📘 Cycling Tour Operator — XML / XSD / XSLT / JSON / Python Data Pipeline

---

This model describes a complete information system for a cycling tour operator, covering the management of clients, guides, tour packages, trip groups, bikes, reservations, maintenance, and cycling paths.
It covers data modeling, schema design, XML dataset creation, visualization scenarios using XSLT, XML-to-JSON transformations, and Python processing (XSD validation, XSL application, and DOM-based extraction).

1. DATA MODEL OVERVIEW

🔹 Clients and Guides and Trip Groups

- A **Client** has personal information (name, email, phone, country), as well as an **experience level** (beginner, occasional, regular, or expert).
- A **Guide** is a certified professional who leads cycling groups and has contact details stored in the system.
- A **TripGroup** represents a a group of clients where one client books a specific tour package and is led by a guide.
    - It references the TourPackage it corresponds to.
    - It is led by a **head guide**.

- It defines the maximum number of participants and tracks the clients currently booked into the group.

🔹 Tour Packages

- A **TourPackage** represents a **tour offer**, made up of multiple daily **Stages**.
  - It includes a title, description, total duration in days, difficulty level, currency, and whether luggage transportation is provided.
  - Each tour package is always composed of **at least one stage**.

🔹 Stages, Destinations, Activities, and Cycling Paths

Each **Stage** describes a specific day of a tour:

- It is linked to a tour, assigned to a day number, and includes a title, description, and daily distance.
- Every stage is associated with:
  - a set of **Activities** (cultural visits, events, experiences)
  - one or more **CyclingPaths** used that day
  - exactly two **Destinations**: the starting point and the ending point.

Destinations include their name, country, region, type (city, historical, mountain, etc.), and GPS coordinates.

CyclingPaths specify the difficulty, distance, elevation gain, and surface type. Each path always connects one starting destination to one ending destination.

🔹 Bike Fleet and Maintenance Management

- A **Bike** has a model, frame size, bike type (trekking or electric), availability, and rental price per day.
- A **RentalBikeListing** groups one or several bikes available for rental.
- Each bike has a **maintenance history**, stored in MaintenanceLog entries containing date, cost, and description.

🔹 Bookings

A **Booking** links together:

- a Client,
- a TripGroup,
- a RentalBikeListing,
- the booking date,
- the booking status (pending, confirmed, cancelled, completed),
- and the total price.

🔹 Key Relationships

- A **TourPackage** is composed of multiple **Stages** (composition).
- A **TripGroup** is led by one **Guide**.
- A **Client** can make several **Bookings** and participate in multiple trip groups.
- A **Stage** offers **Activities** and uses **CyclingPaths**.
- A **Bike** has multiple maintenance logs.
- A **CyclingPath** always connects a start and an end **Destination**.

---

2. Modules

the model is modularized as follows:

- common.xsd - contains common simple types and enumerations used across the model (ExperienceLevel, DifficultyLevel, BikeType, BookingStatus , DestinationType, EmailType, PriceType, LatitudeType, LongitudeType).
- clients_and_guides.xsd - defines the Client, Guide, and TripGroup entities along with their relationships of complex type.
  (client , guide , tripGroup).
- tour_packages.xsd - defines the TourPackage, Stage, Activity, CyclingPath, and Destination entities along with their relationships of complex type.
  (tourPackage , stage , activity , cyclingPath , destination).
- bike_fleet_and_maintenance.xsd - defines the Bike, RentalBikeListing, and MaintenanceLog entities along with their relationships of complex type.
  (bike , rentalBikeListing , maintenanceLog).
- bookings.xsd - defines the Booking entity along with its relationships of complex type.
  (booking).
- main.xsd - imports all the modularized schemas and serves as the root schema for the entire model.
  (imports all other .xsd files).

Imports all modules and defines the root element:

main:cyclingTourDatabase

3. XML DATABASE

The file example_1.xml contains a populated version of the model:

12 clients , 3 guides , 3 trip groups , 5 tour packages , 12 stages
destinations, activities, cycling paths
full bike fleet + maintenance logs
bookings

The XML validates fully against main.xsd.

4. Scenarios and xsl transformation

- Testeur de XSLT : https://xslttest.appspot.com/

Located in xsl_files/scenario-visualize/.

✓ Activities per Stage

Lists all activities grouped by stage.

✓ Clients by Experience Level

Groups clients by BEGINNER / OCCASIONAL / REGULAR / EXPERT.

✓ Available Bikes

Displays available rental bikes with model, frame size, type, price/day.

✓ Destinations by Type

Groups destinations by CITY / HISTORICAL / COASTAL / MOUNTAINOUS, etc.

✓ Bookings by Status

Lists confirmed, cancelled, completed bookings.

✓ Tours and Their Stages

Displays each tour package with its day-by-day stages.

5. XSLT Structure Modification Scenarios

Two XSL files perform structural reorganization of the XML data, including grouping, reordering, and restructuring.

- Additional scenario 1:

- Additional scenario 2:

6. XML to JSON Transformations

Two XSLT transformations produce JSON-like structures representing selected sections of the data. These JSON outputs are then processed in Python (summary stats, grouping, validation-style checks).

- scenario json 1 :
- scenario json 2 :

7. Python Programs

✓ xml_pipeline.py — Generic XML → XSD → XSL Pipeline

This script does:

- loads an XML file
- validates it against an XSD schema
- applies an XSL stylesheet

- generates an HTML output inside the output/ directory

- Usage :
  ./xml_pipeline.py

Example:
./xml_pipeline.py xml_database/example_1.xml xsd_files/main.xsd "xsl_files/scenario-visualize/S1 - activites_par_stage.xsl"

✓ scenario2_dom.py — DOM-Based Scenario Implementation

Implements Scenario 2 using Python's XML DOM API (xml.dom.minidom):

- parses the XML
- extracts clients
- groups them by experience level
- writes the output to:

output/scenario2_python_dom.html

This file satisfies the requirement for a second implementation using DOM or SAX.

✓ run_full_pipeline.py — Complete Pipeline Runner

Located at the root of the repository.

This script:

- Validates the XML and applies all XSL stylesheets automatically
- Executes the Python DOM scenario
- Generates all HTML files in the output/ folder
- Run the full pipeline :
  ./run_full_pipeline.py

Output examples:

```
output/
  ├── S1 - activites_par_stage.html
  ├── S2-clients_par_niveau_experience.html
  ├── S3-bikes_disponibles.html
  ├── S4-destinations_par_type.html
  ├── S5-reservations_par_statut.html
  ├── S6-tours_et_stages.html
  ├── s1bis_.html
  └── scenario2_python_dom.html
```

8. Summary

This project delivers a complete and functional data processing pipeline for a cycling tour operator, integrating:

a rich XML dataset

a fully modularized XSD schema

multiple XSLT visualization scenarios

XML → JSON conversions

a generic Python XML/XSD/XSL pipeline

a DOM-based Python scenario

a full automatic execution pipeline

Everything is designed to be modular, extensible, and strictly valid according to the schema.