Application Symfony sécurisée, testée, dockerisée, et déployée (Apache)

I. Objectif pédagogique

Développer une **application Symfony professionnelle** avec les exigences suivantes :

- Authentification robuste (email + Google)
- Sécurité renforcée (regex, CSRF, honeypot)
- Infrastructure Docker avec **Apache**, MySQL et Mailhog
- Tests automatisés (unitaires, fonctionnels, intégration)
- Pipeline CI/CD avec **gestion du cache Symfony**
- Service d'envoi de mails
- Environnements séparés (`dev`, `test`, `prod`)
- Déploiement en ligne
- Test de performance

II. Fonctionnalités à développer

Pages :

- **Accueil** (publique)
- **Inscription**
- **Connexion (email + Google)**
- **Profil** (réservé aux utilisateurs connectés)

III. Authentification & sécurité

- Formulaire d'inscription avec email + mot de passe
- Connexion avec **Google OAuth2** (`knpu/oauth2-client-bundle`)
- Validation du mot de passe via **regex forte** :
- 12 caractères minimum

ELAN
14 rue du Rhône
67100 STRASBOURG
TEL : 03.88.30.78.30
www.elan-formation.fr

- 1 majuscule, 1 minuscule, 1 chiffre, 1 caractère spécial
- Protection **CSRF** sur tous les formulaires
- Protection anti-bot par **champ honeypot** via `FormEventSubscriber`
- Hash de mot de passe avec `bcrypt` ou `argon2id`

IV. Envoi de mails (MailerService)

Créer un `MailerService` Symfony :

- Envoi d'un email de bienvenue HTML avec Twig
- Envoi d'un email de notification à l'admin
- En local, utiliser Mailhog pour intercepter les mails

V. Docker (Apache)

Créer un environnement Docker composé de :

- **PHP 8 + Apache** (`php:8.x-apache`)
- **MySQL** ou **MariaDB**
- **Mailhog** pour tester les mails
- Configuration Apache (`vhost.conf`) avec `mod_rewrite`

VI. Environnements

L'application doit gérer proprement :

- `.env.local` pour le développement
- `.env.test` pour les tests automatisés
- `.env.prod` pour la production

VII. Tests automatisés

- **Tests unitaires** : ex. vérification de la validité d'un mot de passe
- **Tests fonctionnels** : accès route `/profile` avec/without login
- **Tests d'intégration** : scénario d'inscription et connexion complètes

ELAN
14 rue du Rhône
67100 STRASBOURG
TEL : 03.88.30.78.30
www.elan-formation.fr

VIII. CI/CD avec gestion du cache Symfony

Mettre en place un pipeline CI/CD (GitHub Actions ou GitLab CI) qui :

- Installe les dépendances (Composer, npm/yarn si nécessaire)
- Exécute les linters (`lint:yaml`, `lint:twig`, `lint:container`)
- Lance les tests automatisés
- **Vide puis génère le cache Symfony (`cache:clear + cache:warmup`)**
- (Bonus) Utilise `actions/cache` pour accélérer les jobs
- Déploie automatiquement si les tests passent

IX. Déploiement

L'application doit être accessible publiquement en ligne :

Plateformes recommandées :

- **Railway** (Docker supporté, base de données incluse, gratuit)
- **Render**
- **VPS (Oracle Cloud, gratuit à vie)**

Conditions :

- Déploiement automatisé (ou semi-auto) via SSH ou CLI
- Environnement `prod` activé
- (Bonus) Certificat HTTPS activé via **Traefik** ou **Certbot**

X. Test de performance

- Utiliser un outil de benchmark :
  - `ab` (ApacheBench), `siege`, `wrk`, ou PageSpeed Insights
- Effectuer un test de charge sur `/`
- Analyser les résultats :
- Temps moyen, requêtes/seconde, taille de réponse
- Proposer 1 à 2 pistes d'optimisation :
- GZIP, cache HTTP, assets optimisés, désactiver debug, etc.
- (Bonus) Ajouter Lighthouse ou PageSpeed dans la CI

ELAN
14 rue du Rhône
67100 STRASBOURG
TEL : 03.88.30.78.30
www.elan-formation.fr

XI. Livrables attendus

- Code source sur GitHub
- README clair et structuré :
  - Installation locale (Docker)
  - Configuration `.env`
  - Exécution des tests
  - Pipeline CI/CD
  - Sécurité (regex, CSRF, honeypot)
  - Déploiement
  - Résultats du test de performance
- Lien vers le site en ligne (en `prod`)

ELAN
14 rue du Rhône
67100 STRASBOURG
TEL : 03.88.30.78.30
www.elan-formation.fr