DOSSIER DE PROJET

Titre Professionnel CDA

Sommaire

I. Liste des compétences du référentiel

N° Fiche AT	Activités types	N° Fiche CP	Compétences professionnelles
1	Concevoir et développer des composants d'interface utilisateur en intégrant les recommandations de sécurité	1	Maquetter une application
		2	Développer une interface utilisateur de type desktop
		3	Développer des composants d'accès aux données
		4	Développer la partie front-end d'une interface utilisateur web
		5	Développer la partie back-end d'une interface utilisateur web
2	Concevoir et développer la persistance des données en intégrant les recommandations de sécurité	6	Concevoir une base de données
		7	Mettre en place une base de données
		8	Développer des composants dans le langage d'une base de données
3	Concevoir et développer une application multicouche répartie en intégrant les recommandations de sécurité	9	Collaborer à la gestion d'un projet informatique et à l'organisation de l'environnement de développement
		10	Concevoir une application
		11	Développer des composants métier
		12	Construire une application organisée en couches
		13	Développer une application mobile
		14	Préparer et exécuter les plans de tests d'une application
		15	Préparer et exécuter le déploiement d'une application

II. Résumé du projet

In order to validate the competences for the validation of the title of designer developer of application we have them to realize project in different languages of programming. Partly in group and partly individually, this project is the result of a learning during the training. The client and fictitious company is called "Nesti", it is a reseller of recipe ingredients.

We have then to code in Java, javascript, Android, PHP, and HTML-CSS. For the administrative part we designed the application (mockup, storyboard) from a request. Then we realized a program coded in java which allows the management by an application. And a web application in PHP "from scratch" that manages more or less the same features. For this part we used BootStrap, elements of JavaScript and some style element extracted from the web to reproduce an imposed model. To do this, I had to create a database using MySqlAdmin, managed by the Xampp server. For the client part, we were asked to design a model to meet a need. Then we coded it using the FrameWork CodeIngiter 4 with Twig. The client part uses the same database as the PHP part. Concerning the Android part, the request of the client is to create an application that is supposed to attract the user to the site, there is also a reminder that manages the ingredients. I used Android Studio to create this program.

III. Cahier des charges

1. Description du cahier des charges du projet

La demande du client est de créer quatre applications : trois de type ordinateur et une mobile. Concernant les applications de type ordinateur, celle qui gère les stocks sera codé en Java, celle qui fait le lien entre le stock et la partie client en PHP brut et pour finir l'application gérant la vitrine de l'entreprise est codée avec le cadre 'CodeIngiter' et 'Twig'.

L'entreprise Nesti est un acteur du secteur culinaire désirant revoir son processus de vente à distance. Ainsi, il doit aussi bien gérer ses stocks, sa logistique et sa vente par des plateformes interconnectées. Le projet doit également répondre aux besoins de sécurité informatique sur chaque plateforme.

Dans son fonctionnement, l'entreprise fait appel à :

- Des administrateurs
- Des chefs
- Des modérateurs
- Des clients

Les bénéfices pour ce type d'application sont multiples :

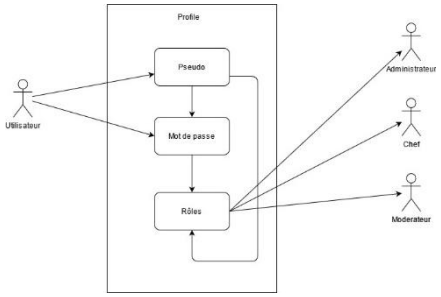
- Gérer les commandes / stock en temps réel vis-à-vis des ventes
- Permettre une vision globale aux acteurs décisionnaires

Il n'est pas à exclure de rajouter des fonctionnalités ultérieurement afin d'enrichir les interactions clients ou des éléments statistiques pour optimiser les rendements financiers. La partie sécurité pourra également être améliorée afin de palier à toutes les possibilités.

2. Les utilisateurs du projet

Les destinataires du projet sont :

- Les administrateurs (décisionnaires) et les déclinaisons chefs, modérateurs.
- Les clients



Les administrateurs

Ils n'ont pas forcément de connaissances dans le code mais ont une vue globale des fonctionnalités et ont accès à toutes les fonctionnalités.

Les modérateurs

Ils ont des fonctionnalités de gestion concernant les utilisateurs mais n'ont pas accès aux éléments de gestion Admin.

Les chefs

Ils ont accès uniquement à la gestion de la création de recettes dans le projet. Ils sont de simples utilisateurs avec des possibilités de création.

Les clients

Ils ne sont pas forcément à l'aise avec l'informatique et peuvent commettre des erreurs involontaires et ont besoin d'être guidés et avertis de la conséquence de leurs actions. Ils sont susceptibles d'être des personnes malveillantes et il faut donc procéder avec prudence dans le cas de réception de données venant de leur part. Il faut aussi penser à s'adapter aux éventuelles déficiences.

3. NESTI STOCK

Précisions

- Un article peut être fourni par plusieurs fournisseurs
- Un fournisseur peut fournir plusieurs articles
- Tous les fournisseurs ne fournissent pas tous les articles

-> L'administrateur doit pouvoir gérer la liste des fournisseurs pour un article et la liste des articles pour un même fournisseur.

- Pas besoin d'image pour la partie administrative Java
- Seul un superAdministrateur pourra ajouter un autre administrateur. Les informations, login et mot de passe, seront fournies au moment de la livraison au client.
- Dans le vocabulaire de l'entreprise Nesti, on appelle "Product" l'ensemble des ingrédients et des ustensiles.

Exemple de products :

Les ingrédients

- Banane
- Farine
- Œuf
- Lait

Les ustensiles

- Couteau
- Cuillère en bois
- Saladier
- Fouet
- Balance

Products = Ingrédients + Ustensiles

Interface graphique

- L'utilisation des JTabPannels permet de passer rapidement d'un onglet à l'autre, en restant sur le même frame.
- L'interface peut être aussi grande que possible : **plein écran**. Les administrateurs doivent pouvoir travailler toute une journée dessus. Ils doivent avoir tout à leur disposition, sur un même écran sans retour en arrière (ou presque) et profiter de toute la place d'une page.

Validation par le client

- Les documents produits dans le dossier de conception devront être validés un par un par le client.
- La maquette de l'application devra être validée écran par écran pour commencer à développer les écrans validés, en attendant la validation des autres écrans.
- Le diagramme de classe, la maquette ... etc. devront correspondre au projet développé.
- S'il y a des modifications, il faudra fournir une nouvelle version du document qui devra être à nouveau validée par le client. Il faudra alors préciser, pourquoi il y a eu un changement.

4. NESTI ADMINISTRATION

Objectif de l'application administrative :

- Édition des recettes par les cuisiniers. (Utilisateur particulier, qui s'occupe uniquement de l'édition des recettes)
- L'importation des articles depuis la base de données "Gestion de stocks"
- Gestion des utilisateurs : Possibilités de bloquer un utilisateur par des modérateurs
- Les statistiques : les notes des recettes, les ventes, les bénéfices.

Les rôles

Un utilisateur de l'application peut avoir trois rôles : Chef, Administrateur et Modérateur.
Un utilisateur peut avoir plusieurs rôles à la fois.
Si un utilisateur n'a pas le droit d'accès, il doit en être averti.

- Les administrateurs auront accès à tout
- Les modérateurs auront accès à l'onglet des **utilisateurs**
- Les chefs pâtissiers auront accès à l'onglet des **recettes.**

Il n'y a pas d'inscription. Les comptes ne peuvent être créés que par le panel Admin. Un Utilisateur lambda ne peut pas se connecter à la partie administrative.

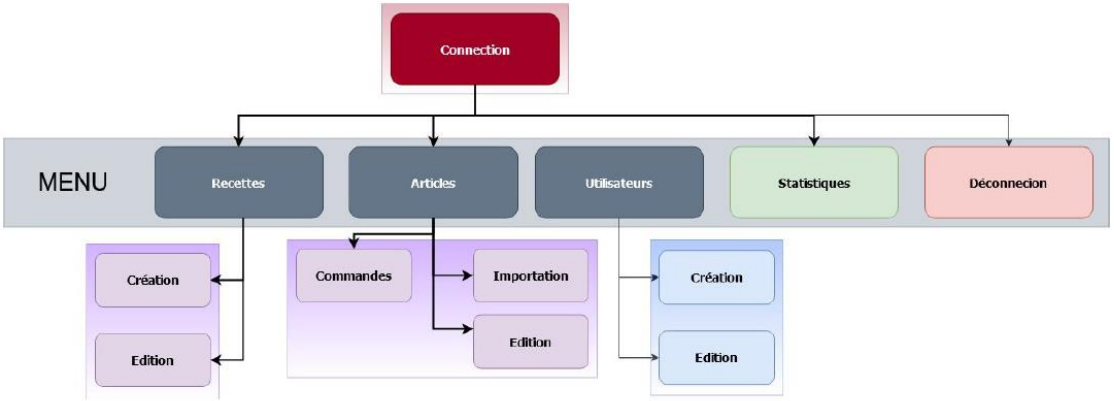


Figure 1 Architecture du site

5. NESTI CLIENT

La vente

Le site de l'entreprise Nesti présente des idées de recettes pour vendre ses produits, il propose une multitude d'ingrédients mais aussi des ustensiles à la vente.
Pour une recette donnée, une proposition d'ingrédients à acheter peut-être ajoutée dans le panier, avec les quantités nécessaires pour la recette.
Nous souhaitons une navigation souple et facile qui permette de retrouver des informations rapidement.

L'avis d'utilisateur

Chaque recette est écrite par un chef pâtissier de Nesti et sera illustrée.
Les utilisateurs peuvent mettre des notes aux recettes et ajouter un commentaire. Chaque utilisateur pourra voir la liste des recettes notées sur sa page profil.
L'objectif étant de cibler les recettes qui ont le plus de succès et d'adapter nos propositions de vente en fonction de ces choix.

Design

On sera libre de proposer une nouvelle charte graphique qui sera adaptée au thème :
Pâtisserie. Une maquette sera fournie et devra être validée avant le commencement du développement.

- On utilisera Twig, comme moteur de templating.
- Le site devra être responsive. (Minimum 3 tailles : Mobile/Tablette/Laptop)
- Le site devra réussir la validation W3c

Point d'attention

- Le site e-shop devra être en accord avec la partie administrative.
  Par exemple, lorsqu'on bloque une recette dans la partie administrative, cette recette ne devra plus être visible dans la partie e-shop de Nesti.
  Inversement, si un client passe une commande sur Nesti e-shop, il faudra que l'on puisse trouver l'information de cette commande dans la partie Admin.
- Les informations du panier seront sauvegardées, non pas dans les cookies, mais en utilisant le **local storage**.
- On supposera que tous les paiements seront valides seulement à condition que le numéro de la carte soit valide. L'**algorithme de Luhn** sera utilisé pour valider une carte bancaire.

6. NESTI MOBILE

Contexte

La société Nesti qui traditionnellement est présente sur le web, souhaite développer une application mobile, dans le but de permettre aux utilisateurs de suivre leurs recettes favorites dans leur cuisine, en regardant soit leur téléphone, soit leur tablette.

Objectifs fonctionnels de l'application

Les objectifs de cette application sont :

- Présenter la société
- Consulter les recettes proposées par catégorie
- Proposer une liste de courses interactives.

Les catégories
Les catégories des recettes sont : Facile, Traditionnelle, Saisonnière, Sans Gluten.

En cliquant sur une recette, sa fiche détaillée s'affiche avec les informations suivantes :

- Nom de la recette
- Difficulté d'exécution
- Temps de préparation
- Nombre de personnes
- Liste des ingrédients
- Étapes de préparation

Au clic d'un bouton, associé à un ingrédient, celui-ci sera ajouté à une liste de courses. L'utilisateur pourra par la suite « cocher » les ingrédients qu'il a achetés.

Liste des courses

Cette liste contient les produits dont l'utilisateur a besoin. L'utilisateur pourra « cocher » les produits qu'il a déjà achetés.

Il pourra aussi vider la liste complète d'un simple clic. A partir du menu principal, l'utilisateur pourra :

- Rechercher une recette par son nom
- Consulter sa liste de courses
- Trouver les coordonnées du développeur
    - o Présentation
    - o Lien vers un portfolio par exemple
- Obtenir des informations sur l'Équipe". Description du titre et de la promo CDA 2020 -2021.
- Consulter la présentation du projet global de Nesti.

IV. Gestion de projet

Pour la réalisation de ce projet, j'ai collaboré avec deux autres développeurs ce qui a nécessité de mettre en place des outils de travail en collaboration. Ainsi, pour travailler sur la partie Java, j'ai utilisé un **Trello.** C'est un outil en ligne qui permet d'interagir à plusieurs. Il répond à la demande de la méthode Agile en donnant la possibilité d'organiser des réunions, ce qu'il reste à faire pour finaliser le projet.

L'application en ligne **Figma** qui utilise une interface en ligne nous a permis de réaliser une **maquette.** Nous avons procédé de même pour réaliser le **storyboard**.

Pour la charte graphique, j'ai utilisé Photoshop par confort. A l'issue de plusieurs réunions entre développeurs nous avons eu à réaliser des **rapports de réunions**.

Les différentes parties du projet se sont chevauchées et ainsi chaque développeur a choisi les tâches qu'il assumerait. Cela nous a permis de construire de façon simultanée le programme sans passer de longs moments à assembler le projet et cela à chaque mise en commun sur Github.

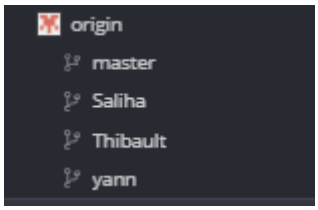


Figure 2 Vue des branche guitHub

Afin d'interagir le plus efficacement possible nous avons travaillé ensemble sur Discord dans un salon privé. Le serveur abritant ce salon a été rendu accessible uniquement aux personnes, faisant partie de la formation, qui ont été invitées par l'administrateur que j'ai secondé lors de la création. Cela nous a permis d'avoir l'esprit tranquille d'un point de vue confidentialité puisqu'en dehors des membres de la formation personne ne pouvait accéder à nos discussions.

Afin de garder une trace de notre avancée et dans le but de clarifier son évolution notre équipe a également produit un **diagramme de Gant**. Ce diagramme permet de faire une représentation des tâches réalisées et celles à effectuer et cela dans un ordre chronologique.

(Annexe « GESTION DE PROJET : JAVA »).

Pour la partie PHP, le travail étant individuel je n'ai pas utilisé d'outil collaboratif par contre j'ai mis en place un Trello afin de garder une note des éléments qui me restaient à développer ou à débugger. A l'issue du projet, il a fallu produire un plan de qualité, basé sur la demande du client.

Yann Delperie

Ref	Détails	Preuves	Dev	Client
1	Le site est accessible en ligne.	*lien*	1	1
2	Les urls sont réécrites	*htaccess*	1	1
3	L'interface est conforme à la maquette de l'application et au dossier de conception technique		1	1
4	Validation du HTML et CSS par W3C	*captures*	1	1
5	Les pages web s'adaptent à la taille de l'écran et sont fluides : - 1400 x 900 - 1280 x 800 - ipad (768 x 1024)	*test sur firefox*	1	1
6	Les bonnes pratiques de développement sont respectées.	*texte*	0	0
7	Le code source des composants est documenté ou auto-documenté		1	0.5
8	Le MLD est avec Workbench.	*L3-3*	1	1
9	Les classes du modèle et des entités sont programmés en Object		1	1
10	Les requêtes sont préparées	*capture du code*	1	1
11	Les mots de passe seront cryptés en base de données.	*capture de la bdd*	1	1
12	L'application applique le design pattern MVC.	capture de l'architecture des fichiers	1	1
13	Création d'utilisateur dans la base de données avec des accès restreints (si c'est possible en distant)	seulement en local si pas possible en distant	0	0

Pour la partie CodeIgniter et afin de répondre à une demande de qualité j'ai eu à réaliser un plan de test. Celui-ci comportait un test sur les éléments graphiques. Le but étant de permettre aux personnes déficientes visuelles de pouvoir interagir sans trop de contraintes. De plus, il a été nécessaire de contrôler la partie Html et css par la validation W3C. Pour les éléments de contraste et d'accessibilité, j'ai utilisé ColorContrast Accessibility, web Accessibility.

(Annexe « PLAN DE TEST CODEINGITER »).

V. Spécifications fonctionnelles du projet

1. NESTI STOCK

Pour réaliser cette partie nous avons utilisé StarUML, cet outil permet de créer des profils utilisateurs, des diagrammes, il nous a permis d'établir de nombreux éléments de conception que nous allons aborder.

Au niveau de la conception du projet Nesti Stock deux cas principaux d'utilisation sont présents (Admin et Super-Admin).

Super-Admin :

Diagramme de cas généralisé pour un Super Administrateur

Admin :

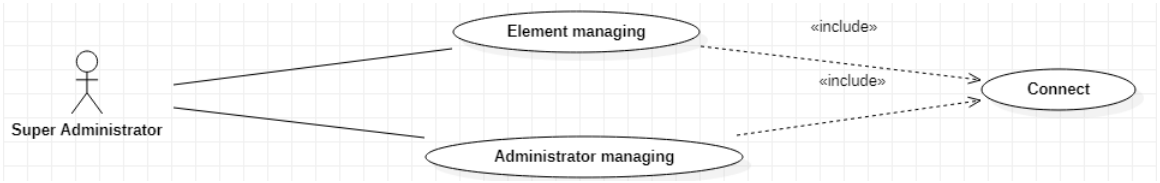


Diagramme de cas généralisé pour un Administrateur

La différence entre Super administrateur et administrateur « simple » réside seulement dans la possibilité de la gestion des administrateurs. Pour la suite des « cas utilisateurs » on étudiera seulement les administrateurs.

Eléments :

Ici, les éléments sont organisés en 4 parties :

- Produit,
- Article,
- Commande,
- Fournisseur.

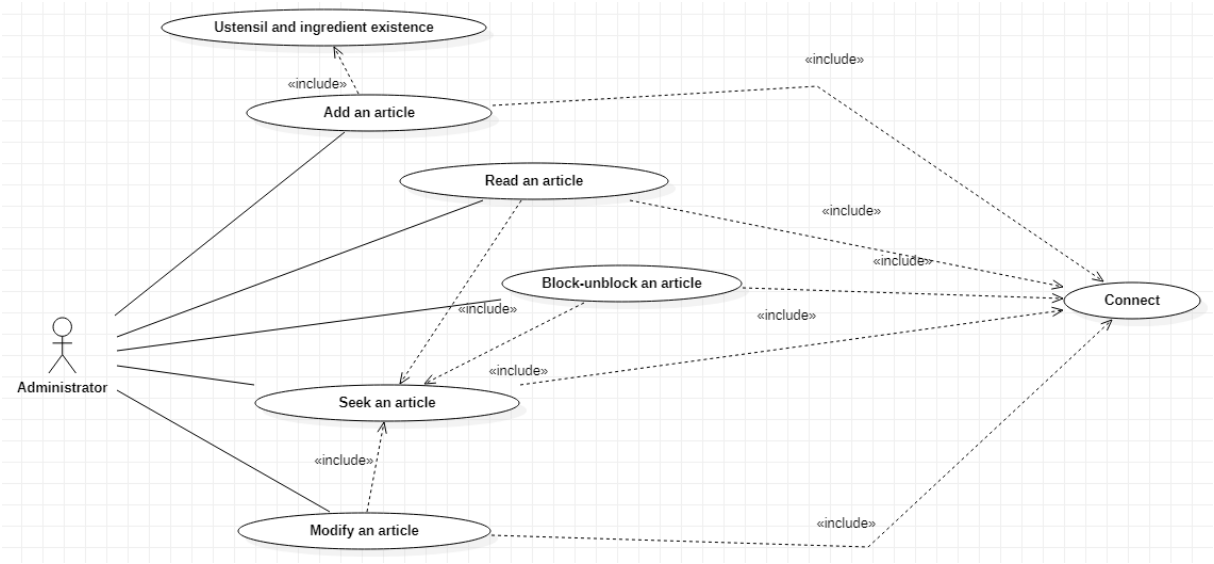


Diagramme de cas utilisateur pour la gestion d'un article

Pour expliquer ce diagramme, il faut partir de l'administrateur. Celui-ci peut ajouter un article, cela inclut ustensile et ingrédient. Il peut accéder aux données qui correspondent à un article ou faire une recherche. Il peut par ailleurs modifier ce même article avec une recherche également. Et éventuellement bloquer un article après une recherche. Dans chacune des actions la connexion est nécessaire.

Nous avons également réalisé un diagramme de cas et cela dans le même esprit pour les différentes configurations d'un fournisseur, d'un produit et d'une commande.

Diagramme de classe

Le schéma ci-contre est l'un des pivots de la conception. Il représente d'une façon statique le système à développer. Chaque classe décrit les données et les traitements dont elle est responsable, pour elle et les autres classes. Cela permet au client de visualiser rapidement l'ensemble des composants d'une classe.

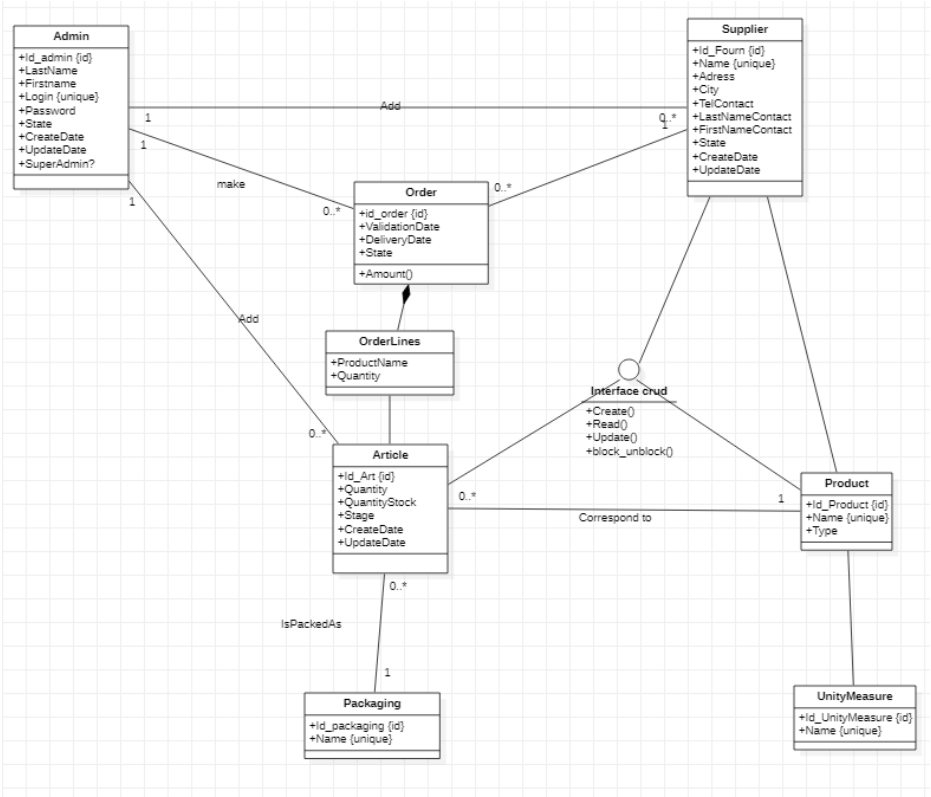


Diagramme de classe Métier

Dictionnaire des données

Ce catalogue de données représente une liste de références nécessaires à la conception de la base de données relationnelles. Il rend explicite aussi les termes employés ultérieurement dans la base de données ainsi que la nature de l'élément et sa longueur. L'exemple donne rapidement un aperçu du rendu ou de la façon dont il pourra être traité.

Donnée	Type	Longueur	Code BDD	Exemple
Admin Identifiant	*COUNTER*		Id_Admin	12
Admin Last Name	*VARCHAR*	50	LastName	Smith
Admin FirstName	*VARCHAR*	50	FirstName	Joe
Admin Login	*VARCHAR*	50	Login	JSmith1234
Admin Password	*VARCHAR*	50	Password	••••••
Admin State	*VARCHAR*	50	State	Unblocked
Admin Creation Date	*DATETIME*		CreatingDate	02/11/2020
Admin Update Date	*DATETIME*		UpdateDate	
Is Admin SuperAdmin	*BOOLEAN*		IsSuperAdmin	Yes/No
Supplier Identifiant	*CONTER*		Id_Supplier	20
Supplier Name	*VARCHAR*	50	Name	Fruity
Supplier Adress	*VARCHAR*	50	Adress	123 John St
City	*VARCHAR*	50	City	London
Supplier Contact First name	*VARCHAR*	50	ContactFirstName	John
Supplier Contact Last name	*VARCHAR*	50	ContactLastName	Smith
Supplier Contact number	*VARCHAR*	*50*	*ContactNumber*	*0102030405*
Supplier State	*VARCHAR*	50	State	Unblocked
Supplier Creating Date	*DATETIME*		CreatingDate	04/05/2020
Supplier Update Date	*DATETIME*		UpdateDate	
Order reference	*VARCHAR*	50	Id_Order	N°987654321
Order Validation Date	*DATETIME*		ValidationDate	
Order Delivery Date	*DATETIME*		DeliveryDate	
Order State	*VARCHAR*	50	State	Valide
Order Line Quantity	*INT*	15	Quantity /	
Unit Identifiant	*COUNTER*		Id_UnitMeasure	
Unit Name	*VARCHAR*	20	Name	Kg / L

Donnée	Type	Longueur	Code BDD	Exemple
Packaging Identifiant	*COUNTER*		Id_Packaging	
Packaging Name	*VARCHAR*	50	Name	
Product Identifiant	*COUNTER*		Id_Product	
Product Name	*VARCHAR*	50	Name	
Product Type	*VARCHAR*	20	Type	
Product State	*VARCHAR*	20	State	
Article reference	*VARCHAR*	50	Id_Article	
Article Quantity	*DECIMAL*	7,2	Quantity	
Article Stock Quantity	*INT*	10	QuantityRealStock	
Article State	*VARCHAR*	50	State	
Article 's Creation Date	*DATETIME*		CreatingDate	
Article 's Update Date	*DATETIME*		UpdateDate	
Buying Price	*DECIMAL*	7,2	Buying_price	
Price Update Date	*DATE*		PriceUpdateDate	

Dictionnaire des données

Matrice de dépendances fonctionnelles

La matrice de dépendances fonctionnelles met en relation les éléments avec les tables à venir, elle permet de définir les clefs primaires et lien qui se feront entre les éléments. C'est une représentation statistique de l'ensemble des données manipulées ainsi que des relations entre ces données. Dans notre matrice les clefs primaires sont identifiables par un symbole alors que les éléments associés le sont par un numéro. Les constituants de clefs étrangères sont identifiables lorsqu'un élément est contenu dans plusieurs colonnes.

	Admin	Supplier	Article	Product	Packaging	Order	Unit
	Id_Admind	Supplie	Id_Article	d_Produc	Id_Packaging	Id_Order	Id_unit
Id_Admin	*						
Admin Last Name	1						
Admin FirstName	1						
Admin Login	1						
Admin Password	1						
Admin State	1						
Admin Creation Date	1						
Admin Update Date	1						
Is Admin SuperAdmin	1						
Id_Supplier		*					
Supplier Name		1					
Supplier Adress		1					
Supplier Contact First name		1					
Supplier Contact Last name		1					
Supplier Contact number		1					
Supplier State		1					
Supplier Creating Date		1					
Supplier Update Date		1					
Order reference (Id_Order)	1	1				*	
Order Validation Date						1	
Order Delivery Date						1	
Order State						1	
Order Line QUantity						1	
Unit Identifiant							
Unit Name							1
Packaging Identifiant					*		
Packaging Name					1		
Product Identifiant				*			
Product Name				1			
Product Type				1			1
Product State				1			
Id_Article			*				
Article Quantity			1				
Article Stock Quantity			1				
Article State			1				
Article 's Creation Date			1				
Article 's Update Date			1				
Buying Price		1	1				
Price Update Date		1	1				

Matrice fonctionnelle

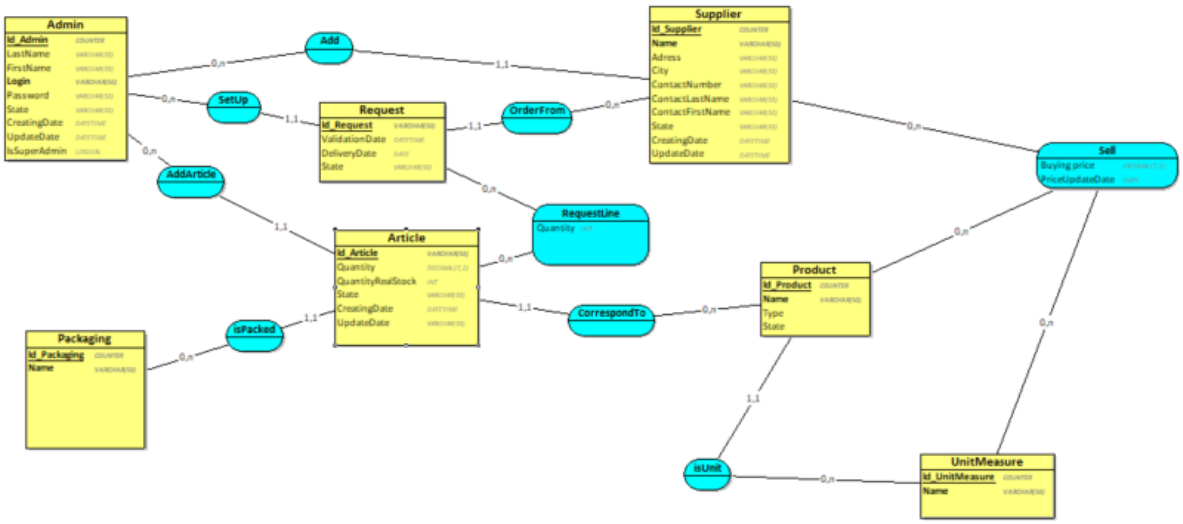
Modèle Conceptuel de Données et Modèle Logique de Données

Grâce à tous ces éléments, nous avons pu créer un MCD sur **Looping**. Celui-ci finalise la conception et les interactions entre les différentes tables. C'est un des points critiques qui nécessite du temps afin de ne pas avoir à faire la base de données.

Dans le MCD on a différents types d'éléments, les entités qui regroupent un ensemble d'objets de même nature. Ces entités expriment un type, une classe et doivent pouvoir être identifiées par leur identifiant.

Puis viennent les relations, il s'agit d'une association de même nature entre deux ou plusieurs occurrences d'entités. Une relation peut avoir aussi des propriétés (ou attributs).

Et enfin les cardinalités, elles traduisent la participation des occurrences d'une entité aux occurrences d'une relation. Elles sont une composition avec les termes « 0, 1 et n », elles sont composées par paire de termes. L'un désigne une entité de la relation et l'autres celle qui lui est associée. Les possibilités sont restreintes à (0, 1) (0, n) (1, 1) (1, n).



MCD

Looping intègre également un générateur de modèles logiques de données (MLD).

Les clés primaires sont en gras et soulignées. Les clés étrangères sont en bleu.

Admin = (Id_Admin *COUNTER*, LastName *VARCHAR(50)*, FirstName *VARCHAR(50)*, Login *VARCHAR(50)*, Password *VARCHAR(50)*, State *VARCHAR(50)*, CreatingDate *DATETIME*, UpdateDate *DATETIME*, IsSuperAdmin *LOGICAL*);
Supplier = (Id_Supplier *COUNTER*, Name *VARCHAR(50)*, Adress *VARCHAR(50)*, City *VARCHAR(50)*, ContactNumber *VARCHAR(50)*, ContactLastName *VARCHAR(50)*, ContactFirstName *VARCHAR(50)*, State *VARCHAR(50)*, CreatingDate *DATETIME*, UpdateDate *DATETIME, #Id_Admin*);
Request = (Id_Request *VARCHAR(50)*, ValidationDate *DATETIME*, DeliveryDate *DATE*, State *VARCHAR(50), #Id_Supplier, #Id_Admin*);**
UnitMeasure = (Id_UnitMeasure *COUNTER*, Name *VARCHAR(50)*);
Packaging = (Id_Packaging *COUNTER*, Name *VARCHAR(50)*);
Product = (Id_Product *COUNTER*, Name *VARCHAR(50)*, Type , State *, #Id_UnitMeasure*);
Article = (Id_Article *VARCHAR(50)*, Quantity *DECIMAL(7,2)*, QuantityRealStock *INT*, State *VARCHAR(50)*, CreatingDate *DATETIME*, UpdateDate *VARCHAR(50), #Id_Packaging, #Id_Admin, #Id_Product*);**
RequestLine = (*#Id_Article, #Id_Request*, Quantity *INT*);**
Sell = (*#Id_Supplier, #Id_Product, #Id_UnitMeasure*, Buying_price *DECIMAL(7,2)*, PriceUpdateDate *DATE*);**

Convention de nommage

Pour la partie code en JAVA, les conventions de nommage seront :

- Package : Le nom du package doit être en minuscule (ex : project, java …).
- Classes : Les classes commencent par une lettre majuscule et doivent être de type « Nom » (ex : Button, Listener, Object …).
- Méthodes : Les méthodes doivent commencer par une lettre minuscule et être de type « Verbe » (ex : create(), check() …).
- Variables : Les variables doivent commencer par une lettre minuscule (ex : name, number …).

- CamelCase : Si le nom d'une classe/méthode/variable est composé de deux mots ou plus, chaque mot suivant doit commencer par une lettre majuscule (ex : firstName, checkPassword() …).
- Global : Code et commentaires à écrire en anglais.

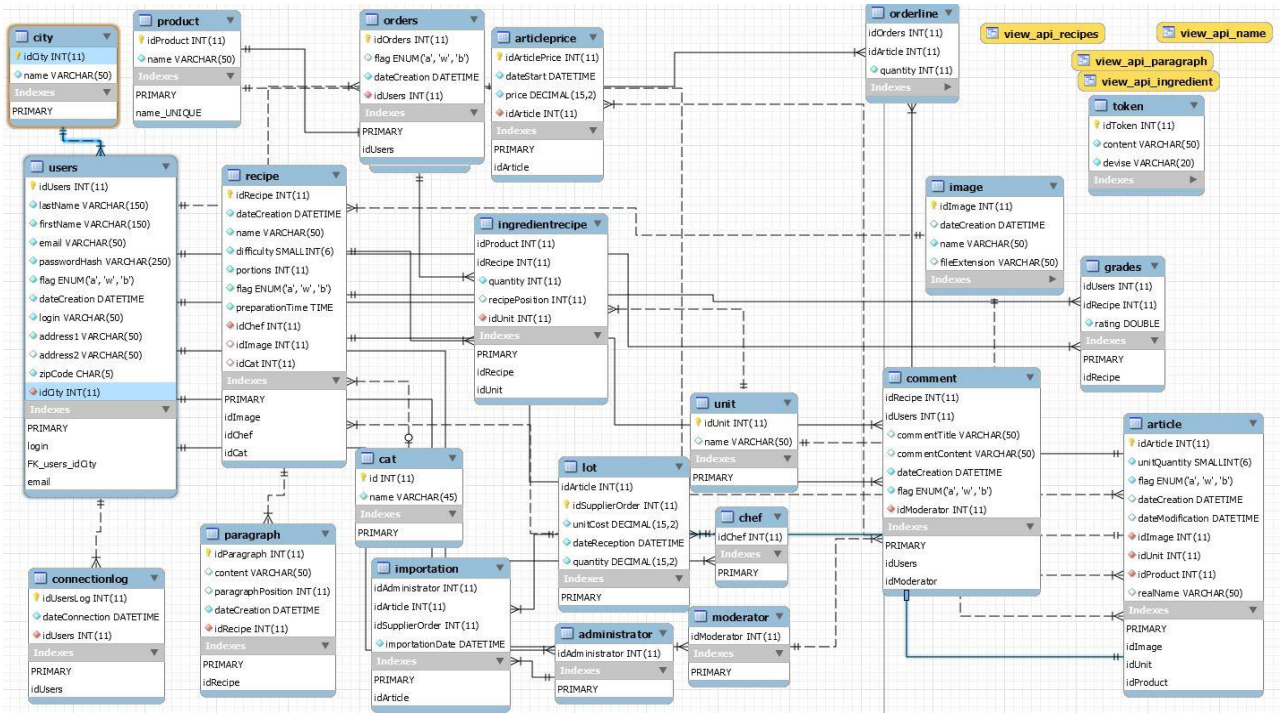
Pour la partie Base de données SQL, les conventions de nommages seront :

- Ne pas utiliser les mots réservés (date, delete, add …).
- Ne pas utiliser de caractères spéciaux.
- Eviter les majuscules, privilégier les underscores pour l'utilisation de deux mots (ex : « date_inscription » plutôt que « DateInscription »).
- Eviter l'utilisation d'abréviation.
- Noms des tables : Utiliser un nom représentatif du contenu, utiliser un seul mot si possible, préfixer les noms des tables (ex : user_data, company_data …).
- Noms des colonnes : Préfixer toutes les colonnes de la même façon pour chaque table (plus pratique pour les jointures).
- Global : Ecrire en anglais.

2. NESTI ADMINISTRATION

MLD

Dans cette partie du projet il nous a été demandé de dériver un MCD donné qui a servi de base pour le projet NESTI client. Pour ce faire j'ai utilisé Workbench. Cet outil a un avantage et pas des moindres : il permet d'exporter (et d'importer) directement les tables du serveur local.



Réalisation du MLD

3. NESTI CLIENT

La maquette

La vision que j'ai de l'application client dans le cadre du projet est davantage axée sur une application simple pour une utilisation sur tablette en cuisine. Pour cela j'ai pris la décision de créer des formes assez importantes pour faciliter la sélection (les doigts n'étant pas forcément propres au moment de la consultation...). Afin de faciliter la visibilité, le site est consultable sans connexion. Le client peut alors voir les différentes recettes, leurs compositions et les commentaires qui leur sont associés. Il peut également accéder à un onglet de suggestion, ainsi qu'à un marché. Au cas où il se connecte, il peut ajouter un commentaire sur une recette et procéder à des achats d'articles qui seront visibles dans son panier. Ensuite, il pourra procéder au paiement. Initialement, la maquette possédait des couleurs qui ont dû être modifiées afin de rendre l'interface moins criarde. J'ai préféré un système de « card » à une longue liste de mots car la navigation s'en trouve plus agréable.

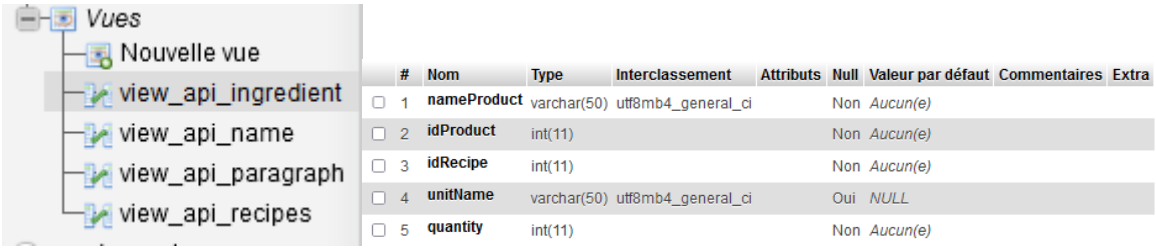
La base de données

Pour ce projet nous avons dû adapter la base de données afin d'ajouter une table de catégories. J'ai procédé de cette manière afin de faciliter la navigation pour les clients sur le site.

Figure 3 Nouvelle table

4. NESTI MOBILE

Pour mettre en place l'API, il a également fallu ajouter des vues dans la base de données (Admin-Client). Ceci permet d'accéder directement à une requête déjà établie. Il suffit pour cela de faire appel au nom que porte la vue et de traiter les données reçues ou de les transférer via une API.

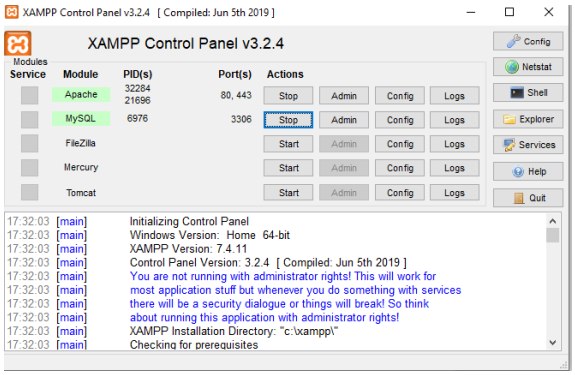


VI. Spécifications techniques du projet

1. Les contraintes techniques

Afin de réaliser l'ensemble du projet, j'ai dû utiliser Eclipse en intégrant le composant Swing. Celui-ci a permis de préparer une interface graphique. Il a fallu également pour ce projet mettre en place

Xampp en local afin d'avoir un serveur local, et de faire fonctionner le langage php en local. Ainsi, c'est en partie grâce à lui que j'ai pu créer une base de données locale.



Pour cette dernière j'ai utilisé la version 5.0.3 de PhpMyAdmin avec comme librairie de connexion mysql-connector-java8.0.22.

Pour le travail sur les éléments conceptuels nous avons utilisé StarUml 4.0.1 pour la gestion des diagrammes, Looping pour la réalisation de MLD.

J'ai également utilisé Visual Studio Code 1.56 afin de coder les parties administratives et clients. Afin de faciliter la création les éléments de style j'ai choisi d'utiliser la collection d'outils Boostrap 4.1.1 et Tailwind 2.1.4. J'ai de même importé une librairie JQuery afin de l'utiliser pour rendre le site plus dynamique et Toastui afin de pouvoir élaborer des graphiques.

Dans le projet, j'utilise également du javascript, notamment dans le projet en php pour gérer de manière dynamique le style des textes qui renseigne sur la force du mot de passe.

J'utilise également la librairie Toastui-chats qui me permet d'afficher les différents graphiques.

```
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
<script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.12.9/umd/popper.min.js" i
<script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/js/bootstrap.min.js" integri
```

```
<link href="//maxcdn.bootstrapcdn.com/bootstrap/4.1.1/css/bootstrap.min.css" rel="stylesheet" id="bootstrap-css">
<link rel="stylesheet" href="<?= BASE_URL ?>public/fontawesome-free-5.15.1-web/css/all.css">
<link rel="stylesheet" href="<?= BASE_URL ?>public/css/styleConnection.css">
<link rel="stylesheet" href="<?= BASE_URL ?>public/css/styleNavigation.css">
<link rel="stylesheet" href="<?= BASE_URL ?>public/css/styleStats.css">
<link rel="stylesheet" href="<?= BASE_URL ?>public/css/recipes.css">
<link rel="stylesheet" href="<?= BASE_URL ?>public/css/toastui-chart.min.css">
<link rel="stylesheet" href="<?= BASE_URL ?>public/css/importation.css">
```

Pour la partie ADMIN

```
<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/5.15.2/css/all.min.css"
<link href="https://unpkg.com/tailwindcss@^2/dist/tailwind.min.css" rel="stylesheet"/>
<link href="https://fonts.googleapis.com/css?family=Raleway:100,100i,200,200i,300,300i,400,400i,500,500i,
<link href="https://fonts.googleapis.com/css?family=Lora:400,400i,700,700i" rel="stylesheet"/>
<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css" integ
<link rel="stylesheet" href="{{base_url('css/style.css')}}"/>
```

Pour la Partie client

Pour cette partie j'ai utilisé Workbench 8.0 afin de modéliser le MCD fourni par le client en UML graphique.

Dans le projet Client, il nous a été demandé de coder avec Framework, j'ai utilisé CodeIgniter v4 ainsi que Twig v3 comme moteur de templating.

Enfin, la forme mobile du projet a été codée avec Android studio 4.13, environnement dans lequel j'ai dû importer la version Pixel 3a API 19 et Pixel 3a API 30 afin de faire des simulations sur des versions différentes. Ces versions sont considérées comme plus stables et sont compatibles Android 4.

Il a fallu également développer une API depuis le site Nesti client afin de sécuriser la communication, j'ai mis en place un token afin d'authentifier l'origine de la connexion. J'ai également mis en place un certificat afin d'initialiser les étapes de mise à disposition utilisateur.

J'ai eu également à disposition un serveur que j'ai connecté avec Filezilla en version 3.51

J'ai mis en place les conventions de nommage suivantes :

En programmation :

- Noms des classes, variables, méthodes en anglais
- Nom des classes, variables, méthodes en Camel Case
- Commentaires en anglais.

En base de données :

- Les tables et noms de champs en anglais
- Les noms de colonnes en Camel case pour Nest-Admin, client et Android et des « _ » pour Stock.
- Toutes les tables sont en minuscule.

2. Les composants d'accès aux données

Dans le cadre de projet de gestion de stock, il a fallu intégrer le plugin mysql connector afin de rendre possible la connexion avec la base de données. Une fois installé, il faut créer un modèle qui va gérer la connexion. Pour ce faire, il faut lui fournir un nom, un mot de passe, le nom de la base de données et l'adresse IP du serveur.

Pour le projet administrateur et afin de se connecter à la base de données j'utilise la classe Connection.php.

```php
class Connection
{

  private static $pdo = null;

  //=============
  // getPdo
  //=============
  /**
   *
   *  launch startConnexion
   *
   */
  public static function getPdo()
  {
    if (self::$pdo == null) {
      self::startConnexion();
    }

    return self::$pdo;
  }
  //=============
  // startConnexion
  //=============
  /**
   *
   *  Give the connection to DB if information is correct
   *
   */
  public static function startConnexion()
  {
    self::$pdo = new PDO(DSN, USERNAME, PWD, [PDO::ATTR_PERSISTENT => true]);
  }
}
```

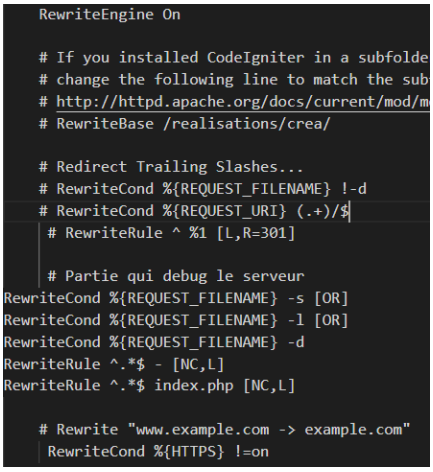
Pour lancer une session il faut au minimum définir les éléments suivants dans un fichier config :

- Le Domain name système ou DNS doit être de la forme suivante :
  mysql : dbname=xxxx ; host = localhost ;
- UserName : ce terme correspond au nom de la base de données par défaut il se nomme ('root' en local) ;
- Un mot de passe : xxxxx (vide dans un cas local) ;
- PS : ATTR_PERSISTENT permet de maintenir la connexion

J'ai créé une méthode contenue dans la classe « Connection » j'appelle celle-ci dans index.php.
Dans la méthode, j'ai pu utiliser l'extension PDO afin de lancer la connexion avec le serveur.

Pour le projet CLIENT, la connexion à la base de données se gère dans le fichier « Database.php » situé dans app/Config. Il faut comme précédemment entrer les informations du serveur. Il faut également modifier le fichier « App.php » pour la base Url.

Les fichiers «.env » et Htacces sont assez particuliers à modifier car ils ont un langage qui leur est propre. Pour le «.env » j'ai changé l'environnement afin d'accéder aux informations du développeur et j'ai défini l'accès à la base de données. Pour le « Htaccess » situé dans public j'ai eu des difficultés lors du passage sur le serveur aussi bien en stage que pour le projet. Après de nombreux tests et un retour sur une page blanche « not Found » j'ai compris qu'il fallait modifier les conditions d'accès sinon après la page d'accueil, la redirection échouait.



```
    RewriteEngine On

    # If you installed CodeIgniter in a subfolde
    # change the following line to match the sub
    # http://httpd.apache.org/docs/current/mod/m
    # RewriteBase /realisations/crea/

    # Redirect Trailing Slashes...
    # RewriteCond %{REQUEST_FILENAME} !-d
    # RewriteCond %{REQUEST_URI} (.+)/$
     # RewriteRule ^ %1 [L,R=301]

     # Partie qui debug le serveur
RewriteCond %{REQUEST_FILENAME} -s [OR]
RewriteCond %{REQUEST_FILENAME} -l [OR]
RewriteCond %{REQUEST_FILENAME} -d
RewriteRule ^.*$ - [NC,L]
RewriteRule ^.*$ index.php [NC,L]

    # Rewrite "www.example.com -> example.com"
     RewriteCond %{HTTPS} !=on
```

Pour l'application Nesti-MOBILE je n'établis pas de connexion à la base de données distante. Par contre, je crée une base de données embarquée (sqlLite) qui stockera les éléments contenus dans le panier. Pour récupérer les informations des articles, j'appelle une Api pour récupérer les données de CodeIgniter.

3. Le principe du développement en MVC ou le design pattern

Dans la réalisation de la structure du site Admin, j'ai pris en compte le Modède Vue Contrôleur fourni afin que l'architecture système corresponde au modèle. De ce fait, j'ai mis en place les Models dans leurs dossiers. Ces fichiers contrôlent les données envoyées au serveur, ils sont constitués de fonctions qui intègrent des requêtes. Ces dernières établissent la demande à la base de données.

Dans la seconde partie de la structure, il y a les Contrôleurs, qui permettent, comme leur nom l'indique de contrôler les données. C'est-à-dire qu'ils reçoivent les différentes données venues des autres fichiers. Ils redirigent également vers les vues une fois les opérations effectuées.

Concernant les derniers éléments de ce principe qui s'appelle les vues, il est important de dire qu'elles sont la partie visible du site. Les vues regroupent le style, le texte et l'architecture de la partie visible du site.

Cette architecture se retrouve dans le Framework également mais elle possède de plus des éléments de sécurité et de contrôle d'accès.

VII. Réalisations du candidat

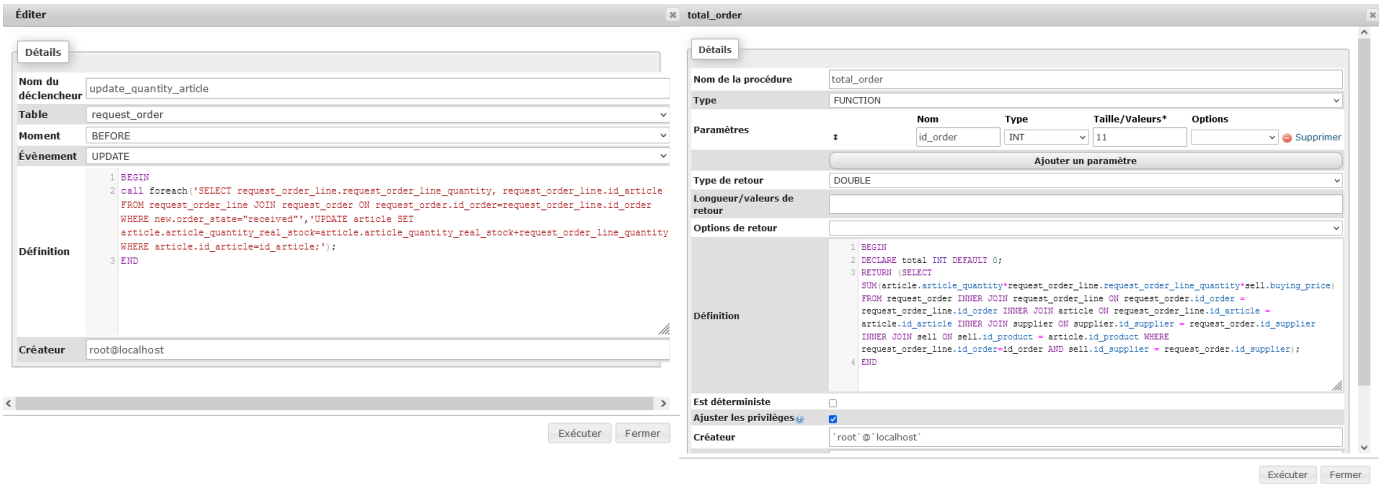
1. NESTI STOCK

A. Aspect graphique

Afin de répondre aux besoins du client nous avons essayé de créer une interface simple qui répond à la problématique de la gestion de stock. Pour rendre l'application moins « brut » nous avons fait le choix de colorer l'interface avec des tons plutôt sombres. Pour cela nous avons choisi de reprendre les couleurs initiales de l'entreprise et d'obtenir ainsi un logiciel unique en design. Nous avons proposé cette version au client mais il a demandé impérativement de ne pas afficher d'image dans les onglets fonctionnels. La taille bien que plus petite que dans le cahier des charges a été cependant acceptée car la composition restait simple.

B. La base de données

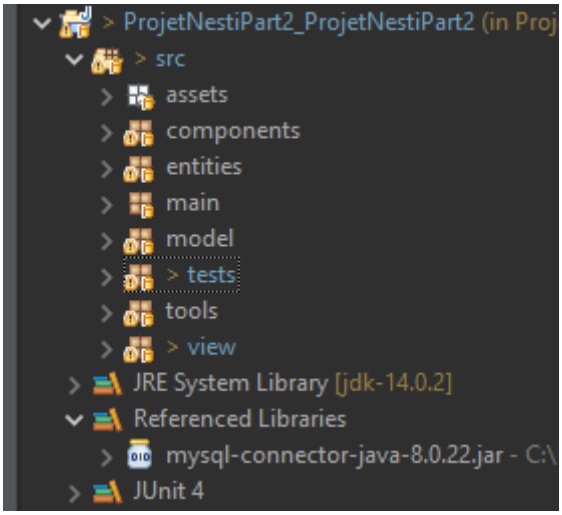
Une fois la maquette validée par le client nous avons mis en place la base de données. Pour cela nous avons utilisé Looping, un logiciel permettant de générer le code SQL en plus du MLD associé.



Plus tard dans le développement, nous avons adapté la base de données pour générer une procédure simple qui sélectionne un fournisseur. Et une qui met à jour l'article quand l'état de la commande est reçu. Nous avons également mis en place une fonction qui crée selon l'identifiant fourni, le prix total des articles en fonction du prix unitaire et des quantités commandées.

C. Structure d'une application desktop

Afin de concrétiser la partie code de ce projet nous avons mis en place des éléments de code au sein des composants générés par Window Builder (Swing) et créé une architecture en couche. Nous avons également créé des classes afin de gérer les composants




```
public class TextField extends JTextField {
    private String nameField;

    public TextField(String name, int x, int y, int L, int l) {

        this.nameField = name;
        this.setBounds(x, y, L, l);
        this.setText("");
        this.setColumns(10);
        this.setBackground(new Color(255, 222, 173));
        this.setForeground(Color.BLACK);
        this.setFont(new Font("Rockwell Nova", Font.PLAIN, 14));
        this.setDisabledTextColor(Color.WHITE);

    }
```

Figure 4Impression Ecran TextField.jav

séparement des classes principales. Ainsi, on définit dans ces classes les propriétés de l'ensemble des TextField (champs d'entrées) directement dans le constructeur de la classe au lieu de les préciser pour chaque TextField.

```
TextField tfPackagingQuantity = new TextField("Quantity", 304, 118, 50, 20);
this.add(tfPackagingQuantity);

TextField tfProductUnit = new TextField("Quantity", 364, 118, 50, 20);
this.add(tfProductUnit);

TextField tfQuantity = new TextField("Quantity", 424, 118, 100, 20);
this.add(tfQuantity);

TextField[] orderTextField = { tfPackagingQuantity, tfProductUnit, tfQuantity };
```

Figure 5 Impression Ecran OrderPanel.java

Ainsi, on définit dans ces classes les propriétés de l'ensemble des TextField (champs d'entrées) directement dans le constructeur de la classe au lieu de les préciser pour chaque TextField.

Certaines données sont affichées dès la connexion à l'application, d'autres sont mises à jour au clic sur l'onglet grâce à l'utilisation d'un listener (écouteur d'évènement) sur le JTabbedPane (conteneur des onglets. Cette fonctionnalité est créée avec l'élément positionner sur Window Builder. Il nous a fallut donc juste remplir son action.

```java
package view;

import java.util.ArrayList;

public class TabbedPaneChangeListener implements ChangeListener {
    boolean supplier = false;
    boolean product = false;
    private QueryAdmin queryAdmin = new QueryAdmin();
    private QueryProduct queryProd = new QueryProduct();
    private QueryArticle queryArt = new QueryArticle();
    @Override
    public void stateChanged(ChangeEvent e) {

        if (e.getSource() instanceof JTabbedPane) {
            JTabbedPane pane = (JTabbedPane) e.getSource();

            if (pane.getSelectedIndex() == 2) {
                SupplierPanel.clearAndEnableFalse();
                SupplierPanel.clearTable();

                try {
                    // create the list of supplier from the database
                    SupplierPanel.updateListSupplier();
                    // create the list of products from the database
                    SupplierPanel.updateListProduct();

                } catch (Exception e1) {

                    e1.printStackTrace();
                }

            } else if (pane.getSelectedIndex() == 3) {

                OrderPanel.clearTable();
                OrderPanel.clearAndEnableFalse();

                try {
                    OrderPanel.listOfOrder();
                    OrderPanel.listOfPackaging();
                } catch (Exception e1) {

                    e1.printStackTrace();
                }
```

Au moment du clic sur l'onglet « Supplier », la liste des fournisseurset la liste des produits sont mis à jour.

Impression EcranTabbedPaneChangeListener.java

```java
/**
 * Profil
 */
ProfilePanel panelProfil = new ProfilePanel();

TabbedPane.addTab("Profil", new ImageIcon(Frame.class.getResource("/assets/Profil.jpg")), panelProfil, null);

/**
 * Supplier
 */

SupplierPanel panelSupplier = new SupplierPanel();

TabbedPane.addTab("Supplier", new ImageIcon(Frame.class.getResource("/assets/Supplier.jpg")), panelSupplier,
        null);

/**
 * Order
 */

OrderPanel panelOrder = new OrderPanel();

TabbedPane.addTab("Order", new ImageIcon(Frame.class.getResource("/assets/Order.jpg")), panelOrder, null);
```

Figure 6 Impression Ecran Frame.java

Les différents onglets sont créés dans la classe Frame.java, au moment de la connexion à l'application. Cela permet de découper l'application en plusieurs parties afin de pouvoir contrôler et développer de la manière la plus organisée possible.

Les constructeurs de chaque onglet sont décrits dans chaque Panel correspondant à l'onglet (ici le panel des commandes – OrderPanel) :

```java
 *
 * @throws Exception
 */
public OrderPanel() throws Exception {
    this.setBackground(new Color(213, 167, 113));
    this.setLayout(null);
    queryArticle = new QueryArticle();
    querySupplier = new QuerySupplier();
    queryProduct = new QueryProduct();
    queryPackaging = new QueryPackaging();
    queryOrder = new QueryOrder();
    querySupplierSell = new QuerySupplierSell();

    Button btnOrderLaunch = new Button("Order_Launch", 360, 20, 86, 23);
    this.add(btnOrderLaunch);

    Button btnOrderArticleLaunch = new Button("OrderArticle_Launch", 500, 62, 86, 23);
    btnOrderArticleLaunch.setName("Add");
    this.add(btnOrderArticleLaunch);

    Button btnOrderAdd = new Button("+_Order2", 544, 116, 50, 23);
    this.add(btnOrderAdd);

    Button btnOrderAdd2 = new Button("+_Order1", 705, 182, 45, 23);
    this.add(btnOrderAdd2);

    Button btnOrderMinus = new Button("-_Order", 705, 210, 45, 23);
    this.add(btnOrderMinus);

    Button btnOrderRemove = new Button("x_Order", 705, 240, 45, 23);
    this.add(btnOrderRemove);

    Button btnOrderModify = new Button("Order_Modify", 154, 371, 113, 32);
    this.add(btnOrderModify);

    Button btnOrderCreate = new Button("Order_Create", 496, 371, 113, 32);
    this.add(btnOrderCreate);

    Button[] orderButtons = { btnOrderLaunch, btnOrderArticleLaunch, btnOrderAdd, btnOrderAdd2, btnOrderMinus,
            btnOrderRemove, btnOrderModify, btnOrderCreate };
    button = orderButtons;

    Label lblOrderSearch = new Label("Search Order", 31, 8, 178, 45);
    this.add(lblOrderSearch);

    Label lblArticleSearch = new Label("Search Article", 31, 50, 178, 45);
    this.add(lblArticleSearch);
```

Figure 7 Impression Ecran OrderPanel.java

Des écouteurs d'évènement sont alors ajoutés pour récupérer les actions de l'utilisateur (ici action sur la liste des états d'une commande) :

Ici, dès lors que l'utilisateur change l'état d'une commande, une fenêtre s'affiche pour demander la confirmation de la modification. Pour cela on fait une condition qui détermine si l'on a sélectionné « create New Order » dans la liste. La condition suivante permet d'éviter des modifications si la liste est inactive.
Puis si l'état de ce que l'on a sélectionné change, on effectue le dernier contrôle.

```
/**
 * Action listener on the state combobox
 */
listOrderState.addItemListener(new ItemListener() {

    @Override
    public void itemStateChanged(ItemEvent event) {
        if (!listOrder.getSelectedItem().equals("Create New Order")) {
            if (listOrderSupplier.isEnabled()) {

                if (event.getStateChange() == ItemEvent.SELECTED) {
                    try {
                        Object source = event.getSource();
                        if (source instanceof JComboBox) {
                            JComboBox cb = (JComboBox) source;
                            String selectedItem = String.valueOf(cb.getSelectedItem());
                            if (!selectedItem.equals(activOrder.getState())) {
                                int answer = JOptionPane.showConfirmDialog(null,
                                        "Are you sure that you want to switch to" + selectedItem,
                                        "PLEASE CONFIRM", JOptionPane.OK_CANCEL_OPTION);
                                if (answer == 2) {
                                    listOrderState.setSelectedItem(activOrder.getState());
                                    ;
                                }
                            }
                        }
                    } catch (Exception e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                    }
                }
            }
```

Figure 8 Impression Ecran OrderPanel.java

Celui-ci conditionne le fait que s'il a un événement du type Jcombox on regarde la valeur de l'item et si la valeur de l'item a changé on envoie une demande de confirmation à l'utilisateur.

```
/**
 * This method is used to calcul the price
 * @param supplier
 * @param article
 * @param quantity
 * @return
 */
public static double calculTotalPrice(Supplier supplier, Article article, int quantity) {

    double TotalTotalPrice = 0;

    try {
        TotalTotalPrice = Math.round((article.getQuantity() * quantity * querySupplierSell
                .getPrice(article.getProduct().getName(), supplier.getName())) * 100.0)
                / 100.0;
    } catch (Exception e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }

    return TotalTotalPrice;
}
```

Figure 9 Impression Ecran OrderPanel.java

Cette fonction permet de récupérer le prix d'un article pour un fournisseur donné, afin d'afficher le prix total (voir impression écran précédent).

Les modèles : Ils permettent de communiquer avec la base de données (lecture, écriture) à travers des requêtes.
Les modèles sont spécifiques aux données qu'ils gèrent. Il y a donc de nombreux modèles (admin, fournisseur, article, produit …).
Toutes les requêtes sont des requêtes préparées, principalement utilisées pour deux raisons :

- Protéger leur application des injections SQL ;
- Gagner en performance dans le cas d'une requête exécutée plusieurs fois par la même session.

```java
/**
 * This method is used to create a new Supplier in the database, during the
 * register process
 *
 * @param Supplier supplier
 * @return boolean
 * @throws Exception
 */
public boolean createPrepared(Supplier supplier) throws Exception {
    boolean flag = false;
    try {
        String query = "INSERT INTO `supplier`(supplier_name, supplier_adress, supplier_city, supplier_contact_number, supplier_contact_lastname, "
                + "supplier_contact_firstname, supplier_state, id_admin) VALUES (?,?,?,?,?,?,?,?)";
        PreparedStatement declaration = accessDataBase.prepareStatement(query);

        declaration.setString(1, supplier.getName());
        declaration.setString(2, supplier.getAdress());
        declaration.setString(3, supplier.getCity());
        declaration.setString(4, supplier.getContactNumber());
        declaration.setString(5, supplier.getContactLastname());
        declaration.setString(6, supplier.getContactFirstname());
        declaration.setString(7, supplier.getState());
        declaration.setInt(8, supplier.getIdAdmin());
        int executeUpdate = declaration.executeUpdate();
        flag = (executeUpdate == 1);
    } catch (Exception e) {
        System.err.println("Erreur d'insertion utilisateur: " + e.getMessage());
    }
    return flag;
}
```

Impression Ecran QuerySupplier.java

Cette requête permet de créer un fournisseur dans la base de données.
Pour sécuriser les envois (requêtes) il faut impérativement :

- Créer un ‘try and catch’ afin d’éviter un arrêt brusque
- Créer une requête SQL
- Préparer la requête afin qu’elle soit analysée pour être comparée
- Attribuer les valeurs ou éléments d’objets (ou les récupérer)
- Exécuter la requête avec les bonnes attributions
- Créer une routine de retour.

Les entités : Les entités correspondent aux objets présents dans la base de données (article, produit, fournisseur, commandes …).
Chaque entité contient des attributs qui correspondent aux données de la table, attributs définis dans le constructeur :

```java
public class Order {
    int id;
    String state;
    Date validationDate;
    Date deliveryDate;
    Supplier supplier;
    int idAdmin;

    /**
     * Constructor
     *
     * @param int      id
     * @param Date     validationDate
     * @param Date     deliveryDate
     * @param string   state
     * @param Supplier supplier
     * @param int      idAdmin
     */
    public Order(int id, Date validationDate, Date deliveryDate, String state, Supplier supplier, int idAdmin) {
        this.id = id;
        this.state = state;
        this.validationDate = validationDate;
        this.deliveryDate = deliveryDate;
        this.supplier = supplier;
        this.idAdmin = idAdmin;
    }
```

Impression Ecran Order.java

Ces entités sont construites grâce aux données récupérées dans les modèles. Les modèles sont les éléments qui lient le code à la base de données.

```java
/**
 * This method is used to get all orders from database
 * @return ArrayList<Order>
 * @throws Exception
 */
public ArrayList<Order> listAllOrder() throws Exception {
    ArrayList<Order> listOrder = new ArrayList<Order>();
    Order ord = null;
    try {

        String query = "SELECT request_order.id_order,request_order.order_validation_date,request_order.order_delivery_date,request_order.order_state,request_order.id_admin, "
                + "supplier.supplier_name FROM request_order JOIN supplier ON request_order.id_supplier=supplier.id_supplier;";
        PreparedStatement declaration = accessDataBase.prepareStatement(query);
        ResultSet rs = declaration.executeQuery();
        /* Boucle pour parcourir les résultats */
        while (rs.next()) {
            Supplier supplier = querySupplier.createSupplierInfo(rs.getString("supplier_name"));
            ord = new Order(rs.getInt("id_order"), rs.getTimestamp("order_validation_date"),
                    rs.getTimestamp("order_delivery_date"), rs.getString("order_state"), supplier,
                    rs.getInt("id_admin"));
            listOrder.add(ord);
        }
    } catch (Exception e) {
        System.err.println("Erreur d'affichage d'ing: " + e.getMessage());
    }
    return listOrder;
}
```

Impression Ecran QueryOrder.java

Les composants : Les composants sont des classes héritées des classes principales nécessaires aux interfaces graphiques (TextField pour JTextField, Button pour JButton, Label pour JLabel …) afin d'éviter les répétitions de codes à la construction des différentes entités des vues.

D. Sécurité

Afin de sécuriser la connexion nous avons choisi de mettre en place de nombreuses 'Combobox' afin de limiter les entrées des utilisateurs et les éventuelles erreurs de saisie.

```java
/**
 *
 * @param String input
 * @return
 */
public static boolean isValidString(String input) {
    String regex = "[A-Za-z àéèêëâôîïö]{3,50}$";
    Pattern p = Pattern.compile(regex);
    if (input==null) {

        return false;
    }

    Matcher m = p.matcher(input);
    return m.matches();
}
```

Pour gérer les éventuelles erreurs de saisie nous avons mis en place des regex. Cet outil permet de limiter les entrées des utilisateurs ce qui a pour but de restreindre les erreurs de saisie. La fonction regex « IsValidString » autorise par sa structure les majuscules ainsi que les minuscules et les différents accents. Ce regex limite également la taille des éléments de chaîne de caractères entre 3 et 50. Grâce à cette fonction, on limite les entrées au strict nécessaire.

Lorsque l'on se connecte et afin de valider l'entrée dans l'application, un compte utilisateur est indispensable. Si l'on a repondu à toutes les étapes de la securité précédente une requête est alors lancée afin de vérifier que l'utilisateur existe et qu'il n'a pas été bloqué.

```
public boolean checkPassword(String username, String password) throws Exception {

    PreparedStatement declaration;
    ResultSet rs;

    boolean checkPassword = false;

    String query = "SELECT `admin_password`, `id_admin` FROM `admin` WHERE `admin_login` =? AND admin_state='Unblocked';";

    try {
        declaration = accessDataBase.prepareStatement(query);

        declaration.setString(1, username);

        rs = declaration.executeQuery();

        if (rs.next()) {

            if (BCrypt.checkpw(password, rs.getString("admin_password"))) {
                checkPassword = true;

                LoginFrame.id = rs.getInt("id_admin"); // stocker l'id de l'admin dans login frame

            }
        }

    } catch (SQLException ex) {
        Logger.getLogger(LoginFrame.class.getName()).log(Level.SEVERE, null, ex);
    }

    return checkPassword;
```

Suite à cela on compare le mot de passe entré et encodé avec celui qui est encodé dans la base de données.

Si cela correspond on renvoie « true ».

En cas de succès et uniquement dans ce cas, on pourra autoriser l'utilisateur à se connecter.

Ainsi nous comparons l'existence de la saisie utilisateur avec son mot de passe. Cela permet de fragmenter l'information retournée et d'éviter les fuites de compte.

E. Test unitaire

Afin de vérifier notre code pour l'application NestiStock nous avons réalisé des tests unitaires sur quelques fonctions. Ces tests permettent de vérifier le bon fonctionnement d'une fonction, sa logique. Il force indirectement les développeurs à avoir un code plus simple qui fonctionne sur de petite fonction. C'est pour cela qu'il est utile de remanier les fonctions afin qu'elle retourne un élément booléen. L'utilisation de la fonction intégrée assertTrue/False permet de créer une comparaison avec la sortie de la fonction étudiée et cela avec une valeur et que l'on définit. On test ainsi les entrées que l'on estime valide en vérifiant qu'elles le sont bien. Et on fait de même avec les entrées non valides.

Dans le cas où toutes les entrées ont des résultats pertinents, on obtient alors un résultat de test positif comme visible sur la capture. Sinon c'est qu'on a mal évalué les résultats ou alors que la fonction ne fait pas ce qu'elle devrait réaliser.

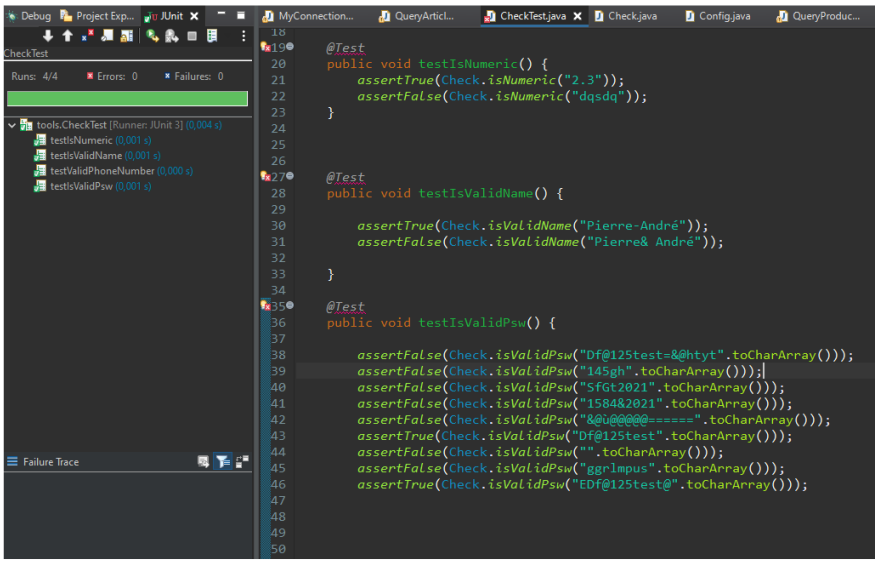


Figure 10 Capture de test unitaire sur Eclipse

2. NESTI ADMIN

A. Aspect graphique et utilisation de Boostrap

Pour la réalisation de la partie Admin, une maquette détaillée nous a été fournie. Il nous a été demandé que la réalisation soit également responsive. Ce terme est employé pour définir l'utilisation multi-plateforme d'un site. Ce qui permet à l'utilisateurs d'accéder ou il veut à son site. Le résultat est visible en partie dans REF.3 en annexe « ANNEXE PLAN DE QUALITE ».

Dans un but de faciliter la mise en place, j'ai utilisé **Boostrap**. Il a l'avantage de proposer de nombreux modules qu'il faut intégrer dans son code. Il a également des raccourcis intégrés pour gérer les éléments html. Il s'agit d'un système de grid :

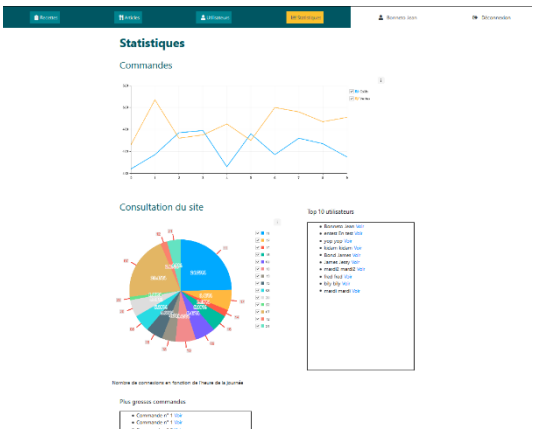
```php
<div id="login">
    <img src="public/images/logo-nesti.png" alt="logo nesti" class="logo">
    <div class="container">
        <?php if (isset($_SESSION["deconnection"])) { ?>
            <div id="login-row" class="row justify-content-center align-items-center">
                <div id="login-column" class="col-md-6">
                    <div class="alert alert-success text-center" role="alert">
                        Déconnexion réussie
                    </div>
                </div>
            </div>
        <?php
            session_unset();
            session_destroy();
        }
        ?>
        <div id="login-row" class="row justify-content-center align-items-center">
            <div id="login-column" class="col-md-6">
                <div id="login-box" class="col-md-12">
                    <form action="<?= BASE_URL . "connection" ?>" method="POST" id="login-form" class="form">
                        <h3 class="text-center text-info text-dark mb-4">Connexion</h3>
                        <div class="form-group">
                            <div class="row">
                                <div class="col-sm-3 ml-4"></div>
                                <div class="col-sm-6"><label for="username" class="text-info text-dark">Identifiant</label></div>
                            </div>
                            <div class="row d-flex justify-content-center">
                                <div class="col-sm-1 d-flex justify-content-end align-items-center"><img src="public/images/identifiant-i
                                <div class="col-sm-6"><input type="text" name="loginUser" id="username" class="form-control"></div>
                            </div>
                        </div>
```

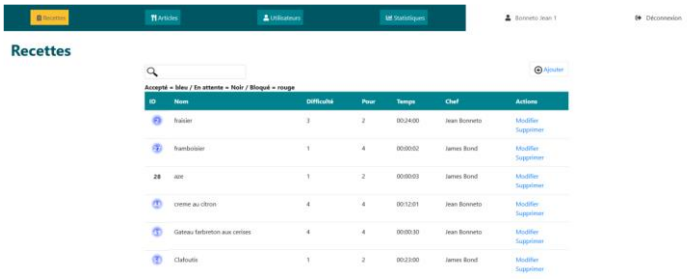
Pour faire simple, ce système découpe l'écran en douze parties auxquelles on attribue une valeur pour une taille d'écran définie. Exemple : col-md-6 signifie que l'élément prendra six colonnes sur douze pour un écran >576px sans 'md' cela serait valable pour tout écran. Bootstrap utilise également un système d'abréviations qui permet de simplifier le style. Par exemple les « mr, ml » permettent de créer rapidement des marges et les « d-flex » de repositionner certains éléments.

Ainsi de nombreux styles peuvent être mis en place dans le langage HTML surtout si l'on ajoute à cela les modules pré créés par des développeurs et disponibles sur le net. Je peux donc dire qu'il est facile d'assembler des éléments visuels afin de constituer une ébauche du rendu. Par contre j'ai perdu beaucoup de temps à faire en sorte que les modules importés ne se télescopent pas. En fait, j'ai remarqué que les développeurs de modules appellent systématiquement leurs classes (liées aux éléments de styles) de la même manière que dans tous les autres css. Je me suis retrouvé pour un exemple rapide avec des h1 composés de trois ou quatre styles différents.

Je trouve qu'il serait plus pratique d'utiliser des Id ou des noms spécifiques si d'aventure je diffusais un contenu. Donc après trois ou quatre importations sur la même page la moindre modification a été fastidieuse à corriger les premières fois. Rapidement j'ai utilisé mes propres noms de classes ccs associées à Bootstrap.

Ainsi grâce à cet outil, il a été plus simple pour moi de réaliser une interface correspondant à la demande client :





J'ai mis en place un système de validation qui informe lorsque le formulaire est correctement rentré. Sinon un message d'erreur doit apparaitre afin d'indiquer ce qui a été mal saisi.

J'ai également, pour répondre au cahier des charges, fait des tests de validation sur W3C. Ce site vérifie la validité du balisage des documents web. Si l'on se trouve dans le cas d'une page gérée par un compte, il faut aller dans le code source de la page à analyser et injecter l'ensemble du code de la page dans l'encadrer de W3C. Après plusieurs tests et des corrections j'ai obtenu un résultat positif. « ANNEXE PLAN DE QUALITE »

B. La base de données

Sur cette partie du projet notre client nous a fourni le MCD en version numérique. Pour plus de confort, car il intègre une interface graphique, j'ai utilisé workbench. C'est un outil très puissant qui met en relation les données saisies et génère un modèle visuel. Après un moment de mise en place de l'intégralité des structures j'ai mis en place les clefs primaires, secondaires et les contraintes.Ce logiciel est très intéressant car il permet également d'exporter directement sa création dans une base de données locale, sans trop de complexité. Pour cela, la commande que l'on utilise se trouve dans « DataBase-> Forward Engineer ». Pour les paramètres, une fois la connexion établie avec le localhost, il faut sélectionner ce que l'on veut exporter puis une dernière étape me donne le script qui sera inséré. Cette dernière étape est souvent cause de bug dans la création de la base de données si une erreur est présente. J'ai vérifié les clefs une fois l'insertion réussie car il peut aussi arriver que des erreurs cassent les liens à défaut de ne pas fonctionner.

C. L'utilisation de méthode asynchrone

Pour faciliter la navigation et créer des éléments dynamiques j'ai aussi utilisé du script jQuery. Il permet de rendre la navigation<>traitement plus rapide. En fait JQuery utilise les ressources du serveur afin de réaliser une tâche (script) c'est pour cela que j'ai lu (durant mes veilles) qu'il était de plus en plus déprécié. En effet, si un petit script ne consomme que peu de ressources, un script lourd et mal codé peut ralentir un serveur de façon importante.

```
$(document).ready(function () {

    var base_url = "https://127.0.0.1/www/PHP_Nesti_Site/";
    // var base_url = "https://delperie.needemand.com/realisations/PHP_Nesti_Site/";
    $('#ingAdd').click(function (e) {
        e.preventDefault();

        // Controle value of input product
        let name = $("#ingName").val();
        let error = 0;
        if (name == "" || !name.match(/^[a-zA-Z]+$/)) {
            error = 1;

            alert('Erreur de la saisi :  Produit ');
        }
        // Controle value of input qunantity
        let quant = $("#ingQty").val();
        if (quant == "" || !quant.match(/^\d+$/)) {
            error = 1;
            alert('Erreur de la saisi : quantité ');
        }
        // Controle value of input unit
        let unit = $("#ingUnit").val();
        if (unit == "" || !unit.match(/^[a-zA-Z]+$/)) {
            error = 1;
            alert('Erreur de la saisi : unité ');
        }
        if (error == 0) {
            let id_recipe = $('#ingAdd').data('id');
            $.post(base_url + "recipes/adding/" + id_recipe, { name: name, quant: quant, unit: unit }).done(function (data) {
                if (data.success == true) {
                    // create a ingredient line
                    $('#ingCtn').append(
                        '<li class="flex justify-between">'
                        + quant + " " + unit + " de " + name
                        + '    <a href="' + base_url + 'recipes/editing/' + data.recipe + '/supp/' + data.idProduct + '">Supprimer</a>'
                        + '</li>');
                } else {
                    alert("erreur ajout ingredient");
                }
            }, "json");
```

Figure 11 vue du script

Reprenons, dans le cas de l'ajout d'un ingrédient, je sélectionne la valeur contenue dans l'élément possédant l'ID 'ingName'.

Je mets en place une routine pour contrôler les éléments de saisie de chaque entrée. S'il n'y a aucune erreur, j'envoie par la méthode post au contrôleur l'id de la recette avec les données. Dans le contrôleur, je traite les données et ajoute ou utilise les éléments à envoyer. A la suite de cela et si tout c'est bien passé j'encode en format JSON les données.

```php
$modelUnit = new ModelUnit();
$isExistUnit = $modelUnit->readOneby("name", $unit);

// if the name of unit still exist return else or give id of the insered unit
if (($isExistUnit)->getName() == null && $unit != "") {
  $insertedUnit = $modelUnit->insertUnitJquery($unit);
  $isUnitid     = $insertedUnit->getIdUnit();
} else {
  $isUnitid = $isExistUnit->getIdUnit();
}
$ing = new Ingredientrecipe();

$ing->setIdRecipe($id);
$ing->setIdProduct($isProdId);
$ing->setIdUnit($isUnitid);
if ($quant != 0) {
  $ing->setQuantity($quant);
}

$model = new ModelIngredientrecipe();

// if the quantity of ingredient is not 0 insert a row of IngredientRecipe
if ($ing->getQuantity() != 0) {
  $model->insertIngredientRecipe($ing);
  header('Content-Type: application/json');
  echo json_encode(array('success' => true, 'recipe' => $id, 'name' => $name, 'quant' => $quant, 'unit' => $unit,
} else {
  header('Content-Type: application/json');
  echo json_encode(array('success' => false));
}
die;
}
```

Figure 12 Vue du contrôleur gérant les données envoyées

Dans le script, si les opérations précédentes se sont bien déroulées j'utilise les données afin de créer une réponse html visible. Cette réponse est composée de balises et de valeurs récupérées dans le JSON. En fait, on parle d'asynchrone lorsque les lignes de code ne s'exécutent pas les unes après les autres.

Retournons dans la vue du script, la requête « $.post() » va s'effectuer si l'on remplit les conditions. Le moment qui montre vraiment l'asynchronisme se situe au niveau du « .done ». L'exécution va attendre une réponse avant de se lancer alors que si le code avait été poursuivi, d'autres éléments aurait été pris en compte et exécuté

D. Sécurité et gestion des accès

Pour répondre aux besoins de l'application j'ai mis en place des objets. Ces objets sont créés à partir d'une classe, utilisés comme 'block' d'informations ils permettent de ne pas accéder directement au contenu des données transférées. Il faudrait pour cela connaître l'architecture et les attributs de l'objet.

```php
foreach ($arrayRecipes as $value) {
?>

    <tr>
        <td id="<?= $value->getIdColor(); ?>" class="font-weight-bold text-white d-flex align-items-center
        <td><?= $value->getName(); ?></td>
        <td><?= $value->getDifficulty(); ?></td>
        <td><?= $value->getPortions(); ?></td>
        <td><?= $value->getPreparationTime(); ?></td>
        <td><?= $value->getChef(); ?></td>
        <td>
            <a href="<?= BASE_URL . "recipes/editing/" . $value->getIdRecipe(); ?>">Modifier</a><br>


            <a data-toggle="modal" href="#myModal<?= $value->getIdRecipe(); ?>">Supprimer</a>
            <div class="container">
                <div class="row">
```

Figure 13 Génération de texte, url par des objets

L'objet est un outil fort sympathique puisqu'il embarque en son sein ses propres données pour peu que soient définies ses

classes et qu'on l'ait instancié. Ses attributs peuvent être de différentes formes pour autant qu'il soit défini. Il peut également être stocké dans une liste (sans altérer ses informations) lorsqu'il est nécessaire d'en manipuler plusieurs. Il est alors agréable de récupérer par des méthodes (le plus souvent auto-générées) les valeurs qu'il contient. Dans le cas d'un tableau d'objets, il faut d'abord parcourir objet par objet puis accéder aux valeurs de chacun.

Afin de sécuriser les données, j'ai également mis en place la préparation des requêtes avec l'insertion de données issues d'objets. Dans de nombreux cas, je récupère un objet ou un tableau d'objets, a minima je retourne un booléen.

```php
//=============
// UPDATE USER
//=============
/**
 *  With user objet insert value
 * and if true
 * return the new object user
 *
 */
/**
 * updateUsers
 *
 *
 * @return object
 */
public function updateUsers(Users &$user)
{
    $pdo = Connection::getPdo();
    try {
        $sql = "UPDATE users SET lastName = ?, firstName = ?, address1 = ?, address2 = ?, zipCode = ?, flag = ?,idCity = ?  where

        $stmt = $pdo->prepare($sql);

        $values = [$user->getLastname(), $user->getFirstname(), $user->getAddress1(), $user->getAddress2(), $user->getZipCode(), $

        // Execute the prepared statement
        $stmt->execute($values);
        $user = $this->readOneBy("idUsers", $user->getIdUser());
    } catch (PDOException $e) {
        die("ERROR: Could not able to execute $sql. " . $e->getMessage());
    }
    unset($pdo);
    return $user;
}
```

Figure 14 fonction qui prépare et exécute en utilisant des valeurs d'un objet

```php
$userLastname = filter_input(INPUT_POST, "userLastname", FILTER_SANITIZE_STRING);
$userFirstname = filter_input(INPUT_POST, "userFirstname", FILTER_SANITIZE_STRING);
$userLogin = filter_input(INPUT_POST, "userLogin", FILTER_SANITIZE_STRING);
$userEmail = filter_input(INPUT_POST, "userEmail", FILTER_SANITIZE_EMAIL);
$userPwd = filter_input(INPUT_POST, "userPwd", FILTER_SANITIZE_STRING);
$userAdress1 = filter_input(INPUT_POST, "userAdress1", FILTER_SANITIZE_STRING);
$userAdress2 = filter_input(INPUT_POST, "userAdress2", FILTER_SANITIZE_STRING);
$userZipCode = filter_input(INPUT_POST, "userZipCode", FILTER_SANITIZE_STRING);
$townInput = filter_input(INPUT_POST, "userTown", FILTER_SANITIZE_STRING);
$error     = 0;
```

Toutes les entrées sont systématiquement filtrées avant d'être traitées. Cette fonctionnalité intégrée nécessite de définir le type d'entrée. J'utilise également des regex afin de limiter les saisies.

```php
if (!filter_var($userEmail, FILTER_VALIDATE_EMAIL) && (!preg_match('/^[_a-z0-9-]+(\.[_a-z0-9-]+)*@[a-z0-9-]+(\.[a-z0-9-]+)*(\.[a-z]{2,3})$/', $userEmail))) {
    $data['email'] = true;
    $error = 1;
}
```

Le regex est une forme d'expression qui permet de faire correspondre ou non une expression avec le schéma logique qu'elle contient. Par exemple, dans le cas de l'adresse e-mail, il faut qu'elle soit proposée sous la forme x@x.xx . Cela protège d'un éventuel oubli de la part de l'utilisateur et limite les possibilités de piratage dans ce champ. La fonction intégrée preg_match() fait une comparaison entre le regex en premier paramètre et l'élément à comparer.

Afin de limiter l'entrée de données dans l'url j'ai mis en place un Htacces. Ce fichier permet de définir les éléments qui peuvent être envoyés dans l'url. En définissant le nombre d'entrées associé à une clef, on limite ainsi les accès au site.

```
RewriteEngine ON
RewriteRule ^recipes$                                   index.php?loc=recipes [L]
RewriteRule ^(articles|statistics)$                       index.php?loc=$1 [L]
RewriteRule ^([a-z]+)$                                    index.php?loc=$1 [L]
#RewriteRule ^recipes/editing$                            index.php?loc=recipes&action=editing [L]

RewriteRule ^([a-z]+)/([a-z]+)$                                                 index.php?loc=$1&action=$2 [L]
RewriteRule ^([a-z]+)/([a-z]+)/([a-z\-_0-9]+)$                                  index.php?loc=$1&action=$2&id=$3 [L]
RewriteRule ^([a-z]+)/([a-z]+)/([a-z\-_0-9]+)/([a-z\-_0-9]+)$                   index.php?loc=$1&action=$2&id=$3&supp=$4  [L]
RewriteRule ^([a-z]+)/([a-z]+)/([a-z\-_0-9]+)/([a-z\-_0-9]+)/([a-z\-_0-9]+)$    index.php?loc=$1&action=$2&id=$3&supp=$4&state=$5 [L]
```

Par exemple : a-z/-_0-9 n'autorise que les lettres minuscules et les chiffres. De cette façon le site est protégé contre l'injection de script ou de commande directement envoyés par URL. Grâce à cette méthode plutôt simple à mettre en place, j'empêche des personnes malveillantes de naviguer à leur guise et de tester les éléments de sécurité par l'url.

Afin de limiter les accès aux différents utilisateurs j'ai mis en place un contrôle des onglets en fonction du statut des usagers. De ce fait, l'administrateur sera le seul à pouvoir accéder à tous les onglets.

Pour finir cette partie, il est à noter que dans la gestion du profil on utilise la méthode « password_verify » cette fonction pré-intégrée permet de vérifier par un booléen si le mot de passe encrypté et importé de la base de données correspond à celui écrit par l'utilisateur.

```php
public function isPassword($plaintextPassword)
{
    return password_verify($plaintextPassword, $this->getPasswordHash());
}
```

L'avantage de cette méthode est que le mot de passe ne soit pas directement vérifié par une requête. En fait, le mot de passe est récupéré avant dans un objet (avec l'ensemble des données utilisateur). Cette étape de récupération se déroule sans interaction avec des entrées venant de l'extérieur car elles pourraient corrompre la requête. Donc, en dehors de la gestion du cryptage, on compare ce qui est stoké avec ce que l'utilisateur écrit. La demande de confirmation ne se fait pas sur une requête au serveur.

3. NESTI CLIENT

A. Aspect graphique et architecture

Pour réaliser cette maquette j'ai utilisé draw.io qui est un outil qui permet de réaliser des formes colorées rapidement et d'afficher du texte. Son principal inconvénient, comme tout logiciel en ligne, est que si le navigateur plante, l'intégralité des modifications jusqu'à la dernière sauvegarde sera perdue.

Pour la photo, qui est l'élément central du décor, j'ai souhaité transmettre une atmosphère chaleureuse et conviviale en utilisant des éléments naturels. Le bois m'a semblé être le matériau le mieux adapté pour le fonds car il représente l'idée de tradition. Dans un cas de modernité j'aurai davantage opté pour un fond métallique type cuisine industrielle. Le napperon à carreaux fait le lien avec le pique-nique et donc en lien avec la nature.

Alors, pour ne pas créer un écart trop important entre cette photo et les autres éléments, j'ai choisi de rester dans le même thème.

C'est pour cette raison que les teintes en dehors de la photo sont chaudes avec une dominante marron. De plus, en jouant sur la saturation des couleurs et en la maintenant assez faible, je pense avoir réussi

à créer une atmosphère rétro et apaisante. La combinaison des deux permet d'obtenir ce type d'ambiance.

Pour l'architecture, il fallait que le site soit consultable même sans avoir créé de compte et qu'une fois connectés les utilisateurs puissent acheter des articles dans un marché, écrire des commentaires sur les recettes et payer éventuellement.

B. CodeIngiter

CodeIngiter est un framework conseillé à ceux qui débutent avec ce genre d'outils. Lors d'une veille j'ai lu que les autres existants étaient considérés comme plus complexes à maitriser. Je me suis donc penché sur celui-là et j'ai trouvé lors d'une recherche sur son fonctionnement, cette représentation qui me semble bien plus parlante qu'une longue explication.

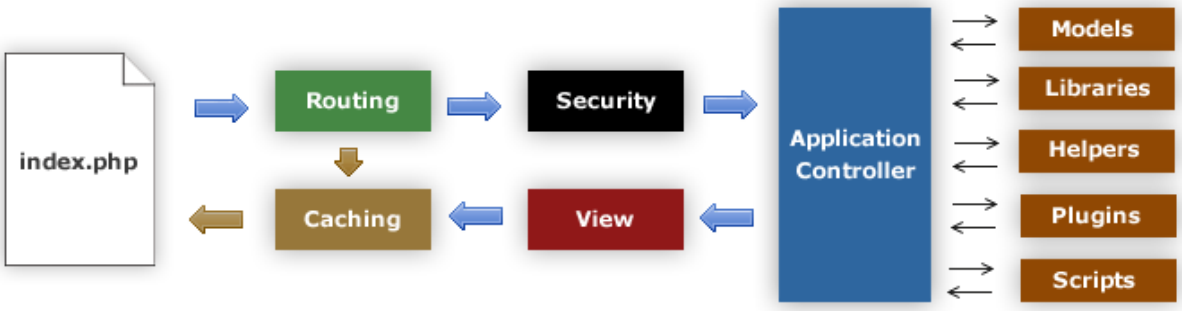


Figure 15 Résume le fonctionnement de CodeIngiter

Ordre	Nom	Description
1	index.php	Ce sera toujours le fichier index.php, situé à la racine du répertoire, qui sera appelé en premier.
2	Routing	C'est le routeur. C'est lui qui récupérera l'URL et la décomposera en actions. Il a pour rôle de trouver le contrôleur à appeler. Nous pouvons modifier son comportement.
3	Caching	Une fois que le routeur a fini son travail, le module de cache va regarder s'il existe des fichiers mis en cache pour cette action. Dans le cas d'une réponse positive, ces fichiers vont être renvoyés au navigateur.
4	Security	Cette partie va sécuriser toutes les données entrantes : cookies, variables get, post, etc. C'est la dernière étape avant le contrôleur.
5	Application Controller	Ce sera ici que nous commencerons à développer.
6	Models Libraries Helpers	Comme vous le voyez, le contrôleur fait appel à différents éléments qui vont leur renvoyer des données.
7	View	Les vues sont les fichiers qui contiennent le code HTML.

CodeIgniter pour accéder aux données à besoin d'être configuré, pour cela il faut modifier (Url, RewriteEngine, puis définir la base de données) et les fichiers ('public'.htacces), (.env) et (App.php) afin que la connexion puisse se faire en local (ou distant).

CodeIgniter facilite le travail de codage car il possède une librairie intégrée de requêtes Sql ('where', 'find', ect…) et de fonctions.

Du point de vue de la sécurité CodeIgniter utilise des 'routes' on les trouve dans le fichier Routes.php situé dans le dossier app.
Celles-ci lient une url avec une fonction dans un fichier 'Controller'. On doit définir également la nature des éléments envoyés dans l'Url.

```
$routes->get('/recipes/(:num)', 'TagController::recipes/$1');
$routes->add('/detailsRecipe/(:num)', 'RecipeController::detailsRecipe/$1');
```

Figure 4 Exemple de 'routes'

Dans ce cas, si l'url est de la forme '/recipe/tarte' , CodeIgniter renverra une erreur de sécurité et redirigera vers un 'page not found'. Ce fonctionnement est assez similaire au htacess car dans les parenthèses on peut définir l'élément entré. C'est gage de sécurité puisque dans notre cas la 'route' n'accepte que des chiffres.

Un des gros avantages du framework est que je n'ai pas besoin de définir tous les éléments et en particulier ceux des classes. En fait, CodeIngiter a seulement besoin que les modèles soient correctement établis.
Pour cela il faut créer un fichier dans le dossier « Models » l'associer à la classe intégrée qui gère les modèles.
Ensuite, il faut définir une table existante dans la base de données, définir chaque colonne active comme dans base de données et enfin lui donner la position de la classe dans l'arborescence de l'application. Il faut également lui indiquer la clef primaire si l'id est nommée différemment de 'id'.
Grâce à ce fonctionnement, CodeIngiter met en place dans les classes tout juste créées les attributs, et des méthodes de façon virtuelle. Les classes doivent être juste héritées des propriétés de « Entity ». « Entity » et comme « Models », ce sont des classes parents intégrées qui gèrent le fonctionnement des enfants. Les contrôleurs sont les chefs d'orchestre, ils regroupent tous les éléments de code. C'est en leur sein qu'on gère les différentes actions et qu'on met en place les sécurités supplémentaires.

C. Twig

Le principal obstacle que j'ai rencontré est qu'il y a peu d'informations et de tutos le concernant lors d'une association avec CodeIgniter 4. Dans la majorité des cas, il est associé à Symfony.

Cependant, son utilisation facilite de manière intuitive le code et rend la lecture du Html plus « esthétique » que celle du php brut. Afin de le rendre actif dans le code il faut d'abord l'installer avec composer, renommer les fichiers l'utilisant avec le terme « .twig ». J'ai donc remodelé le code initial qui était en html afin d'activer les fonctionnalités twig. En fait Twig « réserve » l'emplacement d'un block par une syntaxe particulière : **{% block body %}{% endblock %}** puis dans un fichier qui possède une association au fichier d'origine par le code **{% extends "base.html.twig" %},** je peux alors créer les éléments html. Cela permet de décomposer rapidement le code en plusieurs fichiers afin de gérer indépendamment les éléments d'une page.

```
    </ul>

{% endif %}
{% if user.login == NULL %}
    <ul class="navbar-nav mr-auto">
        <li class="nav-item ">
            <a class="nav-link" href="{{base_url('/login')}}">Login</a>
        </li>
        <li class="nav-item ">
            <a class="nav-link" href="{{base_url('/register')}}">Register</a>
        </li>
    </ul>
{% endif %}
{% if user.login != NULL %}
<ul class="navbar-nav my-2 my-lg-0">
        <li class="nav-item">
            <a  class="btn btn-info" href="{{base_url('/buy')}}">Panier</a>
        </li>
    </ul>

    <ul class="navbar-nav my-2 my-lg-0">
        <li class="nav-item">
            <a class="nav-link" href="{{base_url('/logout')}}">Logout</a>
        </li>
    </ul>

    <ul class="navbar-nav mr-14">
        <li class="nav-item ">
            Utilisateur :
            <br>
            {{user.lastName}}
            {{user.firstName}}
            {{user.name}}
        </li>
    </ul>
```

Figure 16 Vue de header.html.twig

Pour envoyer les informations, il est également plus simple car son langage en balises simplifiées et rend plus claire le code HTML. Dans l'exemple ci-dessus on voit très clairement les conditions 'if' qui entourent les balises <ul> et les objets récupérés dans le dernier <li>.

```
helper("form");
$user = UserController::getLoggedInUser();
$recipesModel = new RecipesModel();
$recipes = $recipesModel->where('idRecipe', $id)
    ->findAll();
foreach ($recipes as $rec) {
    if ($rec->idImage == Null) {
        $rec->idImage = "404";
    }
};
$paragraph = new ParagraphModel();
$Prep = $paragraph->where('idRecipe', $id)->findAll();
$comp = new IngredientrecipeModel();
$compose = $comp->where('idRecipe', $id)->find();

$comm = new CommentModel();
$comment = $comm->where('idRecipe', $id)->findAll();
$grad = new GradesModel();
$data2 = [
    'idRecipe'    => $id,
    'idUsers' => $user->idUsers
];
$hisRating = $grad->where($data2)->findAll();


$this->twig->display('templates/detailRecipe.html', ['user' => $user, 'recipes' => $recipes, 'prep' => $Prep,
```

On peut dans les contrôleurs faire appel à ses fonctionnalités et associer une 'clef' à une variable Php pour utiliser son contenu dans une page utilisant le template twig (bien évidement là où on l'envoie).

D. JavaScript

Afin de réaliser certain élément, il a été nécessaire de développer des scripts afin de rendre dynamique le site. Dans le cas du panier, j'ai créé une id sur le bouton dans un écouteur d'événements sur cette id et récupère les données que je convertis en format JSON. Après cela j'envoie le JSON dans le stockage local.

```
document.addEventListener("DOMContentLoaded", function() {
    var total = 0;
var listElement = document.querySelectorAll('#but');
    listElement.forEach(element => {
        element.addEventListener('click', add);
    });
class Products {
    constructor(type, price, number) {
        this.type = type;
        this.price = price;
        this.number = number;


    }
}

function add() {
    var text = event.target.dataset.name;
    var price = event.target.dataset.price;
    var number = event.target.dataset.quantity;
    var nb = 1
    if (window.localStorage.getItem("Nesti_"+text) != null) {
        var j = JSON.parse(window.localStorage.getItem("Nesti_"+text))
        nb = (j.number) + 1
    }
    var product = new Products(text, price, nb);
    window.localStorage.setItem("Nesti_"+text, JSON.stringify(product))
    alert("le produt " + text + " a été ajoué au panier")
```

Pour récupérer les données dans le panier, j'ai créé une fonction « start » qui recompose le JSON stocké en plusieurs éléments. J'ai ensuite recréé le html pour les boutons et intégré les valeurs.

Les boutons sont gérés dans les autres fonctions à chaque événement sur eux. Il modifie le JSON et la valeur visible dans le html. Le total est alors à recalculer ou initialiser.

J'ai aussi utilisé une création réalisée en binôme afin de créer l'onglet suggestion. N'étant pas l'auteur de la majorité du code j'ai réalisé l'intégralité de la documentation explicative (visible dans l'annexe JAVASCRIPT) par contre j'ai tenu à réaliser la création de la modal pour en comprendre le fonctionnement et découvrir un minimum ce langage (au mois de décembre il nous avait été demandé de préparer en duo cet élément du projet pour s'initier au langage JS)

E. Sécurité et information

L'application que j'ai développé pour la partie CLIENT de Nesti utilise un jeton csrf (ou en français falsifications de requêtes intersites), ce fonctionnement est un gage de sécurité car il associe une valeur différente à chaque nouvelle session. Ainsi une application malveillante ne peut pas y accéder. J'ai également mis en place une sécurité de telle sorte qu'une personne bloquée par un Admin ou un des modérateurs ne puisse pas accéder au site. Comme dans PHP j'utilise la fonction intégrée « password_verify » afin de ne pas utiliser le login et le mot dans passe dans une requête. Dans le cas où l'utilisateur retourne une saisie correcte qui correspond bien à un client existant

```
public function login()
{

    helper("form");

    if (!empty($_POST)) {
        $error = 0;
        $login     = $this->request->getPost('login',FILTER_SANITIZE_STRING);
        $password     = $this->request->getPost('password',FILTER_SANITIZE_STRING);

        // Chek if login is type min Xxxx or max X+x*30
        if (!preg_match("/^[A-Z]{1}[a-z]{3,20}/", $login)) {
            $data['loginError'] = true;
            $error = 1;
        }


        //if chek is ok
        if ($error == 0) {

            $model = new UsersModel();
            //Controle if User exist and if not block
            $array = array('login' => $login, 'flag' => "a");
            $userIsNotBlocked = $model->where($array)->first();

            if ($userIsNotBlocked != array()) {

                //if exist and not block check password
                $isPassword = password_verify($password, $userIsNotBlocked->passwordHash);

                if ($isPassword) {

                    self::setLoggedInUser($userIsNotBlocked, $password);
                    $this->twig->display('templates/intro.html', ['user' => $userIsNotBlocked]);
                } else {
                    $this->twig->display('templates/login.html');
                };
            } else {
                $data['loginError'] = true;
                $this->twig->display('templates/login.html', ['loginError' => $data]);
            }
```

dans la base de données, la connexion se fait et reste maintenue jusqu'à une déconnexion ou l'expiration du jeton.

J'ai découvert, à cette occasion en parcourant les fichiers de code que par défaut un jeton est censé durer deux heures.

```php
public function register()
{
    $data["slug"] = "user";

    $encrypter = \Config\Services::encrypter();
    helper(['form']);

    $model = new UsersModel();
    $data = [];
    if (isset($_POST['profileRegister'])) {

        $rules = [
            "login" => [
                "rules" => 'required|regex_match[/^[A-Z]{1}[a-z]{3,20}/]|is_unique[users.login]',
                "errors" => [
                    "required" => "Un login est nécessaire",
                    "regex_match" => "Votre login doit commencer par une majuscule",
                    "is_unique" => "Le login n'est pas disponible."

                ]
            ],
            "email" => [
                "rules" => 'required|valid_email|is_unique[users.email]',
                "errors" => [
                    "required" => "Un email est nécessaire",
                    "is_unique" => "Email indisponible",
                    "valid_email" => "Le format de l'email est incorrect"
                ]
            ],
            "password" => [
                "rules" => 'required|regex_match[/^(.{0,30}|[^a-z]*|[^\d]*)$/]|min_length[8]|max_length[30]',
                "errors" => [
                    "required" => "Il vous faut un mot de passe",
                    "regex_match" => "Votre mot de pass est trop faible...",
                    "min_length" => "Votre mot de passe est trop court",
                    "max_length" => "Votre mot de passe est trop long"

                ]
            ],
```

Afin de vérifier les entrées des utilisateurs j'ai également utilisé un fonctionnement qui est propre à CodeIgniter. Il est bien plus simple à gérer que pour le langage php. Il suffit de créer des règles puis de définir les textes d'erreurs de chaque condition. Ce qui est pratique c'est qu'il soit également possible d'utiliser des regex et cela sans avoir besoin de procéder de façon complexe. On peut alors aisément et rapidement définir de nombreuses règles pour chaque entrée utilisateur. J'ai également utilisé des filtres afin d'éviter l'injection de caractères de langage

4. NESTI MOBILE

A. Structure

Dans le cadre de la partie mobile du projet, j'ai utilisé Android studio avec un langage de programmation java. Celui-ci permet de fabriquer une interface graphique à l'aide d'une palette d'outils. Ces outils interprètent un langage Xml en rendu graphique ce qui permet de créer l'interface élément par élément. Le fonctionnement est assez similaire à Swing, on dépose un élément sur une vue. Par contre sur Android, il est possible d'avoir accès à de plus nombreuses fonctionnalités.

Le programme fonctionne grâce à plusieurs éléments. Les entités ou classes qui sont en fait les éléments traditionnels pour les objets. Elles contiennent les attributs et les méthodes. Les Xml qui font partie de la structure graphique et qui ont pour fonction entre autres de lier les éléments de texte.

Dans un même registre, les adapteurs définissent le contenu de la ressource quand une activité fait appel à un Xml extérieur pour l'inclure ailleurs.

Les modèles ont pour fonction de récupérer le JSON envoyé par l'API et d'envoyer aux classes les données récupérées.

Et pour terminer, les activités dont le but est d'assembler les contenus des éléments « adapter », « classe » et Xml. Elles permettent de créer des évènements sur les Xml et d'exploiter les « adapter ».

B. Sécurité et protection

Concernant la sécurité, j'ai mis en place une contrainte sur les entrées permises lors de la recherche dans le fichier xml.

```
android:digits="qwertzuiopasdfghjklyxcvbnm"
```

Afin d'éviter les erreurs de recherche dues à une saisie de chiffres ou de symboles, cette ligne limite la saisie aux caractères autorisés. De cette façon, cela renforce la sécurité et limite l'envoi d'informations dans la requête de recherche.

Dans l'exécution des requêtes j'emploie également des éléments « try and catch » afin de récupérer les erreurs dans le cadre d'une requête API.

C'est également à ce moment que je récupère les éléments JSON afin de les associer à l'objet « recette ». J'utilise alors la programmation objet dès la réception des données.

```
@Override
public void onResponse(JSONArray response) {

    ArrayList<Recipe> recipes = new ArrayList<>();
    try {
        for (int i = 0; i < response.length(); i++) {
            JSONObject object_JSON = response.getJSONObject(i);
            Recipe r = new Recipe();
            r.setIdRecipe(object_JSON.getInt( name: "idRecipe"));
            r.setTitle(object_JSON.getString( name: "title"));
            r.setDifficulty(object_JSON.getInt( name: "diff"));
            int z = getRessourceImage( nameImage: "star_" + r.getDifficulty());
            int x = getRessourceImage(object_JSON.getString( name: "img"));
            r.setImgId(x);
            r.setImgStar(z);
            r.setAuthor(object_JSON.getString( name: "author"));

            recipes.add(r);

        }

    } catch (Exception e) {
        Log.e( tag: "LogNesti",  msg: "Erreur de Convertion du Json");
        e.printStackTrace();
    }
    GlutenViewModel.this.recipes.setValue(recipes);
}
```

Dans le cadre de la protection j'ai également mis en place un token afin de sécuriser l'utilisation de l'API. Pour ce faire, l'application du téléphone possède un code qu'elle envoie à l'application CodeIgniter via l'API.

```
String url =
"https://delperie.needemand.com/realisations/Nesti_CodeIngniterdebug/public/index.
php/"+token+"/api/ingredient/"+idRecipe;
```

CodeIgiter va envoyer une requête au serveur afin de vérifier l'existence de la clef du token. Puis s'il y a concordance, il lancera la recherche associée dans l'API et récupèrera les données et les encodera au format JSON afin de les rendre exploitables notamment par l'application Android.

```
public function category($token,$cat)
{
  if($this->token($token)==true){
  $model =new RecipesModel();
  $recipes = $model->findCatForApi($cat);
  header('Content-Type: application/json');
  echo json_encode($recipes);
  die;
 }
}
```

Afin d'expliquer l'utilisation de l'API j'ai mis en place une documentation accessible en PDF depuis le site qui permet de tester les différents liens et explique le fonctionnement de chaque API.

5. Performance générale du projet

Une fois les différents projets bien avancés j'ai utilisé des outils afin de tester mes réalisations. En ce qui concerne l'application NESTI-PHP, j'ai voulu faire un test d'injection avec le logiciel Burp. Il utilise un certificat de navigateur qu'il faut mettre en place afin de réaliser des tests d'intrusion. Une fois terminé, il faut récupérer une liste de code d'injection en faisant une recherche avec le terme 1N3 intruderPayloads que l'on utilise afin de tester des champs. J'ai testé sur l'accès du site de needemand.

```
POST /realisations/PHP_Nesti_Site/connection HTTP/1.1
Host: delperie.needemand.com
Cookie: PHPSESSID=gu04p15om5oa9d6m1rv8i5dtqv
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:89.0) Gecko/20100101 Firefox/89.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en,en-US;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 46
Origin: https://delperie.needemand.com
Dnt: 1
Referer: https://delperie.needemand.com/realisations/PHP_Nesti_Site/connection
Upgrade-Insecure-Requests: 1
Te: trailers
Connection: close

loginUser=$Jean$&password=123&connection=Valider
```

3. Intruder attack of delperie.needemand.com - Temporary attack - Not saved to project file

Attack Save Columns

Results Target Positions Payloads Resource Pool Options

Filter: Showing all items

Request	Payload	Status	Error	Timeout	Lengt
0		302			4905
1	==	302			4878
2	=	302			4878
3	'	302			4878
4	' --	302			4878
5	' #	302			4878
6	' □	302			4878
7	'--	302			4878
8	'/*	302			4878
9	'#	302			4878
10	" --	302			4878
11	" #	302			4878
12	"/*	302			4878

On remarque bien que initalement la connection se fait car la taille de la réponse varie. Puis pour les autres éléments on a simplement une redirection (302). Ce qui est cohérent car dans tous les cas il a redirection vers une autre page mais les autres fois, la validation ou l'extraction de données ne s'est pas faite.

J'ai également entrepris de corriger les erreurs avec phpStan pour le PHP et CodeIngiter. Cet outil s'installe avec composer dans le terminal de VS code. Il fait une analyse statistique du code selon différents niveaux d'exigence. C'est un outil moderne qui va analyser une application PHP sans tenir compte des spécificités des différents frameworks et librairies.

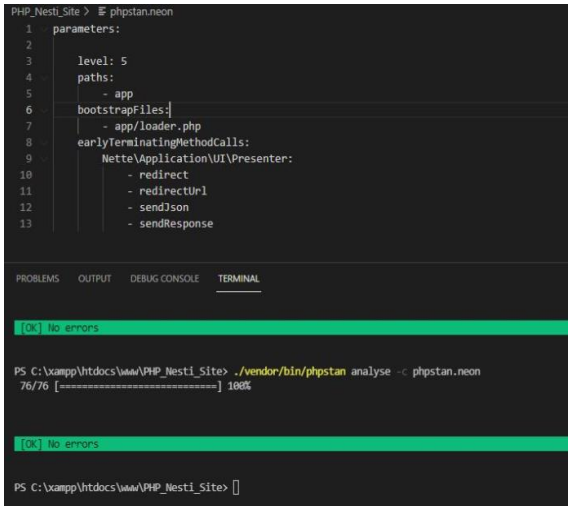


Figure 17 Correction niveau 5 PHP

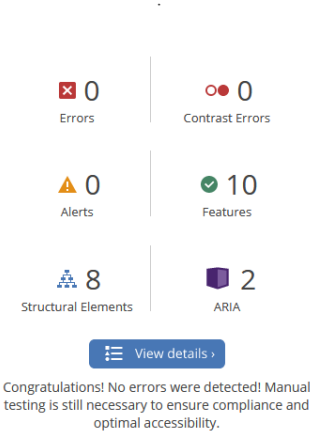
J'ai également procédé à des vérifications concernant la partie client car il fallait respecter l'accessibilité, la stabilité et les performances globales notamment écologiques. Ainsi j'ai fait des tests sur mes réalisations et corrigé des erreurs. « ANNEXE PLAN DE TEST CODEINGITER »

Pour PHP Stan je suis passé de cent soixante et une erreurs à une quarantaine seulement avec un niveau 5. La majorité des erreurs étant liées à une incompréhension de Twig. Je n'ai pas trouvé comment paramétrer correctement le phpstan pour twig.

Pour le contraste des couleurs, j'ai utilisé color Contrast Accessibility qui compare les éléments de couleurs des textes et des contenus, après un léger ajustement j'ai obtenu une validation.

Puis j'ai procédé à un test de performance sur le chargement d'une page. Le framework étant relativement plus lourd qu'une application « from scratch » je ne suis pas étonné que la performance ne soit pas idéale surtout que certaines erreurs non corrigées sur phpStan peuvent être un facteur de ralentissement site.

Afin de vérifier l'accessibilité du site, j'ai fait une vérification sur Wave. Il s'agit d'un vérificateur pour malvoyants. Il vérifie la structure du code et les éléments qui permettent de rendre accessible le site aux déficients visuels.



J'ai également réalisé un test sur GtMetrix. Ce logiciel contrôle les performances d'affichage et de chargement de la page. On peut voir les différentes parties chargées dans une chronologie ainsi que les temps mis pour afficher les différents éléments.

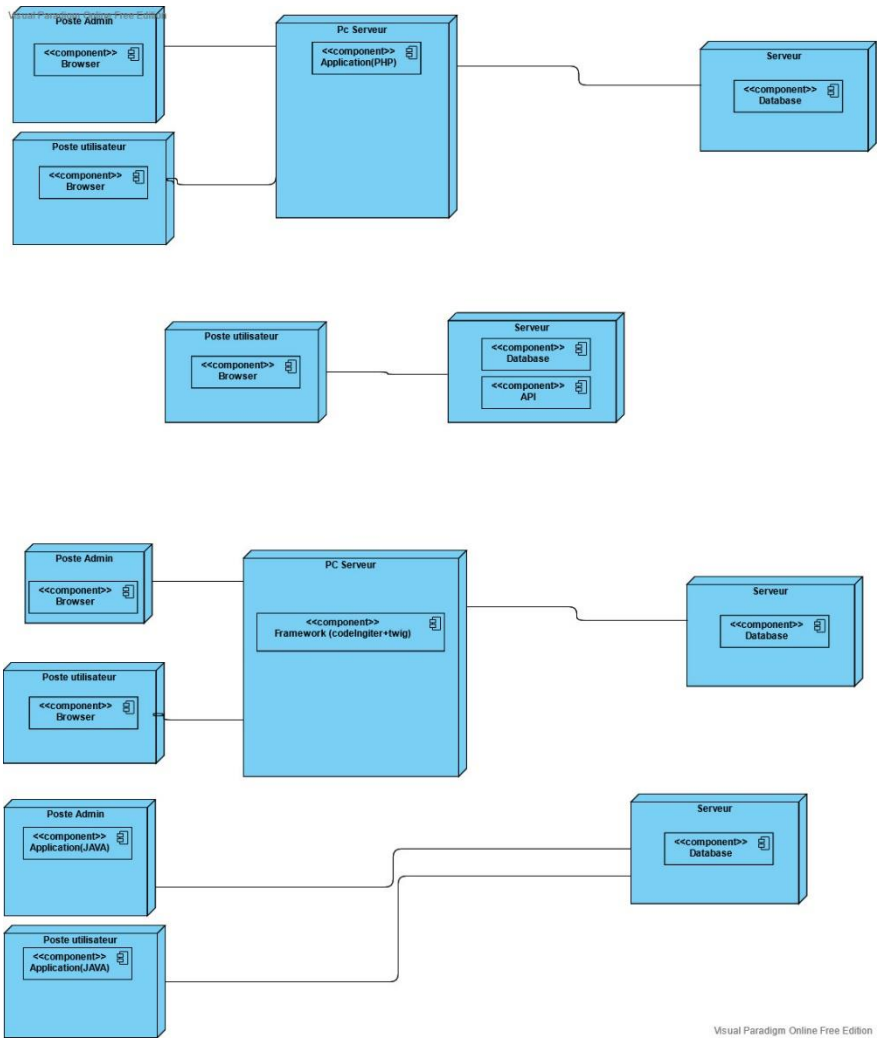
J'avais aussi fait un test sur web Accessibility mais je trouvais que les messages d'erreurs n'étaient pas suffisamment explicites.

Pour finir, j'ai un test d'économie d'énergie avec Greenit, le résultat pour le site CLIENT est plutôt faible mais on pouvait s'y attendre. J'ai une note de C pour le CLIENT alors que la partie Admin jongle entre une note A et une note B.

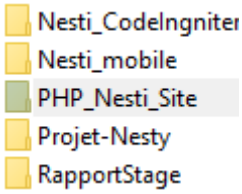
J'en conclus qu'un site codé à la main est pour notre comparaison, plus lourd en fonctionnalités et consomme beaucoup moins de ressources qu'un framework. Un client soucieux de l'environnement mais peu pressé pourrait donc exiger de coder un site « from scratch » dans un avenir proche même si j'en doute

6. Déploiement du projet

Dans le but de préparer le déploiement, j'ai préparé un diagramme qui explique la mise en place des différents projets. Celui-ci lie les composants matériels aux éléments logiciels que j'ai développé.



Pour le déploiement, j'ai téléversé les applications sur le serveur de Needemand grâce à Filezilla.



Pour configurer Filezilla il faut avoir un compte utilisateur sur le serveur de Needemand et entrer ses données utilisateurs. Normalement, les données sont conservées jusqu'à une nouvelle mise à jour. Il faut donc ne pas perde ses identifiants et mot de passe de session sinon on ne peut plus interagir avec ses dossiers.

Une fois connecté, il faut fournir les liens url de chaque racine de dossier aux utilisateurs extérieurs.

Dans le cadre de l'application mobile, il faut créer un fichier apk et faire une certification. La certification permet au développeur d'authentifier l'intégrité de son code. C'est un peu comme un brevet mais ici on ne s'intéresse pas au contenu du code à proprement parler mais seulement à l'enchainement des caractères.

Une fois l'authentification obtenue, le développeur donne (en payant) son certificat a un organisme qui est habilité. Du côté utilisateur, dès lors qu'il installe l'application, il reçoit un certificat qui authentifie que son code est identique à celui du développeur et que l'application n'est pas modifiée.

Une partie du projet n'a pu être aboutis dans les temps.

Projets	Réalisé	Non réalisé	En cours	Commentaires
NESTI STOCK	x			
NESTI ADMINISTRATION			x	Statistique graphique commande et graphqie cout/vente + tableau rupture de Stock
			x	L'encadrer sur les préparations
			x	Les barres de recherche
NESTI CLIENT			x	Reste la réccuperation de la validation du panier pour crée une commande
			x	Section historique
			x	Barre de recherche
Application Mobile	x			

VIII. Présentation du jeu d'essai

Le jeu d'essai regroupe une partie des demandes du cahier des charges. Je l'ai réalisé sur Nesti-ADMIN et j'ai découpé les tests selon les différents onglets.

Test à réaliser		
Nesti administration	Résultat attendu	Etat du fonctionnement
Colonne1	**Colonne2**	**Colonne3**
Connexion		
Connexion avec un utilisateur sans droit (client)	Ne pas accéder au site	Ne pas accéder au site
Connexion avec un utilisateur chef	Connexion et accès a recettes uniquement	Connexion et accès à recettes
Connexion avec un utilisateur moderateur	Connexion et accès a utilisateurs uniquement	Connexion et accès à utilisateurs
Connexion avec un utilisateur Administrateur	Connexion et accès tous le site	Connexion et accès tous le site
Erreur sur l'entrée sur le site	message erreur	message erreur
Colonne1	**Colonne2**	**Colonne3**
onglet utilisateur		
Creation d'un utilisateur avec login déjà existant	Message d'erreur	Message d'erreur
Creation d'un utilisateur avec email déjà existant	Message d'erreur	Message d'erreur
Modification d'un utilisateur avec changement de role(supp)	Message d'erreur	Message d'erreur
Modification d'un utilisateur avec changement de role(ajout)	Message de validation	Message de validation
Modification d'un utilisateur avec mauvais email	Message d'erreur	Message d'erreur
Modification d'un utilisateur avec faux codepostal	Message d'erreur	Message d'erreur
Modification Consultation de ses commandes en cliquant de	Détails visible	Détails visible
Modification de ses commentaire (Bloquer)	Changement état	Changement état
Modification des commentaires si l'utilisateurs est bloquer	Tous ses commentaires sont bloquer	Tous ses commentaires sont bloquer
Colonne1	**Colonne2**	**Colonne3**
Onglet recette		
Suppression d'une recette	Passe en bloquer visible par la pastille rouge sur ID	Passe en bloquer visible par la pastille rouge sur ID
Modification d'une recette	Message de validation	Message de validation
Ajout d'un ingredient avec une mauvaise saisie nom	Message d'erreur	Message d'erreur
Ajout d'un ingredient avec une mauvaise saisie quantité	Message d'erreur	Message d'erreur
Ajout d'un ingredient avec une mauvaise saisie unité	Message d'erreur	Message d'erreur
Ajout ou suppression d'une preparation		Non réalisé
Colonne1	**Colonne2**	**Colonne3**
Importation valide	Détails des importations	Détails des importations
Importation mauvais fichier	Message d'erreur	Message d'erreur
Importation mal réalisé	Message d'erreur	non réalisé
Colonne1	**Colonne2**	**Colonne3**
Onglet statistique		
Consulation du site	données statistique	Fonctionnel
Commande	données statistique	non réalisé
Stat de vente	données statistique	non réalisé
Rupture de stock	données statistique	Incomplet

IX. Description de la veille

Sécurité Android et piratage

Depuis quelques années, des attaques se développent de plus en plus vite sur les systèmes Android. Les malwares Teabots et Flubots en sont souvent la cause. Le fonctionnement des malwares consiste à simuler un formulaire bancaire afin de récupérer les informations de compte. Parfois, ces attaques sont le fruit de la vulnérabilité des composants graphiques.

Le plus souvent ces malwares sont intégrés dans une application hors PlayStore. Le hacker profite de l'engouement pour un procédé et falsifie son utilisation.
Pendant la pandémie, une explosion des achats sur internet a eu lieu créant ainsi des nouveaux utilisateurs non avertis sur ces procédés notamment avec les courses à distance ou les livraisons à domicile. Une aubaine pour les hackeurs qui se sont empressés de créer des simulacres d'email contenant des situations invraisemblables. Ce faisant ils ont utilisé l'ignorance et la panique de certains pour de les diriger vers des copies de site ou des téléchargements.

Après le téléchargement l'utilisateur confiant voit ses données personnelles compromises. Android dans sa conception utilise les applications en tâche de fond, ainsi une fois installées elles peuvent fonctionner sans que l'utilisateur ne puisse constater leurs activités. Les hackeurs réussissent à mettre en place une écoute des flux du téléphone et peuvent même remplacer certaines applications par des faux. Leur but dans ce cas est d'accéder à l'ensemble des données émises et collectées par l'utilisateur. Dans l'éventualité où l'utilisateur est personnellement ciblé, les hackers pourront alors chercher à faire correspondre au mieux les demandes à la réalité.
Il existe en réalité deux cas.
Celui de l'attaque personnelle, la conséquence va se limiter au pire à la perte d'argent qui pourra éventuellement être récupérée en fonction des banques. Comment réagir : https://www.bforbank.com/mag/budget/compte-bancaire-pirate-comment-reagir.html

Mais dans le cas d'une entreprise la facture peut être beaucoup plus lourde. Par exemple : un employé possédant un téléphone entreprise.
Il n'a pas plus de sécurité intégrée qu'un téléphone personnel mais permet d'accéder aux données de tous les clients pour les contacter. Si le hacker identifie l'entreprise par l'opérateur, qu'il arrive à connaître un seul nom de fournisseur (ou le deviner) dans le répertoire il peut très bien faire en sorte d'imiter une demande classique de fournisseur « pouvez-vous confirmer la réception du colis, etc. » et rediriger vers un faux site.
L'installation réussie il peut alors accéder aux contacts de l'entreprise et se faire passer pour un démarcheur chez un client (et se répandre), ou revendre les informations commerciales sensibles à des concurrents ou faire une demande de rançon comme on l'a constaté récemment dans l'affaire d'attaque d'hôpitaux en France ou celle du groupe Sodin/Revil sur Apple. Ces attaques ne sont pas passées forcément par une intrusion mobile mais dans l'avenir il n'est pas exclu que des procédés de ce type puissent être mis en place. On a déjà observé que les utilisateurs se détournaient de plus en plus des ordinateurs en faveur des appareils mobiles favorisant ainsi de nouvelles opportunités pour les piratages.

Que mettre en place ?
Pour éviter les problèmes de ce type, les développeurs et les entreprises ont fort à faire. Dans un premier temps, pour se protéger, les entreprises devront certainement à l'avenir utiliser une application qui gère les contacts de manière indépendante du système classique afin de les sécuriser et de les authentifier.
Le plus rapide serait de mettre leurs données sensibles hors réseaux avec une mise à jour des données locales par un utilisateur physique. Cela ralentira sûrement certains processus internes mais évitera la récupération directe à distance.
Pour les entreprises ayant davantage de moyens, crypter toutes les données, faire des sauvegardes de secours journalières, sensibiliser les employés et maintenir à jour les logiciels semble être déjà une bonne protection contre ces attaques.
Une autre possibilité rendant les demandes de rançons inutiles consiste à utiliser un environnement Cloud car selon « Norton », il est toujours possible de retrouver ses informations passées.

Dans le cas de vulnérabilité de composants, la seule chose à faire est de mettre à jour son appareil afin de palier à un problème. Le plus souvent ce genre de problème existe sur des appareils exploitant une technologie nouvelle. Quand il s'agit d'un problème répandu, le constructeur propose très rapidement un correctif pour éviter que ce type de problème reste longtemps accessible sur ses appareils.

Il est donc important de faire certifier les applications avant de les rendre accessibles au public. En effet, les systèmes Android possèdent une sécurité intégrée qu'il est nécessaire de désactiver lors de l'installation d'une application hors service officiel. C'est déjà une étape cruciale qui limite les codes malveillants « simples ». La contrainte pour le développeur est qu'il doit absolument passer par un service proposé par une grosse structure de certifications.

Source :
https://www.frandroid.com/comment-faire/tutoriaux/184151_comment-installer-un-fichier-apk-sur-son-terminal-android
https://www.phonandroid.com/android-de-fausses-applis-vlc-et-ad-blocker-propagent-de-dangereux-malware-bancaires.html

https://fr.norton.com/internetsecurity-malware-ransomware-5-dos-and-donts.html
https://www.buzzwebzine.fr/teabot-malware-banques-europe/

https://www.futura-sciences.com/tech/actualites/securite-pirates-exploitent-ancienne-faille-securite-android-86422/

X. Description d'une situation de travail ayant nécessité une recherche

Pour réaliser ces projets, j'ai dû réaliser de très nombreuses recherches, allant de la simple expression regex, ou d'élément Bootstrap, au fonctionnement entre CodeIgniter et Twig.
Afin d'expliquer ma méthodologie, je vais prendre l'exemple de la récupération d'images sur un serveur distant au sein de l'application mobile.
Pour commencer, j'ai effectué une recherche avec les mots clefs suivants : 'integration image Android studio'

Le résultat n'a pas été probant car je me suis retrouvé sur un site qui permet de faire des captures avec la caméra et de les envoyer sur serveur.
http://www.codeplayon.com/2018/11/android-image-upload-to-server-from-camera-and-gallery/

Puis j'ai transformé ma recherche : 'get server image to android studio '
https://stackoverflow.com/questions/43557167/download-a-image-from-server-to-android-and-show-in-imageview
Le premier lien envoie vers stackOverflow. Dans ce lien un développeur a un problème. Dans les aides un texte dit « You can use Glide it is simplest way to load image. This is how you can save image » puis un lien github vers Glide.
J'ai donc un nom clef qui me permet de cibler plus précisément un plugin pour une importation.
Recherche : 'Glide image import server tuto'
https://guides.codepath.com/android/Displaying-Images-with-the-Glide-Library
Ce lien donne un tuto en fournissant du code et des méthodes.

Sur la même recherche un peu plus loin dans les réponses google j'ai trouvé un autre tutoriel :

https://www.raywenderlich.com/2945946-glide-tutorial-for-android-getting-started

Ensuite, pour savoir si je pouvais améliorer j'ai essayé une recherche avec : 'image library for Android'
https://ourcodeworld.com/articles/read/929/top-10-best-android-image-loading-and-caching-libraries

Du coup je trouve également d'autres noms de librairies, le deuxième : Picasso
Recherche : 'Picasso tuto android'
https://code.tutsplus.com/fr/tutorials/code-an-image-gallery-android-app-with-picasso--cms-30966
Je me retrouve donc avec deux tutos différents de deux librairies au cas où une ne me conviendrait pas ou me créerait des problèmes.

XI Annexes

A. GESTION DE PROJET : JAVA

Charte Graphique

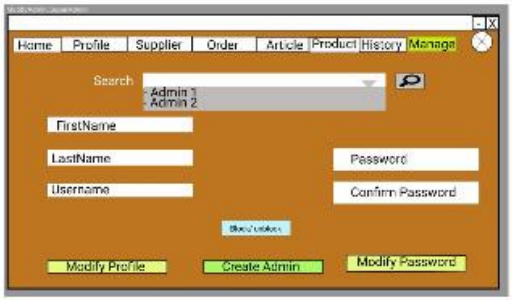
BE7520
E8FA7E

Potta One :

Almost before we knew it, we had left the ground.

c29d23
420505

F43535
ADF664



BE7520
E8FA7E
BFF4FF
ADF664

TRELLO

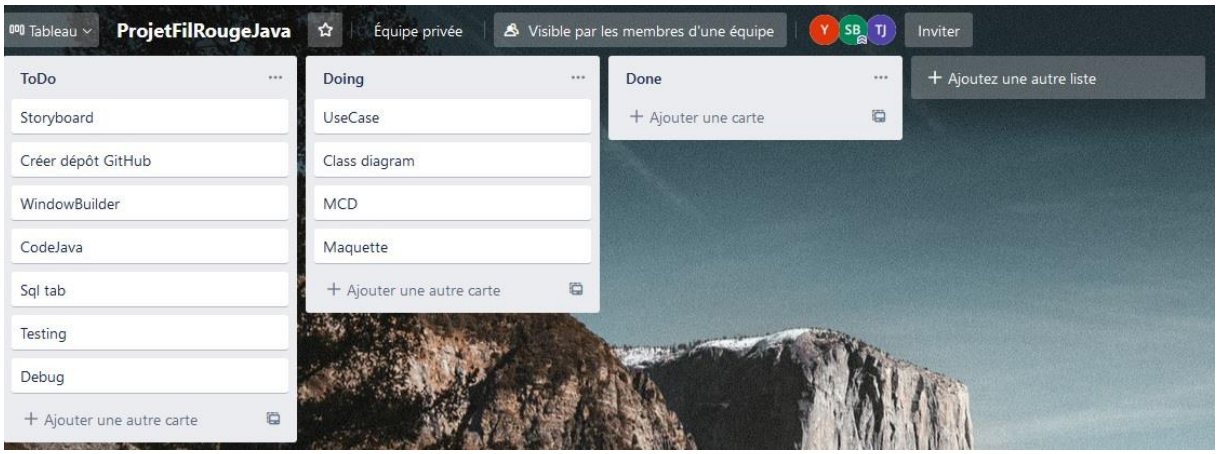
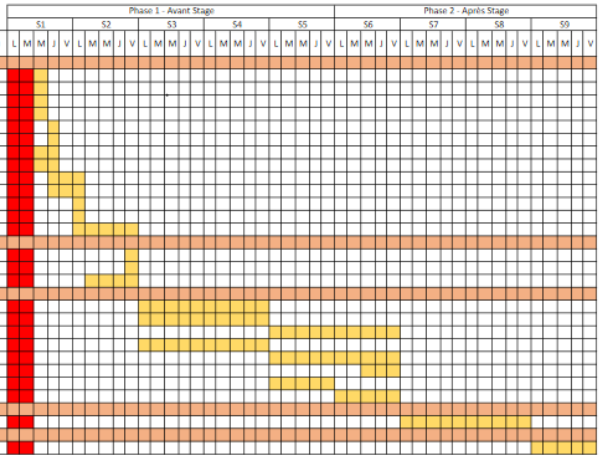


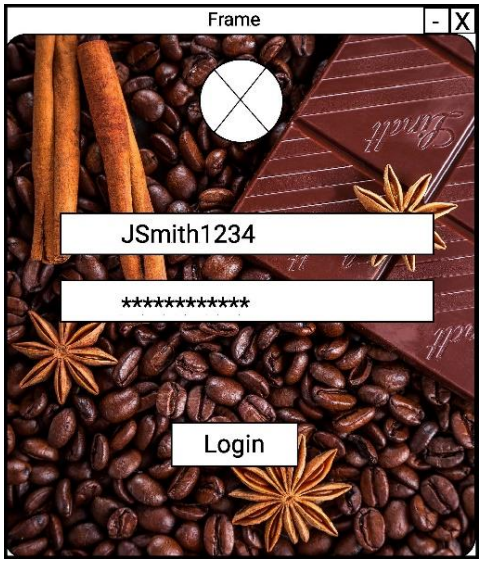
Diagramme de Grant

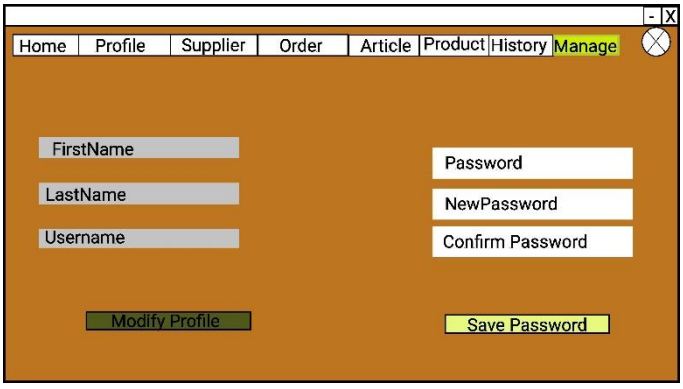
Titre Projet	Projet Fil Rouge - JAVA	Collaborateurs	Saliha Thibault Yann	Beddek Jolivet Delperie

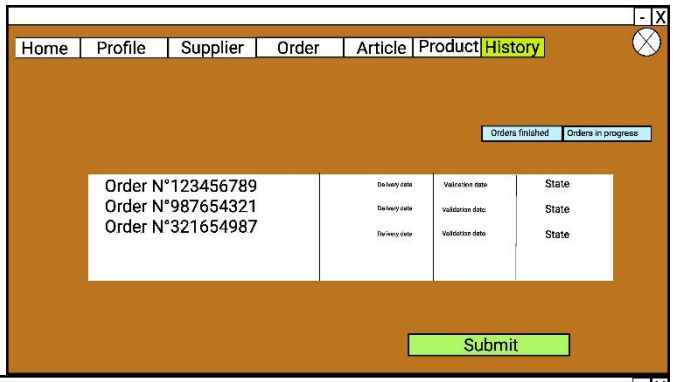
N°	Titre de la tache	Répartition des taches	Date début	Date fin	Durée (j)	Avancement(%)
1	**Analyse et Conception**					
1.1	Diagramme de gantt	Thibault	06/01/2021	06/01/2021	1	100%
1.2	Planning	Thibault	06/01/2021	06/01/2021	1	100%
1.3	Lexique des données	Saliha	06/01/2021	06/01/2021	1	100%
1.4	Matrice de dépendances fonctionnelles	Saliha	06/01/2021	06/01/2021	1	100%
1.5	Modèle Conceptuel de Données	Trio	07/01/2021	07/01/2021	1	100%
1.6	Modèle Logique de Données	Trio	07/01/2021	07/01/2021	1	100%
1.7	Use Case	Trio	06/01/2021	07/01/2021	2	100%
1.8	Diagramme de classe	Trio	06/01/2021	07/01/2021	2	100%
1.9	Maquette	Trio	07/01/2021	11/01/2021	3	100%
1.10	Storyboard	Saliha+Yann	07/01/2021	11/01/2021	3	100%
1.11	Charte Graphique	Yann	11/01/2021	11/01/2021	1	100%
1.12	Convention de nommage	Thibault	11/01/2021	11/01/2021	1	100%
1.13	Dossier de conception	Thibault	11/01/2021	15/01/2021	5	100%
2	**Mise en route du projet**					
2.1	Création base de données	Thibault	15/01/2021	15/01/2021	1	100%
2.2	Données test base de données	Thibault	15/01/2021	15/01/2021	1	100%
2.3	Création interface graphique	Saliha + Yann	12/01/2021	15/01/2021	4	100%
3	**Développement**					
3.1	Connexion application	Saliha	18/01/2021	29/01/2021	10	100%
3.2	Onglet produit	Yann	18/01/2021	29/01/2021	10	100%
3.3	Onglet article	Yann	01/02/2021	14/05/2021	10	100%
3.4	Onglet fournisseur	Thibault	18/01/2021	29/01/2021	10	100%
3.5	Onglet commande	Thibault	01/02/2021	14/05/2021	10	100%
3.6	Onglet historique	Saliha + Thibault	12/05/2021	14/05/2021	3	100%
3.7	Onglet gestion utilisateur	Saliha	01/02/2021	05/02/2021	5	100%
3.8	Onglet profil	Saliha	10/05/2021	14/05/2021	5	100%
4	**Test/contrôles**					
4.1	Test de l'application	Trio	17/05/2021	28/05/2021	10	100%
5	**Livrables**					
5.1	Guide d'utilisation	Trio	31/05/2021	04/06/2021	5	100%

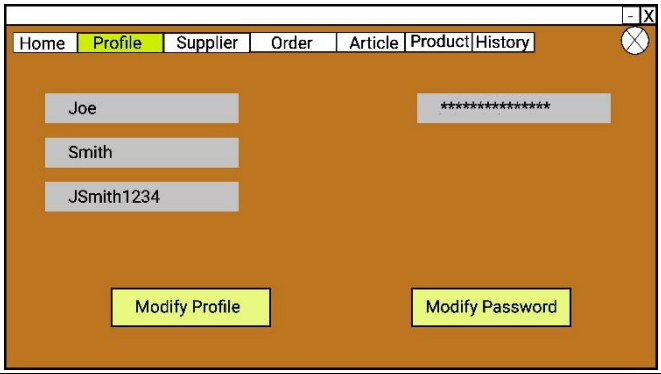


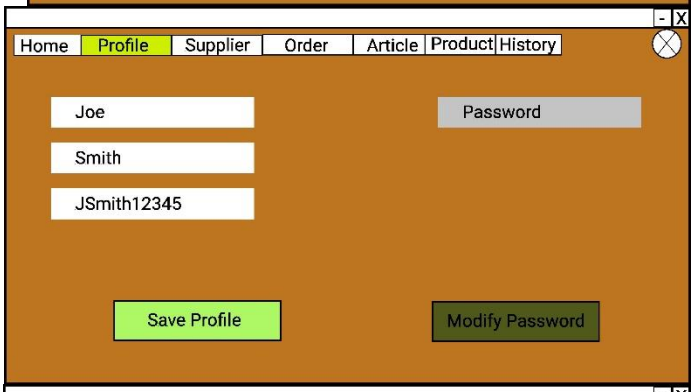
Maquette JAVA

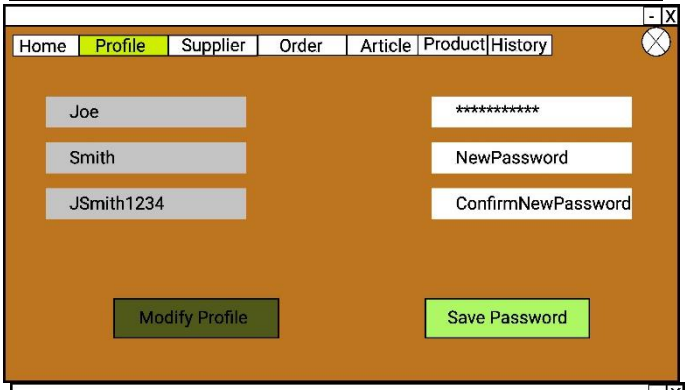


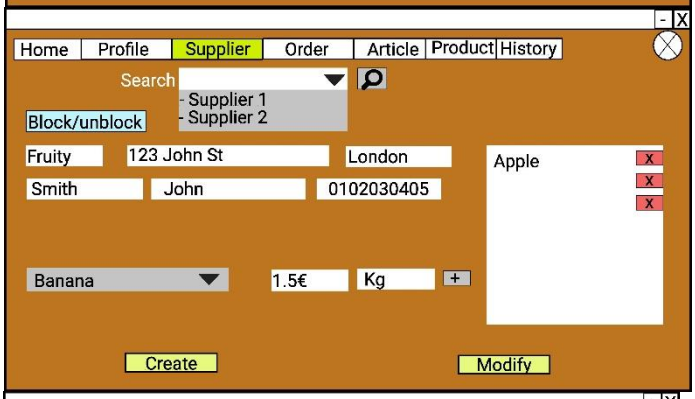


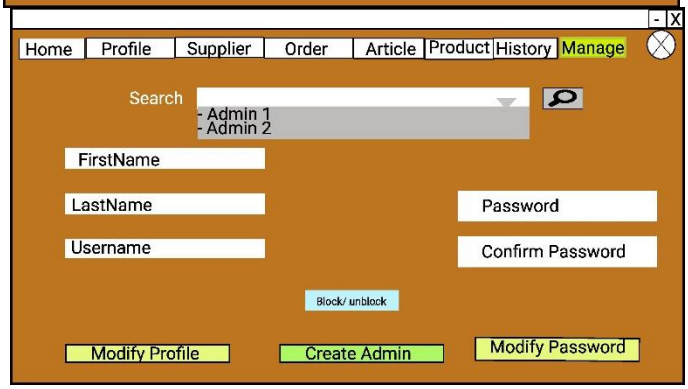

Home
Profile
Supplier
Order
Article
Product
History
Orders finished
Orders in progress
Order N°123456789
Order N°987654321
Order N°321654987
State
State
State
Submit

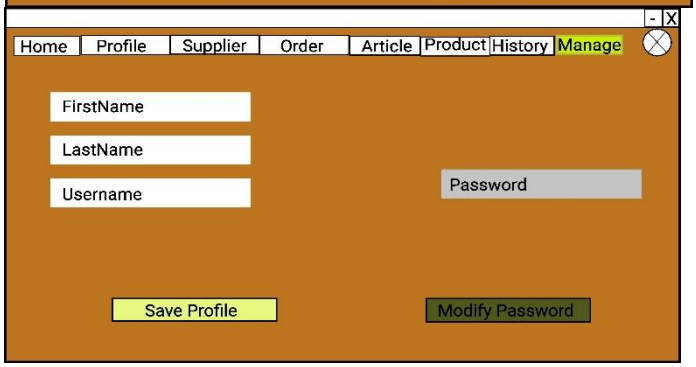

Home
Profile
Supplier
Order
Article
Product
History
Joe
Smith
JSmith1234
****
Modify Profile
Modify Password

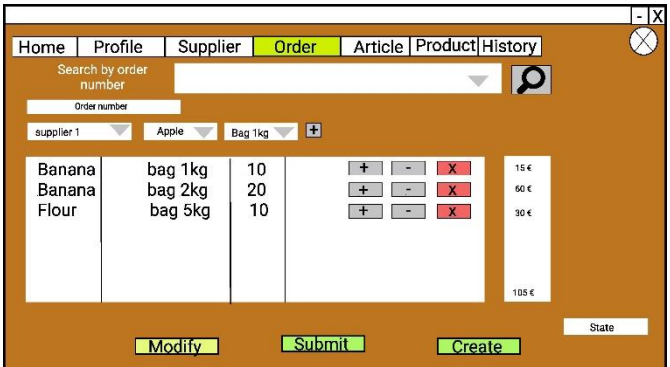

Home
Profile
Supplier
Order
Article
Product
History
Joe
Smith
JSmith12345
Password
Save Profile
Modify Password

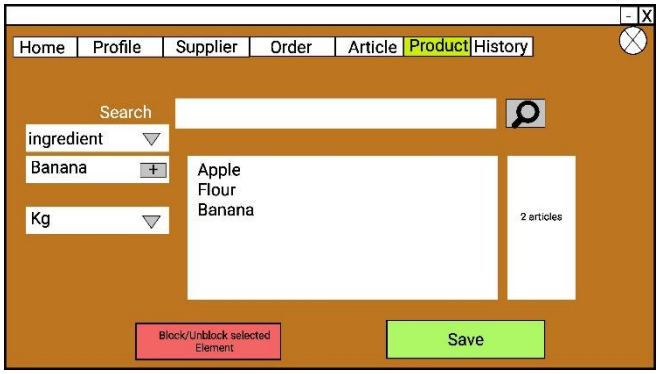

Home
Profile
Supplier
Order
Article
Product
History
Joe
Smith
JSmith1234
****
NewPassword
ConfirmNewPassword
Modify Profile
Save Password

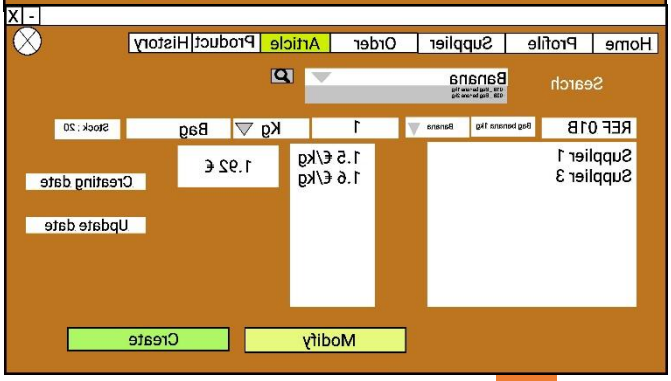

Home
Profile
Supplier
Order
Article
Product
History
Search
- Supplier 1
- Supplier 2
Block/unblock
Fruity
123 John St
London
Smith
John
0102030405
Apple
Banana
1.5€
Kg
Create
Modify


Home
Profile
Supplier
Order
Article
Product
History
Manage
Search
- Admin 1
- Admin 2
FirstName
LastName
Username
Password
Confirm Password
Block/ unblock
Modify Profile
Create Admin
Modify Password

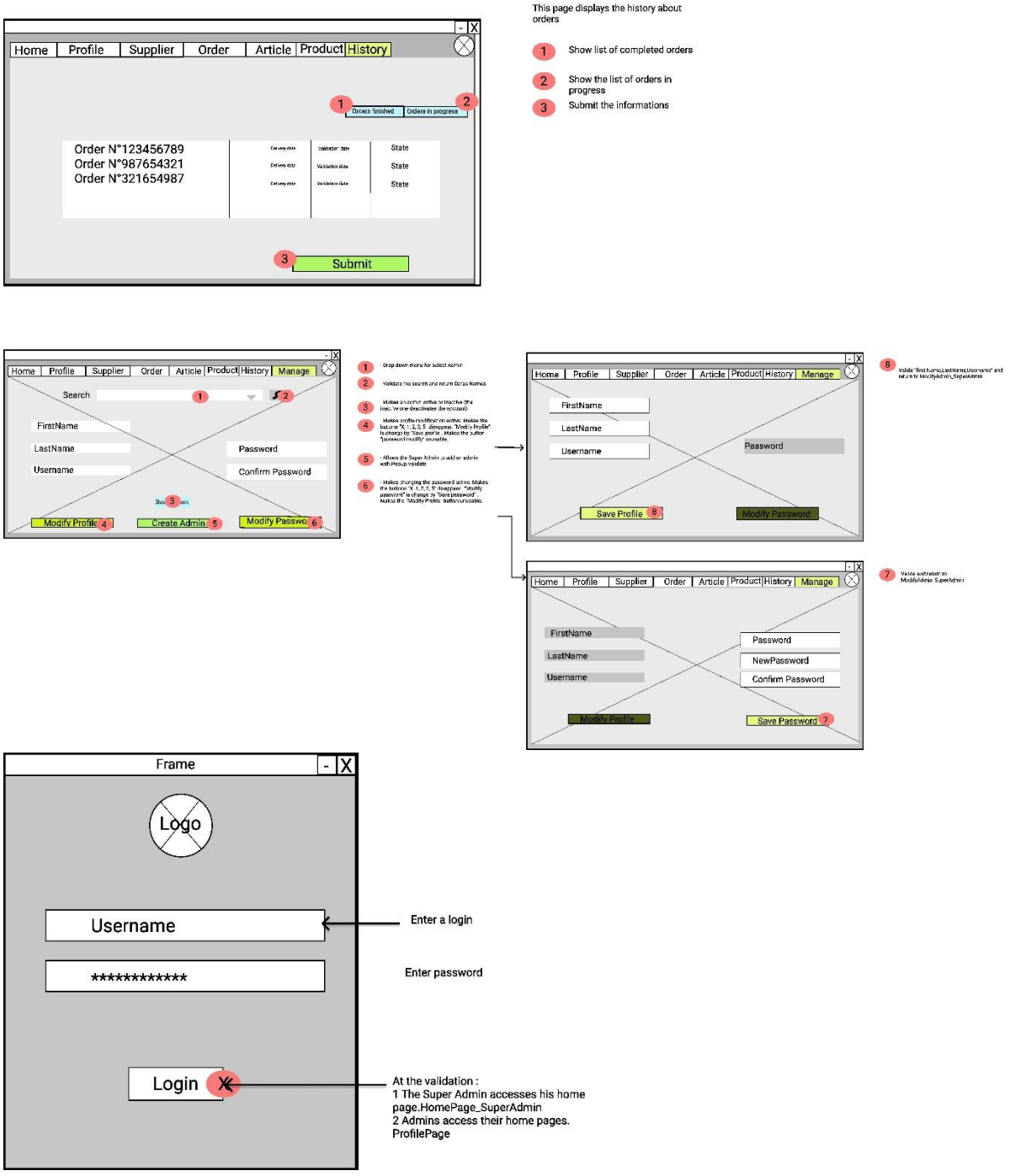

Home
Profile
Supplier
Order
Article
Product
History
Manage
FirstName
LastName
Username
Password
Save Profile
Modify Password

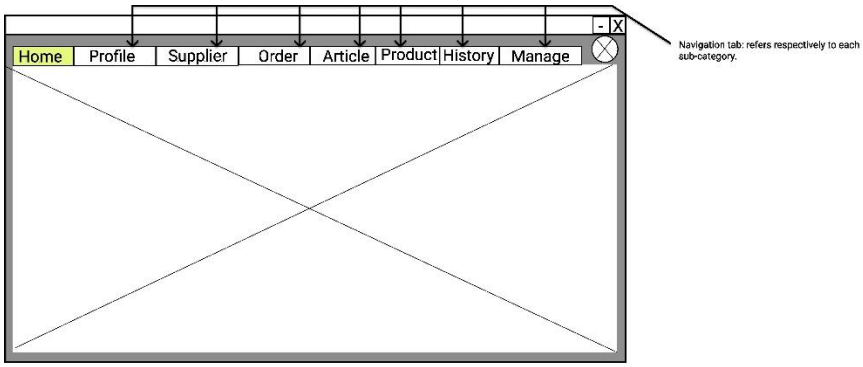

Home
Profile
Supplier
Order
Article
Product
History
Search by order number
Order number
supplier 1
Apple
Bag 1kg
Banana
Banana
Flour
bag 1kg
bag 2kg
bag 5kg
10
20
10
State
Modify
Submit
Create

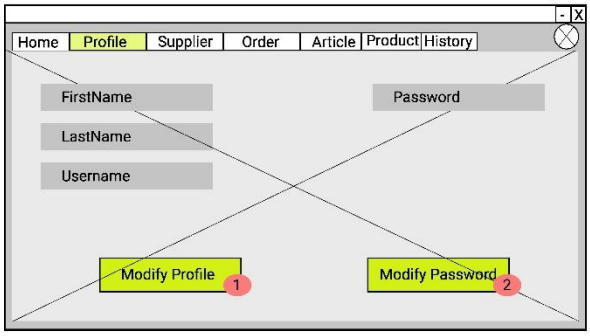

Home
Profile
Supplier
Order
Article
Product
History
Search
ingredient
Banana
Kg
Apple
Flour
Banana
2 articles
Block/Unblock selected Element
Save



StoryBoard


Home
Profile
Supplier
Order
Article
Product
History
Manage
Orders finished
Orders in progress
Order N°123456789
Order N°987654321
Order N°321654987
State
State
State
Submit
This page displays the history about orders
1 Show list of completed orders
2 Show the list of orders in progress
3 Submit the informations
Search
FirstName
LastName
Username
Password
Confirm Password
Modify Profile
Create Admin
Modify Password
Save Profile
Modify Password
NewPassword
Save Password
Frame
Logo
Username
Enter a login
Enter password
Login
At the validation :
1 The Super Admin accesses his home page.HomePage_SuperAdmin
2 Admins access their home pages.
ProfilePage



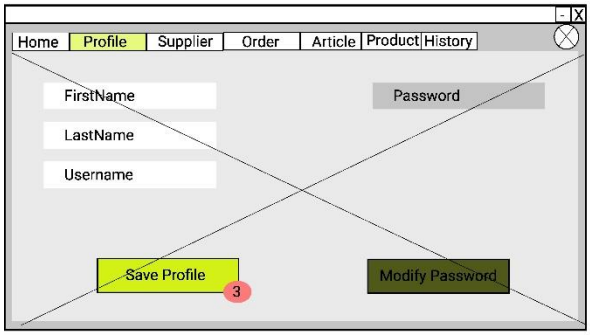


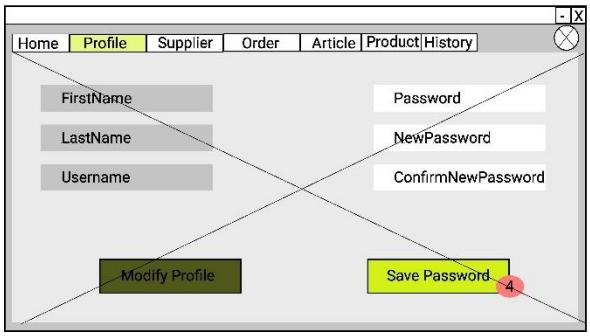
1 Edit profile input fields.
Desable the password button

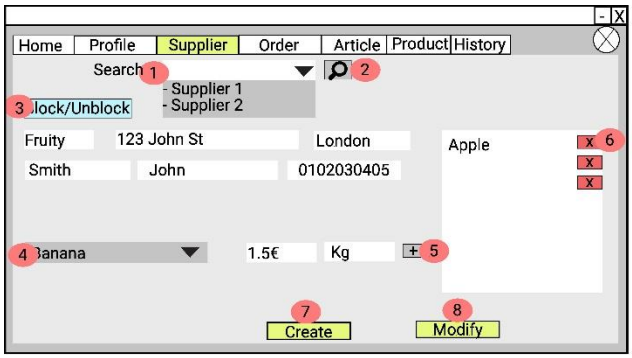
2 Edit password input fields.
Desable the profile button

3 Save input profile changes

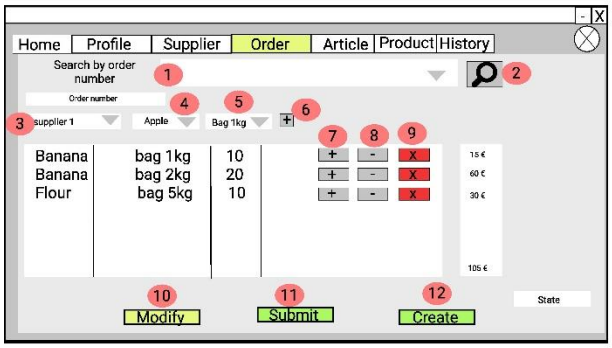
4 Save password changes



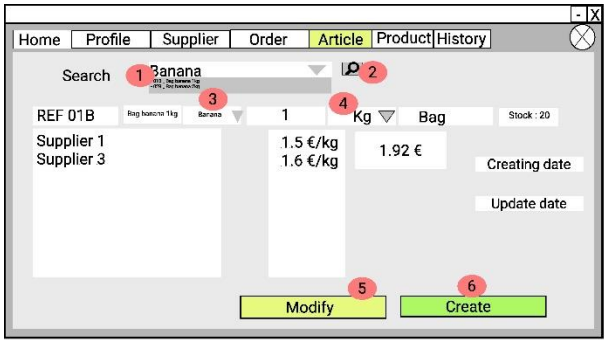




1. Drop down menu to search and/or select the supplier.
2. Launch the search.
3. Blocks the supplier preventing the appearance of its products
4. Scrolling tab to select a product that displays its data in the frame.
5. Button that adds an Product.
6. Button allowing the deletion of an item in the list.
7. Button that creates a supplier.
8. Button that modifies the supplier information.



1. Drop down menu to search an order from its number
2. Launch the search
3. Drop down menu to select a supplier
4. Drop down menu to select a product
5. Drop down menu to select the product unit
6. Icone to add an article
7. Icone to increase article quatity
8. Icone to decrease article quatity
9. Icone to remove an article
10. Save order changes
11. Validate an order
12. Create an order



1. Drop down menu to search an article
2. Launch the search
3. List to choose a product
4. List to choose product unit
5. Save changes of an article
6. Create an article

1. Drop down menu to search a product
2. Launch the search
3. Select an ingredient from the list
4. Icone to add an ingredient
5. Select a unit from the list
6. Change the state of an element
7. Save the products informations

B. ANNEXE PLAN DE QUALITE

Ref-1

https://delperie.needemand.com/realisations/PHP_Nesti_Site/

Ref-2

RewriteEngine ON

RewriteRule ^recipes$ index.php?loc=recipes [L]

RewriteRule ^(articles|statistics)$ index.php?loc=$1 [L]

RewriteRule ^([a-z]+)$ index.php?loc=$1 [L]

#RewriteRule ^recipes/editing$ index.php?loc=recipes&action=editing [L]

RewriteRule ^([a-z]+)/([a-z]+)$ index.php?loc=$1&action=$2 [L]

RewriteRule ^([a-z]+)/([a-z]+)/([a-z\-_0-9]+)$
index.php?loc=$1&action=$2&id=$3 [L]

RewriteRule ^([a-z]+)/([a-z]+)/([a-z\-_0-9]+)/([a-z\-_0-9]+)$
index.php?loc=$1&action=$2&id=$3&supp=$4 [L]

RewriteRule ^([a-z]+)/([a-z]+)/([a-z\-_0-9]+)/([a-z\-_0-9]+)/([a-z\-_0-9]+)$
index.php?loc=$1&action=$2&id=$3&supp=$4&state=$5 [L]

Ref-3

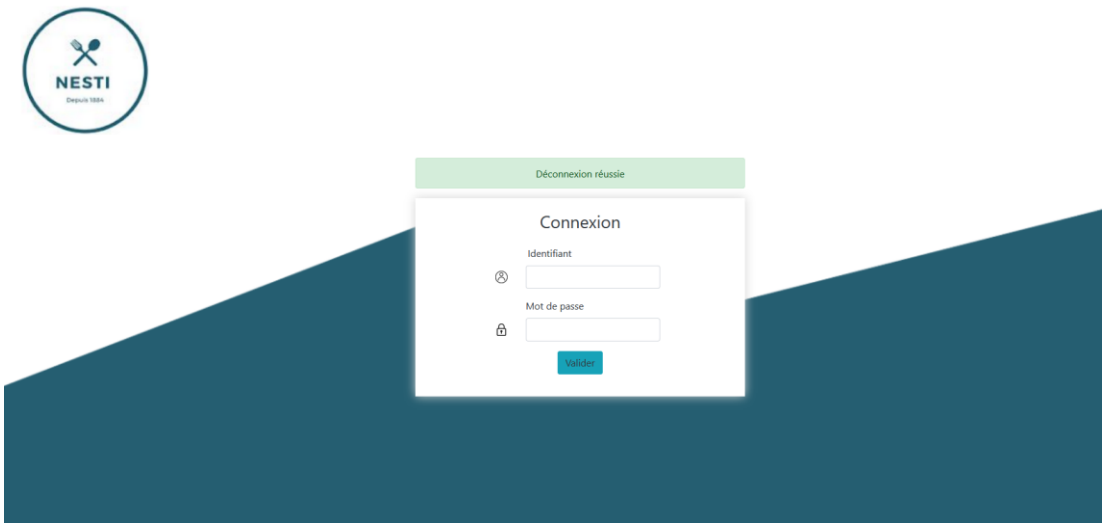


Figure 18 Vue du formulaire de connexion

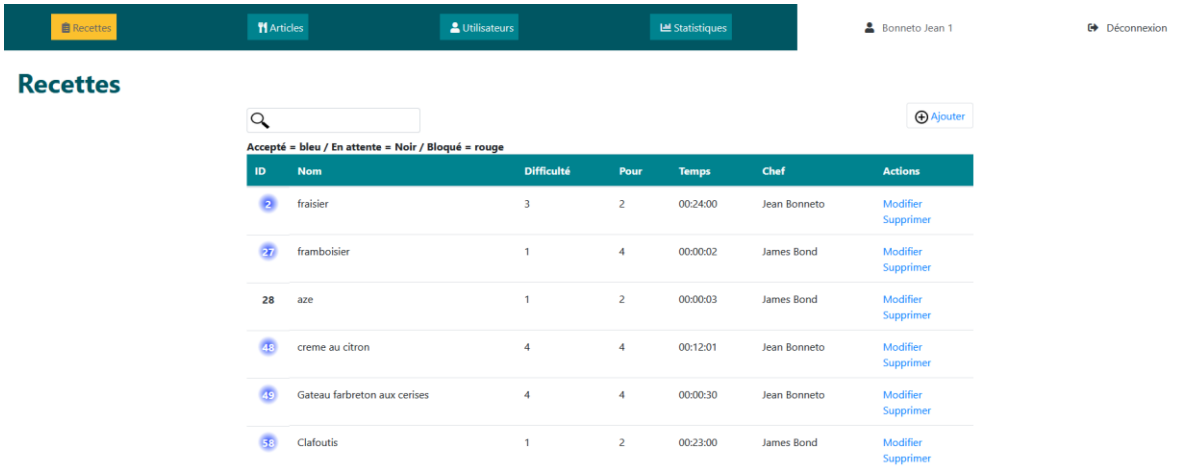


Figure 20 Vue de la page recette

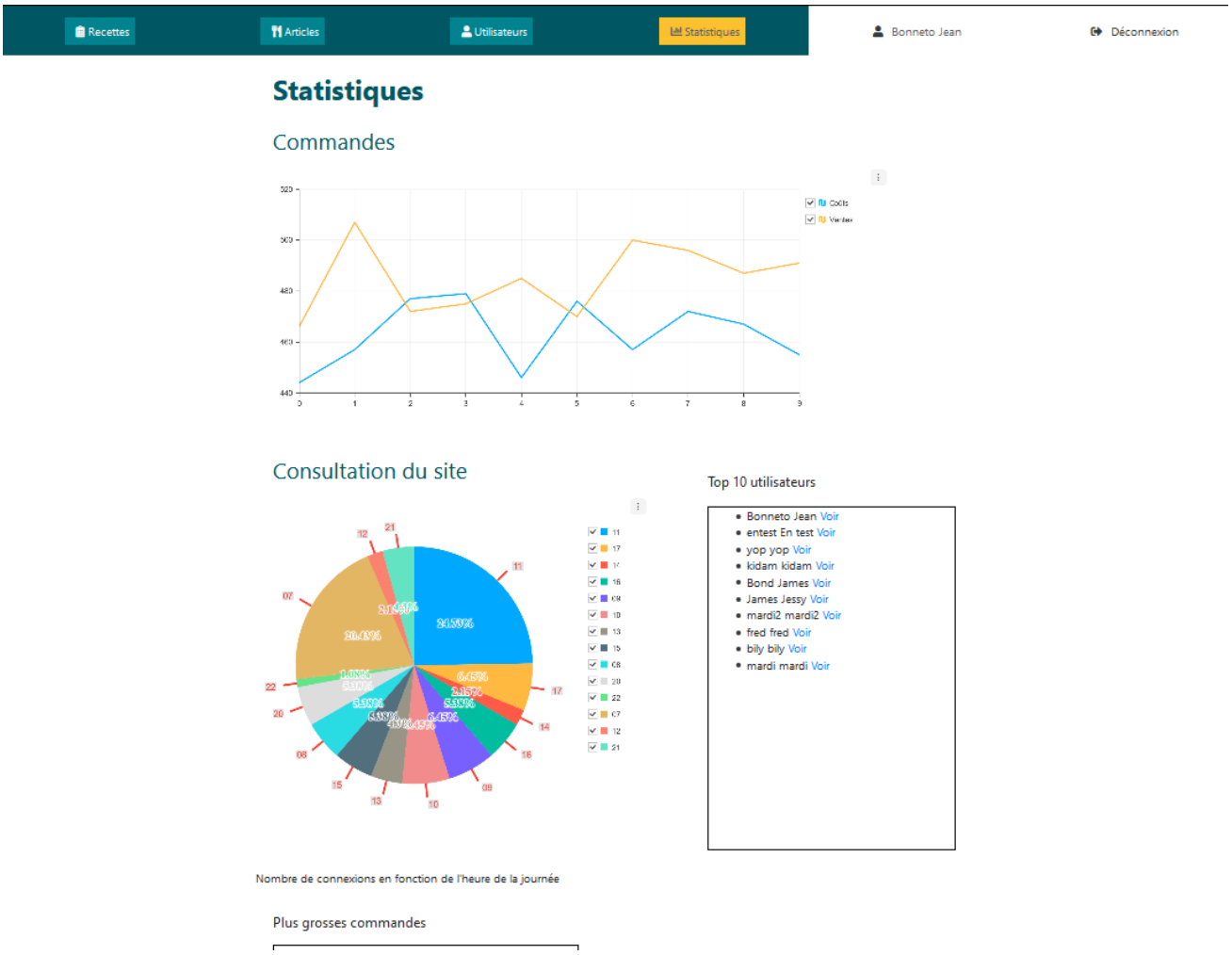


Figure 19Vue de Statistique

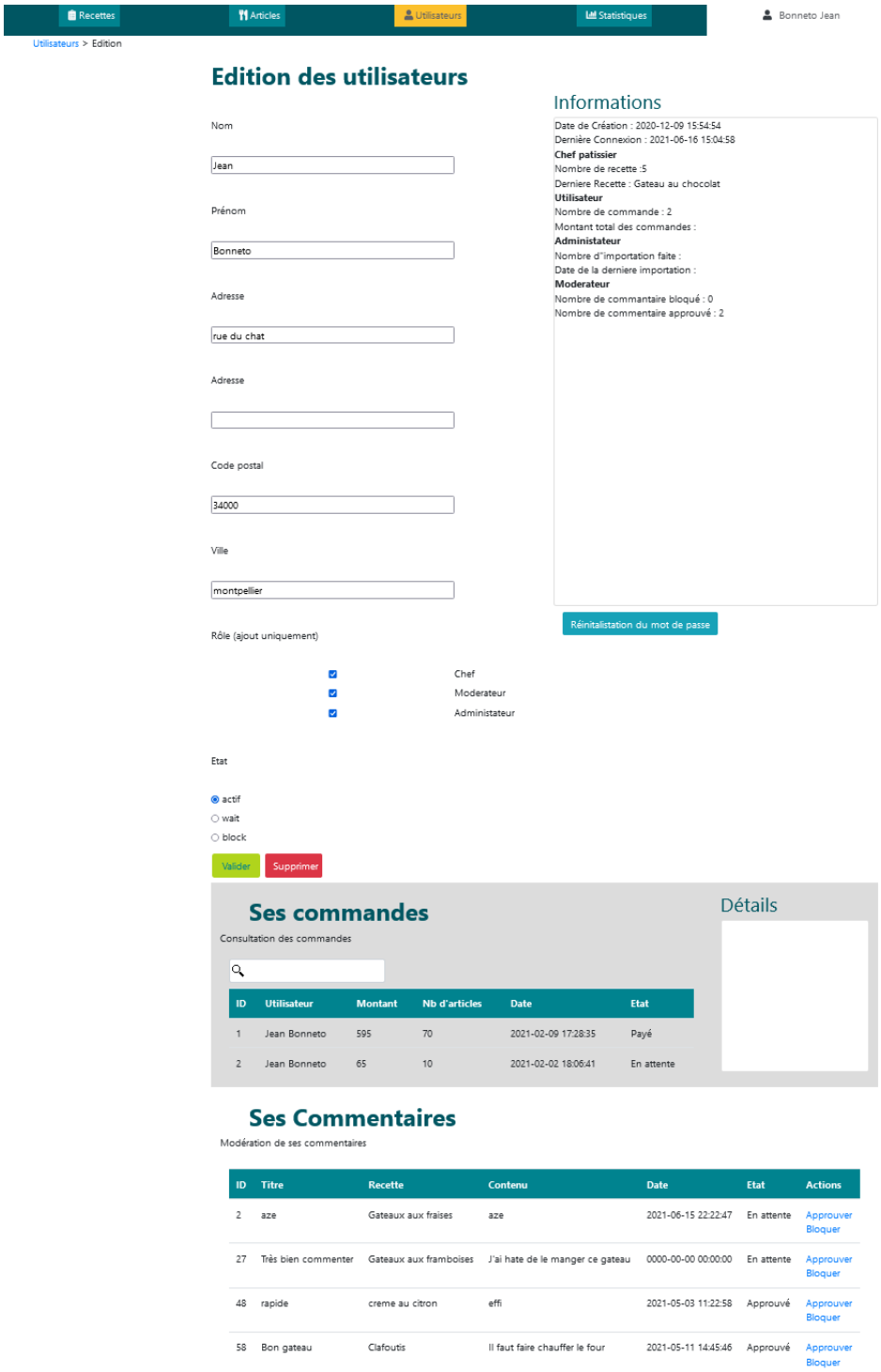


Figure 21Vue couper de Users éditing

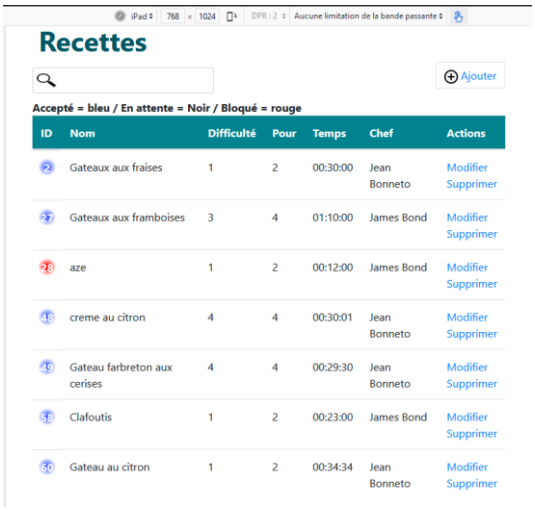


Figure 22 Vue IPAD de Recettes

REF-4

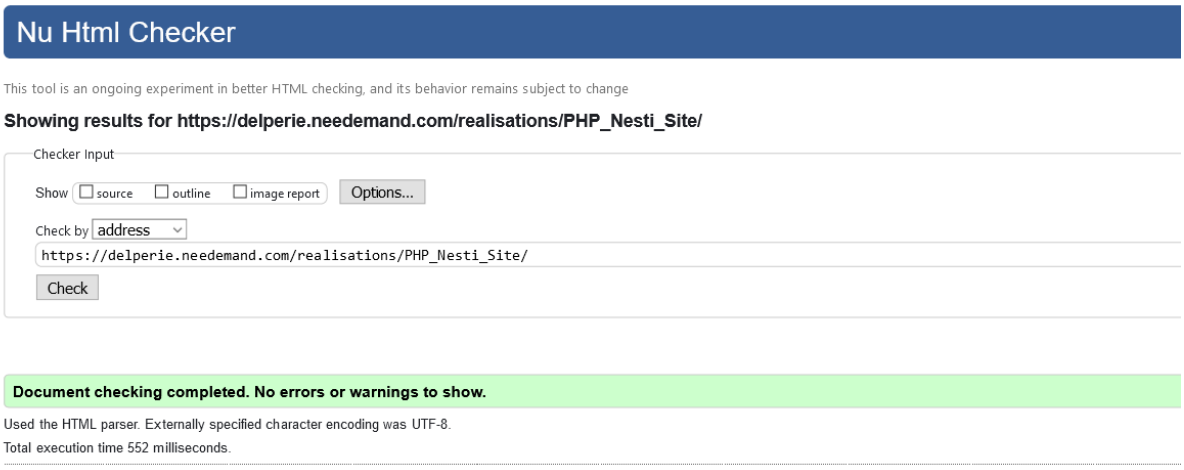


Figure 23 Vue de la validation pour login

Figure 24Vue de la validation pour content_statistic

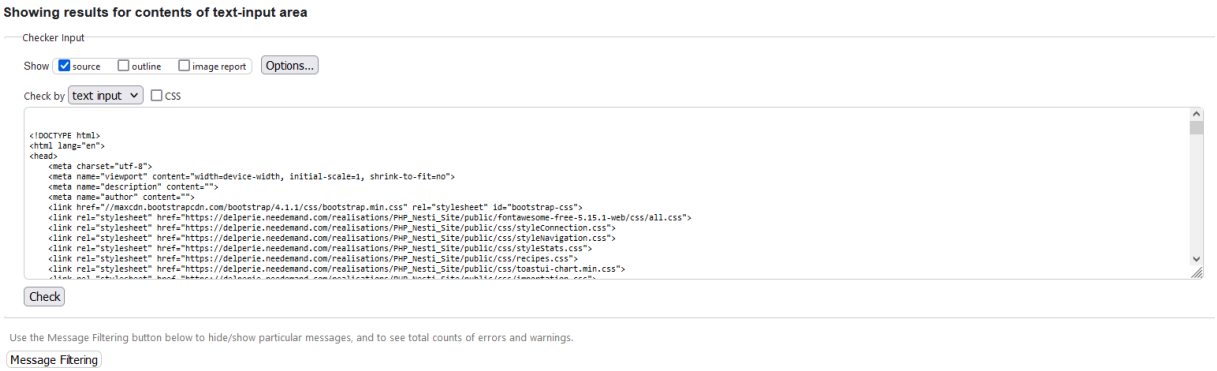


Figure 25 validation Content_edit_recipe

Figure 26Validation recice.css

REF-6

Le code est réalisé en convention de nomenclature *dans son intégralité*

```php
public function editRecipe($idRecipe)
{
  $model                          = new ModelRecipes();
  $recipe                         = $model->readOneBy("idRecipe", $idRecipe);
  $this->data['recipe']           = $recipe;
  $ing                            = new ModelIngredientrecipe();
  $ingredientrecipe               = $ing->readAllBy("idRecipe", $recipe->getIdRecipe());
  $this->data['ingredientrecipe'] = $ingredientrecipe;

  if (isset($_POST["recipeName"])) {

    $recipe->setName(filter_input(INPUT_POST, "recipeName"));
    $recipe->setDifficulty(filter_input(INPUT_POST, "recipedifficult"));
    $recipe->setPortions(filter_input(INPUT_POST, "recipePortion"));
    $recipe->setPreparationTime(filter_input(INPUT_POST, "recipeTimePrepare"));
    $model                          = new ModelRecipes();
    $model->updateRecipes($recipe);
    header('Location:' . BASE_URL . "recipes/editing/" . $idRecipe);
  }
```

REF-7

```php
}
//=============
// readOneBy
//=============
/**
 * Read ingredientrecipe with ele1 at value ele2
 *
 *  $parametrer
 *  $value
 *  return object ingredientrecipe
 */
public function readOneBy($parameter, $value)
{
    //requete
    $pdo = Connection::getPdo();

    $sql = "SELECT * FROM ingredientrecipe where $parameter = '$value'";

    $result = $pdo->query($sql);

    if ($result) {

        $data = $result->fetch(PDO::FETCH_ASSOC);
    } else {

        $data = [];
    }

    $user = new Ingredientrecipe();
    $user->setIngredientRecipeFromArray($data);

    return $user;
}
//=============
// readAllBy
//=============
```

REF-9

https://github.com/Delperie-yann/PHP_Nesti_Site

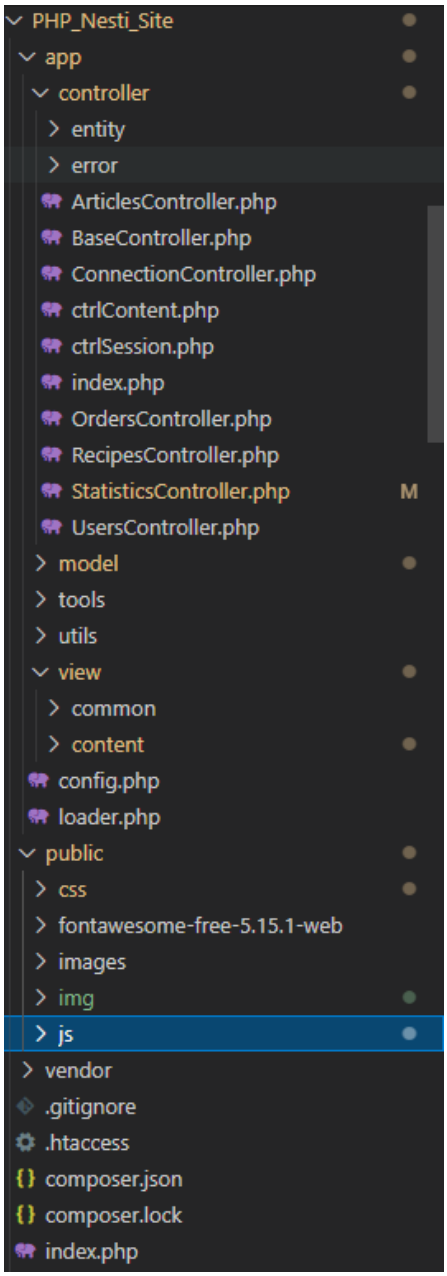
REF-10

```php
public function insertRecipe(Recipes &$recipe)
{
    $pdo = Connection::getPdo();
    try {
        // Create prepared statement
        $sql = "INSERT INTO recipe (name, difficulty,portions,flag,preparationTime,idChef) VALUES (?,?,?,?,?,?)";

        $stmt = $pdo->prepare($sql);

        $values = [$recipe->getName(), $recipe->getDifficulty(), $recipe->getPortions(), $recipe->getFlag(), $recipe->getPreparationTime(), $recipe->getIdC
        // Execute the prepared statement
        $stmt->execute($values);
        $newRecipe = $this->readOneBy("idRecipe", $pdo->lastInsertId());

    } catch (PDOException $e) {
        die("ERROR: Could not able to execute $sql. " . $e->getMessage());
    }
    unset($pdo);
    return $newRecipe;
}
```

REF-11

☐	Edit Copy Delete	1	Jean	Bonneto	jeanbon@gmail.com	$2y$10$UCE.R2etOdOduD4koc2bnOcsw.NH4B4N08WZ5w3YmKL...
☐	Edit Copy Delete	2	James	Bond		$2y$10$UCE.R2etOdOduD4koc2bnOcsw.NH4B4N08WZ5w3YmKL...
☐	Edit Copy Delete	3	yop	yop	yipyop34@hotmail.fr	$2y$10$CJSMlHkJZVSPk3Oq1SCDwOkuC6ts6.SphyGgKq1hsYd...
☐	Edit Copy Delete	71	En test	entest	State@hot.com	$2y$10$9mGXkoHtZ.JCOG0i4lPnUOSVC7ZuZfJ9tMcbUjcihn1...
☐	Edit Copy Delete	76	aze	aze	tototototot@hotmail.fr	~?4`?h????E?^%??N?`?vG0?Go??|??x?]n?$o[?5□□+...
☐	Edit Copy Delete	85	mardi	mardi	mardi@hot.com	$2y$10$lZqVQWdqSNhNTXGaevkoOOMCkNt7p.ITQC.yXT.w9q...
☐	Edit Copy Delete	86	mardi2	mardi2	aze@hot.com	$2y$10$QutxHQyxR63tvoWmk2a5JeK4RPW6uOZR/HWe0TJvc8c...
☐	Edit Copy Delete	112	kidam	kidam	kidam@hot.com	$2y$10$/871g9mlrGL84g6p9zcRueZq97uJd7rugVWsU32mvP1...
☐	Edit Copy Delete	117	fred	fred	fred@hotmail.fr	$2y$10$tlQ1WrOT/kA6pfpC8p84R.G/ofTaNMhzRa5ll0q89SO...
☐	Edit Copy Delete	118	bily	bily	bily@hot.com	$2y$10$/lYhzhir5hr1zjJ7btEG0O8yWcAKh6i1L6xhJxDTMax...
☐	Edit Copy Delete	119	Testcode	Testcode	123tot@hotmail.fr	X?v??B??(?O*??????q?2??hm???U???74h?&???M?...

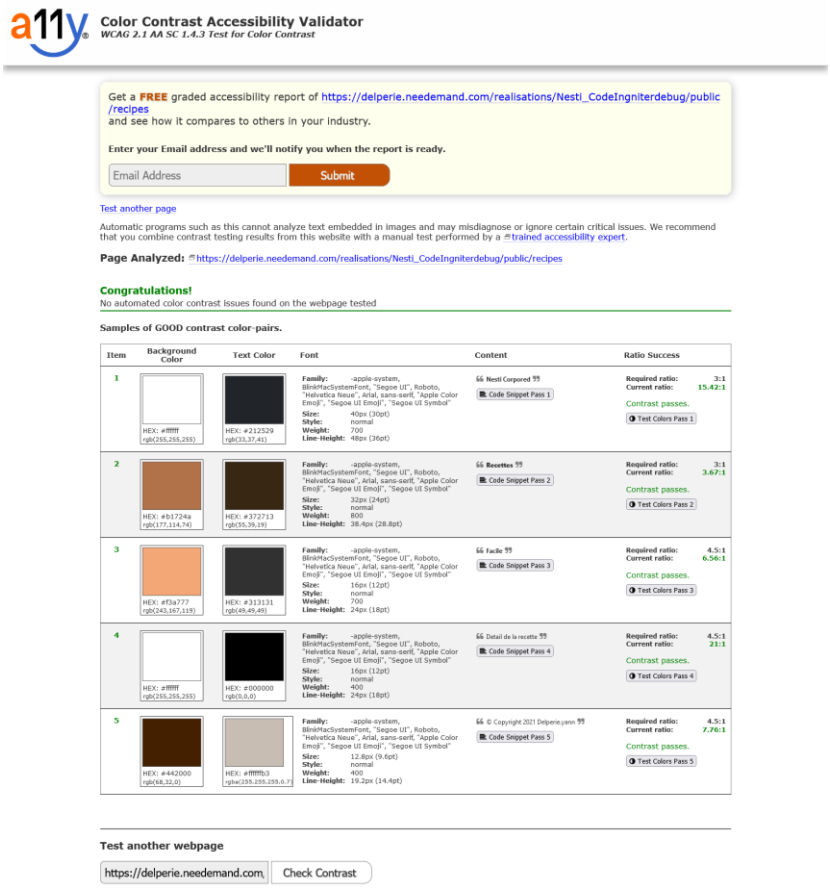
Ref-12

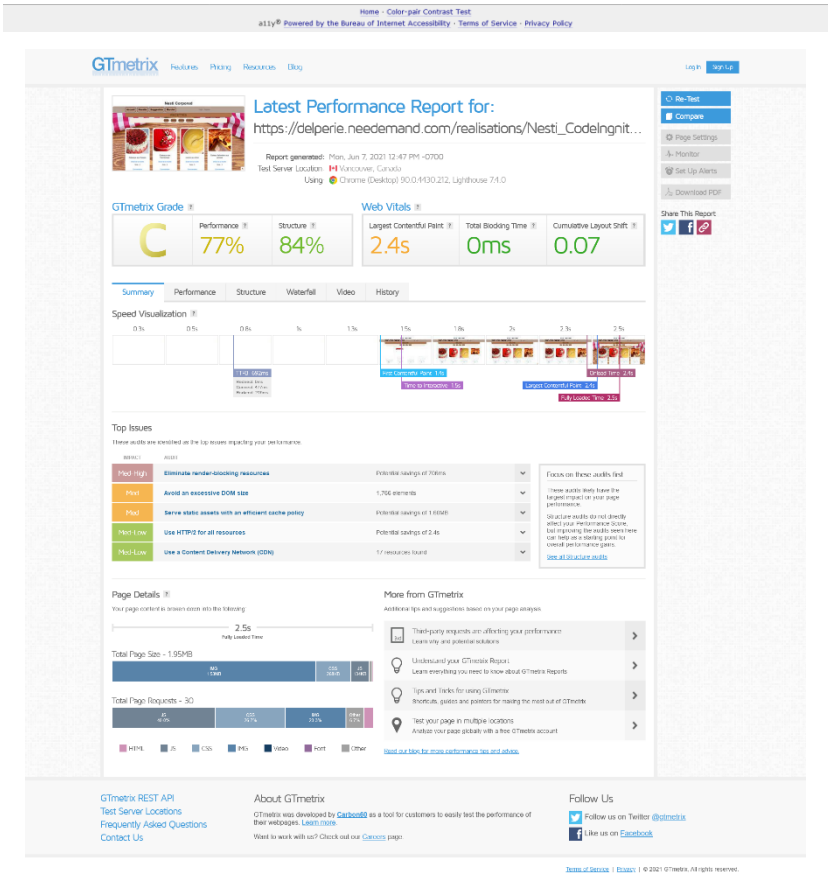


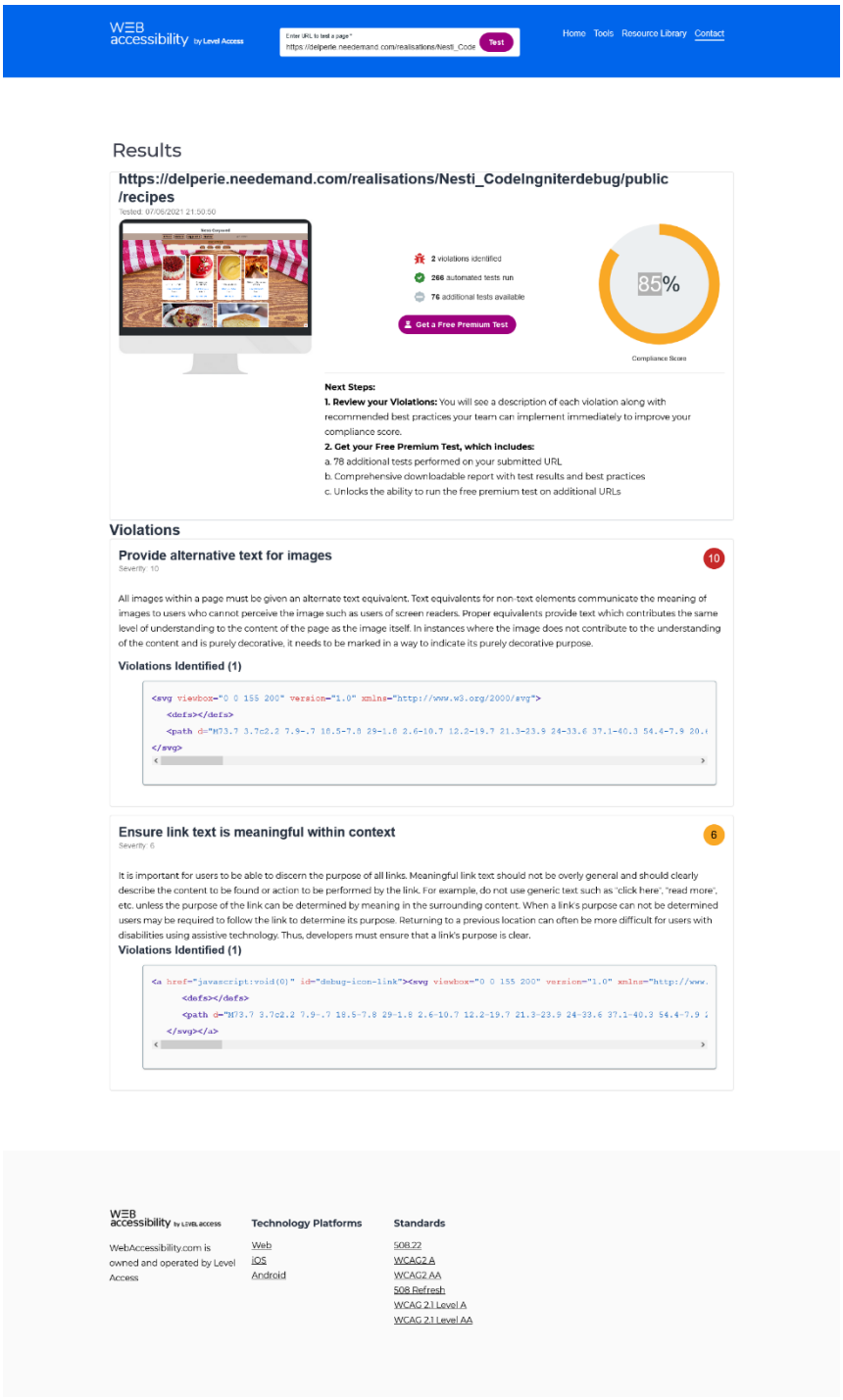
REF-13

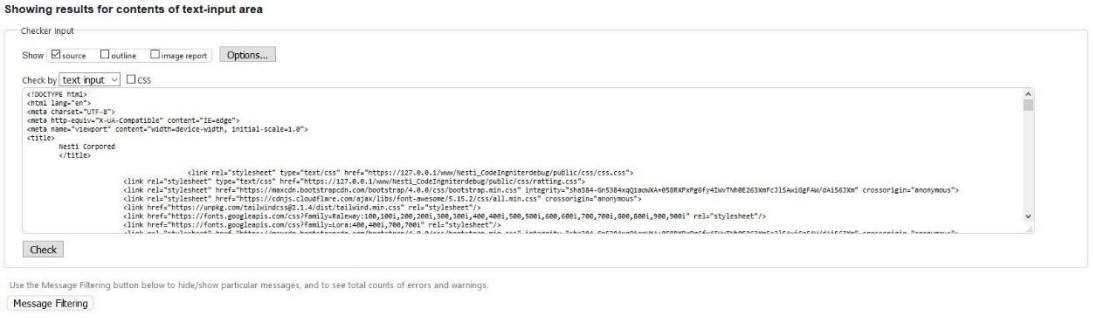
121	REF13test2 REF13test2	Chef, Moderateur	REF13test2@hotmail.fr	-	w	Modifier Supprimer

C. PLAN DE TEST CODEINGITER

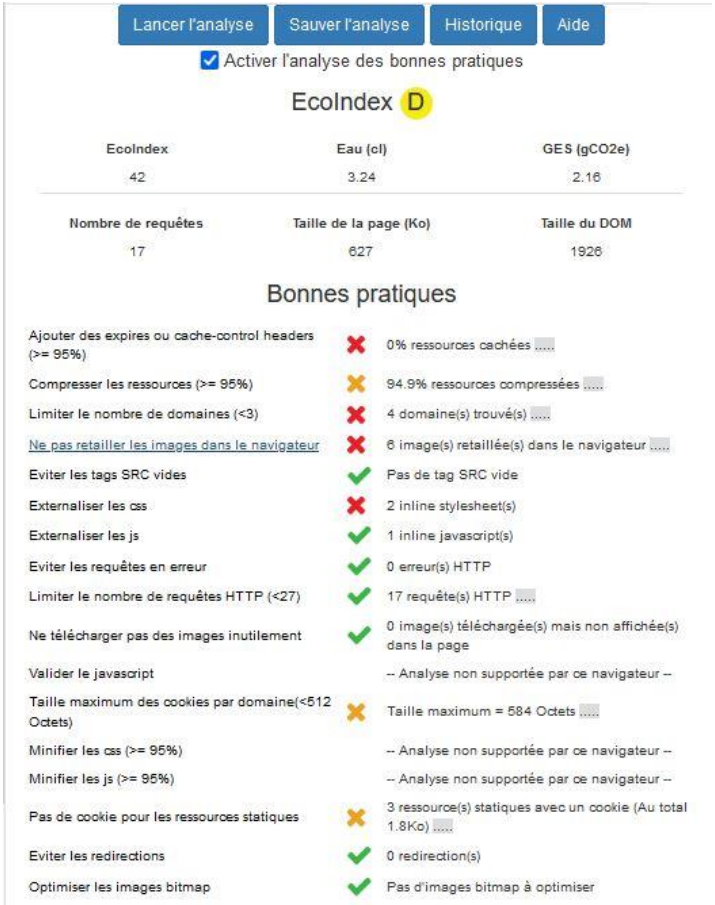





WEB accessibility by Level Access
Enter URL to test a page *
https://delperie.needemand.com/realisations/Nesti_Code
Test
Home Tools Resource Library Contact
Results
https://delperie.needemand.com/realisations/Nesti_CodeIngniterdebug/public/recipes
Tested: 07/06/2021 21:50:50
2 violations identified
266 automated tests run
76 additional tests available
Get a Free Premium Test
85%
Compliance Score
Next Steps:
1. Review your Violations: You will see a description of each violation along with recommended best practices your team can implement immediately to improve your compliance score.
2. Get your Free Premium Test, which includes:
a. 78 additional tests performed on your submitted URL
b. Comprehensive downloadable report with test results and best practices
c. Unlocks the ability to run the free premium test on additional URLs
Violations
Provide alternative text for images
Severity: 10
10
All images within a page must be given an alternate text equivalent. Text equivalents for non-text elements communicate the meaning of images to users who cannot perceive the image such as users of screen readers. Proper equivalents provide text which contributes the same level of understanding to the content of the page as the image itself. In instances where the image does not contribute to the understanding of the content and is purely decorative, it needs to be marked in a way to indicate its purely decorative purpose.
Violations Identified (1)
<svg viewbox="0 0 155 200" version="1.0" xmlns="http://www.w3.org/2000/svg">
<defs></defs>
<path d="M73.7 3.7c2.2 7.9-.7 16.5-7.8 29-1.8 2.6-10.7 12.2-19.7 21.3-23.9 24-33.6 37.1-40.3 54.4-7.9 20.
</svg>
Ensure link text is meaningful within context
Severity: 6
6
It is important for users to be able to discern the purpose of all links. Meaningful link text should not be overly general and should clearly describe the content to be found or action to be performed by the link. For example, do not use generic text such as "click here", "read more", etc. unless the purpose of the link can be determined by meaning in the surrounding content. When a link's purpose can not be determined users may be required to follow the link to determine its purpose. Returning to a previous location can often be more difficult for users with disabilities using assistive technology. Thus, developers must ensure that a link's purpose is clear.
Violations Identified (1)
<svg viewbox="0 0 155 200" version="1.0" xmlns="http://www.
<defs></defs>
<path d="M73.7 3.7c2.2 7.9-.7 18.5-7.8 29-1.8 2.6-10.7 12.2-19.7 21.3-23.9 24-33.6 37.1-40.3 54.4-7.9 2
</svg>
WEB accessibility by LEVEL access
WebAccessibility.com is owned and operated by Level Access
Technology Platforms
Web
iOS
Android
Standards
508.22
WCAG2 A
WCAG2 AA
508 Refresh
WCAG 2.1 Level A
WCAG 2.1 Level AA


Showing results for contents of text-input area
Checker Input
Show source outline image report Options...
Check by text input CSS
Check
Use the Message Filtering button below to hide/show particular messages, and to see total counts of errors and warnings.
Message Filtering
Document checking completed. No errors or warnings to show.
Source


This tool is an ongoing experiment in better HTML checking, and its behavior remains subject to change
Showing results for contents of text-input area
Checker Input
Show source outline image report Options...
Check by text input CSS
Check
Use the Message Filtering button below to hide/show particular messages, and to see total counts of errors and warnings.
Message Filtering
Document checking completed. No errors or warnings to show.


Lancer l'analyse
Sauver l'analyse
Historique
Aide
Activer l'analyse des bonnes pratiques
EcoIndex D
EcoIndex
Eau (cl)
GES (gCO2e)
42
3.24
2.16
Nombre de requêtes
Taille de la page (Ko)
Taille du DOM
17
627
1926
Bonnes pratiques
Ajouter des expires ou cache-control headers (>= 95%)
0% ressources cachées ....
Compresser les ressources (>= 95%)
94.9% ressources compressées ....
Limiter le nombre de domaines (<3)
4 domaine(s) trouvé(s) ....
Ne pas retailler les images dans le navigateur
6 image(s) retaillée(s) dans le navigateur ....
Eviter les tags SRC vides
Pas de tag SRC vide
Externaliser les css
2 inline stylesheet(s)
Externaliser les js
1 inline javascript(s)
Eviter les requêtes en erreur
0 erreur(s) HTTP
Limiter le nombre de requêtes HTTP (<27)
17 requête(s) HTTP ....
Ne télécharger pas des images inutilement
0 image(s) téléchargée(s) mais non affichée(s) dans la page
Valider le javascript
-- Analyse non supportée par ce navigateur --
Taille maximum des cookies par domaine(<512 Octets)
Taille maximum = 584 Octets ....
Minifier les css (>= 95%)
-- Analyse non supportée par ce navigateur --
Minifier les js (>= 95%)
-- Analyse non supportée par ce navigateur --
Pas de cookie pour les ressources statiques
3 ressource(s) statiques avec un cookie (Au total 1.8Ko) ....
Eviter les redirections
0 redirection(s)
Optimiser les images bitmap
Pas d'images bitmap à optimiser

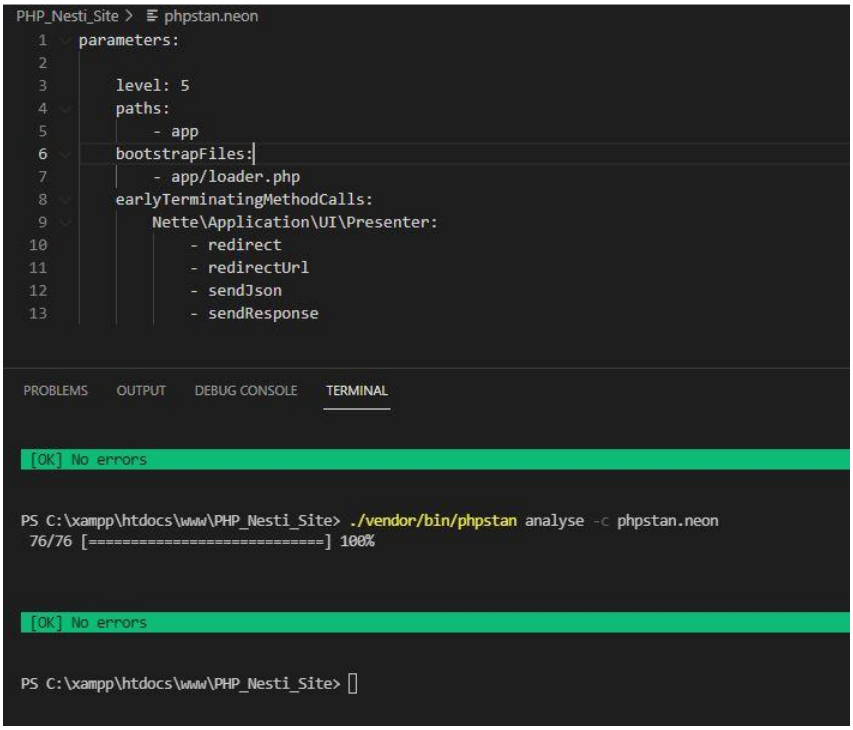
D. MAQUETTE

E. TEST PHPSTAN

```
 Line   Models\TokenModel.php
 ------ ------------------------------------------------------------------------------------------------------------
 10     Property App\Models\TokenModel::$allowedFields type has no value type specified in iterable type array.
        💡 See: https://phpstan.org/blog/solving-phpstan-no-value-type-specified-in-iterable-type
 ------ ------------------------------------------------------------------------------------------------------------

[ERROR] Found 161 errors

PS C:\xampp\htdocs\www\Nesti_CodeIngniterdebug>
```

```
1   parameters:
2
3       level: 6
4       bootstrapFiles:
5           - system/Common.php
6
7       paths:
8           - app/Controllers
9           - app/Models
10          - app/Entities
11
```

```
PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL

 Line   Controllers\UserController.php
 ------ ------------------------------------------------------------------------------------------------------------
 10     Property App\Controllers\UserController::$loggedInUser has no typehint specified.
 11     Property App\Controllers\UserController::$dataCart has no typehint specified.
 23     Access to an undefined property App\Controllers\UserController::$twig.
 35     Call to static method encrypter() on an unknown class Config\Services.
        💡 Learn more at https://phpstan.org/user-guide/discovering-symbols
 51     Access to an undefined property App\Controllers\UserController::$twig.
 54     Access to an undefined property App\Controllers\UserController::$twig.
 58     Access to an undefined property App\Controllers\UserController::$twig.
 68     Call to static method encrypter() on an unknown class Config\Services.
        💡 Learn more at https://phpstan.org/user-guide/discovering-symbols
 89     Call to static method encrypter() on an unknown class Config\Services.
        💡 Learn more at https://phpstan.org/user-guide/discovering-symbols
 90     Method App\Controllers\UserController::setLoggedInUser() should return object but return statement is missing.
 108    Call to static method encrypter() on an unknown class Config\Services.
        💡 Learn more at https://phpstan.org/user-guide/discovering-symbols
 154    Method App\Controllers\UserController::register() with return type void returns CodeIgniter\HTTP\RedirectResponse but should not return anything.
 158    Access to an undefined property App\Controllers\UserController::$twig.
 167    Access to an undefined property App\Controllers\UserController::$twig.
 178    Parameter #2 $value of function setcookie expects string, null given.
 179    Parameter #2 $value of function setcookie expects string, null given.
 182    Access to an undefined property App\Controllers\UserController::$twig.
 ------ ------------------------------------------------------------------------------------------------------------


 Line   Entities\Users.php
 ------ ------------------------------------------------------------------
 31     Access to an undefined property App\Entities\Users::$passwordHash.
 ------ ------------------------------------------------------------------


[ERROR] Found 79 errors

PS C:\xampp\htdocs\www\Nesti_CodeIngniterdebug>
```



F. JavaScript documentation

Projet Nesti 1

Documentation

I.a index

Pour commencer par le commencement. Tout d'abord il y a la création de l'HTML suivit des icônes de fenêtre pour tous les formats d'appareils.

```
<!DOCTYPE html>
<html lang="fr">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Nesti</title>
    <link
```

```
      rel="shortcut icon"
      href="./images/icones/favicon.ico"
      type="image/x-icon"
    />
    <link rel="icon" href="./images/icones/favicon.png" type="image/png" />
    <link
      rel="icon"
      sizes="32x32"
      href="./images/icones/favicon-32.png"
      type="image/png"
    />
    <link
      rel="icon"
      sizes="64x64"
      href="./images/icones/favicon-64.png"
      type="image/png"
    />
```

Suivit des différents imports lier aux fonctionnements :

```
<meta name="msapplication-TileColor" content="#FFFFFF" />

    <link rel="stylesheet" href="./styles/css-reset.css" />
```

Importation du plugin tailwinds

```
<link
      href="https://unpkg.com/tailwindcss@^2/dist/tailwind.min.css"
      rel="stylesheet"
    />
```

Importation des fronts

```
    <link
      href="https://fonts.googleapis.com/css?family=Raleway:100,100i,200,200i,
300,300i,400,400i,500,500i,600,600i,700,700i,800,800i,900,900i"
      rel="stylesheet"
    />
    <link
      href="https://fonts.googleapis.com/css?family=Lora:400,400i,700,700i"
      rel="stylesheet"
```

I.b Le main

Le main est composé de trois éléments principaux :

- La modal

```
 <div id="myModal" class="modal">
        <div class="borderModal rounded">
          <div class="modal-content">
```

- La grille de recette

```
<div
            class="repice grid grid-cols-4 md:grid-cols-5 gap-1 md:gap-4 mt-
0 md:mt-20 mb-10 md:mb-0 place-self-center"
          ></div>
```

- Le corps pour les cartes.

```
 <div class="grid lg:grid-cols-5 grid-cols-3 justify-center">
        <div class="flex flex-col lg:col-span-3 col-span-3 mt-2">
          <h2>Choisissez vos ingrédients</h2>
          <div class="flex flex-col justify-center">
```

Puis le « footer » et enfin les scripts.
Concernant le fonctionnement l'ensemble est principalement géré par les scripts javascript.

II. Le recipe.js et le creatObjets.js

Le fichier « recipe.js » à comme principale fonction de « stocker » les informations. Il est construit sous la forme :

```
var repices = [
  {
      "id":
      "name":
       "number":
      "time":
```

```
      "picture":
      "ingredients": [ , ],
      "lists": {
                  "Principale": [,]
            },
      "preparations": {
            "principale": [ , ],
      "Conseil":
            }
  }, Element 2, ect …]
```

Ce format contient l'ensemble des valeurs qui peuvent être appeler dans les fonctions afin de faciliter le code.

Dans le « CreateObjects.js » les informations « repices » sont extraites puis stockées sous forme de tableau

```
var arrayRepices = [];
repices.forEach((repice) => {
  arrayRepices.push(new Recipe(repice));
```

puis ont crée la liste des ingrédients

```
var listIngredients = [] ;
```

et ont fait de même.

```
var arrayCards = [];
listIngredients.forEach((ingredient) => {
  arrayCards.push(new Card(ingredient));
});
```

III. ListObjects.js

Dans ce fichier les objet Card et recipe sont créé afin d'attribuer les « informations » de chaque catégorie du Json.

```
class Recipe {
  constructor(repice) {…}

class Card {
  constructor(ingredient) {...}
```

La fonction `validRecipe(ingredients)` compare les ingrédients validés par l'utilisateur avec ceux disponible dans les recettes. Par défaut, tant que la liste des ingrédients sélectionnée par l'utilisateur 'keep' est vide elle retourne « False ».
`changeValid(value)` est utilisé pour modifier la valeur de `this.valid.`
Cela qui permettra dans la fonction `recipeDisplay()` d'aller chercher les recettes par l'intermédiaire de `searchRecipe()`.

`checkWriting(text)`
Cette fonction est présente afin d'éviter les confies dans le traitement des mots, elle supprime les accents et majuscule.

IV. function.js

Tout d'abord, une petite explication s'impose sur le fonctionnement. Lorsque l'utilisateur arrive sur la page, il peut cliquer sur Sweep ou Keep mais au niveau code l'ensemble des cartes sont déjà encodé et seul les 2 premières sont physiquement présente. Au moment du clique, la première page s'en va et fait son effet pour revenir en fin...A ce moment la carte «3 » est physiquement crée (pour anticiper et ne pas créer un vide pour un nouveau clique).
`function possitionCard(pos)`

Rentrons dans le détail des fonctions...

```
function countCard() {
  let arrayCardValid = [];
  arrayCards.forEach((ingredient) => {
    if (ingredient.valid) {
      arrayCardValid.push(ingredient.name.toLowerCase());
    }
  });
  return arrayCardValid;
}
```

Cette fonction permet de conserver une liste (avec un texte épuré `toLowerCase` afin d'éviter des erreurs) des cartes validées par l'utilisateur.

La fonction suivante utilise le retour de `countCard() >> arrayCardValid`

```
function searchRecipe() {
  var cards = countCard();
  for (var repice in arrayRepices) {
    arrayRepices[repice].validRecipe(cards);
```

Son but est de valider les recettes `validRecipe` selon le choix des ingrédients utilisateur `arrayRepices`.

function randomize(array) {}

Cette fonction crée un ordre aléatoire et est utilisé afin que les « Card » et les « recipes » ne soit pas toujours présent dans le même ordre initial.

```
function viewportSize() {}
```

Cette fonction contrôle la taille de la fenêtre utilisateurs et renvoie ses dimensions.

```
function possitionCard(pos) {}
```

Cette fonction conditionne les positions, image et taille de texte dans la 'grille' concernant les « Cards ». Le « Style » par défaut étant géré pour des petit écrans l'ensemble est adapté plus pour des écrans « grande taille ». La gestion des images pour le 'responsif' est également géré dans cette partie afin que les images ne prennent pas trop de place.

```
function moveCard(way, move)
```

Cette fonction à pour rôle de crée un effet de mouvement de rotation de lors de la translations de la première carte.

```
function recipeDisplay()
```

Fonction qui génère la partie HTML et gère le conditionnement de l'ouverture de la modal.
Elle fait appel à `searchRecipe` afin d'extraire les informations nécessaires pour la constitution de la grille « recipe ».

```
function openModal(id)
```

La fonction construit la modal en HTML et gère la récupération des donnée Json. Les différentes boucles permettent de parcourir les listes et de partitionnée les informations.

```
      if (repiceModal.valid)
```

Cette partie de la fonction crée toute la partie haute de la modal (avant image). Pour la suite de la fonction il faut récupérer les informations contenues dans le Json. On parcourt chacune des listes par des boucles « for » pour les entêtes et on parcourt de nouveau pour intégrer le texte « informatif ». Puis on injecte dans le `modal.innerHTML = div;`

```
//Click condition on the modal
    var close = document.querySelector(".close");
    close.onclick = function () {
      modal.style.display = "none";
    };
    window.onclick = function (event) {
      if (event.target == modal) {
        modal.style.display = "none";
      }
    };
    var close = document.querySelector(".modal");
    modal.onclick = function (event) {
      modal.style.display = "none";
    };
```

Ces dernières fonctions gèrent la fermeture de la modal soit par le bouton « Close », soit en cliquant sur la fenêtre ou bien la modal.

V. Management.js

Ce fichier est le fichier principal qui regroupe les fonctionnalités et interactivités.

```
var posRotateCard = ["rotate-1", "rotate-2", "-rotate-1", "-rotate-
2"];
```

```
let widthtWindow = viewportSize().width;
if (widthtWindow < 550) {
  var posRotate = posRotateCard;
  var move = 500;
} else {
  var posRotate = ["rotate-6", "rotate-12", "-rotate-6", "-rotate-12"];
  var move = 650;
}
```

Limite le déplacement de la Card et la rotation des autres lors de la translations en fonction de la taille de la fenêtre.

```
for (let i = 0; i < cards.length; i++) {
  cards[i].style.zIndex = i + 2;
}
cardsI.style.zIndex = 10;
```

Cette partie fait en sorte que les cartes Soit devant la grille et ranger dans l'orde.

```
arrayCards = randomize(arrayCards);
possitionCard(posCard);
```

Lance l'effet aléatoire des « Cards»

```
keep.addEventListener("click", function () {
  arrayCards[posCard].changeValid(true);
  posCard++;
  if (posCard == arrayCards.length) {
    posCard = 0;
  }
  moveCard("+", move);
  recipeDisplay();
});
```

Ajout de l'attribut au bouton Keep de lancer l'action de valider la recette de passer à la Card suivante et de relancer la création de la grille de recette à partir des données ingrédients conserver.

```
randomize(arrayRepices);
recipeDisplay();
```

Crée l'effet aléatoire des recettes et crée la grille initial.

VI. Les éléments extérieurs

`tailwindcss` : https://tailwindcss.com/

Tailwind est un framework qui nous a permis de crée plus facilement les Cards. Son fonctionnement est un 'responsif' inversé : il se base sur une window de petite taille. Il faut donc, non pas adapter le « style » pour un téléphone (exemple boostap) mas l'inverse.

VII. Les sources d'images

https://www.pexels.com
https://pixabay.com/fr/