Documentazione progetto TPSIT

# Descrizione del progetto

Lo scopo principale è aiutare gli utenti che desiderano avere un acquario a scegliere le piante giuste per loro da usare nel loro aquascape.

Si vuole realizzare un progetto java che utilizzi un web-service locale con metodo di comunicazione mediante XML e interpretazione dei dati tramite la libreria JAXB.  
Lo scopo è quello di realizzare un web-service mediante **mysql** che dia accesso a tutte le piante per acquario con le relative informazioni (nome, link pagina, link immagine, tipo, origine, tasso crescita, altezza minima, altezza massima, luce, co2, link logo, difficoltà). L’app java ha il compito di avere la parte client che deve permettere la ricerca (per nome, filtrata di vario tipo) delle piante, accedendo al web service e facendosi inviare tramite XML i dati che poi verranno riportati come oggetti da java tramite JAXB, e poi verranno fatti visualizzare all’utente.

# Architettura del sistema

**Descrizione dell'architettura complessiva:** Il sistema è composto da due componenti principali: un web service RESTful in PHP e un client Java. Il web service ospita un database MySQL di informazioni sulle piante d'acquario.   
  
Il client Java ha un’interfaccia collegata al web service per permettere agli utenti di:

* Ricercare le piante per nome, tipo, origine, tasso di crescita, luminosità richiesta, necessità di CO2 e altri criteri.
* Visualizzare i dettagli di una specifica pianta, tra cui immagini, nome, continente di origine, dimensioni, valori di temperatura ideali.
* Salvare le proprie ricerche e creare una lista di piante preferite (Acquario).

**Componenti chiave del sistema:**

**Web service:**

* Sviluppato in PHP con Apache Web Server.
* Database MySQL che memorizza informazioni su:
  + centinaia di specie di piante d'acquario.
  + immagini.
  + link a siti web con informazioni più approfondite.
* API RESTful per la ricerca, la ricezione e la modifica dei dati sulle piante.

**Client Java:**

* Libreria JAXB per la mappatura XML-Java con creazione delle classi tramite xsd e libreria xjc.
* Interfaccia utente grafica intuitiva per:
  + Inserire i criteri di ricerca.
  + Visualizzare i risultati della ricerca in una tabella con filtri e ordinamento.
  + Visualizzare i dettagli di una specifica pianta con immagini e informazioni complete.
  + Salvare le ricerche e le piante preferite.

**Interazioni tra i componenti:**

1. L'utente interagisce con l'interfaccia del client Java.
2. Il client Java invia una richiesta al web service (PHP) tramite l'API RESTful locale.
3. Il web service (PHP) elabora la richiesta, si connette al database MySQL e recupera i dati.
4. Il web service (PHP) costruisce la risposta in formato XML seguendo l’XSD definito.
5. Il client Java utilizza JAXB per mappare i dati XML in oggetti Java.
6. Il client Java visualizza i dati all'utente.

**Diagramma dell'architettura:**

Immagine che contiene testo, diagramma, disegno, design

Descrizione generata automaticamenteDiagramma dell'architettura del sistema:

Diagramma dello schema [xsd](plants.htm) utilizzato.

# 3. Descrizione del Web-service (PHP)

**Scopo:** Fornire un servizio RESTful per la richiesta alle informazioni sulle piante d'acquario.

**Funzionalità:**

* Gestione delle richieste HTTP GET per la ricerca e il recupero dei dati sulle piante.
* Connessione al database MySQL per l'interrogazione dei dati.
* Generazione dinamica di XML seguendo lo schema XSD (plants.xsd) per la risposta alle richieste.
* Restituzione di piante per nome, tipo, origine, tasso di crescita, luminosità richiesta, necessità di CO2 e altri criteri.
* Recupero dei dettagli di una specifica pianta, tra cui immagini, nome scientifico, famiglia, continente di origine, dimensioni, valori di pH e temperatura ideali, consigli per la cura e la propagazione.
* Filtraggio e ordinamento dei risultati della ricerca.
* Paginazione dei risultati della ricerca.

**Parametri di input:**

* **Ricerca per nome:**
  + nome: il nome della pianta da ricercare.
* **Ricerca con filtri avanzati:**
  + tipo: il tipo di pianta (es: erba, anubias, ecc.).
  + origine: l'origine della pianta (es: America, Asia, Africa, ecc.).
  + tasso\_crescita: il tasso di crescita della pianta (es: lento, medio, veloce).
  + luce: la quantità di luce richiesta dalla pianta (es: bassa, media, alta).
  + co2: la necessità di CO2 per la crescita della pianta (es: no, si).
  + ph\_minimo: il valore minimo di pH tollerato dalla pianta.
  + ph\_massimo: il valore massimo di pH tollerato dalla pianta.
  + temperatura\_minima: la temperatura minima tollerata dalla pianta.
  + temperatura\_massima: la temperatura massima tollerata dalla pianta.

# 4. Requisiti del Progetto

**Funzionalità:**

* Il sistema deve permettere agli utenti di:
  + Ricercare le piante per nome, tipo, origine, tasso di crescita, luminosità richiesta, necessità di CO2 e altri criteri.
  + Visualizzare i dettagli di una specifica pianta, tra cui immagini, nome scientifico, famiglia, continente di origine, dimensioni, valori di pH e temperatura ideali, consigli per la cura e la propagazione.
  + Salvare le proprie ricerche e creare una lista di piante preferite.
* Il sistema deve essere in grado di gestire un elevato numero di utenti e di richieste, Il sistema deve essere sicuro e protetto da accessi non autorizzati (implementazione futura).

Immagine che contiene cerchio, disegno, schizzo, diagramma

Descrizione generata automaticamenteDiagramma dei casi d’uso dell’applicazione client:

SELEZIONA N°  
PIANTE

ddassdasada

**Prestazioni:**

* Il sistema deve essere in grado di rispondere alle richieste in modo rapido e efficiente.
* Il tempo di caricamento delle pagine deve essere minimo.

# 5. Implementazione

**Linguaggi di programmazione utilizzati:**

* Java 8 à eclipse v. 2023-09
* PHP 7.4
* SQL

**Framework o librerie utilizzate**

librerie utilizzate:

* Swing e AWT: Questi sono framework standard per la creazione di interfacce utente grafiche (GUI) in Java.
* JAXB (Java Architecture for XML Binding): Utilizzato per la serializzazione e deserializzazione degli oggetti Java in e da XML.
* JGoodies Forms: Utilizzato per la gestione e la disposizione dei componenti GUI.
* MigLayout: Un layout manager flessibile e potente per Swing e AWT.
* JDK Libraries: Java Development Kit (JDK) fornisce una vasta gamma di librerie standard come java.util, java.awt, java.net, java.io, ecc.
* JDK Stream API: Utilizzato per operazioni di streaming e manipolazione di collezioni.
* Jakarta EE (anteriormente Java EE): Jakarta EE per la gestione delle operazioni XML.
* Java Standard Libraries: Alcune delle tue importazioni sono componenti standard di Java come java.util, java.awt, java.io, ecc.
* Window builder: Utilizzato per costruire e gestire la GUI.

**Descrizione delle principali fasi di sviluppo:**

* Analisi dei requisiti: Definizione delle funzionalità dell'applicazione e delle caratteristiche desiderate.
* Progettazione: Creazione di diagrammi UML per illustrare le classi e le loro relazioni, e della prima parte della documentazione.
* Implementazione: Sviluppo del codice sorgente per le diverse classi e funzionalità in Java e PHP.
* Test: Esecuzione di test unitari e di integrazione per la verifica del corretto funzionamento, eseguito da una persona esterna che non conosceva il codice sorgente.
* Debug: Correzione di eventuali errori e bug trovati durante i test.
* Deploy: Distribuzione dell'applicazione su un ambiente di produzione.

**Struttura del codice sorgente:**

Lato Java:

Il codice sorgente è organizzato in pacchetti Java:

* piante: Contiene le classi che modellano le informazioni sulle piante d'acquario.
* view: Contiene la classe Window che rappresenta l'interfaccia utente.
* control: Contiene la classe Controller che gestisce l'interazione con l'utente.
* model: Contiene le classi che gestiscono le funzionalità dell’applicazione.
* org.eclipse.wb.swing: contiene la classe FocusTraversalOnArray

Lato PHP:

Il codice sorgente è organizzato in file PHP:

* **genera\_xml.php**

Genera i file XML per tutte le tabelle del database.

utilizza le funzioni:

* generateXmlForTable($conn, $tableName, $xsdPath, $xmlFileName): genera un XML per una specifica tabella.
* generateXmlFromResultSet($resultSet, $xsdPath, $rootTagName): genera un XML da un ResultSet.
* displayImageFromBlob($blobData, $contentType): visualizza un'immagine da un campo BLOB.
* **carica\_da\_xml.php**

Inserisce i dati XML nel database.

utilizza la funzione:

* insertXmlIntoDatabase($xmlString, $conn): inserisce i dati XML nel database.
* **restEasy.php**

Genera l'XML per la tabella "piante" con filtri.

Richiede i parametri dalla query string:

* table: nome della tabella.
* nome: filtro per nome pianta.
* origine: filtro per origine.
* tasso\_crescita: filtro per tasso di crescita.
* luce: filtro per luce.
* co2: filtro per CO2.
* difficolta: filtro per difficoltà.

Utilizza le funzioni:

* generateXmlFromResultSet($result, $xsdPath, 'piante').
* insertXmlIntoDatabase($xmlString, $conn).
* function createQuery($crud,$table, $nome, $origine, $tasso\_crescita, $luce, $co2, $difficolta, $id\_pianta, $id\_acquario).
* **le tabelle accessibili tramite l'API sono**:

1. Piante:

* L'API permette di recuperare informazioni sulle piante, come nome, origine, tasso di crescita, etc.
* È possibile filtrare i risultati in base a diversi parametri.
* L'API non permette di modificare o eliminare le informazioni sulle piante.

2. Immagini:

* L'API permette di recuperare le immagini delle piante codificate in base64.
* Non è possibile caricare nuove immagini o modificare quelle esistenti tramite l'API.

3. Acquari:

* L'API permette il salvataggio di acquari solo tramite l’invio in POST di un xml adeguatamente formattato.
* Permette la visualizzazione degli acquari salvati.

4. Piante\_Acquari:

* L'API permette il salvataggio di piante in un acquario solo tramite l’invio in POST di un xml adeguatamente formattato.
* Permette la visualizzazione delle piante salvate in un determinato acquario.

**Descrizione delle scelte architetturali o di design:**

* Separazione tra logica e interfaccia utente: Il codice è stato strutturato per separare la logica dell'applicazione dall'interfaccia utente, rendendo il codice più modulare e manutenibile.
* Utilizzo di un design MVC (Model-View-Controller): Il design MVC è stato adottato per migliorare la separazione delle preoccupazioni e la modularità del codice.
* Sviluppo di un'API RESTful: Un'API RESTful è stata sviluppata in PHP per permettere al client Java di comunicare con il database.

# 6. Test e validazione

**Strategia di test adottata:**

**Obiettivo:**Garantire che tutte le funzionalità del sistema, inclusa la gestione delle piante, degli acquari e delle interazioni con l'API, funzionino correttamente e senza errori.

**Tipi di Test:**

1. Test di Unità: Verificare singolarmente le funzionalità delle classi e dei metodi.
2. Test di Integrazione: Verificare l'integrazione tra i componenti del sistema.
3. Test di Sistema: Verificare l'intero sistema nel suo complesso.
4. Test di Accettazione: Verificare che il sistema soddisfi i requisiti specificati dal cliente.

**Procedure di Test:**

1. Test di Unità:
   * Verificare che ogni metodo delle classi di modello (PianteType, AcquariType, ecc.) funzioni correttamente.
2. Test di Integrazione:
   * Verificare che i metodi delle classi di controllo (Controller, RestEasyPlantsClient, ecc.) interagiscano correttamente tra loro.
3. Test di Sistema:
   * Verificare che l'interfaccia utente funzioni correttamente.
   * Verificare l'integrazione con l'API esterna (fetchDataFromApi, sendDataToApi, ecc.).
4. Test di Accettazione:
   * Verificare che il sistema soddisfi i requisiti funzionali e non funzionali specificati nella documentazione.
   * Eseguire test manuali basati su scenari utente reali.

**Risultati e Report:**

**Test di unità superati**

**Test di Integrazione**

**Componenti**: **Controller**, **RestEasyPlantsClient**

* Verificata l'integrazione tra **Controller** e **RestEasyPlantsClient**.
* Tutte le chiamate API sono state eseguite con successo.

**Test di Sistema**

* L'interfaccia utente risponde correttamente agli input dell'utente.
* Le chiamate API esterne (**fetchDataFromApi**, **sendDataToApi**) hanno restituito i dati attesi.
* Nessun errore di sistema riscontrato durante i test.

**Test di Accettazione**

* Il sistema soddisfa tutti i requisiti funzionali specificati nel documento dei requisiti.
* Tutti gli scenari utente reali hanno avuto esito positivo.

# Immagine che contiene testo, schermata, software, schermo Descrizione generata automaticamente7. Deployment e gestione del progetto

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

Descrizione generata automaticamenteImmagine che contiene testo, schermata, software, numero

Descrizione generata automaticamenteImmagine che contiene testo, schermata, software, numero

Descrizione generata automaticamente

Immagine che contiene testo, schermata, software, schermo

Descrizione generata automaticamenteImmagine che contiene schermata, testo, software, schermo

Descrizione generata automaticamenteImmagine che contiene testo, schermata, software, schermo

Descrizione generata automaticamente  
Immagine che contiene testo, schermata, software, Software multimediale

Descrizione generata automaticamente  
Immagine che contiene testo, schermata, software, schermo

Descrizione generata automaticamente