ECOLE MAROCAINE DES SCIENCES DE L'INGENIEUR
Membre de HONORIS UNITED UNIVERSITIES

Gestion d'école

Meryem Fayd - Maryem Radouane - Soukaina Bahij

Sommaire:

Objectif Du Projet

- Développer un site web de gestion d’école
- Faciliter la gestion administrative
- Centraliser toutes les informations scolaires

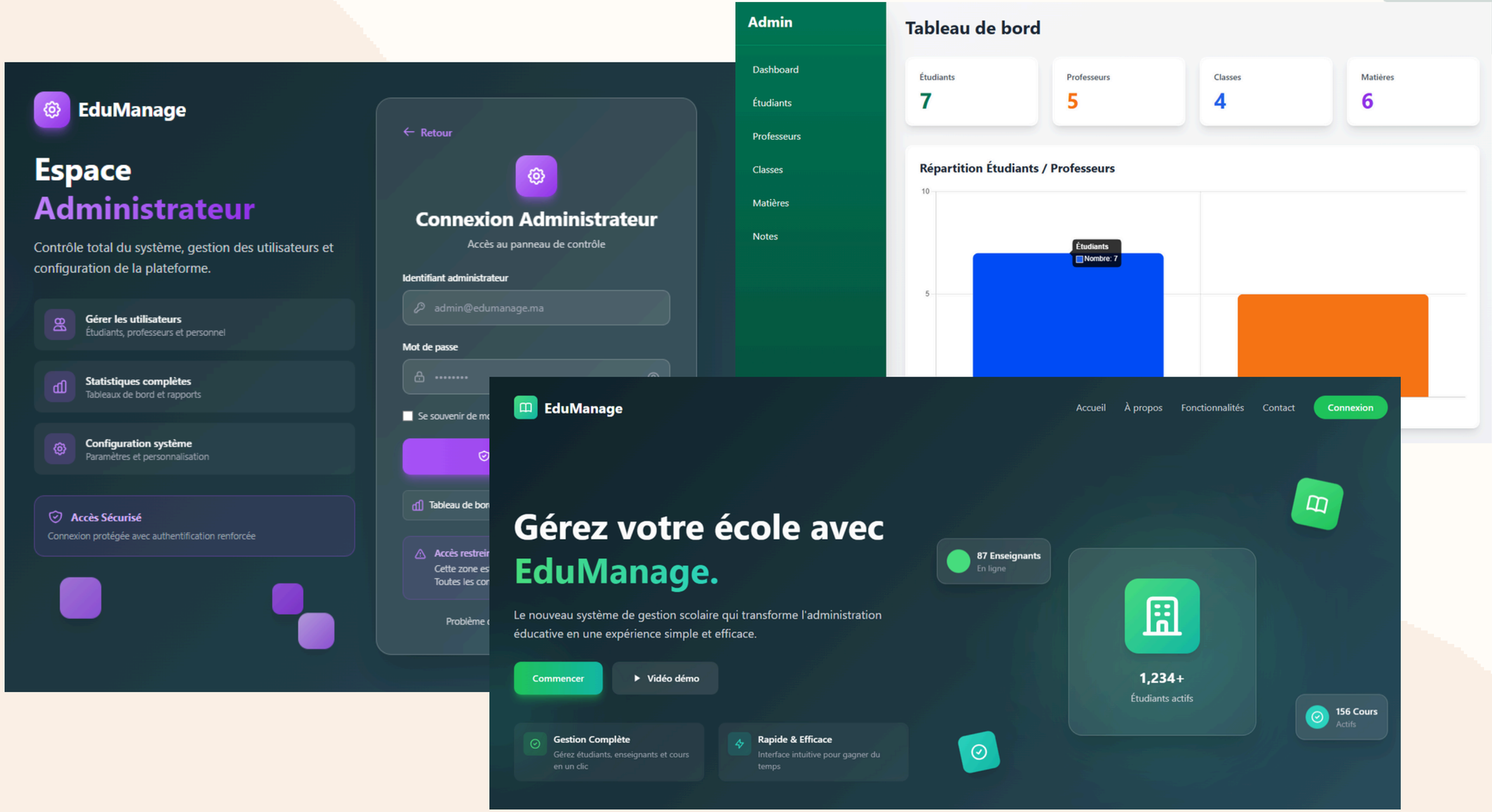
Problématique :

- Gestion manuelle (papier / Excel)
- Perte de données
- Difficulté d’accès à l’information
- Manque d’organisation

Solution :

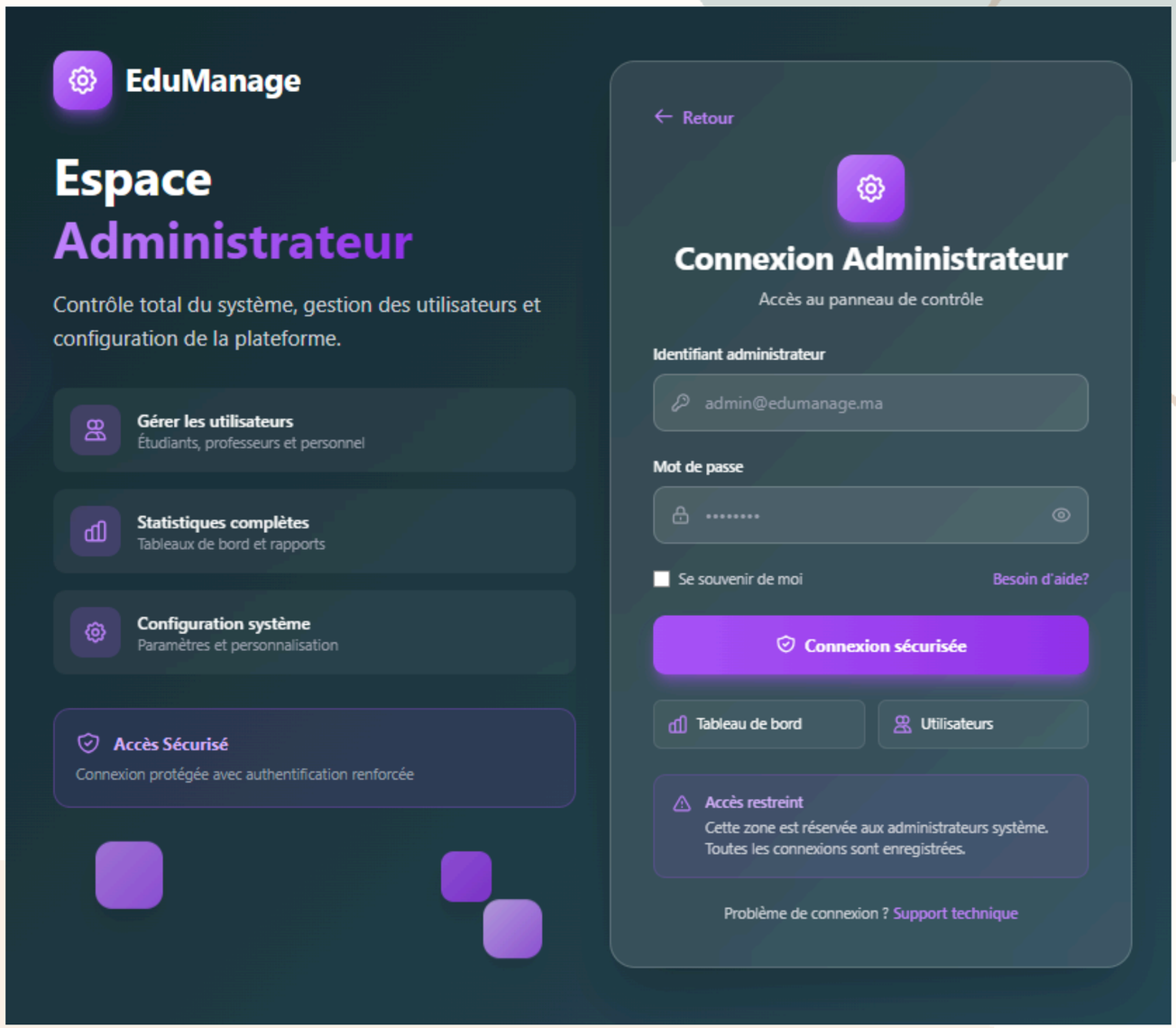
- Site web centralisé
- Accès rapide et sécurisé
- Automatisation des tâches
- Meilleure organisation

Structure et présentation des pages



LOGIN :

- Ce formulaire permet à l'administrateur de se connecter au système via un identifiant et un mot de passe. Il intègre une validation visuelle avec des messages d'erreur pour guider l'utilisateur en cas de saisie incorrecte. Après une connexion réussie, l'administrateur accède au tableau de bord. Des éléments comme "Se souvenir de moi" et "Accès restreint" renforcent la sécurité et la fiabilité de la page.



« Identifiant administrateur »

```
<!-- Admin ID -->
<div>
    <label class="text-white text-sm font-semibold mb-2 block">Identifiant administrateur</label>
    <div class="relative">
        <div class="absolute inset-y-0 left-0 pl-4 flex items-center pointer-events-none">
            <svg class="w-5 h-5 text-gray-400" fill="none" stroke="currentColor" viewBox="0 0 24 24">
                <path stroke-linecap="round" stroke-linejoin="round" stroke-width="2" d="M15 7a2 2 0 012 2m4 0
            </svg>
        </div>
        <input id="email"
            type="text"
            placeholder="admin@edumanage.ma"
            class="w-full pl-12 pr-4 py-3.5 bg-white/10 border border-white/20 rounded-xl text-white placehold
        >
    </div>
    <div id="emailError" class="error-message">
        <svg class="w-4 h-4 flex-shrink-0" fill="currentColor" viewBox="0 0 20 20">
            <path fill-rule="evenodd" d="M10 18a8 8 0 100-16 8 8 0 000 16zM8.707 7.293a1 1 0 00-1.414 1.414L8.
        </svg>
        <span id="emailErrorText"></span>
    </div>
</div>
```

Champ utilisé pour saisir l’identifiant de l’administrateur afin de vérifier son accès au système.

«Mot de passe »

```
<!-- Password -->
<div>
    <label class="text-white text-sm font-semibold mb-2 block">Mot de passe</label>
    <div class="relative">
        <div class="absolute inset-y-0 left-0 pl-4 flex items-center pointer-events-none">
            <svg class="w-5 h-5 text-gray-400" fill="none" stroke="currentColor" viewBox="0 0 24 24">
                <path stroke-linecap="round" stroke-linejoin="round" stroke-width="2" d="M12 15v2m-6 4h12a2 2
            </svg>
        </div>
        <input
            type="password"
            id="password"
            placeholder="••••••••"
            class="w-full pl-12 pr-12 py-3.5 bg-white/10 border border-white/20 rounded-xl text-white placehol
        >
        <button type="button" onclick="togglePassword()" class="absolute inset-y-0 right-0 pr-4 flex items-cen
            <svg id="eyeIcon" class="w-5 h-5" fill="none" stroke="currentColor" viewBox="0 0 24 24">
                <path stroke-linecap="round" stroke-linejoin="round" stroke-width="2" d="M15 12a3 3 0 11-6 0 3
                <path stroke-linecap="round" stroke-linejoin="round" stroke-width="2" d="M2.458 12C3.732 7.943
            </svg>
        </button>
    </div>
    <div id="passwordError" class="error-message">
        <svg class="w-4 h-4 flex-shrink-0" fill="currentColor" viewBox="0 0 20 20">
            <path fill-rule="evenodd" d="M10 18a8 8 0 100-16 8 8 0 000 16zM8.707 7.293a1 1 0 00-1.414 1.414L8.
        </svg>
        <span id="passwordErrorText"></span>
    </div>
</div>
```

Ce champ permet de saisir le mot de passe de manière sécurisée (texte masqué). Il garantit que seul l'administrateur autorisé peut accéder au système.

« Bouton Connexion »

```
<!-- Submit Button -->
<button
    type="submit"
    class="w-full py-4 bg-gradient-to-r from-purple-500 to-purple
>
    <div class="flex items-center justify-center gap-2">
        <svg class="w-5 h-5" fill="none" stroke="currentColor" vi
            <path stroke-linecap="round" stroke-linejoin="round"
        </svg>
        <span>Connexion sécurisée</span>
    </div>
</button>
```

Ce bouton lance le processus d'authentification. Après validation des champs, il permet l'accès au tableau de bord administrateur.

Message « Accès restreint »

```
<!-- Security Warning -->
<div class="bg-purple-500/10 border border-purple-500/20 rounded-xl p-4 mt-4">
    <div class="flex items-start gap-3">
        <svg class="w-5 h-5 text-purple-400 flex-shrink-0 mt-0.5" fill="none" stroke="currentColor"
        viewBox="0 0 24 24">
            <path stroke-linecap="round" stroke-linejoin="round" stroke-width="2" d="M12 9v2m0 4h.
            01m-6.938 4h13.856c1.54 0 2.502-1.667 1.732-3L13.732 4c-.77-1.333-2.694-1.333-3.464 0L3.
            34 16c-.77 1.333.192 3 1.732 3z"/>
        </svg>
        <div>
            <p class="text-purple-300 text-sm font-semibold mb-1">Accès restreint</p>
            <p class="text-gray-300 text-sm">Cette zone est réservée aux administrateurs système.
            Toutes les connexions sont enregistrées.</p>
        </div>
    </div>
</div>
```

Ce message informe que l'espace est réservé uniquement aux administrateurs. Il renforce la sécurité et la crédibilité de la page de connexion.

« Fonctions de gestion des erreurs »

```
// Function to show error message
Windsurf: Refactor | Explain | ×
function showError(inputId, message) {
    const input = document.getElementById(inputId);
    const errorDiv = document.getElementById(inputId + "Error");
    const errorText = document.getElementById(inputId + "ErrorText");

    input.classList.add("input-error");
    errorDiv.classList.add("show");
    errorText.textContent = message;
}

// Function to hide error message
Windsurf: Refactor | Explain | ×
function hideError(inputId) {
    const input = document.getElementById(inputId);
    const errorDiv = document.getElementById(inputId + "Error");

    input.classList.remove("input-error");
    errorDiv.classList.remove("show");
}
```

Ces fonctions affichent et masquent les messages d'erreur. Elles permettent de signaler visuellement les champs incorrects et d'améliorer l'expérience utilisateur.

« Suppression des erreurs lors de la saisie »

```
// Hide errors when user types
document.getElementById("email").addEventListener("input", function() {
    hideError("email");
});

document.getElementById("password").addEventListener("input", function() {
    hideError("password");
});
```

Les messages d’erreur disparaissent automatiquement lorsque l’utilisateur commence à corriger sa saisie.

« Validation du formulaire »

```
// Form submission
const form = document.getElementById("loginForm");

form.addEventListener("submit", function (e) {
    e.preventDefault();

    hideError("email");
    hideError("password");

    const email = document.getElementById("email").value.trim();
    const password = document.getElementById("password").value;

    const adminEmail = "admin@edumanage.ma";
    const adminPassword = "12345678";

    let hasError = false;

    // Validate email
    if (email === "") {
        showError("email", "L'identifiant administrateur est requis");
        hasError = true;
    }

    // Validate password
    if (password === "") {
        showError("password", "Le mot de passe est requis");
        hasError = true;
    }

    // If there are validation errors, stop here
    if (hasError) {
        return;
    }
```

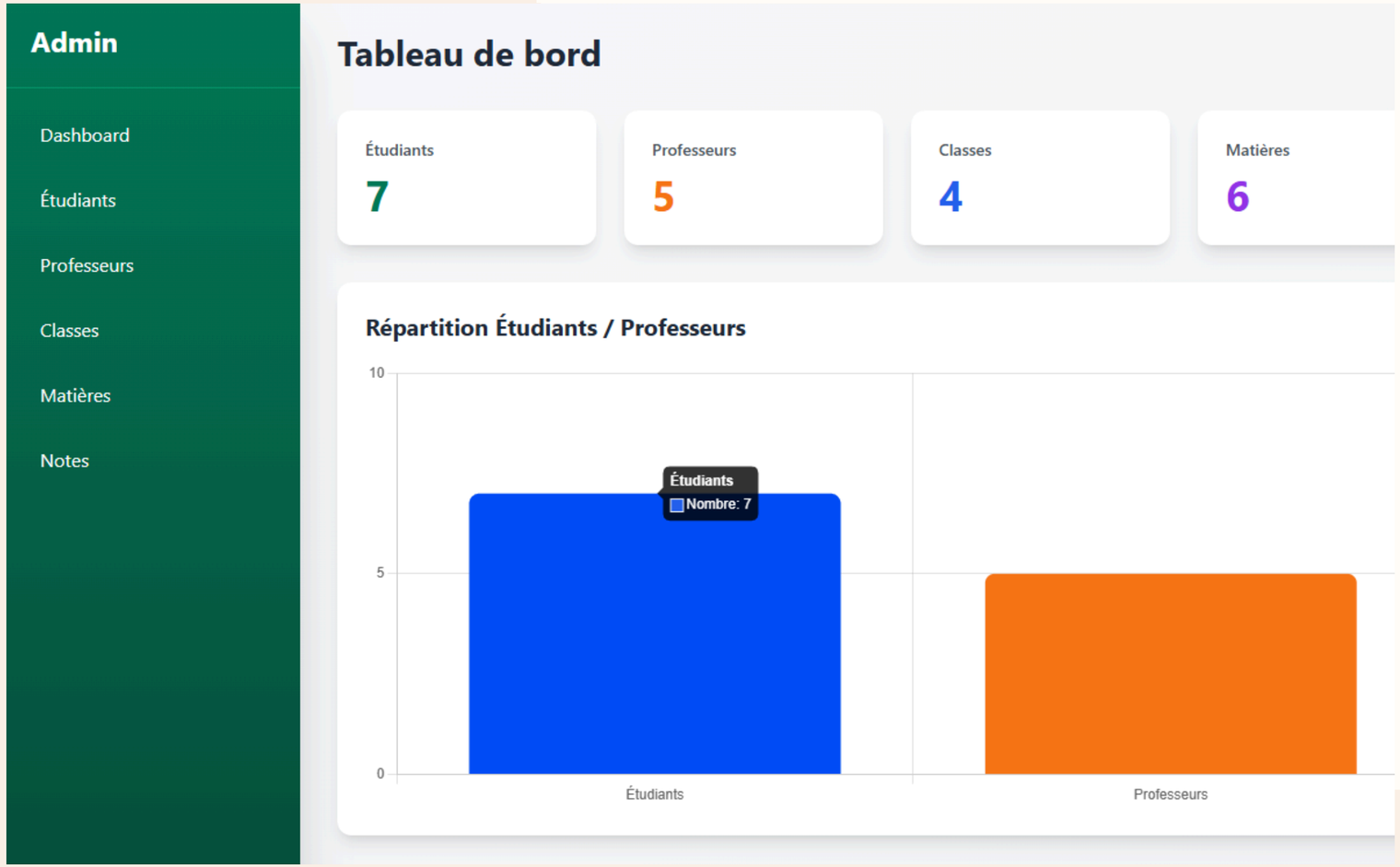
Cette partie empêche l'envoi automatique du formulaire, récupère les données saisies et vérifie que les champs identifiant et mot de passe ne sont pas vides. Des messages d'erreur sont affichés si nécessaire pour guider l'administrateur.

« Authentification et erreurs »

```
    if (email === adminEmail && password === adminPassword) {
        window.location.href = "dashboard_A.html";
    } else {
        if (email !== adminEmail) {
            showError("email", "Identifiant administrateur incorrect");
        }
        if (password !== adminPassword) {
            showError("password", "Mot de passe incorrect");
        }
    }
});
```

Le script compare les identifiants saisis avec ceux de l'administrateur. Si corrects, l'accès au tableau de bord est accordé. Sinon, des messages d'erreur indiquent si l'identifiant ou le mot de passe est incorrect.

DASHBOARD:



Ce tableau de board permet à l'administrateur de visualiser toutes les données du site sous forme de tableaux, d'ajouter, modifier ou supprimer des éléments facilement, et de naviguer entre les différentes sections (étudiants, professeurs, classes...). Il offre également des statistiques et un aperçu rapide de l'activité pour faciliter la gestion du site.

« Dashboard »

```html
<!-- Dashboard -->
<div id="dashboard" class="p-8">
    <h2 class="text-3xl font-bold text-gray-800 mb-8">Tableau de bord</h2>

    <div class="grid grid-cols-1 md:grid-cols-4 gap-6 mb-8">
        <div class="bg-white p-6 rounded-xl shadow-lg hover:shadow-xl transition">
            <h3 class="text-gray-600 text-sm font-semibold mb-2">Étudiants</h3>
            <p class="text-4xl font-bold text-emerald-700" id="count-etudiants">0</p>
        </div>
        <div class="bg-white p-6 rounded-xl shadow-lg hover:shadow-xl transition">
            <h3 class="text-gray-600 text-sm font-semibold mb-2">Professeurs</h3>
            <p class="text-4xl font-bold text-orange-500" id="count-professeurs">0</p>
        </div>
        <div class="bg-white p-6 rounded-xl shadow-lg hover:shadow-xl transition">
            <h3 class="text-gray-600 text-sm font-semibold mb-2">Classes</h3>
            <p class="text-4xl font-bold text-blue-600" id="count-classes">0</p>
        </div>
        <div class="bg-white p-6 rounded-xl shadow-lg hover:shadow-xl transition">
            <h3 class="text-gray-600 text-sm font-semibold mb-2">Matières</h3>
            <p class="text-4xl font-bold text-purple-600" id="count-matieres">0</p>
        </div>
    </div>

    <div class="bg-white p-6 rounded-xl shadow-lg">
        <h3 class="text-xl font-bold text-gray-800 mb-4">Répartition Étudiants / Professeurs</h3>
        <canvas id="chartCanvas" class="max-h-96"></canvas>
    </div>
</div>
```

C’est la page qui donne un aperçu rapide et visuel de l’état de l’école.

- **Titre :** affiche "Tableau de bord".
- **Cartes de statistiques :** montrent le nombre total d’étudiants, professeurs, classes et matières.
- **Graphique :** un diagramme en barres qui compare le nombre d’étudiants et de professeurs.

« Création du graphique (Chart.js) »

```
// GRAPHIQUE
Windsurf: Refactor | Explain | ×
function initChart() {
    const ctx = document.getElementById('chartCanvas');
    if (!ctx) return;

    myChart = new Chart(ctx.getContext('2d'), {
        type: 'bar',
        data: {
            labels: ['Étudiants', 'Professeurs'],
            datasets: [{
                label: 'Nombre',
                data: [0, 0],
                backgroundColor: ['#2563eb', '#f97316'],
                borderWidth: 0,
                borderRadius: 8
            }]
        },
        options: {
            responsive: true,
            maintainAspectRatio: true,
            scales: {
                y: {
                    beginAtZero: true,
                    ticks: {
                        stepSize: 5
                    }
                }
            },
            plugins: {
                legend: {
                    display: false
                }
```

- On utilise Chart.js pour créer un graphique simple en barres.
- Les barres représentent le nombre d'étudiants et de professeurs.
- Les valeurs sont initialisées à 0 et seront mises à jour après.

« Initialisation du dashboard »

```
// INITIALISATION

document.addEventListener('DOMContentLoaded', function() {
    initChart();
    updateCounts();
});
```

Quand la page est complètement chargée (DOMContentLoaded), on lance les fonctions pour afficher le graphique et mettre à jour les chiffres visibles sur le dashboard.

« Mise à jour des compteurs et graphique »

```
// MISE À JOUR DES COMPTEURS

Windsurf: Refactor | Explain | Generate JSDoc | ×
function updateCounts() {
    const countEtu = document.getElementById('count-etudiants');
    const countProf = document.getElementById('count-professeurs');
    const countClasses = document.getElementById('count-classes');
    const countMat = document.getElementById('count-matieres');

    if (countEtu) countEtu.textContent = db.etudiants.length;
    if (countProf) countProf.textContent = db.professeurs.length;
    if (countClasses) countClasses.textContent = db.classes.length;
    if (countMat) countMat.textContent = db.matieres.length;

    if (myChart) {
        myChart.data.datasets[0].data = [db.etudiants.length, db.professeurs.length];
        myChart.update();
    }
}
```

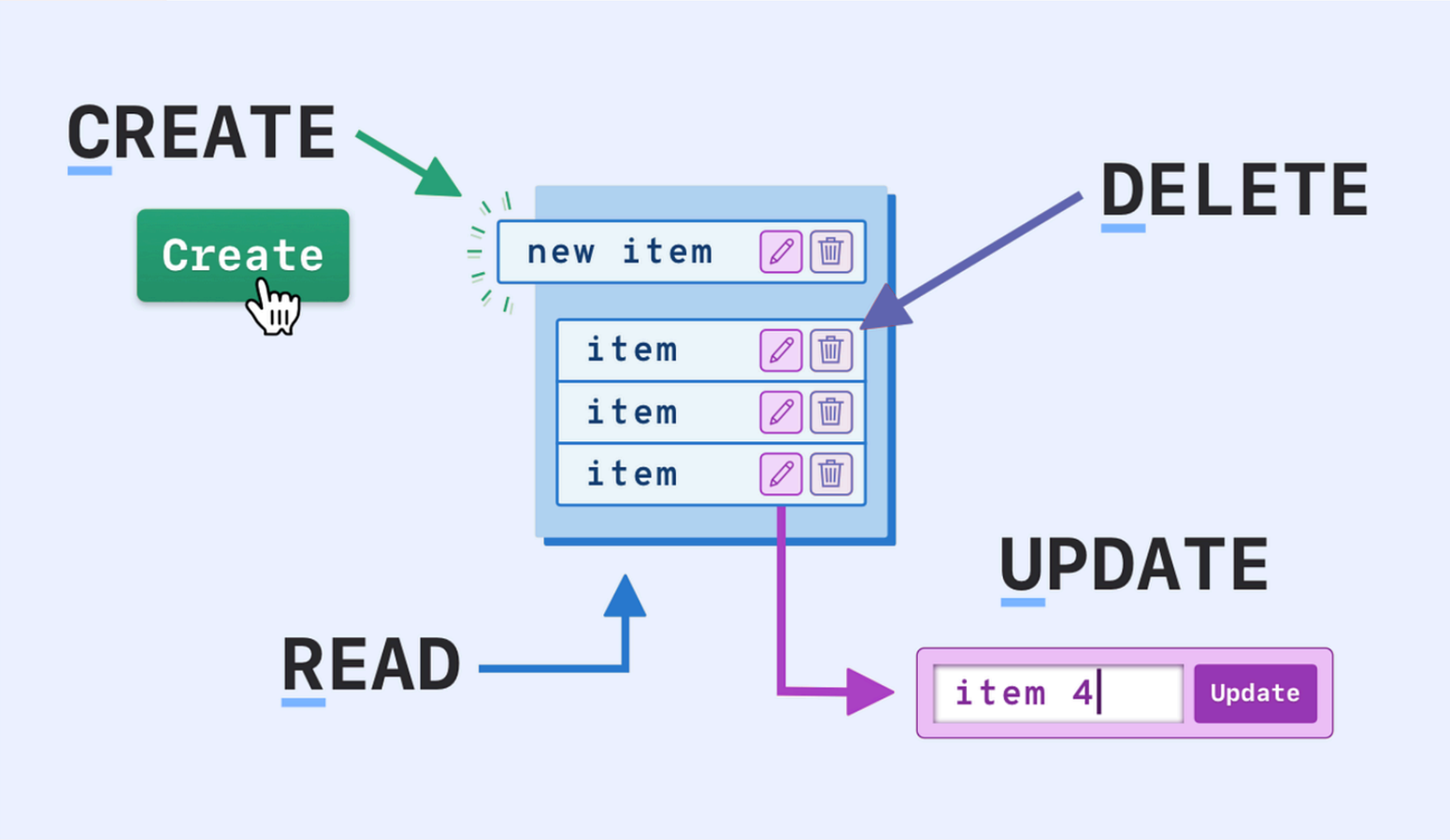
On récupère les éléments HTML qui affichent les chiffres.
On met à jour leur contenu avec le nombre réel d'éléments dans la base de données.
Ensuite, on met à jour le graphique pour refléter ces valeurs.

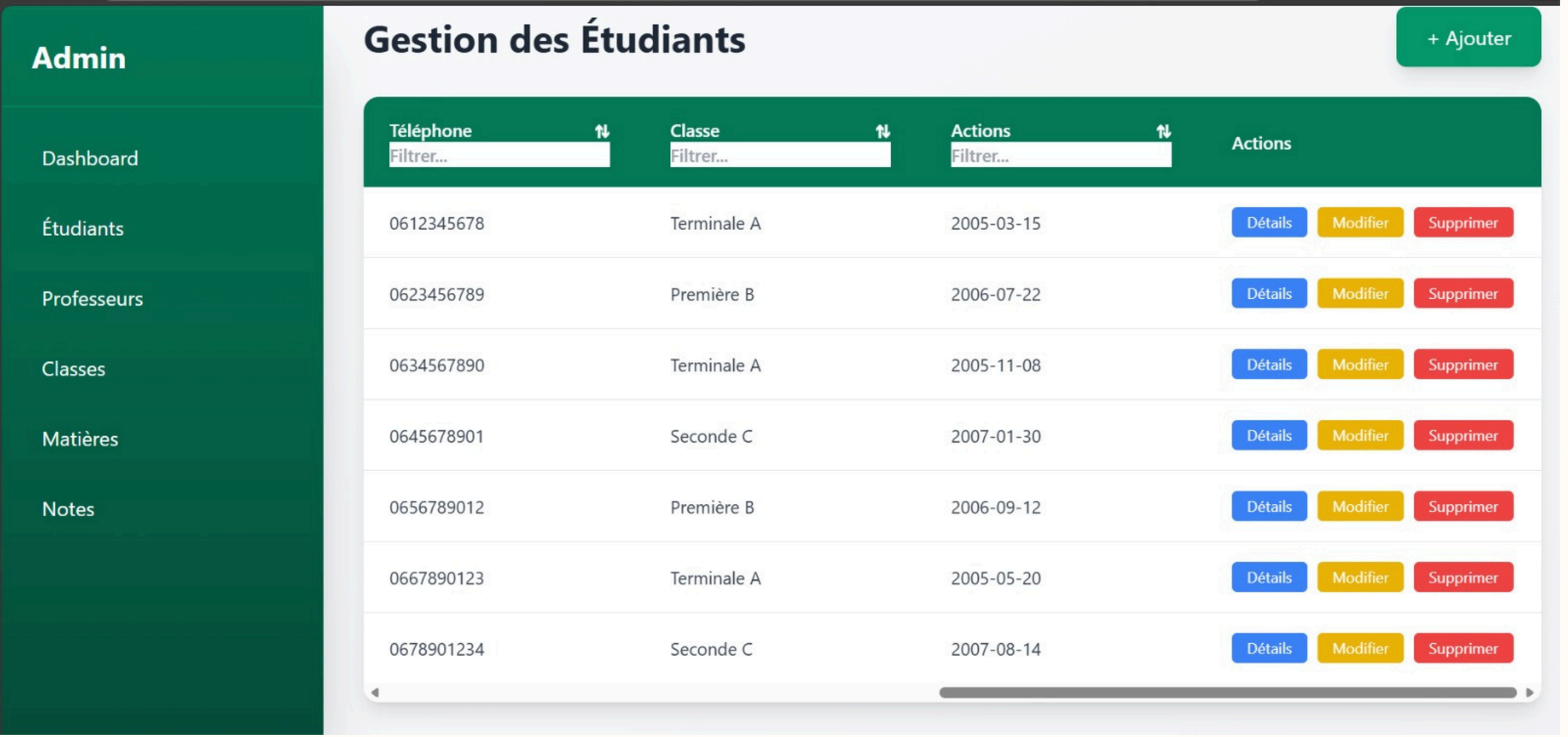
« Navigation vers le dashboard »

```
// NAVIGATION

Windsurf: Refactor | Explain | Generate JSDoc | X
function showDashboard() {
    document.getElementById('dashboard').classList.remove('hidden');
    document.getElementById('crud-section').classList.add('hidden');
    updateMenuActive(event.target);
}

Windsurf: Refactor | Explain | Generate JSDoc | X
function updateMenuActive(element) {
    document.querySelectorAll('.menu-item').forEach(item => {
        item.classList.remove('active');
    });
    if (element) element.classList.add('active');
}
```

- **showDashboard() : affiche le dashboard et cache les autres section**
- **updateMenuActive() : change la couleur ou style du menu pour montrer quelle section est active.**

CRUD:


CREATE
Create
new item
item
item
item
DELETE
UPDATE
item 4
Update
READ



Les fonctionnalités CRUD (Create, Read, Update, Delete) permettent de gérer toutes les données du site. L'administrateur peut ajouter de nouveaux éléments via un formulaire dynamique, consulter les données sous forme de tableaux clairs, modifier les informations existantes et supprimer des éléments après confirmation. Ces opérations assurent une gestion complète, simple et efficace des différentes sections du projet (étudiants, professeurs, classes, etc.).

```
<!-- Section CRUD -->
<div id="crud-section" class="hidden p-8">
    <!-- Vue Tableau -->
    <div id="table-view">
        <div class="flex justify-between items-center mb-6">
            <h2 class="text-3xl font-bold text-gray-800" id="section-title">Gestion</h2>
            <button onclick="showFormSection('add'); return false;" class="bg-emerald-600 te
                + Ajouter
            </button>
        </div>
        <div class="bg-white rounded-xl shadow-lg overflow-hidden">
            <div class="overflow-x-auto">
                <table class="w-full" id="data-table">
                    <thead class="bg-emerald-700 tex
                        <tr id="table-header"></tr>
                    </thead>
                    <tbody id="table-body" class="di
                </table>
```

« Vue Tableau (Read) »

Affiche la liste des éléments dans un tableau et permet d'ouvrir le formulaire pour en ajouter un nouveau.

```
- Vue Formulaire -->
v id="form-view" class="hidden">
<div class="flex justify-between items-center mb-6">
    <h2 class="text-3xl font-bold text-gray-800" id="form-title">Ajouter</h2>
    <button onclick="hideFormSection(); return false;" class="text-gray-500 hover:text-gray-7
        <svg class="w-8 h-8" fill="none" stroke="currentColor" viewBox="0 0 24 24">
            <path stroke-linecap="round" stroke-linejoin="round" stroke-width="2" d="M6 18L18
        </svg>
    </button>
</div>

<div class="bg-white rounded-xl shadow-lg p-8">
    <form id="crud-form" class="space-y-4" onsubmit="handleFormSubmit(event); return false;">
        <div id="form-fields" class="grid grid-cols-1 md:grid-cols-2 gap-4"></div>
        <div class="flex gap-4 pt-4">
            <button type="submit" class="flex-1 bg-emerald-600 text-white py-3 rounded-lg hov
                Enregistrer
            </button>
            <button type="button" onclick="hideFormSection(); return false;" class="flex-1 bg
                Annuler
```

« Vue Formulaire (Create / Update) »

Permet d'ajouter ou de modifier un élément avec des champs dynamiques et des boutons Enregistrer/Annuler.

« Modals (Détails et Confirmation) »

```
<!-- Vue Formulaire -->
<div id="form-view" class="hidden">
    <div class="flex justify-between items-center mb-6">
        <h2 class="text-3xl font-bold text-gray-800" id="form-title">Ajouter</h2>
        <button onclick="hideFormSection(); return false;" class="text-gray-500 hover:text-gray-700">
            <svg class="w-8 h-8" fill="none" stroke="currentColor" viewBox="0 0 24 24">
                <path stroke-linecap="round" stroke-linejoin="round" stroke-width="2" d="M6 18L18 6M6
            </svg>
        </button>
    </div>

    <div class="bg-white rounded-xl shadow-lg p-8">
        <form id="crud-form" class="space-y-4" onsubmit="handleFormSubmit(event); return false;">
            <div id="form-fields" class="grid grid-cols-1 md:grid-cols-2 gap-4"></div>
            <div class="flex gap-4 pt-4">
                <button type="submit" class="flex-1 bg-emerald-600 text-white py-3 rounded-lg hover:b
                    Enregistrer
                </button>
                <button type="button" onclick="hideFormSection(); return false;" class="flex-1 bg-gra
                    Annuler
                </button>
```

- **Details Modal : affiche toutes les informations d'un élément quand on clique sur "Détails".**
- **Confirm Modal : demande confirmation avant de supprimer un élément.**

« Variables et configurations »

```
// CONFIGURATION DES SECTIONS

const config = {
    etudiants: {
        title: 'Gestion des Étudiants',
        fields: ['nom', 'prenom', 'email', 'telephone', 'classe', 'dateNaissance'],
        headers: ['ID', 'Nom', 'Prénom', 'Email', 'Téléphone', 'Classe', 'Actions']
    },
    professeurs: {
        title: 'Gestion des Professeurs',
        fields: ['nom', 'prenom', 'email', 'telephone', 'specialite'],
        headers: ['ID', 'Nom', 'Prénom', 'Email', 'Téléphone', 'Spécialité', 'Actions']
    },
    classes: {
        title: 'Gestion des Classes',
        fields: ['nom', 'niveau', 'capacite', 'professeurPrincipal'],
        headers: ['ID', 'Nom', 'Niveau', 'Capacité', 'Professeur Principal', 'Actions']
    },
    matieres: {
        title: 'Gestion des Matières',
        fields: ['nom', 'code', 'coefficient', 'description'],
        headers: ['ID', 'Nom', 'Code', 'Coefficient', 'Description', 'Actions']
    },
    notes: {
        title: 'Gestion des Notes',
        fields: ['etudiant', 'matiere', 'note', 'date', 'type'],
        headers: ['ID', 'Étudiant', 'Matière', 'Note', 'Date', 'Type', 'Actions']
    }
};
```

```
// VARIABLES GLOBALES

let currentSection = '';
let currentItem = null;
let deleteId = null;
let myChart = null;
let formMode = 'add';
let sortColumn = null;
let sortDirection = 'asc';
let filters = {};
```

Définit les champs et colonnes de chaque type de données et stocke l'état courant (section, élément et mode).

« Affichage section et tableau »

```
Windsurf: Refactor | Explain | Generate JSDoc | ×
function showSection(section) {
    currentSection = section;
    sortColumn = null;
    sortDirection = 'asc';
    filters = {};

    document.getElementById('dashboard').classList.add('hidden');
    document.getElementById('crud-section').classList.remove('hidden');
    document.getElementById('section-title').textContent = config[section].title;

    // Afficher le tableau et masquer le formulaire
    document.getElementById('table-view').classList.remove('hidden');
    document.getElementById('form-view').classList.add('hidden');

    renderTable();
    updateMenuActive(event.target);
}
```

Affiche la section CRUD et cache le dashboard principal.

« Génération du tableau »

```
function renderTable() {
    if (!currentSection) return;

    const conf = config[currentSection];
    const filteredData = applyFilters();

    // Headers avec tri et filtres
    const headerRow = document.getElementById('table-header');
    if (headerRow) {
        headerRow.innerHTML = '';

        // En-tête ID
        const thId = document.createElement('th');
        thId.className = 'px-6 py-4 text-left text-sm font-semibold sortable-header';
        thId.innerHTML = `
            <div class="header-content">
                <div class="flex items-center justify-between" onclick="sortTable('id', 0)"
                    <span>ID</span>
                    <span class="sort-icon ${sortColumn === 'id' ? 'active' : ''}">
                        ${sortColumn === 'id' ? (sortDirection === 'asc' ? '▲' : '▼') : '↕'
                    </span>
                </div>
                <input type="text" class="filter-input" placeholder="Filtrer..."
                        oninput="filterTable('id', this.value)" value="${filters['id'] || ''
            </div>
        `;
        headerRow.appendChild(thId);
```

Crée dynamiquement les lignes du tableau avec les boutons Détails, Modifier et Supprimer.

« Formulaire – afficher et cacher »

```
function showFormSection(mode, id = null) {
    const tableView = document.getElementById('table-view');
    const formView = document.getElementById('form-view');
    const formTitle = document.getElementById('form-title');
    const fields = document.getElementById('form-fields');

    formMode = mode;
    currentItem = id;

    if (formTitle) {
        formTitle.textContent = mode === 'add' ? 'Ajouter un nouv
    }

                    = config[currentSection];
                    = id ? db[currentSection].find(i => i.id === id)
```

Windsurf: Refactor | Explain | Generate JSDoc | ×

```
function hideFormSection() {
    const tableView = document.getElementById('table-view');
    const formView = document.getElementById('form-view');
    const form = document.getElementById('crud-form');

    // Afficher le tableau et masquer le formulaire
    if (tableView) tableView.classList.remove('hidden');
    if (formView) formView.classList.add('hidden');
    if (form) form.reset();

    currentItem = null;
```

Ouvre ou ferme le formulaire et pré-remplit les champs si on modifie un élément.

« Soumission formulaire »

```
function handleFormSubmit(event) {
    event.preventDefault();
    event.stopPropagation();

    const form = document.getElementById('crud-form');
    const formData = new FormData(form);
    const data = {};
    formData.forEach((value, key) => data[key] = value);

    if (formMode === 'add') {
        const maxId = db[currentSection].length > 0
            ? Math.max(...db[currentSection].map(i => i.id))
            : 0;
        data.id = maxId + 1;
        db[currentSection].push(data);
    } else {
        const index = db[currentSection].findIndex(i => i.id === currentItem);
        if (index !== -1) {
            db[currentSection][index] = { ...db[currentSection][index], ...data };
        }
    }

    hideFormSection();
    updateCounts();
    renderTable();

    return false;
}
```

Ajoute ou met à jour un élément dans la base et rafraîchit le tableau.

« Modals »

```
function viewDetails(id) { /* Remplit details-modal */ }
function closeDetailsModal() { /* Cache details-modal */ }

function openConfirmModal(id) { deleteId = id; }
function closeConfirmModal() { deleteId = null; }
function confirmDelete() {
    db[currentSection] = db[currentSection].filter(i => i.id !==
    renderTable();
}
```

Affiche les détails d’un élément ou supprime un élément après confirmation.

Difficultés rencontrées:

- Conception de la base de données
- Relations entre les tables
- Gestion des rôles utilisateurs
- Problèmes techniques et bugs

Améliorations futures:

- Espace parents
- Génération de bulletins PDF
- Notifications
- Interface plus moderne
- Version mobile

Conclusion

ce projet de site web de gestion d'école nous a permis de mettre en pratique nos connaissances en développement web tout en répondant à un besoin réel du domaine scolaire. Il facilite la gestion des étudiants, des professeurs, des classes, des matières et des notes à travers une plateforme centralisée et sécurisée. Ce projet nous a également aidés à développer nos compétences techniques, à travailler en équipe et à mieux comprendre le fonctionnement d'un système de gestion informatisé.

Merci Pour Votre Attention !

Meryem Fayd - Maryem Radouane - Soukaina Bahij