

Nom : BENALI

Prénom : MANEL

Sommaire

Liste des compétences du référentiel couvertes par le projet 3

Résumé du projet en anglais 6

Présentation du projet 7

Cahier des charges 8

I. Présentation d’ensemble du projet 8

II. Description de la charte graphique 9

III. Prestations attendues et modalités de sélection des prestataires 10

Gestion du projet 11

I. Planning et suivi 11

II. Environnement humain et technique 12

III. Objectifs de qualité 14

Spécifications fonctionnelles du projet 15

Spécifications techniques du projet 16

Réalisations 17

I. Exemple 1 17

II. Exemple 2 20

III. Exemple 3 23

Présentation d’un jeu d’essai 26

I. Données en entrée 26

II. Données attendues 27

III. Données obtenues 28

Veille sur les vulnérabilités de sécurité 29

I. Besoin d’information 29

II. Mots clés de recherche 30

III. Liste des sites retournés 31

IV. Critères de sélection des sites 32

V. Vulnérabilités trouvées 33

VI. Failles potentielles corrigées 34

Description d’une situation de travail ayant nécessité une recherche 35

I. Besoin d’information 35

II. Mots clés de recherche 36

III. Liste des sites retournés 37

IV. Critères de sélection des sites 38

V. Solution et mise en œuvre 39

Conclusion 40

Annexes 41

# Liste des compétences du référentiel couvertes par le projet

**Activité type 1 « Concevoir et développer des composants d'interface utilisateur en intégrant les recommandations de sécurité »**

* Maquetter une application
* Développer des composants d’accès aux données
* Développer la partie *front-end* d’une interface utilisateur web
* Développer la partie *back-end* d’une interface utilisateur web

**Activité type 2 « Concevoir et développer la persistance des données en intégrant les recommandations de sécurité »**

* Mettre en place une base de données

**Activité type 3 « Concevoir et développer une application multicouche répartie en intégrant les recommandations de sécurité »**

* Concevoir une application
* Développer des composants métier
* Construire une application organisée en couches

# Résumé du projet en anglais

# Présentation du projet

# Cahier des charges

## Présentation d’ensemble du projet

Description du projet :

L’objectif du blog de recettes, est de partager et consulter des recettes culinaires.

Fonctionnalités principales :

Gestion des utilisateurs : inscription, connexion, profil utilisateur.

Publication de recettes : permettre aux utilisateurs de publier leurs propres recettes.

Consultation de recettes : permettre aux utilisateurs de rechercher, afficher et commenter les recettes.

Catégorisation des recettes : organiser les recettes par catégories (entrées, plats principaux, desserts, etc.).

Système de notation : permettre aux utilisateurs de noter les recettes.

Recherche avancée : possibilité de rechercher des recettes par ingrédients, temps de préparation, etc.

Système de favoris : permettre aux utilisateurs de sauvegarder leurs recettes préférées.

Gestion des commentaires : possibilité de laisser des commentaires sur les recettes.

Structure de base du site :

Page d'accueil : affichage des recettes populaires, nouvelles recettes, etc.

Pages de catégorie : affichage des recettes par catégorie.

Page de détails de la recette : affichage complet d'une recette avec ses informations, instructions, commentaires, etc.

Pages utilisateur : profil utilisateur, recettes publiées par l'utilisateur, recettes favorites, etc.

Pages d'authentification : inscription, connexion, récupération de mot de passe.

Technologies à utiliser :

Symfony : choisir la version stable la plus récente.

Base de données : décider du système de gestion de base de données à utiliser (MySQL, PostgreSQL, etc.).

Interface utilisateur : définir le framework CSS (Bootstrap, Foundation, etc.) ou utiliser un framework front-end tel que Vue.js ou React.js.

Exigences de performance :

Le site doit être rapide et réactif pour offrir une bonne expérience utilisateur.

Optimisation des requêtes et de la gestion de la base de données.

Mise en cache des pages pour réduire le temps de chargement.

Sécurité :

Implémenter des mécanismes de sécurité tels que l'authentification, la protection contre les attaques CSRF, etc.

Gérer les autorisations d'accès aux fonctionnalités du blog (publication de recettes, commentaires, etc.).

Gérer la validation des données utilisateur pour prévenir les failles de sécurité.

Maintenance et évolutivité :

Prévoir la possibilité d'ajouter de nouvelles fonctionnalités à l'avenir (par exemple, un système de recherche avancée, des fonctionnalités sociales, etc.).

Planifier des sauvegardes régulières de la base de données et du code source.

## Description de la charte graphique

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 :

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.

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.

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

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.

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.

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

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.

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

## Prestations attendues et modalités de sélection des prestataires

# Gestion du projet

## Planning et suivi

## Environnement humain et technique

1. Environnement humain
2. Environnement technique

## Objectifs de qualité

# Spécifications fonctionnelles du projet

**UML :**

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 api-recette trois cas principaux d’utilisation sont présents (Admin et visiteur et client)

*Diagramme de cas généralisé pour un Administrateur*

*Diagramme de cas généralisé pour un visiteur*

*Diagramme de cas généralisé pour un client*

**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.

**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.

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).

SHEMA

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.

SHEMA

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.

# Spécifications techniques du projet

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 Wampp 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.

Les composants d’accès aux données

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.

# Réalisations

## Exemple 1

### Contexte

### Aperçus du code

### Arguments et commentaires

## Exemple 2

### Contexte

### Aperçus du code

### Arguments et commentaires

## Exemple 3

### Contexte

### Aperçus du code

### Arguments et commentaires

# Présentation d’un jeu d’essai

## Données en entrée

## Données attendues

## Données obtenues

# Veille sur les vulnérabilités de sécurité

## Besoin d’information

## Mots clés de recherche

## Liste des sites retournés

## Critères de sélection des sites

## Vulnérabilités trouvées

## Failles potentielles corrigées

# Description d’une situation de travail ayant nécessité une recherche

## Besoin d’information

## Mots clés de recherche

## Liste des sites retournés

## Critères de sélection des sites

## Solution et mise en œuvre

# Conclusion

# Annexes