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


import com.vaadin.data.Container.Filter;
import com.vaadin.data.Item;
import com.vaadin.data.Property;
import com.vaadin.data.util.IndexedContainer;
import com.vaadin.data.util.filter.Compare;
import com.vaadin.data.util.sqlcontainer.SQLContainer;
import com.vaadin.data.util.sqlcontainer.connection.SimpleJDBCConnectionPool;
import com.vaadin.data.util.sqlcontainer.query.FreeformQuery;
import com.vaadin.data.util.sqlcontainer.query.QueryDelegate;
import com.vaadin.data.util.sqlcontainer.query.QueryDelegate.RowIdChangeEvent;
import com.vaadin.data.util.sqlcontainer.query.TableQuery;
import fi.paatti.containers.ChoiceContainer;
import fi.paatti.containers.TaskContainer;
import fi.paatti.paattidatabaseutils.dbobjects.TaskData;
import fi.paatti.paattidatabaseutils.dbservice.querydelegates.*;
import fi.paatti.paattidatabaseutils.names.PaattiColumnNames;
import fi.paatti.paattidatabaseutils.names.PaattiTableNames;
import java.io.Serializable;
import java.sql.SQLException;
import java.util.Date;
import java.util.HashMap;
import java.util.Stack;


/**
 * The implementation of the interfaces PaattiResearchDBService and
 * PaattiMobileDBService.
 * <p/>
 * TODO: Remove system.out messages and add logging.
 * <p/>
 * @author Lauri Satokangas, lauri.n.satokangas@student.jyu.fi
 * @author Tapio Keränen, t.tapio.keranen@student.jyu.fi
 * @author Toni Salminen, toni.a.j.salminen@student.jyu.fi
 * @author Jari Salokangas, jari.p.t.salokangas@student.jyu.fi
 */
public class PaattiDBService implements PaattiResearchDBService, PaattiMobileDBService, Serializable {
  private static final long serialVersionUID = 1L;
  private SimpleJDBCConnectionPool connectionPool;
  private final String dbDriverName = "com.mysql.jdbc.Driver";
  private Object scheduleRowId = null;
  private Object eventTimeRowId = null;
  private Object eventRowId = null;
  private Object taskRowId = null;
```

```
  private Object groupRowId = null;
  private Object researchRowId = null;
  private Object userRowId = null;

  /**
   * Constructor.
   *
   * @param dbAddress The address of the database.
   * @param dbName    The name of the database.
   * @param userName  The username that should be used for logging.
   * @param password  The password that should be used for logging.
   */
  public PaattiDBService(String dbAddress, String dbName, String userName, String password) {
    initConnectionPool(dbAddress, dbName, userName, password);
  }

  /**
   * Initializes the connection pool.
   *
   * @param dbAddress The address of the database.
   * @param dbName The name of the database.
   * @param userName The username that should be used for logging in.
   * @param password The password that should be used for logging in.
   */
  private void initConnectionPool(String dbAddress, String dbName, String userName, String password) {
    try {
      connectionPool = new SimpleJDBCConnectionPool(dbDriverName,
          "jdbc:mysql://" + dbAddress + ":3306/" + dbName + "?autoReconnect=true", userName, password);
    } catch (Exception e) {
      System.out.println("Virhe luotaessa tietokantayhteyttä." + e);
    }
  }

  /**
   * Queries the database for the given table and returns an SQLContainer
   * containing the query results.
   *
   * @param tableName the name of the database table.
   * @param nameOfIdColumn the name of the id column.
   * @param noDeletedRows whether to filter deleted rows or not.
   * @return The SQLContainer if successful, otherwise null.
   */
  public SQLContainer getSQLContainerFromDBTable(String tableName, String nameOfIdColumn, boolean noDeletedRows) {
    TableQuery query = new TableQuery(tableName, connectionPool);

    query.setVersionColumn(nameOfIdColumn);
```

```
    SQLContainer container = null;

    try {
      container = new SQLContainer(query);

      if (noDeletedRows) {
        Filter activeRowFilter = new Compare.Equal(tableName + "_rowStatus", PaattiQueryDelegate.ROWSTATUS_ACTIVE);

        container.addContainerFilter(activeRowFilter);
      }
    } catch (SQLException ex) {
      System.out.println("virhe haettaessa taulua " + tableName + ": " + ex.getMessage());
    }
    return container;
  }

  /**
   * 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/>
   * A word of warning: filtered SQLContainers are immutable, so don't use
   * this if you're planning on modifying the container later.
   * <p/>
   * @param tableName the name of the database table.
   * @param nameOfIdColumn the name of the id column.
   * @param noDeletedRows whether to filter deleted rows or not.
   * @param filter the filter that is applied to the search results.
   * @return The filtered SQLContaner if successful, otherwise null.
   */
  public SQLContainer getSQLContainerFromDBTableFiltered(String tableName, String nameOfIdColumn, boolean noDeletedRows, Filter filter) {
    SQLContainer sqlContainerFromDBTable = getSQLContainerFromDBTable(tableName, nameOfIdColumn, noDeletedRows);

    sqlContainerFromDBTable.addContainerFilter(filter);
    return sqlContainerFromDBTable;
  }

  /**
   * Queries the database with the given query string by using a FreeformQuery
   * and returns an SQLContainer containing the query results. A delegate must
   * be given to the FreeformQuery or else all SQL actions (INSERT etc.) will
   * fail.
   *
   * @param queryString the SQL statement.
```

```
   * @param tableColumnId the id column of the target table.
   * @param delegate the delegate related to the table.
   * @return The SQLContainer if successful, otherwise null.
   */
  private SQLContainer getSQLContainerFromDBTableWithDelegate(String queryString, String tableColumnId, PaattiQueryDelegate
delegate) {
    FreeformQuery query = new FreeformQuery(queryString, connectionPool, tableColumnId);

    if (delegate != null) {
      query.setDelegate(delegate);
    }

    SQLContainer container = null;

    try {
      container = new SQLContainer(query);
    } catch (Exception ex) {
      System.out.println("Virhe haussa " + queryString + ": " + ex.getMessage());
    }
    return container;
  }

  /**
   * {@inheritDoc }
   */
  public SQLContainer getTaskContainer(Object eventId) {
    return getSQLContainerFromDBTableWithDelegate("SELECT * FROM " + PaattiTableNames.TASK,
        PaattiColumnNames.TASK_taskID, new TaskQueryDelegate(eventId));
  }

  /**
   * {@inheritDoc }
   */
  public TaskContainer getTaskContainerWithListener(Object eventId) {
    FreeformQuery query = new FreeformQuery("SELECT * FROM " + PaattiTableNames.TASK, connectionPool,
        PaattiColumnNames.TASK_taskID);

    query.setDelegate(new TaskQueryDelegate(eventId));

    TaskContainer container = null;

    try {
      container = new TaskContainer(query);
    } catch (Exception ex) {
      System.out.println("Virhe haettaessa " + ex.getMessage());
    }
```

```
    return container;
  }

  /**
   * {@inheritDoc }
   */
  public SQLContainer getChoiceContainer(Object eventId) {
    return getSQLContainerFromDBTableWithDelegate("SELECT * FROM " + PaattiTableNames.CHOICE,
        PaattiColumnNames.CHOICE_choiceID, new ChoiceQueryDelegate(eventId));
  }

  /**
   * {@inheritDoc }
   */
  public ChoiceContainer getChoiceContainerWithListener(Object eventId) {
    FreeformQuery query = new FreeformQuery("SELECT * FROM " + PaattiTableNames.CHOICE, connectionPool,
        PaattiColumnNames.CHOICE_choiceID);

    query.setDelegate(new ChoiceQueryDelegate(eventId));

    ChoiceContainer container = null;

    try {
      container = new ChoiceContainer(query);
    } catch (Exception ex) {
      System.out.println("Virhe haettaessa " + ex.getMessage());
    }
    return container;
  }

  /**
   * {@inheritDoc }
   */
  public SQLContainer getUserGroups(Object userId) {
    String queryString = "SELECT * FROM " + PaattiTableNames.USERGROUP + " WHERE "
        + PaattiColumnNames.USERGROUP_usergroupID + " IN (" + " SELECT "
        + PaattiColumnNames.BELONGS_USERGROUP_usergroupID + " FROM " + PaattiTableNames.BELONGS + " WHERE "
        + PaattiColumnNames.BELONGS_rowStatus + " = '" + PaattiQueryDelegate.ROWSTATUS_ACTIVE + "' AND "
        + PaattiColumnNames.BELONGS_USER_userID + " = '" + userId + "')" + " AND "
        + PaattiColumnNames.USERGROUP_rowStatus + " = '" + PaattiQueryDelegate.ROWSTATUS_ACTIVE + "';";

    return getSQLContainerFromDBTableWithDelegate(queryString, PaattiColumnNames.USERGROUP_usergroupID, null);
  }

  /**
   * {@inheritDoc }
```

```
   */
  public SQLContainer getGroupUsers(Object groupId) {
    String queryString = "SELECT * FROM " + PaattiTableNames.USER + " WHERE " + PaattiColumnNames.USER_userID + " IN ("
        + " SELECT " + PaattiColumnNames.BELONGS_USER_userID + " FROM " + PaattiTableNames.BELONGS + " WHERE "
        + PaattiColumnNames.BELONGS_rowStatus + " = '" + PaattiQueryDelegate.ROWSTATUS_ACTIVE + "' AND "
        + PaattiColumnNames.BELONGS_USERGROUP_usergroupID + " = '" + groupId + "')" + " AND "
        + PaattiColumnNames.USER_rowStatus + " = '" + PaattiQueryDelegate.ROWSTATUS_ACTIVE + "';";

    return getSQLContainerFromDBTableWithDelegate(queryString, PaattiColumnNames.USER_userID, null);
  }

  /**
   * {@inheritDoc }
   */
  public SQLContainer getGroupSchedules(Object groupId) {
    String queryString = "SELECT * FROM " + PaattiTableNames.SCHEDULE + " WHERE "
        + PaattiColumnNames.SCHEDULE_USERGROUP_usergroupID + " = '" + groupId + "' AND "
        + PaattiColumnNames.SCHEDULE_rowStatus + " = '" + PaattiQueryDelegate.ROWSTATUS_ACTIVE + "';";

    return getSQLContainerFromDBTableWithDelegate(queryString, PaattiColumnNames.SCHEDULE_scheduleID, null);

  }

  /**
   * {@inheritDoc }
   */
  public SQLContainer getGroupResearch(Object groupId) {
    String queryString = "SELECT * FROM " + PaattiTableNames.RESEARCH + " WHERE "
        + PaattiColumnNames.RESEARCH_researchID + " IN (" + " SELECT " + PaattiColumnNames.USERGROUP_RESEARCH_researchID
        + " FROM " + PaattiTableNames.USERGROUP + " WHERE " + PaattiColumnNames.USERGROUP_usergroupID + " = '" + groupId
        + "')" + " AND " + PaattiColumnNames.RESEARCH_rowStatus + " = '" + PaattiQueryDelegate.ROWSTATUS_ACTIVE + "';";

    return getSQLContainerFromDBTableWithDelegate(queryString, PaattiColumnNames.RESEARCH_researchID, null);
  }

  /**
   * {@inheritDoc }
   */
  public SQLContainer getResearchGroups(Object researchId) {
    String queryString = "SELECT * FROM " + PaattiTableNames.USERGROUP + " WHERE "
        + PaattiColumnNames.USERGROUP_RESEARCH_researchID + " = '" + researchId + "' AND "
        + PaattiColumnNames.USERGROUP_rowStatus + " = '" + PaattiQueryDelegate.ROWSTATUS_ACTIVE + "';";

    return getSQLContainerFromDBTableWithDelegate(queryString, PaattiColumnNames.USERGROUP_usergroupID, null);

  }
```

```
  /**
   * {@inheritDoc }
   */
  public SQLContainer getUserEventData(Object userId) {
    String queryString = "SELECT * FROM " + PaattiTableNames.EVENT + " LEFT OUTER JOIN " + PaattiTableNames.TASK
        + " ON " + PaattiColumnNames.TASK_EVENT_eventID + " = " + PaattiColumnNames.EVENT_eventID + " LEFT OUTER JOIN "
        + PaattiTableNames.CHOICE + " ON " + PaattiColumnNames.CHOICE_TASK_contains_taskID + " = "
        + PaattiColumnNames.TASK_taskID + " LEFT OUTER JOIN " + PaattiTableNames.TASKDATA + " ON "
        + PaattiColumnNames.TASKDATA_CHOICE_choiceID + " = " + PaattiColumnNames.CHOICE_choiceID + " WHERE "
        + PaattiColumnNames.TASKDATA_USER_userID + " = " + userId;

    return getSQLContainerFromDBTableWithDelegate(queryString, PaattiColumnNames.TASKDATA_taskDataID, null);
  }

  /**
   * {@inheritDoc }
   */
  public SQLContainer getUserUnfinishedEvents(Object userId) {
    return getSQLContainerFromDBTableWithDelegate("SELECT * FROM " + PaattiTableNames.EVENT,
        PaattiColumnNames.EVENT_eventID, new UserUnfinishedEventsDelegate(userId));
  }

  /**
   * {@inheritDoc }
   */
  public SQLContainer getGroupRole(Object groupId) {
    String queryString = "SELECT * FROM " + PaattiTableNames.ROLE + " WHERE " + PaattiColumnNames.ROLE_roleID + " IN ("
        + " SELECT " + PaattiColumnNames.USERGROUP_ROLE_roleID + " FROM " + PaattiTableNames.USERGROUP + " WHERE "
        + PaattiColumnNames.USERGROUP_usergroupID + " = '" + groupId + "' AND " + PaattiColumnNames.USERGROUP_rowStatus
        + " = '" + PaattiQueryDelegate.ROWSTATUS_ACTIVE + "')" + " WHERE " + PaattiColumnNames.ROLE_rowStatus + " = '"
        + PaattiQueryDelegate.ROWSTATUS_ACTIVE + "';";

    return getSQLContainerFromDBTableWithDelegate(queryString, PaattiColumnNames.ROLE_roleID, null);
  }

  /**
   * {@inheritDoc }
   */
  public Object createUser(Object userID, String userName, String firstName, String lastName, String userPassword, String ↙
userDescription) {
    SQLContainer container = getSQLContainerFromDBTable(PaattiTableNames.USER, PaattiColumnNames.USER_userID, false);

    container.addListener(new QueryDelegate.RowIdChangeListener() {
      private static final long serialVersionUID = 1L;
```

```
      public void rowIdChange(RowIdChangeEvent event) {
        userRowId = event.getNewRowId().getId()[0];
      }
    });

    Object itemId;
    Item item = null;
    Property property;

    boolean notFound = true;

    if (userID != null) {
      for (Object id : container.getItemIds()) {
        item = container.getItem(id);

        if (item.getItemProperty(PaattiColumnNames.USER_userID).toString().equals(userID)) {
          notFound = false;
          break;
        }
      }
      if (notFound) {
        return null;
      }
    } else {
      itemId = container.addItem();
      item = container.getItem(itemId);
    }

    property = item.getItemProperty(PaattiColumnNames.USER_name);
    property.setValue(userName);

    property = item.getItemProperty(PaattiColumnNames.USER_firstName);
    property.setValue(firstName);

    property = item.getItemProperty(PaattiColumnNames.USER_lastName);
    property.setValue(lastName);

    property = item.getItemProperty(PaattiColumnNames.USER_password);
    property.setValue(userPassword);

    property = item.getItemProperty(PaattiColumnNames.USER_description);
    property.setValue(userDescription);

    property = item.getItemProperty(PaattiColumnNames.USER_rowStatus);
    property.setValue(PaattiQueryDelegate.ROWSTATUS_ACTIVE);
```

```
    try {
      container.commit();
    } catch (Exception ex) {
      ex.printStackTrace();
      return null;
    }

    if (notFound) {
      return userRowId;
    } else {
      return userID;
    }
  }

  /**
   * {@inheritDoc }
   */
  public boolean removeRowFromTable(Object itemID, String tableName, String idColumnName) {
    SQLContainer container = getSQLContainerFromDBTable(tableName, idColumnName, false);

    System.out.println("item: " + itemID);
    Item item;
    Property property;

    for (Object id : container.getItemIds()) {
      item = container.getItem(id);

      if (item.getItemProperty(idColumnName).getValue().toString().equals(itemID.toString())) {
        property = item.getItemProperty(tableName + "_rowStatus");
        property.setValue(PaattiQueryDelegate.ROWSTATUS_DELETED);
        break;
      }
    }
    try {
      container.commit();
    } catch (Exception ex) {
      ex.printStackTrace();
      return false;
    }
    return true;

  }

  /**
   * {@inheritDoc }
   */
```

```
  public Object createResearch(String name, String description, Object researchID) {
    SQLContainer container = getSQLContainerFromDBTable(PaattiTableNames.RESEARCH, PaattiColumnNames.RESEARCH_researchID,
        false);

    container.addListener(new QueryDelegate.RowIdChangeListener() {
      public void rowIdChange(RowIdChangeEvent event) {
        researchRowId = event.getNewRowId().getId()[0];
      }
    });

    Object itemId;
    Item item = null;
    Property property;

    boolean notFound = true;

    if (researchID != null) {
      for (Object id : container.getItemIds()) {
        item = container.getItem(id);

        if (item.getItemProperty(PaattiColumnNames.RESEARCH_researchID).toString().equals(researchID)) {
          notFound = false;
          break;
        }
      }
      if (notFound) {
        return null;
      }
    } else {
      itemId = container.addItem();
      item = container.getItem(itemId);
    }

    property = item.getItemProperty(PaattiColumnNames.RESEARCH_name);
    property.setValue(name);

    property = item.getItemProperty(PaattiColumnNames.RESEARCH_description);
    property.setValue(description);

    property = item.getItemProperty(PaattiColumnNames.RESEARCH_rowStatus);
    property.setValue(PaattiQueryDelegate.ROWSTATUS_ACTIVE);

    try {
      container.commit();
    } catch (Exception ex) {
      ex.printStackTrace();
```

```
      return null;
    }

    if (notFound) {
      return researchRowId;
    } else {
      return researchID;
    }
  }

  /**
   * {@inheritDoc }
   */
  public Object createGroup(String name, String description, Object role, Object research, int editor, Object groupID) {
    SQLContainer container = getSQLContainerFromDBTable(PaattiTableNames.USERGROUP,
        PaattiColumnNames.USERGROUP_usergroupID, false);

    container.addListener(new QueryDelegate.RowIdChangeListener() {
      public void rowIdChange(RowIdChangeEvent event) {
        groupRowId = event.getNewRowId().getId()[0];
      }
    });

    Object itemId;
    Item item = null;
    Property property;

    boolean notFound = true;

    if (groupID != null) {
      for (Object id : container.getItemIds()) {
        item = container.getItem(id);
        if (item.getItemProperty(PaattiColumnNames.USERGROUP_usergroupID).toString().equals(groupID)) {
          notFound = false;
          break;
        }
      }
      if (notFound) {
        return null;
      }
    } else {
      itemId = container.addItem();
      item = container.getItem(itemId);
    }

    property = item.getItemProperty(PaattiColumnNames.USERGROUP_name);
```

```
    property.setValue(name);

    property = item.getItemProperty(PaattiColumnNames.USERGROUP_description);
    property.setValue(description);

    property = item.getItemProperty(PaattiColumnNames.USERGROUP_RESEARCH_researchID);
    property.setValue(research);

    property = item.getItemProperty(PaattiColumnNames.USERGROUP_ROLE_roleID);
    property.setValue(role);

    property = item.getItemProperty(PaattiColumnNames.USERGROUP_editedBy);
    property.setValue(editor);

    property = item.getItemProperty(PaattiColumnNames.USERGROUP_rowStatus);
    property.setValue(PaattiQueryDelegate.ROWSTATUS_ACTIVE);

    try {
      container.commit();
    } catch (Exception ex) {
      ex.printStackTrace();
      return null;
    }

    if (notFound) {
      return groupRowId;
    } else {
      return groupID;
    }
  }

  /**
   * {@inheritDoc }
   */
  public Object getCellValueFromTable(String tableName, String columnName, String idColumn, Object rowId) {
    SQLContainer container;
    Item item = null;
    Object cellValue = null;
    String queryString = "SELECT * FROM " + tableName + " WHERE " + idColumn + " = " + rowId;
    FreeformQuery query = new FreeformQuery(queryString, connectionPool, idColumn);

    try {
      container = new SQLContainer(query);
      Object itemId = container.firstItemId();

      item = container.getItem(itemId);
```

```
    } catch (SQLException ex) {
      System.out.println("Error when getting a value from table: " + ex);
    }
    if (item != null) {
      cellValue = item.getItemProperty(columnName).getValue().toString();
    }

    return cellValue;
  }

  /**
   * {@inheritDoc }
   */
  public boolean saveEventTaskData(Object userID, Stack<TaskData> taskDataStack) {
    SQLContainer taskDataCont = getSQLContainerFromDBTable(PaattiTableNames.TASKDATA,
        PaattiColumnNames.TASKDATA_taskDataID, false);

    for (Object columnname : taskDataCont.getContainerPropertyIds()) {
      System.out.println(columnname);
    }
    System.out.println("TASKDATACONTAINER: " + taskDataCont + " : " + taskDataCont.size());

    while (taskDataStack.size() > 0) {
      TaskData taskData = taskDataStack.pop();
      Object addItemID = taskDataCont.addItem();

      Item newTaskDataItem = taskDataCont.getItem(addItemID);

      System.out.println("taskData.getChoiceID(): " + taskData.getChoiceID());

      Property itemProperty = newTaskDataItem.getItemProperty(PaattiColumnNames.TASKDATA_CHOICE_choiceID);

      itemProperty.setValue(taskData.getChoiceID());

      System.out.println("taskData.getChoiceSequence() " + taskData.getChoiceSequence());
      Property itemProperty1 = newTaskDataItem.getItemProperty(PaattiColumnNames.TASKDATA_sequence);

      itemProperty1.setValue(taskData.getChoiceSequence());

      System.out.println("USER ID : " + userID);
      Property itemProperty2 = newTaskDataItem.getItemProperty(PaattiColumnNames.TASKDATA_USER_userID);

      itemProperty2.setValue(userID);

      Property itemProperty3 = newTaskDataItem.getItemProperty(PaattiColumnNames.TASKDATA_rowStatus);
```

```
      itemProperty3.setValue(PaattiQueryDelegate.ROWSTATUS_ACTIVE);

    }
    try {
      taskDataCont.commit();
    } catch (Exception ex) {
      ex.printStackTrace();
      return false;
    }

    return true;
  }

  /**
   * {@inheritDoc }
   */
  public boolean addUserToGroup(Object groupId, Object userId) {
    SQLContainer container = getSQLContainerFromDBTable(PaattiTableNames.BELONGS, PaattiColumnNames.BELONGS_editTime,
        false);

    Object itemId;
    Item item;
    Property property;

    for (Object id : container.getItemIds()) {
      item = container.getItem(id);

      Object g = item.getItemProperty(PaattiColumnNames.BELONGS_USERGROUP_usergroupID).getValue();
      Object c = item.getItemProperty(PaattiColumnNames.BELONGS_USER_userID).getValue();

      if (groupId.toString().equals(g.toString()) && userId.toString().equals(c.toString())) {
        property = item.getItemProperty(PaattiColumnNames.BELONGS_rowStatus);
        property.setValue(PaattiQueryDelegate.ROWSTATUS_ACTIVE);

        try {
          container.commit();
        } catch (Exception ex) {
          ex.printStackTrace();
          return false;
        }

        return true;
      }
    }

    itemId = container.addItem();
```

```
    item = container.getItem(itemId);

    property = item.getItemProperty(PaattiColumnNames.BELONGS_USERGROUP_usergroupID);
    property.setValue(groupId);

    property = item.getItemProperty(PaattiColumnNames.BELONGS_USER_userID);
    property.setValue(userId);

    property = item.getItemProperty(PaattiColumnNames.BELONGS_rowStatus);
    property.setValue(PaattiQueryDelegate.ROWSTATUS_ACTIVE);

    try {
      container.commit();
    } catch (Exception ex) {
      ex.printStackTrace();
      return false;
    }

    return true;
  }

  /**
   * {@inheritDoc }
   */
  public boolean removeUserFromGroup(Object groupId, Object userId) {
    SQLContainer container;

    FreeformQuery groupQuery = new FreeformQuery("asdf", connectionPool, PaattiColumnNames.BELONGS_editTime);

    groupQuery.setDelegate(new BelongsQueryDelegate(groupId, userId));

    try {
      container = new SQLContainer(groupQuery);

      Object itemId = container.firstItemId();

      container.removeItem(itemId);

      container.commit();

    } catch (Exception ex) {
      System.out.println("removeUserFromGroup() soft warning: " + ex.getLocalizedMessage());
      ex.printStackTrace();
      return false;
    }
```

```
    return true;
  }

  /**
   * {@inheritDoc }
   */
  public SQLContainer getEventsInSchedule(Object scheduleId) {
    return getSQLContainerFromDBTableWithDelegate("SELECT * FROM " + PaattiTableNames.EVENT,
        PaattiColumnNames.EVENT_eventID, new EventsInScheduleDelegate(scheduleId));
  }

  /**
   * {@inheritDoc }
   */
  public SQLContainer getUserVoluntaryEvents(Object userID) {
    return getSQLContainerFromDBTableWithDelegate("SELECT * FROM " + PaattiTableNames.EVENT,
        PaattiColumnNames.EVENT_eventID, new VoluntaryEventsDelegate(userID));

  }

  /**
   * {@inheritDoc }
   */
  public SQLContainer getUserGroupsdRolesAndResearches(Integer userID) {
    return getSQLContainerFromDBTableWithDelegate("SELECT * FROM " + PaattiTableNames.USERGROUP,
        PaattiColumnNames.USERGROUP_usergroupID, new UserGroupRolesAndResearchDelegate(userID));
  }

  /**
   * {@inheritDoc }
   */
  public Object createSchedule(String description, Date zeroTime, Object scheduleID, IndexedContainer scheduledEvents) {
    SQLContainer scheduleContainer = getSQLContainerFromDBTable(PaattiTableNames.SCHEDULE,
        PaattiColumnNames.SCHEDULE_scheduleID, false);
    SQLContainer eventTimeContainer = getSQLContainerFromDBTable(PaattiTableNames.EVENTTIME,
        PaattiColumnNames.EVENTTIME_eventTimeID, false);
    SQLContainer eventContainer = getSQLContainerFromDBTable(PaattiTableNames.EVENT, PaattiColumnNames.EVENT_eventID,
        false);

    HashMap<Object, Object> taskHashMap = new HashMap<Object, Object>();

    scheduleContainer.addListener(new QueryDelegate.RowIdChangeListener() {

      private static final long serialVersionUID = 1L;

      public void rowIdChange(QueryDelegate.RowIdChangeEvent event) {
```

```
        scheduleRowId = event.getNewRowId().getId()[0];
      }
    });

    eventTimeContainer.addListener(new QueryDelegate.RowIdChangeListener() {

      private static final long serialVersionUID = 1L;

      public void rowIdChange(QueryDelegate.RowIdChangeEvent event) {
        eventTimeRowId = event.getNewRowId().getId()[0];
      }
    });

    eventContainer.addListener(new QueryDelegate.RowIdChangeListener() {

      private static final long serialVersionUID = 1L;

      public void rowIdChange(QueryDelegate.RowIdChangeEvent event) {
        eventRowId = event.getNewRowId().getId()[0];
      }
    });

    // Creates a schedule ------------------------------------------------
    Object itemId;
    Item scheduleItem;
    Property property;

    itemId = scheduleContainer.addItem();
    scheduleItem = scheduleContainer.getItem(itemId);

    property = scheduleItem.getItemProperty(PaattiColumnNames.SCHEDULE_description);
    property.setValue(description);

    property = scheduleItem.getItemProperty(PaattiColumnNames.SCHEDULE_zeroTime);
    property.setValue(zeroTime);

    // When creating new shcedule, user group id needs to be set to a value
    // that indicates that the schedule is not connected to any group.
    property = scheduleItem.getItemProperty(PaattiColumnNames.SCHEDULE_USERGROUP_usergroupID);
    property.setValue(-1);

    property = scheduleItem.getItemProperty(PaattiColumnNames.SCHEDULE_rowStatus);
    property.setValue(PaattiQueryDelegate.ROWSTATUS_ACTIVE);

    try {
      scheduleContainer.commit();
```

```
    } catch (Exception ex) {
      ex.printStackTrace();
      return null;
    }

    // Creates event times ------------------------------------------------
    Item scheduledEventItem;
    Item eventTimeItem;
    Object eventTimeItemId;
    Property eventTimeProperty;

    for (Object eventItemID : scheduledEvents.getItemIds()) {
      scheduledEventItem = scheduledEvents.getItem(eventItemID);

      // If the event has no event time set to it (the eventTimeID = -1) there is no need to save event time.
      // This means the event is a voluntary event.

      if (!"-1".equals(
          scheduledEventItem.getItemProperty(PaattiColumnNames.EVENT_EVENTTIME_eventTimeID).getValue().toString())) {

        eventTimeItemId = eventTimeContainer.addItem();
        eventTimeItem = eventTimeContainer.getItem(eventTimeItemId);

        for (String string : PaattiColumnNames.getColumnNames(PaattiTableNames.EVENTTIME)) {
          if (string.equals(PaattiColumnNames.EVENTTIME_eventTimeID)) {
            continue;
          }
          eventTimeProperty = eventTimeItem.getItemProperty(string);
          if (scheduledEventItem.getItemProperty(string).getValue() != null) {
            eventTimeProperty.setValue(scheduledEventItem.getItemProperty(string).getValue());
          }
        }

        eventTimeProperty = eventTimeItem.getItemProperty(PaattiColumnNames.EVENTTIME_rowStatus);
        eventTimeProperty.setValue(PaattiQueryDelegate.ROWSTATUS_ACTIVE);

        try {
          eventTimeContainer.commit();
        } catch (Exception ex) {
          ex.printStackTrace();
          return null;
        }
      }

      // Creates events ---------------------------------------------------
      Item eventItem;
```

```
      Object eventItemId;
      Property eventProperty;

      eventItemId = eventContainer.addItem();
      eventItem = eventContainer.getItem(eventItemId);

      for (String string : PaattiColumnNames.getColumnNames(PaattiTableNames.EVENT)) {
        if (string.equals(PaattiColumnNames.EVENT_eventID)) {
          continue;
        }
        eventProperty = eventItem.getItemProperty(string);
        if (scheduledEventItem.getItemProperty(string).getValue() != null) {
          eventProperty.setValue(scheduledEventItem.getItemProperty(string).getValue());
        }
      }

      eventProperty = eventItem.getItemProperty(PaattiColumnNames.EVENT_SCHEDULE_scheduleID);
      eventProperty.setValue(scheduleRowId);

      eventProperty = eventItem.getItemProperty(PaattiColumnNames.EVENT_EVENTTIME_eventTimeID);
      // If eventTimeRowId is null, set -1 to the eventTimeID (makes the event voluntary)
      eventProperty.setValue(eventTimeRowId != null ? eventTimeRowId : -1);

      eventProperty = eventItem.getItemProperty(PaattiColumnNames.EVENT_rowStatus);
      eventProperty.setValue(PaattiQueryDelegate.ROWSTATUS_ACTIVE);

      try {
        eventContainer.commit();
      } catch (Exception ex) {
        ex.printStackTrace();
        return null;
      }

      // Creates events tasks ------------------------------------------
      SQLContainer oldEventTaskContainer = getTaskContainer(
          scheduledEvents.getItem(eventItemID).getItemProperty(PaattiColumnNames.EVENT_eventID));
      SQLContainer newEventTaskContainer = getSQLContainerFromDBTable(PaattiTableNames.TASK,
          PaattiColumnNames.TASK_taskID, false);

      newEventTaskContainer.addListener(new QueryDelegate.RowIdChangeListener() {

        private static final long serialVersionUID = 1L;

        public void rowIdChange(QueryDelegate.RowIdChangeEvent event) {
          taskRowId = event.getNewRowId().getId()[0];
        }
```

```
    });

    Item oldTaskItem;
    Item newTaskItem;
    Object newTaskItemId;
    Property taskProperty;

    for (int i = 0; i < oldEventTaskContainer.size(); i++) {
      oldTaskItem = oldEventTaskContainer.getItem(oldEventTaskContainer.getIdByIndex(i));

      newTaskItemId = newEventTaskContainer.addItem();
      newTaskItem = newEventTaskContainer.getItem(newTaskItemId);

      for (String string : PaattiColumnNames.getColumnNames(PaattiTableNames.TASK)) {
        if (string.equals(PaattiColumnNames.TASK_taskID)) {
          continue;
        }
        taskProperty = newTaskItem.getItemProperty(string);
        if (oldTaskItem.getItemProperty(string).getValue() != null) {
          taskProperty.setValue(oldTaskItem.getItemProperty(string).getValue());
        }
      }

      taskProperty = newTaskItem.getItemProperty(PaattiColumnNames.TASK_EVENT_eventID);
      taskProperty.setValue(eventRowId);

      taskProperty = newTaskItem.getItemProperty(PaattiColumnNames.TASK_rowStatus);
      taskProperty.setValue(PaattiQueryDelegate.ROWSTATUS_ACTIVE);

      try {
        newEventTaskContainer.commit();
      } catch (Exception ex) {
        ex.printStackTrace();
        return null;
      }

      taskHashMap.put(oldTaskItem.getItemProperty(PaattiColumnNames.TASK_taskID).getValue(), taskRowId);
    }

    // Creates events tasks choices ----------------------------------
    SQLContainer oldTasksChoiceContainer = getChoiceContainer(
        scheduledEvents.getItem(eventItemID).getItemProperty(PaattiColumnNames.EVENT_eventID));
    SQLContainer newTasksChoiceContainer = getSQLContainerFromDBTable(PaattiTableNames.CHOICE,
        PaattiColumnNames.CHOICE_choiceID, false);

    Item newChoiceItem;
```

```
      Object newChoiceItemId;
      Item oldChoiceItem;
      Property choiceProperty;

      for (int i = 0; i < oldTasksChoiceContainer.size(); i++) {
        oldChoiceItem = oldTasksChoiceContainer.getItem(oldTasksChoiceContainer.getIdByIndex(i));

        newChoiceItemId = newTasksChoiceContainer.addItem();
        newChoiceItem = newTasksChoiceContainer.getItem(newChoiceItemId);

        for (String string : PaattiColumnNames.getColumnNames(PaattiTableNames.CHOICE)) {
          if (string.equals(PaattiColumnNames.CHOICE_choiceID)) {
            continue;
          }
          choiceProperty = newChoiceItem.getItemProperty(string);
          if (oldChoiceItem.getItemProperty(string).getValue() != null) {
            choiceProperty.setValue(oldChoiceItem.getItemProperty(string).getValue());
          }
        }

        choiceProperty = newChoiceItem.getItemProperty(PaattiColumnNames.CHOICE_TASK_contains_taskID);
        choiceProperty.setValue(
            taskHashMap.get(oldChoiceItem.getItemProperty(PaattiColumnNames.CHOICE_TASK_contains_taskID).getValue()));

        choiceProperty = newChoiceItem.getItemProperty(PaattiColumnNames.CHOICE_TASK_leadsto_taskID);
        choiceProperty.setValue(
            taskHashMap.get(oldChoiceItem.getItemProperty(PaattiColumnNames.CHOICE_TASK_leadsto_taskID).getValue()));

        choiceProperty = newChoiceItem.getItemProperty(PaattiColumnNames.CHOICE_rowStatus);
        choiceProperty.setValue(PaattiQueryDelegate.ROWSTATUS_ACTIVE);

        try {
          newTasksChoiceContainer.commit();
        } catch (Exception ex) {
          ex.printStackTrace();
          return null;
        }
      }
    }

    return scheduleRowId;
  }

  /**
   * {@inheritDoc }
   */
```

```
  public boolean addScheduleToGroup(Object groupID, Object scheduleID) {
    SQLContainer schedules = getSQLContainerFromDBTable(PaattiTableNames.SCHEDULE, PaattiColumnNames.SCHEDULE_scheduleID,
        true);
    Property property = schedules.getContainerProperty(scheduleID, PaattiColumnNames.SCHEDULE_USERGROUP_usergroupID);

    property.setValue(groupID);
    try {
      schedules.commit();
    } catch (Exception ex) {
      ex.printStackTrace();
      return false;
    }
    return true;
  }
}
```