ÉNONCÉ D'EXAMEN : DÉVELOPPEMENT D'UN SITE WEB DYNAMIQUE

Contexte de l'Examen

Type : Projet pratique
Modalités : Travail individuel sur machine

Objectif Général

Développer un **site web dynamique complet** avec frontend interactif, backend fonctionnel et base de données, mettant en œuvre les concepts fondamentaux du développement web moderne.

Scénario Professionnel

Vous êtes engagé(e) comme développeur(euse) full-stack junior chez **"Digital Solutions"**. L'entreprise a besoin d'une **plateforme de gestion de projets collaboratifs** pour ses équipes internes.
Votre mission est de développer un **MVP (Minimum Viable Product)** avec les fonctionnalités essentielles.

Spécifications Techniques

1. EXIGENCES TECHNIQUES OBLIGATOIRES

text
FRONTEND :
✓ Framework JavaScript (React, Vue ou Angular)
✓ Responsive design (mobile-first)
✓ Routing client-side
✓ Gestion d'état centralisée
✓ Formulaires avec validation
✓ Appels API asynchrones

BACKEND :
✓ API RESTful (Node/Express, Django ou Spring)
✓ Base de données (MySQL, PostgreSQL ou MongoDB)
✓ Authentification sécurisée
✓ Architecture MVC
✓ Validation des données

BASE DE DONNÉES :
✓ Minimum 3 entités relationnelles
✓ Relations 1-N et N-N
✓ Contraintes d'intégrité
✓ Données de test

2. MODÈLE DE DONNÉES

sql
-- Exemple de structure minimale
UTILISATEURS (id, email, nom, rôle, date_inscription)

PROJETS (id, titre, description, créateur_id, date_création)
TÂCHES (id, projet_id, titre, description, assigné_à, statut, échéance)

FONCTIONNALITÉS REQUISES

NIVEAU 1 : FONCTIONNALITÉS DE BASE (60%)

A. Authentification & Profil

- **Inscription** : Formulaire avec validation
- **Connexion/Déconnexion** : Sessions sécurisées
- **Profil utilisateur** : Visualisation/édition
- **Rôles** : Administrateur / Membre

B. Gestion des Projets (CRUD)

- **Créer** un nouveau projet
- **Lister** tous les projets (avec pagination)
- **Voir** détails d'un projet
- **Modifier** un projet (créateur seulement)
- **Supprimer** un projet

C. Gestion des Tâches

- **Ajouter** tâches à un projet
- **Assigner** tâches aux membres
- **Changer** statut (À faire, En cours, Terminé)
- **Filtrer** tâches par statut/assignation

NIVEAU 2 : FONCTIONNALITÉS AVANCÉES (30%)

D. Interface Utilisateur Dynamique

- **Recherche** instantanée dans les projets
- **Tri** des projets/tâches (date, statut)
- **Notifications visuelles** (toasts/messages)
- **Drag & drop** pour changer statut des tâches
- **Graphiques** (stats simples)

E. Collaboration Temps Réel

- **Commentaires** sur les tâches
- **Notifications** en temps réel (WebSockets)
- **Historique** des modifications

NIVEAU 3 : BONUS & EXCELLENCE (10%)

F. Fonctionnalités Bonus

- **Upload de fichiers** pour les tâches
- **Export PDF** du projet
- **Calendrier** des échéances
- **API externe** intégration (ex: météo, maps)
- **Tests unitaires** (frontend/backend)

CONTRAINTES DE SÉCURITÉ

Obligatoires :

- **Hash** des mots de passe (bcrypt/scrypt)
- **Validation/sanitisation** des entrées
- **JWT tokens** avec expiration
- **Middleware d'authentification**
- **Protection CORS** configurée

- **Variables d'environnement** pour les secrets

Recommandées :

- Rate limiting
- Helmet.js (headers sécurité)
- Protection CSRF/XSS

DESIGN & UX

Responsive Design :

```
css
/* Points de rupture minimum */
- Mobile : < 768px
- Tablette : 768px - 1024px
- Desktop : > 1024px
```

Accessibilité :

- ARIA labels
- Contraste des couleurs
- Navigation au clavier
- Alt text pour les images

STRUCTURE TECHNIQUE ATTENDUE

Organisation du Projet :

```
text
projet-examen/
├── client/                 # Application frontend
│   ├── public/
│   ├── src/
│   │   ├── components/     # Composants réutilisables
│   │   ├── pages/          # Pages de l'application
│   │   ├── services/       # Appels API
│   │   ├── store/          # Gestion d'état
│   │   └── styles/         # Fichiers CSS
│   └── package.json
│
├── server/                 # API backend
│   ├── controllers/
│   ├── models/
│   ├── routes/
│   ├── middleware/
│   └── config/
│
├── database/               # Scripts SQL/NoSQL
│   ├── schema.sql
│   ├── seed.sql
│   └── migrations/
│
├── docs/                   # Documentation
│   ├── API.md
│   └── README.md
│
└── docker-compose.yml      # Facultatif
```

CRITÈRES D'ÉVALUATION

1. FONCTIONNALITÉ (40%)

- ✓ Toutes les fonctionnalités de base implémentées

- ✓ Code fonctionnel sans erreurs bloquantes
- ✓ Interactions utilisateur fluides

2. CODE & ARCHITECTURE (30%)

- ✓ Code propre et bien structuré
- ✓ Respect des design patterns
- ✓ Séparation des responsabilités
- ✓ Gestion des erreurs

3. SÉCURITÉ & PERFORMANCE (20%)

- ✓ Bonnes pratiques de sécurité
- ✓ Optimisation des requêtes
- ✓ Gestion du cache
- ✓ Temps de réponse

4. UX & DESIGN (10%)

- ✓ Interface intuitive
- ✓ Responsive design
- ✓ Accessibilité
- ✓ Cohérence visuelle

LIVRABLES ATTENDUS

À la fin de l'examen, fournir :

1. **Code source complet** sur GitHub/GitLab
2. **Base de données** avec données de test
3. **Documentation** (README.md) contenant :
    - Instructions d'installation
    - Variables d'environnement
    - Endpoints API
    - Utilisateurs de test
4. **Présentation** (5 minutes) du projet
5. **Démo fonctionnelle** de l'application

CONSEILS PRATIQUES

1. **Commencez simple** : MVP d'abord, extensions ensuite
2. **Testez régulièrement** : Vérifiez chaque fonctionnalité
3. **Versionnez** : Commits Git fréquents et clairs
4. **Documentez au fur et à mesure** : Commentaires dans le code
5. **Priorisez** : Fonctionnel > Beau > Parfait

QUESTIONS TYPES DE L'EXAMEN

Questions théoriques possibles :

1. Expliquez le cycle de vie d'une requête HTTP dans votre application
2. Comment gérez-vous les états globaux vs locaux ?
3. Quelle stratégie de cache avez-vous implémentée ?
4. Décrivez votre approche de sécurité
5. Comment optimisez-vous les performances ?

🆘 RESSOURCES AUTORISÉES

- ✅ Documentation officielle
- ✅ Stack Overflow (lecture seulement)
- ✅ Anciens projets personnels
- ✅ Snippets de code personnels
- ❌ Communication avec d'autres candidats
- ❌ Code généré par IA non supervisée
- ❌ Partage de code en temps réel

✅ CHECKLIST DE VALIDATION

markdown
[ ] Backend fonctionnel avec au moins 5 endpoints
[ ] Base de données connectée avec 3+ tables/collections
[ ] Authentification JWT opérationnelle
[ ] Frontend avec 3+ pages/routes
[ ] CRUD complet sur au moins une entité
[ ] Interface responsive (mobile + desktop)
[ ] Formulaire avec validation
[ ] Gestion des erreurs utilisateur
[ ] Documentation complète
[ ] Code déployé/runnable localement

SUPPORT TECHNIQUE

En cas de problème technique pendant :

1. Documentez l'erreur (screenshots)
2. Essayez une solution alternative
3. Contactez le surveillant si blocage > 15min

BON TRAVAIL !
Votre capacité à créer une application complète, sécurisée et maintenable sera évaluée.
La qualité prime sur la quantité – une application simple mais bien faite vaut mieux qu'une application complexe mais buggée.