TUTORIEL AJOUTER et AFFICHER LISTE

ETAPES



TUTORIEL AFFICHER LISTE

ETAPES



Appel de lister dans le HTML

```
document.addEventListener('DOMContentLoaded', async () => {
    const badges = await telechargerBadges();
    afficherBadges(badges);
    let titre = await window.api.lireTitre();
    console.log(titre);
    if(titre)
    {
        document.getElementById('sousTitre').textContent = titre;
    }
});
```

Contrat Solana minimal

On rajoute des données de test par REMIX

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

Badge[] public badges;

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

function getBadges() public view returns (Badge[] memory)
{
   return badges;
}
```

Pont du HTML vers accesseur.js

```
async function telechargerBadges() {
    const oeuvres = await window.api.listerBadges();
    return oeuvres;
}
```

```
// preload.js
contextBridge.exposeInMainWorld('api', {
    listerBadges: () => ipcRenderer.invoke('badges:lister')
})
```

```
// main.js
ipcMain.handle('badges:lister', async () => {
    return await accesseur.lister()
})
```

```
class Accesseur {
    async lister() {}
```

Accesseur.js pour lire les badges

```
const CONTRACT_ABI = [
    "function getTitre() view returns (string)",
    "function getBadges() view returns (tuple(string nom, string fin, string ecole)[])"
];
```

```
class Accesseur {
  async lister() {
      let badges = await this.contrat.getBadges();
      console.log(badges);
      return badges.map(
          o => ({nom: o.nom,
              description:o.ecole,
              icone:'🎓'
          })
      );
  }
```

TUTORIEL AJOUTER BADGE

ETAPES

Contrat Solana minimal

On rajoute des données de test par REMIX

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

Badge[] public badges;

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

function getBadges() public view returns (Badge[] memory)
{
   return badges;
}
```

HTML

```
<div id="vue-formulaire">
  <div class="formulaire">
    <h2 style="margin-bottom: 25px; color: #333;">Ajouter un Badge</h2>

    <div class="champ">
      <label for="badge-nom">Nom du badge</label>
      <input type="text" id="badge-nom" placeholder="Ex: Solidity Master">
    </div>

    <div class="champ">
      <label for="badge-fin">Description</label>
      <textarea id="badge-fin" placeholder="Ex: 2025-01-01"></textarea>
    </div>
```

Appel de ajouter dans le HTML

```
async function ajouterBadge() {

    let nom = document.getElementById('badge-nom').value;
    let fin = document.getElementById('badge-fin').value;
    let ecole = document.getElementById('badge-ecole').value;

    console.log("nom : " + nom); // on peut logguer toutes les valeurs

    await window.api.ajouterBadge(nom, fin, ecole);
    // TODO ROBUSTESSE (try-catch & erreurs)
    // TODO ERGONOMIE : retroaction utilisateurs
}
```

Ajustement annuler

```
function annulerAjout() {
    // Vider les champs
    document.getElementById('badge-nom').value = '';
    document.getElementById('badge-fin').value = '';
    document.getElementById('badge-ecole').value = '';
    document.getElementById('statut').style.display = 'none';

    // Retourner à la vue principale
    afficherDamier();
}
```

Pont du HTML vers accesseur.js

```
async function ajouterBadge() {
   // CODE DHTML ici qui recupere les objets par id
   await window.api.ajouterBadge(nom, fin, ecole, '');
}
```

```
// preload.js
contextBridge.exposeInMainWorld('api', {
    ajouterBadge: (nom, fin, ecole) => ipcRenderer.invoke('badge:ajouter', nom, fin, ecole)
})
```

```
// main.js
ipcMain.handle('badge:ajouter', async (nom, fin, ecole) => {
    return await accesseur.ajouter(nom, fin, ecole)
})
```

```
class Accesseur {
   async ajouter() {nom, fin, ecole}
```

Accesseur.js pour lire les badges

```
const CONTRACT_ABI = [
    "function getTitre() view returns (string)",
    "function getBadges() view returns (tuple(string nom, string fin, string ecole)[])",
    "function addBadge(string memory _nom, string memory _fin, string memory _ecole)"
];
```

```
async ajouter(nom, fin, ecole) {
    console.log('Ajouter badge', nom, fin, ecole)
    await this.contrat.addBadge(nom,fin,ecole);
    return true;
    // TODO ROBUSTESSE
 }
```