Dinnerline



Kinan Jano – CIN4B

ETML Vennes

2024

Helder Manuel Costa Lopes

Table des matières

[1 Spécifications 3](#_Toc158215156)

[1.1 Titre 3](#_Toc158215157)

[1.2 Description 3](#_Toc158215158)

[1.3 Matériel et logiciels à disposition 3](#_Toc158215159)

[1.4 Prérequis 3](#_Toc158215160)

[1.5 Cahier des charges 3](#_Toc158215161)

[1.5.1 Objectifs et portée du projet (objectifs SMART) 3](#_Toc158215162)

[1.5.2 Caractéristiques des utilisateurs et impacts 3](#_Toc158215163)

[1.5.3 Fonctionnalités requises (du point de vue de l’utilisateur) 4](#_Toc158215164)

[1.5.4 Contraintes 4](#_Toc158215165)

[1.5.5 Si le temps le permet … 4](#_Toc158215166)

[1.5.6 Méthodes de validation des solutions 4](#_Toc158215167)

[1.6 Les points suivants seront évalués 4](#_Toc158215168)

[1.7 Validation et conditions de réussite 4](#_Toc158215169)

[2 Planification Initiale 5](#_Toc158215170)

[3 Analyse 5](#_Toc158215171)

[3.1 Opportunités 5](#_Toc158215172)

[3.2 Document d’analyse et conception 5](#_Toc158215173)

[3.3 Conception des tests 6](#_Toc158215174)

[3.4 Planification détaillée 6](#_Toc158215175)

[4 Réalisation 6](#_Toc158215176)

[4.1 Dossier de Réalisation 6](#_Toc158215177)

[4.2 Modifications 6](#_Toc158215178)

[5 Tests 6](#_Toc158215179)

[5.1 Dossier des tests 6](#_Toc158215180)

[6 Conclusion 6](#_Toc158215181)

[6.1 Bilan des fonctionnalités demandées 6](#_Toc158215182)

[6.2 Bilan de la planification 7](#_Toc158215183)

[6.3 Bilan personnel 7](#_Toc158215184)

[7 Divers 7](#_Toc158215185)

[7.1 Journal de travail 7](#_Toc158215186)

[7.2 Bibliographie 7](#_Toc158215187)

[7.3 Webographie 7](#_Toc158215188)

[8 Annexes 7](#_Toc158215189)

# Spécifications

## Titre

Système de commande pour la cafétéria ABC

Nom de code : « DinerLine »

## Description

La cafétéria ABC, située au sein du Centre de Formation Professionnel de Vennes (CFPV), requiert un système numérique efficace pour moderniser ses opérations. Le projet vise à développer un site web qui facilite la commande de produits de boulangerie et de repas de midi pour les étudiants, les enseignants, et autres. Le but principal est de créer une plateforme intuitive et sécurisée qui permet :

* Aux clients de passer des commandes en ligne facilement.
* Au personnel de la cafétéria de gérer les stocks, les commandes, et les menus de manière efficace.
* D'offrir une expérience utilisateur agréable et accessible sur différents appareils (Responsive).

## Matériel et logiciels à disposition

* PC
* Node.js & npm
* Visual Studio Code
* Postman

## Prérequis

Connaissances de bases dans Node.JS et le Framework React.js

* Connaissances de base dans Node.js
* Connaissances de base dans React.js
* Connaissances de base de modélisation de données
* Connaissances de base de conception d’interface utilisateur

## Cahier des charges

### Caractéristiques des utilisateurs et impacts

Enseignants et élèves à Vennes et le personnel du restaurant COFOP

### Fonctionnalités requises (du point de vue de l’utilisateur)

* **Commande en Ligne** : Une interface conviviale pour choisir et commander des produits (avec la possibilité de payer en ligne, voir ce qu’offre twint).
* **Calendrier des Menus** : Un calendrier interactif présentant les menus quotidiens, permettant aux utilisateurs de planifier leurs commandes.
* **Analyse et Rapports** : Des fonctionnalités d'analyse pour suivre les ventes, les préférences des clients et optimiser le service.
* **Feedback Client** : Un système pour recueillir les avis et suggestions des clients, améliorant ainsi continuellement le service.

### Contraintes

Sécurité : SSL

Sécurité, backups, disponibilité, système utilisé, interfaces avec autres logiciels, etc.

### Si le temps le permet …

* 1. **Gestion des Stocks et Commandes** : Outils pour le suivi des stocks, la gestion des commandes entrantes et la préparation des repas.

### Méthodes de validation des solutions

Comment les tests vont être entrepris, quels tests doivent être entrepris, etc.…

## Les points suivants seront évalués

* Le rapport
* Les planifications (initiale et détaillée)
* Le journal de travail
* Le code et les commentaires
* Les documentations de mise en œuvre et d’utilisation

## Validation et conditions de réussite

* Compréhension du travail
* Possibilité de transmettre le travail à une personne extérieure pour le terminer, le corriger ou le compléter
* Etat de fonctionnement du produit livré

# Planification Initiale

|  |  |  |
| --- | --- | --- |
| 1 | 22.01 – 26.01 | Préparation de l’environnement de travail |
| 2 | 29.01 – 02.02 | Planification + recherche |
| 3 | 05.02 – 09.02 | Back-End - Conception + DB |
| 4 | 12.02 – 16.02 | *Vacances* |
| 5 | 19.02 – 23.02 | Back-End – API |
| 6 | 26.02 – 01.03 | Front-End – Conception |
| 7 | 04.03 – 08.03 | Front-End |
| 8 | 11.03 – 15.03 | Préparer une présentation + Finaliser la documentation |

# Analyse

## Opportunités

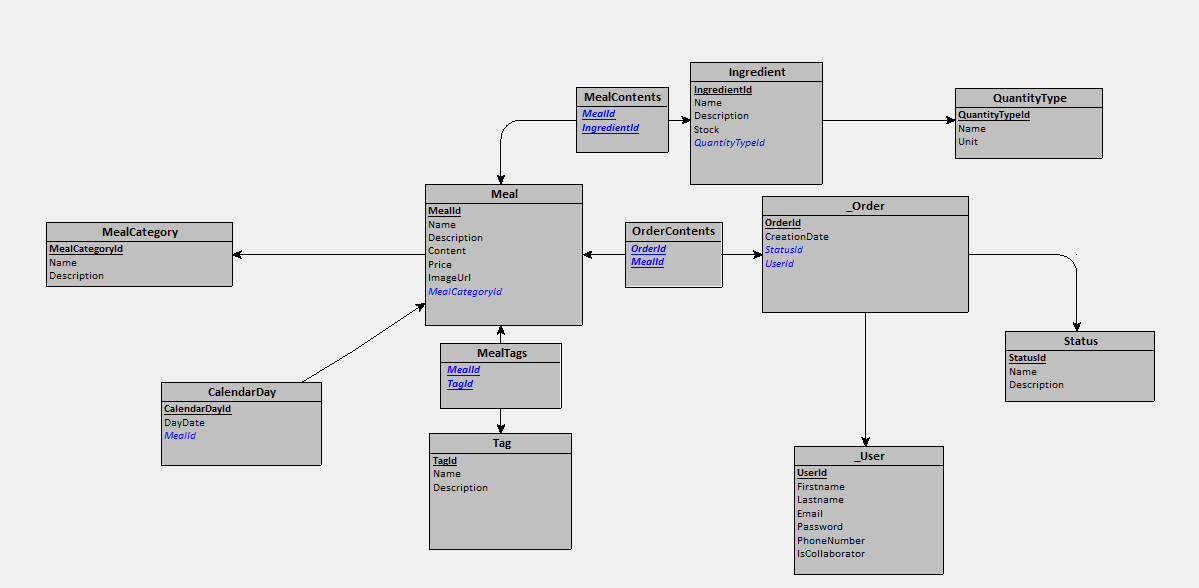
* Approfondir mes connaissances en React.js et le développement d’interfaces utilisateur.
* Approfondir mes connaissances en modélisation des données.
* Approfondir mes connaissances dans l’architecture API RESTful.
* Explorer les bases de données Non-SQL (MongoDB)

## Document d’analyse et conception

### Modelisation

Au début de la modélisation, j’ai imaginé une base de données relationnelle MySQL. Ensuite, M. Lopes m’a proposé de le faire en Non-SQL avec MongoDB.

J’ai décidé de finalement faire en MongoDB pour explorer les bases de données Non-SQL. Je n’arrivais pas à commencer à modéliser les données car je devais apprendre le nouveau concept qui est les collections (non-SQL) qui sont l’équivalent des tables en bases de données SQL.



Expliquer schema SQL to NonSQL

### Base de données

### API

Après avoir fini de monter ma base de données MongoDB sur Mongo Atlas, j’ai doit développer un programme qui s’agit d’un lien entre l’utilisateur et ma base de données.

Selon les conventions des API REST, la communication entre l’utilisateur et l’API est fait avec des requêtes http, avec des méthodes : GET, POST, PUT et DELETE, chaque méthode signifie une des lettres de la désignation CRUD (Create, Read, Update, Delete).

#### POST (Créer)

L’API reçoit des données sous forme d’un objet JSON dans le body de la requête, cet objet doit être ajouté dans la base de données. Si l’ajout dans la base de données a été effectué correctement, l’API retourne l’objet qui a été ajouté, cherché de la base de données avec l’identifiant qui lui a été assigné automatiquement (dans mon cas avec MongoDB).  
En cas d’échec, l’API retourne un code d’erreur 500.

#### GET (Lire)

L’API n’applique aucune modification dans les données, mais il retourne l’objet demandé par l’utilisateur, cela souvent identifié par un identifiant qui est envoyé dans l’url de la requête.

#### PUT

La fonctionnalité de la méthode PUT est souvent confondue avec la fonctionnalité de la méthode UPDATE.

La fonctionnalité de PUT ne s’agit pas seulement d’une modification de données, mais s’agit aussi d’une manière d’insérer les données dans la DB. Cette fonctionnalité est utilisée dans les rares cas où on veut ajouter un objet en spécifiant la valeur de son identifiant.

#### DELETE (Supprimer)

L’API reçoit des données sous forme d’un objet JSON dans le body de la requête, cet objet doit être ajouté dans la base de données. Si l’ajout dans la base de données a été effectué correctement, l’API retourne l’objet qui a été ajouté, cherché de la base de données avec l’identifiant qui lui a été assigné automatiquement (dans mon cas avec MongoDB).  
En cas d’échec, l’API retourne un code d’erreur 500.

### Interface utilistateur

Ce paragraphe décrit le fonctionnement de manière détaillée.

Autant que possible de manière graphique, imagée, tableaux, etc.

Tous les cas particuliers devraient y être spécifiés…

Il s’agit d’y présenter les fonctionnalités à développer :

Découpage en étapes, en modules, en fonctionnalités, etc.

Formulaires, interfaces graphiques, pages web, etc.

Schémas de navigation, schémas événementiels, structogramme, pseudocode, etc.

Si le projet inclut une base de données :

Dictionnaire des données

Modèle conceptuel des données, modèles logique des données.

## Conception des tests

Ce paragraphe permet de spécifier la stratégie de test qui sera menée au point 5.1

Qui, quand, avec quelles données, dans quel ordre, etc.

## Planification détaillée

A ce stade, après l’analyse complète du projet, un planning détaillé et complet (avec tâches, sous-tâches, dépendances, durée, …) peut être finalisé.

Le planning détaillé doit s’inscrire dans le planning initial. Il faut que l’on puisse situer cette planification détaillée par rapport à la planification initiale.

# Réalisation

## Dossier de Réalisation

Cette partie permet de reproduire ou reprendre le projet par un tiers.

Pour chaque étape, il faut décrire sa mise en œuvre. Typiquement :

Versions des outils logiciels utilisés (OS, applications, pilotes, librairies, etc.)

Configurations spéciales des outils (Equipements, PC, machines, outillage, etc.)

Code source commenté des éléments logiciels développés.

Modèle physique d’une base de données.

Arborescences des documents produits.

Il faut décrire le parcours de réalisation et justifier les choix.

## Modifications

Historique des modifications demandées (ou nécessaires) aux spécifications détaillées.

Date, raison, description, etc.

# Tests

## Dossier des tests

On dresse le bilan des tests effectués (qui, quand, avec quelles données…) sous forme de procédure. Lorsque cela est possible, fournir un tableau des tests effectués avec les résultats obtenus et les actions à entreprendre en conséquence (et une estimation de leur durée).

Si des tests prévus dans la stratégie n'ont pas pu être effectués :

raison, décisions, etc.

Liste des bugs répertoriés avec la date de découverte et leur état:

Corrigé, date de correction, corrigé par, etc.

# Conclusion

## Bilan des fonctionnalités demandées

Il s’agit de reprendre point par point les fonctionnalités décrites dans les spécifications de départ et de définir si elles sont atteintes ou pas, et pourquoi.

Si ce n’est pas le cas, estimer en « % » ou en « temps supplémentaire » le travail qu’il reste à accomplir pour terminer le tout.

## Bilan de la planification

Distinguer et expliquer les tâches qui ont généré des retards ou de l'avance dans la gestion du projet. Indiquer les différence entre les planifications initiales et détaillées avec le journal de travail.

## Bilan personnel

Si c’était à refaire:

Qu’est-ce qu’il faudrait garder ? Les plus et les moins ?

Qu’est-ce qu’il faudrait gérer, réaliser ou traiter différemment ?

Qu’est que ce projet m’a appris ?

Suite à donner, améliorations souhaitables, …

Remerciements, signature, etc.

# Divers

## Journal de travail

Date, activité (description qui permet de reproduire le cheminement du projet), durée, liens et références sur des documents externes. Lorsqu’une activité de recherches a été entreprise, il convient d’énumérer ce qui a été trouvé, avec les références.

## Bibliographie

Références des livres, revues et publications utilisés durant le projet.

## Webographie

Références des sites Internet consultés durant le projet.

# Annexes

Listing du code source (partiel ou, plus rarement complet)

Guide(s) d’utilisation et/ou guide de l’administrateur

Etat ou « dump » de la configuration des équipements (routeur, switch, robot, etc.).

Extraits de catalogue, documentation de fabricant, etc.