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


import com.vaadin.data.Container;
import com.vaadin.data.util.IndexedContainer;
import com.vaadin.data.util.sqlcontainer.SQLContainer;
import fi.paatti.containers.ChoiceContainer;
import fi.paatti.containers.TaskContainer;
import java.util.Date;


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

  /**
   * Returns the SQLContainer for the given database table.
   * <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);

  /**
   * Returns the SQLContainer for the given database table. The given filter
   * (if any) will be applied to the SQLContainer to exclude rows from the
   * 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 include "DELETED" rows or not.
   * @param filter the filter to be used.
   * @return SQLContainer containing the filtered items from the given table.
   */
  public SQLContainer getSQLContainerFromDBTableFiltered(String tableName, String nameOfIdColumn, boolean noDeletedRows, 
Container.Filter filter);

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

  /**
   * Returns all the schedules that belong to the group with the given id.
   *
   * @param groupId the group’s id.
   * @return SQLContainer containing the schedules of the group.
   */
  public SQLContainer getGroupSchedules(Object groupId);

  /**
   * Returns all the researches that belong to the group with the given id.
   *
   * @param groupId the group’s id.
   * @return SQLContainer containing all the researches of the group.
   */
  public SQLContainer getGroupResearch(Object groupId);

  /**
   * Returns all the groups that belong to the research with the given id.
   *
   * @param researchId the research’s id.
   * @return SQLContainer containing all the groups of the research.
   */
  public SQLContainer getResearchGroups(Object researchId);

  /**
   * Returns all the groups that belong to the user with the given id.
   *
   * @param userId the user’s id.
   * @return SQLContainer containing all groups of the user.
   */
  public SQLContainer getUserGroups(Object userId);

  /**
   * Returns all the event data that belongs to the user with the given id.
   *
   * @param userId the user’s id.
   * @return SQLContainer containing all the event data of the user.
   */
  public SQLContainer getUserEventData(Object userId);

  /**
   * Returns all the events that belong to the schedule with the given id.
   *
   * @param scheduleId the schedule’s id.
   * @return SQLContainer containing all the events of the schedule.
   */
  public SQLContainer getEventsInSchedule(Object scheduleId);

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

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

  /**
   * Saves user’s information to the database. If the user with the given id
   * already exists in the database, the existing name and description values
   * are replaced. If no user was found, a new row is created to the database
   * with the given values.
   *
   * @param userId the id of the user.
   * @param userName the user name of the user.
   * @param firstName the first name of the user.
   * @param lastName the last name of the user.
   * @param userPassword the password of the user.
   * @param userDescription the description of the user.
   * @return True if successful and no exception was thrown, otherwise false.
   */
  public Object createUser(Object userId, String userName, String firstName, String lastName, String userPassword, String 
userDescription);
```

```java
    /**
     * Removes the row from the table. Removing means setting the row’s
     * row status to ’DELETED’.
     *
     * @param itemID the id of the row to be removed.
     * @param tableName the name of the table containing the row
     * @param idColumnName the id column of the table
     * @return True if successful and no exception was thrown, otherwise false.
     */
    public boolean removeRowFromTable(Object itemID, String tableName, String idColumnName);

    /**
     * Saves the research’s information to the database. If the research with the
     * given id already exists in the database, the existing name and description
     * values are replaced. If no research was found, a new row is created to
     * the database with the given values.
     *
     * @param name the name of the research.
     * @param description the description of the research.
     * @param researchID the id of the research.
     * @return RowId if successful and no exception was thrown; otherwise null.
     */
    public Object createResearch(String name, String description, Object researchID);

    /**
     * Saves the group’s information to the database. If a group with the given
     * id already exists in the database, the existing name, description, role,
     * and editor values are replaced. If no group was found, a new row is
     * created to the database with the given values.
     *
     * @param name the name of ther group.
     * @param description the description of the group.
     * @param role the role of the group.
     * @param research the id of the research this group belong to.
     * @param editor the id of the editor.
     * @param groupID the id of the group.
     * @return RowId if successful and no exception was thrown, otherwise null.
     */
    public Object createGroup(String name, String description, Object role, Object research, int editor, Object groupID);

    /**
     * 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);

    /**
     * Adds the user to the group. If the user already belongs to the group, the
     * rowstatus of the BELONGS table is set to ’ACTIVE’. If no row is found in
     * the table, a new row is created with the given values.
     *
     * @param groupId the id of the group
     * @param userId the id of the user
     * @return True if successful and no exception was thrown, otherwise false.
     */
    public boolean addUserToGroup(Object groupId, Object userId);

    /**
     * Removes the user from the group. Removing the user from the group means
     * setting the row status in BELONGS table to ’DELETED’.
     *
     * @param groupId the ID of the user’s group.
     * @param userId the ID of the user.
     * @return True if successful, otherwise false.
     */
    public boolean removeUserFromGroup(Object groupId, Object userId);

    /**
     * Creates a new schedule.
     *
     * @param description the description of the schedule.
     * @param zeroTime the start date of the schedule.
     * @param scheduleID the id of the schedule.
     * @param scheduledEvents the list of events that belong to the schedule.
     * @return The row id of the schedule if successful, otherwise null.
     */
    public Object createSchedule(String description, Date zeroTime, Object scheduleID, IndexedContainer scheduledEvents);

    /**
     * Adds the schedule to the group.
     *
     * @param groupID the id of the group.
     * @param scheduleID the id of the schedule.
     * @return True if succesful and no exception was thrown, otherwise false.
     */
    public boolean addScheduleToGroup(Object groupID, Object scheduleID);
```

```java
  /**
   * Returns the TaskContainer containing all the tasks from the given event.
   * The TaskContainer is an extended version of SQLContainer. A listener can
   * be added to it for getting the new row ID after a successful insertion to
   * the database.
   *
   * @param eventId the ID of the event.
   * @return All the tasks of the event.
   */
  public TaskContainer getTaskContainerWithListener(Object eventId);

  /**
   * Returns the ChoiceContainer containing all the choices from the given
   * event. The ChoiceContainer is an extended version of SQLContainer. A
   * listener can be added to it for getting the new row ID after a successful
   * insertion to the database.
   *
   * @param eventId the ID of the event.
   * @return All the choices of the event.
   */
  public ChoiceContainer getChoiceContainerWithListener(Object eventId);

  /**
   * Returns the roles and researches of the user’s groups. These properties
   * can be accessed with the values from PaattiColumnNames.
   *
   * @param userID the ID of the user.
   * @return The SQLContainer containing the user’s role and research info.
   */
  public SQLContainer getUserGroupsdRolesAndResearches(Integer userID);

  /**
   * Returns the role of the given group.
   *
   * @param groupId the id of the group.
   * @return The SQLContainer containing the group’s role info.
   */
  public SQLContainer getGroupRole(Object groupId);
}
```