SYNTHÈSE COMPLÈTE DU PROJET

CyberRoles - ATLAS

Framework de Gestion des Compétences Cybersécurité

Document généré le 10/12/2025 à 10:55

TABLE DES MATIÈRES

1. Vue d'ensemble du projet
2. Architecture technique
3. Frameworks gérés
4. Modèles de données
5. Fonctionnalités principales
6. Technologies utilisées
7. Justification des choix techniques
8. Structure du projet
9. Système d'authentification et rôles
10. Déploiement
11. Conclusion

1. VUE D'ENSEMBLE DU PROJET

CyberRoles - ATLAS est une application web Django conçue pour supporter les organisations dans le mapping systématique de leurs exigences spécifiques—couvrant les connaissances, compétences techniques et compétences comportementales—associées à chaque poste au sein de la structure organisationnelle.

Ce processus de mapping facilite l'alignement avec les frameworks de référence établis pour la main-d'œuvre afin d'assurer à la fois l'adéquation des rôles et l'exécution efficace des tâches.

Objectifs principaux :

- Cartographier les compétences requises pour chaque poste
- Comparer et aligner les frameworks DCWF, NCWF et NICE Framework
- Faciliter l'identification des rôles de travail (work roles) appropriés
- Gérer les compétences KSAT (Knowledge, Skills, Abilities, Tasks)
- Fournir des outils d'exploration et de recherche avancés

2. ARCHITECTURE TECHNIQUE

Le projet suit une architecture **MVC (Model-View-Controller)** typique de Django, avec une séparation claire entre les modèles de données, les vues et les templates.

Structure principale :

- **Backend :** Django 5.1.6 - Framework web Python robuste et mature
- **Base de données :** SQLite3 pour le développement et la production
- **Frontend :** HTML5, CSS3, JavaScript avec Tailwind CSS pour le styling
- **Authentification :** Système personnalisé basé sur les matricules
- **API REST :** Endpoints JSON pour les données dynamiques

Organisation du code :

- **dcwf/** : Configuration principale Django (settings, urls, wsgi)
- **web_app/** : Application principale contenant :
    - models/ : Modèles de données (19 fichiers de modèles)
    - views.py, views_auth.py, views_saved_data.py : Logique métier
    - templates/ : Templates HTML organisés par fonctionnalité
    - static/ : Fichiers statiques (CSS, JS, images)
    - management/ : Commandes Django personnalisées
    - helpers/ : Fonctions utilitaires

3. FRAMEWORKS GÉRÉS

L'application gère plusieurs frameworks de compétences cybersécurité, permettant leur comparaison et leur alignement.

3.1 DCWF (DoD Cyberspace Workforce Framework)

Framework du Département de la Défense américain pour la main-d'œuvre cybersécurité. Le projet gère deux versions :

- **DCWF 2017 :** Version originale avec 79 work roles
- **DCWF 2025 :** Version mise à jour avec nouvelles catégories et rôles

3.2 NCWF (NICE Cybersecurity Workforce Framework)

Framework NICE (National Initiative for Cybersecurity Education) pour la main-d'œuvre cybersécurité civile. Le projet gère trois versions :

- **NCWF 2017 :** Version initiale du framework NICE
- **NCWF 2024 :** Version intermédiaire avec améliorations
- **NCWF 2025 :** Version la plus récente avec alignement DCWF 2025

3.3 NICE Framework

Framework NICE standard pour les compétences et rôles cybersécurité, intégré dans les modèles 2025.

3.4 Mapping entre frameworks

Le système maintient des relations entre les work roles des différents frameworks via les OPM IDs (Occupational Position Matrices), permettant :

- Correspondance automatique entre DCWF et NCWF
- Regroupement par OPM ID pour affichage cohérent
- Comparaison des KSATs entre frameworks
- Visualisation des différences et similitudes

4. MODÈLES DE DONNÉES

Le projet utilise 19 modèles Django pour représenter la structure complexe des frameworks.

4.1 Modèles principaux

- **DcwfWorkRole :** Rôles de travail DCWF 2017 avec relations OPM
- **Dcwf2025WorkRole :** Rôles de travail DCWF 2025
- **Ncwf2017WorkRole :** Rôles de travail NCWF 2017
- **Ncwf2024WorkRole :** Rôles de travail NCWF 2024
- **Ncwf2025WorkRole :** Rôles de travail NCWF 2025
- **NiceFrameworkWorkRole :** Rôles du framework NICE
- **AIWorkRole :** Rôles spécialisés en Intelligence Artificielle

4.2 Modèles KSAT

Les KSATs (Knowledge, Skills, Abilities, Tasks) sont les compétences associées aux work roles :

- **DcwfKsat :** KSATs DCWF avec catégories (knowledge, skill, ability, task)
- **Dcwf2025Ksat :** KSATs DCWF 2025
- **Ncwf2017Ksat :** KSATs NCWF 2017
- **Ncwf2024Tks :** TKS (Tasks, Knowledge, Skills) NCWF 2024
- **Ncwf2025Ksat :** KSATs NCWF 2025

4.3 Modèles de relations

Les relations Many-to-Many sont gérées via des modèles intermédiaires :

- **DcwfWorkRoleKsatRelation :** Relation entre work roles DCWF et KSATs
- **Dcwf2025WorkRoleKsatRelation :** Relations pour DCWF 2025
- **Ncwf2025WorkRoleKsatRelation :** Relations pour NCWF 2025
- **Opm :** Occupational Position Matrix - Point de correspondance entre frameworks

4.4 Modèles utilisateur

- **User :** Modèle utilisateur personnalisé avec matricule et rôle
- **UserSavedData :** Données sauvegardées par utilisateur (sélections KSAT, etc.)

5. FONCTIONNALITÉS PRINCIPALES

5.1 Exploration des Work Roles

- **Page d'accueil (t1.html) :** Vue d'ensemble de tous les work roles organisés par catégories
- **Modèles 2025 :** Interface dédiée aux frameworks 2025 avec séparation DCWF/NCWF
- **Détails work role :** Affichage détaillé d'un work role avec ses KSATs
- **DCWF Finder :** Outil interactif avec questionnaire guidé, catalogue complet et recherche rapide

5.2 Comparaison de frameworks

- **Sélection multi-frameworks :** Choix de work roles depuis DCWF 2017, DCWF 2025, NCWF 2017, NCWF 2024, NCWF 2025
- **Comparaison KSAT :** Affichage côte à côte des KSATs avec regroupement par OPM ID
- **Visualisation par catégories :** Organisation par Knowledge, Skills, Abilities, Tasks
- **Détails de compétences :** Pages dédiées pour NF-COM-002 (Sécurité IA) et NF-COM-007 (Cyber Resiliency)

5.3 Gestion des compétences (Étape 2)

- **Saisie de poste :** Interface pour saisir un numéro de poste
- **Cadres de niveaux :** Affichage des cadres de niveaux de maîtrise
- **Domaines de compétence :** Exploration des domaines de compétence NICE Framework (NF-COM)
- **Sauvegarde :** Possibilité de sauvegarder les sélections pour reprise ultérieure
- **Étape2Plus :** Explorateur interactif des domaines de compétence

5.4 Outils d'analyse

- **Summary Chart :** Visualisation graphique des compétences
- **Summary MIL :** Analyse des compétences militaires
- **Project Recap :** Récapitulatif complet du projet
- **Step0 Baseline :** Outil de baseline pour l'analyse initiale

5.5 API et données

- **API JSON :** Endpoints pour récupérer les données KSAT, work roles, etc.
- **Données externes :** Intégration de données SFIA, compétences militaires, postes IDF
- **Export :** Téléchargement de fichiers .md et .mm (FreeMind)

6. TECHNOLOGIES UTILISÉES

6.1 Backend

- **Django 5.1.6 :** Framework web Python - Choix justifié par sa maturité, sa sécurité et sa communauté
- **Python 3.10+ :** Langage de programmation principal
- **SQLite3 :** Base de données relationnelle - Simple, portable, suffisante pour ce projet

6.2 Frontend

- **HTML5 :** Structure sémantique moderne
- **CSS3 + Tailwind CSS :** Framework CSS utilitaire pour un développement rapide et cohérent
- **JavaScript (Vanilla) :** Pas de framework JS lourd - Performance et simplicité
- **Google Fonts (Inter) :** Typographie moderne et lisible

6.3 Outils et bibliothèques

- **Django Admin :** Interface d'administration intégrée
- **Django Management Commands :** Commandes personnalisées pour la gestion
- **JSON :** Format de données pour les APIs et exports
- **Excel/CSV :** Import/export de données depuis fichiers Excel

7. JUSTIFICATION DES CHOIX TECHNIQUES

7.1 Django vs autres frameworks

Pourquoi Django ?

- **ORM puissant :** Facilite la gestion des relations complexes entre frameworks
- **Admin intégré :** Interface d'administration sans développement supplémentaire
- **Sécurité :** Protection CSRF, XSS, SQL injection intégrées
- **Écosystème :** Nombreuses bibliothèques et extensions disponibles
- **Documentation :** Excellente documentation et communauté active
- **Maturité :** Framework stable et éprouvé en production

7.2 SQLite vs PostgreSQL/MySQL

Pourquoi SQLite ?

- **Simplicité :** Pas de serveur de base de données à configurer
- **Portabilité :** Fichier unique, facile à déplacer et sauvegarder
- **Suffisance :** Volume de données gérable (work roles, KSATs)
- **Performance :** Excellente pour les lectures, suffisante pour ce projet
- **Déploiement :** Simplifie le déploiement sur PythonAnywhere

7.3 Tailwind CSS vs Bootstrap/CSS custom

Pourquoi Tailwind CSS ?

- **Développement rapide :** Classes utilitaires pour styling immédiat
- **Cohérence :** Design system intégré pour une interface uniforme
- **Personnalisation :** Facile à personnaliser via configuration
- **Performance :** Purge CSS pour réduire la taille finale
- **Modernité :** Approche moderne du styling CSS

7.4 Architecture modulaire

Le projet utilise une architecture modulaire avec séparation des responsabilités :

- **Modèles séparés :** Un fichier par modèle pour maintenabilité
- **Vues spécialisées :** views.py, views_auth.py, views_saved_data.py
- **Templates organisés :** Par fonctionnalité (main/, ksat/, work_role/, etc.)
- **Helpers :** Fonctions utilitaires réutilisables
- **Management commands :** Scripts CLI pour administration

8. STRUCTURE DU PROJET

8.1 Organisation des répertoires

- **dcwf/** : Configuration Django (settings.py, urls.py, wsgi.py, asgi.py)
- **web_app/** : Application principale
    - models/ : 19 modèles de données
    - templates/ : Templates HTML organisés par fonctionnalité
    - static/ : Fichiers statiques (CSS, JS, images, outils)
    - management/ : Commandes Django personnalisées
    - helpers/ : Fonctions utilitaires
    - templatetags/ : Tags de template personnalisés
    - migrations/ : Migrations de base de données
- **static/** : Fichiers statiques racine
- **staticfiles/** : Fichiers statiques collectés pour production
- **KSAT 2025 FINAL/** : Données JSON des frameworks 2025
- **Step0/** : Outil Baseline
- **FORSAPINNOEL/** : Données SFIA
- **SUMAARYMIL/** : Données compétences militaires

8.2 Fichiers de configuration

- **manage.py :** Point d'entrée Django
- **requirements.txt :** Dépendances Python (Django ~= 5.1.6)
- **db.sqlite3 :** Base de données SQLite
- **pythonanywhere_wsgi.py :** Configuration WSGI pour PythonAnywhere
- **deploy_*.py :** Scripts de déploiement

9. SYSTÈME D'AUTHENTIFICATION ET RÔLES

9.1 Authentification par matricule

Le système utilise une authentification personnalisée basée sur les **matricules** (7 chiffres) plutôt que les emails traditionnels, adapté au contexte organisationnel.

9.2 Système de rôles

Trois niveaux de rôles sont définis avec détection automatique :

- **Super User :** Accès complet, accès direct aux étapes 2 et 3, interface spéciale (bordure violette)
- **Bêta testeur :** Accès étendu, fonctionnalités avancées, interface spéciale (bordure orange)
- **User Normal :** Accès standard, progression normale, interface standard (bordure bleue)

9.3 Gestion des données utilisateur

Le modèle **UserSavedData** permet de sauvegarder :

- Sélections KSAT (ksat_selection_*)
- Données générales utilisateur
- Sélections de work roles
- Configurations personnalisées

9.4 Commandes de gestion

Commande Django personnalisée **manage_user_roles** pour :

- Lister tous les rôles
- Ajouter/Supprimer des Super Users
- Ajouter/Supprimer des Bêta testeurs
- Mettre à jour tous les utilisateurs

10. DÉPLOIEMENT

10.1 Plateforme : PythonAnywhere

Le projet est déployé sur **PythonAnywhere**, une plateforme d'hébergement Python qui simplifie le déploiement Django.

10.2 Configuration de déploiement

- **URL de production :** willaz.pythonanywhere.com
- **STATIC_ROOT :** Configuration spécifique pour PythonAnywhere
- **ALLOWED_HOSTS :** Configuration des hôtes autorisés
- **WSGI :** Configuration WSGI pour servir l'application
- **Static files :** Collecte et configuration des fichiers statiques

10.3 Scripts de déploiement

- **deploy_willaz.py :** Script de déploiement spécifique
- **deploy_pythonanywhere.py :** Script générique PythonAnywhere
- **deploy_simple.py :** Script de déploiement simplifié
- **deploy_static.py :** Script pour fichiers statiques uniquement

10.4 Documentation de déploiement

Plusieurs fichiers de documentation guident le déploiement :

- **DEPLOYMENT_GUIDE.md :** Guide général de déploiement
- **PYTHONANYWHERE_DEPLOYMENT.md :** Guide spécifique PythonAnywhere
- **PYTHONANYWHERE_DEPLOYMENT_COMPLETE.md :** Guide complet
- **CONFIGURATION_WILLAZ.md :** Configuration spécifique utilisateur
- **ETAPE2PLUS_PYTHONANYWHERE_DEPLOYMENT.md :** Déploiement outil Étape2Plus

11. CONCLUSION

Le projet **CyberRoles - ATLAS** représente une solution complète et sophistiquée pour la gestion et l'alignement des frameworks de compétences cybersécurité.

Points forts du projet :

- **Architecture solide :** Django offre une base robuste et maintenable
- **Gestion multi-frameworks :** Support de DCWF, NCWF et NICE Framework avec versions multiples
- **Interface utilisateur moderne :** Tailwind CSS pour une expérience utilisateur agréable
- **Fonctionnalités complètes :** Exploration, comparaison, analyse et sauvegarde
- **Extensibilité :** Architecture modulaire facilitant l'ajout de nouvelles fonctionnalités
- **Documentation :** Documentation complète pour déploiement et utilisation

Utilisations potentielles :

- Cartographie des compétences organisationnelles
- Alignement avec les frameworks de référence
- Planification de carrière et développement professionnel
- Recrutement et évaluation des compétences
- Formation et certification

Ce projet démontre une compréhension approfondie des frameworks de compétences cybersécurité et offre une plateforme puissante pour leur gestion et leur alignement.