GUIDE DE BONNE CONDUITE POUR IMPLEMENTATION I18N RAPIDE (FR -> EN)

1. MODELE POUR STOCKAGE DES TRADUCTIONS

- Créer un modèle TextTranslation

source_lang, target_lang, source_text, translated_text

- Ajouter unique_together.

2. SERVICE text_translate.py

- Fonction translate_text avec :

* Normalisation langue (fr, en)

* Cache Django (avec hashing md5 du texte)

* Requête vers TextTranslation si existe

* Retour direct si trouvé

* Pas d'appel Google live (ou fallback minimal)

- Important : Hash dans les clés cache pour éviter erreurs memcached.

3. SERIALIZER MIXIN (I18nTranslateMixin)

- Appliquer translation sur i18n_fields et i18n_nested

- Ne pas traduire les endpoints dashboard

- Utiliser translate_text

4. PRE-TRADUCTION (accélération)

- Commande management: pretranslate_catalog

- Traduit en avance les catégories, marques, couleurs, produits, blogs

- Stocke dans TextTranslation

5. VIEWS OPTIMISÉES

- LatestProductsView et CategoryProductList

* Utiliser cache niveau view

* Construire category, slug, state avec translate_text

* Eviter traductions live répétées

6. FRONTEND

- React :

* Appeler API avec ?lang=en

* Ne jamais traduire côté front les données DB

* Utiliser seulement i18next pour UI statique (boutons, labels, placeholders)

7. COMBINAISON OPTIMALE

* Backend :

- TextTranslation

- translate_text avec cache + hash

- pretranslate_catalog

- I18nTranslateMixin

* Frontend :

- i18next (UI statique)

- API lang parameter (données dynamiques)

* Résultat : affichage instantané en EN sans délai

8. BONNES PRATIQUES

- Toujours stocker FR comme langue source
- Ne jamais traduire slug dans base, mais générer version traduite si désiré
- Éviter googletrans live en production
- Activer Redis/Memcached pour cache robuste

FIN.