PRACTICAS – MICROSERVICIOS EN C++

- **1: JSON / XML**: Conversor de ficheros CSV a JSON / XML. Utilizar el fichero
  - Para partir en columnas el fichero CSV:

```
void split(const std::string &s, char delim, std::vector<std::string> &elems) {
  std::stringstream ss;
  ss.str(s);
  std::string item;
  while (std::getline(ss, item, delim)) {
    elems.push_back(item);
  }
}
```

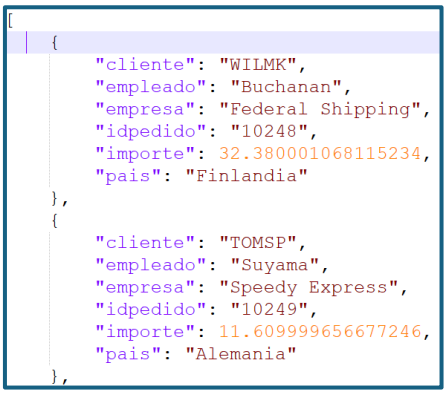
  - Definir una clase con atributos que se correspondan a las columnas del CSV.

idpedido	cliente	empresa	empleado	importe	pais
10248	WILMK	Federal Shipping	Buchanan	32.38	Finlandia
10249	TOMSP	Speedy Express	Suyama	11.61	Alemania
10250	HANAR	United Package	Peacock	65.83	Brasil
10251	VICTE	Speedy Express	Leverling	41.34	Francia
10252	SUPRD	United Package	Peacock	51.3	Belgica
10253	HANAR	United Package	Leverling	58.17	Brasil
10254	CHOPS	United Package	Buchanan	22.98	Suiza
10255	RICSU	Federal Shipping	Dodsworth	148.33	Suiza
10256	WELLI	United Package	Leverling	13.97	Brasil
10257	HILAA	Federal Shipping	Peacock	81.91	Venezuela

  - Generar un conversor de pedidos en CSV a XML, grabar el fichero

```
▼<pedidos>
  ▼<pedido>
     <idpedido>10248</idpedido>
     <cliente>WILMK</cliente>
     <empresa>Federal Shipping</empresa>
     <empleado>Buchanan</empleado>
     <empleado>32.380001</empleado>
     <pais>Finlandia</pais>
   </pedido>
  ▼<pedido>
     <idpedido>10249</idpedido>
     <cliente>TOMSP</cliente>
     <empresa>Speedy Express</empresa>
     <empleado>Suyama</empleado>
     <empleado>11.610000</empleado>
     <pais>Alemania</pais>
   </pedido>
```

- Generar un conversor de pedidos en CSV a JSON, grabar el fichero



- 2: Http: implementar un servidor Http con boost.beast
- 3: Http: implementar un Api Rest para crear, listar y eliminar (solo estas 3 operaciones). ***Activar C++ 17, en propiedades del proyecto → idioma***
  - Los datos no se almacenan en la BD, los mantendremos en una colección en memoria. Por ejemplo:
    - std::unordered_map<int, std::string>
  - Las rutas:
    - GET /ítems → listar los elementos en json
    - POST /ítems → añade uno nuevo, lo recibe en texto plano o en json
    - DELETE /ítems/{id} → elimina por ID si existe
    - La colección para almacenar:
    - std::unordered_map<int, std::string> ítems;
    - int next_id = 1;
    - #include <boost/beast/core.hpp> // Core de Beast: buffers, streams
    - #include <boost/beast/http.hpp> // Manejo de HTTP: request, response
    - #include <boost/beast/version.hpp> // Versión de Beast
    - #include <boost/asio/ip/tcp.hpp> // TCP con Boost.Asio
    - #include <boost/asio/io_context.hpp> // Contexto de I/O

- 4: Base de datos: SQLite3 implementar un patrón DAO con las operaciones CRUD y un método select para recuperar una colección de empleados: id, nombre y cargo. En el método créate pasar el empleado por puntero para escribir el id generado en el objeto.