Il progetto: il *web-service* per le *APP* dei dispositivi mobili

Il *web-service* del sistema di registrazione e notifica dei consumi con cui interagiscono le *APP* dei dispositivi mobili del personale addetto alla lettura dei contatori viene presentato nella forma di una *servlet* Java che implementa la web API documentata in precedenza e utilizza il database definito nella risposta al quesito 1.

```
package it.acquagas.rete.APIAPP;

import java.io.*;
import java.net.*;
import java.sql.*;
import javax.servlet.*;
import javax.servlet.http.*;
import javax.xml.parsers.*;
import org.w3c.dom.*;
import org.xml.sax.*;
import javax.net.ssl.*;

public class APIAPP extends HttpServlet {
  final static private String driver = "com.mysql.jdbc.Driver";
  final static private String dbms_url = "jdbc:mysql://127.0.0.1:3306/";
  final static private String database = "acqua_gas";
  final static private String user = "root";
  final static private String password = "";
  private static Connection connection;
  private static boolean connected;

  // attivazione servlet (connessione a DBMS)
  public void init() {
    String url = dbms_url+database;
    try {
          Class.forName(driver);
          connection = DriverManager.getConnection(url, user, password);
          connected = true;
    }
    catch (SQLException exception) {
          connected = false;
    }
    catch (ClassNotFoundException exception) {
          connected = false;
    }
  }

  // disattivazione servlet (disconnessione da DBMS)
  public void destroy() {
    try {
          connection.close();
    }
    catch (SQLException exception) {
    }
  }
```

```
// richiede la mappa della posizione del contatore al servizio
// Google Static Maps e la restituisce al client
private void getMappa(String contatore, HttpServletResponse response)
                    throws IOException {
  ...
  ...
  ...
}
// verifica API key
private void testKey(String APIkey, HttpServletResponse response)
            throws IOException {
  if (APIkey == null) {
    response.sendError(400, "Missing API key!");
    return;
  }
  try {
    Statement statement = connection.createStatement();
    ResultSet resultset = statement.executeQuery("SELECT * FROM Dispositivo
                                                  WHERE API_key = '" + APIkey
                                                  + "';");
    if (resultset.next()) {
    }
    else {
      response.sendError(400, "Wrong API key!");
      resultset.close();
      statement.close();
      return;
    }
    resultset.close();
    statement.close();
  }
  catch (SQLException exception) {
    response.sendError(500, "DBMS server error!");
  }
}
// richiesta dati cliente o mappa contatore
protected void doGet(HttpServletRequest request, HttpServletResponse response)
               throws ServletException, IOException {
    String url, contatore, APIkey;
    String[] url_section;
    String CF, cognome, nome, indirizzo, numero, CAP, località, comune, provincia;
    // verifica stato connessione a DBMS
    if (!connected) {
      response.sendError(500, "DBMS server error!");
      return;
    }
    // verifica API key
    APIkey = request.getParameter("key");
    testKey(APIkey, response);
    // verifica URL e identificatore contatore
    url = request.getRequestURL().toString();
    url_section = url.split("/");
    if (url_section.length < 2) {
       response.sendError(400, "Malformed URL!");
```

```
   return;
}
contatore = url_section[url_section.length-1];
if (contatore == null) {
  response.sendError(400, "Request syntax error!");
  return;
}
int len = url_section.length;
if (!url_section[len-2].equalsIgnoreCase("cliente")) {
  if (url_section[len-2].equalsIgnoreCase("mappa")) {
    getMappa(contatore, response); // richiesta mappa posizione
    return;
  }
  response.sendError(405, "Wrong request!");
  return;
}
if (contatore.isEmpty()) {
  response.sendError(400, "Request syntax error!");
  return;
}
// ricerca dati cliente nel database
try {
      Statement statement = connection.createStatement();
      String query = "SELECT * FROM Contatore, Cliente WHERE
                     Contatore.Cliente = Cliente.Codice_fiscale
                     AND Contatore.ID_contatore = '" +
                     contatore + "';";
      ResultSet resultset = statement.executeQuery(query);
      if (resultset.next()) {
        CF = resultset.getString("Cliente.Codice_fiscale");
        cognome = resultset.getString("Cliente.Cognome");
        nome = resultset.getString("Cliente.Nome");
        indirizzo = resultset.getString("Cliente.Indirizzo");
        numero = resultset.getString("Cliente.Numero");
        località = resultset.getString("Cliente.Località");
        CAP = resultset.getString("Cliente.CAP");
        comune = resultset.getString("Cliente.Comune");
        provincia = resultset.getString("Cliente.Provincia");
      }
      else {
            response.sendError(404, "Not found!");
            resultset.close();
            statement.close();
            return;
      }
      resultset.close();
      statement.close();
      // scrittura del body della risposta
      response.setContentType("text/xml;charset=UTF-8");
      PrintWriter out = response.getWriter();
      try {
            out.println("<?xml version=\"1.0\" encoding=\"UTF-8\"?>\r\n");
            out.println("<cliente ID_contatore=\"" + contatore + "\">\r\n");
            out.print("<CF>");
```

```
                out.print(CF);
                out.println("</CF>\r\n");
                out.print("<cognome>");
                out.print(cognome);
                out.println("</cognome>\r\n");
                out.print("<nome>");
                out.print(nome);
                out.println("</nome>\r\n");
                out.print("<indirizzo>");
                out.print(indirizzo);
                out.println("</indirizzo>\r\n");
                out.print("<numero>");
                out.print(numero);
                out.println("</numero>\r\n");
                out.print("<località>");
                out.print(località);
                out.println("</località>\r\n");
                out.print("<CAP>");
                out.print(CAP);
                out.println("</CAP>\r\n");
                out.print("<comune>");
                out.print(comune);
                out.println("</comune>\r\n");
                out.print("<provincia>");
                out.print(provincia);
                out.println("</provincia>");
                out.println("</cliente>");
            }
            finally {
                    out.close();
            }
            response.setStatus(200); // OK
      }
      catch (SQLException exception) {
        response.sendError(500, "DBMS server error!");
        return;
      }
    }

    // inoltro dati contatore
    protected void doPost (HttpServletRequest request, HttpServletResponse response)
                  throws ServletException, IOException {
      String url, contatore, APIkey, tipo;
      String[] url_section;
      float lettura = 0;

      // verifica stato connessione a DBMS
      if (!connected) {
        response.sendError(500, "DBMS server error!");
        return;
      }
      // verifica API key
      APIkey = request.getParameter("key");
      testKey(APIkey, response);
      // verifica URL e identificatore contatore
```

```
url = request.getRequestURL().toString();
url_section = url.split("/");
if (url_section.length < 2) {
  response.sendError(400, "Malformed URL!");
  return;
}
contatore = url_section[url_section.length-1];
if (contatore == null) {
  response.sendError(400, "Request syntax error!");
  return;
}
int len = url_section.length;
if (!url_section[len-2].equalsIgnoreCase("lettura")) {
  response.sendError(405, "Wrong request!");
  return;
}
if (contatore.isEmpty()) {
  response.sendError(400, "Request syntax error!");
  return;
}
try {
      // estrazione dell'attributo "tipo" e
      // del valore dell'elemento "lettura" dal body XML
      BufferedReader content = request.getReader();
      DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
      DocumentBuilder builder = factory.newDocumentBuilder();
      InputSource input = new InputSource();
      input.setCharacterStream(content);
      Document document = builder.parse(input);
      Element root = document.getDocumentElement();
      if (!root.getNodeName().equalsIgnoreCase("lettura")) {
        response.sendError(400, "Wrong request body!");
        return;
      }
      tipo = root.getAttribute("tipo");
      if (!tipo.equalsIgnoreCase("A") &&
          !tipo.equalsIgnoreCase("G")) {
        response.sendError(400, "Wrong request body!");
      }
      String tmp = root.getFirstChild().getNodeValue();
      try {
            lettura = Float.valueOf(tmp);
      }
      catch (NumberFormatException exception) {
        response.sendError(400, "Wrong request body!");
        return;
      }
      // inserimento della lettura nel database
      try {
            String command = "INSERT Lettura(Contatore, Data_ora,
                             Consumo, Dispositivo) VALUES('" +
                             contatore + "', NOW(), " +
                             Float.toString(lettura) + ", '" + APIkey + "');";
            Statement statement = connection.createStatement();
```

```
            if (statement.executeUpdate(command) <= 0) {
              statement.close();
              response.sendError(404, "Wrong meter ID?");
            }
            statement.close();
          }
          catch (SQLException exception) {
            response.sendError(500, "DBMS server error!");
            return;
          }
          response.setStatus(201); // OK CREATED
      }
      catch (ParserConfigurationException exception) {
        response.sendError(500, "XML parser error!");
        return;
      }
      catch (SAXException exception) {
        response.sendError(500, "XML parser error!");
        return;
      }
    }

    // inoltro posizione contatore
    protected void doPut (HttpServletRequest request,HttpServletResponse response)
                  throws ServletException, IOException {
      String url, contatore, APIkey;
      String[] url_section;
      double latitudine = 0.0;
      double longitudine = 0.0;

      // verifica stato connessione a DBMS
      if (!connected) {
        response.sendError(500, "DBMS server error!");
        return;
      }
      // verifica API key
      APIkey = request.getParameter("key");
      testKey(APIkey, response);
      // verifica URL e identificatore contatore
      url = request.getRequestURL().toString();
      url_section = url.split("/");
      if (url_section.length < 2) {
        response.sendError(400, "Malformed URL!");
        return;
      }
      contatore = url_section[url_section.length-1];
      if (contatore == null) {
        response.sendError(400, "Request syntax error!");
        return;
      }
      int len = url_section.length;
      if (!url_section[len-2].equalsIgnoreCase("posizione")) {
        response.sendError(405, "Wrong request!");
        return;
      }
```

```
if (contatore.isEmpty()) {
  response.sendError(400, "Request syntax error!");
  return;
}
try {
      // estrazione dell'attributo "tipo" e dei valori degli
      // elementi "latitudine" e "longitudine" dal body XML
      BufferedReader content = request.getReader();
      DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
      DocumentBuilder builder = factory.newDocumentBuilder();
      InputSource input = new InputSource();
      input.setCharacterStream(content);
      Document document = builder.parse(input);
      Element root = document.getDocumentElement();
      if (!root.getNodeName().equalsIgnoreCase("posizione")) {
        response.sendError(400, "Wrong request body!");
        return;
      }
      NodeList list = root.getElementsByTagName("latitudine");
      String tmp = null;
      if (list != null && list.getLength() > 0)
        tmp = list.item(0).getFirstChild().getNodeValue();
      try {
            latitudine = Double.valueOf(tmp);
      }
      catch (NumberFormatException exception) {
        response.sendError(400, "Wrong request body!");
      }
      list = root.getElementsByTagName("longitudine");
      tmp = null;
      if (list != null && list.getLength() > 0)
        tmp = list.item(0).getFirstChild().getNodeValue();
      try {
        longitudine = Double.valueOf(tmp);
      }
      catch (NumberFormatException exception) {
        response.sendError(400, "Wrong request body!");
        return;
      }
      // aggiornamento valori latitudine/longitudine nel database
      try {
            String command = "UPDATE Contatore SET Latitudine = "
                             + Double.toString(latitudine) + ",
                             Longitudine = " +
                             Double.toString(longitudine) + "
                             WHERE ID_contatore = '" + contatore
                             + "';";
            Statement statement = connection.createStatement();
            if (statement.executeUpdate(command) <= 0) {
              statement.close();
              response.sendError(404, "Wrong meter ID?");
              return;
            }
```

```
            statement.close();
          }
          catch (SQLException exception) {
            response.sendError(500, "DBMS server error!");
            return;
          }
          response.setStatus(204); // OK NO-CONTENT
      }
      catch (ParserConfigurationException exception) {
        response.sendError(500, "XML parser error!");
        return;
      }
      catch (SAXException exception) {
        response.sendError(500, "XML parser error!");
        return;
      }
    }

    // descrizione della servlet
    public String getServletInfo() {
      return "APIAPP servlet";
    }
}
```

Mashup del servizio Google *Static Maps*

La mappa della posizione del contatore viene richiesta nella forma di un'immagine in formato PNG al *web-service Static Maps* di Google; l'URL di richiesta comprende i seguenti parametri nella *query-string*:

- *center*: specifica le coordinate geografiche (latitudine/longitudine) del centro della mappa;
- *zoom*: specifica il livello di zoom della mappa, 20 è il massimo livello e corrisponde alla visualizzazione degli edifici;
- *size*: specifica la dimensione in pixel dell'immagine restituita, in questo caso 512×512;
- *maptype*: specifica il tipo di mappa, «hybrid» è il tipo che sovrappone gli elementi cartografici alla fotografia satellitare;
- *markers*: specifica uno o più marcatori, in questo caso uno che rappresenta la posizione del contatore al centro della mappa.

```
// richiede la mappa della posizione del contatore al servizio
// Google Static Maps e la restituisce al client
private void getMappa(String contatore, HttpServletResponse response)
                      throws IOException {
  double latitudine = 0.0, longitudine = 0.0;
  String prefix = "https://maps.googleapis.com/maps/api/staticmap?center=";
  String suffix = "&zoom=20&size=512x512&maptype=hybrid&markers=";
  // ricerca coordinate contatore nel database
  try {
        Statement statement = connection.createStatement();
        String query = "SELECT Latitudine, Longitudine FROM
                        Contatore WHERE Contatore.ID_contatore = '"
                        + contatore + "';";
```

```
        ResultSet resultset = statement.executeQuery(query);
        if (resultset.next()) {
          latitudine = resultset.getDouble(1);
          longitudine = resultset.getDouble(2);
        }
        else {
          response.sendError(404, "Not found!");
          resultset.close();
          statement.close();
          return;
        }
        resultset.close();
        statement.close();
    }
    catch (SQLException exception) {
      response.sendError(500, "DBMS server error!");
      return;
    }
    // richiesta mappa
    String url = prefix + latitudine + "," + longitudine + suffix +
                 latitudine + "," + longitudine;
    URL server = new URL(url);
    HttpsURLConnection service = (HttpsURLConnection) server.openConnection();
    service.setRequestProperty("Accept", "image/png");
    service.setDoInput(true);
    service.setRequestMethod("GET");
    // connessione al web-service
    service.connect();
    // verifica stato risposta
    int status = service.getResponseCode();
    if (status != 200) {
      response.sendError(500, "Google web-service error!");
      return;
    }
    // mashup del body della risposta
    response.setContentType("image/png");
    OutputStream out = response.getOutputStream();
    InputStream in = service.getInputStream();
    byte[] buffer = new byte[65536];
    int n;
    while ((n = in.read(buffer)) >= 0)
      out.write(buffer, 0, n);
    in.close();
    out.flush();
    out.close();
    response.setStatus(200); // OK
}
```