

# Architecture Application PySide6 - Transformation XML vers Site Web

## Structure des fichiers

```
xml_to_web_app/
├── main.py                # Point d'entrée principal
├── ui/
│   ├── __init__.py
│   ├── main_window.py    # Fenêtre principale
│   ├── project_editor.py  # Fenêtre d'édition de projet
│   ├── wysiwyg_editor.py  # Composant éditeur WYSIWYG
│   └── dialogs.py        # Dialogues (upload, export, etc.)
├── core/
│   ├── __init__.py
│   ├── xml_manager.py    # Gestion des fichiers XML
│   ├── project_manager.py # Gestion des projets
│   ├── xslt_processor.py  # Transformation XSLT avec saxonche
│   └── zip_generator.py   # Génération du ZIP final
├── models/
│   ├── __init__.py
│   ├── project.py        # Modèle de données Projet
│   └── xml_data.py       # Modèle de données XML
├── resources/
│   ├── templates/
│   │   └── default.xslt  # Template XSLT par défaut
│   ├── editor/
│   │   ├── tinymce/      # Éditeur TinyMCE
│   │   └── editor.html   # Page HTML pour l'éditeur
│   └── icons/            # Icônes de l'application
└── utils/
    ├── __init__.py
    ├── file_utils.py     # Utilitaires fichiers
    └── xml_utils.py      # Utilitaires XML
```

## Composants principaux

### 1. Fenêtre principale (MainWindow)

python

```
class MainWindow(QMainWindow):  
    """  
    Fenêtre principale de l'application  
    - Menu et toolbar  
    - Zone de gestion des projets (liste/arbre)  
    - Zone de prévisualisation  
    - Barre de statut  
    """  
  
    # Sections principales :  
    # - Chargement XML de base de données  
    # - Liste des projets existants  
    # - Boutons d'action (Ajouter, Modifier, Supprimer)  
    # - Aperçu du projet sélectionné  
    # - Génération et export final
```

## 2. Gestionnaire de projets (ProjectManager)

python

```
class ProjectManager:  
    """  
    Gère la logique métier des projets  
    - CRUD des projets  
    - Validation des données  
    - Sauvegarde/chargement XML projets  
    """  
  
    # Fonctionnalités :  
    # - create_project(name, description_html)  
    # - update_project(project_id, data)  
    # - delete_project(project_id)  
    # - get_projects_xml()  
    # - load_from_xml(xml_content)
```

## 3. Éditeur WYSIWYG (WysiwygEditor)

python

```
class WysiygEditor(QWebEngineView):
    """
    Composant d'édition HTML intégré
    - Utilise TinyMCE ou CKEditor
    - Communication bidirectionnelle avec Python
    - Sauvegarde automatique
    """

    # Fonctionnalités :
    # - load_content(html_content)
    # - get_content() -> html_content
    # - setup_editor_communication()
    # - handle_content_changed()
```

## 4. Processeur XSLT (XsltProcessor)

python

```
class XsltProcessor:
    """
    Gère la transformation XSLT avec saxonche
    - Charge les données XML et projets XML
    - Applique la transformation XSLT
    - Génère les fichiers de sortie
    """

    # Fonctionnalités :
    # - load_data_xml(xml_path)
    # - load_projects_xml(projects_data)
    # - load_xslt_template(xslt_path)
    # - transform() -> output_files
    # - validate_xml(xml_content)
```

## Flux de données

### 1. Initialisation

Application Start

- └─ Chargement interface principale
- └─ Initialisation ProjectManager (XML projets vide)
- └─ Préparation éditeur WYSIWYG

### 2. Import des données

- User Upload XML
  - └ Validation XML (xml\_utils)
  - └ Parsing et extraction projets existants
  - └ Mise à jour interface (liste projets)
  - └ Stockage temporaire des données

### 3. Édition de projet

- User Select/Create Project
  - └ Ouverture ProjectEditor window
  - └ Chargement contenu dans WysiwygEditor
  - └ Édition WYSIWYG
  - └ Sauvegarde dans ProjectManager
  - └ Mise à jour XML projets

### 4. Génération finale

- User Generate Site
  - └ Validation données complètes
  - └ XsltProcessor.transform()
  - └ ZipGenerator.create\_archive()
  - └ Dialogue de sauvegarde

## Interface utilisateur

### Layout principal

|   |                                       |
|---|---------------------------------------|
| Menu: Fichier   Édition   Transformation   Aide             |                                       |
| Toolbar: [Upload] [Nouveau] [Modifier] [Supprimer] [Export] |                                       |
| Liste Projets   | Aperçu Projet                         |
| <div><input type="checkbox"/> Projet A</div>                | <div>Titre: Projet A</div>            |
| <div><input type="checkbox"/> Projet B</div>                | <div></div>                           |
| <div><input type="checkbox"/> Projet C</div>                | <div>Description HTML rendue...</div> |
| <div>+ Nouveau</div>  | <div></div>                           |
| Status: Prêt   3 projets chargés   XML: data.xml            |                                       |

## Modèles de données

### Project

python

@dataclass

```
class Project:
    id: str
    name: str
    description_html: str
    created_at: datetime
    updated_at: datetime

    def to_xml_element(self) -> str
    def from_xml_element(element) -> Project
```

## XMLData

python

```
class XMLData:
    """Encapsule les données XML de la base"""
    raw_xml: str
    parsed_data: ElementTree
    projects_referenced: List[str]

    def get_project_ids(self) -> List[str]
    def validate(self) -> bool
```

## Fonctionnalités clés

### Gestion des erreurs

- Validation XML à l'import
- Gestion des projets manquants/orphelins
- Vérification XSLT avant transformation
- Messages d'erreur utilisateur explicites

### Sauvegarde/Restauration

- Sauvegarde automatique du XML projets
- Récupération en cas de fermeture inattendue
- Export/import configuration projets

## Interface utilisateur

- Drag & drop pour upload XML
- Raccourcis clavier standards
- Indicateurs de progression pour transformations
- Prévisualisation temps réel des modifications

## Technologies utilisées

- **PySide6** : Interface graphique
- **QWebEngineView** : Intégration éditeur web
- **saxonche** : Transformation XSLT
- **lxml** : Manipulation XML Python
- **zipfile** : Génération archives
- **TinyMCE/CKEditor** : Éditeur WYSIWYG

## Déploiement

### Packaging avec PyInstaller

```
bash

pyinstaller --onefile --windowed \
    --add-data "resources;resources" \
    --add-data "templates;templates" \
    main.py
```

### Structure de l'exécutable

- Toutes les ressources embarquées
- Éditeur WYSIWYG inclus
- Templates XSLT par défaut
- Pas de dépendances externes