|  |
| --- |
| TPI : Messaging  Web-App |



Nicolas Maitre

Table des matières

[1 Analyse 2](#_Toc8799894)

[1.1 Introduction 2](#_Toc8799895)

# Analyse

## Introduction

## Utilisateurs

Un utilisateur de l’application est identifié par un pseudo, une adresse email et un mot de passe. L’utilisateur peut prendre part à des groupes de discussion et envoyer des messages. L’aspect d’inscription et de connexion ne sera pas abordé dans ce projet car non demandé dans le cahier des charges.

## Conversations privées

Les conversations privées permettent à deux utilisateurs de s’envoyer des messages entre eux. Une seule conversation privée peut exister entre deux utilisateurs.

## Groupes

Un groupe est créé par un utilisateur. Le groupe doit contenir au minimum 3 personnes. Le groupe permet à ces utilisateurs de converser par message.

Il est possible d’ajouter un titre et une image au groupe.

## Messages

Les messages peuvent être envoyés par les utilisateurs dans les groupes ou dans les conversations privées.

Les messages contiennent du texte mais peuvent également contenir une image ou un lien hypertexte.

Le statut du message (envoyé, distribué, lu) est également affiché mais uniquement dans les conversations privées.

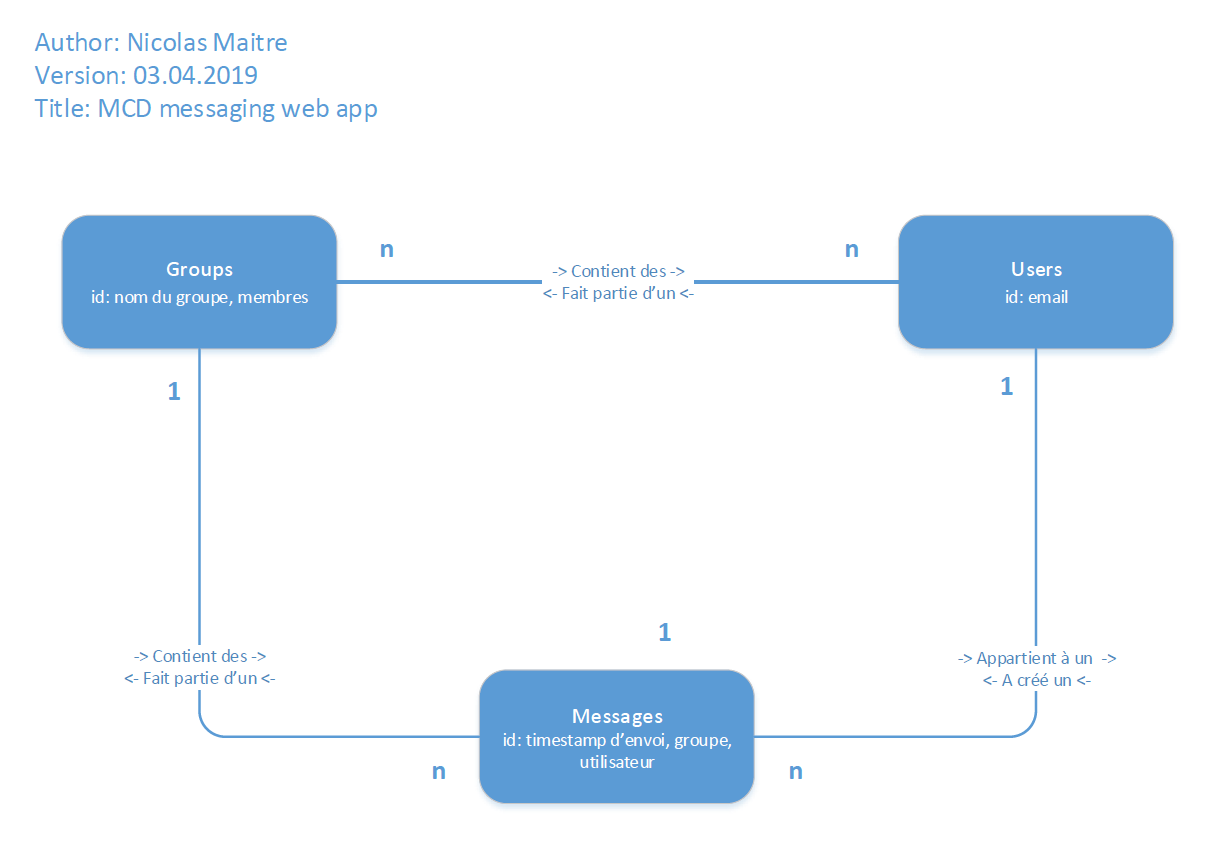
## Changements/précisions dans le cahier des charges

Certains changements ont été apportés au

* La fonctionnalité d’affichage de statut de message n’est accessible uniquement dans les groupes privés.
* Dans la page d’ajout de groupe, la liste d’utilisateurs contient tous les utilisateurs du système. En effet la fonctionnalité « d’amis » a été retirée.
* Le rendu hebdomadaire des documents peut se faire directement dans le repository git au lieu du mail.

## Données

Modèle Conceptuel des données



# Conception

## Création de Groupe

## Serveur WebSocket

## Story 1 : Répondre à un message

Pour pouvoir réaliser cette story, certains éléments doivent être mis en place auparavant :

## API

L’api permet d’interagir avec le serveur via le protocole http.

### Paramètres

Trois paramètres peuvent être passés dans la requête :

* **Action (obligatoire)(url)**  
  L’action définit le type d’action que l’on souhaite réaliser sur le serveur. Par exemple « getMessages » pour récupérer des messages.
* **Paramètres (Obligatoire)(url query)** *Params*  
  Les paramètres sont encodés en JSON et permettent à l’API d’envoyer le bon résultat sur l’endpoint.
* **Options (url query)**  
  Les options définissent le format de données en retour. Par exemple le nombre de données et l’ordre de tri. Ce paramètre est encodé en JSON
* **Authentification (Obligatoire)(Headers)** *Auth*Ce paramètre contient l’identifiant et le *token* de l’utilisateur souhaitant réaliser la requête. Ils permettent de déterminer si l’utilisateur a l’autorisation d’accéder à la ressource. Ce paramètre est également encodé en JSON.
* **Données(body)** *Data*.  
  Ce champ contient des données brutes, par exemple le contenu d’un fichier image. Il peut également contenir des données encodées en JSON.

### Format de réponse

Toutes les réponses de l’api sont encodées en JSON.

La réponse contient les paramètres suivants :

### Exemple de requête de récupération d’informations

L’api permet de récupérer des informations depuis la base de données du serveur.

Dans cet exemple, on va récupérer les 10 derniers messages du groupe référencé par l’id « 0123-1234-2345-3456-4567 ».

Domain.example/api/getMessages ? params = {groupId :’0123-1234-2345-3456-4567’} &options= {start : 0, limit : 20, order : {by : ‘creation\_time’, way : ‘desc’}}

### Exemple de requête d’envoi de fichier

L’api est également utilisée pour l’ajout de fichiers sur le serveur, notamment les images.

Dans cet exemple, la requête permet d’envoyer un fichier sur le serveur.

Domain.example/api/uploadImage ? params = {}}

## Files API

La « Files API » permet de distribuer des fichiers dynamiques. La requête contient uniquement la référence (UUID) du fichier.

Dans cet exemple, on va récupérer le fichier dont la référence est « feb0daf0-77a4-11e9-8f9e-2a86e4085a59 ».

Domain.example/files/feb0daf0-77a4-11e9-8f9e-2a86e4085a59

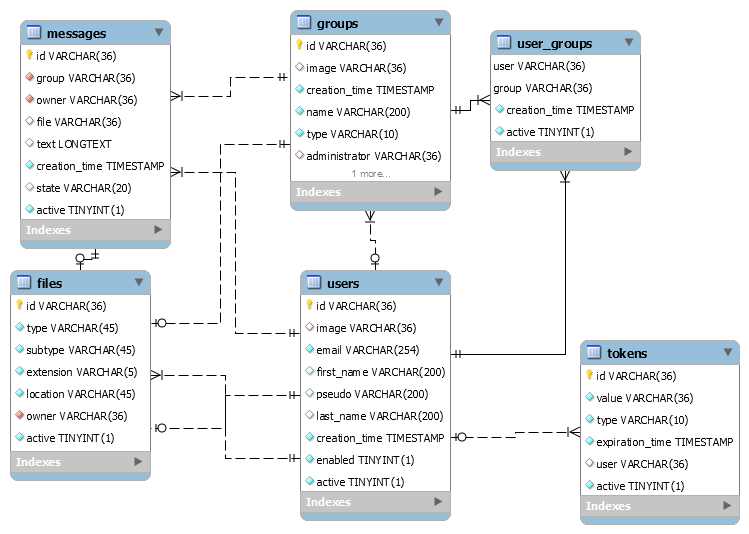
## Base de données

### Technologie

La base de donnée fonctionne sous Maria DB, un *fork* de MySQL. Les fonctionnalités et la syntaxe sont similaires.

Le choix de cette technologie s’est fait pour privilégier le logiciel libre. En effet MySQL appartient à Oracle. Ce moteur de base de données avait également été utilisée lors du PréTPI et a donc été réutilisé.

### Modèle Logique des données



## Contact à l’API

L’API est appelée à l’aide de la « Fetch API ». Malheureusement elle n’est pas encore supportée par tous les navigateurs, comme Safari et IE. Un polyfill éxiste pour combler cette lacune.

<https://github.com/github/fetch#browser-support>

Champs mot de passe

<https://stackoverflow.com/questions/386294/what-is-the-maximum-length-of-a-valid-email-address>