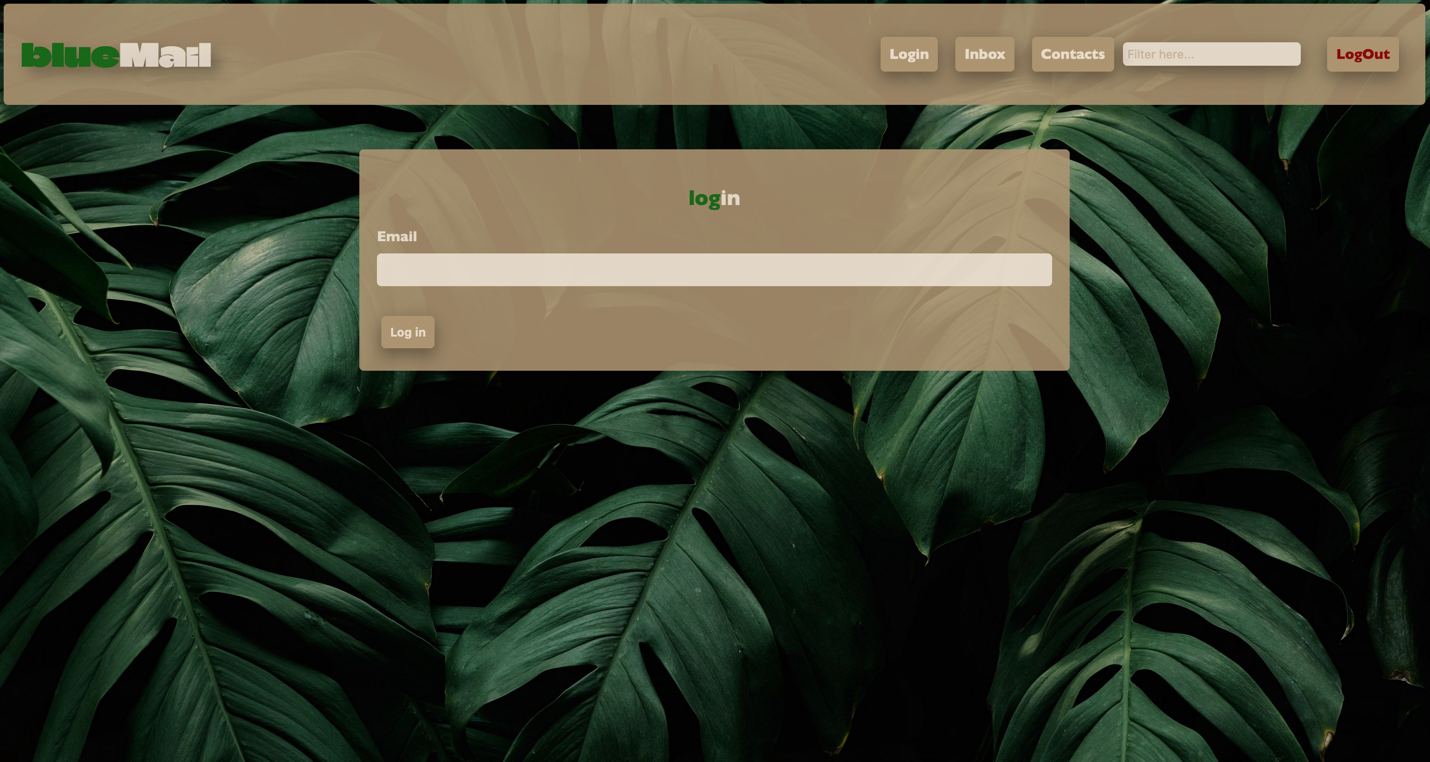
**BlueMail - Documentation**

****

- Schéma global

- Vue d'ensemble et utilisation

- Technologies

- Architecture Front-end

- Architecture Back-end

- Performances

***Schéma d'architecture globale***

Diagram

Description automatically generated

***Vue d'ensemble et utilisation***

**BlueMail** est une sorte de messagerie **privée cryptée** pour un petit groupe de personne comme une entreprise, une association, etc... Il n'est pas destiné au grand public.

BlueMail est composé d'une **interface web** accessible par le client via un **serveur central unique** (fonction d'appairage et de hors-ligne non disponible).

Le client est amené à se connecter sur son appareil personnel (imaginons son laptop professionnel). Pour se connecter il doit entrer son adresse courriel. Mais chaque organisation peut choisir de faire différemment (inscription par nom, par poste, ou par numéro d'employer etc...).

Une fois logué, une paire de clé d'encryptage est créé pour lui. La clé publique est ajoutée avec son adresse email à la base de données des contacts de la plateforme. La clé privée, elle, est gardée en localStorage, ce qui fait qu'il ne pourra jamais se déconnecter, au risque de supprimer la clé.

L'utilisateur aura 2 onglets dans **la barre de navigation principale**. Une lui permettant d'afficher les messages lui étant destinés (**'Inbox'**) et une lui permettant de voir la liste des contacts du groupe (**'Contacts'**).

En double-cliquant sur un contact, une troisième section s'affichera, **'Compose'**, qui lui permettra d'écrire un message au contact sélectionné.

Lorsqu'un message est créé, il est encrypté en utilisant la clé publique du destinataire, puis stocké encrypté par le serveur dans une base de données.

De l'autre côté, l'interface récupère l'intégralité des messages stockés, mais n'affiche que ceux qui ont pu être décryptés par le client logué.

***Technologies***

Côté client :

L'interface web est écrite en utilisant le trio **HTML**, **JavaScript** et **CSS**.

Pour la cryptographie, la librairie **Node-Forge** est utilisée côté client.

Les requêtes au serveur sont faites via **Ajax** et **JQuery**.

Côté server :

Le server est écrit en **JavaScript** en utilisant **Node.js** et **Express.js**.

***Architecture côté client***

Dans un dossier appelé **'Public'**, on retrouve les fichiers suivants :

- **courriel.html** :

Fichier html de l'interface mail, agrémenté par **courriel.css** pour le style, et **courriel.js** pour la fonctionnalité.

Il contient 4 parties principale : la **NavBar**, contenant les 3 boutons **'Login'**, **'Inbox'**, et **'Contacts'**, permettant d'accéder aux 3 autres parties. On retrouve aussi une **barre de recherche** fonctionnelle pour les listes de mails et de contacts.

-**courriel.js** :

Il est le fichier qui gère toutes les fonctions du site. Il est découpé en 3 parties : les **requêtes** pour le serveur, les **fonctions** générales, et **l'affichage** au chargement de la page. (desciption plus précise dans le document ' Fonctionnalités 'server' de courriel\_js')

-**les documents** :Images et icones.

***Architecture côté server***

Le server est composé de plusieurs fichiers :

- **les modules** : modules et package de Node.js

- **letters.json** :

Fichier de stockage des messages comprenant le nom de l'expéditeur ainsi que le message encrypté.

- **contacts.json** :

Fichier de stockage de tous les contacts, soit le nom et le la clé publique.

**- server.js** :

Assure les réponses aux requêtes suivantes :

**GET /**

Retourne le HTML au navigateur.

**GET /getLetters**

Requête envoyée quand l'utilisateur clique sur l'onglet **'Inbox'**. Le server renvoie le fichier letters.json

**GET /getContacts**

Requête envoyée quand l'utilisateur clique sur l'onglet **'Contacts'**. Le server renvoie le fichier contacts.json

**POST /addLetter**

Requête envoyée quand l'utilisateur clique sur le bouton **'Send'**. Le server ajoute le message encrypté au fichier **letters.json**

**POST /addContact**

Requête envoyée quand l'utilisateur clique sur l'onglet **'Log** **In'**. Le server ajoute le contact au fichier **contatcs.json**

***Performances***

Lors des requêtes **Ajax**, il y avait souvent des **erreurs** au premier envoi mais pas au deuxième. J'ai donc choisis de relancer automatiquement la demande si il y a une erreur Ajax.

Dans la Console, on peut reconnaître si il y a eu une erreur selon le message, si il y a un 2 à la fin, c'est pour signaler la **2e tentative**, ex : 'ERROR RETRIEVING PUBLIC KEY 2'.

**L'encryption** est faite en **1024** **bits** pour eviter une attente de 10 secondes au login.