cliquer sur des encadrés où se trouvent les boissons. Il clique sur suivant pour passer au dessert
8. Il est redirigé sur une page où il peut choisir son dessert en cliquant sur les encadrés. Il clique sur le bouton "Terminer" pour finaliser sa commande.

DETAILS ET DEROULEMENT DE C1 (CONNEXION/INSCRIPTION)

Pour l'inscription :

1. L'utilisateur entre son adresse universitaire et son mot de passe (mot de passe entre 6 et 20 caractères) dans des zones de texte distinctes.
2. Il entre son mot de passe une deuxième fois dans une autre zone de texte pour confirmer son mot de passe
3. Une fois les deux mots de passe entrés, l'utilisateur clique sur un bouton qui vérifie si les deux mots de passe sont les mêmes :
4. Si c'est le cas :
5. Un mail de confirmation lui est envoyé dans lequel il peut cliquer sur un lien qui le redirige sur la page d'accueil de l'application.
6. Si ce n'est pas le cas, on redirige l'utilisateur vers une page d'erreur et on lui fait reprendre l'inscription du départ

Pour l'identification :

1. L'utilisateur entre son adresse universitaire et son mot de passe :
2. S'ils sont valides, l'utilisateur est redirigé sur la page d'accueil
3. Sinon il est redirigé sur une page d'erreur et il doit recommencer la saisie de son adresse universitaire et de son mot de passe

DETAILS ET DEROULEMENT DE C2 (UTILISATEUR APERÇOIT LES MENUS INDISPONIBLES)

L'utilisateur aperçoit les encadrés de menus/sandwichs non disponibles de manière grisée et ne peut pas les sélectionner.

DETAILS ET DEROULEMENT DE C3 (SELECTIONNER LE MENU)

L'utilisateur peut cliquer sur les encadrés menus/sandwichs.

DETAILS ET DEROULEMENT DE C4 (AFFICHER LE RECAPITULATIF)

1. Une fois que l'utilisateur a choisi son dessert et cliqué sur 'Valider' : il est redirigé sur une page où il voit toute sa commande :
2. S'il a pris un menu :
3. Il voit son sandwich avec les ingrédients et les sauces, sa boisson, son dessert et le prix total.
4. S'il a commandé à l'unité :
5. Il voit ce qu'il a choisi et le prix.

DETAILS ET DEROULEMENT DE C5 (RECEVOIR UNE NOTIFICATION)

6. Une fois que la commande de l'utilisateur est prête, une notification apparaît sur sa page d'accueil.
7. Si le téléphone de l'utilisateur est verrouillé ou si son téléphone n'est pas verrouillé mais qu'il n'est pas sur la page de l'application :
8. Une notification apparaît en haut de son écran en disant que sa commande est prête.

DETAILS ET DEROULEMENT DE C6 (L'UTILISATEUR REÇOIT UNE NOTIFICATION S'IL N'EST PAS ALLE CHERCHER SA COMMANDE)

1. Si l'étudiant qui a commandé n'est pas venu chercher sa commande après 15 minutes :
2. Une notification lui est envoyé sur sa page d'accueil pour lui rappeler de venir chercher sa commande.

DETAILS ET DEROULEMENT DE C7 (LES ADMINISTRATEURS METTENT LES SANDWICHS DANS L'ETAT DISPONIBLE/INDISPONIBLE)

1. Pour rendre un produit disponible/indisponible, les administrateurs cliquent longuement sur le produit choisi.
2. Puis une liste déroulante avec "disponible" et "indisponible" apparaît. Ils cliquent alors sur l'état qu'ils veulent mettre au produit :
3. S'ils cliquent sur indisponible :
4. Le produit devient alors grisé.
5. Sinon s'ils cliquent sur disponible :
6. Le produit devient "coloré".

DETAILS ET DEROULEMENT DE C8 (LES ADMINISTRATEURS METTENT A JOUR LES COMMANDES)

1. Lorsque les utilisateurs valident une commande, celle-ci est ajoutée à une liste de commandes, qui se trouve dans l'onglet commande. La commande validée a directement le statut de commande non prête. Commande non prête -> couleur rouge
2. Pour rendre une commande en cours, l'administrateur clique sur la commande non prête (couleur rouge) dès qu'il commence à préparer la commande. Celle-ci devient alors orange.
3. Lorsque l'administrateur effectue un autre clic sur la commande en cours (couleur orange), celle-ci devient alors de couleur verte.
4. Dès qu'une commande est prête, les sandwichs/panini/wraps… nécessaires à la commande, voient leur stock se décrémenter automatiquement.

DETAILS ET DEROULEMENT DE C9 (LES ADMINISTRATEURS PEUVENT VOIR LA LISTE DES COMMANDES)

1. L'administrateur doit aller dans l'onglet commande en cliquant dessus.
2. Il est alors redirigé sur une page où il peut apercevoir toutes les commandes numérotées dans l'ordre :
3. Commande de couleur rouge -> commande non prête
4. Commande de couleur orange -> commande en cours
5. Commande de couleur verte -> commande prête

DETAILS ET DEROULEMENT DE C10 (UNE NOTIFICATION EST ENVOYEE LORSQUE LES STOCKS SONT BAS)

1. Une notification est envoyée sur la page d'accueil administrateur en disant que les stocks sont bas, quand les stocks sont en dessous de la valeur fixée (au préalable).

DETAILS ET DEROULEMENT DE C11 (LES ADMINISTRATEURS PEUVENT DESACTIVER LES NOTIFICATIONS LIEES AU STOCK)

1. L'administrateur clique sur l'onglet stock.
2. Il est redirigé sur la page des stocks.
3. Il clique sur le bouton de "power off" prêt de la barre notifications pour désactiver les notifications.

DETAILS ET DEROULEMENT DE C12

1. Puisque chaque commande sera associée à l'adresse universitaire. Les administrateurs pourront voir dans la liste des commandes le nom/prénom qui seront en dessous de la commande, des étudiants qui ne sont pas venus les chercher.

2) COMPARAISON DE CE QUI A ETE FAIT AVEC LE CAHIER DES CHARGES PREVISIONNEL

FONCTIONS LIEES A L'UTILISATEUR

Pour la fonction F1 (passer une commande) :

L'utilisateur peut choisir de prendre un menu ou un produit tout seul. Il peut choisir les ingrédients qui composent son sandwich.

Pour la fonction C1 (inscription/connexion) :

L'utilisateur entre son adresse mail, un pseudo et son mot de passe dans des zones de textes pour s'inscrire, le mot de passe doit contenir entre 8 et 15 caractères avec au moins une majuscule, une minuscule et un caractère spécial. Nous avons rajouté le fait que le mot de passe doit contenir un caractère spécial.

Pour se connecter l'utilisateur entre son adresse mail et son mot de passe.

Nous avons parfaitement ce qui a été écrit dans le cahier des charges pour cette fonctions.

Pour la fonction C2 (l'utilisateur aperçoit les sandwichs non disponibles en gris) :

Contrairement à ce qui était prévu, l'utilisateur n'aperçoit pas du tout les sandwichs non disponibles sur la page des ingrédients/menus.

Pour la fonction C3 (l'utilisateur peut cliquer sur les encadrés menus/sandwichs) :

L'utilisateur peut cliquer sur les menus/sandwichs de son choix. Nous avons rajouté le fait que l'utilisateur puisse faire apparaître seulement une catégorie d'éléments.

Pour la fonction C4 (récapitulatif de la commande) :

Une fois que l'utilisateur a terminé sa commande il aperçoit un récapitulatif de sa commande avec les ingrédients qu'il contient, la boisson, le dessert et le prix s'il a pris un menu. De manière similaire, il aperçoit un récapitulatif de sa commande s'il commande à l'unité.

Pour la fonction C5 (recevoir une notification dès que la commande est prête) :

L'utilisateur reçoit en effet une notification dès que sa commande est prête, cependant il reçoit la notification par mail alors que nous avions initialement prévu que la notification apparaisse sur la page d'accueil de l'utilisateur.

Pour la fonction C6 (recevoir une notification si l'utilisateur n'est pas allé prendre sa commande) :

Malheureusement, nous n'avons pas eu le temps d'implémenter cette fonctionnalité.

FONCTIONS LIEES A L'ADMINISTRATEUR

Pour la fonction C7 (les administrateurs peuvent mettre à jour l'état des produits) :

L'administrateur va dans l'onglet produit puis éditer produit, ensuite il clique sur « disponible », « indisponible », ou « publié ». Un produit peut être disponible mais il peut ne pas apparaître sur la page des utilisateurs car il n'est pas dans l'état publié.

Pour la fonction C8 (les administrateurs mettent à jour les commandes) :

Lorsque les administrateurs se connectent, ils arrivent sur la page des commandes et ils peuvent éditer le statut de la commande en cliquant sur « changer de statut ». Nous avons décidé d'enlever le système de clique car nous avons trouvé ce système peut ergonomique pour l'expérience de l'administrateur (par exemple s'il « miss click »).

Pour la fonction C9 (les administrateurs peuvent voir la liste des commandes) :

Les administrateurs peuvent effectivement voir la liste des commandes.

Pour la fonction C10 (une notification est envoyée lorsque les stocks sont bas) :

Une notification est en effet envoyée lorsque les stocks sont bas.

Pour la fonction C11 (les administrateurs peuvent désactiver les notifications liées au stock) :

Tout comme la fonction C6, nous n'avons pas eu le temps d'implémenter cette fonction dans notre application

Pour la fonction C12 (les administrateurs peuvent savoir qui n'est pas allé chercher sa commande) :

Les administrateurs peuvent effectivement connaître qui n'est pas allé chercher sa commande.

3) CONCLUSIONS ET PERSPECTIVES

Pour conclure cette partie gestion de projet, nous avons découpé les tâches par rapport aux fonctions initialement mises dans le cahier des charges et nous nous sommes ensuite organisés de manière agile avec la méthode Scrum.

Nous avions prévu un diagramme de Gantt mais il y a eu des écarts comme nous pouvons le constater sur le diagramme de Gantt réalisé.

Ces écarts ont notamment été dus à cause d'une part, d'un changement d'architecture de l'application et d'autre part par rapport à une communication difficile avec le client

Nous avons tout de même réussi a trouvé des solutions à ces problèmes.

5. Spécifications fonctionnelles et détaillées

Nous allons détailler dans cette partie les diverses fonctionnalités assurés par notre site. Nous allons commencer par détailler les fonctionnalités et nous expliquerons ensuite leur implémentation.

Tout d'abord, nous allons voir comment nous pouvons nous connecter et comment discerner si nous sommes un client ou un administrateur.

Pour cela nous utilisons des variables de sessions, get/post et les cookies :

- Une session est un mécanisme technique permettant de sauvegarder temporairement sur le serveur des informations relatives à un internaute.
- GET et POST sont des méthodes d'accès définies dans le protocole HTTP et reprises dans la spécification HTML. Le choix de la méthode dépend de la façon dont les données sont reçues, de la taille et la nature des données.
- Un cookie est un petit fichier texte (faisant au maximum 65 Ko) stocké sur le disque dur du visiteur du site. Ce fichier texte permet de sauvegarder diverses informations concernant ce visiteur afin de pouvoir les réutiliser (les informations) lors de la prochaine visite du visiteur sur ce même site.

1) INSCRIPTION

Quand nous nous inscrivons sur le site, nous enregistrons dans la base de données plusieurs hash unique qui nous permettront d'identifier un utilisateur. Premièrement pour qu'il confirme son adresse mail. En effet, au moment de l'inscription, l'utilisateur reçoit un mail de confirmation pour activer son compte pour des raison de sécurité. Il doit juste se rendre sur le lien présent dans le mail afin d'activer son compte. Sur ce lien nous retrouvons l'utilisateur grâce au hash unique.

2) CONNEXION

On commence par vérifier s'il existe un cookie et s'il n'existe pas de variable de session connect. Ce qui veut dire que l'utilisateur n'est pas connecté mais qu'au moment de sa dernière connexion il avait choisi de rester connecter. Pour des raisons de sécurité nous vérifions que le cookie contient bien un utilisateur de la base de données. Si c'est effectivement le cas, alors nous créons des variables de sessions afin de connecter l'utilisateur et de l'identifier rapidement (si c'est un client ou non…).

Sinon, nous regardons les variables post renvoyé via le formulaire de la page de connexion. Nous recryptons le mot de passe de la même manière qu'au moment de l'inscription afin de comparer les mots de passes.

Puis, s'il existe un utilisateur qui a la même adresse mail et le même mot de passe que ceux envoyé via le formulaire alors comme vu précédemment, nous créons les variables de session pour connecter l'utilisateur. De plus, si l'utilisateur a choisi de rester connecter, nous créons alors un cookie pour que la connexion se fasse automatiquement comme vu précédemment.

Pour des raisons de sécurité, quand un utilisateur essaie de se connecter avec un mauvais email ou mot de passe, nous lui disons juste qu'il est impossible de se connecter. En effet, s'il met un bon email et un mauvais mot de passe, nous ne disons pas que le mail est bon mais pas le mot de passe.

D'autre part, nous utilisons notre propre fonction de cryptage de mot de passe dans l'optique d'avoir vraiment des mots de passe inviolables.

Nous venons de voir comment un utilisateur se connecte. Nous allons maintenant voire comment le rediriger vers la bonne partie du site (admin ou client).

Dans le fichier index (chaque page du site passe par ce fichier), nous regardons quelle variable de session existe. S'il existe la variable de session admin, alors nous redirigeons l'utilisateur vers les pages et fichiers de la partie admin. Sinon c'est que nous sommes du côté client (les pages de connexion sont dans cette partie).
En sommes, afin d'avoir un site plus sécuritaire, nous vérifions dans chaque contrôleur s'il existe les bonnes variables de session.

3) ADMINISTRATEUR

Du coté administrateur, l'utilisateur à la possibilité d'ajouter/éditer des ingrédients. Pour ce faire, dans le code nous utilisons le contrôleur ingrédientE. Ce contrôleur permet soit d'ajouter un nouvel ingrédient, soit d'en éditer un existant. Pour cela, nous vérifions si l'ingrédient que nous traitons à un id, si oui alors l'ingrédient existe déjà donc nous l'éditons sinon nous en créons un nouveau.

Comme nous l'avons vu précédemment, nous vérifions en début de code les variables de session. Si nous ne sommes pas connectés, nous sommes redirigés vers l'accueil sinon, si nous sommes bien un admin, nous pouvons accéder aux fonctionnalités de la page souhaitée.

Nous allons voir juste une partie du code : modifier un ingrédient. Nous récupérons la variable get id qui correspond à l'id unique d'un ingrédient. Puis nous vérifions qu'un ingrédient correspond bien à cet id, dans le cas contraire, un message d'alerte informera l'utilisateur que l'ingrédient n'existe pas.
Puis, nous récupérons tous les variables post du formulaire et les traitons pour des raisons de sécurité.
Ces variables contiennent toutes les informations qui définissent un ingrédient.

Puis nous mettons à jour l'ingrédient, si tout c'est bien passé nous en informons l'utilisateur. Il est primordial que l'utilisateur ait un retour sur tout ce qu'il fait afin de ne pas être perdu. De plus, c'est un avantage ergonomique.

C'est le même principe pour la création d'un ingrédient à l'exception qu'au lieu de mettre à jour un existant, nous en créons un nouveau.
De plus, cela fonctionne pareillement pour les menus et les produits.
Or, les produits et menu ont une image optionnelle en plus. En effet, si l'utilisateur décide de ne pas mettre d'image, nous lui attribuons une image par défaut.
Nous enregistrons dans la base de données le chemin sur le serveur où se trouve l'image. Nous aurions pu enregistrer directement l'image dans la base de données mais pour des raisons de performance nous avons opté pour la première solution.

De plus, un produit peut être composé de plusieurs ingrédients. Nous nous sommes posés la question, comment allons-nous enregistrer dans la base de données le fait qu'un produit puisse avoir plusieurs ingrédients. A la suite de nombreuses recherches et de beaucoup de réflexions, nous avons trouvé une solution.
Il fallait une nouvelle table dans la base de données qui puisse faire le lien entre l'id d'un produit et l'id d'un ingrédient. Ainsi, nous pouvons récupérer tous les ingrédients correspondant à un produit.
Or, ne sachant pas le nombre exact d'ingrédients qui composeront le produit que l'utilisateur créé ou édite, nous avons développé une fonction qui liste toutes les variables post passés par le formulaire, et ajoute dans une liste tous les id des ingrédients (la fonction doit bien évidement ne pas prendre en compte les variables dont nous connaissons l'existence comme le nom du produit).
Le principe est le même pour les menus.

```
$prodListe = array();
foreach ( $_POST as $post => $val )  {
    if($post != 'nom' && $post != 'quantite' && $post != 'statut' && $post
!= 'type' && $post != 'prix'){
        $prodListe[] = $val;
    }
}
```

Nous avons développé notre fonction d'envoi de mail. En somme, grâce à notre fonction, nous pouvons envoyer facilement des mails aux clients, avec un design qui correspond aux normes du site. Le design du mail a été pensé pour être ergonomique et très responsive. En effet, selon notre étude auprès de nos clients, les utilisateurs seront en majorité sur leur portable pour utiliser notre application web. Le mail fut développé en format html.

Voici un exemple de mail :

4) CLIENT

Dans la partie client, une des grosses difficultés fut de passer une commande avec un menu. Le menu étant composé de produits, eux-mêmes composés d'ingrédients. Il fallait afficher seulement les produits et ingrédients disponibles (ceux dont les stocks ne sont pas épuisés). De plus, au moment de la commande, le stock des ingrédients et produits se décrémente automatiquement. Si ce stock atteint un seuil limite (inférieur à 5) nous informons premièrement l'administrateur du fait qu'il va falloir se réapprovisionner de l'élément spécifié. Et, si le stock est inférieur à 1, le statut passe automatiquement en indisponible, il n'est donc plus possible de passer des commandes avec le produit/ingrédient.

Nous avons développé un script javascript qui va permettre dans le navigateur de sélectionner le bon nombre de produits, envoyer les bonnes variables post au serveur et vérifier que l'utilisateur sélectionne tout ce qu'il faut.

5) SECURITE

Nous avons recherché les failles auquel notre site pourrait être victime. Nous en avons rapidement découvert et fixé celles-ci.

Par exemple, du fait de l'architecture de notre projet, il était possible d'accéder par l'url du navigateur aux code source. En écrivant par exemple : https://beeleat.lucien-brd.com/models nous pouvions avoir accès aux fichiers du dossier models.

Nous avons donc interdit l'accès à tous les dossiers autre que public qui est un dossier spécial auquel l'utilisateur doit avoir accès (les images du site y sont stockées). En effet, si nous interdisions l'accès à ce dossier, le site n'aurait plus d'image, de style…

Maintenant, si on essaye d'accéder à un dossier interdit, nous sommes redirigés sur la page d'accueil en toute transparence.

Nous avons obligé notre site à utiliser le protocole HTTPS. C'est un protocole de transfert hypertexte plus avancé et bien plus sécurisé que le protocole HTTP standard. En effet, tous les transferts sont chiffrés.

6) APPLICATION

Du fait de la forte demande du client sur smartphone et plus particulièrement sur Android, nous avons développé une application Android. En effet, après notre étude auprès de nos clients nous nous sommes rendu compte que la majorité des utilisateurs sera sur portable et majoritairement Android. Il était donc évident pour nous que BeelEAT soit disponible sur le Play store.

De ce fait, en plus de l'application web, nous nous sommes intéressés à Android Studio. L'application que nous avons développée est directement reliée à notre serveur. Grâce à l'intégration continue que nous avons mis en place, l'application se met à jour automatiquement. De plus, nous travaillons actuellement sur la possibilité de recevoir des notifications directement sur le portable à la place des mails pour l'application.

L'application est un énorme plus, les utilisateurs que nous avons questionnés ont répondu à 95% préférer avoir une application, bien que le site fonctionne parfaitement et de la même manière sur portable.

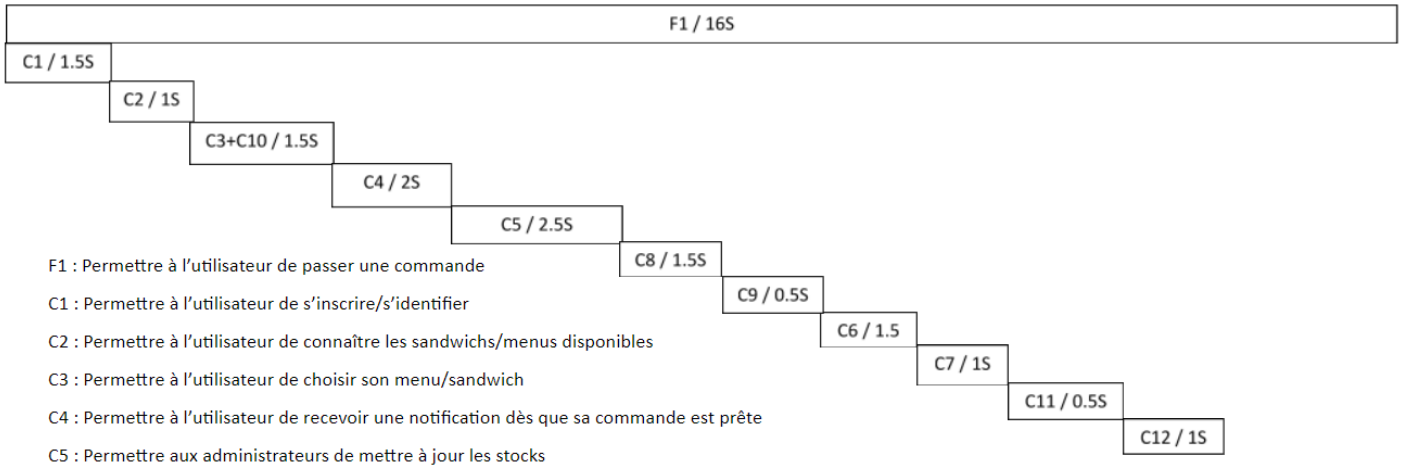
6. Structure et organisation du projet

1) STRUCTURE PREVISIONEL

L'unité choisie est la semaine donc 1S = 7 jours.

Nous considérons avoir un semestre pour le développement, en raccourcissant ce semestre à quatre mois on se retrouve avec environ 16 semaines pour programmer.

L'ordre de développement des fonctionnalités indiquées dans le schéma ci-dessous est probablement amené à changer. Avec la répartition du temps qui a été choisie, nous avons 1.5S de marges, c'est-à-dire environ 10 jours de marge.



F1 : Permettre à l'utilisateur de passer une commande

C1 : Permettre à l'utilisateur de s'inscrire/s'identifier

C2 : Permettre à l'utilisateur de connaître les sandwichs/menus disponibles

C3 : Permettre à l'utilisateur de choisir son menu/sandwich

C4 : Permettre à l'utilisateur de recevoir une notification dès que sa commande est prête

C5 : Permettre aux administrateurs de mettre à jour les stocks

C6 : Permettre aux administrateurs de mettre à jour les commandes

C7 : Permettre aux administrateurs de voir les commandes qui sont prêtes

C8 : Les administrateurs doivent recevoir une notification quand les stocks sont "bas"

C9 : Permettre aux administrateurs de désactiver les notifications citées ci-dessus

C10 : L'application doit faire un récapitulatif de la commande à l'utilisateur

C11 : L'administrateur peut connaître les personnes qui n'ont pas récupéré/payé leur commande

C12 : L'utilisateur doit pouvoir recevoir une notification s'il n'est pas allé prendre sa commande après 15 minutes

2) STRUCTURE ACTUELLE

Voici la structure et l'organisation actuelle de notre projet en fonction des réunions que nous avons eu avec notre tutrice.

REUNION 1

Date : 13 / 10 / 18

Objectifs fixés pour la prochaine réunion :

- Prendre réunion avec le BDE génie civil
- Faire les pages du côté utilisateur
- Faire les pages du côté administrateur
- Montrer un exemple au BDE génie civil

REUNION 2

Date : 03 / 10 / 18

Travail présenté et état d'avancement du projet :

- Inscription utilisateur (avec vérification mail universitaire…)
- Côté admin et client
- Rencontre avec le BDE génie civil

Objectifs fixés pour la prochaine réunion :

- Trier les images, faire des catégories
- Modifier le statut des commandes (case à cocher plutôt que menu déroulant)
- Faire des pages news
- Trier les commandes du côté administrateur
- Créer des menus
- Prévoir des cas de mauvais stocks
- Pouvoir faire une commande

REUNION 3

Date : 23 / 10 / 18

Travail présenté et état d'avancement du projet :

- Passage de commande
- Optimisation du projet (passage au MVC, modèle vue contrôleur)
- Amélioration sécurité

Objectifs fixés pour la prochaine réunion :

- Finir tous les menus
- Faire une commande juste pour un produit
- Pouvoir suivre état d'une commande
- Faire apparaître commande du côté administrateur
- Envoi du mail lorsqu'une commande est passée
- Gestion des stocks (pouvoir refaire les stocks, décrémenter les stocks)
- Prévenir l'utilisateur quand la commande est prête
- Rajouter des produits (demander la carte au BDE)
- Rajouter un type d'utilisateur (cuisine admin)
- FAIRE LE FONCTIONNEL

REUNION 4

Date : 16 / 11 / 18

Travail présenté et état d'avancement du projet :

- Affichage commande côté admin
- Affichage cuisine côté admin
- Affichage commande côté client
- Envoie mail (confirmation commande…)
- Modification et ajout de menu, produits…

Objectifs fixés pour la prochaine réunion :

- Avoir un retour sur BDE
- Faire pages composition de sandwichs
- Hébergement à voir avec le BDE
- Sauvegarde sur machine admin ?
- Sécurité mise en ligne
- (Pouvoir annuler une commande pas en préparation)

REUNION 5

Date : 03 / 12 / 18

Travail présenté et état d'avancement du projet :

- Ajout et modification des ingrédients
- Gestion des stocks

Objectifs fixés pour la prochaine réunion :

- Faire présentation au BDE
- Faire des tests
- Faire une application fonctionnelle
- Changer les titres des onglets
- (Retravailler la mise en page)
- Parler d'hébergement avec BDE
- Rajouter des « News »

REUNION 6

Date : 07 / 01 / 19

Travail présenté et état d'avancement du projet :

- Optimisation gestion des stocks
- Choix ingrédients dans un produit
- Possibilité de commander à l'unité
- Présentation au BDE génie civil

Objectifs fixés pour la prochaine réunion :

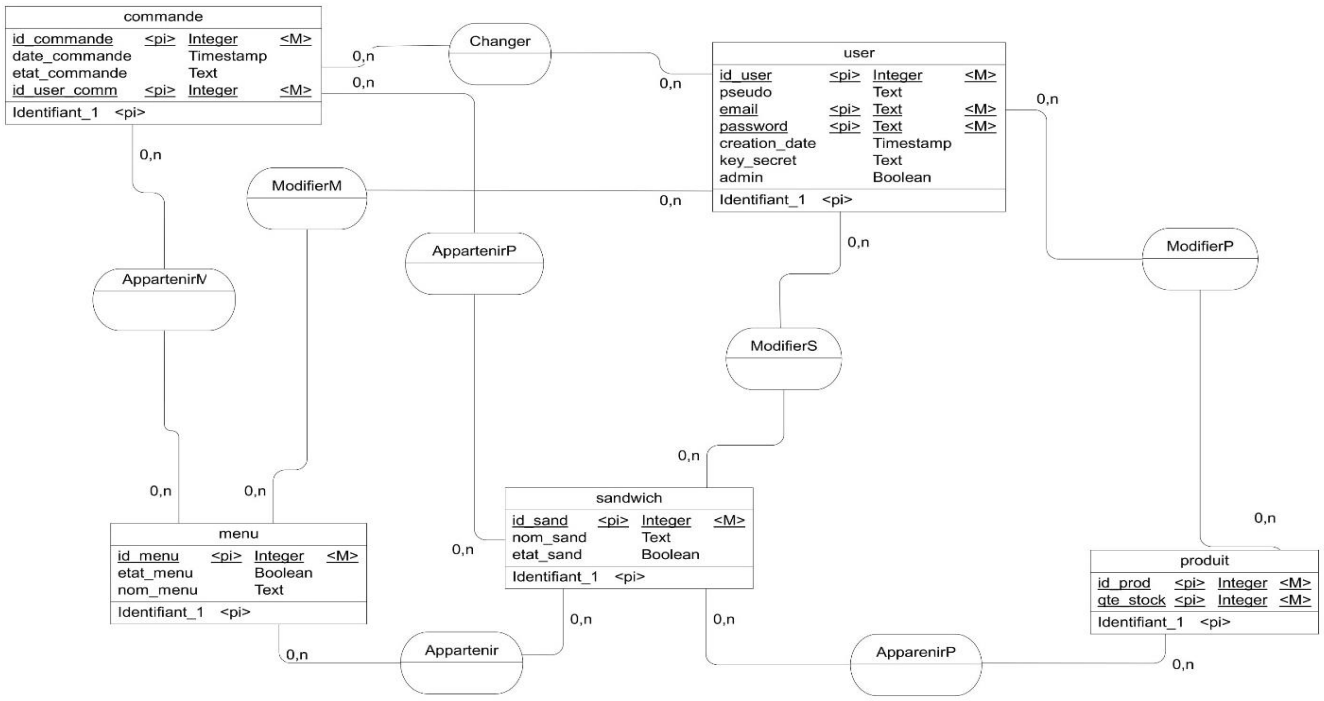
- Hébergeur pour le S4
- Faire apparaitre supplément payant
- Archivage des commandes
- Feuille de retour sur les ventes -> comptabilité

En outre, au quatrième semestre, nous avons grandement améliorer l'ergonomie de notre projet afin, d'une part d'avoir un site le plus fonctionnel possible, et d'autre part de passer en phase de test avec notre client. Cette phase de test nous a déjà permis de corriger de nombreux bug.

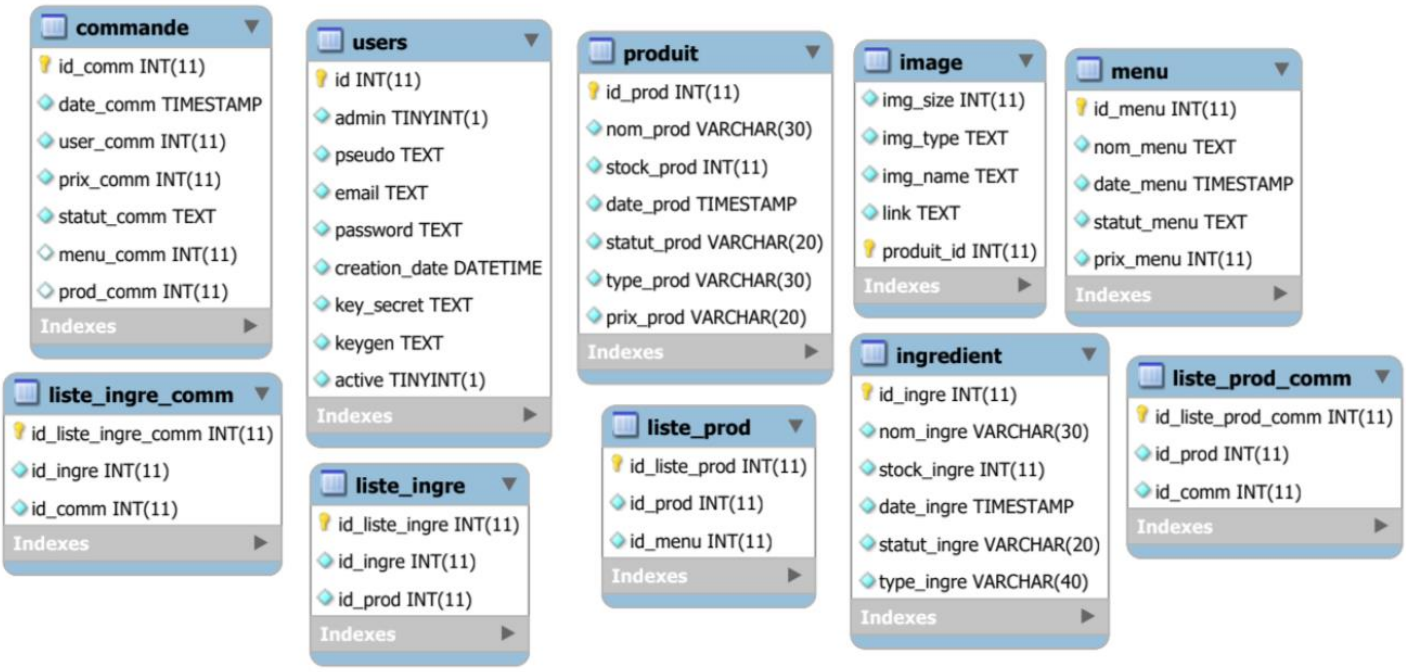
7. Base de données

Afin d'enregistrer et d'utiliser de manière sécuritaire et simple, nous utilisons une base de données. C'est une entité dans laquelle il est possible de stocker des données de façon structurée et avec le moins de redondance possible

1) STRUCTURE INITIALE



2) STRUCULE ACTUELLE



8. Tests et Validation

1) DEBOGAGE DE L'APPLICATION

Nous avons tous développé l'application avec des IDE nous permettant de corriger une grosse partie des fautes de syntaxe et aidant pour le débogage. Malheureusement des bugs sont restés cachés même une fois le développement de certaines fonctionnalités terminé. C'est alors en parcourant l'application dans des cas extrêmes que nous pouvions déceler les bugs restants. C'est lors de brainstormings au moment des réunions du groupe que nous trouvions les cas limites à tester. C'est de cette manière que nous avons par exemple trouvé le problème dans la décrémentation des stocks lors d'une commande utilisateur. En effet, notre application gère les stocks et ceux-ci doivent réduire avec les commandes et en fonction de la composition de celles-ci. Cela n'était pas le cas, l'information était mal transmise à la fonction en charge de cette décrémentation. Pour éviter les oublies, le débogage était la priorité dans les sessions de développement de groupe. Une fois un bug trouvé il fallait le corriger au plus vite pour pouvoir continuer le développement sur des bases toujours propres.

2) TEST DE L'APPLICATION GRACE A LA LIBRAIRIE PYTHON SELENIUM

Il était important pour nous dans ce projet de vérifier sa fonctionnalité, la librairie python Selenium nous a permis de créer des robots simulant un nombre cohérent avec sa future utilisation de connexions et d'actions sur le site simultanément. Nous avons alors simulé connexions et commandes sur le site du côté utilisateur, puis nous avons vérifié les données enregistrées coté administrateur.

```
class Bot:
  def __init__(self, username, password):
    self.username = username
    self.password = password
    self.driver = webdriver.Chrome()
    self.actionChains = ActionChains(self.driver)
    self.dimension = self.driver.get_window_size()
    print(self.dimension)

  def closeBrowser(self):
    self.driver.close()

  def login(self):
    driver = self.driver
    driver.get("https://www.beeleat.lucien-brd.com/")
    WebDriverWait(driver, 20).until(EC.element_to_be_clickable((By.ID, "E-mail"))).send_keys(self.username)
    WebDriverWait(driver, 20).until(EC.element_to_be_clickable((By.ID, "password"))).send_keys(self.password)
    WebDriverWait(driver, 20).until(EC.element_to_be_clickable((By.CLASS_NAME, "btn btn-block btn-round btn-b"))).click()


bot1 = Bot("etienne.dahoumane", "1234abc")
bot1.login()
```

Ci-dessus la création d'un bot se connectant à l'application grâce à la class « Bot » et à travers la fonction « login » avec un identifiant et un mot de passe passé en paramètre. La méthode WebDriverWait attend pour exécuter la ligne suivante qu'un évènement passé en paramètre ce soit produit. Dans ce cas, il s'agit d'attendre qu'un élément HTML choisi grâce à son attribut « id » ou « class » dans la balise HTML soit cliquable. La fonction WebDriverWait renvoie l'élément HTML en question, on peut donc y appliquer des fonctions comme « click() » qui permet de cliquer sur cette élément. C'est comme ça par exemple que l'on clique sur le bouton valider (dernière ligne de la fonction login). La fonction « send_keys » elle, permet d'envoyé une chaine de caractère passée en paramètre à l'élément HTML correspondant généralement à un champ de saisie.

3) TESTS UTILISATEURS DE L'APPLICATION

Pour tester notre application, nous avons tout d'abord sollicité nos collègues de travail ainsi que notre entourage. Cette phase nous a aidé à résoudre quelques problèmes ergonomiques, comme l'emplacement de certains boutons ou la suppression de listes déroulantes perturbant la navigation sur le site. Nous avons eu à ce moment beaucoup de retours positifs sur l'expérience utilisateur, les utilisateurs trouvaient notre application « jolie », « pro », « utile ».

Nous avons ensuite fait tester notre application par les membres du BDE Génie Civil et quelques futurs clients en Génie Civil. Ceux-ci ont tout de suite été impressionnés par notre site. Au début du projet le client nous avais dit vouloir en design similaire avec celui de l'application sur les bornes de commande chez McDonald's. BeelEAT reprend les codes de ce logiciel, mais s'adapte aux exigences du BDE Génie Civil.

4) TEST EN SITUATION REELLE

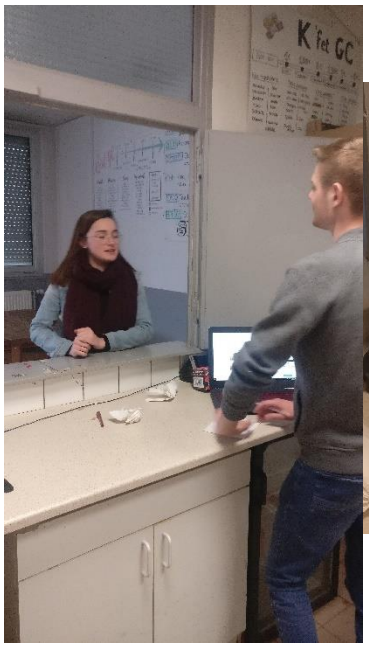
Durant une semaine, BeelEAT a été mis en place en phase de test à la cafétéria. Une dizaine d'élèves sélectionnés se sont vu attribuer un compte leur permettant de commander sur l'application. Voici ce qui est ressorti de cette phase de test.

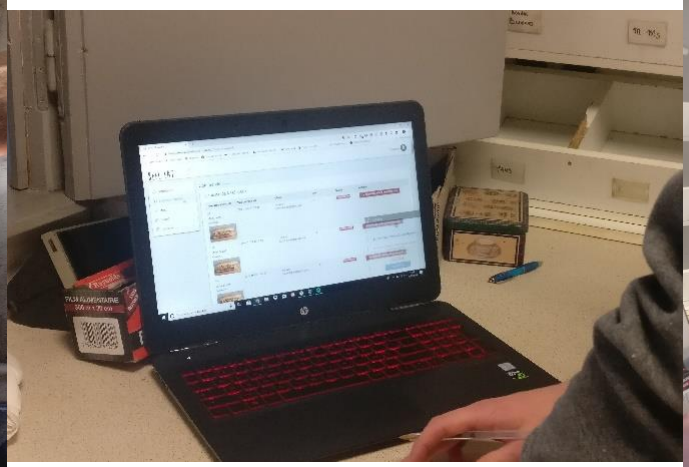
Points positifs :

- Affichage clair pour l'onglet « Commande Cuisine » du côté utilisateur, l'utilisateur voit tout de suite la composition des commandes.
- Application web responsive, le responsive design permet aux utilisateurs de l'application de l'utilisé sur tous les supports. Un PC peut être assez encombrant dans une cuisine, les membres du BDE peuvent alors utiliser une tablette ou leur téléphone pour prendre les commandes.

Points négatifs :

- La personne en cuisine doit contrôler son écran régulièrement pour voir les commandes, les membres du BDE ont proposé une alarme sonore à la réception de chaque commande sur l'application.
- La décrémentation des stocks n'est pas personnalisée en fonction des ingrédients. Par exemple, si dans un sandwich il y a deux tranches de fromage, lors de la commande de ce sandwich les stocks seront décrémenté de 1. En effet c'est au BDE de renseigner dans les stocks non pas le nombre d'ingrédients, mais le nombre de sandwich qu'il est possible de faire avec le stock d'ingrédients.





9. Difficultés rencontrées dans la gestion du projet et solutions apportées

1) COMPLEXIFICATION DU CODAGE DU A UNE ARCHITECTURE FRAGILE

Au début de notre projet, nous avons commencé le développement de l'application sans nous poser de questions. Mais au fur et à mesure que le projet avançait, il était de plus en plus difficile de développer. La structure du projet était de plus en plus floue et le nombre de fichiers augmentait de façon démentielle. Implémenter du code sans tout déranger était devenu difficile, ainsi le développement de manière individuel s'est fait plus rare durant cette période car il était difficile de reprendre le code des autres développeurs seul. C'est n'est qu'à la suite du cours de PHP ou nous avons été introduit au Modèle-Vue-Contrôleur. Nous avons alors repris l'architecture proposée par notre professeur que nous avons adapté à notre projet. Avec ce nouveau modèle, l'implémentation de nouvelle fonction devenait plus simple, plus claire et plus lisible pour le reste du groupe.

2) REGULARITE DANS LE DEVELOPPEMENT ET DANS LA FONCTIONNALITE DU PROJET

Il était pour nous important d'apporter une cohérence dans notre développement pour BeelEAT. Un code qui serait rigoureux et facilement réutilisable. La solution trouvée pour ce souci de cohérence a été de développer le plus possible ensemble, réunir au maximum toute l'équipe de développement. Nous nous sommes basés sur les méthodes de développement agiles vues en cours, pour développer grâce à des « sprints » de développement organisé les jeudi après-midi. Nous avions chaque jeudi des objectifs à atteindre comme la réalisation d'une fonction. Ceci régla par la même occasion le problème d'avoir tout le temps un projet qui marche à montrer au client, car à chaque fin de sprint nous avions un rendu fonctionnel.

3) UNE COMMUNICATION DIFFICILE AVEC LE CLIENT

Dès le départ le client fut difficile à investir dans notre projet, étant donné que celui-ci ne paye pas pour l'application et que nous, les développeurs, sommes à l'origine du projet, le client avait à la base peu d'attentes de BeelEAT. En tant que développeurs nous avions du coup tendance à nous centrer sur nos idées et nous faire des idées sur ce que le client voulait au lieu de lui demander directement. Nous avons finalement réussi à investir le client dans le projet après une démonstration qu'il a trouvé « bluffante ». C'est en voyant les possibilités concrètes de l'application que le client a vu son potentiel à l'utilisation. La communication s'est alors renforcée entre le BDE et les développeurs aidant à la finalisation du projet. Nous sommes conscients que cela ne se passera pas toujours comme ça lors d'un vrai projet informatique : le client ayant de vraies exigences.

4) UNE ANALYSE INADAPTE A LA REALITE DU PROJET

Malgré une analyse conséquente, nous nous sommes rendu compte au fur et à mesure du développement de l'application que nous devions quelquefois penser à de nouvelles fonctionnalités ou passer plus de temps sur certaines fonctions que ce qui était prévu. Nous ne nous sommes également pas assez renseignés dès le début du fonctionnement du système sur l'interaction de BeelEAT, dans ce cas la cafeteria du BDE Génie civil. Certains paramètres de terrain, comme la manière dont le client peut librement composer son sandwich, nous on fait revoir l'ensemble de l'ergonomie de certaines pages. Nous pensons que c'est avec l'expérience que s'affinera notre analyse de début de projet.

10. Conclusion

BeelEAT est une application fonctionnelle. Malgré quelques améliorations possibles, le projet a été testé en condition réel et fonctionne sans problèmes. Ce projet est pour tous les développeurs de l'application, le premier grand projet informatique de groupe, avec un cas concret au bout et un véritable client.

BeelEAT nous a permis d'apprendre à nous auto-former et à travailler en groupe. Par exemple, aucun d'entre nous n'avais utilisé Bootstrap, aujourd'hui nous sommes tous formé à ce Framework. Nous avons également appris à respecter des délais dans le cadre du projet (rendu du rapport, livraison de fonctionnalités…). Cela nous a fait gagner en autonomie et en organisation dans notre travail. La maitrise du gestionnaire de version GIT a été véritablement un plus pour le travail en groupe.

BeelEAT c'est une application web et une application mobile, c'est un projet qui nous tient à cœur et que l'on compte approfondir encore en dehors du projet tutoré. Déjà en ligne, avec un code facilement réutilisable et adaptable pour d'autre client que le BDE Génie Civil, l'application ne s'arrête pas là !

11. Remerciements

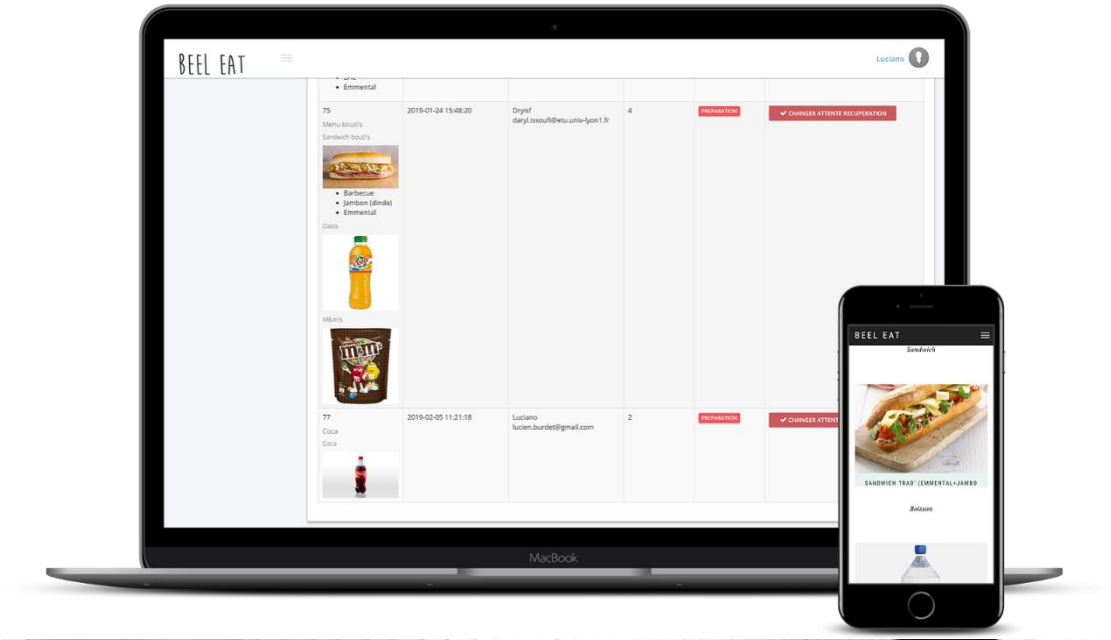
Je tiens à remercier toutes les personnes qui ont contribué au succès de notre projet et qui nous ont aidé lors de la rédaction de ce rapport.

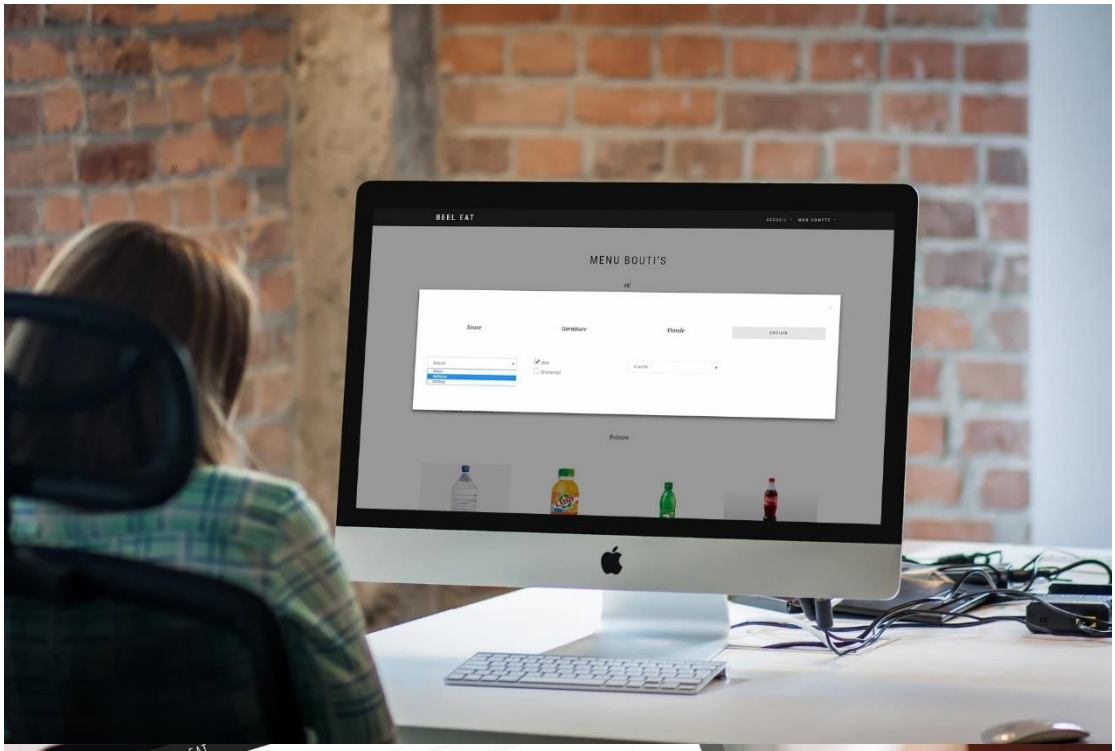
Tout d'abord, j'adresse nos remerciements à notre professeur, **Aude Joubert** qui nous a beaucoup aidé pour cibler les problèmes et trouver des solutions efficaces. Son écoute et ses conseils nous ont permis de cibler nos lacunes, et de trouver rapidement des chemins alternatifs.

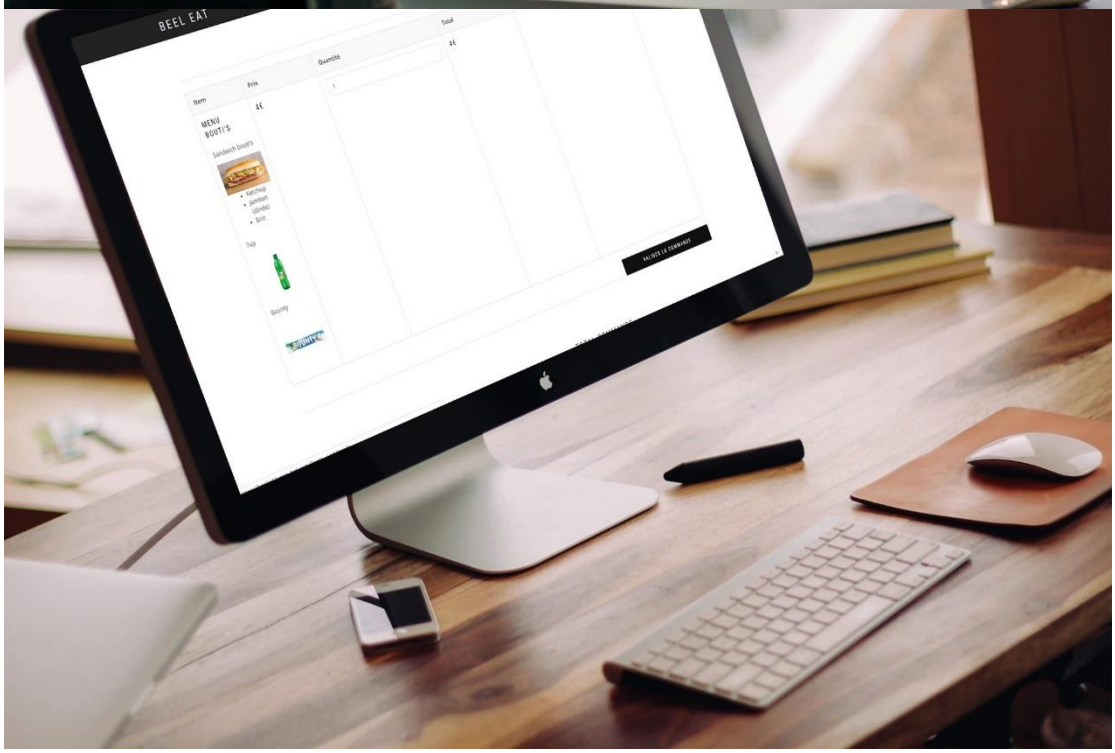
Je tiens à remercier vivement le BDE Génie Civil, pour son attention, le temps passé ensemble et le partage de cette expérience très enrichissante. Grâce aussi à sa confiance nous avons pu accomplir avec plaisir ce projet.

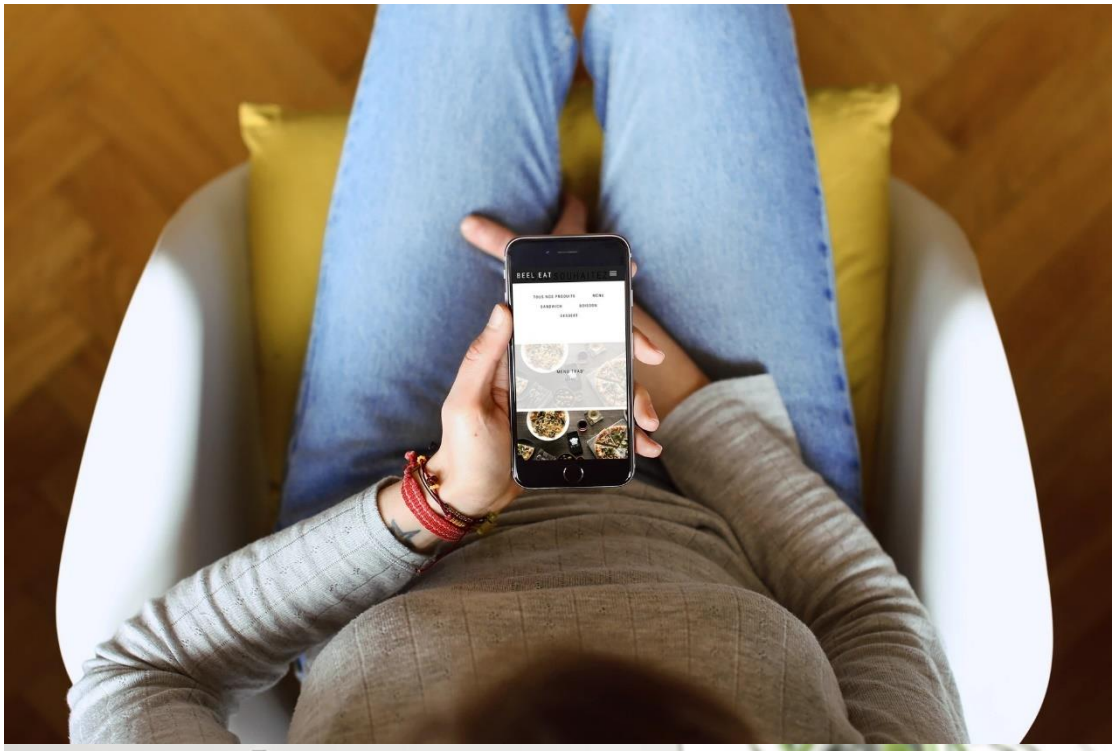
12. Annexe

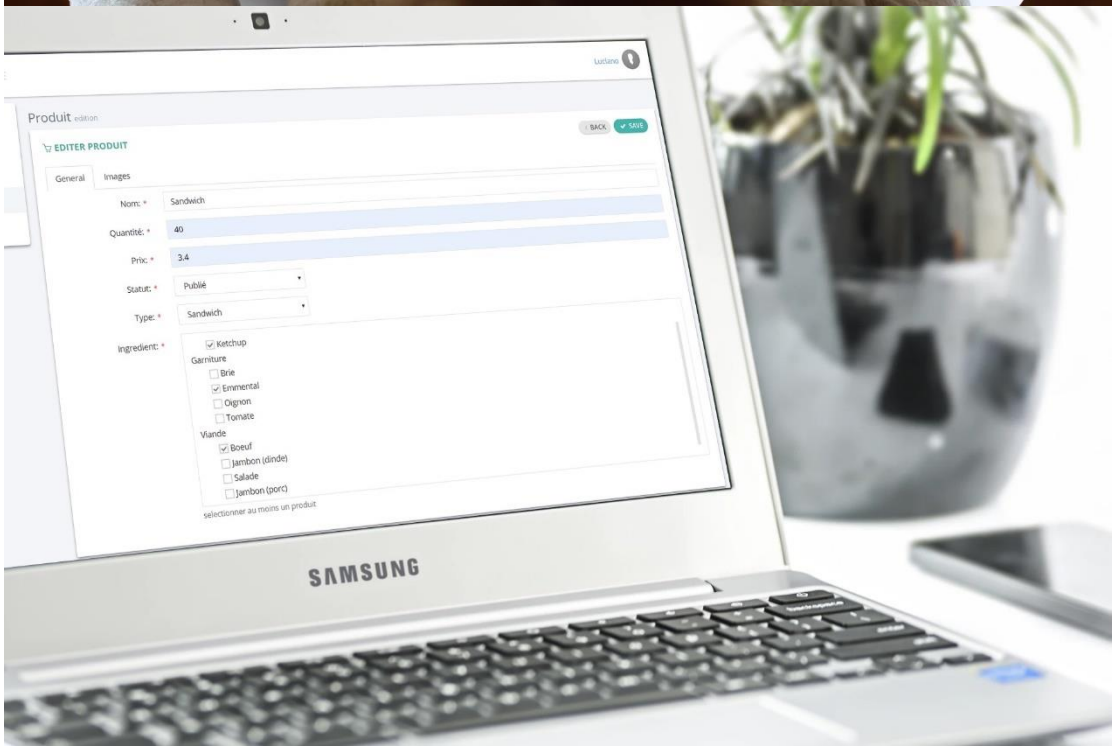
Vous pouvez voir ci-dessous diverses photos de notre application en situation.




BEEL EAT
MENU BOUTI'S


BEEL EAT


BEEL EAT


Produit edition
EDITER PRODUIT
BACK
SAVE
General
Images
Nom: *
Sandwich
Quantité: *
40
Prix: *
3.4
Statut: *
Publié
Type: *
Sandwich
Ingredient: *
Ketchup
Garniture
Brie
Emmental
Oignon
Tomate
Viande
Boeuf
Jambon (dinde)
Salade
Jambon (porc)
selectionner au moins un produit
SAMSUNG