Cahier des Charges - Galerie d'Art OTC

1. PRÉSENTATION DU PROJET

1.1 Contexte

Développement d'une plateforme web de galerie d'art permettant la présentation et la gestion d'œuvres artistiques en ligne.

1.2 Objectif

Créer une galerie d'art virtuelle moderne avec système d'authentification, gestion d'œuvres et interface utilisateur responsive.

---

2. PÉRIMÈTRE FONCTIONNEL

2.1 Fonctionnalités Backend (Node.js/Express)

Authentification & Autorisation

- **Inscription utilisateur** avec validation des données
- **Connexion sécurisée** avec JWT tokens
- **Gestion de session** utilisateur
- **Middleware d'authentification** pour routes protégées
- **Système de rôles** (utilisateur/admin)

Gestion des Œuvres

- **CRUD complet** des œuvres d'art
  - Création avec upload d'images
  - Lecture/Affichage des œuvres
  - Modification des informations
  - Suppression d'œuvres
- **Upload d'images** via Multer
- **Stockage** dans répertoire `uploads/`
- **Métadonnées** : titre, artiste, année, description, catégorie

Gestion des Catégories

- **Création de catégories** artistiques
- **Classification** des œuvres par catégorie
- **Filtrage** des œuvres par catégorie

Base de Données

- **MongoDB** avec Mongoose ODM

- **Modèles** :
  - User (utilisateurs)
  - Artwork (œuvres d'art)
  - Category (catégories)

2.2 Fonctionnalités Frontend

Pages Publiques

- **Page d'accueil** avec galerie d'œuvres
- **Page de détail** d'une œuvre
- **Galerie filtrée** par catégorie
- **Navigation responsive**

Pages Authentifiées

- **Tableau de bord** utilisateur
- **Gestion de profil**
- **Ajout d'œuvres** (formulaire)
- **Mes œuvres** (liste personnelle)

Interface Utilisateur

- **Design responsive** (mobile, tablet, desktop)
- **Navigation intuitive**
- **Cartes d'œuvres** avec images
- **Formulaires de saisie** validés
- **Messages de feedback** utilisateur

---

3. ARCHITECTURE TECHNIQUE

3.1 Stack Technologique

Backend

- **Runtime** : Node.js
- **Framework** : Express.js
- **Base de données** : MongoDB
- **ODM** : Mongoose
- **Authentification** : JWT (jsonwebtoken)
- **Validation** : express-validator
- **Upload de fichiers** : Multer
- **Sécurité** : bcrypt, helmet, cors
- **Variables d'environnement** : dotenv

Frontend

- **HTML5** sémantique

- **CSS3** avec design responsive
- **JavaScript** vanilla
- **Fetch API** pour requêtes HTTP
- **LocalStorage** pour tokens JWT

3.2 Structure des Données

Modèle User

```
{
  username: String (unique, required),
  email: String (unique, required, validated),
  password: String (hashed, required),
  role: String (enum: ['user', 'admin']),
  createdAt: Date
}
```

Modèle Artwork

```
{
  title: String (required),
  artist: String (required),
  year: Number,
  description: String,
  category: ObjectId (ref: Category),
  imageUrl: String,
  owner: ObjectId (ref: User),
  createdAt: Date
}
```

Modèle Category

```
{
  name: String (unique, required),
  description: String,
  createdAt: Date
}
```

3.3 API Endpoints

Authentification

- `POST /api/auth/register` - Inscription
- `POST /api/auth/login` - Connexion

- `GET /api/auth/me` - Profil utilisateur (protégé)

Œuvres

- `GET /api/artworks` - Liste des œuvres
- `GET /api/artworks/:id` - Détail d'une œuvre
- `POST /api/artworks` - Créer œuvre (protégé)
- `PUT /api/artworks/:id` - Modifier œuvre (protégé)
- `DELETE /api/artworks/:id` - Supprimer œuvre (protégé)
- `GET /api/artworks/category/:categoryId` - Œuvres par catégorie

Catégories

- `GET /api/categories` - Liste des catégories
- `POST /api/categories` - Créer catégorie (admin)

---

4. CONTRAINTES TECHNIQUES

4.1 Sécurité

- **Hachage des mots de passe** avec bcrypt (10 rounds)
- **Tokens JWT** avec expiration (24h)
- **Validation** des entrées utilisateur
- **Protection CORS** configurée
- **Helmet.js** pour headers HTTP sécurisés
- **Sanitisation** des données

4.2 Performance

- **Images optimisées** pour le web
- **Pagination** des résultats (si nécessaire)
- **Indexation** MongoDB sur champs recherchés
- **Compression** des réponses HTTP

4.3 Compatibilité

- **Navigateurs modernes** (Chrome, Firefox, Safari, Edge)
- **Responsive design** : mobile-first
- **Support** des formats d'images : JPG, PNG, WebP

---

5. CONFIGURATION ET DÉPLOIEMENT

5.1 Variables d'Environnement

```
PORT=5000
MONGODB_URI=mongodb://localhost:27017/otc-gallery
```

```
JWT_SECRET=your-secret-key
NODE_ENV=development
```

5.2 Installation

```
# Installation dépendances backend
npm install

# Installation dépendances frontend
cd public && npm install

# Démarrage serveur développement
npm run dev
```

5.3 Structure des Répertoires

```
OTC/
├── models/           # Modèles Mongoose
├── routes/           # Routes Express
├── middleware/       # Middlewares (auth, validation)
├── controllers/      # Logique métier
├── config/           # Configuration (DB, etc.)
├── uploads/          # Images uploadées
├── public/           # Frontend statique
│   ├── css/
│   ├── js/
│   └── pages/
└── server.js         # Point d'entrée
```

---

6. LIVRABLES

6.1 Code Source

- ☑ Backend Express.js complet
- ☑ Frontend HTML/CSS/JS
- ☑ Modèles de données
- ☑ API REST documentée

6.2 Documentation

- ☑ README.md
- ☑ Documentation API
- ☑ Guide d'installation
- ⏳ Cahier des charges (ce document)

6.3 Tests

- ⏳ Tests unitaires backend
- ⏳ Tests d'intégration API
- ⏳ Tests frontend

---

7. ÉVOLUTIONS FUTURES

7.1 Fonctionnalités Avancées

- **Système de favoris** pour utilisateurs
- **Commentaires** sur œuvres
- **Notation** des œuvres
- **Partage** sur réseaux sociaux
- **Recherche avancée** (texte, filtres multiples)
- **Galeries personnelles** utilisateur

7.2 Améliorations Techniques

- **Cache Redis** pour performances
- **CDN** pour images
- **Tests automatisés** complets
- **CI/CD pipeline**
- **Monitoring** et logs
- **Backup automatique** base de données

7.3 Interface

- **Mode sombre**
- **Internationalisation** (i18n)
- **Animations** et transitions
- **Accessibilité** (WCAG 2.1)
- **PWA** (Progressive Web App)

---

8. PLANNING PRÉVISIONNEL

Phase 1 - Backend Core ☑ (Complété)

- Setup projet et configuration
- Modèles de données
- Authentification JWT
- CRUD Artworks

Phase 2 - Frontend Base ☑ (Complété)

- Pages principales
- Formulaires
- Intégration API

- Design responsive

Phase 3 - Améliorations (En cours)

- Tests
- Documentation complète
- Optimisations performances
- Sécurité renforcée

Phase 4 - Déploiement (À venir)

- Configuration production
- Déploiement serveur
- Monitoring
- Maintenance

---

9. BUDGET ET RESSOURCES

9.1 Ressources Humaines

- **1 Développeur Full-Stack** (vous)
- **Durée estimée** : 3-4 semaines

9.2 Ressources Techniques

- **Hébergement** : VPS ou PaaS (Heroku, Vercel)
- **Base de données** : MongoDB Atlas (gratuit/payant)
- **Stockage images** : Serveur local ou S3
- **Domaine** : À définir

---

10. CRITÈRES DE VALIDATION

10.1 Fonctionnels

- ☑ Inscription/connexion fonctionnelle
- ☑ Upload et affichage d'œuvres
- ☑ CRUD complet artworks
- ☑ Filtrage par catégories
- ☑ Interface responsive

10.2 Techniques

- ☑ API REST fonctionnelle
- ☑ Authentification sécurisée
- ☑ Validation des données
- ⏳ Tests couvrant >80% du code
- ⏳ Performance <2s chargement

10.3 Qualité

- ☑ Code propre et commenté
- ☑ Structure modulaire
- ⏳ Documentation complète
- ⏳ Pas d'erreurs console
- ⏳ Pas de vulnérabilités critiques

Document rédigé le : 17 décembre 2025
Version : 1.0
Auteur : Équipe Ervoul
Statut : En développement actif