Tech’n Logic pour Infinite Measures

Document de conception

Livrable

Yassine LARAIEDH, William Alexander MBOLLO MBASSY, Théotime POICHOTTE, Benjamin POMBET, Christophe SAURY, Pierre TOMEI

Table des matières

Contexte

Notre entreprise Tech'N'Logic est une start-up spécialisée dans le domaine du numérique, notamment dans la conception de capteurs sensoriels tels que les capteurs cardiaques ainsi que le développement web. Composée de 6 ingénieurs aux profils et compétences différentes mais complémentaires, Infinite Measures, notre nouveau client, nous a confié une mission qui a pour but de créer un produit alliant un boitier électronique et un site web traitant les données récoltées par le boitier en question. Pour réaliser cette mission, il faut maîtriser les quatre domaines suivants : l'électronique, le signal, l'informatique et la télécom. Pour la partie informatique, il faut réaliser une interface web pour les différents employés des chantiers dont le but est d'assurer les bonnes conditions de travail des ouvriers. C'est également un outil utile pour les chefs de chantiers afin assurer la sécurité de tous.

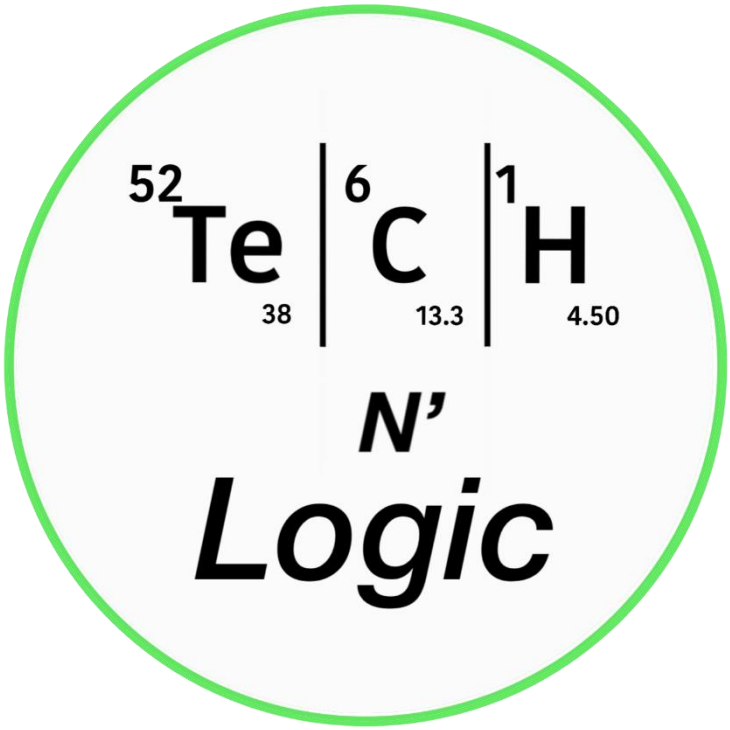

Figure 1 - logo de notre entreprise

Conception de la base de données

Avant de commencer la réalisation de notre site web pour notre client Infinite Measures, nous avons réfléchi à la conception de notre base de données afin de savoir de quelle manière nous allons représenter nos données et les liens qui existent entre elles. Nous avons dans la figure ci-dessous notre modèle de base de données final.

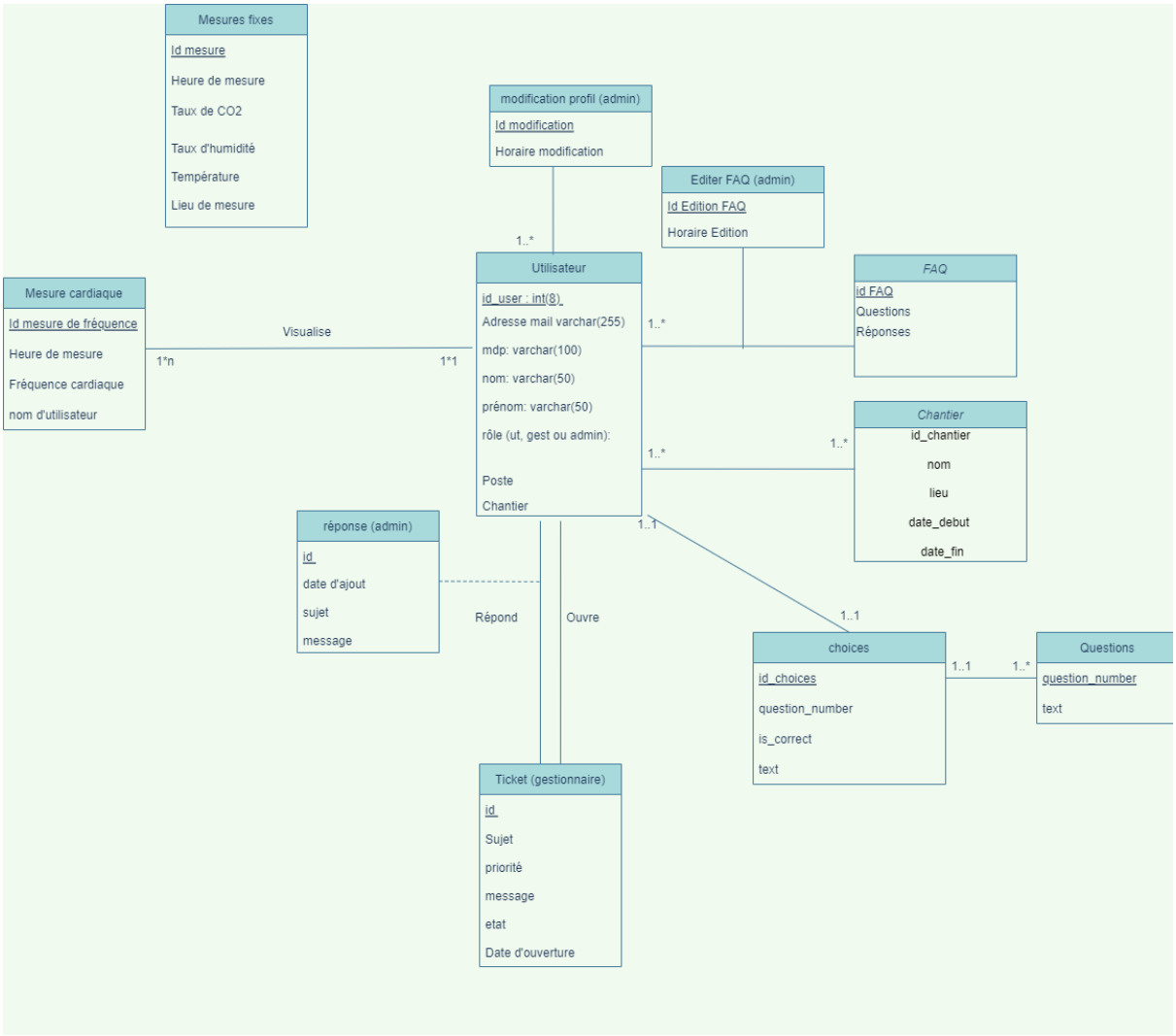

Figure 2 - modèle de notre base de données

Par rapport à notre modèle initial, notre nouveau modèle est resté le même avec quelques modifications notamment l'ajout de l'attribut chantier dans la base de données utilisateur et donc de la table chantier car nous avons décidé d'implémenter cette fonctionnalité plus tardivement. Nous avons également indiqué les bases de données qui concernent les

administrateurs. Par exemple, un gestionnaire ou un utilisateur ne peuvent pas répondre aux tickets, ou encore ne peuvent pas éditer une FAQ.

Décrivons maintenant notre base de données finale par un modèle relationnel.

Pour l'utilisateur nous avons :

Utilisateur (id_user, Nom, Prénom, adresse mail, MDP, Rôle)

Cette table est commune pour toutes les autres tables étant donné que les fonctionnalités changent en fonction du rôle de l'utilisateur. Nous avons défini les trois rôles par des entiers :

0 = admin

1 = gestionnaire

2 = utilisateur

Ensuite nous avons la fonctionnalité ticket définie par deux tables, la première concerne les tickets, la deuxième concerne les réponses, gérer par tous ceux qui ont le rôle d'administrateur. Ci-dessous leur modèle relationnel.

Ticket (id, sujet, priorité, message, état, date ouverture, adresse mail)

Réponse (id, date d'ajout message, sujet, adresse mail, id ticket)

Les gestionnaires ouvrent les tickets, et les administrateurs gèrent les réponses à ses tickets. Pour ce qui concerne les tickets, nous avons mis un ordre de priorité, défini également par des valeurs numériques codés sur un bite.

0 = élevé

1 = moyenne

2 =faible

Passons maintenant à la table de FAQ. Nous avons fait en sorte que l'administrateur puisse ajouter, modifier ou supprimer des questions, et ajouter également le moment des modifications de n'importe quelle question. Cette fonctionnalité nécessite quatre base de données.

FAQ (id FAQ, Questions, Réponses)

Editer FAQ (id Edition FAQ, Horaire Edition, Numéro de l'Editeur)

Question (id Question, contenu de la question)

Réponse (id Réponse, contenu de la réponse, id Question)

Concernant la partie qcm, elle est dédiée pour les utilisateurs (ouvriers). Nous avons réalisé trois tables, QCM, questions et classement. Ces trois tables sont liées entre elles étant donné que le QCM contient des questions et des réponses, et à partir des réponses on détermine le classement de chaque utilisateur. La partie QCM concerne également les gestionnaires et les administrateurs, où leur rôle est d'ajouter, de modifier ou encore de supprimer des questions ainsi que les réponses. Voici leur modèle relationnel :

QCM {id_qcm, Nombre_Réponses_Réussi, Nombre_Réponses_Raté, id_questions, id_user }

Questions {id_question, Question, Bonnes_Réponses, Mauvaises_Réponses}

Classement {id_classement, Placement, Score, Nombre_Réponses_Réussi, Nombre_Réponses_Raté, id_user}

Nous avons également deux autres tables, une pour les mesures fixes, qui permet de présenter aux différents employés les conditions de travail, et une pour les mesures cardiaques, bien évidemment pour les utilisateurs (ouvriers). Ci-dessous leur modèle relationnel.

Mesure_fixe (Id_mesure, temps/heure, taux de CO2, taux humidité, température, lieu de mesure,)

Mesure cardiaque(Id mesure fréquence, temps/heure, fréquence cardiaque, id_user)

Enfin, nous avons rajouté une base de donnée qui concerne les chantiers. Son rôle est d'attribuer à chaque utilisateur le nom, le lieu, la date du début et la date de la fin de chaque chantier sur lequel il travaille. Voici son modèle relationnel :

Chantier (id_chantier, nom, lieux, date_debut, date_fin)

Architecture du site

Organisation du code

Durant ce projet, notre équipe a travaillé par fonctionnalité et par priorité. Le développement intégral de chaque fonctionnalité, par exemple les tickets ou la FAQ, a été attribué à un membre.

Ensuite, pour l'organisation du code, nous avons travaillé en MVC. En effet, afin de mieux organiser notre code, de séparer l'aspect visuel de l'aspect fonctionnel, il nous a été plus rigoureux d'utiliser cette méthode.

Le modèle est la partie qui gère les données de notre site. Elle récupère les informations dans les bases de données, les organisent puis les assemblent pour qu'elles puissent être traités par le contrôleur.

La vue correspond à l'affichage des pages du site. Elle contient majoritaire les pages HTML/CSS mais aussi du PHP. A l'intérieur de chaque dossier « vue » nous avons également créé un dossier css, étant donné que le css est une partie essentielle de l'affichage du site.

Le contrôleur gère la logique du code qui prend des décisions. Elle contient exclusivement les pages PHP.

Nous avons ci-dessous un schéma résumant l'architecture MVC.

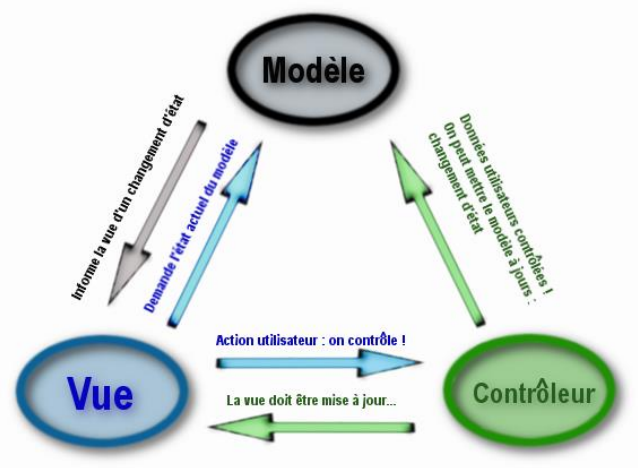

Figure 3 - Modélisation de l'architecture MVC

Concernant l'organisation des fichiers nous avons travaillé par « brique », de la même manière que nous l'avons développé. Autrement dit, chaque partie a son propre MVC et elle est indépendante des autres fonctionnalités.

Ci-dessous une capture d'écran qui illustre l'organisation de notre code. On peut voir que l'on a trois MVC séparés, la première pour les pages de menu, la deuxième pour la page de connexion et la troisième pour les pages de tickets

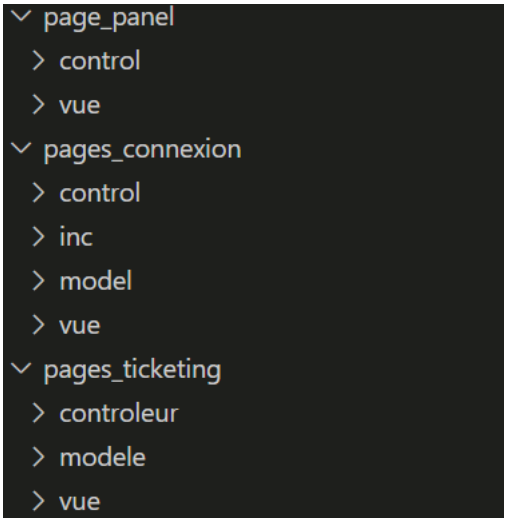

Figure 4 - Capture d'écran de notre architecture MVC par brique

Charte graphique

Concernant la charte graphique, nous avons décidé d'utiliser des couleurs claires et basiques, étant donné que notre site reste tout de même un site pour les professionnels.

Nos couleurs principales sont les suivantes :

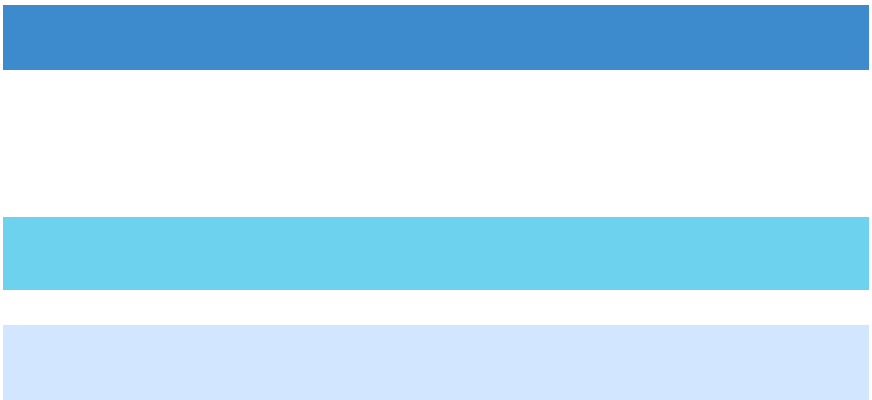

Figure 5 - charte graphique de notre site web

Points à améliorer

Ce projet nous a permis de montrer à notre client de quoi notre équipe est capable. Nous avons réussi d'avoir toutes les fonctionnalités efficaces, tout en respectant une charte graphique que nous avons déterminée tout au début de notre projet. Cependant nous pouvons améliorer certains éléments pour que notre site soit meilleur.

Tout d'abord nous pourrion améliorer la sécurité du site en ajoutant le protocole xss afin d'empêcher l'injection d'un utilisateur venu de l'extérieur.

On peut également mieux faire au niveau du design, rendre les pages plus dynamiques, moins statiques, cela est dû essentiellement à un défaut de temps.