Documentation Outil d'Audit Interne

1 Introduction

1.1 Objectif de l'outil

L'outil d'audit interne permet de :

- gérer un référentiel d'audit (normes, services, critères/lignes),
- planifier et réaliser des audits sur une norme pour un ou plusieurs services,
- saisir les résultats (statut, commentaires, preuves),
- suivre l'avancement des audits (brouillon, en cours, clôturé),
- historiser les audits réalisés.

1.2 Public cible

- Auditeurs: réalisent les audits.
- Administrateurs système : déploient l'application sur IIS, gèrent la base SQL Server, les sauvegardes, la supervision.

2 Documentation fonctionnelle

2.1 Vue d'ensemble de l'interface

L'application est composée de trois grandes zones :

2.1.1 Page principale 'Audits'

Point d'entrée des utilisateurs pour lancer un nouvel audit.

L'utilisateur sélectionne :

- **Une norme**
- **Un ou plusieurs services**
- Les **lignes/critères** liées à cette norme (filtrées par service)

L'interface propose :

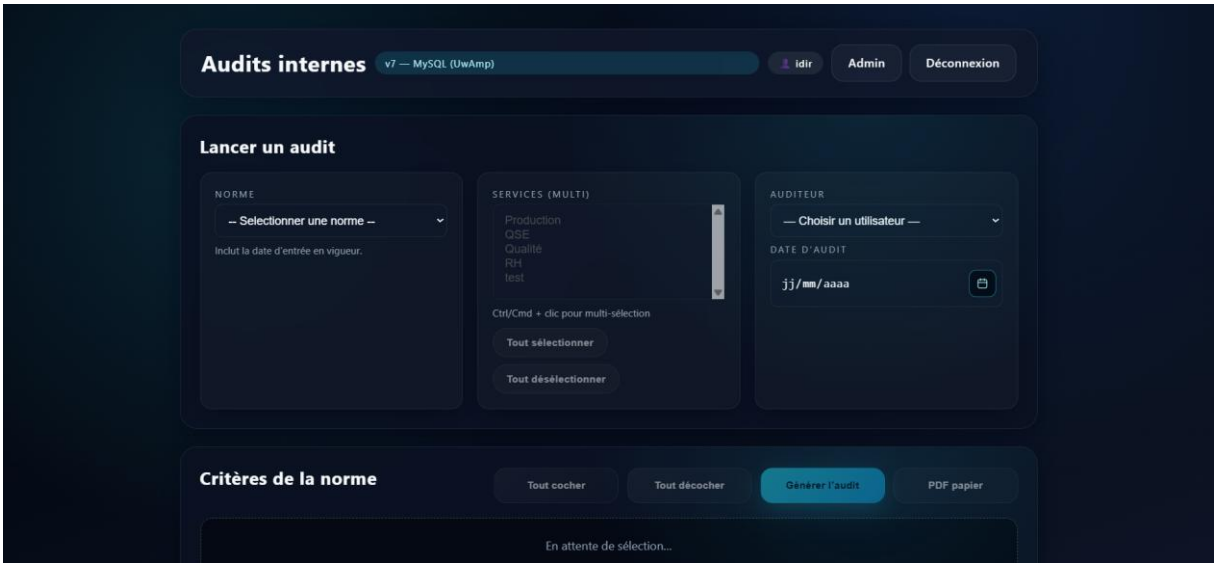
1. Une liste déroulante des **normes disponibles**
2. Une liste de **services** (multi-sélection)
3. Un tableau des **lignes correspondantes**, contenant :
    - Code
    - Intitulé
    - Service concerné
    - Obligation (Oui/Non)
    - Poids
    - Case à cocher "Inclure dans l'audit"
4. Les **informations d'en-tête** de l'audit :

- Date d'audit
- Auditeur

5. Un bouton **"Générer l'audit"** Crée automatiquement :
   - un enregistrement dans audit
   - une entrée dans audit_resultat pour chaque ligne cochée

- **Visualisation dynamique des lignes**
- Le tableau se met automatiquement à jour selon :
  - la norme choisie
  - les services sélectionnés
- **Comportement attendu**

Action	**Résultat**
Choisir norme	Active liste des services
Choisir services	Filtre les lignes
Cocher lignes	Active le bouton "Générer audit"
Générer audit	Enregistre tout en BD et ouvre l'édition



2.1.2 Page 'Administration / Détail d'un audit'

Edition/consultation des résultats d'un audit existant.

Cette page permet :

- **de consulter ou modifier les résultats d'un audit**

Elle contient :

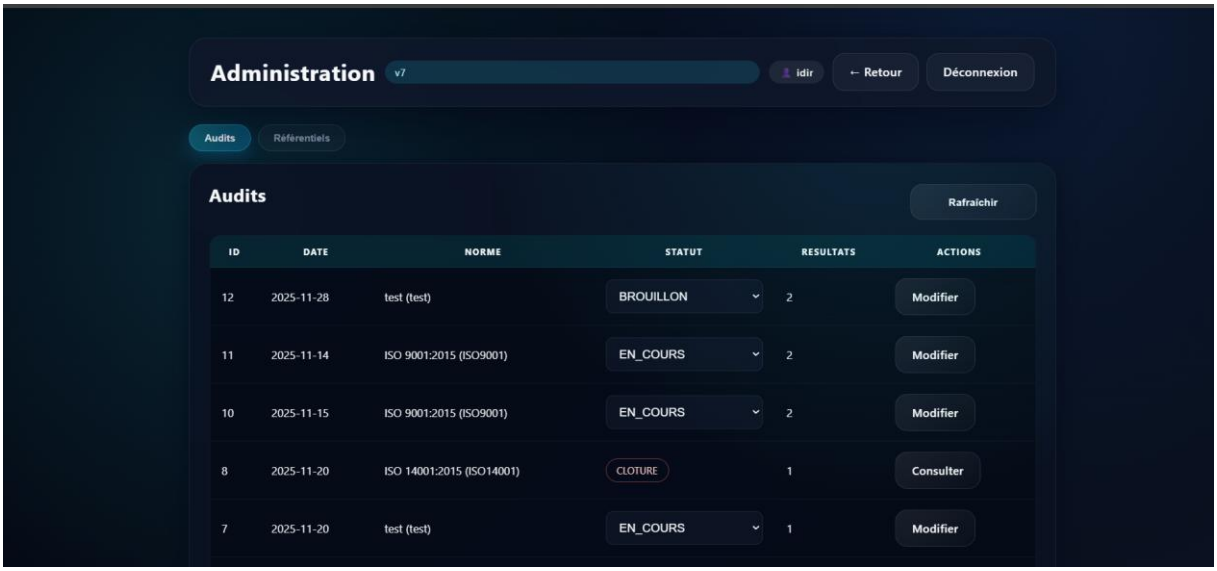
- Informations de l'audit :

    - Date
    - Norme
    - Auditeur
    - Services concernés
    - Statut (modifiable ou pas selon conditions)
- Tableau complet des résultats :
    - Code
    - Intitulé de la ligne
    - Service
    - Statut
    - Commentaire
    - Preuve
    - Score

Règles métiers importantes

Statut de l'audit	Comportement
BROUILLON	Modifiable
EN_COURS	Modifiable
CLOTURE	Non modifiable (consultation uniquement)

Le bouton "Enregistrer" apparaît **uniquement** si le statut ≠ "CLOTURE"



2.1.3 Panneau ‘Administration / Référentiels’

Le module Référentiels permet aux administrateurs fonctionnels de gérer l’ensemble des éléments nécessaires à la réalisation d’audits :

- Normes
- Services
- Lignes (critères)
- Modification des lignes existantes

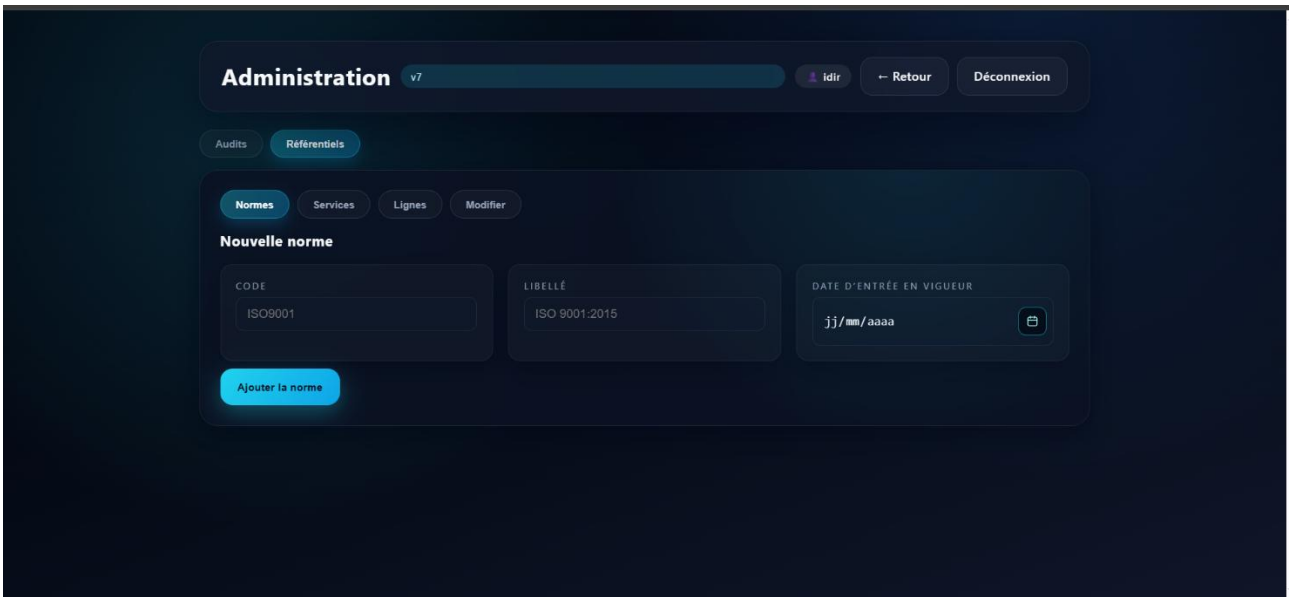
La section Référentiels est divisée en quatre sous-sections accessibles via des onglets :

1. **Gestion des Normes**

La page affiche un formulaire simple composé de :

- Code
  Exemple : ISO9001
- Libellé
  Exemple : ISO 9001:2015
- Date d’entrée en vigueur
  Format : jj/mm/aaaa
  Un sélecteur de date est disponible (icône calendrier)
- 3.3 Fonctionnement

1. L’administrateur saisit les informations.
2. Il clique sur “Ajouter la norme”.
3. L'API enregistre la norme dans la base SQL.
4. La norme devient immédiatement disponible :
   - Dans l’onglet “Lignes”
   - Dans la création d’audit côté utilisateur



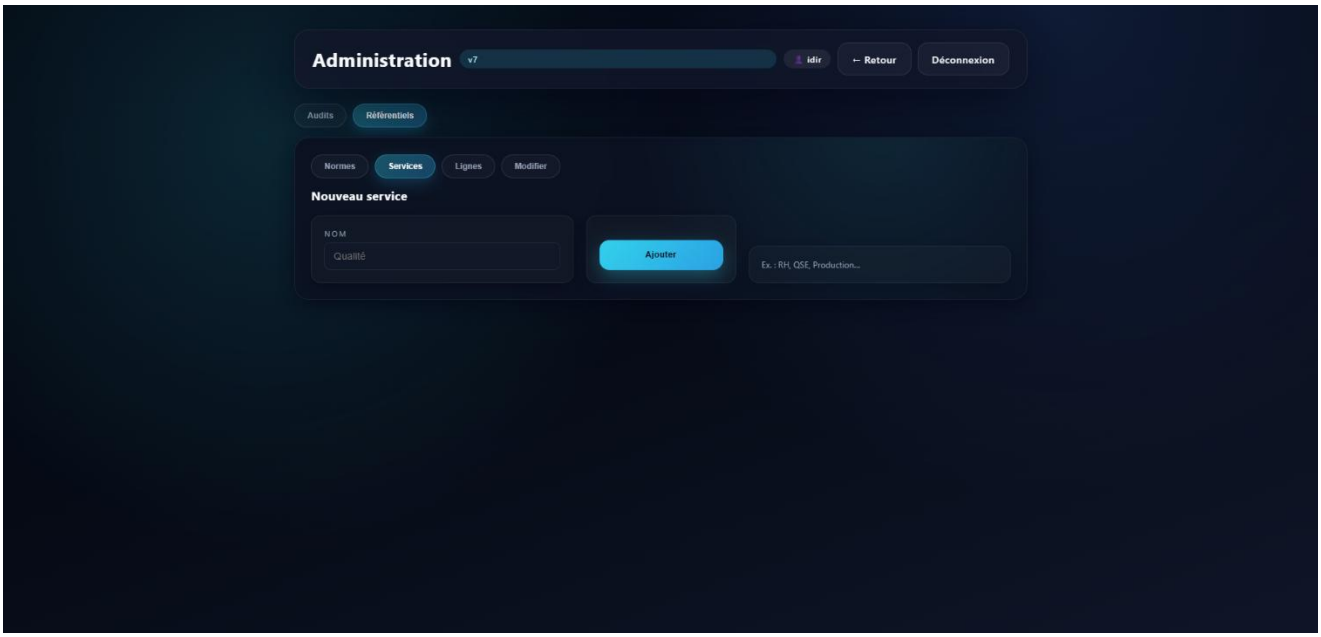
2. **Gestion des Services**

Créer la liste des services qui peuvent être concernés par les audits.

Exemples : Production / Qualité.

Fonctionnement

1. Saisie du nom du service
2. Clic sur “Ajouter”
3. Le service s’enregistre en base
4. Il devient disponible :
    - Dans les lignes (critères)
    - Dans les audits



3. **Création de Lignes (Critères)**

Les lignes correspondent aux critères d’audit détaillés d’une norme.
Ce sont les éléments que les auditeurs verront ligne par ligne dans un audit.

L’interface comporte :

- **Sélecteur de norme**
  Liste déroulante obligatoire.
- **Liste multi-sélection des services concernés**
    - Cases à cocher
    - Si aucune sélection → la ligne est créée pour **tous** les services.
    - Texte explicatif : *“Laisser vide → la ligne sera créée pour tous les services.”*

- **Champs ligne par ligne** :
    - **Code** (ex: 4.1)
    - **Intitulé** (ex: “Maîtrise de la production”)
    - **Obligatoire** (Oui / Non)
    - **Poids** (ex : 1.00)
    - **Ordre** (entier, pour afficher les lignes dans le bon ordre)
- **Bouton “Ajouter la ligne”**

Comportement fonctionnel :

- **Si plusieurs services sont cochés**

L’outil crée **une ligne par service**.
Exemple : Production + Qualité → 2 lignes créées.

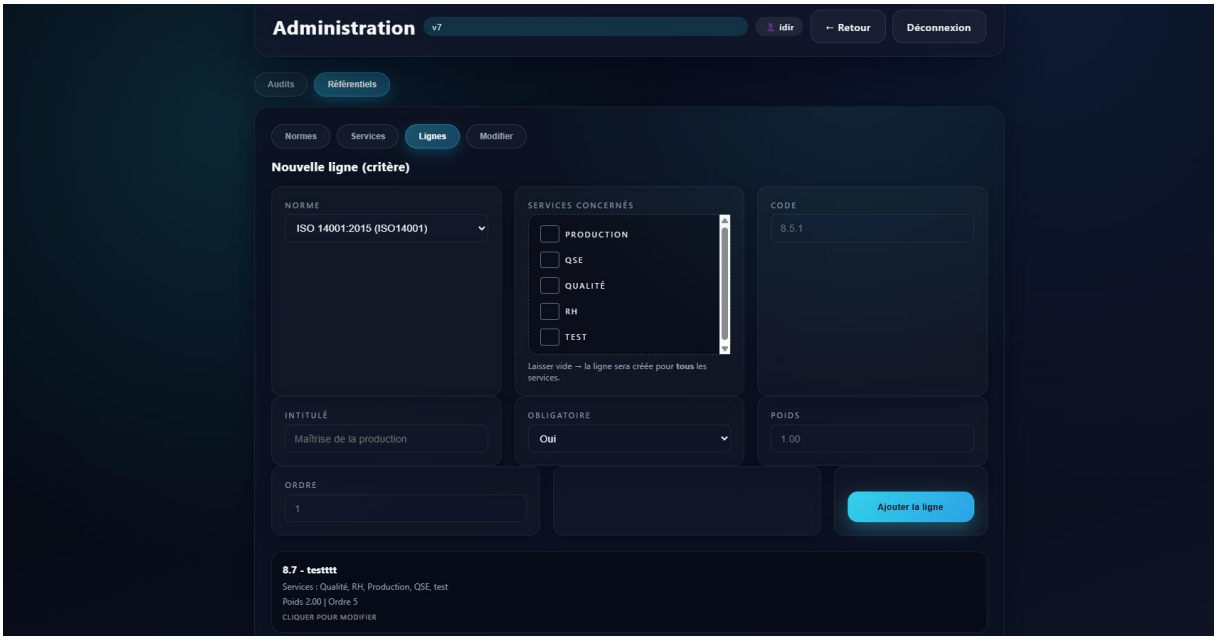
- **Si aucun service n’est sélectionné**

L’outil crée **une ligne pour chaque service existant**, automatiquement.

- **Affichage en bas des lignes créées (aperçu)**

Une prévisualisation des lignes déjà existantes s’affiche en bas de la page :

- Code
- Intitulé
- Services associés
- Poids
- Ordre
- Lien clickable : “Cliquer pour modifier”



4. **Modifier les Lignes**

Cette section permet de modifier les lignes existantes d'une norme.

Filtres :

- **Norme** (obligatoire)
- **Service** (optionnel : Tous / Production / RH / etc.)
- Bouton **Actualiser**

Après sélection, un tableau apparaît avec :

- **Code**
- **Intitulé**
- **Services concernés** (cases cochées/décochées)
- **Obligatoire** (Oui/Non)
- **Poids**
- **Ordre**
- Bouton **Mettre à jour**

Fonctionnement

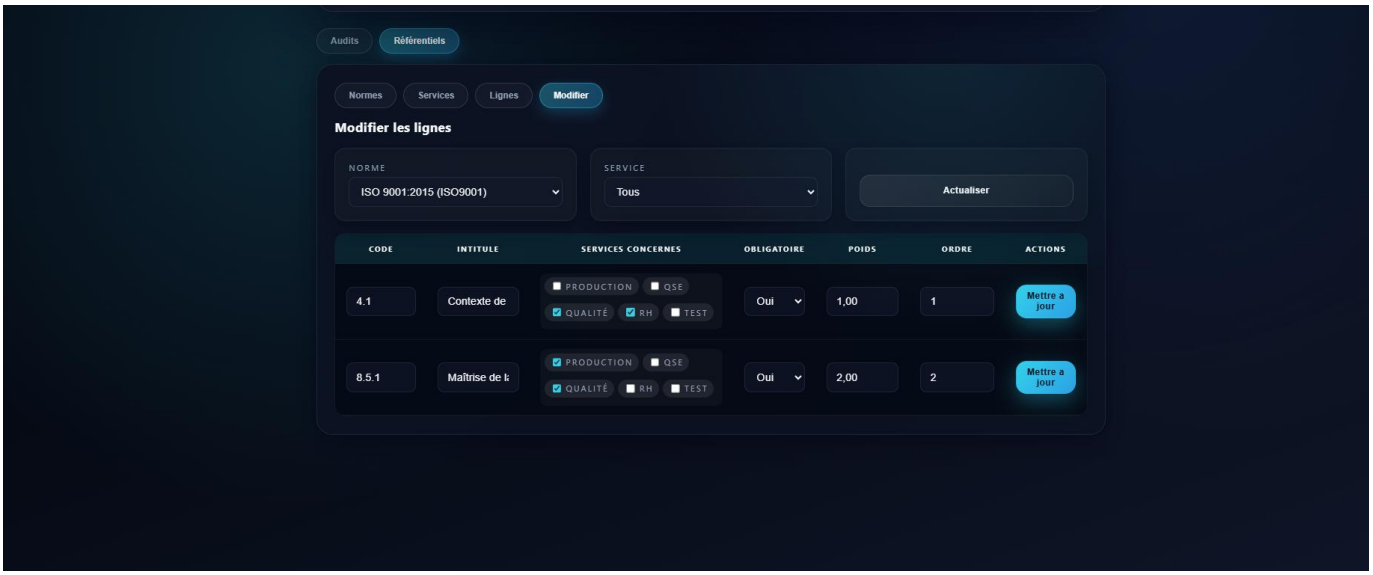
Pour chaque ligne :

1. L'utilisateur peut :
    - Changer le code
    - Modifier l'intitulé
    - Ajuster les services associés (ajout ou retrait)
    - Changer obligatoire Oui/Non
    - Ajuster le poids
    - Ajuster l'ordre

    Clic sur **Mettre à jour**

Les modifications sont **immédiatement visibles** dans :

- La création d'audit.
- Les audits non clôturés.



3 Documentation technique

3.1 Architecture générale

3.1.1 Arborescence

```
Projet_audit/
├── admin.php                # Interface d'administration (audits + référentiels)
├── edit_audit.php           # Page de saisie/détail d'un audit existant
├── index.php                # Page principale de préparation des audits
├── login.php / register.php  # Authentification / inscription
├── logout.php
├── css/
│   └── style.css            # Feuilles de styles communes
├── js/
│   ├── app.js               # Logique de la page principale
│   ├── admin.js             # Logique de l'interface admin
│   ├── edit_audit.js        # Logique de la page audit détaillé
│   ├── pdf_template.js      # Générateur partagé de PDF (mise en page)
│   ├── jspdf.umd.min.js     # Bibliothèque jsPDF
│   ├── login.js / register.js# Scripts optionnels des formulaires d'accès
├── libs/                    # Répertoire réservé pour éventuelles dépendances
├── api/
│   ├── db.php               # Connexion PDO MySQL
│   ├── session.php          # Démarrage session
│   ├── auth_guard.php       # Vérification d'authentification (HTTP 401 sinon)
│   ├── password_helper.php  # Hash / vérification des mots de passe
│   ├── login.php / register.php / logout.php
│   ├── get_normes.php / add_norme.php
│   ├── get_services.php / get_services_all.php / add_service.php
│   ├── get_lignes.php / get_lignes_grouped.php / add_ligne.php / add_ligne_multi.php / update_ligne.php
│   ├── create_audit.php / get_audits.php / get_audit.php
│   ├── save_audit.php / update_audit_status.php
│   └── get_users.php
└── bdd_schéma.txt           # Script SQL (création des tables + données exemples)
```

3.1.2 Technologies & principes

- Front-end : HTML5, CSS3, JavaScript vanilla. Les pages sont statiques, enrichies par les scripts JS qui consomment l'API via fetch.
- Exports PDF : jsPDF (js/jspdf.umd.min.js) + un gabarit interne (pdf_template.js) pour dessiner les tableaux/stats.
- Back-end : PHP 8.x (fonctionne sous UwAmp/WAMP/IIS+PHP). Toutes les API passent par PDO (MySQL) et sont protégées par auth_guard.php.
- Authentification : sessions PHP ; les mots de passe sont salés/hachés (SHA-256) dans password_helper.php.
- Architecture logique :
  - index.php → js/app.js (préparation audit, filtrage, création).
  - admin.php → js/admin.js (gestion référentiels + audits).
  - edit_audit.php → js/edit_audit.js (saisie détaillée + finalisation).

3.2 Base de données

3.2.1 Tables

Norme /

id INT PK AI

code VARCHAR(50) UNIQUE

libelle VARCHAR(255)

description TEXT (facultatif) + date_norme (utilisé côté PHP)

service /

id INT PK AI

nom VARCHAR(150) UNIQUE

description TEXT

utilisateur /

id INT PK AI

username UNIQUE, display_name, password_hash, role (USER/ADMIN), created_at

lignes /

id INT PK

norme_id FK → norme (ON DELETE CASCADE)

service_id FK → service (ON DELETE RESTRICT)

code, intitule, description, obligatoire (TINYINT), poids DECIMAL, ordre INT

audit /

id INT PK

norme_id FK → norme (ON DELETE RESTRICT)

date_audit DATE, auditeur, scope, statut ENUM(BROUILLON, EN_COURS, CLOTURE)

audit_resultat /

id INT PK

audit_id FK → audit (ON DELETE CASCADE)

ligne_id FK → lignes (ON DELETE RESTRICT)

statut ENUM(CONFORME, NON_CONFORME, NA, OBS)

score DECIMAL, commentaire, preuve

Contrainte UNIQUE (audit_id, ligne_id)

3.3 APIs & flux

3.3.1 Auth / session

Fichier	Méthode	Description
api/login.php	POST JSON { username, password }	Vérifie utilisateur, initialise $_SESSION. Renvoie { ok, user } ou Identifiants incorrects.
api/register.php	POST JSON { username, display_name, password }	Crée un utilisateur (contrôle regex + longueur). Enregistre la session et renvoie { ok, user }.
api/logout.php	POST/GET	Ferme la session, renvoie { ok: true }.
api/auth_guard.php	*inclus*	Vérifie $_SESSION['user_id'], sinon 401 + JSON { ok:false, error:'Authentification requise' }.

3.3.2 Référentiels

- **Normes** : get_normes.php (GET), add_norme.php (POST JSON {code, libelle, date_norme?}) — validations sur les champs obligatoires et format de date.
- **Services** : get_services_all.php, add_service.php (nom requis). get_services.php?norme_id retourne uniquement les services ayant des lignes dans cette norme.
- **Lignes** :
    - get_lignes.php?norme_id&service_ids? : liste brute (page principale).
    - get_lignes_grouped.php : agrégation (admin) → structure { line_ids[], service_ids[], services[], code, intitule, … }.
    - add_ligne.php : insertion unitaire.
    - add_ligne_multi.php : duplication pour tous les services cochés (si liste vide, insertion pour tous les services existants).
    - update_ligne.php : mise à jour massive + recalcul des associations services (supprime/ajoute selon les cases cochées).

3.3.3 Audits

- create_audit.php :
    - Entrée { norme_id, date_audit?, auditeur?, ligne_ids[] }.
    - Contrôle : toutes les lignes doivent appartenir à la norme.
      Crée audit (statut EN_COURS) + audit_resultat (statut NA par défaut).
- get_audits.php :
    - Liste des audits pour l'écran Admin (ID, date, statut, auditeur, norme, nb_resultats).
- get_audit.php?id=… :

    - Détails entête + résultats (jointure sur lignes et service).
- save_audit.php :
    - type=header → met à jour date_audit, auditeur, statut (si statut final ≠ CLOTURE). Validation du format date.
    - type=results → resultats: [{ ligne_id, statut, score, commentaire, preuve }]. Refuse les statuts hors liste, ignore les entrées invalides, rollback si mix invalide.
- update_audit_status.php :
    - Couche admin pour changer rapidement le statut (BROUILLON/EN_COURS/CLOTURE) tant que l'audit n'est pas déjà cloturé.
- get_users.php :
    - Alimente la liste d'auditeurs (display_name + username).

Tous les endpoints renvoient { ok: true/false, data?, detail?, error? }. Les erreurs sont propagées aux scripts JS via requestJson (alert utilisateur).

3.4 Logique front-end

3.4.1 js/app.js (Accueil)

- state: mémorise norme sélectionnée, services, lignes chargées, lignes cochées, audit courant.
- loadInitialData() : charge normes, services, utilisateurs en parallèle.
- refreshLignes() : récupère get_lignes en fonction des serviceIds sélectionnés.
- generateAudit() : appels create_audit, puis construit une table de saisie locale (sans rechargement complet).
- saveAudit() : compose un tableau [{ ligne_id, statut, score, commentaire, preuve }] et POST save_audit.
- setServiceSelection() / toggleAllRows() : manipulent state.selectedLigneIds.
- exportPdf() :
    - Prépare l'en-tête (auditLabel, norme, services, date, auditeur).
    - Filtre les lignes (cochées ou toutes) et appelle buildAuditSheet(..., showStatusMarks=false).

3.4.2 js/admin.js

- setupTabs() / setupSubtabs() : navigation.
- loadAudits() : dessine le tableau des audits + écouteurs sur <select class="adm-status">.
- populateRefs() : remplit les <select> et checklist (services).
- loadEditableLignes() : charge get_lignes_grouped, dessine un tableau éditable (inputs, checkboxes par service). Lors d'une carte cliquée dans l'aperçu (refreshLignesPreview), l'onglet "Modifier" s'ouvre sur la ligne correspondante.

- updateLigneFromRow() : récupère les valeurs d'une ligne, appelle api.updateLigne.

3.4.3 js/edit_audit.js

- fetchAuditData() : charge simultanément les utilisateurs et l'audit ciblé, remplit l'entête + résultats, active/désactive les contrôles selon le statut.
- saveHeader() / saveResults() / finalizeAudit() : wrappers save_audit.
- exportAuditPdf() : convertit les résultats en format buildAuditSheet(..., showStatusMarks=true) (statuts cochés + commentaires/préuves intégrés).

3.4.4 js/pdf_template.js

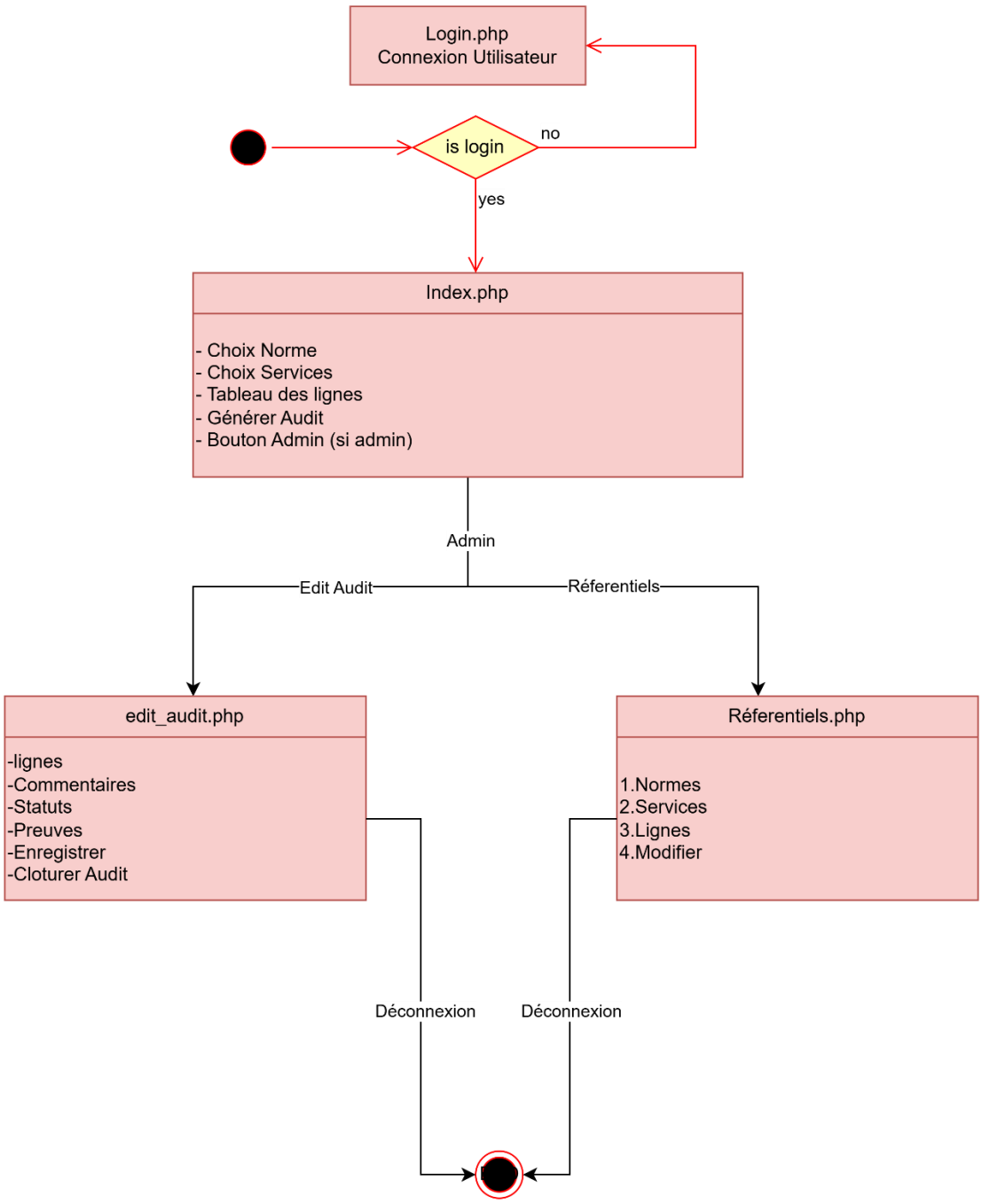
- buildAuditSheet(doc, { header, rows, showStatusMarks }) :
    - Détermine les colonnes : Code (28mm), Critère (48mm), Service (36mm), Statuts (40mm), Commentaires (reste).
    - Dessine l'en-tête (titre, audit, norme, services, date, auditeur, statut).
    - Pour chaque ligne : rectangle + texte + colonne statuts (cases ☐, croix si showStatusMarks et statut correspondant) + colonne commentaires vide ou remplie.
    - Gère la pagination (Ajout doc.addPage() si la hauteur restant < margin).

3.5 Points d'attention & bonnes pratiques

- Sessions / auth : toutes les pages (sauf login/register) require api/session.php et redirigent si user_id absent.
- Validation côté serveur : chaque API vérifie les paramètres (format date, liste blanche de statuts, etc.).
- Transactions : create_audit, save_audit(type=results) et update_ligne utilisent BEGIN / COMMIT pour garantir la cohérence.
- Multi-services : ne jamais modifier manuellement la table lignes sans respecter la contrainte "une ligne = norme + service". Utiliser add_ligne_multi / update_ligne.
- Exports : la largeur des colonnes est définie dans pdf_template.js; toute modification d'alignement se fait à ce niveau.

Déploiement IIS :

- api/db.php doit être adapté (host/user/mot de passe).
- Base MySQL importée via bdd_schéma.txt.


Login.php
Connexion Utilisateur
is login
no
yes
Index.php
- Choix Norme
- Choix Services
- Tableau des lignes
- Générer Audit
- Bouton Admin (si admin)
Admin
Edit Audit
Réferentiels
edit_audit.php
-lignes
-Commentaires
-Statuts
-Preuves
-Enregistrer
-Cloturer Audit
Réferentiels.php
1.Normes
2.Services
3.Lignes
4.Modifier
Déconnexion
Déconnexion