```
package fi.paatti.paattidatabaseutils.dbservice;


import com.vaadin.data.Container;
import com.vaadin.data.util.sqlcontainer.SQLContainer;
import fi.paatti.paattidatabaseutils.dbobjects.TaskData;
import java.util.Stack;


/**
 * The interface for the mobile client.
 *
 * @author Lauri Satokangas, lauri.n.satokangas@student.jyu.fi
 * @author Tapio Keränen, t.tapio.keranen@student.jyu.fi
 */
public interface PaattiMobileDBService {

  /**
   * Queries the database for the given table and returns an SQLContainer
   * containing the query results.
   * <p/>
   * SHOULD NOT BE USED FOR DELETING ROWS! (instead, just change the row
   * status to 'DELETED'.
   * <p/>
   * @param tableName the name of the table.
   * @param nameOfIDColumn the name of the id column.
   * @param noDeletedRows whether to filter deleted rows or not.
   * @return SQLContainer containing the items from the given table.
   */
  public SQLContainer getSQLContainerFromDBTable(String tableName, String nameOfIDColumn, boolean noDeletedRows);

  /**
   * Queries the database for the given table and returns an SQLContainer
   * containing the query results. The given filter (if any) will be applied
   * to the SQLContainer to exclude rows from the results.
   * <p/>
   * <b>SHOULD NOT BE USED FOR DELETING ROWS! (instead, just change the row
   * status to 'DELETED'.</b>
   * <p/>
   * @param tableName the name of the table.
   * @param nameOfIDColumn the name of the ID column
   * @param noDeletedRows whether to filter deleted rows or not.
   * @param filter the filter to be used.
   * @return SQLContainer containing all the items from the given table.
   */
  public SQLContainer getSQLContainerFromDBTableFiltered(String tableName, String nameOfIDColumn, boolean noDeletedRows, ⬋
```

```
Container.Filter filter);

  /**
   * Returns the value of the given column.
   *
   * @param tableName the name of the table the column belongs to.
   * @param columnName the name of the column.
   * @param IDColumn the ID column of the table.
   * @param rowID the ID of the row where the value will be taken from.
   * @return The found value or null if none found.
   */
  public Object getCellValueFromTable(String tableName, String columnName, String IDColumn, Object rowID);

  /**
   * Saves all the data from a single event.
   *
   * @param userID the ID of the user.
   * @param taskDataStack the list of TaskData items.
   * @return True if successful, otherwise false.
   */
  public boolean saveEventTaskData(Object userID, Stack<TaskData> taskDataStack);

  /**
   * Returns all the available events for the user.
   *
   * @param userID the ID of the user.
   * @return SQLContainer containing all the available events.
   */
  public SQLContainer getUserUnfinishedEvents(Object userID);

  /**
   * Returns all the tasks that belong to the event with the given id.
   *
   * @param eventID the ID of the event.
   * @return SQLContainer containing all the tasks of the event.
   */
  public SQLContainer getTaskContainer(Object eventID);

  /**
   * Returns all the choices that belong to the event with the given id.
   *
   * @param eventID the ID of the event.
   * @return SQLContainer containing all the choices of the event.
   */
  public SQLContainer getChoiceContainer(Object eventID);
```

```
  /**
   * Returns all the users that belong to the group with the given id.
   *
   * @param groupID the ID of the group.
   * @return SQLContainer containing the members of the group.
   */
  public SQLContainer getGroupUsers(Object groupID);

  /**
   * Returns all the voluntary events that are assigned to the user.
   *
   * @param userID the ID of the user.
   * @return SQLContainer containing the voluntary events.
   */
  public SQLContainer getUserVoluntaryEvents(Object userID);

  /**
   * Returns the event data from the user’s finished events.
   *
   * @param userId the ID of the user.
   * @return SQLContainer containing the event data.
   */
  public SQLContainer getUserEventData(Object userId);
}
```