Documentation fonctionnelle & technique

Plateforme ASGF

Version 1.0 – Dernière mise à jour : 02/12/2025

Projet : association-asgf.fr (`https://association-asgf.fr/`)

1. Vue d'ensemble

La plateforme ASGF est composée de :

- **Frontend** : application React + Vite (`asgf-app/`) déployée sur Firebase Hosting (`https://asgf-siteweb.web.app`), accessible via le domaine `https://association-asgf.fr/` .
- **Backend / API** :
  - **Supabase Edge Functions** (Deno / TypeScript) pour la plupart des fonctionnalités métier (adhésions, formations, webinaires, trésorerie, mentorat, secrétariat, etc.).
  - Un **backend Node/Express** historique (`backend/`) utilisé principalement pour certaines intégrations spécifiques (scripts SQL, tâches batch, etc.).
- **Base de données** : PostgreSQL fournie par Supabase, organisée par **schémas** fonctionnels :
  - `public` (contenu public : contact, projets, etc.)
  - `adhesion` , `formation` , `webinaire` , `tresorerie` , `mentorat` , `recrutement` , `secretariat`
  - `auth` (comptes utilisateurs / admins via Supabase Auth).
- **Google Apps Script** : `apps-script/contact_webhook.gs` pour l'envoi des e-mails (contact public, inscriptions, notifications, e-mails membres, génération de PDF vers Google Drive, etc.).

2. Frontend – Architecture générale

Dossier : `asgf-app/src/`

- `main.jsx` : point d'entrée React.
- `App.jsx` : composition des layouts publics / admin.
- `router/` :
  - `router/index.jsx` : définition du `BrowserRouter`.
  - `router/publicRoutes.jsx` : routes publiques.
  - `router/adminRoutes.jsx` : routes du back-office.
- `layouts/` :
  - `PublicLayout.jsx` : squelette des pages publiques (header, navbar, footer, contenu).
  - `AdminLayout.jsx` : squelette du back-office (sidebar, topbar, contenu central).
- `components/` :
  - `ScrollToTop.jsx` : remonte en haut lors des changements de route.

Les styles globaux sont gérés via :

- `index.css`, `App.css` : base CSS globale.
- `public/styles/style.css`, `public/styles/responsive.css`, `public/styles/variables.css` : thème public (grilles, hero, about, missions, etc.).
- `admin/index.css`, `admin/App.css`, `admin/pages/AdminDashboard.css`, `admin/pages/AdminLogin.css`, `admin/components/AdminSettingsPanel.css`, etc. : thème du back-office (dashboard, cartes, tableaux, modals…).

3. Module Public – Pages & flux

3.1. Page d'accueil – `Home.jsx`

Fichier : `src/public/pages/Home.jsx`
URL : `/`

Objectif :

- Présenter l'ASGF, ses missions, activités et partenaires.
- Permettre à l'utilisateur :
  - de découvrir les missions via les onglets ("Découvrir nos missions"),
  - de naviguer vers les autres pages (Adhésion, Formations, Webinaires, Projets, Bureau),
  - de contacter l'association via le formulaire de contact.

Sections principales :

- **Header / Navbar** : `public/components/Navbar.jsx`
- **Hero** : slogan + bouton "Découvrir nos missions" (scroll vers la section d'onglets).
- **Section à onglets** (dans `Home.jsx` + `PageStyles.jsx`) :
  - `À propos` : texte de présentation de l'association.
  - `Missions` : cartes listant les missions (mise en avant).
  - `Activités` : activités récurrentes.
  - `Partenariats` : partenaires clés (logos, descriptions).
- **Section Contact** :
  - Formulaire avec champs `fullName` , `email` , `subject` , `message` .

Intégration back-end :

- **Contact** :
  - `PUBLIC_CONTACT_URL = ${VITE_SUPABASE_URL}/functions/v1/public-contact`
  - `handleContactSubmit` fait un `fetch` `POST` sur `/functions/v1/public-contact/messages` .

Edge Function liée :

- `supabase/functions/public-contact/index.ts`

- Point d'entrée HTTP (sans JWT, `--no-verify-jwt`).
- Routes principales :
  - `POST /messages` : enregistre le message dans `public.contact_messages` et envoie 2 e-mails via `contact_webhook.gs` (confirmation au visiteur + notification ASGF).
  - `GET /messages` (avec JWT admin) : liste paginée des messages.
  - `PATCH /messages/:id` : mise à jour du statut (lu, traité).
  - `DELETE /messages/:id` : suppression.
- Utilisation du client `supabasePublic` pour écrire dans `public.contact_messages`.

3.2. Page d'adhésion – `Adhesion.jsx`

Fichier : `src/public/pages/Adhesion.jsx`
URL : `/adhesion`

Objectif :

- Permettre à un visiteur de déposer une demande d'adhésion en tant que membre de l'ASGF.

Formulaire :

- Informations personnelles : `nom`, `prenom`, `email`, `telephone`, `dateNaissance`, `adresse`, `ville`, `pays`, `paysAutre`.
- Informations académiques et pro : `universite`, `niveau`, `annee`, `specialite`, `statut_pro`, `domaine`.
- Intérêts & motivation : `interets[]`, `motivation`, `competences`.
- Options : `conditions` (obligatoire), `newsletter`.

Validation :

- Vérifications côté front dans `validate()` (champs obligatoires, format e-mail, etc.).

Intégration back-end :

- Utilise `supabase` défini dans `public/config/supabase.config.js`.
- Insertion dans `adhesion.members` via Supabase (Ex: `supabase.from('members').insert(...)`).
- En parallèle, un e-mail de confirmation est déclenché côté back via Supabase (trigger ou Edge Function) :
  - **Edge function** : `admin-adhesion-members` (voir plus bas) traite les inscriptions et envoie les e-mails via `contact_webhook.gs` (évènement `FORMATION_INSCRIPTION`, `MEMBER_EMAIL`, etc. selon les flux).

Page de succès :

- `src/public/pages/AdhesionSuccess.jsx` : affiche un message de confirmation après une adhésion validée côté front.

Tables associées (schéma `adhesion`) :

- `adhesion.members`
  - Colonnes principales : `id`, `numero_membre`, `prenom`, `nom`, `email`, `telephone`, `pays`, `date_naissance`, `universite`, `niveau_etudes`, `annee_universitaire`,

`specialite`, `interets`, `motivation`, `competences`, `statut_pro`, `domaine`, `is_active`, `conditions_acceptees`, `created_at`, `updated_at`.

3.3. Pages Formations – `Formation.jsx`, `InscriptionFormation*.jsx`

- **Liste des formations** : `public/pages/Formation.jsx`
  - Récupère les formations actives via `supabaseFormation` (`public/config/supabase.config.js`, schéma `formation.formations` + `formation_formateurs`, `formateurs`).
  - Affiche cartes de formation (titre, description, niveau, catégorie, formateurs).
- **Inscription à une formation** : `public/pages/InscriptionFormation.jsx`
  - Formulaire d'inscription pour une formation / session donnée.
  - Envoi des données à l'Edge Function :
    - Front : `supabaseFormation.from('inscriptions').insert(...)`.
    - Back : `supabase/functions/public-formation-inscription/index.ts`.
      - Écrit dans `formation.inscriptions`.
      - Déclenche l'e-mail de confirmation via `contact_webhook.gs` (`FORMATION_INSCRIPTION`).
- **Page de succès** : `InscriptionFormationSuccess.jsx`

Tables (`formation`) :

- `formation.formations`, `formation.sessions`, `formation.inscriptions`, `formation.formateurs`, `formation.formation_formateurs`.

Edge Functions :

- `admin-formation` : gestion des formations, sessions, formateurs, inscriptions, exports (appelée par l'admin).
- `public-formation-inscription` : endpoint public pour les inscriptions.
- Notifications e-mail via `contact_webhook.gs` avec types :
  - `FORMATION_INSCRIPTION`, `FORMATION_STATUS`, `FORMATION_INVITATION`, `FORMATION_REMINDER`.

3.4. Pages Webinaires – `Webinaires.jsx` , `InscriptionWebinaire*.jsx`

- **Liste des webinaires** : `public/pages/Webinaires.jsx`
  - Récupère `webinaire.webinaires` + `presentateurs` .
- **Inscription** : `InscriptionWebinaire.jsx`
  - Envoi vers `public-webinaire-inscription` (Edge Function).
  - Insertion dans `webinaire.inscriptions` .
  - E-mails via `WEBINAIRE_INSCRIPTION` , `WEBINAIRE_STATUS` , `WEBINAIRE_INVITATION` , `WEBINAIRE_REMINDER` .
- **Page de succès** : `InscriptionWebinaireSuccess.jsx` .

3.5. Page Projets – `Projets.jsx`

- Liste dynamique des projets publics (schéma `public.projets` + `public.projets_inscriptions`).
- Edge Function `projet-inscription` pour gérer les inscriptions / manifestations d'intérêt.

3.6. Page Bureau – `Bureau.jsx`

- Charge les membres du bureau via `supabase/functions/public-bureau` :
  - `public-bureau/index.ts` lit dans `secretariat.bureau` / `secretariat.roles` (selon le schéma implémenté).
- Affiche les cartes membres du bureau (nom, rôle, contacts).

4. Module Admin – Back-Office global

4.1. Authentification admin – `AdminLogin.jsx` / Edge function `admin-login`

- **Fichier** : `src/admin/pages/AdminLogin.jsx` , `AdminLogin.css`
  - Formulaire e-mail + mot de passe.
  - Gestion du scroll mobile (classe `login-page-active` sur `<body>`).
- **Back-end** :
  - `supabase/functions/admin-login/index.ts`
    - Vérifie les identifiants via Supabase Auth.
    - Charge les rôles / modules autorisés depuis `public.admins` (`sql_grant_permissions_*.sql`).
    - Retourne un JWT admin utilisé par le front pour appeler les autres Edge Functions (header `Authorization: Bearer <token>`).

4.2. Dashboard global – `AdminDashboard.jsx` + `admin-dashboard` (Edge)

Fichier : `src/admin/pages/AdminDashboard.jsx`
Styles : `admin/pages/AdminDashboard.css`

Fonctions :

- **Vue d'ensemble** :
  - KPIs : nombre total de membres, adhésions en attente, cotisations en retard, nombre de webinaires, formations, actions de secrétariat, etc.
  - Graphiques : répartition par pays, par statut, timeline des cotisations, dépenses, webinaires, etc.
- **Carte des membres** :
  - `react-leaflet` + `react-leaflet-cluster`.

Back-end :

- Edge Function `admin-dashboard-stats/index.ts` :
  - Agrège les statistiques depuis :
    - `adhesion.members`,
    - `tresorerie.cotisations`, `tresorerie.paiements`, `tresorerie.depenses`, `tresorerie.cartes_membres`,
    - `formation.*`, `webinaire.*`, `secretariat.*`, `mentorat.*`, `recrutement.*`.
  - Retourne des objets typés (`mentorat`, `recrutement`, `tresorerieStats`, `secretariatStats`, etc.).

4.3. Module Membres / Adhésions – `AdminDashboard.jsx` (onglet "Membres") + `admin-adhesion-members`

Front :

- L'onglet "Membres" du dashboard (`AdminDashboard.jsx`) utilise `admin/services/api.js` :
  - `fetchPendingMembers` , `fetchAllMembers`
  - `approveMember` , `rejectMember`
  - `updateMember` , `deleteMember`
  - `createCarteMembre` , `geocodeMemberAddress` , `listCartesMembres` , etc.
- Interface :
  - Tableau des membres (en attente, actifs, etc.).
  - Détail d'un membre (fiche, carte membre, cotisations, formations, webinaires, mentoring…).

Edge Function : `supabase/functions/admin-adhesion-members/index.ts`

Principales responsabilités :

- `doPost` :
  - Parse le JWT (`Authorization: Bearer`).
  - Récupère `supabaseAdhesion` , `supabaseFormation` , `supabaseWebinaire` , `supabaseMentorat` , `supabaseRecrutement` , `supabase.tresorerie` , `supabase.public` .
- **Handlers** :
  - `handleFormationInscription` , `handleFormationStatus` , `handleFormationInvitation` , `handleFormationReminder` (relais vers `contact_webhook.gs`).
  - `handleWebinaireInscription` , `handleWebinaireStatus` , `handleWebinaireInvitation` , `handleWebinaireReminder` .
  - `handleMemberEmail` : envoi d'e-mails en masse aux membres (template HTML avec variables `{{prenom}}` , `{{nom}}` , `{{numero_membre}}` , `{{pays}}`).
  - **CRUD sur** `adhesion.members` :
    - `GET /members` : liste paginée, filtres (statut, recherche, etc.).
    - `POST /members` : création (si utilisé).
    - `PATCH/PUT /members/:id` : mise à jour (notamment pour corriger les infos d'un membre).
    - `DELETE /members/:id` : suppression d'un membre.

- **Gestion des contraintes de clé étrangère** (suppression) :
  - Avant suppression d'un membre, la fonction supprime ou met à `NULL` toutes les références dans :
    - `tresorerie.cartes_membres`, `tresorerie.cotisations`, `tresorerie.depenses`
    - `formation.inscriptions`, `webinaire.inscriptions`, `public.projets_inscriptions`
    - `secretariat.reunions` (`presente_par`), `secretariat.actions` (`assigne_a`), `secretariat.documents` (`cree_par`)
    - `mentorat.mentors`, `mentorat.mentees`, `mentorat.relations`, `recrutement.candidatures`, etc.
- **Synchronisation adhésion ↔ cartes de membre** :
  - Lors d'un `updateMember`, après mise à jour de `adhesion.members`, la fonction met aussi à jour les entrées associées dans `tresorerie.cartes_membres` (actuellement le champ `pays`, via `numero_membre` et/ou `membre_id`).

4.4. Module Trésorerie – `Treasury*` + `admin-tresorerie`

Front :

- Composants principaux (`src/admin/components/treasury/`) :
  - `TreasuryFilters.jsx`
  - `TreasuryActionsBar.jsx`
  - `TreasuryAnalytics.jsx`
  - `TreasuryTimeline.jsx`
- Intégrés dans `AdminDashboard.jsx` via l'onglet "Trésorerie".
- Fonctions front disponibles dans `admin/services/api.js` :
  - `fetchCotisations` , `createCotisation` , `updateCotisation` , `deleteCotisation`
  - `fetchPaiements` , `createPaiement` , `updatePaiement` , `validatePaiement` , `cancelPaiement`
  - `fetchDepenses` , `createDepense` , `validateDepense` , `rejectDepense` , `deleteDepense`
  - `downloadCotisationsExport` , `downloadPaiementsExport` , `downloadTresorerieReport`
  - `generateMonthlyCotisations` , `updateOverdueCotisations` , `createMissingCartes` , etc.

Edge Function : `supabase/functions/admin-tresorerie/index.ts`

- Routes principales (exemples) :
  - `GET /cotisations` , `POST /cotisations` , `PATCH /cotisations/:id` , `DELETE /cotisations/:id`
  - `GET /paiements` , `POST /paiements` , `PATCH /paiements/:id` , etc.
  - `GET /stats` : agrégation des montants, soldes, répartition par période.
  - `POST /cartes` : création d'une carte membre (`tresorerie.cartes_membres`) avec génération de PDF (via `CarteMembreGenerator` + `contact_webhook.gs` / Google Drive).

Table `tresorerie.cartes_membres` :

- Colonnes : `id` , `numero_membre` , `membre_id` (optionnel), `issue_date` , `expiry_date` , `lien_pdf` , `pays` , `created_at` , etc.
- Signature sur la carte : `signature_omar.png` .

- Pied de carte (après modifications) :
  - `Émis le JJ/MM/AAAA`
  - `© association-asgf.fr/`

4.5. Module Projets – `ProjetsContent.jsx` + `admin-projets`

- **Front** : `admin/components/ProjetsContent.jsx`
  - Gestion des projets (création, modification, activation/inactivation).
  - Gestion des inscriptions aux projets (`projets_inscriptions`).
- **Back** : `supabase/functions/admin-projets/index.ts`
  - CRUD sur `public.projets` , `public.projets_inscriptions` .

4.6. Module Mentorat – `RelationDrawer.jsx` + `admin-mentorat`

- **Front** :
  - `admin/components/mentorat/RelationDrawer.jsx` : drawer pour créer/éditer des relations mentor/mentee, rendez-vous, objectifs.
  - L'onglet "Mentorat" dans `AdminDashboard.jsx` :
    - Statistiques : `mentors_actifs` , `mentees_en_recherche` , `relations_actives` , `total_rendezvous` .
    - Table des binômes + actions (+ création de relation, mentor, mentoré, rendez-vous).
- **Back** : `supabase/functions/admin-mentorat/index.ts`
  - Tables : `mentorat.mentors` , `mentorat.mentees` , `mentorat.relations` , `mentorat.rendez_vous` .

4.7. Module Recrutement – `RecrutementDashboard.jsx` + `recrutement/*`

- **Front** :
  - `admin/pages/RecrutementDashboard.jsx`
  - Composants :
    - `AjouterCandidatureModal.jsx`
    - `AjouterRecommandationModal.jsx`
    - `AjouterSuiviModal.jsx`
    - `CandidatureDrawer.jsx`
- **Back** : `supabase/functions/admin-recrutement/index.ts`
  - Tables : `recrutement.offres` , `recrutement.candidatures` , `recrutement.recommandations` , `recrutement.suivis` .

4.8. Module Secrétariat – `SecretariatDashboard.jsx` + `secretariat/*`

- **Front** :
  - `admin/pages/SecretariatDashboard.jsx`
  - Composants :
    - `secretariat/ReunionDrawer.jsx` : création/édition de réunions, comptes rendus, présence.
    - `secretariat/ReunionTimeline.jsx` , `KPICard.jsx` , `StatusBadge.jsx` , `EmptyState.jsx` .
- **Back** : `supabase/functions/admin-secretariat/index.ts`
  - Tables : `secretariat.reunions` , `secretariat.participants` , `secretariat.actions` , `secretariat.documents` .

4.9. Module Paramètres – `AdminSettingsPanel.jsx` + `admin-parametres`

- **Front** :
  - `admin/components/AdminSettingsPanel.jsx` + `AdminSettingsPanel.css`
  - Permet de gérer :
    - les comptes **administrateurs** (création, suspension, désactivation, réactivation),
    - les droits d'accès par **module** (`adhesion` , `tresorerie` , `formation` , `webinaire` , `mentorat` , `recrutement` , `secretariat` , `audit` , etc.).
- **Back** : `supabase/functions/admin-parametres/index.ts`
  - Tables : `public.admins` , `public.admin_modules` , `public.audit_logs` .

5. E-mails & Notifications – `apps-script/contact_webhook.gs`

Fichier : `apps-script/contact_webhook.gs`

Rôle :

- Point d'entrée Web App Google Apps Script (`doPost` , `doGet`).
- Reçoit des événements depuis les Edge Functions Supabase :
  - `CONTACT` , `FORMATION_INSCRIPTION` , `FORMATION_STATUS` , `FORMATION_INVITATION` , `FORMATION_REMINDER` ,
  - `WEBINAIRE_*` ,
  - `MEMBER_EMAIL` ,
  - `upload_pdf` , `find_pdf_file` (gestion des fichiers PDF sur Google Drive).

Principales fonctions :

- `handleContactMessage` :
  - Envoie 2 e-mails :
    - au visiteur (confirmation),
    - à l'ASGF (`association.geomaticiens.sf@gmail.com`).
  - Utilise `buildHtmlEmail` pour formater le HTML.
  - Version texte (`plainText`) nettoyée via `stripHtmlToPlain` (suppression des `<br>` , balises HTML, `????` , etc.).
- `handleFormation*` et `handleWebinaire*` :
  - Gèrent les scénarios d'inscription, validation, invitation, rappels (48h / 2h).
  - Utilisation systématique de `buildHtmlEmail` pour le HTML + `normalizePlainText` + `stripHtmlToPlain` pour la partie texte.
- `handleMemberEmail` :
  - Envoi d'e-mails en masse à une liste de `recipients[]` :
    - Body basé sur un `bodyTemplate` HTML avec placeholders `{{prenom}}` , `{{nom}}` , `{{numero_membre}}` , `{{pays}}` .
  - `rendered` (HTML) → `htmlBody` sécurisé (`sanitizeHtml`) pour autoriser seulement certaines balises (`strong` , `em` , `br` , `a` , etc.).
  - `plainText` généré via `stripHtmlToPlain` (aucune balise visible).
  - Support des pièces jointes via `attachments[]` (base64 → blob → Gmail).

- `buildHtmlEmail(params)` :
  - Gabarit commun (header bleu, contenu, footer ASGF).
  - `sanitize` (texte) et `sanitizeHtml` (HTML autorisé).
- **Helpers** :
  - `stripHtmlToPlain(input)` :
    - Décodage d'entités (`<` , `>` , ` ` , `&`),
    - Conversion des `<br>` / `</p>` en `\n` / `\n\n` ,
    - Suppression des balises restantes,
    - Nettoyage des séquences `????` (encodage),
    - Normalisation des sauts de ligne.
  - `normalizePlainText(input)` : normalisation simple (\r\n → \n, suppression de `????`).

> **Tous les e-mails envoyés sont donc cohérents** :
>
> - HTML propre, avec seulement les balises autorisées.
> - Texte brut lisible, sans `<br>` ni balises visibles.

6. Déploiement & scripts utiles

6.1. Déploiement du frontend (Firebase Hosting)

Depuis `C:\Users\serig\OneDrive\Bureau\sites_asgf\asgf-admin\asgf-app` :

```
npm install          # une seule fois
npm run build        # compilation Vite → dist/
npx firebase deploy --only hosting
```

Le site est servi sur `https://asgf-siteweb.web.app` et relié au domaine `https://association-asgf.fr/` .

6.2. Déploiement des Edge Functions Supabase

Depuis la racine du projet `C:\Users\serig\OneDrive\Bureau\sites_asgf\asgf-admin` :

```
.\supabase.exe functions deploy admin-adhesion-members
.\supabase.exe functions deploy admin-tresorerie
.\supabase.exe functions deploy admin-formation
.\supabase.exe functions deploy admin-webinaire
.\supabase.exe functions deploy admin-mentorat
.\supabase.exe functions deploy admin-recrutement
.\supabase.exe functions deploy admin-secretariat
.\supabase.exe functions deploy admin-parametres
.\supabase.exe functions deploy admin-dashboard-stats
.\supabase.exe functions deploy admin-login
.\supabase.exe functions deploy public-contact
.\supabase.exe functions deploy public-formation-inscription
.\supabase.exe functions deploy public-webinaire-inscription
.\supabase.exe functions deploy public-bureau
.\supabase.exe functions deploy projet-inscription
.\supabase.exe functions deploy geocode
```

> *Des scripts d'aide au déploiement existent également :* `deploy-supabase.ps1`, `deploy-frontend.ps1`, `deploy-simple.ps1`, *etc. (voir les fichiers* `.md` *à la racine pour les procédures détaillées).*

---

7. Pistes d'évolution

- Internationalisation (FR / EN) des pages publiques et des e-mails.
- Découpage plus fin du bundle front (code-splitting des modules admin).
- Ajout d'un module "Statistiques Membres" côté public (carte interactive, filtres par pays / ville).
- Intégration de paiements en ligne pour les cotisations / adhésions (Stripe / PayFesk).