**Progetto 2 - Caso Webservices**

Gruppo **GossipWeb**:

*Lara Longhi (1079261)*

*Federico Carrara (1080669)*

*Alessia Lazzari (1078863)*

*Consegna*:

Con i webservice deve essere possibile migrare i dati che avete inserito su Altervista nel 1° progetto e migrarli verso un database locale su Postgres

Il webservice remoto è consultabile al link: <http://servsanitariopw9.altervista.org/WS.php>

Il webservice locale e la servlet sono visibili al link: <https://github.com/LLonghi02/PW24_WebService>

**“Dump” del database di origine**: DB – Ex3: Servizio Sanitario

Immagine che contiene testo, schermata, software, Icona del computer

Descrizione generata automaticamente

|  |  |
| --- | --- |
| Tabella Cittadino Immagine che contiene testo, software, Icona del computer, Software multimediale  Descrizione generata automaticamente | Tabella Ospedale Immagine che contiene testo, schermata, software, Icona del computer  Descrizione generata automaticamente |
| Tabella Patologia | Tabella Ricovero  Immagine che contiene testo, schermata, software, Icona del computer  Descrizione generata automaticamente |

|  |  |  |
| --- | --- | --- |
| Tabella PatologiaCronica  Immagine che contiene testo, software, Icona del computer, Software multimediale  Descrizione generata automaticamente | Tabella PatologiaMortale | Tabella PatologiaRicovero  Immagine che contiene testo, schermata, software, Icona del computer  Descrizione generata automaticamente |

***Fasi per l’installazione e l’avviamento***

1. **Scaricare il repository da GitHub**: <https://github.com/LLonghi02/PW24_WebService>
   * In questa repository, premere il pulsante code → download ZIP
   * Estrarre il file ZIP
2. **Avviare Apache Tomcat**
   * Comando windows+R → Aprire: services.msc
   * Click col tasto destro su “Apache Tomcat 10.1 Tomcat10” (o simile in base alla versione presente sul sistema) → avvia
   * Copiare la cartella “Servlet” presente al percorso "C:\...\PW24\_WebService\PW24\_SSanitario” nella cartella "C:\Tomcat\webapps" presente nel tuo sistema
3. **Creare un Database PostgreSQL**
   * Aprire “pgAdmin4”
   * Aprire il tuo Server
   * Fare clic con il tasto destro su Databases → Create → Database…
   * Chiamare il Database “ServSanitario” e salva
4. **Inserire i tuoi dati di accesso PostgreSQL in settings.py**
   * Aprire settings.py in VSC, trovabile al percorso "C:\...\PW24\_WebService\PW24\_SSanitario\PW24\_SSanitario"
   * Modificare DATABASES alla riga 80 con il tuo USER e la tua PASSWORD

Immagine che contiene testo, schermata, software, schermo

Descrizione generata automaticamente

1. **Aprire il terminale**
   * Fare clic con il tasto destro sulla cartella PW24\_WebService appena estratta e selezionare “Apri nel terminale”
2. **Installare le dipendenze e configurare l’ambiente**
   * Copiare i codici presenti nel file step\_SSanitario.ipynb nella console del terminale e seguire le indicazioni
3. **Entrare in Apache Tomcat**
   * Aprire il browser
   * Inserire nella barra di ricerca <http://localhost:8080/AppWeb/servletSanitario>
4. **Visualizzare le tabelle migrate in PostgreSQL**
   * Aprire “pgAdmin4” o effettua un refresh Databases se ancora aperto dal punto 3
   * Aprire il Database “ServSanitario”, creato in precedenza, estendi Schemas → public → Tables

Descrizione file presenti nella cartella *PW24\_SSanitario*:

1. importer
2. Servelet

Il file web.xml serve per la configurazione di una servlet in un'applicazione web Java. Si specifica il nome della servlet (Servlet), la classe java che la implementa (com.example.SanitarioServlet), il nome della servlet a cui si fa riferimento e il pattern dell’URL a cui questa servlet risponderà.

La classe Serlet.java implementa una servlet Java che:

* + Effettua una richiesta HTTP GET a un webservice remoto.
  + Legge e raccoglie la risposta da questo webservice.
  + Invia i dati ottenuti a un altro webservice locale (applicazione Django) tramite una richiesta HTTP POST.

1. PW24\_SSanitario

wsgi.py: Questo file configura l'interfaccia WSGI (Web Server Gateway Interface) per il progetto Django. WSGI è uno standard per la comunicazione tra i server web e le applicazioni web scritte in Python.

urls.py: Questo file mappa gli URL alle viste corrispondenti nel progetto Django.

asgi.py: Questo file configura l'interfaccia ASGI (Asynchronous Server Gateway Interface) per il progetto Django. ASGI supporta applicazioni web asincrone in Python.

settings.py: Questo file contiene tutte le impostazioni di configurazione per il progetto Django.