ITuniversity

Compléments Web Services

ITuniversity

Objectifs du module

- Comprendre les bonnes pratiques pour structurer les réponses JSON d'une API.
- Uniformiser les formats de **succès** et **erreurs**.
- Définir des conventions de nommage claires.
- Introduire l'usage de métadonnées.

Il existe 3 modèles

- Le modèle robuste
- Le modèle REST pur
- Le modèle inspiré des APIs (Stripe, Twitter)

ITuniversity

Modèle Robuste

📦 Format JSON — Exemple de réponse

Un format cohérent facilite la maintenance et l'intégration.

Modèle robuste

```
{
  "status": "success",
  "data": {},
  "error": null
}
```

ITuniversity

Exemple de succès

```
{
  "status": "success",
  "data": {
    "studentId": 123,
    "notesS1": [12, 15, 17]
  },
  "error": null
}
```

ITuniversity

Format JSON — Réponse d'erreur

Toujours **la même structure** que la réponse de succès.

Exemple :

```
{
  "status": "error",
  "data": null,
  "error": {
    "code": "STUDENT_NOT_FOUND",
    "message": "Aucun étudiant trouvé.",
    "details": null
  }
}
```

ITuniversity

Format JSON — Interêt uniformisation

Toutes les réponses ont la même forme JSON, ce qui simplifie :

- les tests
- les parseurs front-end
- la documentation
  les middlewares

Le front sait toujours lire :

- status → succès ou erreur
- data → résultats
- error → null ou objet erreur

ITuniversity

Format JSON — Interêt uniformisation

Côté front, on utilise toujours le même code

```
if (res.status === "success") {
    // traiter les data
} else {
    // afficher l'erreur
}
```

```
{
  "status": "success",
  "data": {},
  "error": null
}
```

ITuniversity

Conventions de nommage

Pour les clés JSON :

✓ anglais

✓ cohérence

✓ privilégier **camelCase**

Exemples :

Élément	camelCase	snake_case
Nom de clé	studentName	student_name
Pagination	currentPage	current_page

Métadonnées (meta)

Utiles pour la pagination, version ou info complémentaire.

```
{
  "meta": {
    "page": 1,
    "perPage": 10,
    "total": 57,
    "apiVersion": "1.0.0"
  }
}
```

ITuniversity

Gestion des erreurs

Inclure :

- Un **message descriptif**
- Un **code interne**
- Un champ facultatif `details`
- Le code HTTP approprié

Correspondance recommandée

Cas	Code HTTP	Code interne
Étudiant introuvable	404	STUDENT_NOT_FOUND
Non authentifié	401	UNAUTHORIZED
Erreur BD	500	DB_ERROR
Erreur inconnue	500	UNKNOWN_ERROR

Résumé

- Toujours retourner une structure uniforme.
- Anglais + camelCase : standard international.
- Ajouter `meta` lorsque pertinent.
- Erreurs = messages + codes internes + cohérence.

Modèle REST pur

Le standard officiel des API REST

ITuniversity

Principes du REST Pur

- Structure **différente** entre succès et erreur
- Le **code HTTP** porte la signification principale
- Réponses **épurées**, conformes aux standards
- Compatible **Swagger / OpenAPI**
- Pas de champs inutiles (“status”, “error”: null, etc.)

ITuniversity

Exemple REST Pur — Succès

`GET /students/123/notes`

HTTP 200 OK

```
{
  "notes": [14, 16, 18]
}
```

ITuniversity

Exemple REST Pur — Étudiant introuvable

`GET /students/999/notes`

HTTP 404 Not Found

```
{
  "error": "Student not found",
  "code": "STUDENT_NOT_FOUND"
}
```

ITuniversity

Exemple REST Pur — Erreur Système

HTTP 500 Internal Server Error

```
{
  "error": "Internal server error",
  "code": "INTERNAL_ERROR"
}
```

Quand utiliser le REST Pur ?

- ✓ API publiques
- ✓ Applications mobiles
- ✓ API exposées à des partenaires
- ✓ Conformité stricte aux normes REST

Modèle Inspiré des Grandes APIs

(Stripe, GitHub, Twitter API v2)

Principes des grandes APIs

- Séparation stricte des concepts : `object`, `error`, `errors`, `type`
- Erreurs en **liste** (meilleure extensibilité)
- Réponses très lisibles
- Parfait pour les SDK clients
- Haut niveau de professionnalisme

ITuniversity

Style Stripe — Succès

```
{
  "object": "student_notes",
  "id": 123,
  "notes": [14, 16, 18]
}
```

ITuniversity

Style Stripe / GitHub — Erreur

```
{
  "error": {
    "type": "invalid_request_error",
    "code": "student_not_found",
    "message": "The student with ID 458 does not exist.",
    "param": "student_id"
  }
}
```

ITuniversity

Style Twitter / GitHub — Liste d'erreurs

```
{
  "errors": [
    {
      "code": "student_not_found",
      "title": "Student not found",
      "detail": "No student matches ID 458."
    }
  ]
}
```

Quand utiliser ce modèle ?

- ✓ APIs SaaS
- ✓ Fintech (style Stripe)
- ✓ APIs exposées publiquement
- ✓ Versioning actif
- ✓ Microservices + Gateway

📚 Ressources conseillées

- REST API Design — Microsoft
- API Best Practices — Mozilla MDN
- Google JSON Style Guide
- RFC 8259 — JSON Standard