Six Degrees of Kevin Bacon

Jeu web inspiré du concept des **six degrés de séparation** (variante "Six Degrees of Kevin Bacon"). Le but : relier deux acteurs entre eux en passant par des films qu'ils ont tournés en commun, en un minimum d'étapes.

- **Dépôt** : https://git.hugo-pierret.be/Hugyouu/six-degrees
- **App** : https://six-degrees.hugo-pierret.be

Fonctionnalités

- **Défi quotidien** — une paire d'acteurs par jour, commune à tous les joueurs
- **Mode infini** — parties en solo avec des paires aléatoires
- **Multijoueur temps réel** — lobbys via WebSocket (SignalR)
- **Calendrier** — historique des défis quotidiens passés
- **Visualisation graphe** — représentation interactive acteur → film → acteur (D3.js)

Stack technique

Couche	Technologie
Backend	ASP.NET Core 8.0 (C#)
Base de données	PostgreSQL + Entity Framework Core 8.0
Temps réel	SignalR
Frontend	React 18 + Vite
Styles	Tailwind CSS + Material Tailwind
Graphe	react-force-graph-2d (D3.js)
API externe	TMDB
Déploiement	Docker + Docker Compose

Architecture

```
six-degrees/
├── backend/
│   ├── SixDegrees.Api/            # Contrôleurs, hubs SignalR, config
│   ├── SixDegrees.Domain/         # Logique métier, services, entités
│   └── SixDegrees.Infrastructure/ # Accès données, migrations EF Core
├── frontend/
│   └── src/
│       ├── pages/                 # Vues (Home, DailyChallenge, Lobby…)
│       ├── components/            # Composants réutilisables
│       └── hooks/                 # Hooks custom (SignalR, username…)
├── docker-compose.yml
└── .env.example
```

Algorithme

Bidirectional BFS pour trouver le chemin le plus court entre deux acteurs (profondeur max = 6 degrés).

Modèle de données

Stocké en PostgreSQL via Entity Framework Core :

- **Person** `{ id, name, popularity }` — acteurs/célébrités
- **Title** `{ id, name, year, type }` — films et séries
- **Relation** `Person → Title` (ACTED_IN)

Données importées depuis l'API TMDB.

Lancer le projet

Prérequis

- Docker et Docker Compose
- Clés API TMDB (compte gratuit sur themoviedb.org)

Démarrage rapide

```
cp .env.example .env
# Remplir TMDB_API_KEY, TMDB_ACCESS_TOKEN, etc.
docker compose up --build
```

Application disponible sur `http://localhost` (ou `APP_PORT`).

Variables d'environnement

Variable	Description
`TMDB_API_KEY`	Clé API TMDB (requise)
`TMDB_ACCESS_TOKEN`	Token d'accès TMDB (requis)
`DB_CONNECTION_STRING`	Chaîne de connexion PostgreSQL
`APP_PORT`	Port exposé pour le frontend (défaut : `80`)

Développement local

Backend

```
cd backend
dotnet restore
dotnet run --project SixDegrees.Api/
# API sur http://localhost:8080
# Swagger : http://localhost:8080/swagger
```

Frontend

```
cd frontend
npm install
npm run dev
# Dev server sur http://localhost:5173
```

CI/CD

Workflow Gitea (`.gitea/workflows/deploy.yaml`) déclenché automatiquement sur chaque tag `v*.*.*` — rebuild et redéploiement via Docker Compose.

```
git tag v1.0.0
git push --tags
```

Notes de conception initiale

Lors de la phase de prototype, plusieurs choix ont été évalués :

Critère	Option retenue	Alternatives envisagées
BDD graphe	PostgreSQL + EF Core	Apache AGE (extension PG), Neo4j
Backend	ASP.NET Core (C#)	Node.js / Express
Algo chemin	Bidirectional BFS	BFS simple, Dijkstra
Source données	TMDB	Wikidata, IMDb

Scoring (mode challenge)

- **Base** : `(6 - degrés) × 100` points
- **Bonus temps** : +50 pts si trouvé en moins d'1 minute
- **Bonus rareté** : +50 pts si le chemin inclut un titre peu populaire

---

Revision #6
Created 8 October 2025 16:08:42 by Hugo
Updated 29 March 2026 14:38:12 by Hugo