Test Technique - Développeur Backend Symfony

Durée maximale : **48 heures** (temps effectif à noter dans le README)

Objectif : Évaluer votre capacité à concevoir une **mini-API** robuste et propre. Le périmètre cible couvre l'authentification JWT, la gestion minimale des comptes, l'upload des fichiers, et un flux de réservation simplifié avec calcul de disponibilités par durée de prestation.

1) Enoncé fonctionnel

Construire une API REST qui permette :

1. **Comptes & Authentification**
   - Création de compte client et salon (owner).
   - Connexion par email + mot de passe via **JWT** (access token + refresh token).
   - Vérification d'email (lien de validation) et **réinitialisation de mot de passe**.
   - **RBAC** minimal : ROLE_CLIENT, ROLE_STYLIST, ROLE_OWNER, ROLE_ADMIN.
2. **Salons / Prestations / Coiffeurs**
   - CRUD minimal des salons (owner only), lecture publique (list + détail).
   - **Prestations** : nom, description, **durée (min)**, **prix**.
   - Association des **coiffeurs (stylists)** à un salon + **compétences** (quelles prestations ils réalisent) + **langues** parlées.
3. **Disponibilités & Réservations**
   - Déclaration d'horaires hebdo du salon et exceptions (fermetures / congés) *simple*.
   - Création d'un **rendez-vous** par un client sur un **créneau disponible**, en fonction de la durée de la prestation et des créneaux du coiffeur sélectionné.
   - Annulation par client ou owner (règles simples : annulation > 2h avant).
   - **Liste d'attente** minimale : si créneau complet, inscription sur une waitlist ; lorsqu'un créneau se libère, **notification simulée** (log + email console).
4. **Recherche & Filtres**
   - Endpoint public pour **rechercher des salons** avec filtres : ville, nom, **prix (asc/desc)**, **genre** (H/F), **disponibilités** sur un créneau (date+heure+durée), **langue**.
5. **Fichiers / Portfolio**
   - **Upload d'images** (photo « après » prestation) par un coiffeur vers son **portfolio**.
   - Stockage local + URL publique ; **validation MIME/poids** ; génération d'un **thumbnail** (optionnel bonus).
6. **Avis (reviews)**
   - Un client peut laisser une note (1–5) + texte après un RDV.
   - **Modération par owner** : statut PENDING|APPROVED|REJECTED.
   - La note globale d'un salon = moyenne des notes approuvées de ses coiffeurs.

Important : Le but n'est pas de couvrir toutes les fonctionnalités, mais de démontrer que vous pouvez **modéliser** correctement, **coder proprement**, sécuriser, et livrer une API **documentée et testée** qui reflète l'ADN d'un projet.

2) Contraintes techniques (si vous pouvez proposer mieux, n'hésitez pas)

- **Symfony 6.4+ ou 7.x**, PHP **8.2+**.
- ORM : **Doctrine** (PostgreSQL ou MySQL).
- Auth : **lexik/jwt-authentication-bundle** (access) + refresh tokens (ex. gesdinet/jwt-refresh-token-bundle ou implémentation maison documentée).
- Upload : **VichUploaderBundle** ou implémentation Symfony HttpFoundation ; stockage **local** (adapter **Flysystem** en bonus).
- E-mails : **Symfony Mailer** (transport smtp ou filesystem pour tests). Templates basiques.
- **Validation** (Symfony Validator), **Serializer**, **Rate Limiter**.
- **Migrations** (Doctrine Migrations) + **Fixtures** (Faker) pour un jeu de données minimal.
- **Tests** : PHPUnit + tests d'API (WebTestCase) ; bonus : coverage CI.
- **Documentation** : **OpenAPI** via **NelmioApiDocBundle** ou **API Platform** (autorisé, mais justifiez). Fournir un **Postman/Insomnia collection**.
- **Qualité** : PHP-CS-Fixer, PHPStan (niveau >= 6) **ou** Psalm, conventions de code.
- **Sécurité** : politiques d'accès (Voter/Security), contrôle d'ownership, validation d'input, **CORS**.
- **Docker** (bonus apprécié) : docker-compose (app + db + mailcatcher).

3) Modèle de données (minimum)

Représentation indicative (vous pouvez ajuster) :

- **User**(id, email*, password_hash, roles[ ], first_name, last_name, phone?, email_verified_at?)
- **Salon**(id, owner:User, name, slug, address, city, lat?, lng?, open_hours_json)
- **Stylist**(id, salon, user, languages:string, skills:Service)
- **Service**(id, salon, name, description, duration_minutes, price_cents)
- **AvailabilityException**(id, salon|stylist, date, closed:boolean, reason?)
- **Booking**(id, salon, stylist, client:User, service, start_at, end_at, status: PENDING|CONFIRMED|CANCELLED)
- **WaitlistEntry**(id, salon, service, desired_start_range_start, desired_start_range_end, client)
- **Media**(id, stylist, path, original_name, mime, size_bytes, created_at)
- **Review**(id, booking, stylist, client, rating:int, comment, status: PENDING|APPROVED|REJECTED)

Disponibilités : calculez les slots disponibles en croisant : horaires salon (open_hours_json type {mon:["09:00-12:00","14:00-18:00"], …}), exceptions, RDV existants, **durée de la prestation**.

4) Endpoints requis (exemples de contrat)

Auth & Comptes

- POST /auth/register — crée un compte (client par défaut). Body: email, password, first_name, last_name.
- POST /auth/login — renvoie tokens JWT (access + refresh).
- POST /auth/refresh — refresh token -> nouvel access token.
- POST /auth/forgot-password — envoie lien de réinit.
- POST /auth/reset-password — applique la réinit.
- POST /auth/verify-email — consomme un token de vérif (lien email).

Salons & Prestations

- GET /salons — liste + **filtres** : city, q, lang, gender, priceSort (asc|desc), availableAt=2025-09-20T15:00, duration=45.
- GET /salons/{id} — détail (incl. moyenne des notes, coordonnées, liens maps).
- POST /salons — création (owner). **RBAC**.
- PATCH /salons/{id} — update (owner/admin).
- POST /salons/{id}/services — CRUD basique services (owner/admin).
- POST /salons/{id}/stylists — associer stylist + langues + skills.

Disponibilités & Réservations

- GET /salons/{id}/availability — calcule les **créneaux disponibles** pour serviceId, stylistId?, date.
- POST /bookings — crée un RDV (client) sur un slot libre.
- DELETE /bookings/{id} — annule (client si owner==client et > 2h ; owner/admin sinon).
- POST /waitlist — s'inscrire à la liste d'attente (fenêtre préférée). Notifier **simulée**.

Portfolio & Upload

- POST /stylists/{id}/media — **upload image** (multipart/form-data). Validation (max 5 Mo, MIME image/*). Retourne URL publique.
- GET /stylists/{id}/media — lister.
- DELETE /media/{id} — supprimer (owner/stylist owner).

Avis

- POST /bookings/{id}/reviews — créer un avis (1–5 + texte) si booking passé et client == owner booking.
- PATCH /reviews/{id} — **modération** APPROVE/REJECT (owner/admin).

Pagination standard (ex. page, limit) + DTOs stables. **OpenAPI** à jour.

5) Règles métier minimales

- Un **stylist** ne peut être réservé que s'il **sait faire** la prestation demandée.
- Un **slot** est disponible si : ouvertures salon OK, pas d'exception « fermé », pas de booking chevauchant la **fenêtre [start, start+duration]** du service.
- Annulation < 2h avant : **refusée** (422) pour client ; owner/admin peuvent forcer.
- La **langue** filtrable est l'une des langues déclarées du stylist.
- La **note du salon** = moyenne des reviews APPROVED de ses stylists.

6) Sécurité & conformité (RGPD minimal)

- **Hashage** des mots de passe (bcrypt/argon2id via password_hasher).
- **JWT** avec TTL courts (ex. access 15 min, refresh 7 j) ; blacklisting refresh (ou rotation) documenté.
- **Rate limiting** sur /auth/* et /upload.
- Validation stricte (constraints) + erreurs **RFC 7807** (application/problem+json).
- **CORS** configuré ; logs d'audit basiques (création/suppression booking, modération avis).
- **Politique d'accès** (Voters) : un owner ne modifie que son salon ; un client que ses bookings.

7) Livrables

1. **Repo** (GitHub/GitLab) public ou privé (accès à contact@seeds.cm et seedinnov@gmail.com) avec :
    - Code Symfony, **.env.example**, **migrations**, **fixtures**.
    - **README** :
        - Contexte & choix techniques
        - Setup (Docker & hors Docker)
        - Variables env (JWT_PASSPHRASE, DB_*, MAILER_DSN, CORS, UPLOAD_DIR)
        - Commandes (migrations, fixtures, tests)
        - Durée réelle (heures) + priorisations + limites + TODO
    - **OpenAPI.yaml/json** (Nelmio) + **collection Postman/Insomnia**.
    - **Tests** (unitaires/fonctionnels, happy + error paths).
    - (Bonus) **CI** GitHub Actions : lint + phpstan + tests.
    -

2. **Démonstration** : courte note ou script curl/HTTPie montrant un parcours complet :

- Register → Verify → Login → Refresh
- Owner crée salon + services + stylists
- Client liste salons filtrés, calcule disponibilités, réserve, annule
- Upload photo portfolio
- Laisse un avis ; owner modère ; note globale remonte

8) Barème d'évaluation (100 pts)

- **Qualité de code & archi** (25 pts)
  - DDD léger, séparation contrôleurs/handlers/services/repos, DTO, mapping clair
  - Nommage, lisibilité, commentaires pertinents, erreurs normalisées
- **Sécurité & Auth** (15 pts)
  - JWT solide, policies d'accès, rate limit, validation
- **Modèle & règles métier** (15 pts)
  - Disponibilités correctes, compétences/ langues, annulation
- **Upload & médias** (10 pts)
  - Validation, URLs publiques, suppression sécurisée
- **Recherche & perfs** (10 pts)
  - Filtres, pagination, tri ; requêtes optimisées (indexes, SELECT parcimonieux)
- **Tests** (10 pts)
  - Couverture endpoints clés, scénarios edge, données fixtures
- **Docs & DX** (10 pts)
  - OpenAPI, README exemplaire, collection Postman
- **Bonus** (5 pts)
  - Docker, CI, thumbnails, Flysystem S3-ready, webhooks simulés

9) Jeux de données & scénarios à couvrir

- **Fixtures** :
  - 2 owners (chacun 1 salon à Paris/Lyon), 3 stylists/salon, 6 services (durées variées : 20, 30, 45, 60, 90 min).
  - Langues : fr, en, es ; compétences variées.
  - 10 clients ; quelques bookings passés & futurs (chevauchements à tester).
  - 5 reviews APPROVED, 2 PENDING.
- **Scénarios test** :
  1. Filtre par **langue** + **créneau** (ex. Jeudi 15:00–17:00, 45 min) → salons/stylists compatibles.
  2. Réservation sur un slot libre → succès ; réservation sur slot chevauchant → 409.
  3. Annulation < 2h → 422 (client) ; owner → 204.
  4. Upload image non-image → 415 ; > 5 Mo → 413.
  5. Review sans booking passé → 403.

10) Guidelines d'implémentation

- **Contrats d'API stables** : versionnez sous /api/v1/*.
- **Erreurs** : utilisez ProblemDetails (RFC 7807) cohérentes.
- **Services** : AvailabilityService, BookingService, WaitlistNotifier (Mailer + Logger), MediaStorage.
- **Requêtes** : évitez le N+1 (joins, fetch=EXTRA_LAZY le cas échéant, indexes sur booking(start_at, stylist_id) et service(salon_id, duration_minutes)).
- **Sécurité** : Voters par ressource (SalonVoter, BookingVoter, MediaVoter).
- **Config** : .env propres, secrets gérés via bin/console secrets:set (si possible).

11) Submission

- Dépôt Git hébergé + accès.
- README avec : temps passé, arbitrages, axes d'amélioration si + de temps, points de vigilance pour la **scalabilité** (sharding salons, queues pour notifications/sync agenda, etc.).
- Facultatif : petite vidéo (≤ 5 min) montrant les endpoints clés via Postman.

Bonne chance!

Le but est de **montrer votre niveau réel** en production-like (sécurité, clarté, pragmatisme).