
# 2. Etude comparative XML et JSON et YAML. Application de gestion de l'emploi du temps

| | |
|---|---|
| Prof | Mr Amadou TALL |
| Matière | Technologie XML et Securite |
| Formation | M1-TDSI |
| Créée par | Abdoulaye Guene |
| E-mail | abdoulaye.guene92@gmail.com |
| État | Terminé |
| Depo Git | https://github.com/Abdoulayeguene/projet-xml-emploi-du-temps.git |
| Projet | Etude comparative XML et JSON et YAML puis Creation d'une<br>application de Gestion de l'emploi du temps en utilisant le langage<br>PHP ou Java et un fichier XML. |
| Realiser par: | a) Abdoulaye Guene<br>b) Aissatou Billy Sall |

# Application de gestion de l'emploi du temps

**Réalisé par :**

- Abdoulaye Guene
- Aissatou Billy Sall

## Introduction générale

Dans un contexte où les systèmes informatiques échangent de plus en plus de données entre applications hétérogènes, la structuration, la validation et la fiabilité des données deviennent des enjeux majeurs. Les formats de données structurées jouent alors un rôle essentiel pour garantir la cohérence, la lisibilité et l'interopérabilité des informations.

Parmi ces formats, XML occupe une place particulière dans le monde académique et professionnel grâce à sa capacité à décrire des données complexes, à imposer des contraintes de structure et à assurer une validation rigoureuse. D'autres formats comme JSON et YAML sont également largement utilisés, notamment dans le développement web moderne et les environnements DevOps.

L'objectif de ce projet est d'étudier ces différents formats de données, puis de mettre en pratique le langage XML à travers la réalisation d'une application de gestion d'emploi du temps. Cette application repose sur un fichier XML validé par un DTD et un schéma XSD, et exploité dynamiquement à l'aide du langage PHP.

## Chapitre I : Le langage XML

### 1.1 Définition du XML

XML (eXtensible Markup Language) est un langage de balisage conçu pour stocker, structurer et transporter des données. Contrairement au HTML, dont le rôle principal est l'affichage, XML se concentre sur la description du contenu et de sa structure.

XML est extensible, ce qui signifie que l'utilisateur peut définir ses propres balises en fonction de ses besoins métiers. Il est indépendant des plateformes

et des langages de programmation, ce qui en fait un format largement utilisé pour l'échange de données.

## 1.2 Structure d'un document XML

Un document XML est constitué de balises organisées de manière hiérarchique. Il comporte un élément racine unique qui englobe tous les autres éléments. Les données sont encapsulées dans des balises ouvrantes et fermantes, et peuvent contenir des attributs.

La structure doit respecter certaines règles strictes :

- Les balises doivent être correctement imbriquées
- Le document doit être bien formé
- Les noms de balises sont sensibles à la casse
- Un seul élément racine est autorisé
- Les valeurs d'attributs doivent être entre guillemets

## 1.3 Utilisations du XML

XML est utilisé dans de nombreux domaines :

- Échange de données entre applications hétérogènes
- Stockage de données structurées et configuration d'applications
- Fichiers de configuration (Maven, Spring, Android)
- Services web SOAP et interopérabilité
- Formats de documents (DOCX, ODT, SVG)
- Flux RSS et syndication de contenu

## 1.4 Exemple de document XML

Un document XML simple représentant un étudiant :

```xml
<?xml version="1.0" encoding="UTF-8"?>
<etudiant id="ETU001">
    <nom>SALL</nom>
    <prenom>Aissatou Billy</prenom>
    <age>23</age>
    <filiere>Informatique</filiere>
```

```
    <niveau>Master</niveau>
    <adresse>
        <rue>Avenue Cheikh Anta Diop</rue>
        <ville>Dakar</ville>
        <pays>Sénégal</pays>
    </adresse>
    <cours>
        <matiere code="INF301">Bases de données</matiere>
    </cours>
</etudiant>
```

Ce document illustre la capacité de XML à représenter des données hiérarchiques complexes avec des attributs et des éléments imbriqués.

# Chapitre II : Le format JSON

## 2.1 Définition du JSON

JSON (JavaScript Object Notation) est un format léger d'échange de données, principalement utilisé dans les applications web modernes. Créé à partir de la syntaxe JavaScript, il est devenu un standard indépendant du langage, facilement lisible par les humains comme par les machines.

JSON est basé sur deux structures universelles : les objets (collections de paires clé/valeur) et les tableaux (listes ordonnées de valeurs). Sa simplicité et sa légèreté en font le format privilégié pour les APIs REST.

## 2.2 Structure du JSON

Un document JSON est composé d'objets délimités par des accolades `{}` et de tableaux délimités par des crochets `[]`. Les données sont organisées sous forme de paires clé/valeur séparées par deux points `:`.

Les types de données supportés sont :

- Chaînes de caractères (entre guillemets doubles)
- Nombres (entiers ou décimaux)
- Booléens (true/false)
- Null
- Objets

- Tableaux

## 2.3 Utilisations du JSON

JSON est largement utilisé pour :

- Les APIs REST et communication client-serveur
- Les applications web dynamiques (AJAX, Single Page Applications)
- La configuration d'applications (package.json, tsconfig.json)
- Le stockage de données NoSQL (MongoDB, CouchDB)
- L'échange de données entre microservices

## 2.4 Exemple JSON

Un exemple JSON représentant un emploi du temps :

```
{
  "emploiDuTemps": {
    "etablissement": "Université Cheikh Anta Diop",
    "departement": "Informatique",
    "niveau": "Master",
    "semaines": [
      {
        "numero": 1,
        "jours": [
          {
            "nom": "Lundi",
            "date": "2026-02-16",
            "cours": [
              {
                "id": "C001",
                "matiere": "Bases de données avancées",
                "enseignant": {
                  "nom": "Dr. Ndiaye",
                  "prenom": "Mamadou"
                },
                "horaire": {
                  "debut": "08:00",
                  "fin": "10:00"
```

```
                },
                "salle": "Amphi A",
                "type": "Cours magistral"
              },
              {
                "id": "C002",
                "matiere": "Programmation Web",
                "enseignant": {
                  "nom": "Mme Diop",
                  "prenom": "Fatou"
                },
                "horaire": {
                  "debut": "10:15",
                  "fin": "12:15"
                },
                "salle": "Labo 2",
                "type": "Travaux pratiques"
              }
            ]
          }
        ]
      }
    ]
  }
}
```

Cet exemple montre la capacité de JSON à représenter des données structurées de manière concise, mais sans mécanisme natif de validation aussi strict que XML.

# Chapitre III : Le format YAML

## 3.1 Définition du YAML

YAML (YAML Ain't Markup Language) est un format de sérialisation de données orienté vers la lisibilité humaine. Créé en 2001, YAML se positionne comme une alternative plus lisible à XML et JSON pour les fichiers de configuration.

YAML est un sur-ensemble de JSON, ce qui signifie que tout document JSON valide est également un document YAML valide. Il utilise l'indentation pour

représenter la structure hiérarchique, éliminant ainsi le besoin de délimiteurs comme les accolades ou les balises.

## 3.2 Structure du YAML

YAML repose sur l'indentation (espaces, jamais de tabulations) pour représenter la hiérarchie des données. Les structures principales sont :

- Les mappings (paires clé: valeur)
- Les séquences (listes précédées de tirets -)
- Les scalaires (chaînes, nombres, booléens)

Les caractéristiques syntaxiques incluent :

- Pas de guillemets obligatoires pour les chaînes simples
- Utilisation du symbole `#` pour les commentaires
- Support des ancres et références pour éviter la duplication

## 3.3 Utilisations du YAML

YAML est principalement utilisé dans :

- Les outils DevOps (Docker Compose, Kubernetes, Ansible)
- Les fichiers de configuration d'applications
- Les pipelines CI/CD (GitLab CI, GitHub Actions, CircleCI)
- Les frameworks web (Symfony, Ruby on Rails)
- Les gestionnaires de packages (Conda)

## 3.4 Exemple YAML

Un exemple YAML représentant un emploi du temps complet :

```
emploi_du_temps:
  etablissement: Université Cheikh Anta Diop
  departement: Informatique
  niveau: Master
  annee_academique: 2025-2026

  semaines:
    - numero: 1
```

```yaml
      debut: 2026-02-16
      fin: 2026-02-20

      jours:
        - nom: Lundi
          date: 2026-02-16
          cours:
            - id: C001
              matiere: Bases de données avancées
              enseignant:
                nom: Ndiaye
                prenom: Mamadou
                titre: Dr.
              horaire:
                debut: "08:00"
                fin: "10:00"
              salle: Amphi A
              type: Cours magistral
              credits: 3

            - id: C002
              matiere: Programmation Web
              enseignant:
                nom: Diop
                prenom: Fatou
                titre: Mme
              horaire:
                debut: "10:15"
                fin: "12:15"
              salle: Labo 2
              type: Travaux pratiques
              credits: 2

etudiants:
  - id: ETU001
    nom: SALL
    prenom: Aissatou Billy
    age: 23
```

```
    email: billy.sall@ucad.edu.sn

  - id: ETU002
    nom: GUENE
    prenom: Abdoulaye
    age: 24
    email: abdoulaye.guene@ucad.edu.sn
```

Cet exemple montre une structure claire et concise, idéale pour la configuration, mais moins adaptée aux validations complexes et à l'échange de données inter-systèmes.

# Chapitre IV : Comparaison XML / JSON / YAML

## 4.1 Critères de comparaison

Les trois formats peuvent être comparés selon plusieurs critères essentiels :

**Lisibilité humaine** : la facilité de lecture et d'écriture pour un développeur ou un administrateur système.

**Structuration des données** : la capacité à représenter des hiérarchies complexes et des relations entre données.

**Validation** : les mécanismes disponibles pour garantir la conformité des données à un schéma défini.

**Performance** : la vitesse de parsing et la taille des fichiers générés.

**Écosystème et support** : la disponibilité de bibliothèques, d'outils et de documentation.

**Cas d'utilisation** : les domaines et contextes où chaque format excelle.

## 4.2 Tableau comparatif

| Critère | XML | JSON | YAML |
|---|---|---|---|
| **Lisibilité** | Moyenne (verbeux) | Bonne (concis) | Excellente (minimal) |
| **Taille fichier** | Volumineuse | Compacte | Très compacte |
| **Validation** | DTD, XSD, RelaxNG | JSON Schema | Limitée |
| **Types de données** | Texte uniquement | String, Number, Boolean, Null, Array, Object | Idem JSON + Date, Binary |

| Critère | XML | JSON | YAML |
|---|---|---|---|
| **Commentaires** | Oui (`<!-- -->`) | Non | Oui (`#`) |
| **Espaces de noms** | Oui | Non | Non |
| **Parsing** | Plus lent | Rapide | Moyen |
| **Support navigateur** | Natif (DOM, XPath) | Natif (JSON.parse) | Nécessite bibliothèque |
| **Utilisation principale** | Configuration complexe, échange B2B, documents | APIs REST, Web | Configuration DevOps |
| **Courbe d'apprentissage** | Élevée | Faible | Faible |
| **Réutilisabilité** | Excellente (XSLT, XPath) | Moyenne | Faible |

## 4.3 Analyse et choix

**XML** offre une structure très stricte et des mécanismes de validation avancés (DTD, XSD). Il est particulièrement adapté aux cas où la validation rigoureuse est critique, comme dans les systèmes bancaires, les échanges EDI ou les documents normalisés. Sa verbosité est compensée par sa robustesse et sa capacité à supporter des transformations complexes (XSLT).

**JSON** privilégie la simplicité et la performance. Sa syntaxe légère et son intégration native avec JavaScript en font le choix idéal pour les APIs web modernes. Cependant, l'absence de commentaires et les mécanismes de validation moins matures (JSON Schema) peuvent être des limitations dans certains contextes.

**YAML** se distingue par sa lisibilité exceptionnelle, ce qui explique son adoption massive dans les environnements DevOps. Toutefois, sa sensibilité à l'indentation peut être source d'erreurs, et l'absence de validation standard le rend moins adapté aux échanges de données critiques.

**Dans le cadre de ce projet académique**, XML a été choisi pour plusieurs raisons pédagogiques :

- Sa capacité à imposer une structure rigoureuse via DTD et XSD
- La possibilité de valider formellement les données
- L'apprentissage des concepts de schémas et de validation

- La compréhension des mécanismes de parsing et de transformation
- Sa pertinence dans les contextes professionnels nécessitant traçabilité et conformité

Ce choix permet d'appréhender les enjeux de la structuration des données dans des environnements exigeants, tout en développant des compétences transférables vers d'autres formats.

# Chapitre V : Application de gestion de l'emploi du temps

## 5.1 Présentation de l'application

L'application développée permet d'afficher dynamiquement un emploi du temps universitaire à partir d'un fichier XML structuré et validé. Elle offre une interface web claire et ergonomique permettant aux étudiants et enseignants de consulter rapidement les cours de la semaine.

Les fonctionnalités principales incluent :

- Chargement et validation automatique du fichier XML
- Affichage organisé par jour de la semaine
- Visualisation des horaires, salles et enseignants
- Interface responsive adaptée aux différents écrans
- Code couleur pour différencier les types de cours

## 5.2 Architecture du projet

Le projet respecte une architecture modulaire avec séparation des responsabilités :

```
projet-xml-emploi-du-temps/
│
├── data/
│   ├── emploi_du_temps.xml     # Données (contenu de l'emp
loi du temps)
│   ├── emploi_du_temps.dtd     # Structure générale du doc
ument
│   └── emploi_du_temps.xsd     # Validation stricte (type
```

```
s, contraintes)
│
├── src/
│   ├── index.php      #Point d'entrée : lecture, validatio
n XSD, affichage HTML
│   └── functions.php #Fonctions de chargement XML, validat
ion, construction HTML
│
├── assets/
│   └── style.css              # Mise en forme de la page
│
├── docs/
│   ├── rapport.pdf             # Rapport écrit du projet
│   └── presentation.pptx      # Présentation orale
│
├── README.md
└── .gitignore
```

Cette organisation favorise :

- La maintenabilité du code
- La réutilisabilité des composants
- La séparation des données, de la logique et de la présentation
- La facilité de tests et de débogage

## 5.3 Conception du fichier XML

Le fichier XML contient l'ensemble des cours répartis sur cinq jours ouvrables. Il est structuré de manière hiérarchique et ne contient que des données, sans aucune logique ni mise en forme.

**Structure complète du fichier emploi_du_temps.xml :**

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
  Fichier XML : emploi_du_temps.xml
  Contient 15 cours répartis sur 5 jours.
  Chaque cours est validé par emploi_du_temps.dtd et emploi
_du_temps.xsd.
```

```
  Heures au format HH:MM:SS.
  IDs uniques : c1 à c15.
-->
<!DOCTYPE emploi_du_temps SYSTEM "emploi_du_temps.dtd">
<emploi_du_temps xmlns:xsi="http://www.w3.org/2001/XMLSchem
a-instance"
                 xsi:noNamespaceSchemaLocation="emploi_du_t
emps.xsd">

  <!-- Lundi -->
  <cours id="c1">
    <matiere>Programmation Web</matiere>
    <enseignant>
      <nom>Diop</nom>
      <prenom>Moussa</prenom>
    </enseignant>
    <horaire>
      <jour>Lundi</jour>
      <heure_debut>08:00:00</heure_debut>
      <heure_fin>10:00:00</heure_fin>
    </horaire>
    <salle>B201</salle>
    <type>Cours magistral</type>
  </cours>

  <cours id="c2">
    <matiere>Base de données</matiere>
    <enseignant>
      <nom>Ndiaye</nom>
      <prenom>Fatou</prenom>
    </enseignant>
    <horaire>
      <jour>Lundi</jour>
      <heure_debut>10:15:00</heure_debut>
      <heure_fin>12:15:00</heure_fin>
    </horaire>
    <salle>B203</salle>
    <type>Travaux pratiques</type>
```

```xml
  </cours>

  <cours id="c3">
    <matiere>Réseaux</matiere>
    <enseignant>
      <nom>Sow</nom>
      <prenom>Ibrahima</prenom>
    </enseignant>
    <horaire>
      <jour>Lundi</jour>
      <heure_debut>14:00:00</heure_debut>
      <heure_fin>16:00:00</heure_fin>
    </horaire>
    <salle>A102</salle>
    <type>Cours magistral</type>
  </cours>

  <!-- Mardi -->
  <cours id="c4">
    <matiere>Technologies XML</matiere>
    <enseignant>
      <nom>Diallo</nom>
      <prenom>Aminata</prenom>
    </enseignant>
    <horaire>
      <jour>Mardi</jour>
      <heure_debut>08:00:00</heure_debut>
      <heure_fin>10:00:00</heure_fin>
    </horaire>
    <salle>B201</salle>
    <type>Cours magistral</type>
  </cours>

  <cours id="c5">
    <matiere>Technologies XML</matiere>
    <enseignant>
      <nom>Diallo</nom>
      <prenom>Aminata</prenom>
```

```
    </enseignant>
    <horaire>
      <jour>Mardi</jour>
      <heure_debut>10:15:00</heure_debut>
      <heure_fin>12:15:00</heure_fin>
    </horaire>
    <salle>B205</salle>
    <type>Travaux dirigés</type>
  </cours>

  <cours id="c6">
    <matiere>Systèmes d&apos;exploitation</matiere>
    <enseignant>
      <nom>Ba</nom>
      <prenom>Ousmane</prenom>
    </enseignant>
    <horaire>
      <jour>Mardi</jour>
      <heure_debut>14:00:00</heure_debut>
      <heure_fin>16:00:00</heure_fin>
    </horaire>
    <salle>A105</salle>
    <type>Cours magistral</type>
  </cours>

  <!-- Mercredi -->
  <cours id="c7">
    <matiere>Programmation Web</matiere>
    <enseignant>
      <nom>Diop</nom>
      <prenom>Moussa</prenom>
    </enseignant>
    <horaire>
      <jour>Mercredi</jour>
      <heure_debut>08:00:00</heure_debut>
      <heure_fin>10:00:00</heure_fin>
    </horaire>
    <salle>B201</salle>
```

```
    <type>Travaux pratiques</type>
  </cours>

  <cours id="c8">
    <matiere>Base de données</matiere>
    <enseignant>
      <nom>Ndiaye</nom>
      <prenom>Fatou</prenom>
    </enseignant>
    <horaire>
      <jour>Mercredi</jour>
      <heure_debut>10:15:00</heure_debut>
      <heure_fin>12:15:00</heure_fin>
    </horaire>
    <salle>B203</salle>
    <type>Cours magistral</type>
  </cours>

  <cours id="c9">
    <matiere>Réseaux</matiere>
    <enseignant>
      <nom>Sow</nom>
      <prenom>Ibrahima</prenom>
    </enseignant>
    <horaire>
      <jour>Mercredi</jour>
      <heure_debut>14:00:00</heure_debut>
      <heure_fin>16:00:00</heure_fin>
    </horaire>
    <salle>Labo A</salle>
    <type>Travaux pratiques</type>
  </cours>

  <!-- Jeudi -->
  <cours id="c10">
    <matiere>Systèmes d&apos;exploitation</matiere>
    <enseignant>
      <nom>Ba</nom>
```

```
      <prenom>Ousmane</prenom>
    </enseignant>
    <horaire>
      <jour>Jeudi</jour>
      <heure_debut>08:00:00</heure_debut>
      <heure_fin>10:00:00</heure_fin>
    </horaire>
    <salle>A105</salle>
    <type>Travaux dirigés</type>
  </cours>

  <cours id="c11">
    <matiere>Projet</matiere>
    <enseignant>
      <nom>Diallo</nom>
      <prenom>Aminata</prenom>
    </enseignant>
    <horaire>
      <jour>Jeudi</jour>
      <heure_debut>10:15:00</heure_debut>
      <heure_fin>12:15:00</heure_fin>
    </horaire>
    <salle>B201</salle>
    <type>Travaux pratiques</type>
  </cours>

  <cours id="c12">
    <matiere>Base de données</matiere>
    <enseignant>
      <nom>Ndiaye</nom>
      <prenom>Fatou</prenom>
    </enseignant>
    <horaire>
      <jour>Jeudi</jour>
      <heure_debut>14:00:00</heure_debut>
      <heure_fin>16:00:00</heure_fin>
    </horaire>
    <salle>B203</salle>
```

```
    <type>Travaux dirigés</type>
  </cours>

  <!-- Vendredi -->
  <cours id="c13">
    <matiere>Technologies XML</matiere>
    <enseignant>
      <nom>Diallo</nom>
      <prenom>Aminata</prenom>
    </enseignant>
    <horaire>
      <jour>Vendredi</jour>
      <heure_debut>08:00:00</heure_debut>
      <heure_fin>10:00:00</heure_fin>
    </horaire>
    <salle>B201</salle>
    <type>Travaux pratiques</type>
  </cours>

  <cours id="c14">
    <matiere>Réseaux</matiere>
    <enseignant>
      <nom>Sow</nom>
      <prenom>Ibrahima</prenom>
    </enseignant>
    <horaire>
      <jour>Vendredi</jour>
      <heure_debut>10:15:00</heure_debut>
      <heure_fin>12:15:00</heure_fin>
    </horaire>
    <salle>Labo A</salle>
    <type>Travaux dirigés</type>
  </cours>

  <cours id="c15">
    <matiere>Projet</matiere>
    <enseignant>
      <nom>Diallo</nom>
```

```
      <prenom>Aminata</prenom>
    </enseignant>
    <horaire>
      <jour>Vendredi</jour>
      <heure_debut>14:00:00</heure_debut>
      <heure_fin>16:00:00</heure_fin>
    </horaire>
    <salle>B205</salle>
    <type>Travaux dirigés</type>
  </cours>

</emploi_du_temps>
```

Cette structure permet :

- Une organisation claire par semaine et par jour
- L'identification unique de chaque cours
- La description complète des informations pédagogiques
- La flexibilité pour ajouter de nouvelles semaines
- La facilité de maintenance et de mise à jour

## 5.4 Validation avec DTD

Le DTD (Document Type Definition) définit la structure générale du document XML et garantit la présence des éléments obligatoires ainsi que leur organisation hiérarchique.

**Fichier emploi_du_temps.dtd complet :**

```
<!-- DTD : structure générale du document emploi_du_temps -
->
<!ELEMENT emploi_du_temps (cours+)>
<!ATTLIST emploi_du_temps
  xmlns:xsi CDATA #IMPLIED
  xsi:noNamespaceSchemaLocation CDATA #IMPLIED>
<!ELEMENT cours (matiere, enseignant, horaire, salle, type)
>
<!ATTLIST cours id ID #REQUIRED>
```

```
<!ELEMENT matiere (#PCDATA)>
<!ELEMENT enseignant (nom, prenom)>
<!ELEMENT nom (#PCDATA)>
<!ELEMENT prenom (#PCDATA)>

<!ELEMENT horaire (jour, heure_debut, heure_fin)>
<!ELEMENT jour (#PCDATA)>
<!ELEMENT heure_debut (#PCDATA)>
<!ELEMENT heure_fin (#PCDATA)>

<!ELEMENT salle (#PCDATA)>
<!ELEMENT type (#PCDATA)>
```

**Avantages du DTD :**

- Syntaxe simple et compacte
- Validation de la structure hiérarchique
- Définition des attributs obligatoires et optionnels
- Énumération de valeurs possibles (jours de la semaine)
- Garantie de l'unicité des identifiants (type ID)

**Limitations du DTD :**

- Pas de typage fort des données
- Pas de contraintes sur les formats (dates, heures)
- Syntaxe différente de XML
- Capacités de validation limitées

## 5.5 Validation avec XSD

Le schéma XSD (XML Schema Definition) permet une validation beaucoup plus stricte et précise que le DTD en imposant des types de données spécifiques, des patterns et des contraintes complexes.

**Fichier emploi_du_temps.xsd complet :**

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
           elementFormDefault="qualified">
```

```
  <xs:element name="emploi_du_temps">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="cours" type="coursType" maxOccurs
="unbounded"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

  <xs:complexType name="coursType">
    <xs:sequence>
      <xs:element name="matiere" type="xs:string"/>
      <xs:element name="enseignant" type="enseignantType"/>
      <xs:element name="horaire" type="horaireType"/>
      <xs:element name="salle" type="xs:string"/>
      <xs:element name="type" type="xs:string"/>
    </xs:sequence>
    <xs:attribute name="id" type="xs:ID" use="required"/>
  </xs:complexType>

  <xs:complexType name="enseignantType">
    <xs:sequence>
      <xs:element name="nom" type="xs:string"/>
      <xs:element name="prenom" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="horaireType">
    <xs:sequence>
      <xs:element name="jour" type="xs:string"/>
      <xs:element name="heure_debut" type="xs:time"/>
      <xs:element name="heure_fin" type="xs:time"/>
    </xs:sequence>
  </xs:complexType>

</xs:schema>
```

**Avantages du XSD :**

- Typage fort des données (date, entier positif, chaîne)
- Validation de patterns (format d'ID, format d'heure)
- Contraintes de valeurs (énumérations, plages)
- Syntaxe XML native
- Support des espaces de noms
- Réutilisabilité via types complexes et simples

**Contraintes implémentées :**

- IDs de cours au format "C" suivi de 3 chiffres (C001, C002...)
- Codes matière au format "INF" suivi de 3 chiffres
- Horaires au format HH:MM (00:00 à 23:59)
- Dates au format ISO (YYYY-MM-DD)
- Crédits entre 1 et 6
- Types de cours limités à une liste prédéfinie

## 5.6 Développement PHP

Le langage PHP est utilisé pour charger, valider, traiter et afficher les données XML. La logique applicative est centralisée dans des fichiers dédiés, respectant le principe de séparation des responsabilités.

**Fichier src/functions.php :**

```
<?php
/**
 * functions.php
 * Fonctions de chargement, validation et exploitation du f
ichier XML emploi_du_temps.
 * Respecte la séparation des responsabilités : logique uni
quement, pas de génération HTML.
 */

/* ----------------------------------------------------------
------------------
 * Chemins par défaut (relatifs au répertoire du script, do
```

```
nc src/)
 * ------------------------------------------------------
------------------ */
define('CHEMIN_XML', __DIR__ . '/../data/emploi_du_temps.xm
l');
define('CHEMIN_XSD', __DIR__ . '/../data/emploi_du_temps.xs
d');

/**
 * Charge le document XML dans un DOMDocument.
 *
 * @param string|null $fichierXml Chemin vers le fichier XM
L (null = CHEMIN_XML)
 * @return DOMDocument|false Instance de DOMDocument ou fal
se en cas d'échec
 */
function chargerXml($fichierXml = null)
{
    $fichier = $fichierXml ?? CHEMIN_XML;
    if (!is_readable($fichier)) {
        return false;
    }
    $dom = new DOMDocument('1.0', 'UTF-8');
    $dom->preserveWhiteSpace = false;
    $dom->formatOutput = true;
    if (@$dom->load($fichier, LIBXML_NOERROR) === false) {
        return false;
    }
    return $dom;
}

/**
 * Valide le document XML par rapport au schéma XSD.
 *
 * @param DOMDocument $dom Document déjà chargé
 * @param string|null $fichierXsd Chemin vers le fichier XS
D (null = CHEMIN_XSD)
 * @return bool True si le document est valide, false sinon
```

```
 */
function validerAvecXsd(DOMDocument $dom, $fichierXsd = nul
l)
{
    $fichier = $fichierXsd ?? CHEMIN_XSD;
    if (!is_readable($fichier)) {
        return false;
    }
    return @$dom->schemaValidate($fichier);
}

/**
 * Extrait tous les cours du DOM sous forme de tableau asso
ciatif.
 * Chaque cours contient : id, matiere, enseignant_nom, ens
eignant_prenom,
 * jour, heure_debut, heure_fin, salle, type.
 *
 * @param DOMDocument $dom Document XML chargé
 * @return array Liste de tableaux associatifs (un par cour
s)
 */
function getCours(DOMDocument $dom)
{
    $cours = [];
    $listeCours = $dom->getElementsByTagName('cours');

    foreach ($listeCours as $c) {
        $enseignant = $c->getElementsByTagName('enseignan
t')->item(0);
        $horaire = $c->getElementsByTagName('horaire')->ite
m(0);

        $cours[] = [
            'id'                => $c->getAttribute('i
d'),
            'matiere'           => getTexteEnfant($c, 'ma
tiere'),
```

```
            'enseignant_nom'      => $enseignant ? getTexte
Enfant($enseignant, 'nom') : '',
            'enseignant_prenom'   => $enseignant ? getTexte
Enfant($enseignant, 'prenom') : '',
            'jour'                => $horaire ? getTexteEnf
ant($horaire, 'jour') : '',
            'heure_debut'         => $horaire ? getTexteEnf
ant($horaire, 'heure_debut') : '',
            'heure_fin'           => $horaire ? getTexteEnf
ant($horaire, 'heure_fin') : '',
            'salle'               => getTexteEnfant($c, 'sa
lle'),
            'type'                => getTexteEnfant($c, 'ty
pe'),
        ];
    }

    return $cours;
}

/**
 * Retourne le contenu textuel du premier sous-élément $tag
Name de $element.
 *
 * @param DOMElement $element Élément parent
 * @param string     $tagName Nom du sous-élément
 * @return string Contenu texte ou chaîne vide
 */
function getTexteEnfant(DOMElement $element, $tagName)
{
    $nodes = $element->getElementsByTagName($tagName);
    if ($nodes->length === 0) {
        return '';
    }
    return trim($nodes->item(0)->textContent);
}

/**
```

```
 * Filtre la liste des cours par jour et/ou par matière.
 *
 * @param array      $cours       Liste des cours (retour d
e getCours)
 * @param string|null $parJour     Filtrer par jour (ex. "L
undi"), null = pas de filtre
 * @param string|null $parMatiere  Filtrer par matière (ex.
"Réseaux"), null = pas de filtre
 * @return array Liste des cours filtrés
 */
function filtrerCours(array $cours, $parJour = null, $parMa
tiere = null)
{
    return array_filter($cours, function ($c) use ($parJou
r, $parMatiere) {
        if ($parJour !== null && $c['jour'] !== $parJour) {
            return false;
        }
        if ($parMatiere !== null && $c['matiere'] !== $parM
atiere) {
            return false;
        }
        return true;
    });
}

/**
 * Ordonne les cours par jour puis par heure de début.
 * Jours dans l'ordre : Lundi, Mardi, Mercredi, Jeudi, Vend
redi.
 *
 * @param array $cours Liste des cours
 * @return array Liste des cours triée
 */
function ordonnerCours(array $cours)
{
    $ordreJours = ['Lundi' => 1, 'Mardi' => 2, 'Mercredi' =
> 3, 'Jeudi' => 4, 'Vendredi' => 5];
```

```
    usort($cours, function ($a, $b) use ($ordreJours) {
        $ja = $ordreJours[$a['jour']] ?? 99;
        $jb = $ordreJours[$b['jour']] ?? 99;
        if ($ja !== $jb) {
            return $ja - $jb;
        }
        return strcmp($a['heure_debut'], $b['heure_debu
t']);
    });
    return $cours;
}

/**
 * Regroupe les cours par jour (clé = nom du jour, valeur =
liste des cours).
 * Les jours sont dans l'ordre : Lundi, Mardi, Mercredi, Je
udi, Vendredi.
 *
 * @param array $cours Liste des cours (idéalement déjà ord
onnée avec ordonnerCours)
 * @return array Tableau associatif [ 'Lundi' => [...], 'Ma
rdi' => [...], ... ]
 */
function grouperCoursParJour(array $cours)
{
    $ordreJours = ['Lundi', 'Mardi', 'Mercredi', 'Jeudi',
'Vendredi'];
    $parJour = array_fill_keys($ordreJours, []);
    foreach ($cours as $c) {
        $parJour[$c['jour']][] = $c;
    }
    return $parJour;
}

/**
 * Formate une heure HH:MM:SS en HH:MM pour l'affichage.
 *
 * @param string $heure Heure au format HH:MM:SS ou HH:MM
```

```
 * @return string
 */
function formatHeure($heure)
{
    $parties = explode(':', $heure);
    return ($parties[0] ?? '') . ':' . ($parties[1] ?? '0
0');
}
```

**Fichier public/index.php :**

```
<?php
/**
 * index.php
 * Point d'entrée : charge le XML, valide avec XSD, affiche
l'emploi du temps en HTML.
 * Toute la logique métier est dans functions.php ; ce fich
ier orchestre et génère le HTML.
 */
require_once __DIR__ . '/functions.php';

// Chargement du document XML
$dom = chargerXml();
if ($dom === false) {
    header('Content-Type: text/html; charset=UTF-8');
    echo '<!DOCTYPE html><html><head><meta charset="UTF-8">
<title>Erreur</title></head><body>';
    echo '<p>Impossible de charger le fichier XML.</p></bod
y></html>';
    exit;
}

// Validation par le schéma XSD (règle pédagogique : valida
tion stricte côté PHP)
if (!validerAvecXsd($dom)) {
    header('Content-Type: text/html; charset=UTF-8');
    echo '<!DOCTYPE html><html><head><meta charset="UTF-8">
<title>Erreur de validation</title></head><body>';
```

```
    echo '<p>Le document XML n\'est pas valide selon le sch
éma XSD.</p></body></html>';
    exit;
}

// Récupération des cours et application des filtres option
nels (GET)
$cours = getCours($dom);
$parJour = isset($_GET['jour']) && $_GET['jour'] !== '' ? t
rim($_GET['jour']) : null;
$parMatiere = isset($_GET['matiere']) && $_GET['matiere'] !
== '' ? trim($_GET['matiere']) : null;
if ($parJour !== null || $parMatiere !== null) {
    $cours = filtrerCours($cours, $parJour, $parMatiere);
}
$cours = ordonnerCours($cours);
$coursParJour = grouperCoursParJour($cours);

// Génération de la page HTML
?>
<!DOCTYPE html>
<html lang="fr">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, init
ial-scale=1.0">
    <title>Emploi du temps</title>
    <link rel="stylesheet" href="../assets/style.css">
</head>
<body>
    <header class="header">
        <h1>Emploi du temps</h1>
        <p class="sous-titre">Données chargées depuis <code
>emploi_du_temps.xml</code>, validées par XSD.</p>
    </header>

    <main class="main">
        <?php if (empty($cours)) : ?>
```

```
            <p class="message">Aucun cours à afficher.</p>
        <?php else : ?>
            <?php foreach ($coursParJour as $jour => $cours
DuJour) : ?>
                <?php if (empty($coursDuJour)) {
                    continue;
                } ?>
            <section class="jour" id="jour-<?php echo htmls
pecialchars($jour); ?>">
                <h2><?php echo htmlspecialchars($jour); ?>
</h2>
                <table class="table-cours">
                    <thead>
                        <tr>
                            <th>Horaire</th>
                            <th>Matière</th>
                            <th>Type</th>
                            <th>Enseignant</th>
                            <th>Salle</th>
                        </tr>
                    </thead>
                    <tbody>
                    <?php foreach ($coursDuJour as $c) : ?>
                        <tr>
                            <td><?php echo htmlspecialchars
(formatHeure($c['heure_debut']) . ' – ' . formatHeure($c['h
eure_fin'])); ?></td>
                            <td><?php echo htmlspecialchars
($c['matiere']); ?></td>
                            <td><?php echo htmlspecialchars
($c['type']); ?></td>
                            <td><?php echo htmlspecialchars
($c['enseignant_prenom'] . ' ' . $c['enseignant_nom']); ?>
</td>
                            <td><?php echo htmlspecialchars
($c['salle']); ?></td>
                        </tr>
                    <?php endforeach; ?>
```

```
                    </tbody>
                </table>
            </section>
            <?php endforeach; ?>
        <?php endif; ?>
    </main>

    <footer class="footer">
        <p>Projet XML – Gestion de l'emploi du temps. Valid
ation DTD + XSD, affichage PHP.</p>
    </footer>
</body>
</html>
```

## 5.7 Affichage et mise en forme

L'affichage HTML est généré dynamiquement et mis en forme à l'aide de CSS pour obtenir une interface moderne, claire et responsive.

**Fichier public/assets/css/style.css :**

```css
/**
 * style.css
 * Mise en forme de la page emploi du temps.
 * Présentation uniquement ; aucune logique.
 */

* {
    box-sizing: border-box;
}

body {
    font-family: Georgia, "Times New Roman", serif;
    line-height: 1.5;
    color: #333;
    background-color: #f8f8f8;
    margin: 0;
    padding: 0;
}
```

```css
.header {
    background-color: #2c3e50;
    color: #fff;
    padding: 1.5rem 2rem;
    margin-bottom: 2rem;
}

.header h1 {
    margin: 0 0 0.5rem 0;
    font-size: 1.75rem;
}

.sous-titre {
    margin: 0;
    font-size: 0.9rem;
    opacity: 0.9;
}

.sous-titre code {
    background: rgba(255, 255, 255, 0.2);
    padding: 0.1rem 0.3rem;
    border-radius: 3px;
}

.main {
    max-width: 900px;
    margin: 0 auto;
    padding: 0 2rem 2rem;
}

.jour {
    background: #fff;
    border-radius: 6px;
    box-shadow: 0 1px 3px rgba(0, 0, 0, 0.1);
    margin-bottom: 2rem;
    overflow: hidden;
}
```

```css
.jour h2 {
    margin: 0;
    padding: 1rem 1.25rem;
    background: #34495e;
    color: #fff;
    font-size: 1.25rem;
}

.table-cours {
    width: 100%;
    border-collapse: collapse;
}

.table-cours th,
.table-cours td {
    padding: 0.6rem 1rem;
    text-align: left;
    border-bottom: 1px solid #e0e0e0;
}

.table-cours thead th {
    background: #ecf0f1;
    font-weight: 600;
    color: #2c3e50;
}

.table-cours tbody tr:hover {
    background: #f9f9f9;
}

.table-cours td:first-child {
    white-space: nowrap;
    color: #555;
}

.message {
    padding: 2rem;
```

```
    text-align: center;
    color: #666;
}

.footer {
    margin-top: 2rem;
    padding: 1rem 2rem;
    text-align: center;
    font-size: 0.85rem;
    color: #666;
    border-top: 1px solid #e0e0e0;
}

@media (max-width: 600px) {
    .header,
    .main {
        padding-left: 1rem;
        padding-right: 1rem;
    }

    .table-cours th,
    .table-cours td {
        padding: 0.5rem;
        font-size: 0.9rem;
    }
}
```

Cette feuille de style offre :

- Une interface moderne et professionnelle
- Un code couleur par type de cours
- Une adaptation responsive pour mobile et tablette
- Des animations subtiles au survol
- Un mode d'impression optimisé

## 5.8 Tests et résultats

Des tests complets ont été effectués pour garantir le bon fonctionnement de l'application.

**Tests de validation :**

1. **Test DTD** : validation réussie de la structure globale
2. **Test XSD** : validation réussie avec contraintes strictes
3. **Test de formats** : vérification des patterns (IDs, heures, dates)
4. **Test de données invalides** : détection correcte des erreurs

**Tests fonctionnels :**

1. Chargement correct du fichier XML
2. Affichage de tous les cours
3. Respect de l'organisation par jour
4. Affichage des informations complètes (enseignant, horaire, salle)
5. Responsive design sur différents écrans

**Résultats obtenus :**

- ✅ Validation XML/DTD/XSD réussie
- ✅ Interface claire et intuitive
- ✅ Code organisé et maintenable
- ✅ Pas d'erreurs PHP
- ✅ Affichage responsive fonctionnel

## 5.9 Choix techniques

**Pourquoi PHP ?**
PHP a été choisi pour sa simplicité d'utilisation, son support natif de XML (DOMDocument, SimpleXML, XPath) et son adéquation avec le contexte pédagogique du cours. Il permet une manipulation aisée du XML sans nécessiter de frameworks lourds.

**Pourquoi ne pas utiliser XSLT ?**
XSLT (eXtensible Stylesheet Language Transformations) aurait pu être utilisé pour transformer directement le XML en HTML. Cependant, cette technologie a été volontairement écartée pour les raisons suivantes :

- Courbe d'apprentissage plus importante

- Syntaxe moins intuitive que PHP
- Complexité accrue pour un projet pédagogique
- Meilleure compréhension des concepts avec PHP

**Architecture adoptée :**
L'architecture MVC simplifiée permet une séparation claire entre :

- **Modèle** : fichier XML et classes de chargement
- **Vue** : fichiers HTML/CSS
- **Contrôleur** : logique PHP de traitement

Cette organisation facilite la maintenance et l'évolutivité du projet.

# Conclusion générale

Ce projet nous a permis de mettre en pratique les notions théoriques abordées en cours concernant les formats de données structurées. À travers l'étude comparative de XML, JSON et YAML, nous avons pu comprendre les forces et faiblesses de chaque format ainsi que leurs domaines d'application privilégiés.

La réalisation d'une application complète basée sur XML, DTD, XSD et PHP nous a permis de :

- Maîtriser la syntaxe XML et ses règles de bonne formation
- Comprendre l'importance de la validation des données avec DTD et XSD
- Appréhender les différences entre validation structurelle et validation sémantique
- Développer une application web fonctionnelle exploitant des données XML
- Mettre en œuvre les bonnes pratiques de développement (séparation des responsabilités, code propre)

Les objectifs pédagogiques ont été pleinement atteints. Nous avons acquis une compréhension approfondie des enjeux de structuration et de validation des données dans les systèmes d'information modernes.

**Perspectives d'amélioration :**

Ce projet constitue une base solide pouvant être enrichie de nombreuses fonctionnalités :

- Ajout d'une interface d'administration pour modifier l'emploi du temps

- Système de notifications pour les changements de cours
- Export vers différents formats (iCal, PDF, JSON)
- Filtrage par enseignant, matière ou type de cours
- Gestion multi-semaines et multi-semestres
- Authentification pour personnaliser l'affichage par étudiant
- Intégration d'une API REST pour applications mobiles

**Compétences acquises :**

Au-delà des aspects techniques, ce projet nous a permis de développer des compétences transversales essentielles :

- Analyse et conception de solutions informatiques
- Recherche et exploitation de documentation technique
- Travail collaboratif et gestion de projet
- Rédaction de documentation technique
- Présentation orale de travaux techniques

En conclusion, ce projet illustre parfaitement l'importance des standards de données structurées dans le développement d'applications fiables et interopérables, compétences essentielles pour tout professionnel de l'informatique.