IndieHub

Plateforme de Lancement de Jeux Indépendants

Réalisé par :

Zeini Cheikh Sidi Ely	Matricule : **23025**
Roughaya Bebane	Matricule : **22078**

Filière : Développement de Systèmes d'Information

Niveau : Licence (3ème année)

Année Universitaire : **2025 – 2026**

“Le numérique est le moteur de l’innovation moderne.
Ce travail est dédié à tous ceux qui croient en la puissance
de la technologie pour changer le monde.”

Résumé

Le présent rapport décrit la conception et le développement d'**IndieHub**, une plateforme web et desktop dédiée à la centralisation et à la promotion des jeux indépendants. La plateforme est ouverte à **tous les développeurs indépendants** dans le monde, avec un soutien particulier aux développeurs arabophones et aux contenus en langue arabe.

IndieHub offre un écosystème complet ouvert à **tous les développeurs indépendants**, avec un support natif des contenus arabes et de la directionnalité RTL/LTR. Les administrateurs valident chaque soumission selon des critères culturels, éthiques et de qualité stricts. Tout contenu à caractère sexuel est explicitement exclu de la plateforme. Les utilisateurs peuvent découvrir, télécharger et évaluer les jeux approuvés.

Sur le plan technique, le projet adopte une architecture découplée client-serveur reposant sur **Django REST Framework** pour le backend, **React 19 avec TypeScript** pour le frontend, et **Electron** pour la version desktop multiplateforme. La conteneurisation via Docker garantit la reproductibilité de l'environnement de déploiement.

Mots-clés : Jeux indépendants, plateforme web, Django, React, TypeScript, Electron, API REST, bilinguisme, validation de contenu.

Abstract

This report describes the design and development of **IndieHub**, a web and desktop platform dedicated to centralizing and promoting independent games. The platform is open to **all independent developers** worldwide, with dedicated support for Arabic-speaking developers and Arabic-language content.

IndieHub provides a complete ecosystem open to **all independent developers worldwide**, with dedicated support for Arabic-language content and RTL/LTR directionality. The platform enforces a strict content policy that explicitly excludes sexual or ethically inappropriate content. Administrators validate every submission before it becomes publicly visible.

Technically, the project adopts a decoupled client-server architecture based on **Django REST Framework** for the backend, **React 19 with TypeScript** for the frontend, and **Electron** for the cross-platform desktop version. Docker containerization ensures deployment environment reproducibility.

Keywords : Independent games, web platform, Django, React, TypeScript, Electron, REST API, bilingualism, content validation.

Remerciements

Nous tenons à exprimer notre profonde gratitude à toutes les personnes qui ont contribué, de près ou de loin, à l'aboutissement de ce Projet Python.

Nos remerciements s'adressent en premier lieu à la direction de l'**Institut Supérieur du Numérique (SupNum)** pour la formation de qualité dispensée tout au long de notre cursus, ainsi qu'à l'ensemble du corps enseignant pour leur encadrement et leur disponibilité.

Nous remercions particulièrement notre encadreur pour ses précieux conseils, son suivi rigoureux et sa bienveillance tout au long de ce projet.

Enfin, nous exprimons notre reconnaissance à nos familles et amis pour leur soutien indéfectible et leurs encouragements constants qui ont été une source de motivation inestimable.

Contents

List of Tables

Chapter 1

Introduction Générale

1.1 Contexte et Motivation

L'industrie du jeu vidéo connaît une croissance exponentielle à l'échelle mondiale. Cependant, les développeurs indépendants — qu'ils soient issus de régions arabophones, africaines, ou de n'importe quelle autre partie du monde — rencontrent d'importantes difficultés pour se faire connaître et distribuer leurs créations. Les plateformes existantes, telles que Steam ou Itch.io, ne prennent pas toujours en compte les spécificités culturelles, linguistiques et éthiques de ces communautés.

Face à ce constat, le projet **IndieHub** est né de la volonté de créer un espace ouvert à **tous les développeurs indépendants**, avec un soutien particulier aux créateurs arabophones et aux contenus en langue arabe. La plateforme se distingue également par une **politique de contenu stricte** : tout contenu à caractère sexuel ou contraire aux valeurs culturelles et éthiques des communautés servies est formellement exclu. Cette ligne directrice est au coeur du système de validation mis en place.

1.2 Problématique

La problématique centrale de ce projet peut se formuler ainsi :

Problématique

Comment concevoir et réaliser une plateforme de distribution de jeux indépendants qui soit à la fois culturellement adaptée aux communautés arabophones, techniquement performante, et accessible aussi bien sur le web que sur desktop ?

1.3 Objectifs du Projet

Les objectifs principaux d'IndieHub sont les suivants :

1. Fournir aux développeurs indépendants un canal de distribution fiable et accessible.
2. Mettre en place un système de validation garantissant la conformité culturelle, éthique et religieuse des contenus.
3. Offrir une expérience utilisateur moderne et bilingue (Anglais/Arabe, LTR/RTL).
4. Assurer la disponibilité de la plateforme sur le web et en application desktop multiplateforme.

5. Implémenter des outils analytiques permettant de suivre la popularité des jeux.

1.4 Structure du Rapport

Ce rapport est organisé comme suit. Le Chapitre 2 présente l'analyse des besoins et la spécification des exigences. Le Chapitre 3 décrit l'architecture et la conception du système. Le Chapitre 4 détaille les choix technologiques et l'implémentation. Le Chapitre 5 présente la stratégie de tests. Enfin, une conclusion générale résume les apports du projet et envisage les perspectives futures.

Chapter 2

Analyse des Besoins

2.1 Identification des Acteurs

Le système IndieHub est destiné à trois catégories d'acteurs aux rôles bien distincts :

2.1.1 L'Utilisateur (Gamer)

L'utilisateur final est un joueur souhaitant découvrir, télécharger et évaluer des jeux indépendants. Il dispose d'une bibliothèque personnelle et peut interagir avec la communauté via le système de notation.

2.1.2 Le Développeur (Developer)

Le développeur est un créateur de jeux indépendants. Il peut soumettre ses productions, suivre leur statut de validation depuis un tableau de bord dédié et gérer ses contenus.

2.1.3 L'Administrateur (Admin)

L'administrateur est le garant de la qualité et de la conformité des contenus. Il dispose des droits d'approbation ou de rejet des soumissions, ainsi que de la gestion des utilisateurs.

2.2 Besoins Fonctionnels

Le tableau 2.1 synthétise les besoins fonctionnels identifiés lors de la phase d'analyse.

2.3 Besoins Non Fonctionnels

Au-delà des fonctionnalités, plusieurs contraintes non fonctionnelles ont guidé la conception :

- **Performance :** Les temps de réponse de l'API doivent être inférieurs à 500 ms pour les requêtes courantes.
- **Sécurité :** Toutes les routes sensibles sont protégées par authentification par token (Token Auth). Les fichiers de jeux sont accessibles via des flux sécurisés.
- **Maintenabilité :** L'architecture modulaire garantit une séparation claire des responsabilités, facilitant les évolutions futures.

Table 2.1: Besoins fonctionnels du système IndieHub

ID	Acteur	Description
BF01	Développeur	Soumettre un jeu avec fichiers, descriptions et captures d'écran
BF02	Développeur	Suivre le statut de validation de ses soumissions
BF03	Administrateur	Approuver ou rejeter les soumissions : vérification qualité, conformité culturelle/éthique et exclusion des contenus inappropriés
BF04	Utilisateur	Rechercher et filtrer les jeux par catégorie, titre, ou popularité
BF05	Utilisateur	Télécharger les jeux approuvés de manière sécurisée
BF06	Utilisateur	Gérer une bibliothèque personnelle de jeux
BF07	Utilisateur	Rédiger des critiques et attribuer des notes aux jeux
BF08	Tous	S'inscrire, se connecter et gérer son profil
BF09	Administrateur	Consulter les statistiques de téléchargement
BF10	Système	Proposer une interface bilingue Anglais/Arabe

- **Portabilité :** La conteneurisation Docker assure le déploiement sur n'importe quelle infrastructure.
- **Accessibilité :** Le support RTL/LTR garantit une expérience optimale pour les utilisateurs arabophones.
- **Scalabilité :** La base de données peut migrer de SQLite (développement) vers PostgreSQL (production) sans modification du code applicatif.

2.4 Politique de Contenu

La politique de contenu est un pilier fondamental d'IndieHub. Elle repose sur des règles claires et non négociables appliquées lors de chaque validation :

Règles de Contenu – IndieHub

- **Ouverture universelle :** IndieHub accueille les soumissions de **tous les développeurs indépendants**, quelle que soit leur origine géographique ou culturelle.
- **Support arabe :** Un soutien particulier est apporté aux développeurs arabophones et aux jeux en langue arabe, avec une interface entièrement adaptée (RTL).
- **Contenu familial obligatoire :** Tout contenu à caractère sexuel ou explicite est **strictement interdit** et systématiquement rejeté.
- **Conformité culturelle et éthique :** Les contenus doivent respecter les valeurs culturelles et religieuses des communautés desservies par la plateforme.

2.5 Cas d'Utilisation

Les cas d'utilisation principaux sont illustrés à travers les scénarios suivants.

2.5.1 Soumission d'un Jeu (Développeur)

1. Le développeur s'authentifie sur la plateforme.
2. Il accède à son tableau de bord et initie une nouvelle soumission.
3. Il renseigne les métadonnées du jeu (titre, description, catégories) et téléverse le fichier exécutable.
4. Le système enregistre la soumission avec le statut `pending`.
5. Un administrateur est notifié pour évaluation.

2.5.2 Validation d'un Jeu (Administrateur)

1. L'administrateur consulte la liste des jeux en attente.
2. Il examine les métadonnées, les captures d'écran et le contenu du jeu.
3. Il vérifie la conformité selon la politique de contenu : absence de contenu sexuel, respect des valeurs culturelles et éthiques.
4. Il approuve (`approved`) ou rejette (`rejected`) la soumission selon ces critères.
5. Le jeu approuvé devient visible dans le catalogue public.

Chapter 3

Conception du Système

3.1 Architecture Générale

IndieHub adopte une architecture **client-serveur découplée**, conforme aux principes REST. Cette approche garantit une séparation nette entre la logique métier (backend) et la présentation (frontend), facilitant la maintenance et l'évolutivité du système.

Principes Architecturaux

- Découplage total entre frontend et backend via une API REST.
- Authentification stateless par token.
- Application Electron comme wrapper de l'application web pour le desktop.
- Conteneurisation complète via Docker Compose.

3.1.1 Vue d'Ensemble

Le système est composé de trois couches principales :

1. **Couche Présentation (Client) :** Application React (TypeScript) accessible via navigateur ou encapsulée dans Electron pour le desktop.
2. **Couche Logique Métier (Serveur) :** API REST Django exposant des endpoints pour chaque ressource du domaine.
3. **Couche Persistance :** Base de données relationnelle (SQLite en développement, PostgreSQL en production).

3.2 Modèle de Données

Le modèle de données d'IndieHub s'articule autour des entités principales suivantes :

3.3 Architecture API REST

L'API est organisée en modules thématiques, chacun correspondant à un domaine métier distinct. La base URL est `http://{host}:8000/api/`.

3.3.1 Module Utilisateurs et Authentification

3.3.2 Module Jeux

Table 3.1: Entités principales du modèle de données

Entité	Description
User	Représente tout utilisateur du système avec son rôle (user, developer, admin)
Game	Entité centrale contenant les métadonnées du jeu, son fichier, son statut et ses relations
Category	Catégorie de classification des jeux (RPG, Action, Puzzle, etc.)
Download	Enregistrement de chaque acte de téléchargement pour les statistiques
LibraryEntry	Association entre un utilisateur et les jeux de sa bibliothèque personnelle
Review	Critique et note attribuée par un utilisateur à un jeu
Screenshot	Captures d'écran associées à un jeu

Table 3.2: Endpoints – Module Utilisateurs

Méthode	Endpoint	Description
POST	`/users/register/`	Création d'un nouveau compte
POST	`/users/login/`	Authentification et obtention du token
GET	`/users/users/`	Liste des utilisateurs (Admin uniquement)

Table 3.3: Endpoints – Module Jeux

Méthode	Endpoint	Description
GET	`/games/games-list/`	Catalogue public des jeux approuvés (avec filtres)
POST	`/games/games/`	Soumission d'un nouveau jeu (Développeur)
PATCH	`/games/games/{id}/`	Mise à jour du statut (Admin)
POST	`/games/reviews/`	Soumission d'une critique

3.3.3 Module Téléchargements et Analytique

Table 3.4: Endpoints – Module Téléchargements

Méthode	Endpoint	Description
GET	`/downloads/popular-games/`	Jeux tendances par nombre de téléchargements
GET	`/downloads/games/{id}/download/`	Téléchargement sécurisé du fichier
GET	`/downloads/games/{id}/stats/`	Statistiques d'utilisation

3.4 Architecture Frontend

Le frontend est structuré selon les meilleures pratiques React :

- **components**/ : Composants réutilisables (cartes de jeux, barres de navigation, formulaires).
- **pages**/ : Vues complètes de l'application (accueil, catalogue, tableau de bord, administration).
- **services**/ : Couche d'abstraction pour les appels API (Axios).
- **contexts**/ : Gestion d'état global via React Context API (authentification, langue).

3.5 Application Desktop

L'application desktop est réalisée via **Electron**, qui encapsule l'application web React dans une fenêtre native. Cette approche présente l'avantage de réutiliser intégralement le code frontend sans duplication, tout en offrant aux utilisateurs une expérience desktop native.

Chapter 4

Implémentation

4.1 Environnement Technologique

Le choix des technologies a été guidé par trois critères : la maturité de l'écosystème, la productivité de développement et la pertinence par rapport aux besoins du projet.

Table 4.1: Stack technologique d'IndieHub

Catégorie	Technologie	Justification
Backend	Django 4.x	Framework Python mature, ORM puissant, admin intégré
	DRF	Sérialisation, validation, permissions et vues génériques REST
	SQLite / PostgreSQL	Flexibilité dev/prod via l'abstraction ORM
Frontend	React 19	Composants réactifs, écosystème riche, performances élevées
	TypeScript	Typage statique réduisant les erreurs à la compilation
	Vite	Build ultra-rapide, HMR (Hot Module Replacement)
	TailwindCSS	Design utilitaire cohérent, support RTL natif
	Axios	Client HTTP promesse avec intercepteurs
Desktop	Electron	Wrapper Chromium multiplateforme (Win/-Mac/Linux)
DevOps	Docker / Compose	Conteneurisation, reproductibilité, déploiement simplifié

4.2 Structure du Projet

Le projet est organisé en trois répertoires principaux correspondant aux trois composantes du système.

4.2.1 Backend Django

Le backend est structuré en applications Django modulaires :

- **games**/ : Modèles, sérialiseurs, vues et permissions pour la gestion des jeux.
- **users**/ : Authentification par token, gestion des rôles et profils utilisateurs.

- **downloads/** : Suivi des téléchargements et calcul des métriques de popularité.
- **library/** : Gestion de la bibliothèque personnelle de chaque utilisateur.
- **api/** : Configuration globale de l'API (URLs, permissions par défaut, CORS).

4.2.2 Gestion de l'Authentification

L'authentification repose sur le **Token Authentication** de Django REST Framework. Chaque requête authentifiée doit inclure le header :

```
Authorization: Token <votre_token>
```

Listing 4.1: Exemple de requête authentifiée

4.2.3 Système de Validation des Jeux

Le cycle de vie d'un jeu dans IndieHub suit le workflow suivant :

$$\texttt{pending} \xrightarrow{\text{Admin : approuve}} \texttt{approved} \xrightarrow{\text{Visible publiquement}}$$

$$\texttt{pending} \xrightarrow{\text{Admin : rejette}} \texttt{rejected}$$

4.3 Support Bilingue

Le support Anglais/Arabe est implémenté à deux niveaux :

1. **Frontend :** Un `LanguageContext` React gère l'état de la langue courante. Les composants s'abonnent à ce contexte et adaptent dynamiquement la direction (RTL pour l'Arabe, LTR pour l'Anglais) et les libellés.
2. **Données :** Les métadonnées des jeux peuvent être saisies dans les deux langues, permettant un affichage localisé.

4.4 Déploiement Docker

Le déploiement complet de la stack est réalisé via Docker Compose :

```
# Construire et lancer tous les services
docker compose up --build

# Lancer en arri re-plan
docker compose up -d
```

Listing 4.2: Démarrage de la stack complète

Les services exposés sont :

- Frontend : `http://localhost:80`
- Backend API : `http://localhost:8000`

Chapter 5

Tests et Validation

5.1 Stratégie de Tests

La stratégie de tests adoptée dans IndieHub couvre deux niveaux complémentaires : les tests unitaires et d'intégration du backend, et les tests de composants du frontend.

5.2 Tests Backend (Django)

Les tests backend sont écrits avec `APITestCase` de Django REST Framework. Ils couvrent les modèles, les sérialiseurs, les permissions et les flux API complets.

Table 5.1: Tests unitaires Backend

Test	Description
`test_create_game_as_developer`	Vérifie qu'un développeur authentifié peut soumettre un jeu
`test_public_game_list`	S'assure que seuls les jeux approuvés sont visibles publiquement
`test_registration`	Vérifie la logique de création de compte utilisateur

```
cd backend
python manage.py test
```

Listing 5.1: Exécution des tests backend

5.3 Tests Frontend (React)

Les tests frontend utilisent **Vitest** et **React Testing Library**, offrant une approche centrée sur le comportement des composants.

Table 5.2: Tests Frontend React

Fichier de Test	Couverture
`Login.test.tsx`	Validation des inputs du formulaire et appel API à la soumission
`Games.test.tsx`	Rendu des cartes de jeux et fonctionnalité de recherche
`AuthContext.test.tsx`	Persistance de l'état d'authentification entre les composants

```
cd frontend
npm test
```

Listing 5.2: Exécution des tests frontend

Conclusion Générale et Perspectives

Bilan du Projet

Le projet IndieHub constitue une réponse concrète et techniquement aboutie au besoin d'une plateforme de distribution de jeux indépendants adaptée aux spécificités culturelles et linguistiques des communautés arabophones et africaines.

Au terme de ce Projet Python, nous avons réalisé :

- Une architecture client-serveur moderne et découplée, tirant parti des meilleures pratiques du développement web actuel.
- Un système de validation de contenu robuste garantissant la conformité culturelle et éthique des jeux publiés.
- Une interface bilingue Anglais/Arabe avec support RTL/LTR natif.
- Une application desktop multiplateforme via Electron.
- Un pipeline de déploiement conteneurisé avec Docker Compose.
- Une suite de tests couvrant les fonctionnalités critiques du système.

Ce projet nous a permis d'approfondir nos compétences dans des domaines variés : développement backend avec Django, développement frontend avec React et TypeScript, architecture logicielle, et pratiques DevOps.

Perspectives

Plusieurs pistes d'amélioration et d'extension sont envisageables pour les versions futures d'IndieHub :

1. **Monétisation :** Intégration d'un système de paiement pour les jeux premium ou les donations aux développeurs.
2. **IA et Recommandations :** Système de recommandation personnalisé basé sur l'historique de l'utilisateur.
3. **Extension mobile :** Développement d'une application mobile native (React Native) pour iOS et Android.
4. **Internationalisation étendue :** Support de langues additionnelles (Français, Wolof, Amazigh).
5. **CDN et performances :** Migration vers un CDN pour la distribution des fichiers de jeux à l'échelle mondiale.

Références

1. Django Software Foundation. *Django Documentation*. https://docs.djangoproject.com/
2. Tom Christie. *Django REST Framework Documentation*. https://www.django-rest-framework.org/
3. Meta Open Source. *React Documentation*. https://react.dev/
4. Microsoft. *TypeScript Documentation*. https://www.typescriptlang.org/docs/
5. Electron Authors. *Electron Documentation*. https://www.electronjs.org/docs/
6. Docker Inc. *Docker Documentation*. https://docs.docker.com/
7. Evan You. *Vite Documentation*. https://vitejs.dev/
8. Adam Wathan et al. *Tailwind CSS Documentation*. https://tailwindcss.com/docs