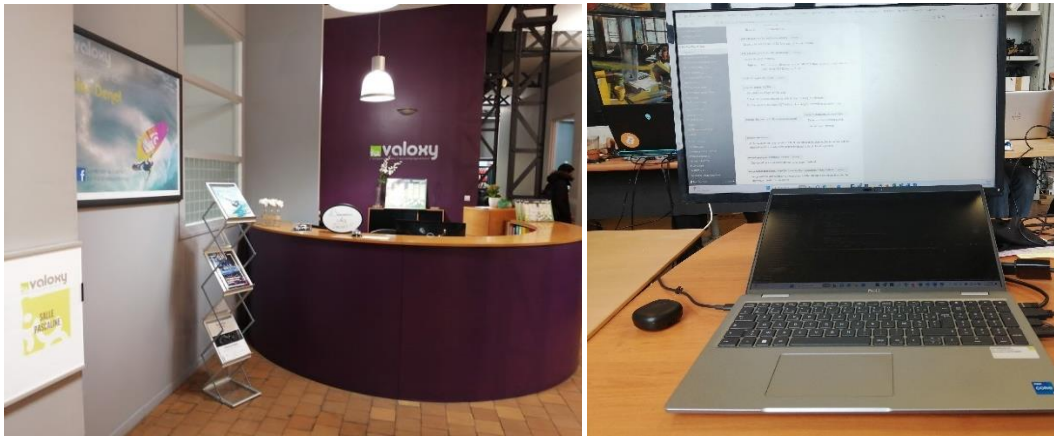


## Stage Valoxy 2023

### Introduction :

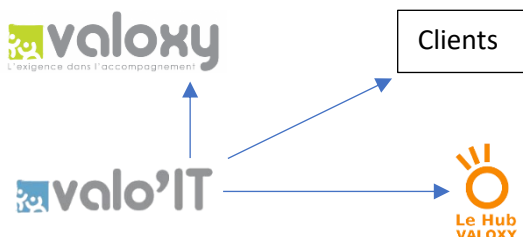
#### a) Valoxy



Valoxy est un cabinet d'expertise comptable fondé en 2004 par Ludovic Tiberghien à Lille et Saint-Omer avec une devise forte : « L'exigence dans l'accompagnement ». Toute entreprise doit tenir une comptabilité complète et détaillée. Valoxy propose donc ses services aux entreprises qui souhaitent externaliser leur comptabilité. La société offre des services adaptés à chaque entreprise. Aujourd'hui, Valoxy dispose d'un peu plus de 700 clients qui proviennent de secteurs divers et variés. Le groupe Valoxy se concentre sur ses clients au travers d'un réseau de proximité réparti sur dix sites dans les Hauts de France. Le cabinet s'est doté d'une équipe d'experts spécialisés comptant environ 170 personnes. <https://valoxy.org/>

#### b) Valo'it

Valo'IT est une société de services informatiques faisant partie du groupe Valoxy qui se situent au siège social de l'entreprise Valoxy à La Madeleine. Ses services englobent la mise en place de solutions IT, une offre téléphonique, la création de sites internet et la simplification digitale. L'équipe est restreinte, composée actuellement de seulement 4 employés. Le HUB Valoxy (espace de coworking) est administré par Valo'IT. Valo'IT compte Valoxy comme principal client, mais également d'autres entreprises qui sont également clientes de Valoxy. Étant rattaché(e) à la direction du service informatique. <https://valo.it.fr/>



## 1) Contexte

Valostart s'inspire du site web Legalstart dans le but de simplifier et de numériser le processus de création d'entreprise à travers un formulaire en ligne. Pour garantir la sécurité et l'identification des réponses, il est impératif de mettre en place un système d'authentification. Ainsi, j'étais chargé de développer cette fonctionnalité.

### Formulaire :

La page formulaire affichera les questions les unes après les autres en tenant compte de l'ordre et des conditions des questions.

Les réponses du client seront enregistrées dans la base, ce qui permettra par la suite de traiter les données reçues.

Il n'y a pas encore d'action définie, mais elles pourraient être d'envoyer un récapitulatif par mail à valoxyz ou au client, ou de générer un fichier texte ou pdf contenant les informations.

The screenshot shows a web form with the title "Souhaitez-vous créer votre société seul(e) ou à plusieurs?". Below the title, there is a subtitle "Vous pouvez créer une SARL seule ou avec d'autres associés". There are two main options presented in rounded rectangular boxes. The first option is "Je me lance seul" with a single person icon and a "Je me lance seul" button below it. The second option is "On se lance à plusieurs" with a group of three people icon and a "On se lance à plusieurs" button below it.

### Authentification :

Pour l'instant,

Des pages d'inscription et de connexion seront créées, ce qui permettra par la suite éventuellement :

Au client : d'avoir un suivi de ses réponses aux formulaires ou de prendre un rendez-vous téléphonique par exemple.

A valoxyz : de faire un retour au client, soit par mail, soit par téléphone.

The screenshot shows a web form with the title "C'est parti ! Qui effectue la démarche sur Valo'Start.fr ?". Below the title, there is a subtitle "Ces informations nous permettront de vous assister au cours de votre processus de création, et seront nécessaires pour constituer votre dossier". There are four input fields: "Prénom", "Nom", "Email", and "Téléphone". Below the input fields, there is a "Continuer" button with a right arrow.

Dans les faits, n'importe quel invité pourra remplir un formulaire (même sans être connecté).

Les informations du prospect seront demandées à la fin du formulaire.

Le cahier de charge du projet :

Création site internet Avec node.js

- 1- Créer le projet node.js
- 2- Création des différentes pages (`App.get('locals').pageContext`)
- 3- Création d'un système de compte (pages inscription/login)
- 4- Création de la base qui servira de modèle pour les formulaires
- 5- Ajouter contenu page d'accueil (demander à Océane le contenu à afficher + lien vers formulaire de création d'entreprise)
- 6- Créer le formulaire de création d'entreprise dynamique en se reposant sur la base de données
- 7- Traiter les résultats du formulaire (mail, stockage des informations, liaison avec le compte client)

#### Pour la création des pages :

Structure : reprendre le système de Legel start  
Design : Demander à Océane la charte graphique

#### Pour le système de création de compte :

Un simple système Login/Registration suffit pour le moment.

Il faudra donc créer une table :

Table Client

`id_client` | Nom | `email` | `password` | Identifiant | mot de passe

#### Pour le modèle de base de données :

Les formulaires devront être dynamique, ce qui veut dire que l'affichage se reposera sur le contenu d'une base de données que l'on (les collaborateurs) pourront remplir eux même.

#### La structure de la base pourrait être :

Table formulaire :

`id_form` (pk) | titre formulaire

1 | formulaire de création d'entreprise

Table questions :

`id_form` | `id_question` (pk) | titre question | ordre

1 | 1 | forme juridique 1

Les questions seront affichées dans l'ordre qui est défini dans la base, et seulement si les conditions sont respectées

Pour les conditions :

On ira chercher dans la table Rules, les différentes conditions à remplir pour afficher la question.

On y récupérera aussi les réponses à ces questions qui valident la condition :

Table Rules

`id_client` | `id_question` | `question_requise` | `reponse_requise`

1 | 16 | 1 | SAS\_ou\_EURL

2 | 17 | 1 | AUTOENTREPRISE

Par exemple ici :

Lorsque l'on va arriver à la question 16, on va récupérer dans cette table la liste des conditions pour la question 16

En l'occurrence, il n'y en a qu'une : Il faut que le que l'utilisateur ai renseigné « SAS », « EURL » ou « SARL » pour la question 1.

Si oui, la question 16 sera affichée, sinon, on passe à la suivante

On peut donc remarquer qu'`id_question` doit forcément être supérieur à `question_requise`, puisque la question requise doit être renseignée, et donc affichée avant.

Il va donc falloir stocker les réponses de l'utilisateur dans une base :

Table réponses :

`id_formulaire` | `id_question` | `id_client` | réponse

1 | 1 | 1 | SARL

Une fois la base de données créée, il va falloir coder en javascript, un module qui, en appelant un formulaire, affichera les questions une par une, en respectant les conditions.

À la validation du formulaire, on aura donc toutes les réponses que le client a renseigné à celui-ci, et on pourra les traiter comme on le souhaite (mail de confirmation pour le client, mail récapitulatif pour `l'admin`, etc...)

Si problème, des tutoriels sont disponibles un peu partout.

## 2) Les technologies utilisées

On a initialement prévu d'effectuer le projet sur Symfony. Mais le responsable informatique a voulu faire le projet en javascript car il est plus familier avec le langage. On a donc décidé de le faire sur node.js avec le framework Express car c'est le plus répandu et le plus commun. Dans le framework Express, le langage qui permet l'affichage est pug (jade).

```
EXPRESS (SSH: 192.168.20)
> config
> ctrl
> mysql
> node_modules
> routes
  JS inscription.js
  JS juridique.js
  JS login.js
  > serveurs
  > sockets
  > stylesheets
  > tracing
  > utilsSQL
  > views
    IF ____word_sauve
    bdata_CA.pug
    bdata_cours_yahoo
    bdata_cours.pug
    bdata_menu.pug
    bdata_TO_CA.pug
    bdata_TO.pug
    bdata_VALID_CA.pug
    formulaire.pug
    gestion_console.pug
    gestion_copier_kati
    gestion_copier_rich
    gestion_copier.pug
    gestion_copier2.pug
    gestion_drop.pug
```

Pug est un moteur de templates JavaScript permettant de générer du HTML de manière dynamique en utilisant l'indentation pour structurer le code au lieu de balises traditionnelles.

Exemple : connexion

Html :

```
<center>
<h1>Connexion</h1>
<div class="form" style="border: 1px solid black; border-radius: 10px; width: 65%;>
  <div style="width: 60%;>
    <form id="formlogin" method="POST"><small>&nbsp;</small><small class="error" id="email-error"></small>
    <input class="form-control" type="email" name="email" id="email" placeholder="Email" required="required"/><small>&nbsp;</small><small class="error" id="pass-error" type="hidden"></small>
    <input class="form-control" type="password" name="pass" id="pass" placeholder="Mot de passe" required="required"/>
    <p></p><small class="error" id="sessid-error"></small>
    <input type="hidden" name="uid" id="sessid"/>
    <input class="g-recaptcha btn btn-primary" type="submit" value="Connexion" data-sitekey="6LeZM8AmAAAAAPT5j4hipSWT83804z7u8STuWV" data-callback="onSubmit" id="btnLogin" style="border: 1px solid black; border-radius: 10px; width: 65%;>
  </div>
</div>
<p></p><a href="/oubliMdp" class="btn btn-outline-danger" style="border: 1px solid black; border-radius: 10px;">Oubli de mot de passe? </a>
<p></p>
<p>Vous n'avez pas de compte ? <a href="/inscription">Inscrivez vous ! </a></p><small class="correct" id="correct"></small><small class="correct" id="correct2"></small>
</center>
<script src="https://www.google.com/recaptcha/api.js" async="" defer=""></script>
<script type="text/javascript" src="/js/login.js"></script>
```

Pug :

```
views > login.pug
1 extends base.pug
2
3 block append body
4   center
5     h1 Connexion
6     div.form(style="border: 1px solid black; border-radius: 10px; width: 65%;")
7       div(style="width: 60%;")
8         form#formlogin(method="POST")
9
10         small &nbsp;
11         small#email-error(class="error")
12         input(type="email", name="email", id="email", placeholder="Email", class="form-control", required)
13
14         small &nbsp;
15         small#pass-error(class="error" type="hidden")
16         input(type="password", name="pass", id="pass", placeholder="Mot de passe", class="form-control", required)
17
18         p
19         small#sessid-error(class="error")
20         input(type="hidden", name="uid" id="sessid")
21
22         input.g-recaptcha(type="submit", value="Connexion", data-sitekey="6LeZM8AmAAAAAPT5j4hipSWT83804z7u8STuWV" data-callback="onSubmit", id="btnLogin", style="border: 1px solid black; border-radius: 10px; width: 65%;")
23
24
25     p <a href="/oubliMdp" class="btn btn-outline-danger", style="border: 1px solid black; border-radius: 10px;">Oubli de mot de passe?
26
27     p
28     | Vous n'avez pas de compte ?
29     | <a href="/inscription">Inscrivez vous !
30     small#correct(class="correct")
31     small#correct2(class="correct")
32
33     script(src="https://www.google.com/recaptcha/api.js" async="" defer="")
34     script(type="text/javascript", src="/js/login.js")
```

L'environnement travaillé est sur un serveur ubuntu 20 ou le terminal peut être accédé sur VScode grâce à l'extension remote ssh. Ceci a rendu capable de travailler en collaboration avec le développeur.

Le système de gestion de bases de données relationnelles est MySQL.

Et le Workbench utilisé est DBeaver.

### 3) Réalisations professionnelles

The first screenshot shows the 'Inscription' (Registration) form. It includes fields for email (olivenvaloy@gmail.com), a password (0938832539), a confirmation password, a name, and a surname. There are 'S' and 'O' buttons at the bottom.

The second screenshot shows the 'Connexion' (Login) form. It includes fields for email and password, a 'Connexion' button, and a link for 'Oubli de mot de passe?' (Forgot password?).

The third screenshot shows the 'Espace client' (Client Space) page. It displays the user's email (olivenvaloy@gmail.com) and a 'Modifier l'email' button. Below this, there are fields for a phone number (0101010102) and buttons to 'Modifier le numéro de téléphone', 'Modifier votre nom', and 'Modifier votre prénom'. At the bottom, there is a 'Modifier votre mot de passe' button.

#### a. Base de données

client
id_client
nom
premier
mail
telephone
identifiant
mot_de_passe
role
id_session
status
tokenEmail
tokenLife
tokenMDP
tokenMDPLife

Le nom et prenom sont des valeurs facultatives que l'utilisateur peut remplir

Le mail, téléphone et le mot\_de\_passe sont des valeurs obligatoires

role permet de déterminer les droits, il peut être « CLIENT » ou « ADMIN »

id\_session est un id généré quand un individu visite le site, on l'enregistre dans la base de données quand un utilisateur est connecté

status détermine la validité de l'email, il peut être « PENDING » ou « VERIFIED »

tokenEmail et tokenLife sont des valeurs qui permet la fonctionnalité de la vérification de l'email

tokenMDP et tokenMDPLife sont des valeurs qui permet la fonctionnalité de l'oubli du mot de passe

## b. Le Routage

Le routage désigne le processus par lequel une application détermine comment répondre à une demande spécifique vers un point final, identifié par un URI (Uniform Resource Identifier) ou un chemin, ainsi qu'une méthode HTTP (GET, POST, etc.). Chaque route peut être associée à une ou plusieurs fonctions de gestionnaire, qui sont exécutées lorsque la route correspond à la demande entrante.

Dans le fichier racine :

```
JS index.js > ...
132 // --- router -> init -----
133 var router = express.Router();
134 app.use('/', router);
135
136 const jurid = require('./routes/juridique');
137 app.use('/', jurid);
138
139 const inscrRouter = require('./routes/inscription');
140 app.use(inscrRouter);
141
142 const loginRouter = require('./routes/login');
143 app.use(loginRouter);
144
145 const verifEmailRouter = require('./routes/verifEmail');
146 app.use(verifEmailRouter);
147
148 const oublMDPRouter = require('./routes/oublMDP');
149 app.use(oublMDPRouter);
150
151 const modifMDPRouter = require('./routes/modifMDP');
152 app.use(modifMDPRouter);
153
154 const errorMDPRouter = require('./routes/errorMDP');
155 app.use(errorMDPRouter);
156
157 const espaceUserRouter = require('./routes/espaceUser');
158 app.use(espaceUserRouter);
```

A la fin de chaque fichier dans le dossier /routes on place `module.exports = router;`

Ce code va générer le fichier inscription.pug quand l'utilisateur tape /inscription dans l'URL du site

```
routes > JS inscription.js > ...
17
18 router.get('/inscription', function(req, res, next) {
19 |   res.render('inscription');
20 | });
21
22
23
24 module.exports = router;
```

### c. Les Sockets

WebSocket est un protocole réseau et une API permettant de créer un canal de communication à double-sens (full-duplex) entre un serveur et un navigateur.

Ceci permet donc de faire des échanges entre serveur et client sans recharger la page web. Ceci est utile pour les messages d'erreurs et de succès pour la connexion/ inscription d'un utilisateur.

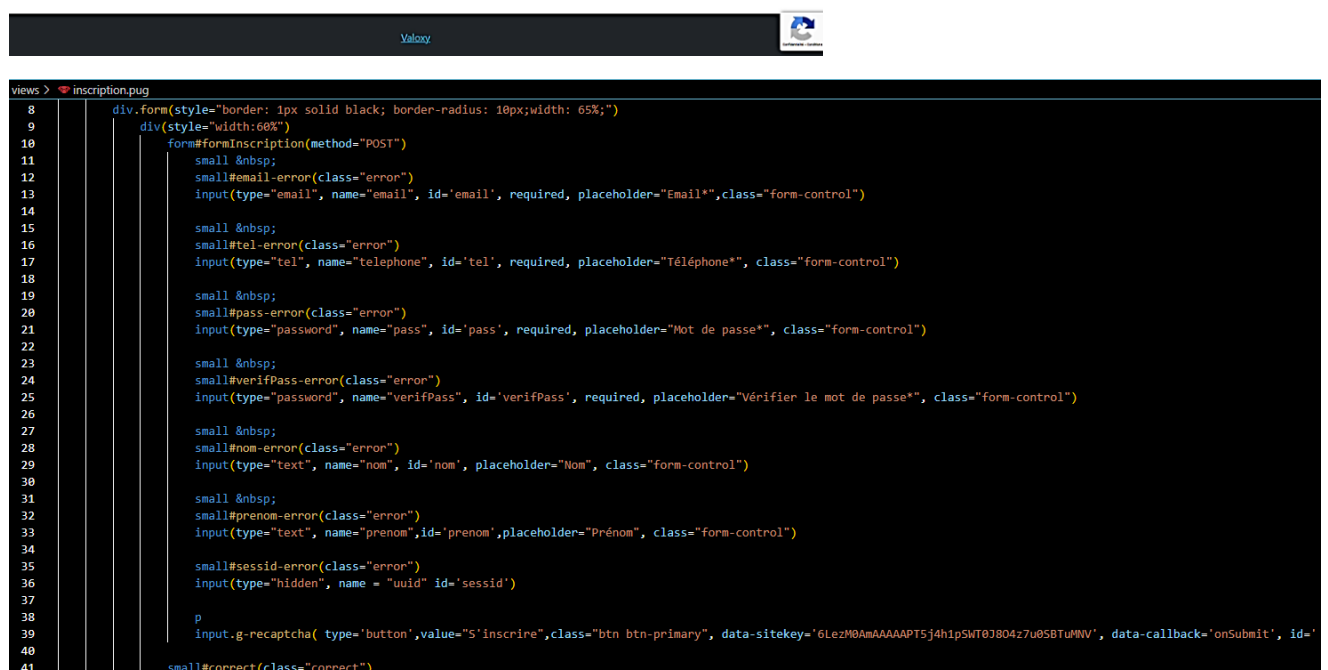
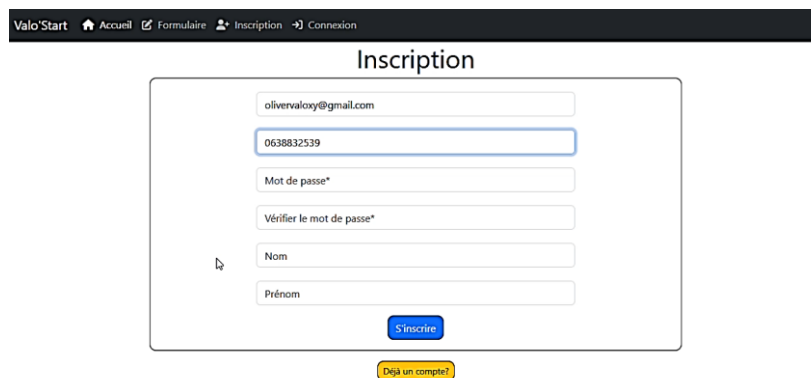
Dans le terminal :

```
npm install socket.io
```

Dans le code :

```
const { Server } = require("socket.io");
```

Exemple inscription :



```
views > inscription.pug
8   div.form(style="border: 1px solid black; border-radius: 10px; width: 65%;")
9     div(style="width: 60%;")
10      form#formInscription(method="POST")
11        small &nbsp;
12        small#email-error(class="error")
13        input(type="email", name="email", id='email', required, placeholder="Email*", class="form-control")
14
15        small &nbsp;
16        small#tel-error(class="error")
17        input(type="tel", name="telephone", id='tel', required, placeholder="Téléphone*", class="form-control")
18
19        small &nbsp;
20        small#pass-error(class="error")
21        input(type="password", name="pass", id='pass', required, placeholder="Mot de passe*", class="form-control")
22
23        small &nbsp;
24        small#verifPass-error(class="error")
25        input(type="password", name="verifPass", id='verifPass', required, placeholder="Vérifier le mot de passe*", class="form-control")
26
27        small &nbsp;
28        small#nom-error(class="error")
29        input(type="text", name="nom", id='nom', placeholder="Nom", class="form-control")
30
31        small &nbsp;
32        small#prenom-error(class="error")
33        input(type="text", name="prenom", id='prenom', placeholder="Prénom", class="form-control")
34
35        small#sessid-error(class="error")
36        input(type="hidden", name="uuid" id='sessid')
37
38        p
39        input.g-recaptcha( type='button', value="S'inscrire", class="btn btn-primary", data-sitekey='6LezMBAmAAAAAPT5j4h1pSWT0J804z7u0SBTUMNV', data-callback='onSubmit', id='
40
41        small#correct(class="correct")
```

Dans le fichier pug, on place des balises `small` possédant un id en paramètre dans le format :  
**nom de la balise-error**

On récupère les données saisies par le client dans le fichier public inscription.js

```
public > js > JS inscription.js > ...
1 document.getElementById('sessid').value = window.localStorage.getItem('id').slice(10,46);
2
3 function onSubmit(token) {
4     var user = {
5         email : document.getElementById('email').value,
6         tel : document.getElementById('tel').value,
7         pass : document.getElementById('pass').value,
8         verifPass : document.getElementById('verifPass').value,
9         nom : document.getElementById('nom').value,
10        prenom : document.getElementById('prenom').value,
11        sessid : document.getElementById('sessid').value
12    };
13
14    for (var key in user) {
15        document.getElementById(key + '-error').innerHTML = '';
16    }
17    socket.emit('user.save', user);
18 }
```

Puis on envoie les données au serveur en utilisant `socket.emit`:

```
socket.emit('user.save', user);
```

Dans le coté serveur :

```
sockets > JS index.js > start
75
76 socket.on('user.save', function (user) {
77     validateUser(user).then(async function (data) {
78         socket.emit('user.save.success', data);
79     }).catch(function (data) {
80         socket.emit('user.save.error', data);
81     });
82 });
```

On fait passer les informations dans la fonction `validateUser(user)` qui va vérifier si les informations saisies suivent le format et si les informations non-redondant comme l'email et le téléphone sont unique.



```

sockets > JS index.js > start
280
281 function validateUser(user){
282
283   return new Promise (async function (resolve, reject) {
284
285     var isEmailUnique = await ctrl.isMailUnique(user.email)
286     var isTelephoneUnique = await ctrl.isTelUnique(user.tel)
287
288     //Vérifie si tous les champs sont remplis
289     if ((!user.email.length) || (!user.tel.length) || (!user.pass.length) || (!user.verifPass.length)){
290       return reject({email: 'Les champs ne peuvent pas etre vide'});
291     }
292
293     //Vérifie la taille du contenu rempli
294
295     else if (user.email.length < 8){return reject({email: "L'email est trop court (8 caractères min)"});}
296
297     else if (user.email.length > 30){return reject({email: "L'email est trop long (30 caractères min)"});}
298
299     else if (user.pass.length < 8){return reject({pass: "Le mot de passe est trop court (8 caractères min)"});}
300
301     else if (user.pass.length > 30){return reject({pass: "Le mot de passe excède la longueur maximale"});}
302
303     //Vérifie le format du contenu rempli
304     else if (!user.tel.match(/^\(?\d{3}\)?[- ]?\d{3}[- ]?\d{4}$/)){
305       return reject({tel: "Le numéro de téléphone est incorrecte"});
306     }
307     else if (!user.pass.match(/^(?=.*[!@#$%^&'è_çàù%$(){}~])(?=.*[a-z])(?=.*[A-Z])(?=.*[0-9])(?=.*[\s])$/)){
308       return reject({pass: "Le mot de passe doit contenir lettres minuscules, majuscule, des chiffres et des caractères spéciaux"});
309     }
310
311     else if (!user.email.toLowerCase().match(/^[^<>()[]\V\\.,;:~\s@"]+([\.\<>()[]\V\\.,;:~\s@"]+)*|(\\".+\\")@((([^\<>()[]\V\\.,;:~\s@"]+)|\.)+)[^\<>()[]\V\\.,;:~\s@"]{2,})$/)){
312       return reject({email: "L'email n'est pas valide"});
313     }
314
315     else if (user.pass !== user.verifPass){
316       return reject({
317         | pass: 'Les mots de passes ne sont pas identiques'
318       });
319     }
320
321     //Vérifie si le contenu est unique
322     else if (isEmailUnique == false){
323       return reject({
324         | email: 'Un utilisateur possède déjà le meme email'
325       });
326     }
327
328     else if (isTelephoneUnique == false){
329       return reject({
330         | tel: 'Un utilisateur possède déjà le meme numéro de téléphone'
331       });
332     }
333
334     else {
335       const tokenID = uuidv4();
336       const now = new Date();
337       const expires = now.getTime() + (1000 * 60 * 15);
338       await ctrl.addUser(user, tokenID, expires);
339       place = 'verifEmail';
340       sendEmail(user.email, tokenID, place, 'Vérifiez votre email');
341       return resolve();
342     }
343   });

```

Si une erreur, la fonction renvoie un reject qui va associer un message a un clé puis les données sont envoyées au socket `user.save.error`

```

public > js > JS inscription.js > ...
25 socket.on('user.save.error', function (data) {
26
27   for (var key in data) {
28     if (data.hasOwnProperty(key)) {
29       document.getElementById(key + '-error').innerHTML = data[key];
30     }
31   }
32
33 });

```

On va parcourir les clés puis les associés aux id des balises `small`

Exemple :

# Inscription

L'email n'est pas valide

Sinon les données sont correctes, ils seront enregistré au base de données puis envoyés au socket `user.save.success`

```
sockets > JS indexjs > start
333
334     else {
335         const tokenID = uuidv4();
336         const now = new Date();
337         const expires = now.getTime() + (1000 * 60 * 15);
338         await ctrl.addUser(user,tokenID,expires);
339         place = 'verifEmail';
340         sendEmail(user.email,tokenID,place,'Verifiez votre email');
341         return resolve();
342     }
343 }
```

```
ctrl > JS indexjs > <unknown> > ctrl
80
87     async addUser(user,token,expires){
88         let sqlCon = this.mysql;
89         bcrypt.hash(user.pass, 10,async function(err, hash) {
90             const sqlCommand = `USE \`${databaseData}\`;INSERT INTO client (nom,prenom,mail,telephone,mot_de_passe,role,status,tokenEmail,tokenLife) VALUES (?);`;
91             let liste = await sqlCon.query(sqlCommand,[
92                 user.nom,
93                 user.prenom,
94                 user.email,
95                 user.tel,
96                 hash,
97                 'CLIENT',
98                 'PENDING',
99                 token,
100                 expires
101             ],true);
102         });
103     }
```

Le message dans la balise `small#correct(class="correct")` va être afficher et le formulaire va disparaître.

```
public > js > JS inscriptionjs > ...
19
20 socket.on('user.save.success', function (data) {
21     document.getElementById('correct').innerHTML = 'Vous avez presque fini! Maintenant vérifiez votre email <br>L\'email est valable que pour 15 minutes <br> Si vous ne l\'avez pas re
22     document.getElementById('formInscription').style = "display:none";
23     document.getElementById('formInscription').submit()
24 });
```

## Inscription

**Vous avez presque fini! Maintenant vérifiez votre email**

**L'email est valable que pour 15 minutes**

**Si vous ne l'avez pas reçu, vérifiez vos spam/courriel indésirable**



Déjà un compte?

Le même principe est utilisé pour la connexion, l'oubli de mot de passe, et l'espace client.

#### d. Session

Une variable session est une variable globale qui permet de mémoriser les informations de l'utilisateur.

Cette fonctionnalité est utile pour la connexion des utilisateurs.

Il faut d'abord mettre ceci au début du fichier :

```
const session = require('express-session');
```

Pour déclarer une variable session on met `req.session.email = variable`

```
routes > JS formulaire.js > ...
  9  router.get('/formulaire', function(req, res, next) {
 10    req.session.email = req.body.email
 11    res.render('formulaire',{
 12      email : req.session.email
 13    });
 14  });
```

Pour afficher sur une page web :

```
JS index.js > [e] setUpExpress
181 // XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX //
182 router.get('/', async (req, res, next) => {
183
184   res.render('root.pug',{
185     email : req.session.email
186   });
187
188   next();
189 });
```

```
views > root.pug
  1  extends base.pug
  2
  3  block append body
  4    center
  5    | h1 Bienvenue sur Valo'Start !
  6    if email
  7    | p Hello #{email}!
```

Pour supprimer une variable session : on utilise `req.session.destroy`

```
JS index.js > [e] setUpExpress
161 //bouton log-out
162 router.get('/logout', (req, res) => {
163   if (req.session) {
164     req.session.destroy(err => {
165       if (err) {
166         res.status(400).send('Unable to log out');
167       } else {
168         io.in(req.session).disconnectSockets();
169         res.clearCookie(process.env.SESSION_NAME);
170         res.redirect('/login');
171       }
172     });
173   } else {
174     res.end()
175   }
176 });
```

#### e. Nodemailer

Nodemailer permet d'envoyer des emails automatiquement.

Ceci permet de créer plusieurs fonctionnalités comme la vérification de l'email et l'oubli de mot de passe

Ceci permet de initialise le module :

```
sockets > JS index.js > ...
20 const nodemailer = require('nodemailer');
21 const transporter = nodemailer.createTransport({
22   host: "smtp-mail.outlook.com",
23   secureConnection: false,
24   port: 587,
25   tls: {
26     ciphers: "SSLv3"
27   },
28   auth: {
29     user: process.env.ACCOUNT_NAME,
30     pass: process.env.ACCOUNT_PASS
31   }
32 })
```

Ceci est la fonction cote serveur qui envoie l'email :

```
sockets > JS index.js > ...
371 function sendEmail(email,token,place,message){
372   const mailOptions = {
373     from: process.env.ACCOUNT_NAME,
374     to: email,
375     subject: message,
376     html: '<html><body><a href="https://valostart.fr:3000/'+place+'?token='+token+'&email='+email+'">'+message+'</a></body></html>'
377   }
378
379   transporter.sendMail(mailOptions,(error,info) =>{
380     if (error){
381       console.log('error');
382     } else {
383       console.log('this works '+info.response);
384     }
385   })
386 }
387 }
```

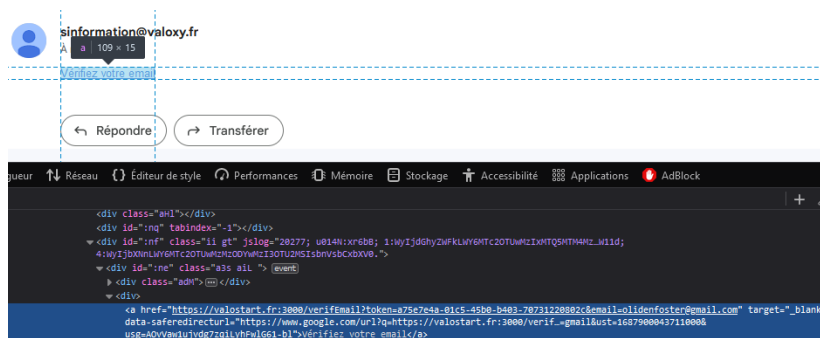
Exemple : vérification de l'email

Lors l'inscription, quand un utilisateur a saisi les bonnes informations, un token et la vie du token (15 min) est stocker dans la base de données.

```
sockets > JS index.js > ...
334   else {
335     const tokenID = uuidv4();
336     const now = new Date();
337     const expires = now.getTime() + (1000 * 60 * 15);
338     await ctrl.addUser(user,tokenID,expires);
339     place = 'verifEmail';
340     sendEmail(user.email,tokenID,place,'Verifiez votre email');
341     return resolve();
342   }
343 }
```

id_session	status	tokenEmail	tokenLife	token
f81b28a2-961d-4c9a-9067-8	VERIFIED	f9da4a50-66ef-47b4-9901-7	1686820699024	a49ec07
88df206b-79c4-40c0-abf9-e	VERIFIED	33e65bc7-2b89-4741-ac1d-c	[NULL]	86
97490dfe-ef7b-49e1-8f24-3c	VERIFIED	8dbdc13b-226e-4703-8348-	1687530663968	95
97490dfe-ef7b-49e1-8f24-3c	PENDING	a75e7e4a-01c5-45b0-b403-;	1687530785530	51

Puis l'email est envoyé.



Le token et l'email est placée dans les paramètres du lien envoyée, si le token est valide, l'email devient valide en changeant la colonne status, le client est redirigé dans la page de connexion.

```

routes > JS verifEmail.js > ...
12 router.get('/verifEmail', async function(req, res, next) {
13   var fullUrl = req.protocol + '://' + req.get('host') + req.originalUrl;
14   console.log(fullUrl);
15   const token = req.query.token
16   const email = req.query.email
17   console.log(token);
18
19   var now = new Date();
20   now = now.getTime()
21
22   var isUserVerified = await verifyToken(token, req);
23   var isTokenValid = await verifyTokenLife(token, now, req);
24   if ((isUserVerified == false) || (isTokenValid == false)){
25     console.log('invalid user');
26     res.render('verifEmail');
27   } else {
28     await updateStatus(token, req);
29     await updateEmail(token, email, req)
30     req.session.email = email;
31     console.log('réussi');
32     res.redirect('/login');
33   }
34 });

```

```

routes > JS verifEmail.js > ...
38 async function verifyToken(token, req){
39   const sqlCommand = 'use ?? ; SELECT * FROM client WHERE tokenEmail = ?';
40   let liste = await req.bases[servers.myName()].query(sqlCommand, [databaseData, token]);
41   liste = liste[1];
42   console.log(liste.length)
43   if (liste.length != 1){
44     return false;
45   } else {
46     return true;
47   }
48 }
49
50 async function verifyTokenLife(token, now, req){
51   const sqlCommand = 'use ?? ; SELECT tokenLife FROM client WHERE tokenEmail = ?';
52   let liste = await req.bases[servers.myName()].query(sqlCommand, [databaseData, token], false);
53   if (liste[1][0]){
54     liste = liste[1][0].tokenLife;
55     console.log(liste);
56     console.log(now);
57     if (liste < now){
58       return false
59     } else {
60       return true
61     }
62   } else {
63     return false
64   }
65 }
66
67 async function updateStatus(token, req){
68   const sqlCommand = 'use ?? ; UPDATE client SET status="VERIFIED" WHERE tokenEmail = ?';
69   await req.bases[servers.myName()].query(sqlCommand, [databaseData, token]);
70 }
71
72 async function updateEmail(token, email, req){
73   const sqlCommand = 'use ?? ; UPDATE client SET mail=? WHERE tokenEmail = ?';
74   await req.bases[servers.myName()].query(sqlCommand, [databaseData, email, token]);
75 }
76 module.exports = router;

```

https://valostart.fr:3000/login

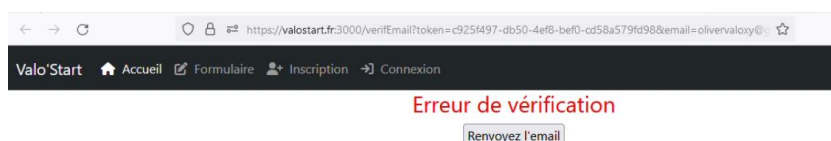
Formulaire Deconnexion Espace client

## Connexion

[Oubli de mot de passe?](#)

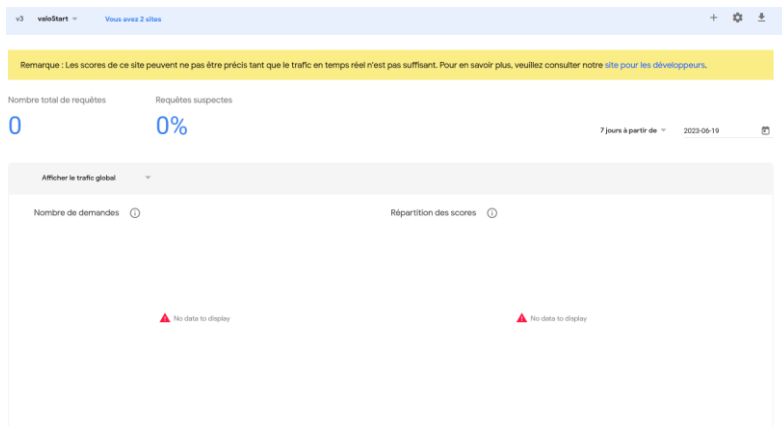
Vous n'avez pas de compte ? [Inscrivez vous !](#)

Sinon, il est dirigé dans une page qui renvoie l'email de vérification et qui recrée le token et une nouvelle vie de token.



## f. ReCaptchaV3

Le CAPTCHA est une série de tests de Turing conçus pour distinguer de manière automatisée un utilisateur humain d'un ordinateur. Utilisé en informatique, ce défi-réponse vise à vérifier que l'utilisateur n'est pas un robot. On utilise ReCaptchaV3 de Google qui est une solution qui évalue un score pour chaque interaction. Il effectue une analyse des risques en arrière-plan afin de détecter tout trafic suspect, sans perturber l'expérience de navigation des utilisateurs sur le site.



Exemple dans l'inscription :

```
views > inscription.pug
38 |
39 |   |
40 |   |
41 |   |
42 |   |
43 |   |
44 |   |
45 |   |
46 |   |
47 |   |
48 |   |
49 |   |
50 |   |
51 |   |
52 |   |
53 |   |
54 |   |
55 |   |
56 |   |
57 |   |
58 |   |
59 |   |
60 |   |
61 |   |
62 |   |
63 |   |
64 |   |
65 |   |
66 |   |
67 |   |
68 |   |
69 |   |
70 |   |
71 |   |
72 |   |
73 |   |
74 |   |
75 |   |
76 |   |
77 |   |
78 |   |
79 |   |
80 |   |
81 |   |
82 |   |
83 |   |
84 |   |
85 |   |
86 |   |
87 |   |
88 |   |
89 |   |
90 |   |
91 |   |
92 |   |
93 |   |
94 |   |
95 |   |
96 |   |
97 |   |
98 |   |
99 |   |
100 |  
```

Il faut ajouter :

```
script(src='https://www.google.com/recaptcha/api.js' async='' defer='')
```

Puis une fonction de callback pour gérer le jeton :

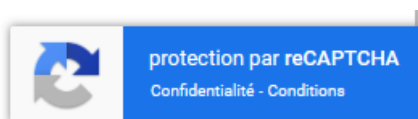
```
document.getElementById('formInscription').submit()
```

```
public > js > inscription.js > ...
19 |
20 | socket.on('user_save_success', function (data) {
21 |   document.getElementById('correct').innerHTML = 'Vous avez presque fini! Maintenant vérifiez votre email <br>\l'email est valable que pour 15 minutes <br> Si vous ne l\'avez pas re
22 |   document.getElementById('formInscription').style = "display:none";
23 |   document.getElementById('formInscription').submit()
24 | });
```

Ajout des attributs sur le bouton html :

```
input.g-recaptcha( type='button',value="S'inscrire",class="btn btn-primary", data-
sitekey='6LezM0AmAAAAAPT5j4h1pSWT0J804z7u0SBTuMNV', data-callback='onSubmit',
id='btnInscription', style=" border: 1px solid black; border-radius: 10px")
```

Pour vérifier que ceci fonctionne, il y a une icône en bas à droite de l'écran



## g. Certificat https avec Certbot

Le protocole HTTPS (Hyper Text Transfer Protocol Secure) **garantit la sécurité du site web par le chiffrement des données transmises entre le navigateur et le site internet**. La confidentialité et l'intégrité sont garanties, c'est-à-dire que ces données ne peuvent ni être modifiées, ni être espionnées.

On peut mettre un certificat https gratuitement grâce à Certbot.

Il faut tout d'abord un nom de domaine, puis on tape les commandes suivantes :

```
$ sudo add-apt-repository ppa:certbot/certbot
```

```
$ sudo apt-get update
```

```
$ sudo apt-get install certbot
```

```
root@ubuntu-js:~# certbot
Saving debug log to /var/log/letsencrypt/letsencrypt.log
Enter email address (used for urgent renewal and security notices)
(Enter 'c' to cancel): si@valoxy.fr

-----
Please read the Terms of Service at
https://letsencrypt.org/documents/LE-SA-v1.3-September-21-2022.pdf. You must
agree in order to register with the ACME server. Do you agree?
-----
(Y)es/(N)o: y

-----
Would you be willing, once your first certificate is successfully issued, to
share your email address with the Electronic Frontier Foundation, a founding
partner of the Let's Encrypt project and the non-profit organization that
develops Certbot? We'd like to send you email about our work encrypting the web,
EFF news, campaigns, and ways to support digital freedom.
-----
(Y)es/(N)o: n
Account registered.
Please enter the domain name(s) you would like on your certificate (comma and/or
space separated) (Enter 'c' to cancel): valostart.fr
Requesting a certificate for valostart.fr

Successfully received certificate.
Certificate is saved at: /etc/letsencrypt/live/valostart.fr/fullchain.pem
Key is saved at: /etc/letsencrypt/live/valostart.fr/privkey.pem
This certificate expires on 2023-09-10.
These files will be updated when the certificate renews.
Certbot has set up a scheduled task to automatically renew this certificate in t
he background.

Deploying certificate
Successfully deployed certificate for valostart.fr to /etc/apache2/sites-availab
le/000-default-le-ssl.conf
Congratulations! You have successfully enabled HTTPS on https://valostart.fr

-----
If you like Certbot, please consider supporting our work by:
* Donating to ISRG / Let's Encrypt: https://letsencrypt.org/donate
* Donating to EFF: https://eff.org/donate-le
-----
```

```
// Dependencies
const fs = require('fs');
const http = require('http');
const https = require('https');
const express = require('express');

const app = express();

// Certificate
const privateKey = fs.readFileSync('/etc/letsencrypt/live/yourdomain.com/privkey.pem', 'utf8');
const certificate = fs.readFileSync('/etc/letsencrypt/live/yourdomain.com/cert.pem', 'utf8');
const ca = fs.readFileSync('/etc/letsencrypt/live/yourdomain.com/fullchain.pem', 'utf8');

const credentials = {
  key: privateKey,
  cert: certificate,
  ca: ca
};

// Starting both http & https servers
const httpServer = http.createServer(app);
const httpsServer = https.createServer(credentials, app);

httpServer.listen(80, () => {
  console.log("HTTP Server running on port 80");
});

httpsServer.listen(443, () => {
  console.log("HTTPS Server running on port 443");
});

//force user à https
app.enable('trust proxy')
app.use(function(request, response, next) {
  if (process.env.NODE_ENV !== 'development' && !request.secure) {
    return response.redirect("https://" + request.headers.host + request.url);
  }
  next();
})
```

## h. Hashage du mot de passe

Le hachage est un algorithme qui permet de rendre un texte illisible. Il est très important de crypter les mots de passes dans une base de données.

On utilise le module bcrypt qui va encrypter le mot de passe avant de le placer dans la base de données.

Dans le terminal on tape :

```
npm install bcrypt
```

Dans le code il faut mettre :

```
const bcrypt = require('bcrypt');
```

```
ctrl > JS indexjs > <unknown> > ctrl
80
87   async addUser(user,token,expires){
88     let sqlCon = this.mysql;
89     bcrypt.hash(user.pass, 10,async function(err, hash) {
90       const sqlCommand = `USE \${databaseData}\`;INSERT INTO client (nom,prenom,mail,telephone,mot_de_passe,role,status,tokenEmail,tokenLife) VALUES (?);`;
91       let liste = await sqlCon.query(sqlCommand,[
92         user.nom,
93         user.prenom,
94         user.email,
95         user.tel,
96         hash,
97         'CLIENT',
98         'PENDING',
99         token,
100        expires
101      ],true);
102    });
103  }
```

bcrypt va ajouter une chaîne de caractères aléatoire au mot de passe saisi puis il va hasher ceci.

La fonction `bcrypt.hash()` prend en paramètre le mot de passe saisi, la complexité du chaîne de caractères aléatoire et une fonction qui va ajouter les valeurs dans la base de données.

On a donc :

id_client	nom	prenom	mail	telephone	identifiant	mot_de_passe
158	Oliver	Foster	olivervalox@gmail	0101010101	[NULL]	\$2b\$10\$/Og0nM/MmJS.VGc

Lors connexion, pour vérifier que le mot de passe est correcte, on utilise la fonction :

```
bcrypt.compare(mot de passe en clair, mot de passe encrypté);
```

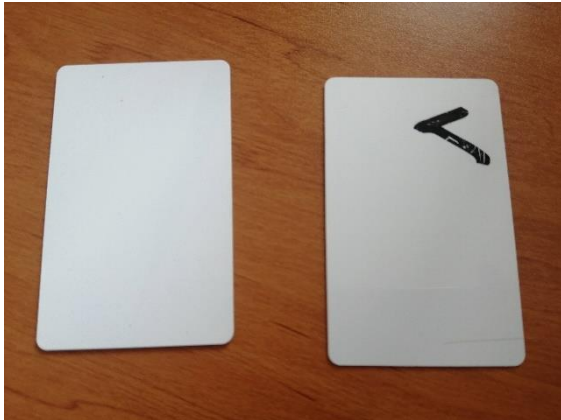
```
sockets > JS indexjs > ...
345   function validateUserLogin(user){
346     return new Promise (async function (resolve, reject) {
347
348       if ((user.email.length) || (user.pass.length)){
349         return reject({email: 'les champs ne peuvent pas etre vide'});
350       } else {
351         var doesEmailExist = await ctrl.isMailUnique(user.email);
352         if (doesEmailExist == true){
353           return reject({
354             email: "L'email n'existe pas"
355           });
356         } else {
357           var isPassCorrect = await ctrl.isUserCorrect(user.email,user.pass);
358           if (isPassCorrect == false){
359             return reject({
360               pass: "Le mot de passe n'est pas correcte"
361             });
362           } else {
363             await ctrl.addSession(user);
364             return resolve();
365           }
366         }
367       }
368     });
369   }
```

```
ctrl > JS indexjs > <unknown> > ctrl
137
138   async isUserCorrect(email,mdp) {
139     const sqlCommand = `use ?? ; SELECT * FROM client WHERE mail = ?`;
140     let liste = await this.mysql.query(sqlCommand, [databaseData,email], false);
141     liste = liste[1][0];
142     let result = await bcrypt.compare(mdp,liste.mot_de_passe);
143     return result;
144   }
```



#### 4) Autres réalisations professionnelles

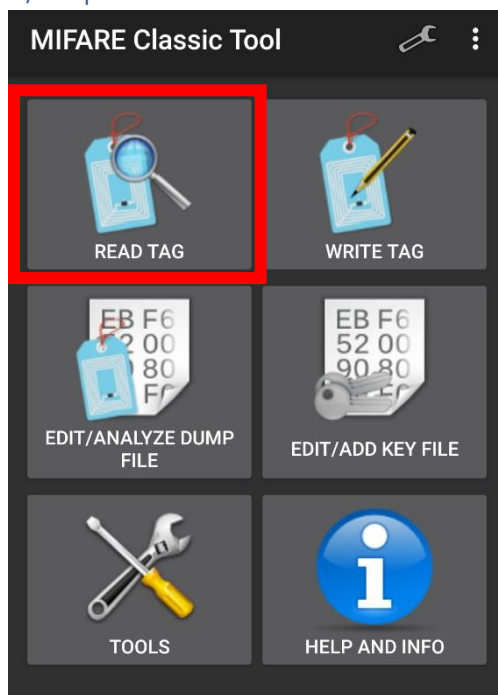
Copie de carte sur MIFARE classic (version 4.0.4)

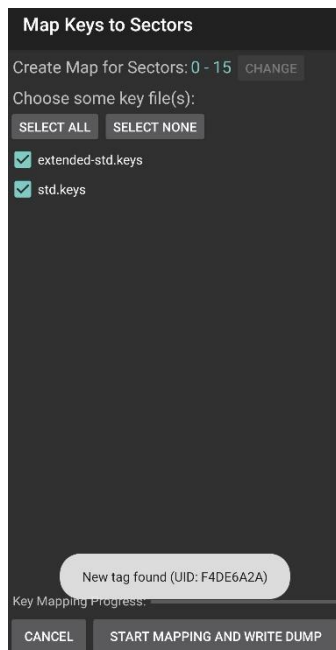


Télécharger l'application



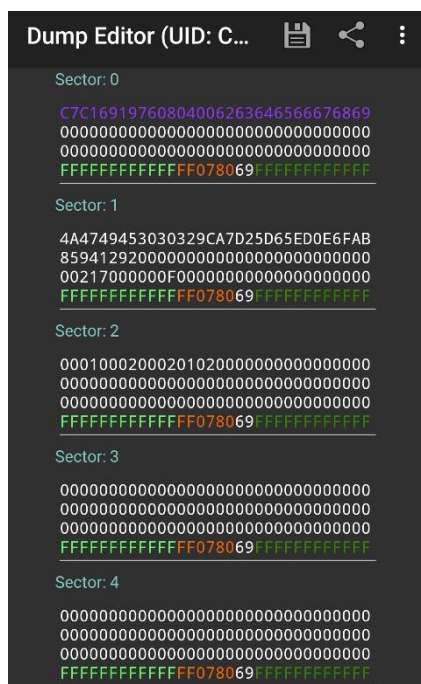
1/ Copié les données de la carte à l'application





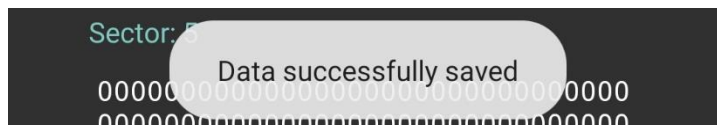
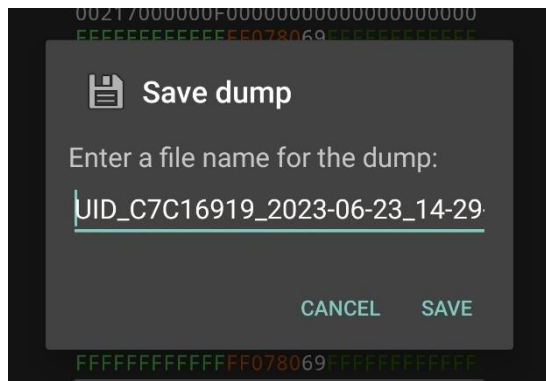
Placer la carte/badge sur le lecteur NFC puis cliquez sur

START MAPPING AND WRITE DUMP

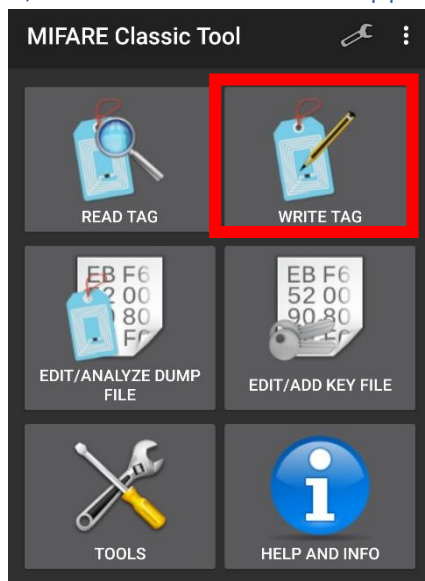


Puis appuyiez sur





2/ Collé les données de l'application à une carte



Placez la carte/badge sur le lecteur NFC

Write Tag

☒ Write Block

Sector  (Starting at 0)

Block  (Starting at 0)

Data  HEX, 16 bytes (e.g. 0A4F...)

WRITE BLOCK

☐ Write Dump (Clone)

☐ Clone UID

☐ Factory Format

☐ Incr./Decr. Value Block

New tag found (UID: F4DE6A2A)

Sélectionnez Write Dump (Clone)

**Write Tag**

☐ Write Block

---

☒ **Write Dump (Clone)**

Write a dump (or some sectors of it) to a tag. Blocks containing unknown data (---) will be skipped!

**SELECT DUMP**

☐ Show Options

---

☐ Clone UID

---

☐ Factory Format

---

☐ Incr./Decr. Value Block

Puis cliquez sur



Sélectionnez le dump que vous voulez copier :

Orange 4 56% 12:29

**Open Tag Dump**

Choose the dump to be written:


☒ UID\_C7C16919\_2023-06-23\_14-22-19.mct

☐ UID\_C7C16919\_2023-06-23\_14-29-08.mct

**SELECT DUMP**

Puis cliquez sur



 **Write Sectors**

Select the sectors you want to write:

SELECT ALL

SELECT NONE

☒ Sector 0

☒ Sector 1

☒ Sector 2

☒ Sector 3

☒ Sector 4

☒ Sector 5

☒ Sector 6

☒ Sector 7

☒ Sector 8

☒ Sector 9

☒ Sector 10

☒ Sector 11

☒ Sector 12

☒ Sector 13

☒ Sector 14

☒ Sector 15

CANCEL

OK

**Map Keys to Sectors**

Create Map for Sectors: 0 - 15 

CHANGE

Choose some key file(s):

SELECT ALL

SELECT NONE

☒ extended-std.keys

☒ std.keys

Key Mapping Progress:

CANCEL

START MAPPING AND WRITE DUMP

1231 keys loaded. Creating key map...  
Please wait...

Key Mapping Progress:

CANCEL

START MAPPING AND WRITE DUMP

Write Tag

☐

Write Block

☒

Write Dump (Clone)

Write a dump (or some sectors of it) to a tag. Blocks containing unknown data (--) will be skipped!

SELECT DUMP

☐ Show Options

Writing...

Writing tag...  
(Don't remove tag)

☐

Incr./Decr. Value Block

1231 keys loaded. Creating key map...  
Please wait...

MIFARE Classic Tool

READ TAG

WRITE TAG

EDIT/ANALYZE DUMP FILE

EDIT/ADD KEY FILE

TOOLS

HELP AND INFO

Data successfully written

App Version: 4.0.4

### 5) Evolution du projet

Une partie du projet à été repris par un autre projet de Valoxy qui s'appelle ValoAgri

<https://valoagri.fr/>

Qui fait les mêmes fonctionnalités que Valostart mais dans le secteur de l'agriculture



### 6) Avis sur le projet

J'ai commencé le stage sans avoir de connaissances sur le développement node.js. Je rencontrais des difficultés à différencier le javascript du côté client et du côté serveur. Mais progressivement, j'ai commencé à maîtriser des connaissances. Finalement, je suis satisfait de ce stage.