

Application « Tractage »

Spécification technique, modèle de données et synthèse de sécurité

Application web interne de gestion des opérations de **tractage syndical** : planification, suivi et archivage des campagnes de distribution de tracts sur les sites d'entreprise, par des **représentants**.

Document — destiné au responsable sécurité **Hébergement cible** — OVH mutualisé (exposition Internet)
Développement — NAS QNAP interne **Statut données** — partiellement fictives (voir §6)

1. Présentation générale

L'application permet aux **représentants** et aux **administrateurs** de planifier des campagnes de tractage, de suivre les demandes d'entraide entre représentants, de consulter les sites et leurs effectifs, et de gérer un référentiel (sites, établissements, tracts, types d'opération, expertises, comptes).

- **Utilisateurs cibles** : administrateurs (comptes web) et représentants (comptes de terrain).
- **Accès** : réseau interne uniquement — poste client sans accès Internet sortant.
- **Finalité** : coordination opérationnelle des campagnes et de l'entraide entre représentants.

2. Architecture technique

Élément	Détail
Framework	Symfony 7 (PHP 8.2)
Moteur de rendu	Twig (auto-échappement HTML activé par défaut)
ORM	Doctrine — requêtes paramétrées
Hébergement cible (production)	OVH — Hébergement mutualisé (Web) : Apache + PHP-FPM managés par OVH, exposé sur Internet. TLS/Let's Encrypt disponible, version PHP sélectionnable, base MySQL/MariaDB managée fournie par OVH.
Base de données	MariaDB (moteur MyISAM , sans clés étrangères, charset <code>utf8mb4</code>). ⚠ En production OVH, l'hôte/port sont ceux fournis par OVH — la configuration de développement (<code>127.0.0.1:3307</code> du NAS) ne s'applique pas.
Racine web	Doit pointer sur le sous-dossier <code>public/</code> uniquement (le reste du code ne doit pas être servi — voir §7).
Environnement	<code>APP_ENV=prod</code> (mode debug désactivé)
Développement	NAS QNAP interne (nginx + MariaDB natifs) — hors périmètre de production.
Contraintes mutualisé	Pas d'accès système bas niveau : ni firewall/ports personnalisés, ni configuration serveur libre. Durcissement principalement via <code>.htaccess</code> (Apache) et la configuration applicative. Exécution des migrations selon l'offre (SSH sur offres Pro+ / sinon SQL via phpMyAdmin OVH).

Surface front : aucun script tiers

L'application n'inclut **aucun script JavaScript tiers ni CDN** : les bibliothèques front sont auto-hébergées (ex. Leaflet dans `public/leaflet/`) et les ressources externes (tuiles cartographiques) sont récupérées côté serveur. Historiquement motivé par un réseau interne sans Internet, ce choix reste un **facteur de durcissement** une fois l'application exposée sur Internet : il facilite la mise en place d'une politique de sécurité de contenu (CSP) stricte et réduit la surface d'attaque.

3. Modèle de données

Douze entités Doctrine. Le moteur MyISAM n'impose **aucune contrainte d'intégrité référentielle** au niveau base : les liens sont gérés applicativement (voir recommandations §7).

3.1 Comptes & personnes

Entité	Table	Champs principaux
User compte web	User	email , password (haché), roles , displayName , blocked (bool). Implémente <code>UserInterface</code> .
Porteur représentant	porteur	lastName , firstName , birthDate , entity , email , password (haché), roles , blocked . Lié à un <code>Site</code> ; rattaché à des Établissements (<code>porteur_perimetre</code>). Implémente <code>UserInterface</code> .
PersonAssignment	person_assignment	Rattachement polymorphe d'un compte (User ou Porteur) à un objet : target_type , target_id , role (sdd , site_referent , creator), owner_type , owner_id . Permet de désigner une personne quel que soit son type de compte.

3.2 Référentiel

Entité	Table	Champs principaux
Site	Site	name , city , department , sizeCategory , effectif (saisi, nullable), sdd (nullable), latitude / longitude (cache géocodage).
Perimetre Établissement	perimetre	label . Indépendant des sites.
Tract	tract	label , filename (fichier uploadé), updatedAt , status (a_tracter / en_attente / perime).
OperationType	operation_type	label .
Expertise	expertise	label (catalogue).
AccountExpertise	account_expertise	owner_type (user/porteur), owner_id , expertise , level (initié/avancé/expert).
SectionPermission	section_permission	section (clé unique), role_required — pilote le contrôle d'accès dynamique (voir §5.2).

3.3 Activité

Entité	Table	Champs principaux & relations
Campaign	Campaign	scheduledAt , createdAt , remarks , status (créée/prévue/réalisée/annulée), perimetreType (mono/multi/tout). Relations : Site , types d'opération (M:N), Établissements (campaign_perimetre), tracts (campaign_tract), participants (campaign_participant), créateur (voir §5.4).
HelpRequest	help_request	Demande d'entraide : campaign , porteur (destinataire), status (en attente/accepté/refusé), createdAt , respondedAt . Unicité applicative (campagne, représentant).

3.4 Nature des données (sensibilité)

- **Données personnelles** : nom, prénom, e-mail, **date de naissance**, entité de rattachement des représentants ; e-mails des comptes web. → Périmètre RGPD.
- **Données d'authentification** : mots de passe **hachés** (jamais en clair) — voir §5.1.
- **Données de géolocalisation** : latitude/longitude des sites (issues du géocodage de la ville).
- **Documents** : fichiers de tracts uploadés (PDF, images, documents).

4. Fonctionnalités principales

Module	Description
Tableau de bord	Indicateurs (campagnes, sites couverts, sites oubliés avec filtre jours + effectif minimum), liste filtrable des opérations.
Campagnes	Création/édition : site, date, statut, types d'opération, Établissements (mono/multi/tous), tracts multiples, remarques. Participants renseignables quand la campagne est « Réalisée ».
Demande d'entraide	Envoi d'une sollicitation aux représentants d'un site (enregistrée + e-mail pré-rempli reprenant un résumé de la campagne). Réponse accepté/refusé par le représentant.
Sites	Consultation publique (aux utilisateurs connectés) : effectifs, encadrement (SDD, référent), répartition par Établissement.
Carte des représentants	Carte Leaflet auto-hébergée, tuiles proxifiées par le NAS.
Mon espace	Profil, changement de mot de passe, expertises, demandes d'entraide reçues.
Administration	CRUD sites, représentants, Établissements, types d'opération, tracts, expertises ; gestion des comptes & droits ; configuration des accès par section.

5. Sécurité

5.1 Authentification

- **Deux types de comptes** implémentent `UserInterface` : `User` (comptes web) et `Porteur` (représentants). Un **chain provider** (`app_chain_provider`) tente successivement les deux fournisseurs sur le même formulaire e-mail/mot de passe.
- **Hachage des mots de passe** : `bcrypt` , **coût 13**, pour les deux entités. Aucun mot de passe stocké en clair.
- **Formulaire de connexion** : `form_login` (route `app_login`), redirection vers l'accueil après succès ; déconnexion via `app_logout` .
- **Blocage de compte** : un `UserChecker` refuse en pré-authentification tout compte `blocked = true` (message générique). Un abonné d'événement déconnecte également en temps réel une session déjà ouverte dont le compte vient d'être bloqué.
- **Jetons CSRF de connexion** : le gabarit inclut `csrf_token('authenticate')` —  voir §7.

5.2 Autorisation

- **Rôles** : `ROLE_USER` , `ROLE_ADMIN` , `ROLE_SUPER_ADMIN` . Hiérarchie : `ROLE_SUPER_ADMIN` → `ROLE_ADMIN` . Tout compte authentifié dispose au minimum de `ROLE_USER` (garanti par `getRoles()`).
- **Règles statiques** (pare-feu `main`) : `/login` public ; `/admin` et `/` exigent `ROLE_USER` .
- **Contrôle d'accès dynamique** : une entité `SectionPermission` associe chaque « section » de l'application à un rôle minimum, configurable par un administrateur. Un abonné d'événement (`AccessControlSubscriber`) applique ces règles à chaque requête ; l'interface masque les liens inaccessibles.
- **Super-administrateur protégé** : un compte `ROLE_SUPER_ADMIN` ne peut être ni bloqué ni rétrogradé depuis l'interface d'administration (protection vérifiée aussi côté serveur), même par un autre administrateur.

- **Cloisonnement métier** : seul le **créateur** d'une campagne peut en renseigner les participants (voir §5.4).

5.3 Protections applicatives

- **CSRF** : jetons sur les actions d'administration sensibles (suppressions, enregistrements de droits) via `csrf_token()` / validation serveur.
- **Sessions** : gestion de session Symfony native (cookie de session).
- **Injection SQL** : accès base via Doctrine (requêtes paramétrées) — pas de concaténation de requêtes.
- **XSS** : Twig échappe le HTML par défaut ; les rares insertions `|raw` concernent des données encodées en JSON côté serveur.
- **Surface réseau réduite** : aucun script tiers chargé côté navigateur (pas de CDN) ; ressources externes médiées par le serveur.
- **Uploads** : fichiers de tracts (types PDF/image/document, plafond de taille applicatif) stockés et servis via un contrôleur dédié.

5.4 Traçabilité du créateur de campagne

Comme **les représentants (comptes Porteur) peuvent créer des campagnes** et que le champ historique `createdBy` ne référence que les comptes web (User), le créateur réel est enregistré dans `person_assignment` (rôle `creator`), qui accepte les deux types de comptes. Ce rattachement est la source de vérité du **verrou de saisie des participants** : seul le créateur — User ou Porteur — est autorisé à les modifier.

6. Données fictives (avertissement)

Certaines données affichées sont générées, non réelles

Un service applicatif (SiteInfoService) produit de façon **déterministe** (à partir de l'identifiant du site) des valeurs de démonstration : **effectifs**, répartition par Établissement, encadrement (SDD, référent) et **DSCO** lorsqu'aucune donnée réelle n'a été saisie. Ces valeurs sont stables d'un affichage à l'autre mais **ne reflètent pas la réalité**. Elles seront progressivement remplacées par les données saisies (champ effectif du site, rattachements SDD/référent réels, etc.).

Pour une évaluation de sécurité, retenir que le code est aujourd'hui en **développement** (jeu de données de test possible) et que la **cible de production est un hébergement OVH mutualisé exposé sur Internet** : le durcissement Internet du §7 doit être en place avant la mise en ligne.

7. Points d'attention & recommandations (cible : OVH mutualisé, Internet)

Priorisation dans l'hypothèse d'une **exposition sur Internet**. L'accès n'étant pas encore tranché, la première recommandation est de **restreindre l'exposition au strict nécessaire**.

Sévérité	Constat	Recommandation
Critique	Comptes de démonstration affichés sur la page de connexion, mots de passe faibles (<code>admin1234</code> / <code>user1234</code>) — exposés à tout Internet.	Supprimer ces comptes et l'affichage des identifiants avant la mise en ligne ; imposer une politique de mots de passe robustes.
Critique	Interface d' administration exposée sur Internet (<code>/admin</code>) une fois hébergée sur OVH.	Restreindre l'accès à <code>/admin</code> par liste blanche d'IP via <code>.htaccess</code> si possible, et/ou authentification renforcée (MFA) ; journaliser les accès.
Élevée	Chiffrement du transport indispensable (identifiants et données personnelles). Configuration actuelle en <code>http://</code> .	Activer le certificat Let's Encrypt fourni par OVH, forcer la redirection HTTP→HTTPS et l'en-tête HSTS .
Élevée	Racine web : sur mutualisé, un mauvais paramétrage du dossier racine peut exposer le code source, <code>.env</code> (secrets), <code>config/</code> , <code>migrations/</code> .	Pointer le domaine sur <code>public/</code> uniquement ; vérifier qu'aucun fichier hors <code>public/</code> n'est accessible par URL.
Élevée	Absence de protection contre la force brute sur le formulaire de connexion.	Activer la limitation de tentatives de connexion de Symfony (<i>login throttling</i>) ; envisager un délai/captcha après échecs répétés.
Moyenne	Jeton CSRF de connexion : le gabarit émet le jeton, mais <code>enable_csrf</code> n'est pas explicitement activé sur <code>form_login</code> .	Activer <code>enable_csrf: true</code> sur <code>form_login</code> et vérifier la validation effective.
Moyenne	Absence d' en-têtes de sécurité HTTP (CSP, X-Frame-Options, X-Content-Type-Options, Referrer-Policy, HSTS).	Ajouter ces en-têtes via <code>.htaccess</code> (Apache) ou un abonné Symfony ; l'absence de scripts tiers permet une CSP stricte.
Moyenne	Cookies de session : attributs de sécurité à confirmer.	Une fois HTTPS actif : cookie de session en <code>Secure</code> + <code>HttpOnly</code> + <code>SameSite</code> .
Moyenne	Base de données managée OVH : identifiants et accès à repointer depuis la config de développement.	Utiliser des identifiants dédiés ; ne pas activer l'accès distant public à la base ; mettre à jour <code>DATABASE_URL</code> .
Moyenne	Moteur MyISAM sans clés étrangères : pas d'intégrité référentielle ni de transactions ACID.	Verrouiller les suppressions en cascade côté applicatif ; envisager InnoDB si l'intégrité devient critique.
Moyenne	Validation MIME réelle des tracts uploadés à confirmer (au-delà de l'extension).	Valider le type MIME côté serveur ; stocker hors racine web ou servir via contrôleur (déjà le cas) ; plafonner la taille.
Faible	Secrets (<code>APP_SECRET</code> , identifiants base) dans <code>.env</code> .	Protéger <code>.env</code> (hors racine web, non versionné) ; roter <code>APP_SECRET</code> en cas d'exposition.
Faible	Procédure de migration/déploiement sur mutualisé (pas de SSH garanti) et sauvegardes .	Définir la stratégie de migration (SSH selon l'offre, sinon SQL via phpMyAdmin) ; planifier des sauvegardes régulières de la base.
Faible	Données fictives affichées comme réelles (§6) ; colonne héritée <code>referencePorteur_id</code> inutilisée.	Baliser les valeurs estimées ; nettoyer la colonne morte après validation.

8. Synthèse

bcrypt / 13

HACHAGE MOTS DE PASSE

2 fournisseurs

CHAIN PROVIDER (USER + PORTEUR)

3 rôles

USER / ADMIN / SUPER_ADMIN

Dynamique

ACCÈS PAR SECTION CONFIGURABLE

Les fondations d'authentification et d'autorisation sont conformes aux pratiques Symfony (hachage fort, chain provider, contrôle d'accès à deux niveaux, blocage de compte, protection du super-admin). Le passage sur un **hébergement OVH mutualisé exposé sur Internet** déplace l'essentiel du risque vers l'**exposition** : avant toute mise en ligne, traiter en priorité la suppression des comptes de démonstration, la restriction de l'accès à `/admin`, l'activation de TLS/HSTS, la protection anti-force-brute et le bon paramétrage de la racine web (`public/`). Le durcissement système étant limité sur mutualisé, il repose sur `.htaccess` et la configuration applicative.

Document généré pour revue de sécurité — environnement DEV / pré-production. Contenu à valider avec l'équipe de développement.