


TP Pratique - Explorateur d'API Thématique

Durée : 3 heures | 1ère année Web/JavaScript

Objectif du TP

Créer une application web interactive qui exploite une API REST publique pour afficher, rechercher et explorer des données thématiques. Inspiré du Pokédex que nous avons réalisé en cours avec l'API Tyradex Pokemon, vous allez créer votre propre explorateur de données !

Choix du sujet

 **Règle importante** : Chaque étudiant doit choisir un sujet différent de son voisin direct.

API Animaux (Zoo ou Animaux Domestiques)

- Lister les animaux disponibles
- Rechercher un animal par nom ou espèce
- Voir les détails (âge, habitat, alimentation, régime alimentaire, etc.)
- **Suggestions d'API** : Zoo API, Cat API, Dog API, Animal Facts API

API Films

- Afficher des films sur la page
- Rechercher un film spécifique
- Afficher les détails d'un film (réalisateur, acteurs, année, synopsis, etc.)
- **Suggestions d'API** : TMDb API, OMDB API, Studio Ghibli API

API Météo

- Lister des villes sur la page
- Rechercher une ville spécifique
- Voir le détail de la météo pour une ville (température, conditions, prévisions)
- **Suggestions d'API** : OpenWeatherMap, WeatherAPI, AccuWeather

API Recettes de Cuisine

- Lister des recettes sur la page
- Rechercher une recette spécifique
- Voir le détail d'une recette (ingrédients, instructions, temps de préparation)
- **Suggestions d'API** : Spoonacular, TheMealDB, Recipe Puppy

API Jeux Vidéo

- Afficher une liste de jeux vidéo
- Rechercher un jeu par nom
- Voir les détails d'un jeu (studio, plateforme, genre, note, date de sortie)
- **Suggestions d'API** : RAWG API, IGDB API, Cheapshark API



Méthodologie en 4 étapes



Étape 1 : Recherche et sélection d'API (30 min)

- ☐ Identifier une API publique correspondant à votre sujet
- ☐ Vérifier que l'API est accessible (pas besoin d'authentification complexe)
- ☐ Lire la documentation de l'API
- ☐ Noter les endpoints principaux et les paramètres



Ressources pour trouver des APIs :

- [Public APIs GitHub](#)
- [RapidAPI Hub](#)
- [APIList](#)



Étape 2 : Test avec Bruno (30 min)

- ☐ Installer et configurer Bruno (ou Postman/Insomnia)
- ☐ Tester l'endpoint de liste (GET)
- ☐ Tester l'endpoint de recherche (si disponible)
- ☐ Tester l'endpoint de détails d'un élément
- ☐ Documenter les réponses et la structure des données



Étape 3 : Création HTML/CSS (45 min)

- ☐ Créer la structure HTML de base
- ☐ Implémenter le design CSS (IA autorisée pour inspiration)
- ☐ Créer les composants visuels : liste, carte de détail, barre de recherche
- ☐ Rendre l'interface responsive



IA autorisée pour : Génération de palettes de couleurs, suggestions de layout, code CSS de base



Étape 4 : Intégration JavaScript (75 min)

- ☐ Récupérer les données depuis l'API
- ☐ Afficher la liste des éléments
- ☐ Implémenter la fonction de recherche
- ☐ Créer l'affichage des détails
- ☐ Gérer les erreurs et les états de chargement

Fonctionnalités attendues

Fonctionnalités OBLIGATOIRES (15 points)

Affichage de liste (5 points)

- ☐ Récupération des données depuis l'API
- ☐ Affichage en grille ou liste
- ☐ Au moins 3 informations par élément (nom, image, info supplémentaire)
- ☐ Design cohérent et lisible

Fonction de recherche (4 points)

- ☐ Barre de recherche fonctionnelle
- ☐ Filtrage en temps réel ou par validation
- ☐ Gestion du cas "aucun résultat"
- ☐ Réinitialisation possible

Page/Modal de détails (4 points)

- ☐ Clic sur un élément ouvre les détails
- ☐ Affichage de toutes les informations importantes
- ☐ Navigation retour vers la liste
- ☐ Mise en page soignée

Gestion des erreurs (2 points)

- ☐ Gestion des erreurs API
- ☐ États de chargement (loader/spinner)
- ☐ Messages informatifs pour l'utilisateur

Fonctionnalités BONUS (5 points supplémentaires)

Interface avancée (2 points)

- ☐ **Animations et transitions** (1 point)
- ☐ **Mode sombre/clair** (1 point)

Fonctionnalités premium (3 points)

- ☐ **Système de favoris** (1 point) - Sauvegarder ses éléments préférés
- ☐ **Filtres avancés** (1 point) - Par catégorie, date, popularité, etc.
- ☐ **Pagination ou scroll infini** (1 point) - Gérer de grandes listes

Structure de fichiers

```
mon-projet/  
├─ index.html      # Page principale  
├─ style.css       # Styles CSS  
├─ script.js       # Logique JavaScript  
├─ assets/         # Images et ressources (optionnel)  
└─ README.md       # Documentation de votre API (optionnel)
```

💡 Exemple de structure JavaScript

```

// Configuration API
const API_BASE_URL = 'https://votre-api.com';
const API_KEY = 'votre-clé-si-nécessaire';

// Fonctions principales
async function fetchData(endpoint) {
  try {
    const response = await fetch(`${API_BASE_URL}${endpoint}`);
    if (!response.ok) throw new Error('Erreur API');
    return await response.json();
  } catch (error) {
    console.error('Erreur:', error);
    showError('Impossible de charger les données');
  }
}

async function loadItems() {
  showLoader();
  const data = await fetchData('/items');
  displayItems(data);
  hideLoader();
}

function displayItems(items) {
  const container = document.getElementById('items-container');
  container.innerHTML = items.map(item => createItemCard(item)).join('');
}

function createItemCard(item) {
  return `
    <div class="item-card" onclick="showDetails(${item.id})">
      
      <h3>${item.name}</h3>
      <p>${item.description}</p>
    </div>
  `;
}

// Fonction de recherche
function searchItems(query) {
  // Filtrer les données ou faire une nouvelle requête API
}

// Affichage des détails
function showDetails(itemId) {

```

```
// Récupérer et afficher les détails complets
```

```
}
```

Grille d'évaluation

Critère	Points	Description
API et tests Bruno	2	API trouvée et testée correctement
Affichage liste	5	Liste complète et bien présentée
Fonction recherche	4	Recherche fonctionnelle et intuitive
Détails élément	4	Informations complètes et navigation
Gestion erreurs	2	États de chargement et erreurs gérés
Code qualité	3	Code propre, commenté et organisé
BONUS Animations	+1	Transitions et micro-interactions
BONUS Mode sombre	+1	Toggle dark/light mode
BONUS Favoris	+1	Système de sauvegarde local
BONUS Filtres	+1	Filtres par catégories/propriétés
BONUS Pagination	+1	Gestion de grandes quantités de données
Total	/20	(15 + 5 bonus max)

Conseils pour réussir

Recherche d'API

- Privilégiez les APIs sans authentification complexe
- Vérifiez les limites de taux (rate limits)
- Testez avec Bruno AVANT de coder

Développement

- Commencez par les fonctionnalités de base
- Testez régulièrement dans le navigateur
- Gérez les cas d'erreur dès le début
- Utilisez `console.log()` pour déboguer

Design

- Inspirez-vous d'applications existantes
- Gardez l'interface simple et intuitive
- Assurez-vous que tout soit lisible
- Testez sur mobile

⚠ Points d'attention

- **CORS** : Certaines APIs peuvent bloquer les requêtes depuis le navigateur
- **Rate Limiting** : Ne pas abuser des requêtes API
- **Données manquantes** : Toujours vérifier si les propriétés existent
- **Responsive** : Votre app doit fonctionner sur mobile
- **Performance** : Optimiser les images et requêtes

📖 Ressources utiles

- **Bruno/Postman** : Pour tester les APIs
- **MDN Web Docs** : Documentation JavaScript
- **Can I Use** : Compatibilité des fonctionnalités web
- **CSS Grid Generator** : Pour les layouts
- **Color Hunt** : Palettes de couleurs

✍ Livrables

- **Fichiers sources** : HTML, CSS, JS complets
- **Démonstration** : Application fonctionnelle
- **Documentation Bruno** : Collection des tests API
- **Présentation** (3 min) : Demo et explication des choix techniques

Bonne exploration ! 🚀

N'oubliez pas : l'objectif est d'apprendre en pratiquant. Commencez simple et ajoutez des fonctionnalités progressivement !