# TUTORIEL TITRE

ETAPES



# Contrat Solana minimal

on rajoute le champs image dans la struct

```
struct Badge
{
    string nom;
    string fin;
    string ecole;
    string image;
}

Badge[] public badges;

function addBadge(string memory _nom, string memory _fin, string memory _ecole, string memory _image) public
{
    badges.push(Badge(_nom,_fin,_ecole,_image));
}
```

# Code javascript

La fonction  addBadge de l’acesseur
L’appel dans preload.js
L’appel dans main.js
L’appel dans index.html

# Ajouter le champs dans le formulaire

```
<div class="champ">
   <label for="badge-image">Image du badge</label>
   <input type="file" id="badge-image" accept="image/*" style="padding: 8px;">
   <div id="badge-image-preview" style="margin-top: 10px; display: none;">
      <img id="badge-image-preview-image"
       style="max-width: 150px; max-height: 150px; border-radius: 8px;">
   </div>
</div>
```

# Traiter l'image au début de ajouterBadge()

```
let champsImage = document.getElementById('badge-image');
 if (champsImage.files && champsImage.files.length > 0) {
    //afficherStatut('Upload de l\'image sur IPFS...', true);
    try {
       const fichier = champsImage.files[0];
       const base64 = await new Promise((resolve, reject) => {        // Lire le fichier en base64
          const reader = new FileReader();
          reader.onload = () => resolve(reader.result.split(',')[1]);
          reader.onerror = reject;
          reader.readAsDataURL(fichier);
       });
          // TODO appeler l accesseur et recuperer l’id de l’image
    } catch (error) {
       //afficherStatut('Erreur upload image: ' + error.message, false);
       return;
    }}
```

# Pont du HTML vers accesseur.js

```
// TODO appeler l accesseur et recuperer l'id de l'image
window.api.televerserImage(base64, fichier.name);
```

```
// preload.js
contextBridge.exposeInMainWorld('api', {
    televerserImage: (base64, fichier) => ipcRenderer.invoke('image:televerser', base64, fichier)
})
```

```
// main.js
ipcMain.handle('image:televerser', async (event, base64, fichier) => {
    return await accesseur.televerserImage(base64, fichier)
})
```

```
async televerserImage(base64, fichier)
{
    console.log("accesseur.televerserImage");
}
```

# Code de Accesseur.televerserImage()

```
let formulaire = new FormData();
let buffer = Buffer.from(base64, 'base64');
let blob = new Blob([buffer]);
formulaire.append('file', blob, fichier);
```

```
const URL_PINATA = 'https://api.pinata.cloud/pinning/pinFileToIPFS';
let parametres = {
   method:'POST',
   headers : {'Authorization':'Bearer ' + CONFIG.PINATA_JWT},
   body: formulaire
};

fetch(URL_PINATA, parametres);
```

```
let reponse = await fetch(URL_PINATA, parametres);
let infosImage = await reponse.json();
if(infosImage && infosImage.IpfsHash) return infosImage.IpfsHash;
```

HTML :

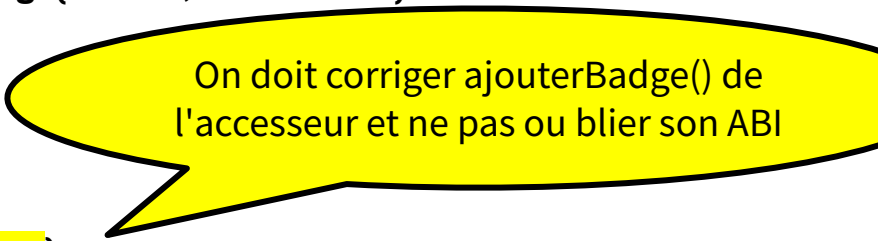
```
let idImage = await window.api.televerserImage(base64,
console.log("IPFS ID : " + idImage);
```

# Connecter l'id de l'IPFS pinata au nouveau Badtge

Dans la fonction ajouterBadge() du HTML, on veut récupérer l'id de l'image généré par IPFS et retourné par Pinata, et l'enregistrer avec nos informations d'images pour pouvoir la récupérer ultérieurement.  Voici comment faire le lien :

```
let idImage = '';

try
{
        idImage = await window.api.televerserImage(base64, fichier.name);
        console.log("IPFS ID : " + idImage);
}
 catch(e){}
```



# Ajuster notre affichage selon l'image réelle

- Utiliser le champs
- Ajuster la taille d'affichage, les margin, etc.