|  |
| --- |
| Pre-TPI : Messaging web-app |



Nicolas Maitre

Table des matières

[1 Analyse préliminaire 3](#_Toc5315045)

[*1.1* Introduction 3](#_Toc5315046)

[*1.2* Objectifs 3](#_Toc5315047)

[1.2.1 Groupes 3](#_Toc5315048)

[1.2.2 Messages 3](#_Toc5315049)

[1.2.3 Comptes utilisateur 3](#_Toc5315050)

[1.4 Planification initiale 5](#_Toc5315051)

[2 Analyse 6](#_Toc5315052)

[*2.1* Serveur HTTP 6](#_Toc5315053)

[2.2 Serveur WebSocket 6](#_Toc5315054)

[2.3 Wireframes des pages 6](#_Toc5315055)

[2.3.1 Page login (connexion) 6](#_Toc5315056)

[2.3.2 Page signup (Inscription) 7](#_Toc5315057)

[2.3.3 Page « Messaging » 8](#_Toc5315058)

[2.4 Base de données 9](#_Toc5315059)

[2.4.1 Modèle Conceptuel des données 9](#_Toc5315060)

[2.5 Stratégie de test 10](#_Toc5315061)

[2.6 Risques techniques 10](#_Toc5315062)

[*2.7* Livrables 10](#_Toc5315063)

[3 Conception 11](#_Toc5315064)

[3.1 Matériel Hardware 11](#_Toc5315065)

[3.2 Systèmes d’exploitation 11](#_Toc5315066)

[3.3 Logiciels 11](#_Toc5315067)

[3.4 Site web 11](#_Toc5315068)

[3.5 Base de données 12](#_Toc5315069)

[3.5.1 Modèle Logique des données 12](#_Toc5315070)

[3.6 Serveur http 13](#_Toc5315071)

[3.6.1 Serveur de distribution de fichiers 13](#_Toc5315072)

[3.6.2 API de distribution d’images 13](#_Toc5315073)

[3.6.3 API 13](#_Toc5315074)

[3.7 WebSocket server 14](#_Toc5315075)

[4 Réalisation 15](#_Toc5315076)

[4.1 Structure de répertoires 15](#_Toc5315077)

[4.2 Description des fichiers 16](#_Toc5315078)

[4.3 Librairies externes 16](#_Toc5315079)

[*4.4* Description des tests effectués 17](#_Toc5315080)

[*4.5* Erreurs restantes 17](#_Toc5315081)

[*4.6* Liste des documents fournis 17](#_Toc5315082)

[5 Conclusion 17](#_Toc5315083)

# Analyse préliminaire

## Introduction

Le but de ce projet est de créer une application de messagerie instantanée, accessible via un navigateur web. Il sera possible d’envoyer des messages à des groupes ainsi qu’à une personne par message privé. La solution devra également intégrer des profils utilisateur personnalisable ainsi que des possibilités de modération par les administrateurs.

Ce projet est réalisé dans le cadre du CPNV lors du pré-TPI. Il est réalisé dans le but d’entrainer les compétence/ apprendre les technologies pour le TPI (travail final de CFC).

## Objectifs

### Groupes

L’envoi de message se fait à l’intérieur de groupes de discussion composé de 3 personnes ou plus. Un groupe implicite est créé pour les discussions privées.

Un groupe est créé en y ajoutant des utilisateurs par leur pseudo.

Un utilisateur peut à tout moment quitter le groupe de discussion.

Le créateur du groupe a les permissions suivantes :

* Changer le titre et l’image du groupe
* Ajouter et expulser un membre au groupe
* Dissoudre le groupe

### Messages

Les messages peuvent contenir :

* Du texte (sans formatage)
* Une image (jpg, png)
* Une animation (.gif)
* Des liens hypertexte

Il n’est pas possible de mettre plus qu’une pièce jointe dans un message.

### Comptes utilisateur

#### Type de comptes

Il y a trois types de compte utilisateur :

* Messager  
  *Peut modifier son propre profil, créer des groupes et envoyer des messages.*
* Modérateur  
  *Hérite des actions du messager. Peut avoir accès à un échange sur la demande d’un des membres. Il peut y bannir un membre pour comportement toxique p.ex. Dans ce cas le créateur du groupe recevra une notification pour l’informer de la demande de modération.*
* Administrateur  
  *Hérite des actions du messager et du modérateur. Il peut :*
  + *Supprimer un compte utilisateur.*
  + *Mettre le site en maintenance (affichage d’une page spéciale)*
  + Restaurer un des backups quotidiens de la base de données.

#### Paramètres de visibilité

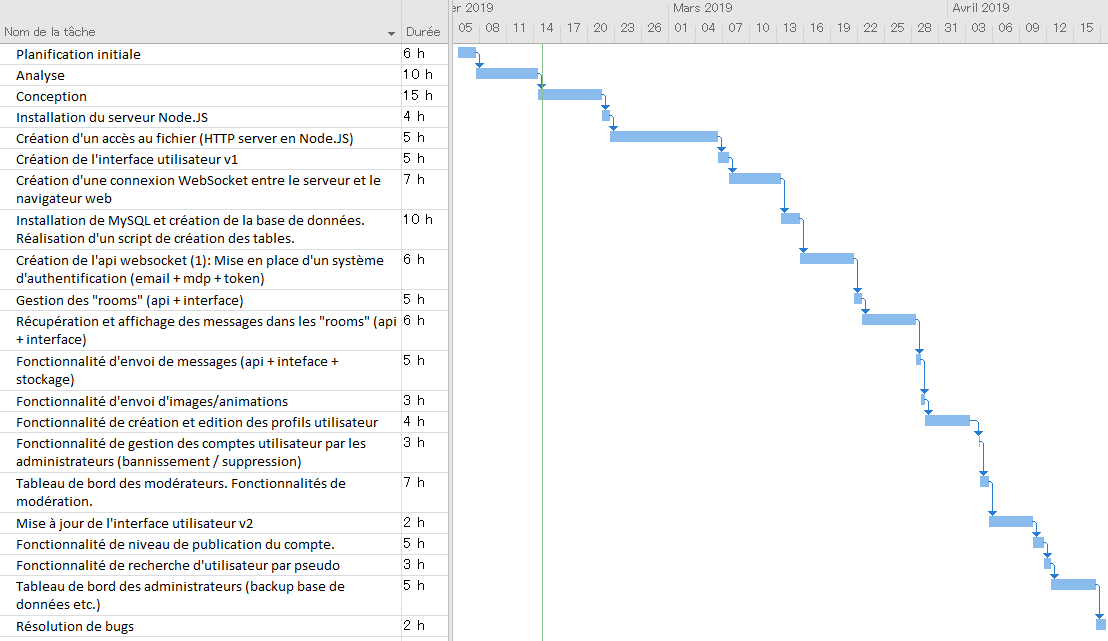
L’utilisateur peut choisir entre deux niveaux de visibilité de son compte :

* Publique  
  *Les informations du compte (prénom, nom, email, image) sont visibles par tous les utilisateurs.*
* Privé  
  *Les informations du compte autres que le pseudo sont cachées aux utilisateurs n’étant pas des « amis ».*

#### « Amis »

Pour pouvoir communiquer (en dehors des groupes), les utilisateurs doivent s’entre ajouter en « ami ». Ils peuvent le faire à l’aide du pseudo de l’autre.

## Planification initiale



# Analyse

## Serveur HTTP

Le serveur http a trois utilisations :

* **Serveur de distribution de fichiers**  
  Permet à la page web de s’afficher au client.
* **Distribution d’images dynamiques**  
  Permet à la page web d’afficher les images « dynamiques » tels que les images contenues dans les messages ou les photos de profils utilisateur.
* **API**  
  Permet de récupérer les informations à afficher tels que la listes des discussions actives.  
  Permet également de mettre à jour les informations (du profil p.ex)

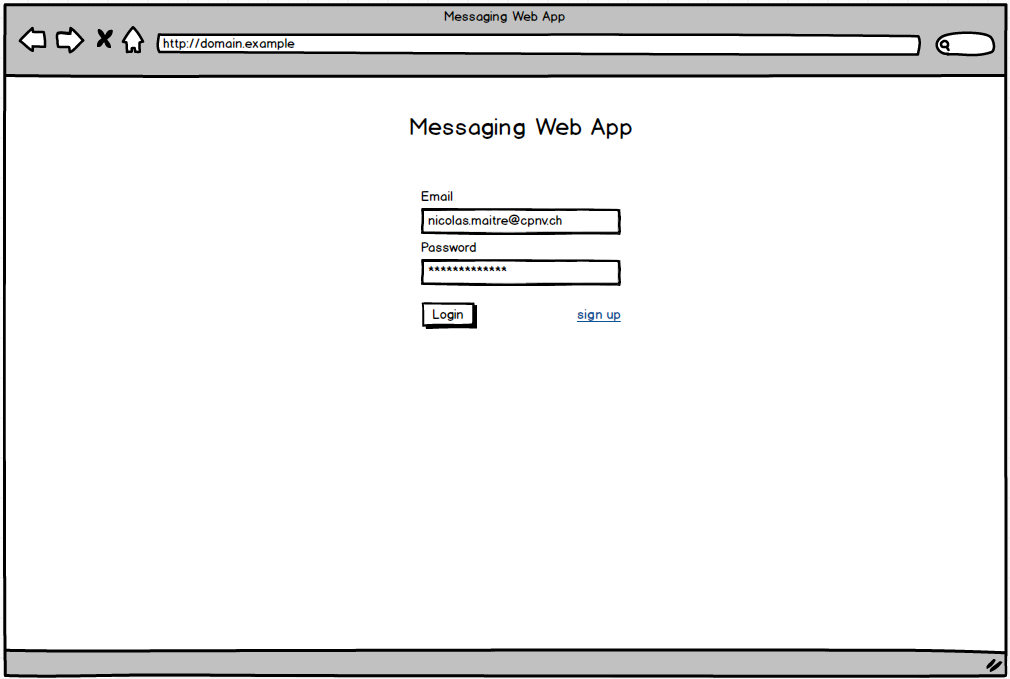
## Serveur WebSocket

Le serveur WebSocket servira à communiquer avec les clients pour se transmettre les messages.

Pour des raisons de séparation des processus, le WebSocket utilisera le port 8080 et non le port 80.

## Wireframes des pages

### Page login (connexion)

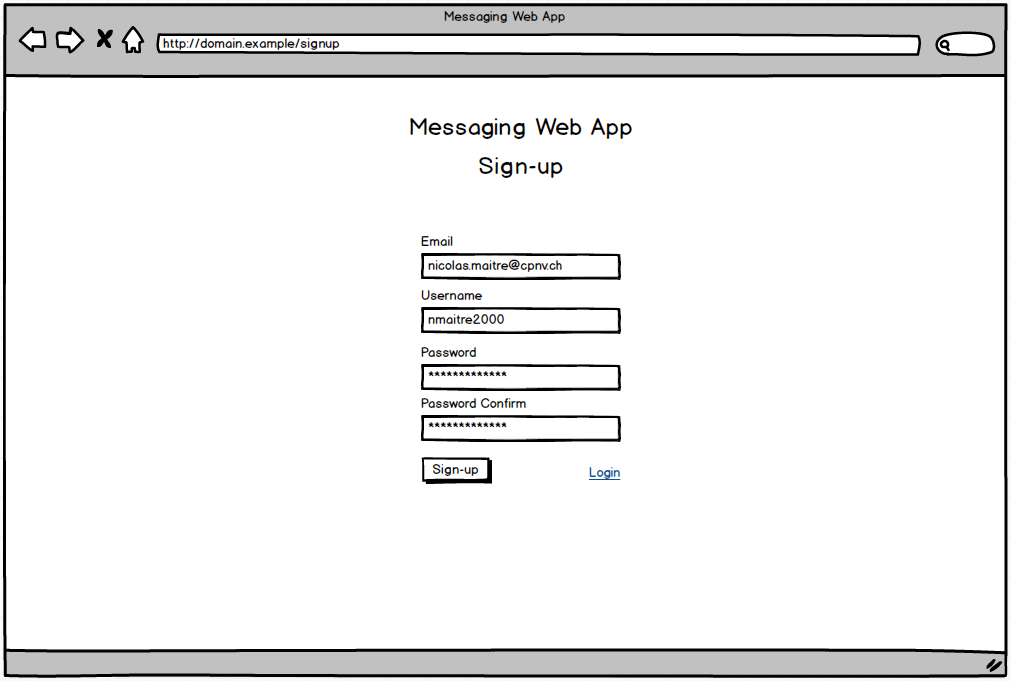


La page de login (connexion) contient un formulaire de connexion.

Le bouton « Login » permet de se connecter avec les informations entrées dans les champs textes.

Le bouton « sign up » permet d’afficher la page d’inscription.

### Page signup (Inscription)

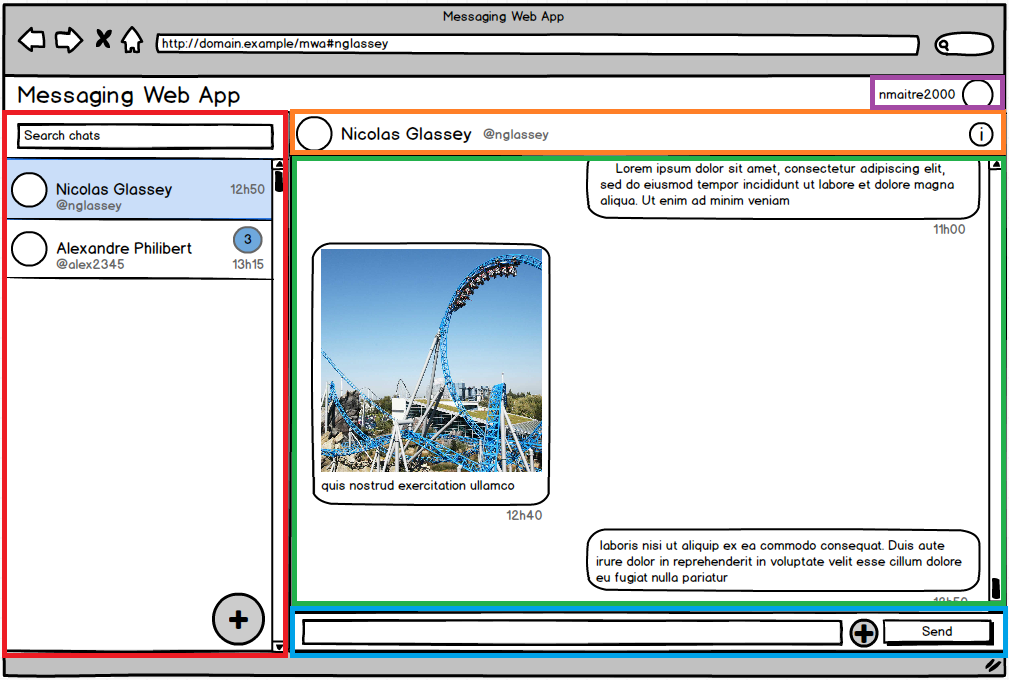


La page de signup contient un formulaire d’inscription.

Le bouton « Sign-up » permet de s’inscrire avec les informations fournies dans les champs du formulaire.

Le bouton login permet d’aller sur la page « login »

### Page « Messaging »



La page de « messaging » contient l’interface permettant de converser avec d’autres utilisateurs.

La zone des conversations contient la liste des conversations de l’utilisateur. Une barre de recherche permet de rechercher parmi les conversations (par nom).

La zone de messages contient les messages envoyés dans la conversation sélectionnée.

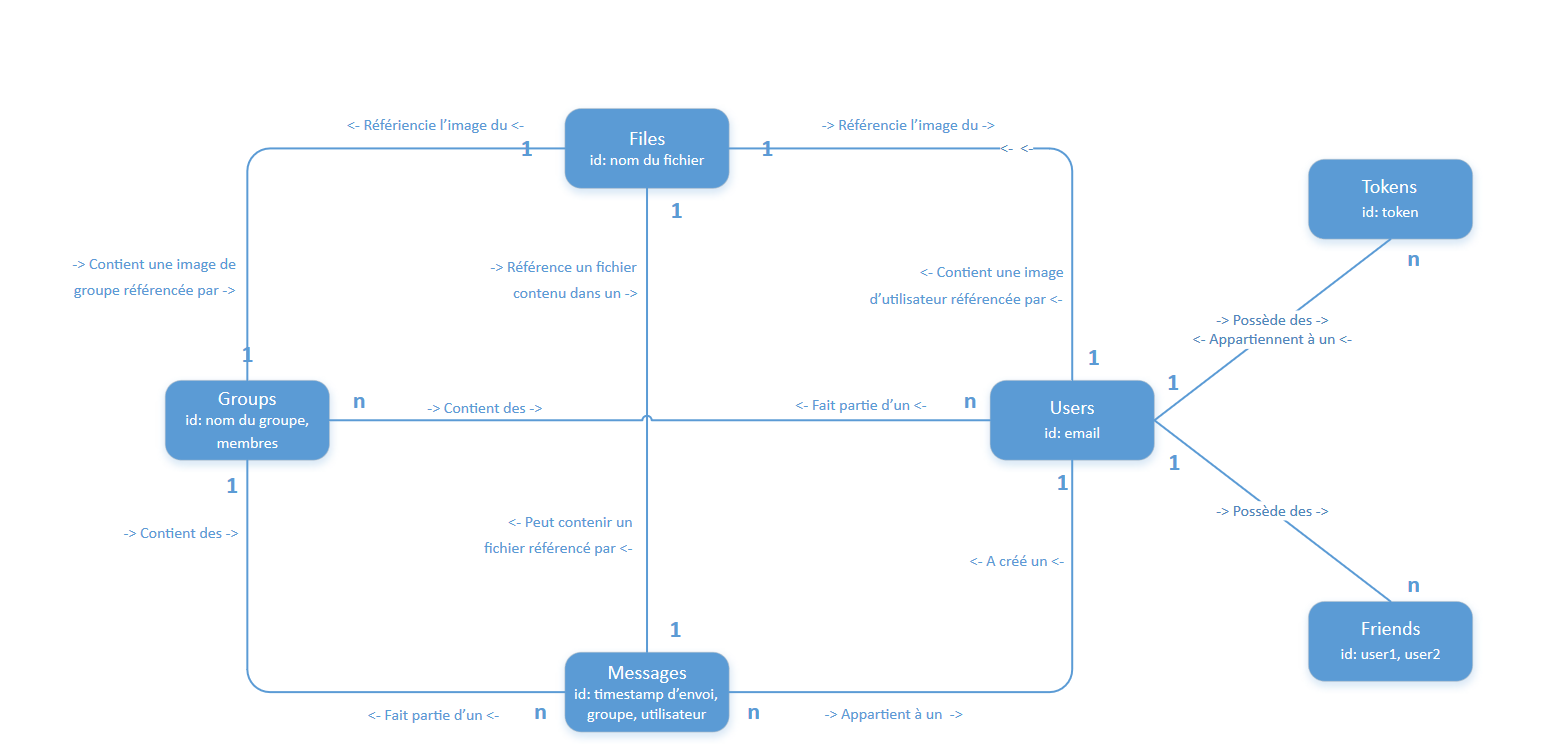
La zone de rédaction de messages contient un champ texte pour y écrire le message. Le bouton « + » à sa droite permet d’y attacher une image. Le bouton « Send » permet d’envoyer le message.

L’en-tête de conversation contient le nom de la conversation ainsi qu’un bouton permettant d’accéder aux informations de celle-ci. Si l’utilisateur est l’administrateur du groupe, il peux modifier les paramètres du groupe dans le même menu.

La zone utilisateur contient le nom et l’image de l’utilisateur. Lorsque cliqué, un menu contenant l’option d’afficher les paramètres de compte et de se déconnecter s’affiche.

## Base de données

### Modèle Conceptuel des données



## Stratégie de test

Beaucoup de tests seront effectués durant le développement des fonctionnalités.

Ces tests seront effectués en ayant accès au code et à l’environnement d’execution.

Une série de tests sera également réalisée en fin de projet en se basant sur les fonctionnalités demandées. Ces tests seront effectuée sur le modèle de la « boite noire ».

## Risques techniques

Dans le cadre de ce PreTPI, je suis face à plusieurs technologies que je n’ai que très peu abordées auparavant :

* Serveur Linux (debian)
* Language NodeJs
* Technologie WebSocket

Apprendre ces technologies prendra une place considérable dans le planning du PreTPI. Il se peut donc que le projet accumule un retard.

## Livrables

**SEMAINE 1**

* Planification initiale envoyée par email aux experts, copie au chef de projet

**SEMAINE 2**

* Configuration et mise en place du dépôt git
* Installation des outils de développement

**SEMAINE 3**

* Première version de l’analyse avec stories
* Première version de la conception
* MCD et MLD de la base de données
* Première version du design graphique

**SEMAINE 8**

* Livraison finale du code
* Livraison finale de la documentation

# Conception

## Matériel Hardware

* Serveur : Raspberry Pi Modèle 3B
* Client : Poste CPNV

## Systèmes d’exploitation

* Serveur : Raspbian (version Raspberry pi de Debian). Sans interface graphique.
* Client : Windows 10

## Logiciels

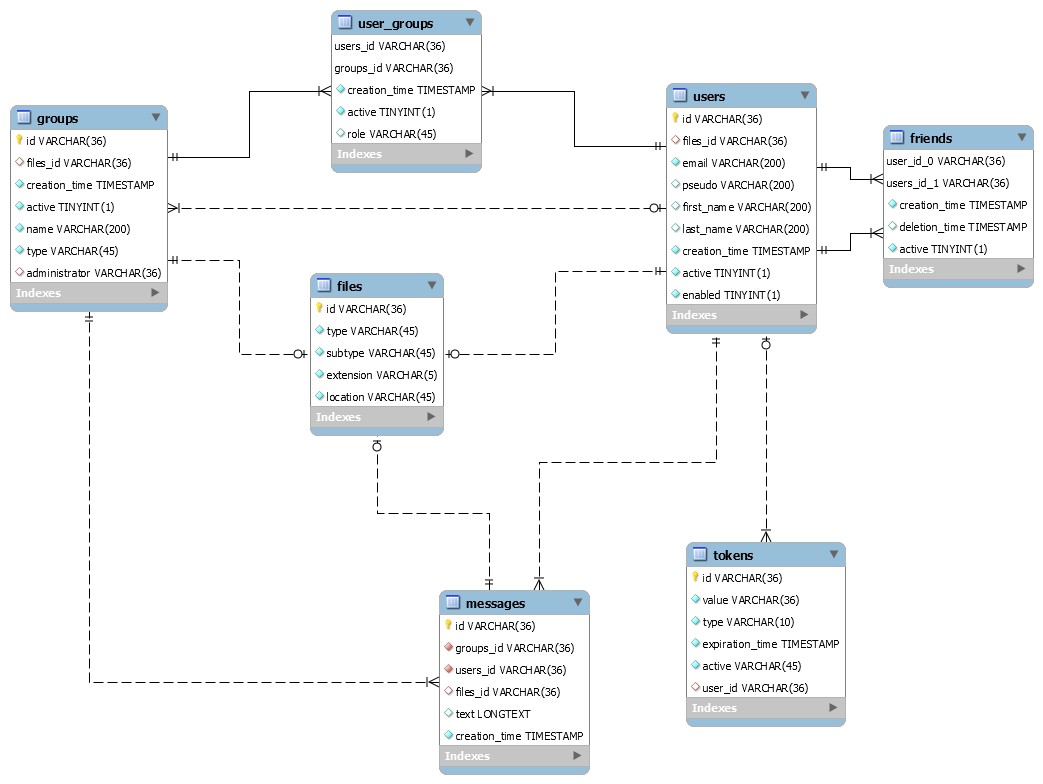
* Développement : Notepad++
* Accès au site web : Mozilla Firefox ou Google Chrome
* Serveur : NodeJS

## Site web

*Voir la section « Wireframe des pages » de l’analyse.*

## Base de données

### Modèle Logique des données



Les «id » des tables sont des UUIDs. Ils sont généré grâce à la fonction UUID() de MariaDB.

Les champs « active » des tables définissent si une entrée est considérée comme existante par le système. Cela permet de ne jamais supprimer d’enregistrements.

## Serveur http

Le serveur http a pour but d’écouter sur le port 80(http) pour donner accès à différent services en fonction de l’endpoint.

|  |  |
| --- | --- |
| Endpoint | Serveur |
| domain/ | Distribution de fichiers |
| domain/imagesApi | API de distribution d’images |
| domain/api | API |

Le serveur http sera entièrement développé dans le langage *NodeJs*. Cela me permet de maitriser complètement le comportement du serveur et des données renvoyées au client, sans devoir configurer un Framework lourd (comme Apache p.ex.)

### Serveur de distribution de fichiers

Ce service remplace l’utilisation de base d’*Apache* en permettant l’accès par http au répertoire web (/client\_resources).

Le script interprète l’url puis renvoie le fichier demandé.

### API de distribution d’images

Ce service permet un appel aux images dynamiques (p.ex images inclues dans les messages).

L’image est appelée via l’id trouvé dans la table «files» de la base de données.

Exemple d’appel :

domain.example/imagesApi/b209afec-a1cf-4c75-a91d-43f7701763c5 ?options={}

### API

Ce service permet de récupérer des informations de la base de données ainsi que d’ajouter et modifier son contenu.

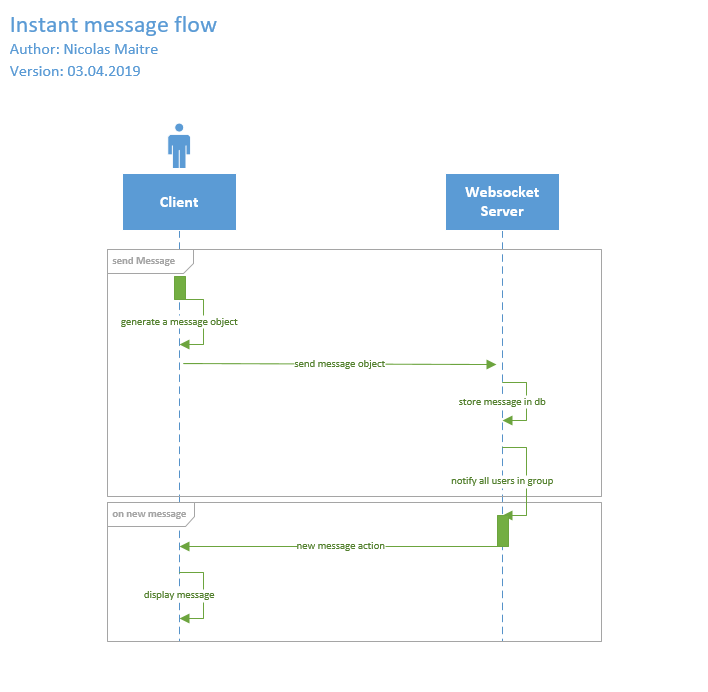
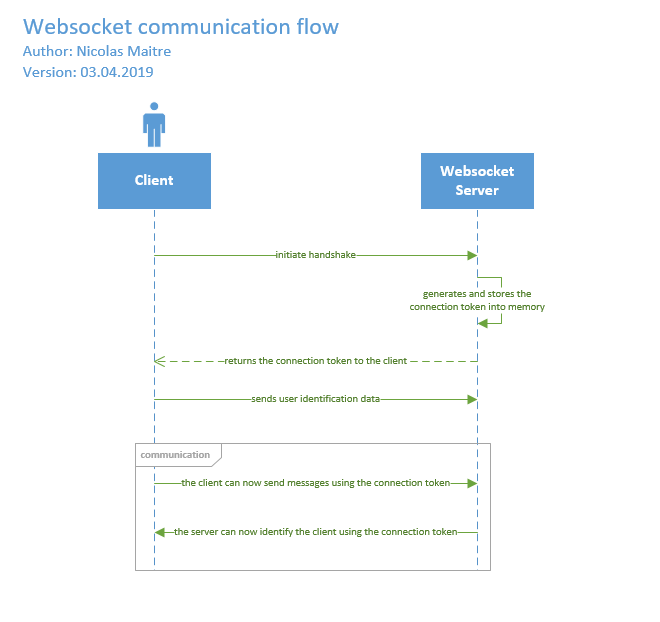
Exemple d’appel :

domain.example/api/getContacts ?options={}

## Serveur WebSocket

Le serveur WebSocket permet d’établir une connexion bidirectionnelle avec le client, permettant au serveur d’envoyer des requêtes au client (en cas de nouveau message instantané p.ex.).

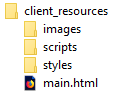
Le langage utilisé est le Node.js, car étant asynchrone par nature, il se prête bien à de la programmation évènementielle (utilisée en communication ws).



# Réalisation

## Structure de répertoires

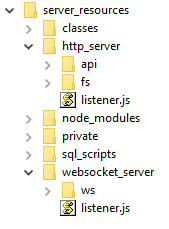
**Dossier Client**



Le dossier client contient le répertoire web. Ce sont les fichiers accessibles par le web.

* « images » contient les images fixes du site. (logo par exemple)
* « scripts » contient tous les fichiers javascript nécessaires au fonctionnement du site
* « styles » contient les fichiers css du site.

**Dossier Serveur**

****

* « classes » contient les classes javascripts utilisables globalement
* « node\_modules » contient la liste des modules node installés.
* « private » contient des fichiers non inclus dans le repository git
* « sql\_scripts » contient les scripts sql.
* « http\_server » contient les scripts permettant de faire fonctionner le serveur de distributioon de fichiers et l’api.
* « websocket\_server » contient les scripts permettant de faire fonctionner le websocket ainsi que l’envoi de messages.

## Description des fichiers

|  |  |
| --- | --- |
| client\_resources/main.html | Document html envoyé au client |
| client\_resources//styles/renderer.css | Contient toutes les classes |

*La description des fichiers JavaScript se trouvent dans l’entête des fichiers respectifs.*

## Librairies externes

**Côté Client**

Aucune librairie externe n’a été utilisée pour la réalisation du côté client. La librairie de base du JavaScript est utilisée.

**Côté serveur**

Pour simplifier le développement, les modules node suivant ont été utilisés :

* « mariadb » : permet d’accéder à la base de donnée MariaDB.
* « uuid » : permet de générer des UUIDs valides.
* « websocket » : permet de faciliter la mise en place d’un serveur websocket.

La librairie de base de Node.JS a également été utilisée.

## Description des tests effectués

*Pour chaque partie testée de votre projet, il faut décrire:*

* *les conditions exactes de chaque test*
* *les preuves de test (papier ou fichier)*
* *tests sans preuve: fournir au moins une description*

## Erreurs restantes

*S'il reste encore des erreurs:*

* *Description détaillée*
* *Conséquences sur l'utilisation du produit*
* *Actions envisagées ou possibles*

## Liste des documents fournis

Les documents fournis au chef de projet seront :

* La documentation de projet
* Le résumé de projet
* Le journal de bord de projet
* Une archive contenant le code
* (Un manuel vidéo d’utilisation au format vidéo)

# Conclusion

L’objectif de pouvoir utiliser l’application pour communiquer entre deux utilisateurs a été atteint même si les points tels que la gestion des utilisateurs et des groupes n’ont pas été terminés.

La difficulté principale a été la mise-en-en place du serveur websocket pour établir la connection entre le client et le serveur. L’autre difficulté a été la gestion de mon temps qui ne s’est que très peux axé sur la documentation.

Les extensions possibles au projet sont :

* Une gestion des comptes
* Une gestion des groupes
* L’ajout de médias dans les messages
* L’implémentation d’un chat vidéo (technologie WebRTC)

En conclusion, je pense que même si ce projet n’a pas vraiment réussi en tant que « simulation de TPI », il a réussi en tant qu’outil d’apprentissage, en me permettant d’apprendre de nouvelles technologies et de me permettre de ne pas répéter les mêmes erreurs au TPI.