**NOTES DU PROJET (CAMPUS INSPECT)**

1. UTILISATION DE FIREBASE
2. Nous devons commencer par importer firebase et les fonctionnalités que nous allons utiliser et ensuite les initialiser

 <script type="module">

*// Importer les fonctions nécessaires depuis les SDK de Firebase*

*import* { initializeApp } *from* "https://www.gstatic.com/firebasejs/10.14.0/firebase-app.js"; *// Pour initialiser l'application Firebase*

*import* { getAnalytics } *from* "https://www.gstatic.com/firebasejs/10.14.0/firebase-analytics.js"; *// Pour les analyses*

*import* { getFirestore } *from* "https://www.gstatic.com/firebasejs/10.14.0/firebase-firestore.js"; *// Pour utiliser Firestore*

*import* { getStorage, ref, uploadBytes, getDownloadURL } *from* "https://www.gstatic.com/firebasejs/10.14.0/firebase-storage.js"; *// Pour utiliser Storage*

*// Configuration de votre application Firebase*

const firebaseConfig = {

  apiKey: "AIzaSyDibbuBJ2p88T26P0BAB-o\_exunK0GYFdA", *// Clé API de votre projet*

  authDomain: "inspecteur-de-classes.firebaseapp.com", *// Domaine d'authentification*

  projectId: "inspecteur-de-classes", *// ID de votre projet*

  storageBucket: "inspecteur-de-classes.appspot.com", *// Bucket de stockage pour les fichiers*

  messagingSenderId: "572661846292", *// ID de l'expéditeur de messages*

  appId: "1:572661846292:web:aeb0374db2d414fef9f201", *// ID de votre application*

  measurementId: "G-NVN5GERDV6" *// ID de mesure pour les analyses*

};

*// Initialiser Firebase avec la configuration fournie*

const app = initializeApp(firebaseConfig);

const analytics = getAnalytics(app); *// Initialiser les analyses*

*// Initialiser Firestore (base de données de Firebase)*

const db = getFirestore(app); *// Maintenant, Firestore est prêt à être utilisé*

*// Initialiser Firebase Storage (pour stocker des fichiers)*

1. const storage = getStorage(app); *// Utilisez getStorage pour initialiser le stockage*
2. Ajout de données dans firestore

*// Importez les fonctions nécessaires depuis Firebase*

*import* { getFirestore, collection, addDoc, getDocs } *from* "https://www.gstatic.com/firebasejs/10.14.0/firebase-firestore.js";

*// Assurez-vous que Firebase est déjà initialisé dans votre fichier HTML*

const db = getFirestore(); *// Assurez-vous que cela soit défini après l'initialisation de Firebase*

async function addUser() {

*try* {

        const docRef = *await* addDoc(collection(db, "users"), {

            first: "Ada",

            last: "Lovelace",

            born: 1815

        });

        console.log("Document écrit avec l'ID : ", docRef.id);

    } *catch* (e) {

        console.error("Erreur lors de l'ajout du document : ", e);

    }

}

1. **Récupérer les données enregistrées dans une collection (Sur firestore)**

async function getElements() {

const querySnapshot = *await* getDocs(collection(db, "users"));

querySnapshot.forEach((*doc*) => {

  console.log(`${*doc*.id} => ${*doc*.data().born}`);

});

}

1. **Sauvegarder un fichier(Image) dans Firestore storage**

*// Fonction pour télécharger un fichier*

async function uploadFile(*file*) {

*// Créer une référence au lieu où le fichier sera stocké*

  const fileRef = ref(storage, 'uploads/' + *file*.name); *// Référence au fichier avec son nom*

*try* {

*// Télécharger le fichier à l'emplacement spécifié*

    const snapshot = *await* uploadBytes(fileRef, *file*); *// Mettre le fichier dans le stockage*

    console.log('Uploaded a file!', snapshot); *// Afficher un message de succès*

*// Obtenir l'URL du fichier téléchargé*

    const downloadURL = *await* getDownloadURL(fileRef); *// Récupérer l'URL pour accéder au fichier*

    console.log('File available at', downloadURL); *// Afficher l'URL dans la console*

  } *catch* (error) {

    console.error('Upload failed:', error); *// Afficher une erreur si le téléchargement échoue*

  }

}

*// Ajouter un écouteur d'événements pour le bouton de téléchargement*

document.getElementById('uploadButton').addEventListener('click', () => {

*// Obtenir l'élément d'entrée de fichier*

  const fileInput = document.getElementById('fileInput');

  const file = fileInput.files[0]; *// Récupérer le premier fichier sélectionné*

*// Vérifier si un fichier a été sélectionné*

*if* (file) {

    uploadFile(file); *// Appeler la fonction pour télécharger le fichier*

  } *else* {

    console.error('No file selected!'); *// Afficher un message d'erreur si aucun fichier n'est sélectionné*

  }

});

1. **Sauvegarder les informations d’un étudiant sur Firestore et sa photo sur Storage**

*// Fonction pour inscrire un étudiant*

async function registerStudent(*studentInfo*, *file*) {

*try* {

*// 1. Upload de la photo dans Firebase Storage*

    const storageRef = ref(storage, 'photos/' + *file*.name); *// Référence du fichier photo*

    const snapshot = *await* uploadBytes(storageRef, *file*); *// Uploader la photo*

    const downloadURL = *await* getDownloadURL(storageRef); *// Récupérer l'URL de la photo*

*// 2. Stocker les informations de l'étudiant dans Firestore, y compris l'URL de la photo*

    const studentData = {

      name: *studentInfo*.name,      *// Nom de l'étudiant*

      email: *studentInfo*.email,    *// Email de l'étudiant*

      photoURL: downloadURL        *// URL de la photo*

    };

*// Ajouter l'étudiant dans la collection "students" de Firestore*

*await* addDoc(collection(db, "students"), studentData);

    console.log("Étudiant inscrit avec succès !");

  } *catch* (error) {

    console.error("Erreur lors de l'inscription de l'étudiant :", error);

  }

}

*// Exemple d'utilisation*

document.getElementById('registerButton').addEventListener('click', () => {

  const name = document.getElementById('nameInput').value; *// Récupérer le nom*

  const email = document.getElementById('emailInput').value; *// Récupérer l'email*

  const fileInput = document.getElementById('fileInput');

  const file = fileInput.files[0]; *// Récupérer la photo*

  const studentInfo = { name, email }; *// Créer l'objet des infos de l'étudiant*

*if* (file) {

    registerStudent(studentInfo, file); *// Appeler la fonction d'inscription*

  } *else* {

    console.error('Aucune photo sélectionnée !');

  }

});

1. **Récupérer les données de l’étudiant (Avec sa photo)**

*/ Fonction pour récupérer et afficher les informations d'un étudiant*

async function displayStudentInfo(*studentId*) {

*try* {

*// Obtenir les informations de l'étudiant à partir de Firestore*

    const studentDoc = *await* getDoc(doc(db, "students", *studentId*));

*if* (studentDoc.exists()) {

      const studentData = studentDoc.data(); *// Récupérer les données*

      console.log("Nom :", studentData.name);

      console.log("Email :", studentData.email);

      console.log("Photo URL :", studentData.photoURL);

*// Afficher la photo en utilisant l'URL*

      document.getElementById('studentPhoto').src = studentData.photoURL;

    } *else* {

      console.log("Aucun étudiant trouvé avec cet ID");

    }

  } *catch* (error) {

    console.error("Erreur lors de la récupération des infos de l'étudiant :", error);

  }

}

*// Exemple d'utilisation pour afficher les informations d'un étudiant*

displayStudentInfo("studentId123"); *// Remplacer par l'ID de l'étudiant*

1. **Configuration du switch pour changer le statut d’une classe**

async function getElements() {

*// Récupérer tous les documents dans la collection "classes"*

    const querySnapshot = *await* getDocs(collection(db, "classes"));

*// Sélectionner le conteneur où les cartes vont être ajoutées*

    const classListContainer = document.getElementById('class-list');

*// Parcourir chaque document récupéré*

    querySnapshot.forEach((*doc*) => {

*// Les données de chaque classe*

        const classeData = *doc*.data();

*// Créer la carte pour chaque classe*

        const classCard = `

        <div class="class-card">

          <img src="classe.jpg" alt="Image de la classe" class="class-image">

          <div class="class-info">

              <h2>${classeData.name}</h2>

              <p><strong>Capacité :</strong> ${classeData.capacity}</p>

              <p><strong>Équipements Disponibles :</strong>

              ${classeData.equipements.length === 0

                  ? "Aucun équipement disponible"

                  : classeData.equipements.map(*equipement* => `<span>${*equipement*}</span>`).join(', ')

              }

              </p>

              <p class="status"><strong>Statut d'Occupation :</strong> ${classeData.status\_occupation}</p>

              <p><strong>Horaires d'Occupation :</strong> 8h - 16h</p>

              <p><strong>Localisation :</strong> ${classeData.localisation}</p>

              <p><strong>Occupants :</strong> ${classeData.occupants === "" ? "Aucun occupant"

                  : classeData.occupants

              }</p>

          </div>

          <br>

          <!-- Switch pour changer le statut de la classe -->

          <div class="switch-container">

            <label class="switch">

              <input type="checkbox" class="status-toggle" data-class-id="${*doc*.id}" ${classeData.status\_occupation === 'Occupée' ? 'checked' : ''}/>

              <span class="slider round"></span>

            </label>

          </div>

        </div>

      `;

*// Insérer la carte dans le conteneur*

        classListContainer.innerHTML += classCard;

    });

*// Ajoute les écouteurs d'événements après avoir ajouté les cartes*

    addToggleListeners();

}

function addToggleListeners() {

    const statusToggles = document.querySelectorAll('.status-toggle');

    statusToggles.forEach((*toggle*) => {

*toggle*.addEventListener('change', async (*e*) => {

*// On récupère l'id de la classe correspondante*

            const classId = *e*.target.getAttribute('data-class-id');

*// On récupère l'élément qui affiche le statut*

            const statusText = *e*.target.closest('.class-card').querySelector('.status');

*if* (*e*.target.checked) {

                statusText.innerHTML = '<strong>Statut d\'Occupation :</strong> Occupée';

                statusText.style.color = 'red';

*// Appeler la fonction pour mettre à jour le statut dans Firestore*

*await* updateClassStatus(classId, "Occupée"); *// true = occupée*

            } *else* {

                statusText.innerHTML = '<strong>Statut d\'Occupation :</strong> Libre';

                statusText.style.color = 'green';

*// Appeler la fonction pour mettre à jour le statut dans Firestore*

*await* updateClassStatus(classId, 'Libre'); *// false = libre*

            }

        });

    });

}

*// Fonction pour modifier le statut de la classe*

async function updateClassStatus(*classId*, *newStatus*) {

*// Référence au document de la classe dans Firestore*

    const classRef = doc(db, "classes", *classId*); *// Remplace "classes" par le nom de ta collection*

*// Mettre à jour l'attribut `status\_occupation` avec le nouveau statut*

*await* updateDoc(classRef, {

        status\_occupation: *newStatus*

    });

}

*// Appeler la fonction pour afficher les éléments*

getElements();

1. **Connaitre l’utilisateur connecté et récupérer ses données**

async function getUserData(*uid*) {

*// Crée une requête pour rechercher l'utilisateur par son uid*

    const q = query(collection(db, "users"), where("uid", "==", *uid*));

    const querySnapshot = *await* getDocs(q);

*if* (!querySnapshot.empty) {

      querySnapshot.forEach((*doc*) => {

        const userData = *doc*.data();

        console.log('Données de l\'utilisateur:', userData);

*// Utilise les données de l'utilisateur selon tes besoins*

      });

    } *else* {

      console.log('Aucune donnée trouvée pour cet utilisateur');

    }

  }

  onAuthStateChanged(auth, (*user*) => {

*if* (*user*) {

      const uid = *user*.uid; *// Obtenir le uid de l'utilisateur connecté*

      getUserData(uid); *// Appeler la fonction pour obtenir les données*

    } *else* {

      console.log("L'utilisateur n'est pas connecté");

    }

  });

1. **Générer un Code QR**

*import* { donneeUtilisateur } *from* "./index.js";

document.getElementById('generateQRCode').addEventListener('click', function() {

*if* (donneeUtilisateur && typeof donneeUtilisateur.a\_jour !== 'undefined') {

*// Afficher le QR code et l'overlay*

        document.getElementById("qrcode-container").style.display = 'block';

        document.getElementById("overlay").style.display = 'block';

*// Générer le QR code avec des informations*

        var qrcode = new QRCode(document.getElementById("qrcode"), {

            text: donneeUtilisateur.a\_jour ? "Autorisé" : "Pas autorisé",

            width: 300,

            height: 300,

            colorDark: "#000000",  *// Couleur unie (noir)*

            colorLight: "#ffffff",  *// Fond blanc*

            correctLevel: QRCode.CorrectLevel.H *// Niveau de correction des erreurs*

        });

    } *else* {

        alert("Les informations de l'utilisateur ne sont pas disponibles ou complètes.");

    }

});

*// Événement pour fermer le QR code*

document.getElementById('closeQRCode').addEventListener('click', function() {

*// Masquer le QR code, l'overlay et le bouton Fermer*

    document.getElementById("qrcode-container").style.display = 'none';

    document.getElementById("qrcode").innerHTML = '';

    document.getElementById("overlay").style.display = 'none';

});

1. **Gestion du tableau du comptable**

*// Fonction pour récupérer les données et mettre à jour le tableau*

async function loadStudents() {

    const tableBody = document.querySelector("tbody"); *// Sélectionner le corps du tableau*

*// Récupérer les documents de la collection 'users'*

    const q = query(collection(db, "users"), where("role", "==", "etudiant"));

    const querySnapshot = *await* getDocs(q);

*if* (!querySnapshot.empty) {

        querySnapshot.forEach((*doc*) => {

*// Récupérer les données de chaque étudiant*

            const student = *doc*.data();

*// Créer une nouvelle ligne de tableau*

            const row = document.createElement("tr");

*// Ajouter les cellules de données*

            row.innerHTML = `

                <td>${*doc*.id}</td>

                <td>${student.pseudoOk}</td>

                <td>${student.classe}</td>

                <td class="status ${student.a\_jour ? 'up-to-date' : 'not-up-to-date'}">

                    ${student.a\_jour ? 'À jour' : 'Non à jour'}

                </td>

                <td class="action-buttons">

                    <button class="icon-btn up-to-date" onclick="markAsUpToDate(this, '${*doc*.id}')">

                        <i class="fas fa-check-circle"></i>

                    </button>

                </td>

                <td class="action-buttons">

                    <button class="icon-btn not-up-to-date" onclick="markAsNotUpToDate(this, '${*doc*.id}')">

                        <i class="fas fa-times-circle"></i>

                    </button>

                </td>

                <td>

                <img src="${student.photoURLOk}">

                </td>

            `;

*// Ajouter la nouvelle ligne dans le tableau*

            tableBody.appendChild(row);

        });

    }

}

*// Appeler la fonction pour charger les étudiants quand la page est chargée*

window.onload = loadStudents;

*// Fonction pour marquer un étudiant comme "À jour"*

window.markAsUpToDate = function(*button*, *studentId*) {

*// 1. Trouver la ligne du tableau contenant le bouton cliqué*

    const row = *button*.closest('tr');

*// `button.closest('tr')` cherche le premier élément parent `<tr>` de l'élément `button`.*

*// Cela nous permet de savoir dans quelle ligne du tableau se trouve le bouton cliqué.*

*// 2. Trouver la cellule qui affiche le statut de l'étudiant*

    const statusCell = row.querySelector('.status');

*// `row.querySelector('.status')` sélectionne la première cellule (élément `<td>`)*

*// de la ligne qui a la classe `status`, où nous allons afficher le statut de l'étudiant.*

*// 3. Mettre à jour le texte du statut pour indiquer que l'étudiant est à jour*

    statusCell.textContent = 'À jour';

*// `textContent` met à jour le contenu textuel de `statusCell` pour afficher "À jour".*

*// 4. Modifier les classes CSS pour le style*

    statusCell.classList.remove('not-up-to-date'); *// Retire la classe 'not-up-to-date'*

    statusCell.classList.add('up-to-date'); *// Ajoute la classe 'up-to-date'*

*// `classList.remove` et `classList.add` permettent de gérer les classes CSS de l'élément.*

*// Cela permet de changer l'apparence de la cellule en fonction du statut.*

*// 5. Mettre à jour Firestore pour marquer l'étudiant comme à jour*

    const docRef = doc(db, "users", *studentId*);

*// `doc(db, "users", studentId)` crée une référence au document de l'étudiant dans la collection 'users'*

*// en utilisant son identifiant `studentId`.*

    updateDoc(docRef, { a\_jour: true }) *// Met à jour le champ 'a\_jour' à `true`*

        .then(() => {

            console.log("Étudiant marqué comme à jour"); *// Message de confirmation dans la console*

        })

        .catch((*error*) => {

            console.error("Erreur lors de la mise à jour : ", error); *// Affiche une erreur en cas de problème*

        });

};

*// Fonction pour marquer un étudiant comme "Non à jour"*

window.markAsNotUpToDate = function(*button*, *studentId*) {

*// 1. Trouver la ligne du tableau contenant le bouton cliqué*

    const row = *button*.closest('tr');

*// Comme précédemment, cela nous permet de trouver la ligne du tableau.*

*// 2. Trouver la cellule qui affiche le statut de l'étudiant*

    const statusCell = row.querySelector('.status');

*// 3. Mettre à jour le texte du statut pour indiquer que l'étudiant n'est pas à jour*

    statusCell.textContent = 'Non à jour';

*// 4. Modifier les classes CSS pour le style*

    statusCell.classList.remove('up-to-date'); *// Retire la classe 'up-to-date'*

    statusCell.classList.add('not-up-to-date'); *// Ajoute la classe 'not-up-to-date'*

*// 5. Mettre à jour Firestore pour marquer l'étudiant comme non à jour*

    const docRef = doc(db, "users", *studentId*);

*// Comme pour la fonction précédente, cela crée une référence au document de l'étudiant.*

    updateDoc(docRef, { a\_jour: false }) *// Met à jour le champ 'a\_jour' à `false`*

        .then(() => {

            console.log("Étudiant marqué comme non à jour"); *// Message de confirmation dans la console*

        })

        .catch((*error*) => {

            console.error("Erreur lors de la mise à jour : ", *error*); *// Affiche une erreur en cas de problème*

        });

};