Saint-Jean
Matéo
CNED

Compte rendu du stage à INGEFOX

(Deuxième année)

Présentation de l'entreprise :

Que fait Ingefox ?

Ingefox est une entreprise de création de logiciel sur mesure.

Ingefox accompagne les entreprises dans la création et le développement de logiciels métiers sur mesure, en fonction de leurs besoins, de leurs attentes et de leurs usages.

Chaque entreprise a des besoins spécifiques et précis, c'est pourquoi ils s'engagent à fournir un haut niveau de qualité dans le déploiement de vos solutions.

Pourquoi faire appel à Ingefox ?

Tout commence par une question, un problème vaguement posé et surtout un constat que si on avait le bon outil, on gagnerait en efficacité, en rapidité, en confort ou encore en satisfaction client…

Ingefox vous accompagne tout au long de cette réflexion. Ils mettent en place ensemble la solution qui répondra à cette question ou résoudra cette problématique.

Qu'est-ce que l'éco-conception ?

L'éco-conception numérique vise à maitriser et implémenter les mécanismes afin de diminuer la consommation énergétique dès la création d'un système numérique (sites web, applications, logiciels, objets connectés, etc).

L'entreprise qui est engagée dans une démarche d'éco-conception numérique va chercher à **optimiser** son service **en prenant en compte l'environnement** sur tout son cycle de vie. L'éco-conception est une méthodologie **standardisée** au niveau **national**.

Les missions :

Ayant déjà effectué un stage dans cette entreprise en première année et en ayant réalisé par la suite des prestations pour eux, je connaissais déjà l'environnement.

Etant plus expérimenté qu'en première année les tâches ont été beaucoup plus complexe.

Avec toujours la même idée qu'au départ me laisser me débrouiller afin d'apprendre à chercher par moi-même. Voici donc les différentes missions :

Première Mission « SonarQube »

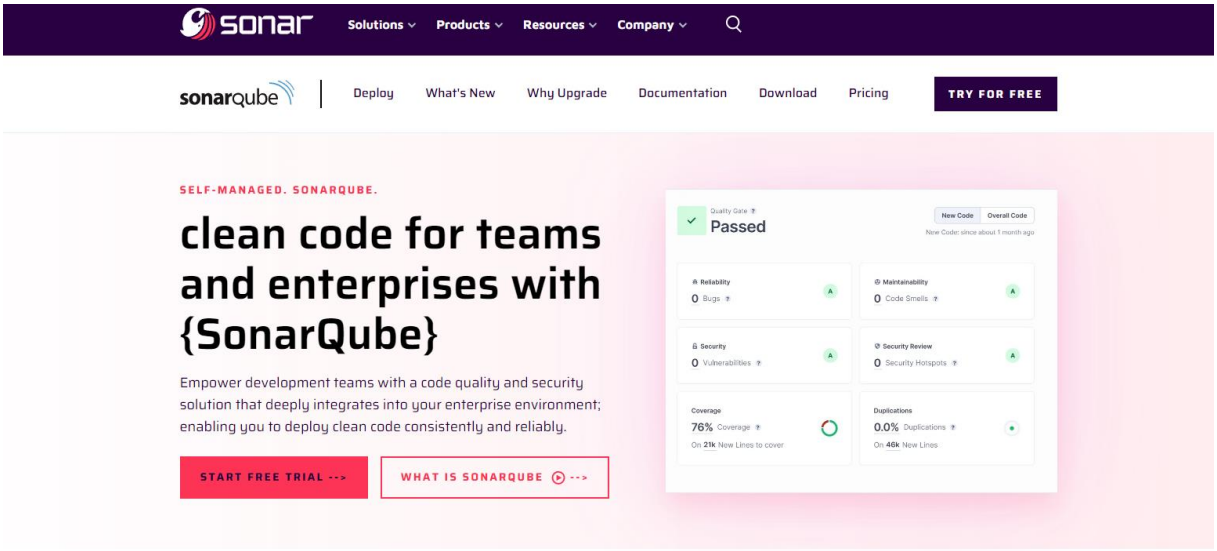
J'ai eu l'occasion de travailler sur le logiciel SonarQube via ma formation au CNED.

J’en ai parlé à mon tuteur de stage et cela l’a intéressé ne connaissant pas le logiciel il m’a confié comme mission d’intégrer SonarQube à leurs activités et comment cela pourrait être bénéfique pour eux.

J’ai donc repris mes cours et lu la documentation de SonarQube pour comprendre le logiciel plus en profondeur.

Présentation du logiciel :

SonarQube est un outil open source pour l'inspection continue de la qualité du code source, il identifie les problèmes, les bugs et les vulnérabilités de sécurité, et fournit des rapports détaillés pour améliorer la qualité du code.



SonarQube est également compatible avec une extension pour les IDE nommée « Sonarlint » permettant de corriger en temps réel les erreurs de conventions, les duplications… Tout ce que peux faire le logiciel SonarQube mais sur la page courante.

Réalisation de la tâche :

J’ai donc testé SonarQube sur un projet (réalisation d’un scanner), puis j’ai installé Sonarlint sur mon IDE (phpstorm) pour leurs montrer.

Par la suite d’une discussion mon tuteur m’a demandé de réaliser si c’était possible, l’hébergement d’un serveur sonarqube sur une machine locale pour que tous les employés y aient accès, puis d’ajouter des « règles » de code personnalisées propre à l’entreprise.

Je me suis donc documenté sur le sujet et vu que SonarQube est open source cela est possible d’ajouter mais également de supprimer des règles ou des conventions de code.

C’est assez complexe mais si je devais résumer, il faut installer un plugin pour réaliser des « customs rules » dans le langage PHP, puis ajouter une classe en Java (langage de

programmation de sonarqube), et cette classe comprendra notre règle. Il faut alors la relier aux autres classes java qui centralisent les règles, puis réaliser un test unitaire obligatoire, donc faire une classe de test etc…

Il faut ensuite compiler le plugin en .jar et l'ajouter dans les plugins sonarqube (sonarqube>extensions>plugin). Puis redémarrer le serveur via le gestionnaire de tâches.

Ensuite sur sonarqube il faut créer un profil personnalisé et sélectionner les règles que l'on souhaite utiliser dans ce profil (ducoup la suppression se fait comme ça) et notre nouvelle règle apparait.

J'ai ensuite lié le compte sonarlint au serveur sonarqube et nos nouvelles règles peuvent apparaître en direct ! Magie.

J'ai héberger un serveur sonarqube sur un mac mini de l'entreprise afin qu'il soit disponible à tous.

Lien du repo github avec les nouvelles règles : https://github.com/Yuvem10/sonarqube

Documentation de sonarqube : https://docs.sonarsource.com/sonarqube/latest/

Code d'une règle permettant que les variables soit écrites en demarrant par une minuscule :

```java
/**
 * Example of implementation of a check by extending {@link PHPVisitorCheck}.
 * PHPVisitorCheck provides methods to visit nodes of the Abstract Syntax Tree
 * that represents the source code.
 * <p>
 * Those methods can be overridden to process information
 * related to node and issue can be created via the context that can be
 * accessed through {@link PHPVisitorCheck#context()}.
 */
@Rule(
  key = VariableNameCheck.KEY,
  priority = Priority.MAJOR,
  name = "Variable must be start by lower case",
  tags = {"convention"})
public class VariableNameCheck extends PHPVisitorCheck {

  public static final String KEY = "S2";

  @Override
  public void visitVariableIdentifier(VariableIdentifierTree tree) {
    String name = tree.text();
    name = name.replace("$", "");

    String firstLetter = Character.toString(name.charAt(0));
    String toUpperCase = firstLetter.toUpperCase();
    if (firstLetter == toUpperCase) {
      context().newIssue(this, tree, "The variable name must be start by a lower case");
    }
    super.visitVariableIdentifier(tree);
  }
}
```

Le code est assez simple car dans le plugin des méthodes sont faites pour retrouver une variable, ou un nom de classe etc… C'est le cas ici de visitVariableIdentifier().

La classe de test correspondante :

```java
import java.io.File;
import org.junit.jupiter.api.Test;
import org.sonar.plugins.php.api.tests.PHPCheckVerifier;

/**
 * Test class to test the check implementation.
 */
class VariableNameCheckTest {

  @Test
  void test() {
    PHPCheckVerifier.verify(new VariableNameCheck(), new File("src/test/resources/checks/VariableNameCheck.php"));
  }

}
```

Le fichier que vérifie la classe de test :

```php
<?php

$variable = "";//OK
$Variable = "";//NonCompliant
```

J'ai ensuite réalisé une documentation afin d'expliquer le procédé exact (ici c'est un résumé) pour que les developpeurs de la boite puisse ajouter et supprimer des règles si ils en ont besoin.

Deuxième Mission « OpenAI »

Présentation :

OpenAI est une entreprise de recherche spécialisée dans l'intelligence artificielle qui développe des technologies et des outils de pointe dans le domaine de l'IA. Fondée en 2015, OpenAI s'efforce de créer une intelligence artificielle généraliste avancée et de la rendre accessible de manière éthique et sécurisée.

L'API OpenAI est l'un de leurs produits phares. Il s'agit d'une interface de programmation d'applications (API) qui permet aux développeurs d'accéder aux modèles d'IA de pointe développés par OpenAI. Avec cette API, les développeurs peuvent intégrer des fonctionnalités d'IA avancées telles que la génération de texte, la classification de texte, la traduction, et bien plus encore, dans leurs propres applications et services. L'API OpenAI offre aux développeurs un accès à des capacités d'IA puissantes tout en simplifiant le processus d'intégration et de déploiement.

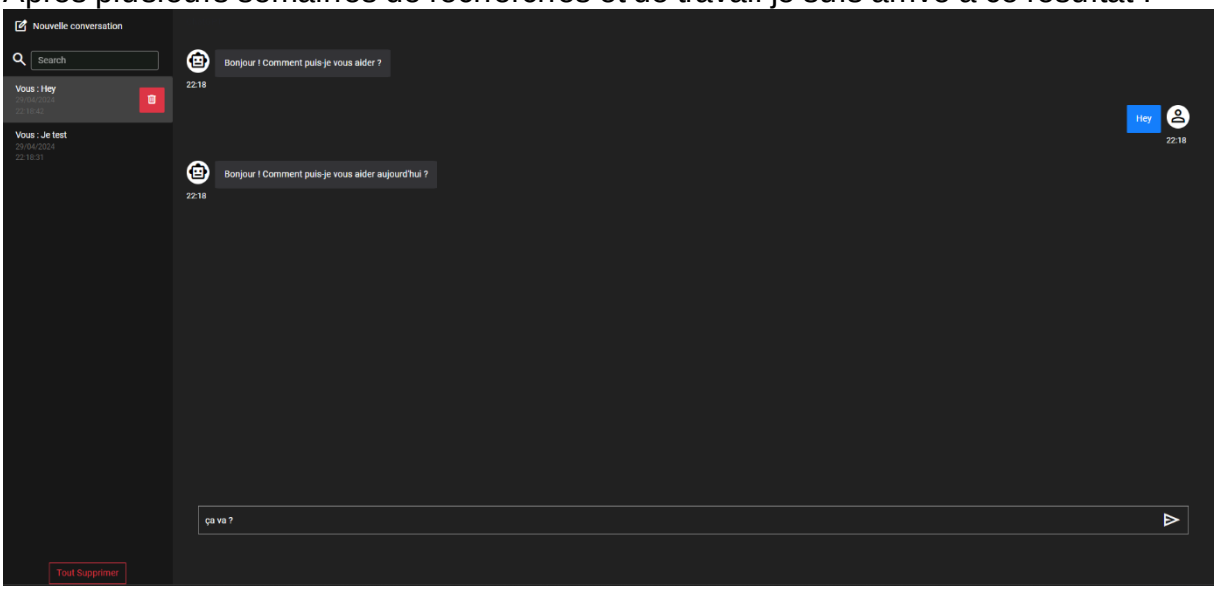
Réalisation de la tâche :

Un des clients de l'entreprise venait de faire la demande d'une option intégrante chatGPT (voir plus bas).

Afin de me familiariser avec l'API mon tuteur de stage m'a proposé de créer une interface de dialogue avec chatGPT de permettre la récupération de discussions etc…

J'ai réalisé cela dans le langage PHP, via le framework codeigniter4 et bien sur HTML, CSS et javascript pour la partie front-end.

Après plusieurs semaines de recherches et de travail je suis arrivé à ce résultat :



Tout est fonctionnel c'est quasiment l'interface de base à l'identique avec des ajouts personnels.

Alors comment tout cela fonctionne ?

Premièrement j'ai utilisé l'architecture MVC, un modèle permettant de jouer avec les différents endpoints de l'api, une vue permettant d'afficher le contenu et d'envoyer les données de l'utilisateur via des requêtes ajax, ainsi qu'un controlleur faisant le lien entre la vue et le modèle.

Voici un exemple de méthode pour chaque fichier :

Modèle :

```php
/**
 * Create a chat completion
 * @return false|string
 */
0 references | 0 overrides
public function createChatCompletion()
{
    $ret = ['status' => SC_INTERNAL_SERVER_ERROR, 'reason' => 'Une erreur interne est survenue', 'data' => []];

    try {
        /** @var string $question */
        $question   = $this->request->getPostGet('question');
        $sessionID  = $this->request->getPostGet('sessionID');
        $sessionID  = !empty($sessionID) ? $sessionID : null;

        if (!empty($question)) {

            $M_ChatGPT = new M_ChatGPT();

            $data = [
                [
                    'role'      => 'user',
                    'content'   => $question
                ]
            ];

            $response = $M_ChatGPT->createChatCompletion($data, false, $sessionID);

            $ret = ['status' => $response['status'], 'reason' => $response['reason'], 'data' => $response['data']];
        }
        else{
            $ret['status'] = SC_BAD_REQUEST;
            $ret['reason'] = 'La question est vide';
        }
    }
    catch (Exception $e) {
        log_message('error', __CLASS__ . '::' . __FUNCTION__ . '() : ' . $e);
    }

    $this->response->setStatusCode($ret['status'], $ret['reason']);
    $this->response->setContentType('application/json');

    return json_encode($ret);

}
```

Controlleur :

```php
/**
 * Create a new empty chat completion session and return its ID
 * @return array
 */
0 references | 0 overrides
public function createEmptyChatCompletionSession() : array
{
    $ret = ['status' => SC_INTERNAL_SERVER_ERROR, 'reason' => 'Une erreur interne est survenue', 'data' => null];

    try {
        helper('text');
        $sessionID = random_string('alnum', 32);

        // Check if the folder exists, if not, create it
        if (!is_dir(self::CHAT_GPT_SESSIONS_FOLDER))
        {
            mkdir(self::CHAT_GPT_SESSIONS_FOLDER, 0777, true);
        }

        $sessionFilePath = self::CHAT_GPT_SESSIONS_FOLDER . $sessionID . '.json';

        $data = self::CHAT_GPT_BASE;

        file_put_contents($sessionFilePath, json_encode($data, JSON_PRETTY_PRINT));

        $ret['status']  = SC_SUCCESS;
        $ret['reason']  = 'La session de chat à été créée';
        $ret['data']    = $sessionID;
    }
    catch (Exception $e) {
        log_message('error', __CLASS__ . '::' . __FUNCTION__ . '() : ' . $e);
    }

    return $ret;
}
```

Vue :

```
/**
 * Ask Chat GPT a question and display the answer
 */
function createChatCompletion()
{
    // Retrieve the question from the input
    let message = document.getElementById("inputsend").value;

    // Display the question
    createQuestion(message);

    // Display the placeholder
    displayPlaceholder();

    // Clear the input
    document.getElementById("inputsend").value = "";

    disableLoadingAnimation = true;

    // Send the Ajax request to the server (POST : /ChatGPT/createCompletion)
    $.ajax({
        url: '<?= base_url('createCompletion') ?>',
        type: 'POST',
        dataType: 'json',
        data: {
            question: message,
            sessionID: sessionID
        },
        success: function (response) {
            if (response['status'] === <?= SC_SUCCESS ?>) {
                createAnswer(response['data']['choices'][0]['message']['content'], false);
                newSession = false;
                getChatCompletionSessions();
            }else{
                console.log("IL Y A UNE ERREUR ICI");
            }
        },
        error: function (response) {
            let json    = response['responseJSON'];
            let reason  = json['reason'] ?? 'Une erreur est survenue';
            createAnswer(reason, true);
        }
    });

    disableLoadingAnimation = false;
}
```

Je n’ai pas montré la classe de chatcompletion pour le modèle car trop longue.

Pour plus d’information, le repo du projet ici : https://github.com/Yuvem10/ChatGPT

Troisième Mission « Intégration d'un module chatGPT dans un projet »

La demande :

Pour en revenir à cette fameuse demande, maintenant que je maîtrise l'api openai je vais directement pouvoir travailler sur un projet concret.

Je vais travailler sur un logiciel de gestion d'une agence d'intérim dans le médical. Ce logiciel au vu de son activité comporte une quantité de donnée très très importante (j'ai vu une datatable avec 5000 pages…).

Il y a donc énormément de champs à remplir par exemple pour enregistrer un nouvel intérim etc.… et c'est redondant. Le propre d'un développeur c'est aussi de trouver des solutions. Alors après un entretient avec mon tuteur et le directeur général, ils ont convenu de réaliser cette tâche grâce à chatGPT.

La réalisation :

Pour faire simple (car ça ne l'est pas) il y aurait un modal (une petite page s'ouvrant à l'aide d'un bouton) en bas à droite de toutes les pages du projet, permettant de réaliser de nombreuses actions. Et aussi une page de configuration des requêtes pour chatGPT.

Donc les requêtes vont pouvoir être chargées dans le modal et être exécutées sur la page courante si possible. Par exemple :

Une requête : réalise un message pour {#idname} et dis-lui que je retiens sa candidature.

Cette reqûete s'envoi à chatGPT avec le name qui CORRESPOND au champ {#idname} de la page courante et cela s'active via un bouton sur le modal.

Donc si le champ contient « Hervé » chatGPT va rédiger un message pour Hervé avec la demande, et ce message s'affiche dans le modal, donc le travail de rédaction de mail est réduit.

MAIS ce n'est pas tout.

Il y a également la fonction de remplissage automatique de champs ce que je parlais au début. En fait c'est le chemin inverse.

Une autre requête : Récupère le nom et le prénom du mail que je vais t'envoyer. Et repond moi au format json. Avec comme clé pour le nom {#inputname} et pour le prénom {#inputfirstname}.

Donc la j'envoi un mail à chatGPT et comme il fonctionne avec des threads (discussions enregistrées) il possède ma requête en mémoire donc si le mail est signé : Hervé Dumont. Il va me répondre :

« {#inputname} : « Dumont »,

« {#inputfirstname} : « Hervé »

(Imaginons que c'est un json correct)

Et là dans mon modal j'ai un algorithme spécial que j'ai codé pour pouvoir lire ce json et remplir les champs correspondant dans la page courante.

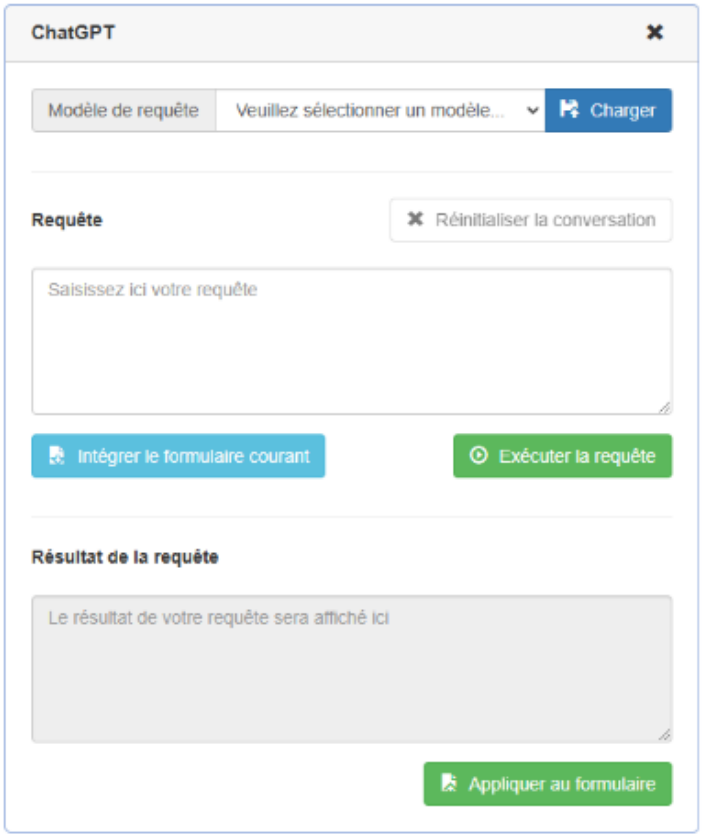
Voila les deux axes principaux de mon travail.

J'ai dû exporter le projet en local via une clé usb, l'ouvrir en serveur local avec wamp, migrer la base de données en local, ainsi que redéfinir les paramêtre nécessaires au fonctionnement.

Le projet est en php sous codeigniter3 cette fois donc il fallait réapprendre les bases du Framework.

Toujours en suivant l'architecture mvc c'était le même principe. Je ne vais pas dévoiler le code car cela reste dans le cadre du privé.

3.2.2 Présentation de l'interface

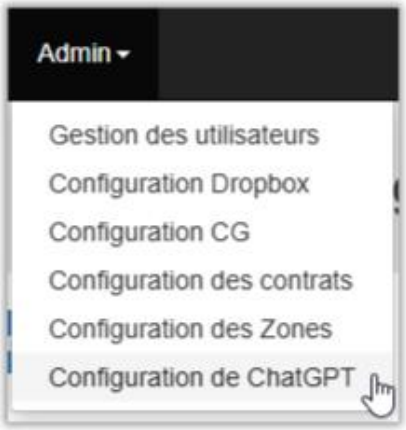


Extrait de la doc : L'interface

3.1 Configuration de ChatGPT

L'accès à l'interface de configuration se fera à l'aide du parcours suivant :

Admin > Configuration de ChatGPT



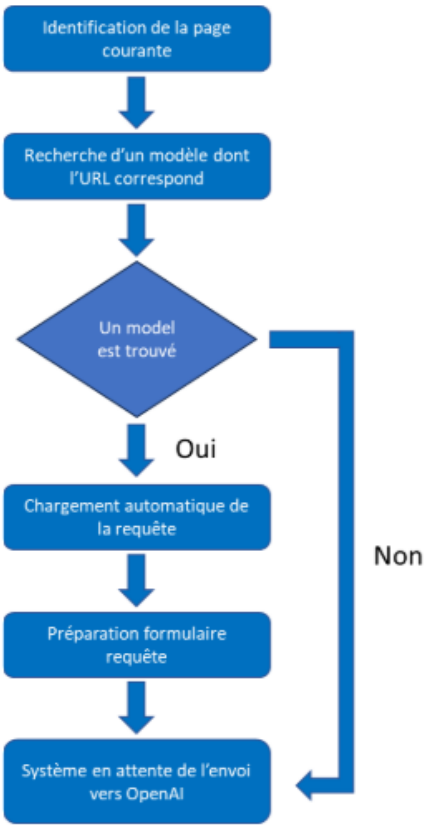
Extrait de la doc : Le menu de configuration

3.1.2.2 *Interface :*

Extrait de la doc : L'interface de configuration

3.2.3 Ouverture de l'interface

A l'ouverture de l'interface, un certain nombre d'actions se déclencheront de façon automatique :



Si un modèle par défaut a été défini pour l'URL courant, il sera alors chargé en priorité

Le process se contentera de charger le modèle mais aucune requête ne sera automatiquement envoyée à ChatGPT / OpenAI. Cette action nécessitera obligatoirement une intervention humaine.

Extrait de la doc : L'ouverture de l'interface

Quatrième Mission « Les formulaires Batify »

Exactement la même mission qu'en première année, des formulaires avait été demandés par des clients donc je m'en suis occupé.

En parallèle de son activité de logiciel sur-mesure, INGEFOX développe une application mobile et web, à destination du personnel du bâtiment, permettant le suivi de chantier.

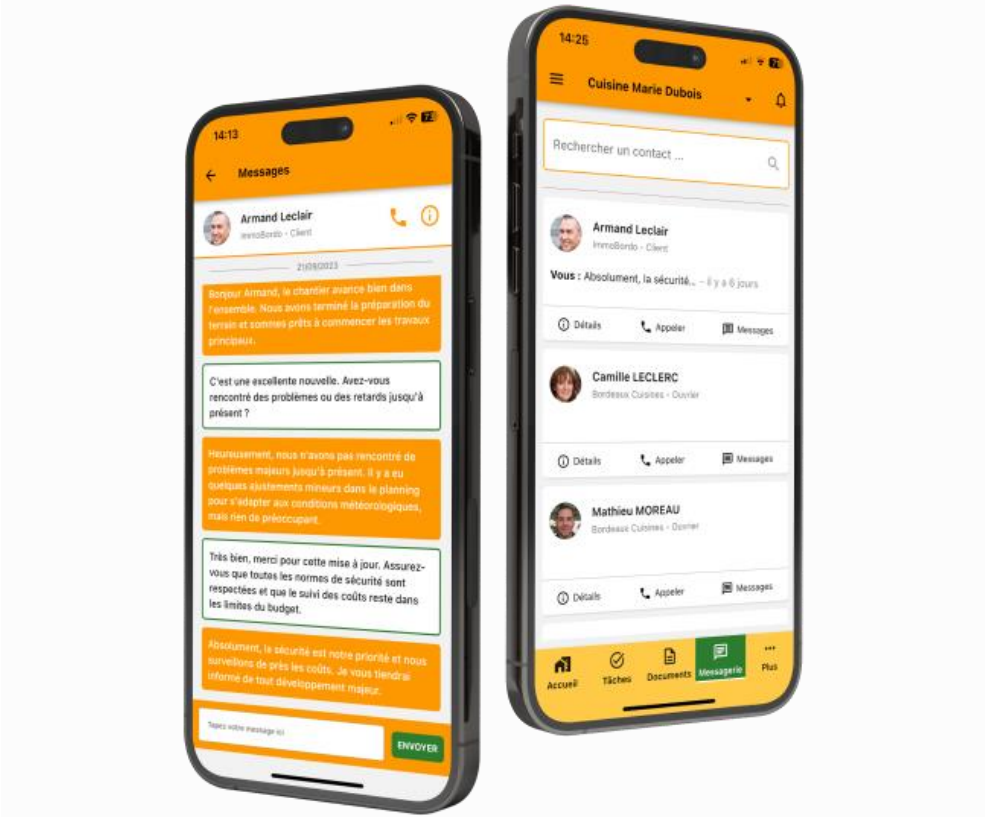
Batify

BY INGEFOX

Cette application permet :

- D'optimiser les chantiers en temps réel
- De simplifier la communication entre les équipes
- De diminuer les tâches administratives fastidieuses grâce à la centralisation à l'automatisation documentaire et à une validation rapide.

L’application mobile :



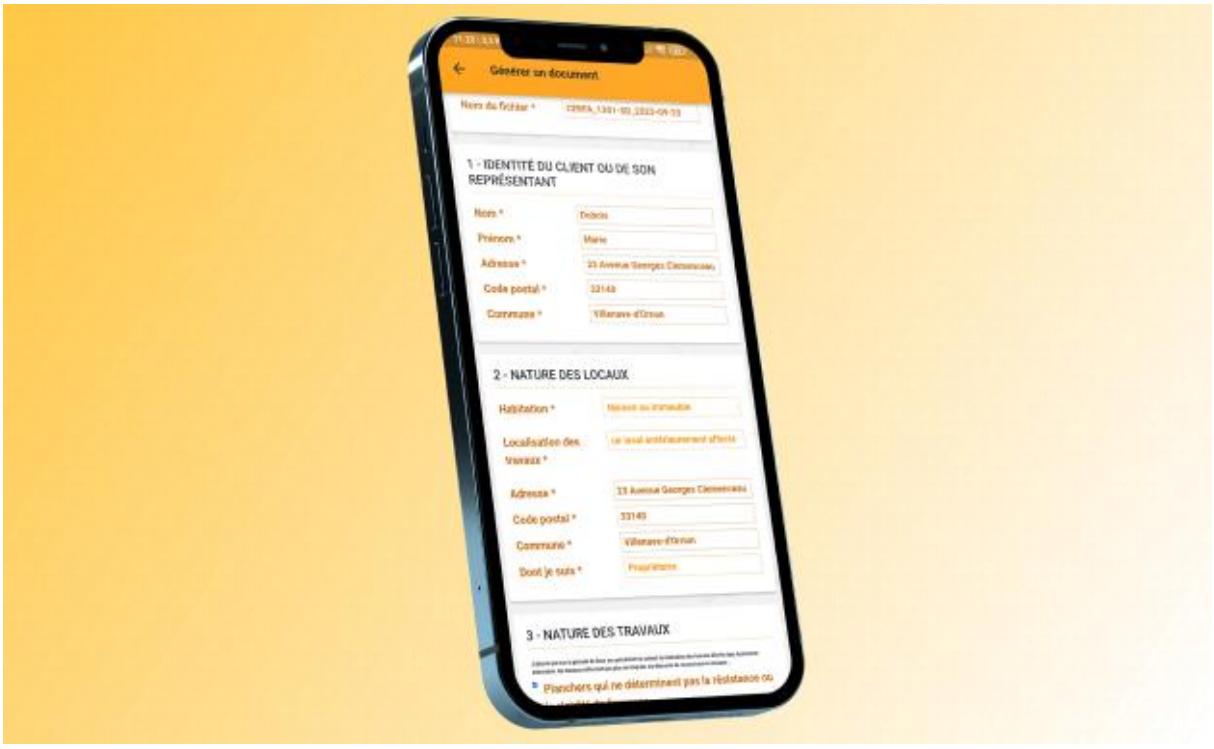
J’ai été chargé de développer des formulaires se basant sur des documents officiels du bâtiment tel qu’une fiche d’intervention, un CERFA etc...

De telle sorte à ce que quand nous le remplissons l’application nous génère le document officiel rempli. Cela limite les impressions et le temps perdu à remplir à la main les éléments.

J’ai codé 2 formulaires, en HTML, CSS, Javascript et PHP. J’ai pu préparer les PDF avec Adobe Acrobat Reader (mettre les zones de texte pour que l’ordinateur comprenne où écrire les données).

La partie technique (back end) étant déjà codée je ne m’en suis pas occupé, mon tuteur de stage m’avais déjà au préalable préparé la génération automatique du document.

Exemple de formulaire mobile :

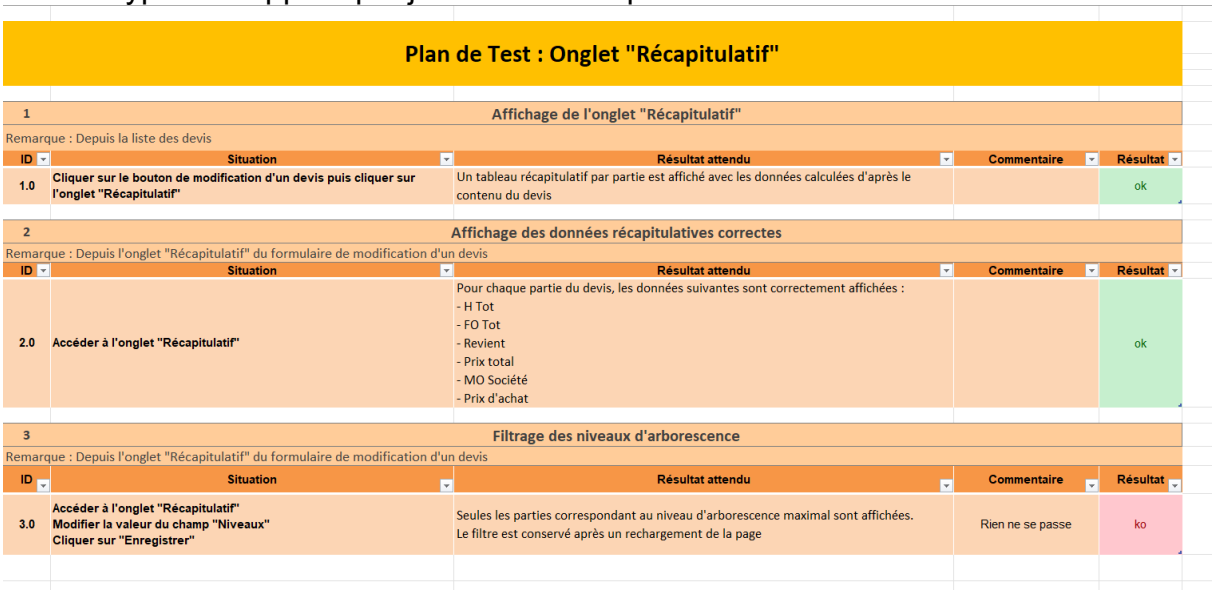


Cinquième Mission « Gérer des tests fonctionnels sur une application WEB »

J’ai également réalisé cette activité en première année la tâche était la même car les tests ce n’est pas ce qu’il manque.

Cette troisième mission consistait à suivre un plan de test précis, et d’évaluer si la demande était réalisable ou non. Si elle ne l’était pas je devais réaliser un ticket d’incident sur l’application Mantis. https://mantisbt.org/

Voici le type de support que j'avais à ma disposition :



Plan de Test : Onglet "Récapitulatif"

1	Affichage de l'onglet "Récapitulatif"			
Remarque : Depuis la liste des devis				
ID	Situation	Résultat attendu	Commentaire	Résultat
1.0	**Cliquer sur le bouton de modification d'un devis puis cliquer sur l'onglet "Récapitulatif"**	Un tableau récapitulatif par partie est affiché avec les données calculées d'après le contenu du devis		ok
2	Affichage des données récapitulatives correctes			
Remarque : Depuis l'onglet "Récapitulatif" du formulaire de modification d'un devis				
ID	Situation	Résultat attendu	Commentaire	Résultat
2.0	**Accéder à l'onglet "Récapitulatif"**	Pour chaque partie du devis, les données suivantes sont correctement affichées : - H Tot - FO Tot - Revient - Prix total - MO Société - Prix d'achat		ok
3	Filtrage des niveaux d'arborescence			
Remarque : Depuis l'onglet "Récapitulatif" du formulaire de modification d'un devis				
ID	Situation	Résultat attendu	Commentaire	Résultat
3.0	**Accéder à l'onglet "Récapitulatif"** **Modifier la valeur du champ "Niveaux"** **Cliquer sur "Enregistrer"**	Seules les parties correspondant au niveau d'arborescence maximal sont affichées. Le filtre est conservé après un rechargement de la page	Rien ne se passe	ko

Je devais compléter chaque page du plan et déclarer les tickets en fonction.

Conclusion :

Ce stage tout comme le premier m'a fait énormément évoluer. Cette fois ci j'ai pu travailler sur des projets concrets et de fournir une réelle assistance à l'entreprise je n'étais plus le stagiaire qui attendait ses petites missions.

Le projet sur chatGPT m'a été donné à finir sous la forme de prestations payantes via mon auto-entreprise créée l'année précédente.

J'ai pu découvrir le travail en équipe, les relations avec les clients et surtout s'adapter aux demandes.