Maxime DAGNIÈRE

Présentation différée du projet de stage avec la formation Aformac. Du 5 février 2018 au 23 mars 2018 dans l’entreprise WS312

Création d’une messagerie instantanée

WS312 Agrandir les possibles

**Sommaire :**

* Abstract
* Résumé
* Présentation de l’entreprise
* Liste des compétences du référentiel qui sont couvertes par le projet
* Spécifications fonctionnelles
  + Contexte et objectif du projet
  + Contraintes
  + Fonctionnalités
* Spécifications techniques
  + Langages et modules utilisés
  + Description de la base de donnée
* Réalisation
  + Création salon de discussion
  + Mise en place de discussion privé
* Conclusion

**Abstract :**

WS est une entreprise qui crée ses propres logiciels. L’un d’entre eux Wospi est un ERP (Enterprise Resource Planning) modulaire qui permet aux utilisateurs d’ajouter des modules optionnels à leurs sites. Ces derniers sont déployés dans plusieurs configurations.

Actuellement, il manque au logiciel un module supplémentaire qui permettrait au site de proposer un service client intégré pour les utilisateurs finaux.

La création de ce module est mon objectif de stage. Le projet fonctionne en deux parties:

-Pour l’utilisateur l’appel du script permet de générer un environnement t’chat de discussion (pop-up de messagerie permettant d’interagir avec les opérateurs).

-Pour les opérateurs cela permet d’accéder à l’ensemble des interfaces de gestion du module: Dashboard, interface de modération, gestion des profils opérateurs et l’administration des comptes modérateurs pour le ‘super-administrateur’ (validation, suppression des comptes).

Dans le cadre de mon intervention durant le stage, j’ai travaillé dans l’objectif de rendre le module totalement autonome (création et gestion des comptes opérateurs) pour une utilisation future.

Langages et technologies utilisés:

-JavaScript.

-En front, j’ai eu la possibilité de m’appuyer sur les librairies JQuery et Bootstrap.

-Pour le back, mise en place d’un serveur nodeJS auquel j’ai ajouté les Framework express et socket.io pour la liaison entre le back et le front.

**Résumé :**

WS est une entreprise qui crée ses propres logiciels. L’un d’entre eux Wospi est un ERP (Enterprise Resource Planning) modulaire qui permet aux utilisateurs d’ajouter des modules optionnels à leurs sites. Ces derniers sont déployés dans plusieurs configurations.

Actuellement, il manque au logiciel un module supplémentaire qui permettrait au site de proposer un service client intégré pour les utilisateurs finaux.

La création de ce module est mon objectif de stage. Le projet fonctionne en deux parties:

-Pour l’utilisateur l’appel du script permet de générer un environnement t’chat de discussion (pop-up de messagerie permettant d’interagir avec les opérateurs).

-Pour les opérateurs cela permet d’accéder à l’ensemble des interfaces de gestion du module: Dashboard, interface de modération, gestion des profils opérateurs et l’administration des comptes modérateurs pour le ‘super-administrateur’ (validation, suppression des comptes).

Dans le cadre de mon intervention durant le stage, j’ai travaillé dans l’objectif de rendre le module totalement autonome (création et gestion des comptes opérateurs) pour une utilisation future.

Langages et technologies utilisés:

-JavaScript.

-En front, j’ai eu la possibilité de m’appuyer sur les librairies JQuery et Bootstrap.

-Pour le back, mise en place d’un serveur nodeJS auquel j’ai ajouté les Framework express et socket.io pour la liaison entre le back et le front.

**Présentation de l’entreprise :**

L’entreprise WS312 avec slogan « Agrandir les possibles » a été fondé le 01/01/0000 par Baptiste et Carole Moreau. Elle est spécialisée dans le développement web.

**Listes des compétences de référentiel qui sont couvertes par le projet :**

• Mettre en place une base de données

• Développer des pages web en lien avec une base de données

• Maquetter une application

•Développer une interface utilisateur

Ayant travaillé sur le modèle de données No SQL et en back-end le langage JavaScript durant le stage, je n’ai pas eu l’occasion de validé les compétences SQL et PHP.

Mais celle-ci ont déjà été acquise durant ma formation à travers de nombreux projets. Pour cette raison que mon maître de stage m’a confié ce projet totalement différent de tout ce que j’avais rencontré jusqu’à présent.

Mais si vous me le demandiez lors de mon oral, je pourrais vous décrire mes autres projets qui pourrais valider ces compétences.

Merci de votre compréhension.

**Spécifications fonctionnelles :**

**1. Contexte et objectif du projet :**

WS312 est une entreprise qui crée ses propres logiciels. L’un d’entre eux Wospi est un ERP (Enterprise Resource Planning) modulaire qui permet aux utilisateurs d’ajouter des modules optionnels à leurs sites.

Le but est d’ajouter à l’ERP un module permettant aux utilisateurs finaux de pouvoir contacté le service client du site par un service de messagerie instantanée.

**2. Contraintes :**

• L’application doit être en temps réelle.

• Elle doit pouvoir être totalement indépendante du site.

**3. Fonctionnalités :**

*En commun :*

• Messagerie instantané

*Utilisateur :*

• Possibilité de « noter » son opérateur.

*Opérateur :*

• Gérer ses nouvelles conversations (lorsque qu’un utilisateur créer une conversation celle-ci est en attente d’opérateur).

• Mettre à jour les informations du client si celui les donnent.

• Ajouter des commentaires sur la conversation.

• Transférer la conversation à un opérateur au choix parmi ceux en ligne.

• Définir son statuts (En ligne, break, hors ligne).

• Créer son compte opérateur (Il seras en attente de validation)

• Modifier son profil (email, nom, mot de passe, avatar)

• Avoir son Dashboard avec statistique jour, semaine et moi.

Administrateur :

• Gestion des comptes opérateurs (création, suppression).

• Validation des comptes en attentes.

• Affichages Dashboard des opérateurs.

**Spécifications techniques :**

**1. Langages et modules utilisé :**

• NodeJS

• Express

• Socket.io

• EJS

• JavaScript

• HTML

• CSS

• Mongo DB

**2. Description de la base de donnée :**

La base de donnée est de type No SQL avec Mongo DB, elle compte quatre table (user, admin, convers, message).

**Description des tables :**

*Table user :*

Champs : Idclient, name, email, phone et date.

Quand un utilisateur se connecte sur le site, une entrée est faîtes dans la table.

Son identifiant est enregistré côte client sous forme de cookie pour une récupération de ses informations et anciennes conversation.

*Table admin :*

Champs : Idadmin, email, password, name, rôle (pending, moderateur, administrateur), image, date.

La table admin contient tous les opérateurs et administrateur du service client.

*Table convers :*

Champs : Idconvers, idclient, idadmin, nameadmin, archive (yes, no, break), date, adresseIp, agent, plateform, nombremsg, appréciation.

La table convers contient tous les conversations existantes. Une conversation est créer dans deux cas de figure :

• Quand le client se connecte pour la première au site cookie idconvers est créée mais ne créer pas d’entrer dans la base de donnée.

• Quand il se reconnecte le cookie convers est analysé pour voir si une entrée existe déjà dans la table, si c’est le cas on vérifie son archive. Si l’archive correspond à ‘yes’, un nouvelle idconvers est créer.

Une conversation devient archivée quand le dernier message échanger date de plus de trois heures et qu’elle est eu un opérateur. Elle devient de type break après que le client se soit déconnecté plus de trois minutes du site mais sa conversation sera archivée seulement au bout de trois heures.

Sur la conversation sont aussi enregistré les informations personnelles du client adresse Ip, plateform et navigateur ces informations sont enregistré sur la conversation mais supprimer lorsque celle-ci est archivée. Elles sont présentes juste pour permettre à l’opérateur de trouver le problème plus rapidement si le site à des points faibles sur certaine plateforme ou navigateur.

*Table message :*

Champs : Idconvers, auteur, message, type, date.

La table message contient tous les messages échangés, ils se divisent en quatre type :

• Message : comme son nom l’indique les messages de base on se type.

• Commentaire : quand un opérateur ajoute un commentaire à une conversation, il est enregistré comme message mais de type comment.

• Log : le type log est en deux parties log et logP pour différencier les logs qui s’affichent ou non au client.

**Réalisation :**

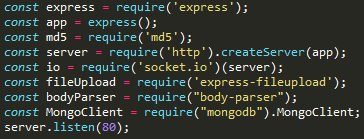
Partie 1 : Début et partie utilisateur

Phase 1 : Mise en place de l’environnement de travail

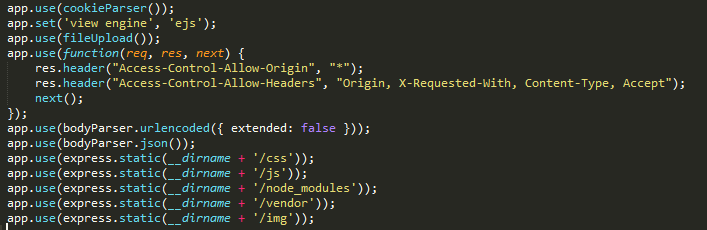
Installation des composants nécessaire sur le pc pour travailler (Mongo DB, No SQL Booster, Sublime Text ainsi que Slack principal moyen de discussion de l’entreprise).

Installation des composants pour le projet (Node JS, Socket.io, Express, EJS).

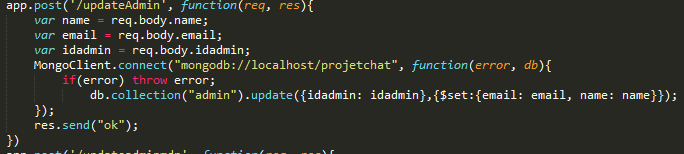
La partie serveur de NodeJS contient en premier lieu tous les appels aux différents modules :



Ensuite viennent les appels use d’Express :



Le routage est fait avec Express :



Description des éléments :

• App pour faire appel à Express.

• Get ou Post pour le paramètre.

• Entre guillemet la route sur cet exemple ‘/’.

• Ensuite on utilise (req, res) en arguments en référence au élément du header (response et request).

• Pour récupérer les données qui sont reçu, j’utilise req.body suivi du nom de l’élément reçu

• Ainsi pour finir on renvoi ce qu’on désire avec res (write, send, end).

Phase 2 : Création de la vue utilisateur :

Vue de l’utilisateur et maquette consulter les annexes numéros 1 et 2.

Pour faciliter l’intégration du module, la vue utilisateur contient seulement deux éléments :

• Un icon qui servira à l’appel du script.

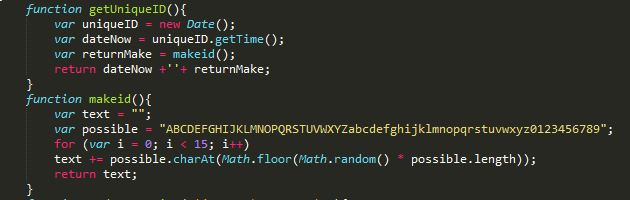
• Une fenêtre modal qui contient le système de messagerie instantanée.

Phase 3 : Fonctionnement du script client:

Première partie côté client.

A l’arrivée de l’utilisateur sur le site une première fonction est lancée permettant de savoir si l’utilisateur a un cookie IdNav.

S’il n’en possède pas un créer automatiquement avec la fonction getUniqueID ci-dessous



La première partie récupère la date actuelle en timestamp et la seconde une chaîne aléatoire de 15 caractères pour former un identifiant unique.

J’ai utilisé cette fonction car elle permet des milliards de combinaison possible par milliseconde.

Une fois la recherche du cookie utilisateur, on recherche le cookie conversation.

S’il n’en possède pas, on le créer avec getUniqueID. S’il en possède un, on recherche dans la table user s’il existe, si c’est le cas on regarde son archive, si elle est de type ‘yes’ on en créer un nouveau.

Ensuite une fois les créations ou récupérations cookies effectué, on connecte la page au serveur NodeJS avec Socket.io



Envoi du premier événement socket.io ‘emit’ avec l’idclient et l’idconvers, pour permettre au serveur de créer la session de l’utilisateur et rechercher si une conversation est en cours.

Deuxième partie côté serveur

Sur le serveur une variable globale sockets permet d’enregistrer en session toutes les informations du client, la clef est sont idclient :

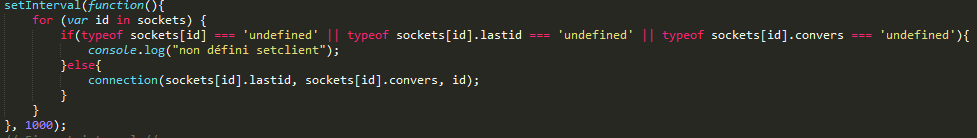
Image création sockets

De plus chaque fois que le client recharge, change ou ouvre un autre onglet, un nouvelle identifiant socket est créée pour cela. Le socket de la session est tableau qui enregistre chaque nouvelle entrer mais supprime celle qui vient d’être déconnecter. Cela permet d’avoir plusieurs pages ouvertes en même temps et permettre aux utilisateurs d’écrire et recevoir les messages sur toutes leurs pages :

Image enregistrement des sockets dans la session

Une fois la variable créer on enregistre sur la variable du client les informations avec la fonction addInfos :

Image fonction addInfos

La recherche et l’envoi des nouveaux messages reçus se fait par le biais d’un setInterval de 1 seconde :

La boucle for permet de rechercher tous les utilisateurs connecter dans la session sockets.

Cette boucle lance la fonction searchMsgClient :

Image fonction connexion

Elle fonctionne de deux façon :

• Si aucuns sockets[id].lastid est connu on récupère tous les messages de la conversation, la date du dernier sert à renseigner le lastid.

• Si un lastid est connu on recherche seulement les messages qui viennent à la suite et comme pour l’autre façon la date du dernier sert à renseigner le lastid.

A la fin si des messages ont été trouvé, on effectue un ‘emit’ de ces derniers.

Troisième partie côté client

Si des nouveaux messages ont été reçu le client reçois le ‘on’ du serveur :

A la réception de l’évènement on lance la fonction recupinfosclient qui permet de récolter les informations adresse IP, plateforme et navigateur :

Image fonction push message

Ensuite on créer une boucle for sur l’argument renvoyé par le ‘on’, on vérifie trois éléments :

• Le type message ou logP.

• Si l’id de la conversation correspond bien à celui du client

• Si l’auteur correspond au cookie du client pour savoir comment écrire le message dans la conversation.

Autres fonctionnalités du client

Envoi de message :

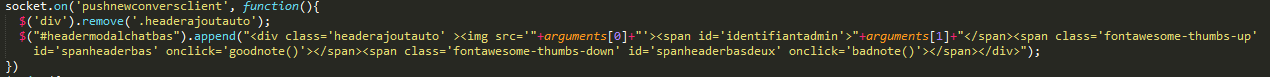
L’envoi de message se fait à l’appel du script sur le bouton envoyer.

On effectue ensuite une fonction Ajax qui envoi les données.

Fin description envoi message

Affichage de l’opérateur dans la conversation :

Quand un utilisateur envoi pour la première fois de la conversation un message aucun opérateur lui est attribué. Donc quand la conversation est attribuée un événement socket se créer pour afficher les informations de l’opérateur (Prénom et avatar).



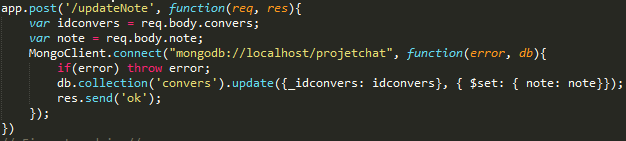
Une recherche d’opérateur est aussi faite quand la page est chargée, si un opérateur est attribué ses informations sont affichées.

Notation des opérateurs :

Image fonction notation opérateur

L’utilisateur à la possibilité de noter son opérateur en cliquant sur le pouce levé ou baissé, chacun des deux appels de script appel la même fonction :

Ensuite sur le serveur l’information est mis à jour dans la table convers :



Partie 2 : Partie opérateur

L’interface opérateur du module est composé de plusieurs page :

• Page de connexion / inscription.

• Accueil.

• Interface de modération.

• Gestion du profil.

• Administration.

Page de connexion / inscription :

*Phase 1 : Création de la vue:*

La page de login contient deux boutons qui font chacun un appel de script différent inscription et connexion.

En fonction de l’appel de script effectué soit le formulaire d’inscription ou celui de connexion apparaît.

*Phase 2 : Fonctionnement du script :*

Partie côté client

En fonction de l’appel script effectué, une des deux ‘div’ perd sa class « disparait » qui correspond à display : none.

Les deux formulaires sont traités en JavaScript et Ajax :

Image fonction Ajax connexion / inscription

Pour l’inscription et la connexion on vérifie si l’email est valide avec une fonction :

A l’inscription on vérifie que les deux mots de passe correspond et font plus de six caractères.

Si tout est bon l’Ajax se fait, en cas de succès la personne est redirigé vers la page attente ou on lui précise que son compte est en attente de validation.

Pour la connexion si le retour Ajax est un succès, on créer le cookie idadmin puis on le redirige sur la page d’accueil de l’interface.

Partie côté serveur

Inscription :



A la réception de l’Ajax on extrait les données reçu (email, password, name).

On crypte le mot de passe et on créer un identifiant unique à l’admin avec getUniqueID.

Ensuite on vérifie que l’email n’est pas déjà utilisé. Puis on peut créer le compte avec comme rôle Pending, tant qu’il n’a pas été validé par un administrateur.

Connexion :

Image connexion admin serveur

A la réception de l’Ajax on extrait les données (email et password). On crypte le password.

On recherche l’entré qui correspond à l’email puis on compare son password avec celui extrait puis crypter, s’il correspond on renvoi id de l’admin sinon on renvoi un message d’erreur.

Page Accueil de l’interface :

*Phase 1 : Création de la vue*

La page d’accueil comme toutes les pages de l’interface sont créer avec le générateur de Template EJS. Les variables transmises permettent de :

• Pouvoir afficher le nom de l’opérateur dans la barre de navigation.

Image barre de navigation

• S’il a le grade administrateur avoir accès à la partie administration.

Image menu gauche

• Pour la page interface de modération afficher les boutons des conversations en cours de l’opérateur et au terme du projet créer un bouton de notification sur toutes les autres pages.

Image création des boutons

La page d’accueil fait office de Dashboard pour l’opérateur, il peut ainsi consulter ses statistiques par jour, semaine et mois.

*Phase 2 : Fonctionnement du script*

Page interface de modération :

*Phase 1 : Création de la vue*

L’interface de modération se divise en deux parties :

Vue gauche ou l’on retrouve les éléments suivant :

Image de la vue

• Select pour le choix du statut (En ligne, break, hors ligne), par default l’opérateur est hors ligne.

• T’chat en attente (Conversation avec aucun opérateur attribué).

• Opérateur (Liste de tous les opérateurs en ligne).

Vue droite qui là encore, elle se divise en deux partie :

Partie gauche où conversation client :

Image de la vue

• Fenêtre contenant tous les messages de la discussion en cours.

• Textarea pour permettre à l’opérateur de répondre au client.

• Enfin en bas se trouve tous les boutons des conversations en cours de l’opérateur.

Partie droite où action sur la conversation de l’opérateur :

Images des deux vues

• Un select qui permet de choisir l’action par default ça affiche les information client.

• Transfert de conversation permet comme son nom l’indique de transférer l’information à un autre opérateur après avoir choisis la raison (Au choix dans un select) et cliquer sur l’opérateur voulu dans la liste.

• La vue information client permet de :

- Renseigner des informations sur l’utilisateur (Prénom, email, téléphone).

- Ajouter des commentaires.

- Le chemin du visiteur à chaque fois qu’il change de page, l’opérateur le voit pour ainsi surveiller s’il va sur les bonnes pages pour résoudre son problème.

- Afficher les statistiques du client (date de la première conversation, nombre de conversation et durée de la conversation actuel).

- En cliquant sur nombre de conversation, ça affiche les informations de ces conversation (Opérateur, note, date, nombre de message). En cliquant sur une de ces conversations ça affiche dans une fenêtre modal les messages des conversations.

Image de la fenêtre

- Affichage des informations « machine » du client (adresse IP, plateforme et navigateur).

*Phase 2 : Fonctionnement du script*

Chargement de la page :

Première partie côté client

La première action effectuée au chargement de la page est de récupérer l’idadmin contenu dans les cookies.

Page gestion du profil :

*Phase 1 : Création de la vue*

*Phase 2 : Fonctionnement du script*

Page administration :

*Phase 1 : Création de la vue*

*Phase 2 : Fonctionnement du script*