

Code de main.html :

C'est une excellente structure de base pour un formulaire d'inscription en popup utilisant Django et W3.CSS. Cependant, pour que ce code soit fonctionnel, il vous manque l'élément déclencheur (le bouton qui appelle la fonction showPopup()) et quelques optimisations de sécurité.

Voici comment compléter et améliorer votre fichier main.html :

1. Ajout du bouton déclencheur

Pour que l'utilisateur puisse voir le formulaire, vous devez ajouter ce bouton dans le corps de votre page (en dehors du div popup-overlay) :

HTML

```
<button onclick="showPopup()" class="w3-button w3-blue w3-round-large">
```

S'inscrire maintenant

```
</button>
```

2. Améliorations suggérées du code

Voici quelques points pour rendre votre code plus robuste :

- **Sécurité des mots de passe** : Changez type="text" en type="password" pour le champ mot de passe afin de masquer les caractères à la saisie.
 - **Accessibilité** : L'utilisation de display: flex dans le JavaScript est une excellente idée car elle centre parfaitement le contenu grâce au CSS que vous avez déjà écrit.
 - **Validation HTML5** : Vous avez déjà inclus required, ce qui est parfait.
-

3. Aperçu du fonctionnement

Voici une représentation visuelle de la hiérarchie de votre interface :

4. Code complet et nettoyé

Voici une version légèrement plus propre, tout en gardant votre logique :

HTML

```
<button onclick="showPopup()" class="w3-button w3-blue w3-round-large">S'inscrire</button>
```

```
<div id="myPopup" class="popup-overlay">
```

```
  <div class="popup-content">
```

```
    <span class="close-button" onclick="closePopup()">&times;</span>
```

```
    <h3>Formulaire d'inscription</h3>
```

```
    <form action="{% url 'votre_nom_de_vue' %}" method="post">
```

```
      {% csrf_token %}
```

```
      <p>
```

```
        <label for="nom">Nom et Prénom :</label>
```

```
        <input type="text" id="nom" name="nom" class="w3-input w3-border" required>
```

```
      </p>
```

```
      <p>
```

```
        <label for="email">Adresse Mail :</label>
```

```
        <input type="email" id="email" name="email" class="w3-input w3-border" required>
```

```
      </p>
```

```
    {% if diso3 or diso2 %}
```

```
<p class="w3-text-red w3-center">{{ diso3 }} {{ diso2 }}</p>
```

```
{% endif %}
```

```
<p>
```

```
<label for="password">Mot de passe :</label>
```

```
<input type="password" id="password" name="password" class="w3-input w3-border" required>
```

```
</p>
```

```
<div style="margin-top: 15px;">
```

```
<button type="submit" class="w3-button w3-green w3-round-large">Envoyer</button>
```

```
<button type="button" onclick="closePopup()" class="w3-button w3-red w3-round-large">Annuler</button>
```

```
</div>
```

```
</form>
```

```
</div>
```

```
</div>
```

```
<script>
```

```
function showPopup() {
```

```
    document.getElementById("myPopup").style.display = "flex";
```

```
}
```

```
function closePopup() {
```

```
document.getElementById("myPopup").style.display = "none";  
}  
// Fermer si on clique à l'extérieur de la boîte blanche  
window.onclick = function(event) {  
    let popup = document.getElementById("myPopup");  
    if (event.target == popup) {  
        closePopup();  
    }  
}  
</script>
```

Souhaitez-vous que je vous aide à rédiger la vue Django (views.py) correspondante pour traiter les données de ce formulaire ?

Code de header.html :

C'est un excellent début d'intégration. En plaçant l'appel de la fonction dans votre header.html, vous permettez à l'utilisateur d'accéder au formulaire d'inscription depuis n'importe quelle page du site.

Cependant, il y a un point crucial à noter : comme vous avez mis le code CSS et JavaScript en **commentaire** dans votre fichier header.html, la popup ne fonctionnera que si ces éléments sont actifs dans votre fichier main.html (ou votre fichier de base).

Voici une analyse et quelques recommandations pour finaliser votre header :

1. Analyse du Header

- **Logique de Session** : Votre utilisation de `{% if request.session.user %}` est correcte pour alterner entre l'affichage de "Déconnexion" et "S'inscrire".

- **Déclencheur** : Le lien `` est parfaitement configuré pour appeler la fonction JavaScript définie précédemment.

2. Organisation recommandée

Pour éviter de dupliquer le code CSS et JS sur chaque page, la meilleure pratique est de centraliser le code dans votre fichier de base (souvent nommé `base.html`) :

1. **Dans le Header** : Gardez uniquement les liens de navigation.
2. **Dans le Main/Base** : Gardez la structure HTML de la popup (`<div id="myPopup">...</div>`), le CSS et le JavaScript.

3. Petit correctif pour les messages d'erreur

Dans votre `main.html` précédent, vous aviez des variables comme `{{diso3}}`. Si une erreur survient lors de l'inscription, la page va se recharger et la popup sera fermée par défaut. L'utilisateur ne verra pas l'erreur.

Astuce : Ajoutez ce petit script à la fin de votre page pour que la popup s'ouvre automatiquement s'il y a une erreur :

JavaScript

```
// Si le serveur renvoie une erreur, on réaffiche la popup automatiquement
```

```
{% if diso2 or diso3 %}
```

```
    window.onload = function() {
```

```
        showPopup();
```

```
    };
```

```
{% endif %}
```

4. Liens manquants

N'oubliez pas de remplir les attributs `href` pour la déconnexion et la connexion afin que votre navigation soit complète :

HTML

```
<li class="nav-item">
```

```
  <a class="nav-link" href="{% url 'logout' %}">Déconnexion</a>
```

```
</li>
```

```
<li class="nav-item">
```

```
  <a class="nav-link" href="{% url 'login' %}">Connexion</a>
```

```
</li>
```

Souhaitez-vous que je vous montre comment structurer votre fichier base.html pour que la popup fonctionne partout sans répéter le code ?

Code de

C'est une excellente pratique pour l'expérience utilisateur (UX). En plaçant l'erreur directement sous le champ concerné, l'utilisateur comprend immédiatement ce qu'il doit corriger.

Pour que cela soit propre avec **W3.CSS**, je vous conseille d'utiliser une petite balise `` ou `<p>` avec une classe de couleur rouge et une police légèrement plus petite.

Voici comment modifier votre code dans main.html :

Code modifié pour le champ Email

HTML

```
<p>
```

```
  <label for="email"><b>Adresse Mail :</b></label>
```

```
  <input type="email" id="email" name="email"
```

```
    class="w3-input w3-border {% if diso2 or diso3 %}w3-border-red{% endif %}"
```

```
value="{{ request.POST.email }}" required>
```

```
{% if diso3 %}
```

```
<span class="w3-text-red w3-small" style="display: block; margin-top: 5px;">
```

```
<i class="fa fa-exclamation-triangle"></i> {{ diso3 }}
```

```
</span>
```

```
{% endif %}
```

```
{% if diso2 %}
```

```
<span class="w3-text-red w3-small" style="display: block; margin-top: 5px;">
```

```
<i class="fa fa-exclamation-triangle"></i> {{ diso2 }}
```

```
</span>
```

```
{% endif %}
```

```
</p>
```

Les améliorations apportées :

- **w3-border-red** : J'ai ajouté une condition dans la classe de l'input. Si une erreur existe (diso2 ou diso3), la bordure de la case devient rouge.
- **w3-small** : Rend le texte de l'erreur un peu plus discret que le label.
- **value="{{ request.POST.email }}"** : C'est très important ! Cela permet de **garder l'email écrit par l'utilisateur** après le rechargement de la page, pour qu'il n'ait pas à tout retaper.
- **display: block** : Assure que le message d'erreur passe bien à la ligne en dessous du champ.

Rappel important pour la Popup

Comme nous l'avons vu précédemment, si la page se recharge pour afficher l'erreur, la popup se fermera. N'oubliez pas d'ajouter ce script en bas de votre fichier pour qu'elle reste ouverte si une erreur est détectée :

JavaScript

```
<script>

// Si l'une des variables d'erreur existe, on ouvre la popup au chargement

{% if diso2 or diso3 %}

    window.onload = function() {

        showPopup();

    };

{% endif %}

</script>
```

Souhaitez-vous que je fasse la même chose pour le champ "Nom et Prénom" ou "Mot de passe" ?

Au fait, pour déverrouiller toutes les fonctionnalités de toutes les applis, vous devez activer [Gemini Apps Activity](#).

Code de main.html :

Pour intégrer les messages d'erreur directement sous le champ de l'adresse mail de manière propre et professionnelle, voici le code mis à jour.

J'ai utilisé des classes **W3.CSS** pour le style et ajouté une condition pour que le champ devienne rouge en cas d'erreur.

Code HTML mis à jour pour main.html

HTML

```
<p>
```



```
<label for="email">Adresse Mail :</label>

<input type="email" id="email" name="email"

    class="w3-input w3-border {% if diso2 or diso3 %}w3-border-red{% endif %}"

    value="{{ request.POST.email }}" required>

{% if diso3 %}

    <small class="w3-text-red" style="font-weight: bold;">

        <i class="fa fa-warning"></i> {{ diso3 }}

    </small>

{% endif %}

{% if diso2 %}

    <small class="w3-text-red" style="font-weight: bold;">

        <i class="fa fa-warning"></i> {{ diso2 }}

    </small>

{% endif %}

</p>
```

Pourquoi ces modifications ?

- **w3-border-red** : Cette classe W3.CSS colore la bordure de l'input en rouge. Elle ne s'active que si diso2 ou diso3 contient une valeur.
- **Balise <small>** : Elle permet d'afficher le texte de l'erreur dans une taille plus petite que le reste pour bien le distinguer.

- **value="{{ request.POST.email }}"** : C'est un point crucial. En cas d'erreur, la page se recharge. Cet attribut permet de **conserver l'email saisi par l'utilisateur** afin qu'il n'ait pas à le réécrire.
-

Rappel : Maintenir la POPUP ouverte

Si vous rechargez la page pour afficher ces erreurs, la popup se fermera automatiquement par défaut. Pour éviter cela, ajoutez ce script à la fin de votre fichier :

JavaScript

```
<script>
```

```
// Si l'une des erreurs est présente au chargement de la page, on ouvre la popup
```

```
{% if diso2 or diso3 %}
```

```
    window.onload = function() {
```

```
        showPopup();
```

```
    };
```

```
{% endif %}
```

```
</script>
```

Souhaitez-vous que j'applique également ce système de messages d'erreur pour les champs **Nom** et **Mot de passe** ?

Code de

Pour réaliser cela de manière propre en Django, l'astuce consiste à utiliser un **dictionnaire Python** dans votre views.py. Un dictionnaire permet de lier un nom d'erreur à un message précis, que vous passerez ensuite au fichier HTML (le "context").

Voici comment structurer votre code pour que la validation soit fluide.

1. Le fichier views.py

Au lieu d'un try/except (qui est plutôt utilisé pour des erreurs système), on utilise des conditions if pour vérifier si les champs sont vides.

Python

```
from django.shortcuts import render
```

```
def inscription_view(request):
```

```
    # On initialise le dictionnaire d'erreurs vide
```

```
    erreurs_champs = {}
```

```
    if request.method == "POST":
```

```
        nom = request.POST.get('nom', '').strip()
```

```
        email = request.POST.get('email', '').strip()
```

```
        # Validation du champ NOM
```

```
        if not nom:
```

```
            erreurs_champs['nom'] = "Le champ nom ne devrait pas être vide dans votre popup !"
```

```
        # Validation du champ EMAIL (exemple supplémentaire)
```

```
        if not email:
```

```
            erreurs_champs['email'] = "L'adresse mail est obligatoire."
```

```
        # S'il n'y a aucune erreur, on traite les données
```

```
if not erreurs_champs:

    # Code pour enregistrer l'utilisateur ici...

    pass


return render(request, 'main.html', {

    'erreurs_champs': erreurs_champs,

    # On renvoie les données saisies pour ne pas vider le formulaire

    'donnees': request.POST

})
```

2. Le fichier main.html

Dans votre HTML, vous allez tester si la clé nom existe dans votre dictionnaire erreurs_champs.

HTML

<p>

<label for="nom">Nom et Prénom :</label>

<input type="text" id="nom" name="nom"

class="w3-input w3-border {% if erreurs_champs.nom %}w3-border-red{% endif %}"

value="{{ donnees.nom|default:" }}">

{% if erreurs_champs.nom %}

<p class="w3-text-red w3-small" style="margin-top: 5px;">

```
        {{ erreurs_champs.nom }}  
    </p>  
{% endif %}  
</p>
```

3. Le script pour maintenir la POPUP ouverte

Pour que l'utilisateur voie l'erreur, la popup doit être ouverte immédiatement après le rechargement de la page. Ajoutez ceci en bas de votre fichier HTML :

HTML

```
<script>  
    // Si le dictionnaire erreurs_champs contient quelque chose, on affiche la popup  
    {% if erreurs_champs %}  
        window.onload = function() {  
            showPopup();  
        };  
    {% endif %}  
</script>
```

Pourquoi cette méthode est la meilleure ?

1. **Ciblage précis** : L'erreur du "nom" s'affiche sous le "nom", celle de l'email sous l'email.
2. **Persistance** : Grâce à `value="{{ donnees.nom }}"`, si l'utilisateur a rempli son nom mais fait une erreur sur l'email, son nom ne s'effacera pas.
3. **Clarté** : Le code est beaucoup plus lisible qu'une longue liste de variables `diso2`, `diso3`, etc.

Souhaitez-vous que j'ajoute une validation pour vérifier si l'email a un format valide (avec un @) dans le views.py ?

Code de

C'est parfait. En ajoutant la validation du format de l'email, vous rendez votre formulaire vraiment professionnel.

Voici comment combiner la validation du champ vide et la validation du format (présence d'un @) dans votre views.py, ainsi que la mise à jour du template.

1. Le fichier views.py (Validation complète)

On utilise ici la logique du dictionnaire erreurs_champs. Pour l'email, on vérifie d'abord s'il est vide, puis s'il contient le caractère @.

Python

```
from django.shortcuts import render
```

```
def inscription_view(request):
```

```
    erreurs_champs = {}
```

```
    if request.method == "POST":
```

```
        nom = request.POST.get('nom', "").strip()
```

```
        email = request.POST.get('email', "").strip()
```

```
        password = request.POST.get('password', "").strip()
```

```
        # 1. Validation du NOM
```

```
        if not nom:
```

```
            erreurs_champs['nom'] = "Le champ nom ne devrait pas être vide dans votre popup !"
```

2. Validation de l'EMAIL (Vide + Format)

if not email:

erreurs_champs['email'] = "L'adresse mail est obligatoire."

elif "@" not in email:

erreurs_champs['email'] = "Veuillez saisir une adresse mail valide (ex: nom@domaine.com)."

3. Validation du PASSWORD

if not password:

erreurs_champs['password'] = "Le mot de passe ne peut pas être vide."

elif len(password) < 6:

erreurs_champs['password'] = "Le mot de passe doit contenir au moins 6 caractères."

Traitement final

if not erreurs_champs:

OK : Enregistrement ou connexion

return render(request, 'success.html')

Si erreurs ou premier chargement

return render(request, 'main.html', {

'erreurs_champs': erreurs_champs,

```
'donnees': request.POST
```

```
})
```

2. Le fichier main.html (Mise à jour)

Grâce au dictionnaire, vous pouvez maintenant placer chaque erreur précisément.

HTML

```
<form action="#" method="post">
```

```
    {% csrf_token %}
```

```
<p>
```

```
    <label for="nom">Nom et Prénom :</label>
```

```
    <input type="text" id="nom" name="nom"
```

```
        class="w3-input w3-border {% if erreurs_champs.nom %}w3-border-red{% endif %}"
```

```
        value="{{ donnees.nom|default:'' }}">
```

```
    {% if erreurs_champs.nom %}
```

```
        <span class="w3-text-red w3-small">{{ erreurs_champs.nom }}</span>
```

```
    {% endif %}
```

```
</p>
```

```
<p>
```

```
    <label for="email">Adresse Mail :</label>
```



```
<input type="email" id="email" name="email"
      class="w3-input w3-border {% if erreurs_champs.email %}w3-border-red{% endif %}"
      value="{{ donnees.email|default:'' }}">
{% if erreurs_champs.email %}
  <span class="w3-text-red w3-small">{{ erreurs_champs.email }}</span>
{% endif %}
</p>
```

```
<p>
  <label for="password">Mot de passe :</label>
  <input type="password" id="password" name="password"
        class="w3-input w3-border {% if erreurs_champs.password %}w3-border-red{% endif %}">
  {% if erreurs_champs.password %}
    <span class="w3-text-red w3-small">{{ erreurs_champs.password }}</span>
  {% endif %}
</p>
```

```
<button type="submit" class="w3-button w3-green w3-round-large">Envoyer</button>
<button type="button" onclick="closePopup()" class="w3-button w3-red w3-round-large">Annuler</button>
</form>
```

3. Rappel du Script d'ouverture automatique

N'oubliez pas d'inclure ce script juste avant la balise `</body>` pour que la popup s'affiche d'elle-même si le serveur renvoie des erreurs.

JavaScript

```
<script>

    {% if erreurs_champs %}

        // Si le dictionnaire contient des erreurs, on réaffiche la popup

        showPopup();

    {% endif %}

</script>
```

Résumé de ce que vous avez maintenant :

- Une gestion **centralisée** des erreurs dans `views.py`.
- Des messages **ciblés** sous chaque champ.
- Une popup qui **reste ouverte** tant que le formulaire n'est pas correct.
- Les données saisies qui **ne s'effacent pas** (sauf le mot de passe pour la sécurité).

Souhaitez-vous que je vous aide à coder la partie "Enregistrement en base de données" dans le `if not erreurs_champs`: ?

Code de

Voici comment intégrer l'enregistrement des données dans votre `views.py` en utilisant votre modèle `User`.

Pour des raisons de sécurité, j'ai ajouté le hachage du mot de passe avec `make_password`. Il ne faut jamais enregistrer un mot de passe en texte brut dans une base de données.

1. Le fichier `views.py` complété

Python

```
from django.shortcuts import render, redirect  
  
from django.contrib.hashers import make_password  
  
from .models import User # Importez votre modèle
```

```
def inscription_view(request):
```

```
    erreurs_champs = {}
```

```
    if request.method == "POST":
```

```
        nom = request.POST.get('nom', "").strip()
```

```
        email = request.POST.get('email', "").strip()
```

```
        password = request.POST.get('password', "").strip()
```

```
        # --- VALIDATIONS ---
```

```
        if not nom:
```

```
            erreurs_champs['nom'] = "Le champ nom ne devrait pas être vide !"
```

```
        if not email:
```

```
            erreurs_champs['email'] = "L'adresse mail est obligatoire."
```

```
        elif "@" not in email:
```

```
            erreurs_champs['email'] = "Format d'email invalide."
```

```
# Vérifier si l'email existe déjà

elif User.objects.filter(email=email).exists():

    erreurs_champs['email'] = "Cet email est déjà utilisé."


if not password:

    erreurs_champs['password'] = "Le mot de passe est obligatoire."

elif len(password) < 6:

    erreurs_champs['password'] = "Minimum 6 caractères."


# --- ENREGISTREMENT ---

if not erreurs_champs:

    # Création de l'utilisateur dans la base de données

    nouvel_utilisateur = User(

        nom=nom,

        email=email,

        password=make_password(password) # Hachage sécurisé

    )

    nouvel_utilisateur.save()


# Optionnel : Connecter l'utilisateur via la session

request.session['user'] = nouvel_utilisateur.nom
```

```
return redirect('accueil') # Rediriger vers une page de succès
```

```
return render(request, 'main.html', {  
    'erreurs_champs': erreurs_champs,  
    'donnees': request.POST  
})
```

2. Rappel pour le fichier models.py

Votre modèle est bien défini. Notez que comme vous avez commenté le champ prenom, j'ai également commenté la méthode `__str__` qui l'utilisait pour éviter des erreurs.

3. Pourquoi utiliser `make_password` ?

Si un pirate accède à votre base de données, il ne pourra pas lire les mots de passe des utilisateurs car ils seront transformés en une suite de caractères illisibles (un "hash"). Django gère cela nativement.

4. Rappel sur le Template (main.html)

Pensez à bien utiliser `type="password"` dans votre HTML pour que les caractères ne s'affichent pas à l'écran lors de la saisie :

HTML

```
<p>  
    <label for="password">Password :</label>  
    <input type="password" id="password" name="password"  
        class="w3-input w3-border {% if erreurs_champs.password %}w3-border-red{% endif %}" required>  
    {% if erreurs_champs.password %}
```

```
<span class="w3-text-red w3-small">{{ erreurs_champs.password }}</span>

{% endif %}

</p>
```

Souhaitez-vous que je vous montre comment créer la vue de "Connexion" pour vérifier l'email et le mot de passe stockés ?

Code de

Voici le code complet et harmonisé pour vos deux fichiers. Cette version intègre la gestion des erreurs par dictionnaire, la persistance des données (pour ne pas tout retaper en cas d'erreur) et l'ouverture automatique de la popup si le serveur renvoie une validation négative.

1. Fichier views.py

Ce code gère la validation, le hachage du mot de passe et l'enregistrement dans votre modèle User.

Python

```
from django.shortcuts import render, redirect

from django.contrib.hashers import make_password

from .models import User # Import de votre classe User depuis models.py


def inscription_view(request):

    erreurs_champs = {}

    donnees_saisies = {}


    if request.method == "POST":
```

```
# Récupération des données

nom = request.POST.get('nom', '').strip()
email = request.POST.get('email', '').strip()
password = request.POST.get('password', '').strip()


# On garde les données pour les réafficher en cas d'erreur
donnees_saisies = {'nom': nom, 'email': email}


# --- VALIDATIONS ---

# Nom

if not nom:
    erreurs_champs['nom'] = "Le champ nom ne devrait pas être vide dans votre popup !"


# Email (Obligatoire + Format + Unicité)

if not email:
    erreurs_champs['email'] = "L'adresse mail est obligatoire."

elif "@" not in email:
    erreurs_champs['email'] = "Veuillez saisir une adresse mail valide."

elif User.objects.filter(email=email).exists():
    erreurs_champs['email'] = "Cette adresse mail est déjà enregistrée."
```

```
# Mot de passe

if not password:

    erreurs_champs['password'] = "Le mot de passe ne peut pas être vide."

elif len(password) < 6:

    erreurs_champs['password'] = "Le mot de passe doit contenir au moins 6 caractères."


# --- TRAITEMENT FINAL ---

if not erreurs_champs:

    # Création de l'utilisateur

    nouvel_utilisateur = User(

        nom=nom,

        email=email,

        password=make_password(password) # Sécurité : mot de passe haché

    )

    nouvel_utilisateur.save()


    # Création de la session

    request.session['user'] = nouvel_utilisateur.nom


    # Redirection vers l'accueil ou page de succès

    return redirect('accueil')
```



```
return render(request, 'main.html', {  
    'erreurs_champs': erreurs_champs,  
    'donnees': donnees_saisies  
})
```

2. Fichier main.html

Ce fichier contient le CSS, la structure HTML du formulaire et le JavaScript pour gérer l'affichage.

HTML

```
<link rel="stylesheet" href="https://www.w3schools.com/w3css/4/w3.css">
```

```
<style>
```

```
/* Style pour le popup */
```

```
.popup-overlay {
```

```
    display: none;
```

```
    position: fixed;
```

```
    z-index: 999;
```

```
    left: 0; top: 0;
```

```
    width: 100%; height: 100%;
```

```
    background-color: rgba(0,0,0,0.7);
```

```
    justify-content: center;
```

```
    align-items: center;
}
```

```
.popup-content {
    background-color: #fefefe;
    padding: 20px;
    border: 1px solid #888;
    width: 90%;
    max-width: 500px;
    border-radius: 8px;
    box-shadow: 0 4px 20px rgba(0,0,0,0.3);
}
```

```
.close-button {
    color: #aaa;
    float: right;
    font-size: 28px;
    font-weight: bold;
    cursor: pointer;
}
```

```
.close-button:hover { color: black; }
```

```
</style>
```

```
<button onclick="showPopup()" class="w3-button w3-blue w3-round">S'inscrire</button>
```

```
<div id="myPopup" class="popup-overlay">
```

```
  <div class="popup-content">
```

```
    <span class="close-button" onclick="closePopup()">&times;</span>
```

```
    <h3>Formulaire d'inscription</h3>
```

```
    <form action="#" method="post">
```

```
      {% csrf_token %}
```

```
      <p>
```

```
        <label for="nom">Nom et Prénom :</label>
```

```
        <input type="text" id="nom" name="nom"
```

```
          class="w3-input w3-border {% if erreurs_champs.nom %}w3-border-red{% endif %}"
```

```
          value="{{ donnees.nom|default:'' }}">
```

```
        {% if erreurs_champs.nom %}
```

```
          <span class="w3-text-red w3-small"><b>{{ erreurs_champs.nom }}</b></span>
```

```
        {% endif %}
```

</p>

<p>

<label for="email">Adresse Mail :</label>

<input type="email" id="email" name="email"

class="w3-input w3-border {% if erreurs_champs.email %}w3-border-red{% endif %}"

value="{{ donnees.email | default:" }}">

{% if erreurs_champs.email %}

{{ erreurs_champs.email }}

{% endif %}

</p>

<p>

<label for="password">Mot de passe :</label>

<input type="password" id="password" name="password"

class="w3-input w3-border {% if erreurs_champs.password %}w3-border-red{% endif %}">

{% if erreurs_champs.password %}

{{ erreurs_champs.password }}

{% endif %}

</p>

```
<button type="submit" class="w3-button w3-green w3-round-large" style="margin-top: 10px;">Envoyer</button>
```

```
<button type="button" onclick="closePopup()" class="w3-button w3-red w3-round-large" style="margin-top: 10px;">Annuler</button>
```

```
</form>
```

```
</div>
```

```
</div>
```

```
<script>
```

```
function showPopup() {
```

```
    document.getElementById("myPopup").style.display = "flex";
```

```
}
```

```
function closePopup() {
```

```
    document.getElementById("myPopup").style.display = "none";
```

```
}
```

```
// Fermer si clic à l'extérieur
```

```
window.onclick = function(event) {
```

```
    var popup = document.getElementById("myPopup");
```

```
    if (event.target == popup) { closePopup(); }
```

```
}
```

```
// AUTO-OUVERTURE : Si Django renvoie des erreurs, on réouvre la popup immédiatement
{% if erreurs_champs %}

    showPopup();

{% endif %}

</script>
```

Points clés de cette solution :

1. **Hachage** : Le mot de passe est sécurisé via `make_password`.
2. **Validation d'unicité** : Le `views.py` vérifie si l'email existe déjà dans la base pour éviter les doublons.
3. **UX (Expérience Utilisateur)** : La popup ne disparaît pas si l'utilisateur fait une erreur, et les champs (Nom/Email) restent remplis.
4. **W3.CSS** : Utilisation des classes `w3-border-red` pour surligner visuellement les erreurs.

Souhaitez-vous que je vous aide à mettre en place la **connexion (Login)** avec vérification du mot de passe haché ?

Code de

C'est une étape essentielle pour que vos utilisateurs puissent accéder à leur compte après s'être inscrits. Pour la connexion, nous allons utiliser `check_password`, qui permet de comparer le mot de passe saisi avec le "hash" (le code crypté) stocké dans votre base de données.

Voici comment structurer votre vue de connexion et l'intégrer à votre système de popup.

1. Fichier `views.py` (Ajout de la fonction connexion)

Nous allons créer une nouvelle fonction `connexion_view`. Elle utilise la même logique de dictionnaire d'erreurs.

Python

```
from django.shortcuts import render, redirect
```

```
from django.contrib.hashers import check_password # Pour vérifier le mot de passe haché
```

```
from .models import User
```

```
def connexion_view(request):
```

```
    erreurs_connexion = {}
```

```
    if request.method == "POST":
```

```
        email = request.POST.get('email', "").strip()
```

```
        password = request.POST.get('password', "").strip()
```

```
    # 1. Vérification si l'utilisateur existe
```

```
    try:
```

```
        utilisateur = User.objects.get(email=email)
```

```
    # 2. Vérification du mot de passe
```

```
    if check_password(password, utilisateur.password):
```

```
        # SUCCÈS : On crée la session
```

```
        request.session['user_id'] = utilisateur.id
```

```
        request.session['user_nom'] = utilisateur.nom
```

```
        return redirect('accueil')
```

```
    else:
```

```
erreurs_connexion['login'] = "Email ou mot de passe incorrect."
```

```
except User.DoesNotExist:
```

```
    erreurs_connexion['login'] = "Email ou mot de passe incorrect."
```

```
return render(request, 'main.html', {  
    'erreurs_connexion': erreurs_connexion,  
    'email_saisi': request.POST.get('email', '')  
})
```

2. Modification du fichier main.html

Si vous voulez une popup séparée pour la connexion, vous pouvez dupliquer la structure de la popup d'inscription en changeant l'ID (par exemple id="popupConnexion").

Voici comment afficher l'erreur de connexion de manière globale en haut du formulaire :

HTML

```
<div id="popupConnexion" class="popup-overlay">  
    <div class="popup-content">  
        <span class="close-button" onclick="closeConnexion()">&times;</span>  
        <h3>Connexion</h3>  
  
        <form action="{% url 'votre_url_connexion' %}" method="post">
```



```
{% csrf_token %}
```

```
{% if erreurs_connexion.login %}
```

```
<div class="w3-panel w3-red w3-display-container w3-round">
```

```
<p class="w3-small">{{ erreurs_connexion.login }}</p>
```

```
</div>
```

```
{% endif %}
```

```
<p>
```

```
<label>Email :</label>
```

```
<input type="email" name="email" class="w3-input w3-border" value="{{ email_saisi|default:'' }}" required>
```

```
</p>
```

```
<p>
```

```
<label>Mot de passe :</label>
```

```
<input type="password" name="password" class="w3-input w3-border" required>
```

```
</p>
```

```
<button type="submit" class="w3-button w3-blue w3-round-large">Se connecter</button>
```

```
</form>
```

```
</div>
```

</div>

<script>

```
// Script pour ouvrir la popup de connexion si erreur

{% if erreurs_connexion %}

    document.getElementById("popupConnexion").style.display = "flex";

{% endif %}
```

</script>

3. Gestion de la Déconnexion

Pour la déconnexion, c'est très simple : il suffit de vider la session.

Python

```
def deconnexion_view(request):

    # On supprime toutes les données de session

    request.session.flush()

    return redirect('accueil')
```

Points à retenir pour la sécurité :

- **check_password** : C'est la seule façon sécurisée de comparer un mot de passe avec un utilisateur créé via make_password.
- **Messages d'erreur flous** : Notez que j'ai mis "Email ou mot de passe incorrect" dans les deux cas (si l'email n'existe pas OU si le mot de passe est faux). Cela empêche un pirate de savoir si un email existe déjà dans votre base de données.

- **request.session.flush()** : Cette fonction détruit la session actuelle, ce qui déconnecte l'utilisateur proprement.

Souhaitez-vous que je vous aide à paramétrer le fichier urls.py pour lier toutes ces fonctions ensemble ?

Code de

Voici le code complet pour votre fichier urls.py (situé dans votre application). Il permet de lier les boutons de votre header et les formulaires de vos popups aux fonctions que nous avons créées.

1. Configuration du fichier urls.py

Assurez-vous d'importer vos vues correctement au début du fichier.

Python

```
from django.urls import path
```

```
from . import views
```

```
urlpatterns = [
```

```
    # Page d'accueil (là où se trouvent vos popups)
```

```
    path("", views.inscription_view, name='accueil'),
```

```
    # URL pour le formulaire de connexion
```

```
    path('connexion/', views.connexion_view, name='connexion'),
```

```
    # URL pour la déconnexion
```

```
    path('deconnexion/', views.deconnexion_view, name='logout'),
```

]

2. Rappel pour le Header (header.html)

Maintenant que vos URLs sont nommées, vous devez mettre à jour les liens dans votre menu pour qu'ils pointent vers les bonnes fonctions Django :

HTML

```
{% if request.session.user_nom %}

    <li class="nav-item">

        <span class="nav-link">Bienvenue, {{ request.session.user_nom }}</span>

    </li>

    <li class="nav-item">

        <a class="nav-link" href="{% url 'logout' %}">Déconnexion</a>

    </li>

{% else %}

    <li class="nav-item">

        <a class="nav-link" onclick="showPopup()" style="cursor: pointer;">S'inscrire</a>

    </li>

    <li class="nav-item">

        <a class="nav-link" onclick="showConnexionPopup()" style="cursor: pointer;">Connexion</a>

    </li>

{% endif %}
```

3. Schéma de fonctionnement global

Voici comment les données circulent entre votre utilisateur, les fichiers HTML et votre base de données :

4. Résumé technique de votre système

Action	Fichier concerné	Outil utilisé
Saisie	main.html	Formulaire HTML + W3.CSS
Validation	views.py	Dictionnaire erreurs_champs
Sécurité	views.py	make_password / check_password
Stockage	models.py	Modèle User (SQLite/PostgreSQL)
Mémoire	views.py	request.session

Dernière astuce : Le message de succès

Si vous voulez afficher un petit message de confirmation quand l'inscription réussit, vous pouvez utiliser le framework de messages de Django.

Voulez-vous que je vous montre comment ajouter une petite alerte "Inscription réussie !" qui apparaît juste après la redirection ?

Code de

C'est une excellente touche finale. Pour afficher ce message "Inscription réussie !", nous allons utiliser le framework **messages** de Django. C'est le moyen le plus propre pour transmettre une information d'une vue à une autre après une redirection.

Voici comment mettre cela en place :

1. Mise à jour de views.py

Ajoutez l'import des messages et utilisez messages.success juste avant le redirect.

Python

```
from django.shortcuts import render, redirect
```

```
from django.contrib import messages # Import indispensable
```

```
from django.contrib.hashers import make_password
```

```
from .models import User
```

```
def inscription_view(request):
```

```
    erreurs_champs = {}
```

```
    if request.method == "POST":
```

```
        # ... (vos validations existantes) ...
```

```
    if not erreurs_champs:
```

```
        # Enregistrement
```

```
        nouvel_utilisateur = User(
```

```
            nom=request.POST.get('nom'),
```

```
            email=request.POST.get('email'),
```

```
            password=make_password(request.POST.get('password'))
```

```
        )
```

```
        nouvel_utilisateur.save()
```

```
# AJOUT DU MESSAGE DE SUCCÈS
```

```
messages.success(request, "Félicitations ! Votre compte a été créé avec succès.")
```

```
return redirect('accueil')
```

```
return render(request, 'main.html', {'erreurs_champs': erreurs_champs})
```

2. Affichage dans main.html

Placez ce bloc de code en haut de votre page (juste après l'ouverture de la balise <body> ou dans votre header) pour que le message soit bien visible.

HTML

```
{% if messages %}
```

```
{% for message in messages %}
```

```
<div id="msg-success" class="w3-panel w3-green w3-display-container w3-animate-top" style="margin: 10px;">
```

```
<span onclick="this.parentElement.style.display='none'"
```

```
class="w3-button w3-display-topright">&times;</span>
```

```
<p>{{ message }}</p>
```

```
</div>
```

```
<script>
```

```
// Faire disparaître le message automatiquement après 5 secondes
```

```
setTimeout(function() {
```

```
        document.getElementById('msg-success').style.display = 'none';  
    }, 5000);  
</script>  
{% endfor %}  
{% endif %}
```

3. Fonctionnement Final

1. **L'utilisateur** remplit le formulaire dans la popup.
2. **Django** valide les données.
3. Si tout est correct, l'utilisateur est enregistré, un message est mis en "mémoire tampon" et la page est redirigée.
4. À la réactualisation, la balise {% if messages %} détecte l'alerte et l'affiche en haut de l'écran avec une animation W3.CSS.

Votre projet est maintenant complet !

- **Popups** interactives pour l'inscription et la connexion.
- **Validations** précises champ par champ.
- **Sécurité** avec hachage des mots de passe.
- **Sessions** pour reconnaître l'utilisateur connecté.
- **Messages Flash** pour confirmer les actions.

Souhaitez-vous que je vous aide à créer une page de **profil utilisateur** où l'on peut voir et modifier ses informations ?

Code complet de views.py et main.html :

Voici le code complet, centralisé et nettoyé pour vos deux fichiers. Cette version assure que les erreurs s'affichent au bon endroit, que la popup reste ouverte en cas de problème, et que les messages de succès s'affichent après l'enregistrement.

1. Fichier **views.py**

Ce fichier contient toute la logique de validation pour l'inscription et la connexion, ainsi que la gestion des messages de succès.

Python

```
from django.shortcuts import render, redirect

from django.contrib import messages

from django.contrib.hashers import make_password, check_password

from .models import User # Assurez-vous que votre modèle User est bien dans models.py


def index_view(request):

    erreurs_champs = {}

    donnees_saisies = {}


    if request.method == "POST":

        # Récupération des données du formulaire

        nom = request.POST.get('nom', '').strip()

        email = request.POST.get('email', '').strip()

        password = request.POST.get('password', '').strip()


        # On stocke les données pour les réafficher en cas d'erreur (sauf le password)
```

```
donnees_saisies = {'nom': nom, 'email': email}
```

```
# --- VALIDATIONS PERSONNALISÉES ---
```

```
if not nom:
```

```
    erreurs_champs['nom'] = "Le champ nom ne devrait pas être vide dans votre popup !"
```

```
if not email:
```

```
    erreurs_champs['email'] = "L'adresse mail est obligatoire."
```

```
elif "@" not in email:
```

```
    erreurs_champs['email'] = "Format d'email invalide (manque @)."
```

```
elif User.objects.filter(email=email).exists():
```

```
    erreurs_champs['email'] = "Cet email est déjà utilisé."
```

```
if not password:
```

```
    erreurs_champs['password'] = "Le mot de passe est obligatoire."
```

```
elif len(password) < 6:
```

```
    erreurs_champs['password'] = "Le mot de passe doit contenir au moins 6 caractères."
```

```
# --- ENREGISTREMENT ---
```

```
if not erreurs_champs:
```

```
    try:
```

```
nouvel_utilisateur = User(
    nom=nom,
    email=email,
    password=make_password(password) # Hachage sécurisé
)
nouvel_utilisateur.save()

# Message de succès et redirection
messages.success(request, "Félicitations ! Votre compte a été créé avec succès.")
return redirect('accueil') # Nom de l'URL de votre page principale
except Exception as e:
    erreurs_champs['global'] = "Une erreur est survenue lors de l'enregistrement."

return render(request, 'main.html', {
    'erreurs_champs': erreurs_champs,
    'donnees': donnees_saisies
})
```

2. Fichier main.html

Ce fichier contient le design W3.CSS, la structure de la popup et le script qui détecte les erreurs pour maintenir la fenêtre ouverte.

HTML

```
<!DOCTYPE html>
```

```
<html lang="fr">
```

```
<head>
```

```
  <meta charset="UTF-8">
```

```
  <link rel="stylesheet" href="https://www.w3schools.com/w3css/4/w3.css">
```

```
  <style>
```

```
    /* Style pour le popup */
```

```
    .popup-overlay {
```

```
      display: none; /* Caché par défaut */
```

```
      position: fixed;
```

```
      z-index: 999;
```

```
      left: 0; top: 0;
```

```
      width: 100%; height: 100%;
```

```
      background-color: rgba(0,0,0,0.7);
```

```
      justify-content: center;
```

```
      align-items: center;
```

```
    }
```

```
    .popup-content {
```

```
      background-color: #fefefe;
```

```
      padding: 20px;
```

```
      border-radius: 8px;
```

```
    width: 90%;
    max-width: 500px;
    position: relative;
    box-shadow: 0 4px 15px rgba(0,0,0,0.3);
}

.close-button {
    color: #aaa;
    float: right;
    font-size: 28px;
    font-weight: bold;
    cursor: pointer;
}

.close-button:hover { color: black; }

</style>

</head>

<body>

{% if messages %}

{% for message in messages %}

<div class="w3-panel w3-green w3-display-container w3-animate-top" style="margin: 10px;">

    <span onclick="this.parentElement.style.display='none'" class="w3-button w3-display-topright">&times;</span>
```

```

        <p>{{ message }}</p>
    </div>
{% endfor %}
{% endif %}

<div class="w3-container w3-padding">

    <button onclick="showPopup()" class="w3-button w3-blue w3-round-large">S'inscrire</button>

</div>

<div id="myPopup" class="popup-overlay">

    <div class="popup-content w3-animate-zoom">

        <span class="close-button" onclick="closePopup()">&times;</span>

        <h3>Formulaire d'inscription</h3>

        <form action="#" method="post">

            {% csrf_token %}

            <p>

                <label for="nom">Nom et Prénom :</label>

                <input type="text" id="nom" name="nom"

                    class="w3-input w3-border {% if erreurs_champs.nom %}w3-border-red{% endif %}"

```

```
        value="{{ donnees.nom | default:" }}">
    {% if erreurs_champs.nom %}

        <span class="w3-text-red w3-small">{{ erreurs_champs.nom }}</span>

    {% endif %}

</p>
```

```
<p>

<label for="email">Adresse Mail :</label>

<input type="email" id="email" name="email"

        class="w3-input w3-border {% if erreurs_champs.email %}w3-border-red{% endif %}"

        value="{{ donnees.email | default:" }}">

    {% if erreurs_champs.email %}

        <span class="w3-text-red w3-small">{{ erreurs_champs.email }}</span>

    {% endif %}

</p>
```

```
<p>

<label for="password">Mot de passe :</label>

<input type="password" id="password" name="password"

        class="w3-input w3-border {% if erreurs_champs.password %}w3-border-red{% endif %}">

    {% if erreurs_champs.password %}
```

```
        <span class="w3-text-red w3-small">{{ erreurs_champs.password }}</span>

    {% endif %}

</p>

<div class="w3-padding-16">

    <button type="submit" class="w3-button w3-green w3-round">Envoyer</button>

    <button type="button" onclick="closePopup()" class="w3-button w3-red w3-round">Annuler</button>

</div>

</form>

</div>

</div>

<script>

    // Fonctions de base

    function showPopup() {

        document.getElementById("myPopup").style.display = "flex";

    }

    function closePopup() {

        document.getElementById("myPopup").style.display = "none";

    }

}
```



```
// Fermer en cliquant à l'extérieur  
window.onclick = function(event) {  
    let popup = document.getElementById("myPopup");  
    if (event.target == popup) { closePopup(); }  
}
```

```
// OUVERTURE AUTOMATIQUE SI ERREUR
```

```
{% if erreurs_champs %}  
    showPopup();  
{% endif %}
```

```
</script>
```

```
</body>
```

```
</html>
```

Pourquoi ce code fonctionne parfaitement :

1. **Sémantique** : L'utilisation de {% if erreurs_champs.nom %} permet d'afficher l'erreur uniquement sous le champ concerné.
2. **Expérience Utilisateur** : La popup se ré-ouvre toute seule après le rechargement de la page si une erreur a été trouvée par Django.
3. **Hachage** : Le mot de passe est crypté avant d'être sauvegardé en base de données.
4. **Mémoire** : L'attribut value="{{ donnees.nom|default:'' }}" permet à l'utilisateur de ne pas perdre ce qu'il a déjà écrit si le formulaire est invalide.

Souhaitez-vous que je vous aide à mettre en place la redirection automatique vers une page de profil après l'inscription ?

Code complet de urls.py

Voici le code complet pour votre fichier urls.py. Ce fichier doit se trouver à l'intérieur de votre dossier d'application (celui qui contient vos fichiers views.py et models.py).

1. Fichier urls.py

Ce fichier lie les noms d'URLs utilisés dans vos fichiers HTML (`{% url '...' %}`) aux fonctions correspondantes dans votre views.py.

Python

```
from django.urls import path
```

```
from . import views
```

```
urlpatterns = [
```

```
    # Page principale qui gère l'affichage de base et l'inscription
```

```
    path("", views.index_view, name='accueil'),
```

```
    # URL pour traiter la connexion
```

```
    path('connexion/', views.connexion_view, name='connexion'),
```

```
    # URL pour la déconnexion
```

```
    path('deconnexion/', views.deconnexion_view, name='logout'),
```

```
]
```

2. Rappel pour le fichier urls.py du Projet (Dossier parent)

Pour que Django reconnaisse les URLs de votre application, assurez-vous que le fichier urls.py principal (celui situé dans le dossier du projet, à côté de settings.py) inclut bien les URLs de votre application :

Python

```
from django.contrib import admin
```

```
from django.urls import path, include
```

```
urlpatterns = [  
    path('admin/', admin.site.urls),  
    # Remplacez 'nom_de_votre_app' par le vrai nom de votre dossier d'application  
    path('', include('nom_de_votre_app.urls')),  
]
```

3. Fonctionnement des liens dans vos templates

Grâce au paramètre name='...' défini dans le fichier urls.py, vous pouvez maintenant utiliser ces liens dynamiques dans votre **Header** ou vos **Formulaires** :

- **Pour le formulaire d'inscription** : `<form action="{% url 'accueil' %}" method="post">`
- **Pour le formulaire de connexion** : `<form action="{% url 'connexion' %}" method="post">`
- **Pour le lien de déconnexion** : `Déconnexion`

Résumé de la structure

Votre projet suit maintenant une architecture propre :

1. **L'URL** capte la demande (urls.py).

2. **La Vue** traite la logique (validation des champs vides, format email, hachage du mot de passe) dans views.py.
3. **Le Modèle** enregistre les données de manière permanente dans models.py.
4. **Le Template** affiche les résultats et les erreurs dans main.html.

Souhaitez-vous que je vous aide à tester si la base de données enregistre correctement votre premier utilisateur ?