OBJECTIFS DE LA SÉANCE

À la fin de cette séance, vous serez capable de :

- Installer et configurer votre environnement de développement
- Comprendre l'architecture du projet
- Créer la structure de base des trois applications
- Configurer Tailwind CSS 4.1
- Vérifier que tout fonctionne correctement

VUE D'ENSEMBLE DU PROJET

DESCRIPTION

Vous allez créer une application de location de films type Netflix composée de :

- **Frontend Utilisateur** : Application React (CSR - Client Side Rendering)
- **Frontend Admin** : Application Express avec EJS (SSR - Server Side Rendering)
- **Backend API** : API REST avec Express et MongoDB

TECHNOLOGIES UTILISÉES

- **Frontend User** : React, Vite, Tailwind CSS 4.1, React Router
- **Frontend Admin** : Express, EJS, Tailwind CSS 4.1
- **Backend** : Express, MongoDB, Mongoose, JWT
- **Outils** : Node.js, npm, Git

ARCHITECTURE DU REPO

```
netflix-project/
├── frontend-user/ # React SPA (Client Side Rendering)
├── frontend-admin/ # Express SSR (Server Side Rendering)
├── backend/ # API Express + MongoDB
├── shared/ # Code partagé
├── data/ # Données mock (JSON)
└── docs/ # Documentation
```

🔧 PARTIE 1 : INSTALLATION DES OUTILS (45 MIN)

1.1 VÉRIFICATION DES PRÉREQUIS

- Ouvrez un terminal et vérifiez que vous avez :

Node.js (version 18 ou supérieure)

```bash
node --version # Devrait afficher : v24.x.x ou supérieur
```

npm (Node Package Manager)

```
npm --version # Devrait afficher : 11.x.x ou supérieur
```

Git

```
git --version # Devrait afficher : git version 2.x.x
```

💡 Si une commande ne fonctionne pas :

- Node.js : Téléchargez depuis nodejs.org
- Git : Téléchargez depuis git-scm.com

1.2 INSTALLATION DE VS CODE ET EXTENSIONS

- Si vous ne l'avez pas , téléchargez VS Code : code.visualstudio.com

Extensions recommandées (à installer depuis VS Code) :

ES7+ React/Redux/React-Native snippets	Raccourcis pour React	ID : dsznajder.es7-react-js-snippets
Tailwind CSS IntelliSense	Autocomplétion Tailwind	ID : bradlc.vscode-tailwindcss
Prettier - Code formatter	Formatage automatique du code	ID : esbenp.prettier-vscode
ESLint	Détection d'erreurs JavaScript	ID : dbaeumer.vscode-eslint

MongoDB for VS Code (optionnel)	Gestion de MongoDB	ID : mongodb.mongodb-vscode
Thunder Client (optionnel)	Tester les API REST	ID : rangav.vscode-thunder-client

🏗️ PARTIE 2 : CRÉATION DE LA STRUCTURE (45 MIN)

2.1 INITIALISATION DU PROJET

- Ouvrez un terminal et créez la structure :

```
# Créer le dossier principal
mkdir netflix-project
cd netflix-project

# Initialiser Git
git init

# Créer les sous-dossiers
mkdir frontend-user frontend-admin backend shared data docs

# Créer le fichier .gitignore global
cat > .gitignore << EOF
node_modules/
.env
.DS_Store
dist/
build/
*.log
.vscode/
EOF
```

2.2 CONFIGURATION FRONTEND USER (REACT + VITE + TAILWIND 4.1)

- Créez le projet React avec Vite :

```
cd frontend-user
npm create vite@latest . -- --template react
```

- Répondre aux questions :

- Current directory is not empty. Continue? → **Yes**
- Select a framework → **React**
- Select a variant → **JavaScript**

- Installez les dépendances :

```
npm install
```

- Installez Tailwind CSS 4.1 :

```
npm install tailwindcss@next @tailwindcss/vite@next
```

- Configurez Vite (vite.config.js) :

```
import { defineConfig } from 'vite'
import react from '@vitejs/plugin-react'
import tailwindcss from '@tailwindcss/vite'

export default defineConfig({
  plugins: [react(), tailwindcss()],
  server: {
    port: 3000,
    proxy: {
      '/api': 'http://localhost:5000'
    }
  }
})
```

- Configurez Tailwind (src/index.css) :

```
@import "tailwindcss";

@theme {
  --color-primary: #e50914;
  --color-primary-dark: #b20710;
  --font-sans: 'Inter', system-ui, sans-serif;
}
```

- Créez la structure des dossiers :

```
# Dans frontend-user/src/
mkdir components pages services context utils
mkdir components/common components/movies components/layout
```

- Testez l'application :

```
npm run dev
```

- Ouvrez l'url http://localhost:3000 dans le navigateur.

2.3 CONFIGURATION BACKEND (EXPRESS + MONGODB)

```
cd ../../backend
npm init -y
```

- Installez les dépendances :

```
npm install express mongoose dotenv cors jsonwebtoken bcryptjs
npm install -D nodemon
```

- Modifiez le fichier package.json :

```
{
  "name": "netflix-backend",
  "version": "1.0.0",
  "type": "module",
  "scripts": {
    "dev": "nodemon src/server.js",
    "start": "node src/server.js",
    "seed": "node src/utils/seed.js"
  }
}
```

- Créez la structure :

```
mkdir src
cd src
mkdir config models routes controllers middleware utils
touch server.js
```

- Créez le fichier .env :

```
cd ..
cat > .env << EOF
PORT=5000
NODE_ENV=development
MONGODB_URI=mongodb://localhost:27017/netflix
JWT_SECRET=votre_secret_super_securise_changez_moi
JWT_EXPIRE=7d
CLIENT_URL=http://localhost:3000
EOF
```

- Créez le fichier src/server.js :

```
import express from 'express';
import dotenv from 'dotenv';
import cors from 'cors';

dotenv.config();

const app = express();
const PORT = process.env.PORT || 5000;

// Middlewares
app.use(cors({
  origin: process.env.CLIENT_URL || 'http://localhost:3000',
  credentials: true
}));
app.use(express.json());
app.use(express.urlencoded({ extended: true }));

// Route de test
```

```
app.get('/api/health', (req, res) => {
  res.json({
    status: 'OK',
    message: 'API Netflix is running',
    timestamp: new Date().toISOString()
  });
});

// Démarrer le serveur
app.listen(PORT, () => {
  console.log(`🚀 Server running on http://localhost:${PORT}`);
  console.log(`📝 Environment: ${process.env.NODE_ENV}`);
});
```

- Testez le backend :

```
npm run dev
```

- Ouvrez l'url http://localhost:5000/api/health dans le navigateur.

2.4 CONFIGURATION FRONTEND ADMIN (EXPRESS SSR + EJS)

```
cd ../../frontend-admin
npm init -y
```

- Installez les dépendances :

```
npm install express ejs express-session cookie-parser axios
npm install -D nodemon concurrently
npm install tailwindcss@next @tailwindcss/cli@next
```

- Modifiez le fichier package.json :

```
{
  "name": "netflix-admin",
  "version": "1.0.0",
```

```
  "type": "module",
  "scripts": {
    "dev": "concurrently \"nodemon server.js\" \"npm run watch:css\"",
    "watch:css": "tailwindcss -i ./public/css/input.css -o ./public/css/output.css --watch",
    "build:css": "tailwindcss -i ./public/css/input.css -o ./public/css/output.css --minify",
    "start": "node server.js"
  }
}
```

- Créez la structure :

```
mkdir views public utils middleware
mkdir views/layouts views/partials views/auth views/dashboard views/movies
mkdir public/css public/js public/images
```

- Initialisez Tailwind :

```
npx tailwindcss init –esm
```

- Configurez Tailwind (tailwind.config.js) :

```
export default {
  content: ['./views/**/*.ejs', './public/**/*.js'],
  theme: {
    extend: {
      colors: {
        primary: '#e50914',
        'primary-dark': '#b20710',
      }
    }
  }
}
```

- Créez un fichier public/css/input.css :

```
@import "tailwindcss";

@theme {
  --color-primary: #e50914;
  --color-primary-dark: #b20710;
}
```

- Créez le fichier server.js :

```
import express from 'express';
import path from 'path';
import { fileURLToPath } from 'url';

const __filename = fileURLToPath(import.meta.url);
const __dirname = path.dirname(__filename);

const app = express();
const PORT = process.env.PORT || 4000;

// Configuration EJS
app.set('view engine', 'ejs');
app.set('views', path.join(__dirname, 'views'));

// Middlewares
app.use(express.static('public'));
app.use(express.json());
app.use(express.urlencoded({ extended: true }));

// Route de test
app.get('/', (req, res) => {
```

```
  res.render('index', { title: 'Admin Netflix' });
});

app.listen(PORT, () => {
  console.log(`🎬 Admin panel running on http://localhost:${PORT}`);
});
```

- Créez le fichier views/index.ejs :

```
<!DOCTYPE html>
<html lang="fr">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title><%= title %></title>
  <link href="/css/output.css" rel="stylesheet">
</head>
<body class="bg-gray-900 text-white min-h-screen flex items-center justify-center">
  <div class="text-center">
    <h1 class="text-5xl font-bold text-primary mb-4">NETFLIX</h1>
    <p class="text-xl text-gray-400">Admin Panel</p>
  </div>
</body>
</html>
```

- Compilez Tailwind et démarrez :

```
npm run dev
```

- Ouvrir http://localhost:4000 dans le navigateur.

PARTIE 3 : DONNEES MOCK (30 MIN)

3.1 CREER LE FICHIER DE DONNEES

- Retournez à la racine du projet

```
cd ..
cd data
```

- Créez un fichier movies.json , avec le contenu suivant:

```
[
 {
  "id": 1,
  "title": "Inception",
  "description": "Un voleur qui s'infiltre dans les rêves des gens pour voler leurs secrets doit accomplir l'inverse : implanter une idée.",
  "poster": "https://image.tmdb.org/t/p/w500/9gk7adHYeDvHkCSE-qAvQNLV5Uge.jpg",
  "backdrop": "https://image.tmdb.org/t/p/original/s3TBrRGB1iav7gFOCNx3H31MoES.jpg",
  "genre": "Science-Fiction",
  "year": 2010,
  "duration": 148,
  "price": 3.99,
  "rating": 8.8
 },
 {
  "id": 2,
  "title": "The Dark Knight",
  "description": "Batman affronte le Joker, un criminel anarchiste qui veut plonger Gotham dans le chaos.",
  "poster": "https://image.tmdb.org/t/p/w500/qJ2tW6WMUDux911r6m7haRef0WH.jpg",
  "backdrop": "https://image.tmdb.org/t/p/original/hkBaDkMWbLaf8B1lsWsKX7Ew3Xq.jpg",
```

```
  "genre": "Action",
  "year": 2008,
  "duration": 152,
  "price": 3.99,
  "rating": 9.0
 },
 {
  "id": 3,
  "title": "Interstellar",
  "description": "Une équipe d'explorateurs voyage à travers un trou de ver dans l'espace pour assurer la survie de l'humanité.",
  "poster": "https://image.tmdb.org/t/p/w500/gEU2QniE6E77NI6lCU6MxlN-BvIx.jpg",
  "backdrop": "https://image.tmdb.org/t/p/original/xu9zaAevzQ5nnrsXN6JcahLnG4i.jpg",
  "genre": "Science-Fiction",
  "year": 2014,
  "duration": 169,
  "price": 4.99,
  "rating": 8.6
 },
 {
  "id": 4,
  "title": "Pulp Fiction",
  "description": "Les histoires entrelacées de criminels, d'un boxeur et d'un couple de braqueurs dans le Los Angeles criminel.",
  "poster": "https://image.tmdb.org/t/p/w500/d5iIlFn5s0ImszYzBPb8JPIfbXD.jpg",
  "backdrop": "https://image.tmdb.org/t/p/original/suaE-Otk1N1sgg2MTM7oZd2cfVp3.jpg",
  "genre": "Thriller",
  "year": 1994,
  "duration": 154,
```

```
  "price": 3.99,
  "rating": 8.9
 },
 {
  "id": 5,
  "title": "Forrest Gump",
  "description": "L'histoire de Forrest Gump, un homme simple d'esprit qui traverse plusieurs décennies de l'histoire américaine.",
  "poster": "https://image.tmdb.org/t/p/w500/arw2vcBveWOVZr6pxd9XTd1TdQa.jpg",
  "backdrop": "https://image.tmdb.org/t/p/original/7c9UVPPiTPltouxRVY6N9uYzpxx.jpg",
  "genre": "Drame",
  "year": 1994,
  "duration": 142,
  "price": 3.99,
  "rating": 8.8
 }
]
```

PARTIE 4 : TESTS ET VERIFICATION (10 MIN)

4.1 CHECKLIST FINALE

- Vérifiez que tout fonctionne :

Frontend User (React) :

- npm run dev démarre sans erreur
- http://localhost:3000 affiche une page
- Tailwind CSS fonctionne (styles appliqués)

Backend (Express) :

- npm run dev démarre sans erreur
- http://localhost:5000/api/health retourne du JSON
- Pas d'erreur dans la console

Frontend Admin (Express SSR) :

- npm run dev compile Tailwind et démarre le serveur
- http://localhost:4000 affiche la page admin
- Les styles Tailwind sont appliqués

4.2 STRUCTURE FINALE

Votre projet devrait ressembler à :

```
netflix-project/
├── frontend-user/
│   ├── node_modules/
│   ├── public/
│   ├── src/
│   │   ├── components/
│   │   ├── pages/
│   │   ├── context/
```

```
|   |   ├── services/
|   |   ├── App.jsx
|   |   ├── main.jsx
|   |   └── index.css
|   ├── .gitignore
|   ├── package.json
|   └── vite.config.js
├── frontend-admin/
|   ├── node_modules/
|   ├── public/
|   |   ├── css/
|   |   └── js/
|   ├── views/
|   ├── server.js
|   ├── package.json
|   └── tailwind.config.js
├── backend/
|   ├── node_modules/
|   ├── src/
|   |   ├── config/
|   |   ├── models/
|   |   ├── routes/
|   |   ├── controllers/
|   |   ├── middleware/
|   |   ├── utils/
```

```
│   │   └── server.js
│   ├── .env
│   └── package.json
├── data/
│   └── movies.json
└── .gitignore
```

EXERCICE DE VALIDATION

Objectif :

Créez une page "Hello World" personnalisée dans chaque application pour vérifier que tout fonctionne.

FRONTEND USER

- Modifier frontend-user/src/App.jsx :

```
function App() {
  return (
    <div className="min-h-screen bg-black text-white flex items-center justify-center">
      <div className="text-center">
        <h1 className="text-6xl font-bold text-primary mb-4">
          Hello Netflix 🎬
        </h1>
        <p className="text-xl text-gray-400">
          Frontend User - React + Tailwind 4.1
        </p>
      </div>
    </div>
  )
}
export default App
```

BACKEND

- Ajouter une route personnalisée dans backend/src/server.js :

```
app.get('/api/movies', (req, res) => {
  res.json({
    success: true,
    message: 'API Movies endpoint',
    data: [
      { id: 1, title: 'Inception', year: 2010 },
      { id: 2, title: 'The Dark Knight', year: 2008 }
    ]
  });
});
```

- Testez en ouvrant la page : http://localhost:5000/api/movies

FRONTEND ADMIN

- Modifiez le fichier frontend-admin/views/index.ejs :

```
<body class="bg-gray-900 text-white min-h-screen flex items-center justify-center">
  <div class="text-center">
    <h1 class="text-5xl font-bold text-primary mb-4">NETFLIX Admin</h1>
    <p class="text-xl text-gray-400 mb-8">Panel d'administration SSR</p>
    <div class="inline-block bg-gray-800 px-6 py-3 rounded-lg border border-gray-700">
      <p class="text-sm text-gray-400">Serveur Express avec EJS</p>
    </div>
  </div>
</body>
```

COMMANDES IMPORTANTES

FRONTEND USER

```
cd frontend-user
npm run dev          # Démarrer en mode développement
```

BACKEND

```
cd backend
npm run dev          # Démarrer avec nodemon
npm run seed         # Initialiser la base de données (plus tard)
```

FRONTEND ADMIN

```
cd frontend-admin
npm run dev          # Démarrer + compiler Tailwind
```

Ports utilisés

- **3000** : Frontend User (React)
- **4000** : Frontend Admin (Express SSR)
- **5000** : Backend API (Express)