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

Fasi per l'installazione e l'avviamento

Per la realizzazione del progetto sono stati utilizzati i seguenti software:

- Tomcat 10.1
- Python 3.12
- Java 21
- PostgreSQL 16.3

L'utilizzo delle stesse versioni non è obbligatorio, ma consigliato per evitare problemi dovuta a incompatibilità con versioni precedenti.

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. **Verificare i prerequisiti Apache Tomcat**
   Per prima cosa è necessario assicurarsi di aver aggiunto le variabili necessarie alle variabili d'ambiente del proprio pc (sia per quanto riguarda Tomcat che Java). Nel caso non fosse stato fatto, seguire i seguenti passaggi altrimenti proseguire al punto 3.
   Aggiunta variabili d'ambiente Java:
   - Aprire la finestra delle *impostazioni di sistema avanzate* del proprio computer e selezionare *Variabili d'ambiente*
   - In *Variabili di sistema* premere su *Nuova* e inserire "JAVA_HOME" come nome e il percorso della directory della jdk (es. C:\Program Files\Java\jdk-versione) come valore
   - Cliccare su *Ok*
   - Cercare la variabile di sistema "path", selezionarla e premere su *Modifica*
   - Aggiungere il percorso della directory "bin" all'interno della directory della jdk ai campi di tale variabile
   - Confermare per applicare le modifiche
   - Aggiunta variabili d'ambiente Tomcat:
   - Aprire la finestra delle *impostazioni di sistema avanzate* del proprio computer e selezionare *Variabili d'ambiente*
   - In *Variabili di sistema* premere su *Nuova* e inserire "CATALINA_HOME" come nome e il percorso della directory di Tomcat (es. C:\Program Files\Apache Software Foundation\Tomcat Versione) come valore
   - Cliccare su *Ok*
   - Cercare la variabile di sistema "path", selezionarla e premere su *Modifica*
   - Aggiungere il percorso della directory "bin" all'interno della directory di Tomcat ai campi di tale variabile
   - Confermare per applicare le modifiche
3. **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 “AppWeb” presente al percorso "C:\...\PW24_WebService-main\PW24_SSanitario\Servlet” nella cartella "C:\Program Files\Apache Software Foundation\Tomcat 10.1\webapps" presente nel proprio sistema

4. **Creare un Database PostgreSQL**
   - Aprire “pgAdmin4”
   - Aprire Servers
   - Fare clic con il tasto destro su Databases → Create → Database...
   - Chiamare il Database “ServSanitario” e salvare
5. **Inserire i propri dati di accesso PostgreSQL in settings.py**
   - Aprire settings.py in VSC (va bene qualsiasi editor di testo), trovabile al percorso "C:\...\PW24_WebService-main\PW24_SSanitario\PW24_SSanitario"
   - Modificare DATABASES alla riga 80 con la propria PASSWORD di PostgreSQL

```
80  DATABASES = {
81      'default': {
82          'ENGINE': 'django.db.backends.postgresql',
83          'NAME': 'ServSanitario',
84          'USER': 'postgres',
85          'PASSWORD': 'password',
86          'HOST': 'localhost',
87          'PORT': '5432',
88      }
89  }
```

6. **Aprire il terminale**
   - Aprire il prompt dei comandi (attenzione NON deve essere Windows PowerShell), se eventualmente desse errori, aprire il prompt dei comandi da amministratore
7. **Installare le dipendenze e configurare l’ambiente**
   - Copiare le istruzioni presenti nel file step_SSanitario.ipynb nella console del terminale e seguire le indicazioni; tale file contiene delle stringhe di codice di Google colab e per essere visualizzato correttamente va aperto da Github.
   - In alternativa si riportano di seguito gli stessi passaggi presenti nel file:
     i. Spostarsi all’interno della cartella PW24_WebService-main scaricata da Github (utilizzare il seguente comando sostituendo il percorso con il proprio percorso):
        cd :\...\PW24_WebService-main
     ii. Attivare l’ambiente virtuale
        virtualDjango\Scripts\activate
     iii. Aggiornare *pip* (per sicurezza, nel caso non fosse presente l’ultima versione):
        python -m pip install --upgrade pip
     iv. Installare Django (dovrebbe già essere presente, ma eseguire comunque l’istruzione per evitare errori futuri):
        pip install django
     v. Installare psycopg2-binary (crea il collegamento tra Python e PostgreSQL)
        pip install psycopg2-binary
     vi. Installare requests (gestisce le richieste HTTP in Python)
        pip install requests
     vii. Installare django-cors-headers (gestisce le richieste CORS)
        pip install django-cors-headers
     viii. Entrare nella cartella del progetto Django
        cd PW24_SSanitario
     ix. Fare partire il server
        python manage.py runserver
8. **Entrare in Apache Tomcat**
   - Aprire il browser
   - Inserire nella barra di ricerca http://localhost:8080/AppWeb/servletSanitario
9. **Visualizzare le tabelle migrate in PostgreSQL**
   - Aprire “pgAdmin4” o effettuare un refresh del database se ancora aperto dal punto 3
   - Aprire il Database “ServSanitario”, creato in precedenza, e selezionare in sequenza estendi Schemas → public → Tables

- Per visualizzare il contenuto delle tabelle premere il tasto destro una volta posizionati con il cursore sopra una di esse e selezionare *View/Edit Data*

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

Tabella	Azione	Righe
Cittadino	Mostra Struttura Cerca Inserisci Svuota Elimina	110
Ospedale	Mostra Struttura Cerca Inserisci Svuota Elimina	48
Patologia	Mostra Struttura Cerca Inserisci Svuota Elimina	30
PatologiaCronica	Mostra Struttura Cerca Inserisci Svuota Elimina	15
PatologiaMortale	Mostra Struttura Cerca Inserisci Svuota Elimina	15
PatologiaRicovero	Mostra Struttura Cerca Inserisci Svuota Elimina	74
Ricovero	Mostra Struttura Cerca Inserisci Svuota Elimina	74
7 tabelle	Totali	366

Tabella Cittadino

CSSN	nome	cognome	dataNascita	luogoNascita	indirizzo
CSSN001	Mario	Rossi	1975-05-10	Roma	Via Roma 1
CSSN002	Luigi	Verdi	1980-08-15	Milano	Via Milano 2
CSSN003	Giovanna	Bianchi	1982-03-25	Napoli	Via Napoli 3
CSSN004	Giuseppe	Moretti	1978-11-30	Palermo	Via Palermo 4
CSSN005	Francesca	Rizzo	1985-09-20	Torino	Via Torino 5
CSSN006	Antonio	Ferrari	1973-12-05	Genova	Via Genova 6
CSSN007	Maria	Conti	1970-04-15	Bologna	Via Bologna 7
CSSN008	Paola	Martini	1988-07-28	Firenze	Via Firenze 8
CSSN009	Luca	Leone	1981-01-12	Bari	Via Bari 9
CSSN010	Laura	Galli	1977-06-08	Catania	Via Catania 10

Tabella Ospedale

codice	nome	città	indirizzo	direttoreSanitario
COD002	Ospedale	Mantova	Via San Nicola 11	CSSN002
COD004	Ospedale Santa Chiara	Varese	Via Santa Chiara 5	CSSN004
COD005	Ospedale	Monza	Via Maggiore 43	CSSN005
COD006	Ospedale Fatebenefratelli	Milano	Via Fatebenefratelli 5	CSSN006
COD008	Ospedale Mauriziano	Lecco	Via Mauriziano 7	CSSN008
COD009	Ospedale Policlinico	Cremona	Via Policlinico 8	CSSN009
COD010	Ospedale S. Maria della Misericordia	Mantova	Via Misericordia 9	CSSN010
COD011	Ospedale San Filippo	Milano	Via San Filippo 10	CSSN011
COD013	Ospedale San Giuseppe	Milano	Via San Giuseppe 12	CSSN013
COD014	Ospedale Cardarelli	Monza	Via Cardarelli 13	CSSN014

Tabella Patologia

cod	nome	criticità
PAT001	Influenza	7
PAT002	Diabete	4
PAT003	Ipertensione	9
PAT004	Asma	6
PAT005	Artrite	8
PAT006	Depressione	3
PAT007	Aritmia cardiaca	5
PAT008	Malattia di Alzheimer	2
PAT009	Cancro al polmone	6

Tabella Ricovero

codOspedale	cod	paziente	data	durata	motivo	costo
COD004	RIC004	CSSN004	2023-07-25	4	Asma	2200.00
COD005	RIC005	CSSN005	2023-08-30	6	Artrite	3200.00
COD004	RIC009	CSSN009	2023-12-20	6	Cancro al polmone	3500.00
COD005	RIC010	CSSN010	2024-01-25	7	Fibrosi cistica	4000.00
COD004	RIC014	CSSN014	2024-05-20	4	Malattia di Crohn	2200.00
COD005	RIC015	CSSN015	2024-06-25	6	Sindrome del colon irritabile	3200.00
COD004	RIC019	CSSN019	2024-10-15	6	Cataratta	3500.00
COD005	RIC020	CSSN020	2024-11-20	7	Emicrania	4000.00
COD004	RIC024	CSSN024	2025-03-15	4	Pancreatite	2200.00
COD005	RIC025	CSSN025	2025-04-20	6	Tubercolosi	3200.00
COD004	RIC029	CSSN029	2025-08-10	6	Sindrome di Down	3500.00
COD005	RIC030	CSSN030	2025-09-15	7	Malattia di Addison	4000.00

Tabella PatologiaCronica

codPatologia
PAT002
PAT003
PAT005
PAT006
PAT007
PAT014
PAT015
PAT016

Tabella PatologiaMortale

codPatologia
PAT001
PAT004
PAT008
PAT009
PAT010
PAT011
PAT012
PAT013

Tabella PatologiaRicovero

codOspedale	codRicovero	codPatologia
COD005	RIC091	PAT001
COD016	RIC061	PAT001
COD045	RIC062	PAT002
COD046	RIC092	PAT002
COD010	RIC063	PAT003
COD031	RIC093	PAT003
COD004	RIC004	PAT004
COD004	RIC034	PAT004

Informazioni di progetto

Descrizione del progetto

Il progetto utilizza Django per migrare dati da un server remoto a uno locale. L'architettura prevede l'interazione tra un'applicazione Django e una servlet Java, che opera su un server Tomcat, per effettuare la migrazione dei dati tramite richieste HTTP.

Scelte di progetto

Utilizzo di Tomcat

- Server di Servlet Affidabile: Tomcat è uno dei server di servlet più popolari e affidabili.
- Compatibilità e Performance: Tomcat è ottimizzato per eseguire servlet Java e offre buone prestazioni per le applicazioni web basate su Java.
- Configurazione Semplice: La configurazione di servlet e altre risorse web è facilitata tramite il file web.xml, permettendo una gestione centralizzata delle impostazioni dell'applicazione.

Informazioni sulle tabelle in PostgreSQL

Django crea automaticamente alcune tabelle necessarie per il funzionamento del sistema. Grazie al file 'routers.py', abbiamo evitato la creazione della maggior parte di queste tabelle. Al termine del punto 8, in PostgreSQL potrai vedere le tabelle indicate nel dump e una tabella speciale chiamata 'django_migrations'.
La tabella 'django_migration' è gestita direttamente da Django ed è fondamentale per il sistema di migrazioni. Questa tabella è indispensabile e non può essere eliminata, poiché tiene traccia delle migrazioni già applicate e consente a Django di sapere quali migrazioni devono ancora essere eseguite.

La tabella PatologiaRicovero(codOspedale, codRicovero, codPatologia) è strutturata in modo tale da avere tre chiavi esterne, che sono anche chiavi primarie. Tuttavia, in Django, un modello non può avere più di un campo come chiave primaria. Per gestire questo problema, si utilizza il comando 'unique_together', che riconosce questi campi come una chiave primaria composta. Nel database PostgreSQL, questo viene rappresentato con l'aggiunta di una colonna 'id', che definisce una combinazione unica di campi.

Descrizione file presenti nella cartella PW24_SSanitario*:*

1. Applicazione Django: importer
   L'applicazione importer è una componente chiave del progetto le sue principali funzionalità includono:
   - views.py : Contiene le viste che gestiscono le richieste HTTP e restituiscono le risposte appropriate.
   - models.py : Definisce le strutture dei dati e le interazioni con il database.
2. Servlet
   La servlet è strutturata in due directory principali: AppWeb e src.
   - AppWeb/WEB-INF:
     - web.xml: File di configurazione per la servlet in un'applicazione web Java.
     - classes:
       - Servlet.class: Il file compilato della servlet, generato dal file Servlet.java.
   - src:
     - Servlet.java: Classe che implementa la servlet Java con le seguenti funzionalità:
       - doGet e doPost : Gestiscono le richieste HTTP GET e POST, rispettivamente, chiamando il metodo 'handleRequest'.
       - handleRequest :
         Recupera dati da un webservice remoto su Altervista con una richiesta HTTP POST.
         Invia i dati recuperati a un webservice locale Django tramite un'altra richiesta HTTP POST.
       - fetchDataFromRemoteService : Effettua la richiesta al webservice remoto e legge la risposta.
       - sendToDjango : Invia i dati al server Django locale e gestisce la risposta.

   Questa servlet facilita la migrazione dei dati tra un server remoto e un'applicazione Django locale. La servlet opera su un server Tomcat, che gestisce l'esecuzione della servlet e le interazioni con i webservice.
3. Progetto Django: PW24_SSanitario
   - urls.py : File principale che mappa gli URL del progetto Django alle relative viste e applicazioni. Questo file coordina la navigazione all'interno del progetto.

- settings.py : Contiene tutte le impostazioni di configurazione per il progetto Django, come la configurazione del database, le applicazioni installate, le impostazioni di sicurezza, e altri parametri necessari per il funzionamento dell'applicazione.
- routers.py : Personalizza la gestione delle operazioni di lettura e scrittura e controlla le migrazioni, escludendo alcune app di sistema dalle migrazioni.