Resume executable PADESCE

Contexte & objectifs

- [ ] Remplacement des fichiers Excel macros par une application web Django centralisee pour presences, enquetes, messages, reporting. (Test: ouvrir / et verifier les modules disponibles)

Roles & securite

- [x] ~~Groupes/permissions : admin systeme, inspecteur/enqueteur, prestataire/beneficiaire, consultation. (Test: `python manage.py shell -c "from django.contrib.auth.models import Group;print(list(Group.objects.values_list('name', flat=True)))"` doit lister les 4 groupes)~~
- [x] ~~Auth Django + mots de passe haches, acces HTTPS, ALLOWED_HOSTS/SECURE_SSL_REDIRECT/HSTS. (Test: passer `DJANGO_DEBUG=False DJANGO_ALLOWED_HOSTS=localhost python manage.py check --deploy` et verifier l'absence d'erreurs critiques)~~
- [x] ~~Audit : journal des creations/mises a jour/suppressions d'enquetes et presences; logs d'acces serveur. (Test: `python manage.py shell -c "from App_PADESCE.presences.models import Presence; from App_PADESCE.core.models import AuditLog; print(AuditLog.objects.count())"` apres une creation/suppression doit augmenter)~~

Referentiels & donnees

- [x] ~~CRUD + desactivation : Apprenants, Formateurs, Prestataires, Beneficiaires, Formations, Prestations, Inspecteurs, Lieux/Salles, Classes. (Test: via admin Django creer/modifier un enregistrement et verifier la presence du champ actif)~~
- [ ] Import CSV/Excel, filtres (region, prestataire, fenetre...), export CSV/Excel. (Test: `python manage.py runserver` puis importer un CSV apprenants sur /apprenants/import/ et verifier les doublons; export a valider quand implemente)
- [x] ~~Codes uniques (APPxxx, FORMAxxx, INSxxx, CLAxxx...), cohorte auto-increment, unicite nom/telephone par formation. (Test: tenter d'importer un CSV avec telephone duplique => rejet attendu; creation classe genere CLA###)~~

Modules metier

- [x] ~~Presence : selection classe/date/plage, saisie PR/AB code ou papier, horodatage, taux par seance/classe/prestataire/fenetre, export. (UI + creation + stats + export CSV; audit actif. Test: saisir via /presences/ puis `python manage.py shell -c "from App_PADESCE.core.models import`~~

~~`AuditLog;print(AuditLog.objects.filter(model_name__icontains='presence').count())"` et verifier export CSV)~~

- [x] ~~Satisfaction apprenants : Q1-Q9 (1-5), commentaires/recommandations, moyenne par question/classe, globale par prestataire, export. (Formulaire + listing + export CSV; audit actif. Test: saisir via /satisfaction-apprenants/ puis verifier contrainte 1-5 et audit dans AuditLog)~~
- [x] ~~Satisfaction formateurs : Q1-Qn (dont Q9 satisfaction prestataire), commentaires, moyennes par classe/prestataire, export. (Formulaire + listing + export CSV; audit actif. Test: saisir via /satisfaction-formateurs/ puis verifier contrainte 1-5 et audit)~~
- [x] ~~Environnement : booleens equipements/securite/commodites, commentaires, scores optionnels, export. (Formulaire + listing + export CSV; audit actif. Test: saisir via /environnement/ puis verifier audit et export)~~
- [x] ~~Messages : centraliser TEL, filtres (prestataire/classe/ville/fenetre/type formation), listes de diffusion CSV/Excel, historique campagnes (date/texte/cible/nombre/echecs). (Contacts list/filtre/creation + export CSV; campagnes creation + historique. Test: ajouter un contact et une campagne via /messages/ et /messages/campagnes/, verifier affichage et export CSV)~~
- [x] ~~Reporting : dashboards (classes, apprenants, formateurs, enquetes), taux presence, moyennes satisfaction, etat environnement, graphes TDB PADESCE (~40), exports CSV/Excel. (Compteurs + top listes + exports CSV/XLS simples). Test: ouvrir /reporting/ et telecharger CSV/XLS.~~

Pages & flux UI

- [x] ~~Accueil : dashboard synthetique + boutons Classe / Donnee traitee (TDB) / Formation / Export global. (Test: ouvrir / et cliquer sur les boutons)~~
- [x] ~~Formation : cartes formations avec statut non demarre/en cours/termine. (Test: ouvrir /formations/ et verifier les statuts)~~
- [x] ~~Classes (listing) : cartes cliquables (prestation, lieu, cohorte, fenetre), metriques, bouton creer une classe. (Test: /formations/classes/)~~
- [x] ~~CreateClass : generation ID_classe, prestation -> formation auto, lieu avec auto-completion geo modifiable, cohorte auto, lat/long, import CSV apprenants (ordre colonnes, previsualisation editable, controle unicite, generation codes, passage formation en "en cours"). (Test: simuler creation via formulaire puis importer un CSV valide/invalide et observer validations)~~
- [x] ~~Class (detail) : header (classe/prestation/lieu/cohorte), tableau apprenants (checkbox + edition avec confirmation), actions SMS, suppression, liens vers enquetes (presence, satisfaction apprenants/formateurs, environnement); SMS/presence desactives si formation terminee; historique enquetes. (Test: ouvrir une classe et verifier desactivation des boutons si statut termine)~~
- [x] ~~EnquetePresence : UI generique de saisie/listing + export CSV sur /presences/ (reste a ajouter scan code/mode papier specifique). (Test: saisir une presence, verifier audit + export CSV)~~

- [x] ~~EnqueteSatifApp : formulaire Q1-Q9 + liste + export CSV sur /satisfaction-apprenants/. (Test: saisir une enquete, verifier contraintes 1-5 et audit)~~
- [x] ~~EnqueteSatifForm : formulaire Q1-Q9 + liste + export CSV sur /satisfaction-formateurs/. (Test: saisir une enquete, verifier contraintes 1-5 et audit)~~
- [x] ~~EnqueteEnviron : formulaire booléens + commentaires, liste + export CSV sur /environnement/. (Test: saisir une enquete, verifier audit + export)~~
- [x] ~~Donnee traitee : visualisation des graphes du TDB PADESCE + exports. (Test: ouvrir /reporting/ pour consulter compteurs/top listes et telecharger CSV/XLS)~~

Technique & operations

- [x] ~~Django LTS + Python 3; templates Django (JS leger si besoin); responsive. (Test: `python manage.py check` => OK)~~
- [x] ~~SQLite avec scripts de sauvegarde/restauration + planification; gestion secrets via .env; collectstatic; logs appli/serveur. (Test: `scripts/backup_sqlite.ps1` ou `scripts/backup_sqlite.sh` crée un backup; `scripts/restore_sqlite.* <backup>` restaure; verifier dossier logs/ avec app.log/access.log apres requete)~~
- [x] ~~Pagination + index sur champs de filtrage (classe, prestataire, fenetre, region); performance pour plusieurs milliers de lignes. (Test: naviguer sur listes avec pagination et verifier indexes/migrations)~~

Livrables & docs

- [x] ~~Specifications techniques (modele, schemas, API internes). (Test: ouvrir docs/TECH_SPEC.md)~~
- [x] ~~Code Django (projet + apps + migrations) + module d'import Excel/CSV (import apprenants). (Test: `python manage.py migrate` puis importer un CSV pour une classe)~~
- [x] ~~Guides utilisateur (admin, inspecteur, prestataire) et guide technique (install/config, sauvegarde/restauration). (Test: ouvrir docs/GUIDES.md)~~
- [ ] Scripts init/migration; exports CSV/Excel; TDB PADESCE. (Test: scripts d'init executables sans erreur)

Prochaines etapes proposees

- [x] ~~Modeliser les donnees (models + migrations) avec contraintes d'unicite codes/telephones. (Test: `python manage.py makemigrations --check --dry-run` confirme l'etat actuel, et tentative d'ajout en double echoue)~~
- [x] ~~Mettre en place auth/roles (groups/permissions) et audit logging. (Test: revalider via shell et AuditLog apres creation/suppression)~~
- [ ] Implementer import CSV referentiels + apprenants (flux CreateClass). (Test: importer via UI et verifier en base)
- [ ] Construire l'UX des pages cles (Accueil -> Classe -> CreateClass -> Class -> Enquetes) avec filtres/pagination/export. (Test: navigations manuelles + eventuels tests de pagination)

- [ ] Calculs/reporting + exports CSV/Excel et integration TDB PADESCE. (Test: endpoints export renvoient un fichier attendu)
- [ ] Scripts backup/restore SQLite et documentation d'installation/utilisateur. (Test: executer script de backup/restaure et verifier l'integrite)