SWE 4x	**Übung zu Softwareentwicklung mit modernen Plattformen 4**	**SS 2015, Übung 7**

Abgabetermin: SA in der KW 26

Gr. 1, E. Pitzer	**Name** Roman Lumetsberger	**Aufwand in h** 7
Gr. 2, F. Gruber-Leitner		
	Punkte _________ **Kurzzeichen Tutor / Übungsleiter**	______ / ______

Ausbaustufe 2: CaaS-DB (24 Punkte)

Nachdem der Restaurantbetreiber von der Präsentation Ihrer ersten Ausbaustufe restlos begeistert war, steht einem Ausbau bis hin zur Online-Plattform nichts mehr im Wege.

Im nächsten Schritt sollen Sie die Anforderung umsetzen, dass mehrere Mitarbeiter des Restaurantbetreibers die Benutzeroberfläche gleichzeitig benutzen können (im Restaurant und auch von zu Hause aus). Dafür ist es notwendig, die Benutzeroberfläche und die Datenverwaltung zu entkoppeln und die Kommunikation der beiden Komponenten über das Netzwerk durchzuführen.

In dieser Ausbaustufe sollen Sie weiters dafür sorgen, dass die Daten dauerhaft in einer Datenbank gespeichert werden.

Im Detail sollte Ihre Anwendung folgende Anforderungen erfüllen:

- Überlegen Sie sich eine geeignete Repräsentation für die Daten in Ihrem Programm. Im konkreten Anwendungsfall werden Sie dafür Klassen zur Speicherung der Benutzer-, Menü- und Bestelldaten benötigen. Diese Klassen werden häufig als Domänenmodell bezeichnet.
- Implementieren Sie einen RMI-Server, der die Daten des Menü-Bestellsystems zentral verwaltet. Stellen Sie sicher, dass die Serverkomponente parallel beliebig viele Clients mit Daten versorgen kann.
- Bauen Sie die in Übung 6 entwickelte JavaFX-basierte Benutzeroberfläche zu einem RMI-Client aus, der mit der Serverkomponente kommuniziert. Entwerfen Sie ein Datenmodell, das alle in CaaS anfallenden Daten abbildet. Berücksichtigen Sie nicht nur die Daten der Menüverwaltung sondern auch die Daten für die Online-Bestellung.
- Stellen Sie Klassen zum Zugriff auf die Datenbank auf Basis von JDBC zur Verfügung. Bereiten Sie auch bereits Methoden für das Hinzufügen und Laden von Bestellungen vor. Achten Sie darauf, dass die Datenbankzugriffschicht möglichst weitgehend von den anderen Komponenten der Anwendung getrennt ist. Stellen Sie dazu die gesamte Funktionalität der Datenbankzugriffsschicht über Interfaces zur Verfügung.
- Entwickeln Sie eine Testsuite, mit der Sie die Datenzugriffsschicht unabhängig von den anderen Systemkomponenten testen können.

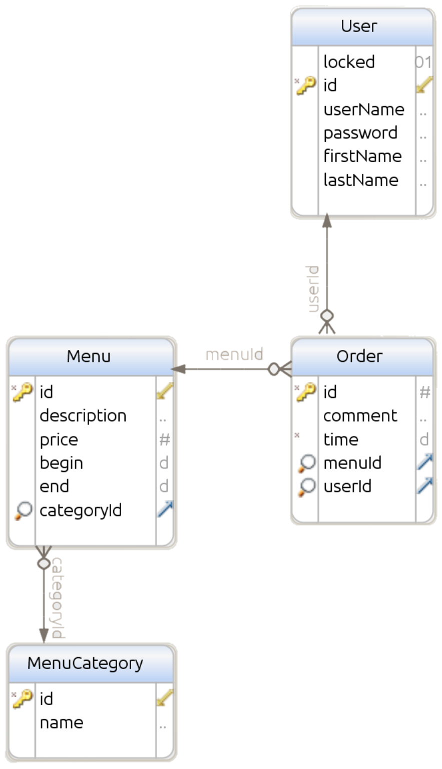
1 CaaS – Campina as a Service

1.1 Lösungsidee

Bei dieser Aufgabe muss das Datenbankschema erstellt und dann pro Tabelle ein *Dao* implementiert werden.
Weiters wird dann der aktuelle BusinessLayer per RMI verwendet. Die neue Implementierung kann die *Dao's* verwenden um die DB Operation durchzuführen.

1.1.1 Datenbank

Folgendes Schema wird für die Datenbank verwendet.



Das Datenbankschema liegt im Projektordner unter **dbscripts/create-tables-mysql.sql**

Achtung: Das Programm benötigt den MYSQL Treiber.

1.1.2 Dao's

Für jede Tabelle des Schemas wird ein Dao Interface erstellt, wobei dies von einem generischen Basisinterface abgeleitet werden kann.
Die JDBC Implementierung kann generisch über *Reflection* umgesetzt werden. Auch hier kann wieder eine generische Basisklasse verwendet werden.

1.1.3 BusinessLayer - Dao Implementierung

Der **Business-Layer** verwendet die **Dao's** um die benötigten Operationen auszuführen.

1.1.4 RMI

Die RMI Implementierung wurde der Einfachheit halber direkt in das Projekt integriert, sollte normalerweise aber in ein eigenes Projekt ausgelagert werden.
Da in Übung 06 bereits ein Business-Layer verwendet wurde, kann dieser nun einfach per RMI Interface bereitgestellt werden.

Der RMI Server Layer kann über die Klasse **at.lumetsnet.caas.rmi.RemoteServiceMain** gestartet werden.

1.2 Sourcecode - SQL

create-tables-mysql.sql

```sql
create schema if not exists CaasDb;
use CaasDb;
SET foreign_key_checks = 0;

drop table if exists User;
create table User (id int auto_increment primary key,
                     userName varchar(50), password varchar(50), firstName varchar(50),
                     lastName varchar(50), locked BOOL);
drop table if exists MenuCategory;
create table MenuCategory (id int auto_increment primary key,
                     name varchar(50));

drop table if exists Menu;
create table Menu (id int auto_increment primary key,
                    description varchar(50),
          price int,
          begin date,
          end date,
          categoryId int,
          FOREIGN KEY (categoryId) REFERENCES MenuCategory(id)
          ON DELETE CASCADE);

drop table if exists `Order`;
create table `Order` (id int auto_increment primary key,
                    comment varchar(100),
          time timestamp,
          menuId int,
          userId int,
          FOREIGN KEY (menuId) REFERENCES Menu(id),
        FOREIGN KEY (userId) REFERENCES User(id)
        ON DELETE CASCADE);

SET foreign_key_checks = 1;
```

1.3 Sourcecode - Java

Main.java

```java
package at.lumetsnet.caas;

import java.util.logging.LogManager;

import javafx.application.Application;
import javafx.stage.Stage;
import at.lumetsnet.caas.business.MenuService;
import at.lumetsnet.caas.business.OrderService;
import at.lumetsnet.caas.business.UserService;
import at.lumetsnet.caas.gui.MainWindow;
import at.lumetsnet.caas.model.User;

public class Main extends Application {

  @Override
  public void start(Stage stage) throws Exception {
    User user = new User();
    user.setFirstName("Roman");
    user.setLastName("Lumetsberger");

    MainWindow wnd = new MainWindow(stage, user);
    wnd.show();

  }

  public static void main(String[] args) {
    LogManager.getLogManager().reset();

    String rmiServer = "localhost:1099";
    if (args.length > 0) {
      rmiServer = args[0];
    }

    initializeServices(rmiServer);

    launch(args);

  }

  private static void initializeServices(String rmiServer) {
    if (!UserService.getInstance().initialize(rmiServer)) {
      System.out.println("Error initializing user service");
      System.exit(-1);
    }
    if (!OrderService.getInstance().initialize(rmiServer)) {
      System.out.println("Error initializing order service");
      System.exit(-1);
    }
    if (!MenuService.getInstance().initialize(rmiServer)) {
      System.out.println("Error initializing menu service");
      System.exit(-1);
    }

  }

}
```

business/MenuService.java

```java
package at.lumetsnet.caas.business;

import java.rmi.RemoteException;
import java.util.Collection;

import at.lumetsnet.caas.model.Menu;
import at.lumetsnet.caas.model.MenuCategory;
import at.lumetsnet.caas.rmi.interfaces.RemoteMenuService;

/***
 * Mock menu business logic class Used as singleton, should be replaced with
 * real logic class
 *
 * @author romanlum
 *
 */
public class MenuService extends RmiService<RemoteMenuService> {

  private static MenuService instance = null;

  private MenuService() {
    super("CaasMenuService");
  }

  /***
   * Singleton instance class
   *
   * @return
   */
  public static MenuService getInstance() {
    if (instance == null) {
      instance = new MenuService();
    }
    return instance;
  }
```

```java

/***
 * Fetches all categories
 *
 * @return
 */
public Collection<MenuCategory> getAllCategories() {
  try {
    return service.getAllCategories();
  } catch (RemoteException e) {
    e.printStackTrace();
  }
  return null;
}

/***
 * Deletes the category with the given id
 *
 * @param id
 */
public void deleteCategory(long id) {
  try {
    service.deleteCategory(id);
  } catch (RemoteException e) {
    e.printStackTrace();
  }
  ;
}

/***
 * Deletes the menu with the given id
 *
 * @param id
 */
public void deleteMenu(long id) {
  try {
    service.deleteMenu(id);
  } catch (RemoteException e) {
    e.printStackTrace();
  }
}

/***
 * Saves or adds the category
 *
 * @param data
 */
public void saveOrUpdateCategory(MenuCategory data) {
  try {
      service.saveOrUpdateCategory(data);
    } catch (RemoteException e) {
      e.printStackTrace();
    }
  }

  /***
   * Saves or adds the menu
   *
   * @param data
   */
  public void saveOrUpdateMenu(Menu data) {
    try {
      service.saveOrUpdateMenu(data);
    } catch (RemoteException e) {
      e.printStackTrace();
    }
  }

  /***
   * Fetches all menus
   *
   * @return
   */
  public Collection<Menu> getAllMenus() {
    try {
      return service.getAllMenus();
    } catch (RemoteException e) {
      e.printStackTrace();
    }
    return null;
  }

}
```

business/OrderService.java

```
package at.lumetsnet.caas.business;

import java.rmi.RemoteException;
import java.util.Collection;

import at.lumetsnet.caas.model.Order;
import at.lumetsnet.caas.rmi.interfaces.RemoteOrderService;

/***
 * Mock order business logic class Used as singleton, should be replaced with
 * real logic class
 *
```

```java
 * @author romanlum
 *
 */

public class OrderService extends RmiService<RemoteOrderService> {

  private static OrderService instance = null;

  private OrderService() {
    super("CaasOrderService");
  }

  /**
   * singleton instance
   */
  public static OrderService getInstance() {
    if (instance == null) {
      instance = new OrderService();
    }
    return instance;
  }

  /***
   * Fetches the orders of today
   *
   * @return
   */
  public Collection<Order> getTodaysOrders() {
    try {
      return service.getTodaysOrders();
    } catch (RemoteException e) {
      e.printStackTrace();
    }
    return null;
  }
}
```

business/RmiService.java

```java
package at.lumetsnet.caas.business;

import java.rmi.Naming;
import java.rmi.Remote;

/***
 * Base class for all services using RMI
 *
 * @author romanlum
 *
 */
public class RmiService<T extends Remote> {

  protected T service;
  protected String serviceHost;
  protected String name;

  /**
   * Create a service with the given name
   *
   * @param name
   */
  public RmiService(String name) {
    this.name = name;
  }

  /**
   * Initializes the service
   *
   * @param serviceHost
   * @return
   */
  public boolean initialize(String serviceHost) {
    try {
      service = (T) Naming.lookup("rmi://" + serviceHost + "/" + name);
    } catch (Exception e) {
      e.printStackTrace();
      return false;
    }
    return true;
  }

}
```

business/UserService.java

```java
package at.lumetsnet.caas.business;

import java.rmi.RemoteException;
import java.util.Collection;

import at.lumetsnet.caas.model.User;
import at.lumetsnet.caas.rmi.interfaces.RemoteUserService;

/***
 * Mock user business logic class Used as singleton, should be replaced with
 * real logic class
 *
 * @author romanlum
 *
 */
public class UserService extends RmiService<RemoteUserService> {

  private static UserService instance = null;

  private UserService() {
    super("CaasUserService");
  }

  /**
   * Singleton instance class
   */
  public static UserService getInstance() {
    if (instance == null) {
      instance = new UserService();
    }
    return instance;
  }

  /***
   * Fetches all users
   *
   * @return
   */
  public Collection<User> getAllUsers() {
    try {
      return service.getAllUsers();
    } catch (RemoteException e) {
      e.printStackTrace();
    }
    return null;
  }

  /***
   * delete the user
   *
   * @param id
   */
  public void deleteUser(long id) {
    try {
      service.deleteUser(id);
    } catch (RemoteException e) {
      e.printStackTrace();
    }
  }

  /***
   * Toggles the locked state of a user
```

```java
   *
   * @param id
   */
  public void toggleLockState(long id) {
    try {
      service.toggleLockState(id);
    } catch (RemoteException e) {
      e.printStackTrace();
    }
  }

  /***
   * Saves or adds the user
   *
   * @param userModel
   */
  public void saveOrUpdate(User userModel) {
    try {
      service.saveOrUpdate(userModel);
    } catch (RemoteException e) {
      e.printStackTrace();
    }
  }
}
```

dal/DaoUtil.java

```java
package at.lumetsnet.caas.dal;

import java.beans.BeanInfo;
import java.beans.Introspector;
import java.beans.PropertyDescriptor;
import java.sql.Connection;
import java.sql.Date;
import java.sql.PreparedStatement;
import java.sql.Statement;
import java.time.LocalDate;
import java.time.LocalDateTime;
import java.time.ZoneId;
import java.util.Arrays;
import java.util.Collection;

import at.lumetsnet.caas.model.Entity;

public class DaoUtil {

  /***
   * Generates a prepared statement for inserting an entity Sets all the
   * values
   *
   * @param con
   * @param entity
   * @param tableName
   * @param filter
   * @return
   */
  public static PreparedStatement generateInsertStatement(Connection con,
      Entity entity, String tableName, Collection<String> filter) {

    BeanInfo info;
    try {
      info = Introspector.getBeanInfo(entity.getClass());
      PropertyDescriptor[] pds = info.getPropertyDescriptors();

      StringBuilder builder = new StringBuilder();
      // build the insert string
      builder.append(String.format("insert into `%s` (", tableName));
      Arrays.stream(pds).forEach(
          x -> {
            if (!filter.stream().anyMatch(
                filterItem -> filterItem.equalsIgnoreCase(x
                    .getName()))) {
              builder.append(x.getName());
              builder.append(",");
            }
          });
      builder.delete(builder.length() - 1, builder.length()); // delete
                                  // last ","
      builder.append(") VALUES (");
      for (int i = 0; i < pds.length - filter.size(); i++) {
        builder.append("?,");
      }
      builder.delete(builder.length() - 1, builder.length()); // delete
                                  // last ","
      builder.append(")");

      // create the statement
      PreparedStatement stmt = con.prepareStatement(builder.toString(),
          Statement.RETURN_GENERATED_KEYS);
      int index = 1;
      for (int i = 0; i < pds.length; i++) {
        final String name = pds[i].getName();
        // filter properties which should not be persisted
        // lazy loading
        if (!filter.stream().anyMatch(
            filterItem -> filterItem.equalsIgnoreCase(name))) {
          Object data = pds[i].getReadMethod().invoke(entity);
          // Handle LocalDate
        if (data instanceof LocalDate) {
          LocalDate date = (LocalDate) data;
          stmt.setObject(
              index,
              Date.from(date.atStartOfDay()
                  .atZone(ZoneId.systemDefault())
                  .toInstant()));
        } else if (data instanceof LocalDateTime) {
          // handle loacalDateTime
          LocalDateTime date = (LocalDateTime) data;
          stmt.setObject(index, Date.from(date.atZone(
              ZoneId.systemDefault()).toInstant()));
        } else
          stmt.setObject(index, data);
        index++;
      }
    }
    return stmt;

  } catch (Exception e) {
    throw new DataAccessException(e.getMessage());
  }

}

/***
 * Creates an update statement for an entity
 *
 * @param con
 * @param entity
 * @param tableName
 * @param filter
 * @return
 */
public static PreparedStatement generateUpdateStatement(Connection con,
    Entity entity, String tableName, Collection<String> filter) {

  BeanInfo info;
  try {
    info = Introspector.getBeanInfo(entity.getClass());
    PropertyDescriptor[] pds = info.getPropertyDescriptors();

    StringBuilder builder = new StringBuilder();
    // build statement
    builder.append(String.format("UPDATE `%s` SET ", tableName));
    Arrays.stream(pds).forEach(
        x -> {
          if (!filter.stream().anyMatch(
              filterItem -> filterItem.equalsIgnoreCase(x
                    .getName()))) {
                builder.append(x.getName() + " = ?");
                builder.append(",");
              }
            });
        builder.delete(builder.length() - 1, builder.length()); // delete
                                            // last ","
        // append id
        builder.append(" WHERE id = ?");

        // create statement
        PreparedStatement stmt = con.prepareStatement(builder.toString(),
            Statement.RETURN_GENERATED_KEYS);
        int index = 1;
        for (int i = 0; i < pds.length; i++) {
          final String name = pds[i].getName();
          if (!filter.stream().anyMatch(
              filterItem -> filterItem.equalsIgnoreCase(name))) {
            Object data = pds[i].getReadMethod().invoke(entity);
            // Handle LocalDate
            if (data instanceof LocalDate) {
              LocalDate date = (LocalDate) data;
              stmt.setObject(
                  index,
                  Date.from(date.atStartOfDay()
                      .atZone(ZoneId.systemDefault())
                      .toInstant()));
            } else if (data instanceof LocalDateTime) {
              // Handle LocalDateTime
              LocalDateTime date = (LocalDateTime) data;
              stmt.setObject(index, Date.from(date.atZone(
                  ZoneId.systemDefault()).toInstant()));
            } else
              stmt.setObject(index, data);
            index++;
          }
        }
        stmt.setLong(index, entity.getId());
        return stmt;

      } catch (Exception e) {
        throw new DataAccessException(e.getMessage());
      }

    }

}
```

dal/DataAccessException.java

```java
package at.lumetsnet.caas.dal;

/***
 * Generic dao exception
 *
 * @author romanlum
 *
 */
@SuppressWarnings("serial")
public class DataAccessException extends RuntimeException {
  public DataAccessException(String msg) {
    super(msg);
  }
}
```

dal/GenericDao.java

```java
package at.lumetsnet.caas.dal;

import java.util.Collection;

import at.lumetsnet.caas.model.Entity;

public interface GenericDao<T extends Entity> {

  T get(long id);

  Collection<T> getAll();

  void saveOrUpdate(T entity);

  void delete(long id);
}
```

dal/GenericJdbcDao.java

```java
package at.lumetsnet.caas.dal;

import java.beans.BeanInfo;
import java.beans.IntrospectionException;
import java.beans.Introspector;
import java.beans.PropertyDescriptor;
import java.io.Closeable;
import java.io.IOException;
import java.sql.Connection;
import java.sql.Date;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Timestamp;
import java.sql.Types;
import java.time.Instant;
import java.time.LocalDateTime;
import java.time.ZoneId;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Collection;

import at.lumetsnet.caas.model.Entity;

/***
 * Generic dao for JDBC
 *
 * @author romanlum
 *
 * @param <T>
 */
public abstract class GenericJdbcDao<T extends Entity> implements Closeable,
    GenericDao<T> {

  protected Connection con;
  protected String conString;
  protected String userName;
  protected String password;
  protected String tableName;
  protected Class<T> entityClass;

  public GenericJdbcDao(Class<T> entityClass, String tableName,
      String conString, String userName, String password) {
    this.conString = conString;
    this.userName = userName;
    this.password = password;
    this.tableName = tableName;
    this.entityClass = entityClass;

  }

  /***
   * Gets and opens the connection
   *
   * @return
   * @throws DataAccessException
   */
  public Connection getConnection() throws DataAccessException {
    try {
      if (con == null) {
        con = DriverManager
            .getConnection(conString, userName, password);
      }
      return con;
    } catch (SQLException ex) {
      throw new DataAccessException(ex.getMessage());
    }
  }

  /***
   * Closes the connection
   */
  @Override
  public void close() throws IOException {
    if (con != null) {
      try {
        con.close();
        con = null;
      } catch (SQLException e) {
        throw new DataAccessException(e.getMessage());
      }
    }
  }

  /***
   * Inserts or updates an entity based on the id ID == -1 means new entity
   * Auto generated id is set after inserting
   */
  @Override
  public void saveOrUpdate(T entity) {
    if (entity.getId() == -1) {
      try (PreparedStatement stmt = DaoUtil.generateInsertStatement(
          getConnection(), entity, tableName, getPropertyFilter())) {
        stmt.executeUpdate();
        try (ResultSet rs = stmt.getGeneratedKeys()) {
          if (rs != null && rs.next()) {
            entity.setId(rs.getInt(1));
          } else {
            throw new DataAccessException(
                "Autogenerated keys are not supported by db");
          }
        }
      } catch (Exception ex) {
        throw new DataAccessException(ex.getMessage());
      }
    } else {
      try (PreparedStatement stmt = DaoUtil.generateUpdateStatement(
          getConnection(), entity, tableName, getPropertyFilter())) {
```

```
      stmt.executeUpdate();
    } catch (Exception ex) {
      throw new DataAccessException(ex.getMessage());
    }

  }
}

/***
 * Gets a list of entities by using a where statement
 *
 * @param query
 * @param args
 * @return
 * @throws DataAccessException
 */
protected Collection<T> getFromWhere(String query, Object... args)
    throws DataAccessException {
  Collection<T> c = new ArrayList<T>();

  BeanInfo info;
  try {
    info = Introspector.getBeanInfo(entityClass);
  } catch (IntrospectionException e) {
    throw new DataAccessException(e.getMessage());
  }

  PropertyDescriptor[] pds = info.getPropertyDescriptors();

  // build statement
  String sql = String.format("Select * from ‘%s‘ %s", tableName, query);

  // / create statement
  try (PreparedStatement pstmt = getConnection().prepareStatement(sql)) {
    for (int i = 0; i < args.length; i++) {
      pstmt.setObject(i + 1, args[i]);
    }

    try (ResultSet rs = pstmt.executeQuery()) {
      while (rs.next()) {
        // create a new instance
        T entity = entityClass.newInstance();

        // fetch the meta data
        java.sql.ResultSetMetaData metaData = rs.getMetaData();
        for (int i = 1; i <= metaData.getColumnCount(); i++) {
          Object data = rs.getObject(i);

          // skip null values because they are already
          // null in the new object
          if (data == null)
            continue;

          String columnName = metaData.getColumnName(i);
          PropertyDescriptor descriptor = Arrays.stream(pds)
              .filter(x -> x.getName().equals(columnName))
              .findFirst().orElse(null);
          if (descriptor != null) {
            if (metaData.getColumnType(i) == Types.DATE) {
              // Date handling
              Instant instant = Instant
                  .ofEpochMilli(((Date) data).getTime());
              data = LocalDateTime.ofInstant(instant,
                  ZoneId.systemDefault()).toLocalDate();
            } else if (metaData.getColumnType(i) == Types.TIMESTAMP) {
              // Time handling
              Instant instant = Instant
                  .ofEpochMilli(((Timestamp) data)
                      .getTime());
              data = LocalDateTime.ofInstant(instant,
                  ZoneId.systemDefault());
            }
            // set the property
            descriptor.getWriteMethod().invoke(entity, data);
          }

        }
        c.add(entity);
      }
    }
    return c;

  } catch (Exception ex) {
    throw new DataAccessException(ex.getMessage());
  }
}

/***
 * Gets the entity by id
 */
@Override
public T get(long id) {
  return getFromWhere("where id = ?", id).stream().findFirst()
      .orElse(null);
}

/***
 * Gets all entites
```

```java
   */
  @Override
  public Collection<T> getAll() {
    return getFromWhere("");
  }

  /***
   * Deletes an entity
   */
  @Override
  public void delete(long id) {
    try (PreparedStatement pstmt = getConnection().prepareStatement(
        String.format("DELETE FROM `%s` where id = ?", tableName))) {
      pstmt.setLong(1, id);
      pstmt.executeUpdate();
    } catch (SQLException ex) {
      throw new DataAccessException(ex.getMessage());
    }

  }

  /***
   * Property filter used for insert and update statements These properties
   * are skipped
   *
   * @return
   */
  public Collection<String> getPropertyFilter() {
    ArrayList<String> filter = new ArrayList<>();
    filter.add("class");
    filter.add("id");
    return filter;
  }

}
```

dal/MenuCategoryDao.java

```java
package at.lumetsnet.caas.dal;

import at.lumetsnet.caas.model.MenuCategory;

/***
 * MenuCategory dao
 *
 * @author romanlum
 *
 */
public interface MenuCategoryDao extends GenericDao<MenuCategory> {

}
```

dal/MenuCategoryDaoJdbc.java

```java
package at.lumetsnet.caas.dal;

import at.lumetsnet.caas.model.MenuCategory;

/***
 * MenuCategory dao jdbc impl
 *
 * @author romanlum
 *
 */
public class MenuCategoryDaoJdbc extends GenericJdbcDao<MenuCategory> implements
    MenuCategoryDao {

  public MenuCategoryDaoJdbc(String conString, String userName,
      String password) {
    super(MenuCategory.class, "MenuCategory", conString, userName, password);
  }

}
```

dal/MenuDao.java

```java
package at.lumetsnet.caas.dal;

import at.lumetsnet.caas.model.Menu;

/***
 * Menu dao
 *
 * @author romanlum
 *
 */

public interface MenuDao extends GenericDao<Menu> {

}
```

dal/MenuDaoJdbc.java

```java
package at.lumetsnet.caas.dal;

import java.util.Collection;

import at.lumetsnet.caas.model.Menu;

/***
 * Menu dao jdbc impl
 *
 * @author romanlum
 *
 */

public class MenuDaoJdbc extends GenericJdbcDao<Menu> implements MenuDao {

  public MenuDaoJdbc(String conString, String userName, String password) {
    super(Menu.class, "Menu", conString, userName, password);
  }

  @Override
  public Collection<String> getPropertyFilter() {
    Collection<String> result = super.getPropertyFilter();
    result.add("category");
    return result;
  }

}
```

dal/OrderDao.java

```java
package at.lumetsnet.caas.dal;

import java.time.LocalDate;
import java.util.Collection;

import at.lumetsnet.caas.model.Order;

/***
 * Order dao
 *
 * @author romanlum
 *
 */

public interface OrderDao extends GenericDao<Order> {

  /***
   * Gets the orders by date
   *
   * @param date
   * @return
   */
  Collection<Order> getOrdersByDate(LocalDate date);
}
```

dal/OrderDaoJdbc.java

```
package at.lumetsnet.caas.dal;

import java.sql.Date;
import java.time.LocalDate;
import java.util.Collection;

import at.lumetsnet.caas.model.Order;

/***
 * Order dao jdbc impl
 *
 * @author romanlum
 *
 */
public class OrderDaoJdbc extends GenericJdbcDao<Order> implements OrderDao {

  public OrderDaoJdbc(String conString, String userName, String password) {
    super(Order.class, "Order", conString, userName, password);
  }

  /***
   * Gets the orders by date
   */
  public Collection<Order> getOrdersByDate(LocalDate date) {
    return getFromWhere("WHERE DATE(time) = ?", Date.valueOf(date));
  }

  /***
   * Sets property filters to filter Referenced entities
   */
  @Override
  public Collection<String> getPropertyFilter() {
    Collection<String> result = super.getPropertyFilter();
    result.add("user");
    result.add("menu");
    return result;
  }

}
```

dal/UserDao.java

```java
package at.lumetsnet.caas.dal;

import at.lumetsnet.caas.model.User;

/***
 * User dao
 *
 * @author romanlum
 *
 */
public interface UserDao extends GenericDao<User> {

}
```

dal/UserDaoJdbc.java

```java
package at.lumetsnet.caas.dal;

import at.lumetsnet.caas.model.User;

/***
 * User dao jdbc impl
 *
 * @author romanlum
 *
 */
public class UserDaoJdbc extends GenericJdbcDao<User> implements UserDao {

  public UserDaoJdbc(String conString, String userName, String password) {
    super(User.class, "User", conString, userName, password);
  }

}
```

gui/ActionTableCell.java

```java
package at.lumetsnet.caas.gui;

import java.util.function.Consumer;

import javafx.scene.control.Button;
import javafx.scene.control.TableCell;
import javafx.scene.layout.HBox;

/***
 * TableCell used for displaying edit / delete buttons inside a TableView
 *
 * @author romanlum
 *
 * @param <S>
 * @param <T>
 */
public class ActionTableCell<S, T> extends TableCell<S, T> {
  protected HBox box = new HBox();
  protected Button editButton = new Button("Bearbeiten");
  protected Button deleteButton = new Button("Löschen");

  protected Consumer<T> editAction;
  protected Consumer<T> deleteAction;

  public ActionTableCell(Consumer<T> editAction, Consumer<T> deleteAction) {
    this.editAction = editAction;
    this.deleteAction = deleteAction;

    box.getStyleClass().add("table-command-container");
    editButton.getStyleClass()
        .addAll("table-command", "table-command-edit");
    deleteButton.getStyleClass().addAll("table-command",
        "table-command-delete");
    box.getChildren().addAll(editButton, deleteButton);
  }

  @Override
  protected void updateItem(T entity, boolean empty) {

    if (entity != null) {
      editButton.setOnAction(x -> {
        if (editAction != null)
          editAction.accept(entity);

      });
      deleteButton.setOnAction(x -> {
        if (deleteAction != null)
          deleteAction.accept(entity);
      });
      setGraphic(box);
    } else {
      setGraphic(null);
    }
  }
}
```

gui/AmountTableCell.java

```java
package at.lumetsnet.caas.gui;

import javafx.scene.control.TableCell;
```

```java

/***
 * TableCell used for formatting an amount inside a TableView Formats the value
 * given in LOWEST currency unit to normal EUR representation
 *
 * @author romanlum
 *
 * @param <S>
 */
public class AmountTableCell<S> extends TableCell<S, Number> {

  @Override
  protected void updateItem(Number entity, boolean empty) {

    if (entity != null) {
      // formats the value given in lowest currency unit
      // to EUR
      setText(((double) ((long) entity) / 100) + " EUR");
    } else {
      setText("");
    }
  }
}
```

gui/DateTableCell.java

```java
package at.lumetsnet.caas.gui;

import java.time.LocalDate;
import java.time.format.DateTimeFormatter;

import javafx.scene.control.TableCell;

/***
 * TableCell used for displaying LocalDate values inside a TableView
 *
 * @author romanlum
 *
 * @param <S>
 */
public class DateTableCell<S> extends TableCell<S, LocalDate> {

  @Override
  protected void updateItem(LocalDate entity, boolean empty) {

    if (entity != null) {
      setText(entity.format(DateTimeFormatter.ISO_LOCAL_DATE));

    } else {
      setText("");
    }
  }
}
```

gui/MainWindow.java

```
package at.lumetsnet.caas.gui;

import java.util.HashMap;

import javafx.scene.Node;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.scene.layout.BorderPane;
import javafx.scene.layout.HBox;
import javafx.scene.layout.Pane;
import javafx.scene.layout.Priority;
import javafx.scene.layout.VBox;
import javafx.stage.Stage;
import javafx.stage.Window;
import at.lumetsnet.caas.gui.pages.ManageMenusPage;
import at.lumetsnet.caas.gui.pages.ManageUsersPage;
import at.lumetsnet.caas.gui.pages.OrdersPage;
import at.lumetsnet.caas.gui.pages.Showable;
import at.lumetsnet.caas.model.User;

/***
 * Main windows class Creates all pages and manages page switching
 *
 * @author romanlum
 *
 */
public class MainWindow {

  private enum VIEW_TYPE {
    ADMIN_ORDERS, ADMIN_USERS, ADMIN_MENUS
  }

  private User user;
  private Stage stage = new Stage();
  private HashMap<VIEW_TYPE, Pane> pages;
  private BorderPane container;

  private Button adminOrdersViewButton;
  private Button adminUsersViewButton;
  private Button adminMenusViewButton;

```

```java
public MainWindow(Window owner, User user) {
  this.user = user;
  container = new BorderPane();
  container.getStyleClass().add("window-container");
  container.setTop(createTopMenu());

  pages = createPages();

  Scene scene = new Scene(container);
  scene.getStylesheets().add(
      getClass().getResource("css/main-window.css").toExternalForm());

  stage.initOwner(owner);
  stage.setScene(scene);
  stage.setTitle("CaaS");
  // start with 1024/768
  stage.setWidth(1024);
  stage.setHeight(768);
  switchView(VIEW_TYPE.ADMIN_ORDERS);

}

private HashMap<VIEW_TYPE, Pane> createPages() {
  HashMap<VIEW_TYPE, Pane> pages = new HashMap<>();
  pages.put(VIEW_TYPE.ADMIN_ORDERS, new OrdersPage());
  pages.put(VIEW_TYPE.ADMIN_USERS, new ManageUsersPage());
  pages.put(VIEW_TYPE.ADMIN_MENUS, new ManageMenusPage());

  return pages;
}

public void show() {

  stage.show();
}

private Node createTopMenu() {
  HBox topContainer = new HBox();
  topContainer.getStyleClass().add("nav-container");

  Label iconLabel = new Label();
  iconLabel.setId("nav-top-icon");
  VBox descBox = new VBox();

  Label userNameLabel = new Label();
  userNameLabel.getStyleClass().add("nav-username");
  userNameLabel.setText(user.getFirstName() + " " + user.getLastName());

  Label descLabel = new Label();
```

```
    descLabel.getStyleClass().add("nav-role");
    descLabel.setText("Administrator");

    descBox.getChildren().add(userNameLabel);
    descBox.getChildren().add(descLabel);

    HBox commandBox = new HBox();
    commandBox.getStyleClass().add("nav-command-container");

    adminOrdersViewButton = new Button();
    adminOrdersViewButton.getStyleClass().add("nav-command");
    adminOrdersViewButton.setText("Bestellungen");
    adminOrdersViewButton
        .setOnAction(x -> switchView(VIEW_TYPE.ADMIN_ORDERS));
    commandBox.getChildren().add(adminOrdersViewButton);

    adminUsersViewButton = new Button();
    adminUsersViewButton.getStyleClass().add("nav-command");
    adminUsersViewButton.setText("Benutzerverwaltung");
    adminUsersViewButton
        .setOnAction(x -> switchView(VIEW_TYPE.ADMIN_USERS));
    commandBox.getChildren().add(adminUsersViewButton);

    adminMenusViewButton = new Button();
    adminMenusViewButton.getStyleClass().add("nav-command");
    adminMenusViewButton.setText("Speisekarte verwalten");
    adminMenusViewButton
        .setOnAction(x -> switchView(VIEW_TYPE.ADMIN_MENUS));
    commandBox.getChildren().add(adminMenusViewButton);

    HBox.setHgrow(commandBox, Priority.ALWAYS);

    topContainer.getChildren().add(iconLabel);
    topContainer.getChildren().add(descBox);
    topContainer.getChildren().add(commandBox);
    return topContainer;
  }

  /***
   * Switches the view
   *
   * @param type
   */
  private void switchView(VIEW_TYPE type) {
    adminOrdersViewButton.getStyleClass().remove("nav-command-selected");
    adminUsersViewButton.getStyleClass().remove("nav-command-selected");
    adminMenusViewButton.getStyleClass().remove("nav-command-selected");

    Button selected = null;
    switch (type) {
    case ADMIN_USERS:
      selected = adminUsersViewButton;

      break;
    case ADMIN_ORDERS:
      selected = adminOrdersViewButton;
      break;
    case ADMIN_MENUS:
      selected = adminMenusViewButton;
      break;
    }
    if (selected != null)
      selected.getStyleClass().add("nav-command-selected");

    Pane page = pages.get(type);
    if (page instanceof Showable)
      ((Showable) page).show();
    container.setCenter(page);

  }

}
```

gui/ManageUserActionCell.java

```
package at.lumetsnet.caas.gui;

import java.util.function.Consumer;

import javafx.scene.control.Button;
import at.lumetsnet.caas.viewmodel.UserViewModel;

/***
 * TableCell used for displaying additional lock/unlock button inside the
 * TableView
 *
 * @author romanlum
 *
 * @param <S>
 * @param <T>
 */
public class ManageUserActionCell<S, T> extends ActionTableCell<S, T> {

  protected Button lockButton = new Button("Sperren");
  protected Consumer<T> lockAction;

  public ManageUserActionCell(Consumer<T> editAction,
      Consumer<T> deleteAction, Consumer<T> lockAction) {
```

```java
    super(editAction, deleteAction);
    this.lockAction = lockAction;
    box.getChildren().addAll(lockButton);
  }

  @Override
  protected void updateItem(T entity, boolean empty) {

    if (entity != null) {
      lockButton.setOnAction(x -> {
        if (lockAction != null)
          lockAction.accept(entity);

      });
      if (getTableRow() != null && getTableRow().getItem() != null) {
        Object model = getTableRow().getItem();
        if (model instanceof UserViewModel) {
          lockButton.getStyleClass().removeIf(
              x -> x.startsWith("table-command"));
          if (((UserViewModel) model).getLockedProperty().get()) {
            lockButton.setText("Entsperren");
            lockButton.getStyleClass().addAll("table-command",
                "table-command-unlock");
          } else {
            lockButton.setText("Sperren");
            lockButton.getStyleClass().addAll("table-command",
                "table-command-lock");
          }
        }
      }

    }
    super.updateItem(entity, empty);
  }

}
```

gui/Util.java

```java
package at.lumetsnet.caas.gui;

import java.util.ArrayList;
import java.util.Collection;

import javafx.beans.property.ObjectProperty;
import javafx.beans.property.StringProperty;
import javafx.beans.value.ChangeListener;
import javafx.beans.value.ObservableValue;
import javafx.collections.ObservableList;
import javafx.scene.Node;
import javafx.scene.control.ComboBox;
import javafx.scene.control.Label;
import javafx.scene.control.PasswordField;
import javafx.scene.control.TextField;

/***
 * Common gui utils
 *
 * @author romanlum
 *
 */
public class Util {

  /***
   * Creates a form label
   *
   * @param text
   * @return
   */
  public static Node getFormLabel(String text) {
    Label lab = new Label();
    lab.setText(text);
    lab.getStyleClass().add("form-label");
    return lab;
  }

  /***
   * Creates a text field form and label
   *
   * @param text
   * @param property
   * @return
   */
  public static Collection<Node> getTextFieldForm(String text,
      StringProperty property) {

    Collection<Node> nodes = new ArrayList<Node>();
    nodes.add(getFormLabel(text));

    TextField field = new TextField();
    field.getStyleClass().add("form-textfield");
    field.textProperty().bindBidirectional(property);
    nodes.add(field);
    return nodes;
  }

  /***
   * Creates a password form field and label
   *
   * @param text
   * @param property
   * @return
   */
  public static Collection<Node> getPasswordFieldForm(String text,
      StringProperty property) {

    Collection<Node> nodes = new ArrayList<Node>();
    nodes.add(getFormLabel(text));

    PasswordField field = new PasswordField();
    field.getStyleClass().add("form-textfield");
    field.textProperty().bindBidirectional(property);
    nodes.add(field);
    return nodes;
  }

  /***
   * Creates a combo box form and label
   *
   * @param text
   * @param property
   * @param data
   * @return
   */
  public static <T> Collection<Node> getComboboxForm(String text,
      ObjectProperty<T> property, ObservableList<T> data) {

    Collection<Node> nodes = new ArrayList<Node>();
    nodes.add(getFormLabel(text));

    ComboBox<T> field = new ComboBox<>();
    field.getStyleClass().add("form-combobox");
    field.setItems(data);
    field.getSelectionModel().selectedItemProperty()
        .addListener(new ChangeListener<T>() {
          @Override
          public void changed(ObservableValue<? extends T> item,
              T arg1, T arg2) {
            property.setValue(item.getValue());

          }
        });
    field.selectionModelProperty().get().select(property.get());
    nodes.add(field);
    return nodes;
  }

```

```
}
```

gui/dialogs/Dialog.java

```java
package at.lumetsnet.caas.gui.dialogs;

import javafx.scene.Node;
import javafx.scene.Scene;
import javafx.scene.control.Label;
import javafx.scene.layout.BorderPane;
import javafx.scene.layout.HBox;
import javafx.scene.layout.Pane;
import javafx.scene.layout.VBox;
import javafx.stage.Modality;
import javafx.stage.Stage;
import javafx.stage.StageStyle;
import javafx.stage.Window;

/***
 * Dialog base class used for all dialogs
 *
 * @author romanlum
 *
 */
public abstract class Dialog {

  protected Stage dialogStage = new Stage();

  protected final void createGui(Window owner, String title,
      String description) {
    BorderPane layout = new BorderPane();
    layout.getStyleClass().add("dialog-container");
    layout.setTop(createTopPane(title, description));
    layout.setCenter(createContentPane());
    layout.setBottom(createBottomPane());

    Scene dialogScene = new Scene(layout);
    dialogScene.getStylesheets().add(
        getClass().getResource("../css/dialog.css").toExternalForm());

    dialogStage.initOwner(owner);
    dialogStage.setScene(dialogScene);
    dialogStage.setTitle(title);
    dialogStage.initModality(Modality.WINDOW_MODAL);
    dialogStage.initStyle(StageStyle.UTILITY);
    dialogStage.setResizable(false);
  }

  protected Node createTopPane(String title, String description) {
    HBox box = new HBox();
    box.getStyleClass().add("dialog-top-container");

    Label iconLabel = new Label();
    iconLabel.setId("dialog-icon");
    VBox descBox = new VBox();

    Label titleLabel = new Label();
    titleLabel.getStyleClass().add("dialog-title");
    titleLabel.setText(title);

    Label descLabel = new Label();
    descLabel.getStyleClass().add("dialog-description");
    descLabel.setText(description);

    descBox.getChildren().add(titleLabel);
    descBox.getChildren().add(descLabel);

    box.getChildren().add(iconLabel);
    box.getChildren().add(descBox);
    return box;
  }

  protected Pane createBottomPane() {
    HBox box = new HBox();
    box.getStyleClass().add("dialog-bottom-container");
    return box;

  }

  /***
   * Content used for the dialog
   *
   * @return
   */
  protected abstract Node createContentPane();

}
```

gui/dialogs/ErrorDialog.java

```java
package at.lumetsnet.caas.gui.dialogs;

import javafx.scene.Node;
import javafx.scene.control.Button;
import javafx.scene.layout.Pane;
import javafx.stage.Window;

/***
```

```java
 * Own dialog class used because javafx does not have dialogs before update 40
 * :(
 *
 * @author romanlum
 *
 */
public class ErrorDialog extends Dialog {

  public ErrorDialog(Window owner) {

    createGui(owner, "Fehler", "Bitte korrigieren Sie Ihre Eingaben.");

    dialogStage
        .getScene()
        .getStylesheets()
        .add(getClass().getResource("../css/error-dialog.css")
            .toExternalForm());
  }

  @Override
  protected Pane createBottomPane() {
    Pane pane = super.createBottomPane();
    Button button = new Button();
    button.setText("OK");
    button.getStyleClass().add("command-button");
    button.setOnAction(x -> okCommand());
    pane.getChildren().add(button);
    return pane;
  }

  private void okCommand() {
    dialogStage.close();
  }

  @Override
  protected Node createContentPane() {
    return null;
  }

  public boolean show() {
    dialogStage.showAndWait();
    return true;
  }

  public static boolean show(Window owner) {
    ErrorDialog dialog = new ErrorDialog(owner);
    return dialog.show();

  }
}
```

gui/dialogs/ManageEntityDialog.java

```
package at.lumetsnet.caas.gui.dialogs;

import javafx.beans.property.BooleanProperty;
import javafx.beans.property.SimpleBooleanProperty;
import javafx.scene.control.Button;
import javafx.scene.layout.Pane;
import at.lumetsnet.caas.viewmodel.Validatable;

/***
 * Base dialog used for editing a viewmodel entity
 *
 * @author romanlum
 *
 * @param <T>
 */
public abstract class ManageEntityDialog<T> extends Dialog {

  protected boolean canceled = false;
  protected T viewModel;
  protected BooleanProperty validationErrorProperty = new SimpleBooleanProperty();

  /***
   * Shows the dialog and waits for exit returns if the action was canceled
   *
   * @return
   */
  public boolean show() {
    dialogStage.showAndWait();
    return canceled;
  }

  /***
   * Save action
   */
  protected abstract void saveCommand();

  /***
   * Cancel action Default: closes the dialog and sets canceled flag
   */
  protected void cancelCommand() {
    canceled = true;
    dialogStage.close();

  }

  /***
   * Validates the viewmodel if it is Validatable Shows error dialog if
   * validation fails
   *
   * @return true if the viewmodel is not validatable true on successfull
   *         validation otherwise false
   */
  protected boolean validate() {
    if (!(viewModel instanceof Validatable)) {
      return true;
    }

    boolean result = ((Validatable) viewModel).validate();
    if (!result) {
      ErrorDialog.show(dialogStage);
    }
    return result;
  }

  @Override
  protected Pane createBottomPane() {
    Pane pane = super.createBottomPane();
    Button button = new Button();
    button.setText("Speichern");
    button.getStyleClass().add("command-button");
    button.setOnAction(x -> saveCommand());
    pane.getChildren().add(button);

    button = new Button();
    button.setText("Abbrechen");
    button.getStyleClass().add("command-button");
    button.setOnAction(x -> cancelCommand());
    pane.getChildren().add(button);

    return pane;
  }

}
```

gui/dialogs/ManageMenuCategoryDialog.java

```
package at.lumetsnet.caas.gui.dialogs;

import javafx.scene.Node;
import javafx.scene.layout.VBox;
import javafx.stage.Window;
import at.lumetsnet.caas.business.MenuService;
import at.lumetsnet.caas.gui.Util;
import at.lumetsnet.caas.model.MenuCategory;
```

```java
import at.lumetsnet.caas.viewmodel.MenuCategoryViewModel;

/***
 * Edit/Add dialog for MenuCategories
 *
 * @author romanlum
 *
 */
public class ManageMenuCategoryDialog extends
    ManageEntityDialog<MenuCategoryViewModel> {

  public ManageMenuCategoryDialog(Window owner, MenuCategory data) {
    viewModel = new MenuCategoryViewModel(data);
    createGui(owner, "Bereich verwalten",
        "Bitte geben Sie die Daten des Bereichs ein.");

    dialogStage
        .getScene()
        .getStylesheets()
        .add(getClass().getResource(
            "../css/manage-menu-category-dialog.css")
            .toExternalForm());
  }

  @Override
  protected Node createContentPane() {

    VBox box = new VBox();
    box.getStyleClass().add("form-container");
    box.getChildren().addAll(
        Util.getTextFieldForm("Name", viewModel.getNameProperty()));

    return box;
  }

  @Override
  protected void saveCommand() {
    if (validate()) {
      // Saves the entity
      MenuService.getInstance().saveOrUpdateCategory(
          viewModel.toCategoryModel());
      dialogStage.close();
    }

  }

  /***
   * Static method for showing the dialog
   *
   * @param owner
   * @param model
   * @return
   */
  public static boolean show(Window owner, MenuCategory model) {
    ManageMenuCategoryDialog dialog = new ManageMenuCategoryDialog(owner,
        model);
    return dialog.show();

  }
}
```

gui/dialogs/ManageMenuDialog.java

```java
package at.lumetsnet.caas.gui.dialogs;

import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import javafx.scene.Node;
import javafx.scene.control.DatePicker;
import javafx.scene.layout.HBox;
import javafx.scene.layout.VBox;
import javafx.stage.Window;
import at.lumetsnet.caas.business.MenuService;
import at.lumetsnet.caas.gui.Util;
import at.lumetsnet.caas.model.Menu;
import at.lumetsnet.caas.viewmodel.MenuCategoryViewModel;
import at.lumetsnet.caas.viewmodel.MenuViewModel;

/***
 * Edit/Add dialog for menus
 *
 * @author romanlum
 *
 */
public class ManageMenuDialog extends ManageEntityDialog<MenuViewModel> {

  public ManageMenuDialog(Window owner, Menu menu) {
    viewModel = new MenuViewModel(menu);
    createGui(owner, "Hauptspeise verwalten",
        "Bitte geben Sie die Daten der Hauptspeise ein.");

    dialogStage
        .getScene()
        .getStylesheets()
        .add(getClass().getResource("../css/manage-menu-dialog.css")
            .toExternalForm());
  }

```

```java
@Override
protected Node createContentPane() {

  VBox box = new VBox();
  box.getStyleClass().add("form-container");
  box.getChildren().addAll(
      Util.getTextFieldForm("Beschreibung",
          viewModel.getDescriptionProperty()));
  box.getChildren().addAll(
      Util.getComboboxForm("Bereich",
          viewModel.getCategoryProperty(), getMenuCategories()));

  box.getChildren().addAll(
      Util.getTextFieldForm("Preis",
          viewModel.getPriceAsStringProperty()));
  DatePicker picker = new DatePicker();
  box.getChildren().add(Util.getFormLabel("Zeitraum"));

  HBox rangeBox = new HBox();
  rangeBox.setSpacing(10);

  picker.valueProperty().bindBidirectional(viewModel.getBeginProperty());
  picker.getStyleClass().add("form-datepicker");
  rangeBox.getChildren().add(picker);

  picker = new DatePicker();
  picker.getStyleClass().add("form-datepicker");
  picker.valueProperty().bindBidirectional(viewModel.getEndProperty());
  rangeBox.getChildren().add(picker);
  box.getChildren().add(rangeBox);
  return box;
}

private ObservableList<MenuCategoryViewModel> getMenuCategories() {
  ObservableList<MenuCategoryViewModel> data = FXCollections
      .observableArrayList();
  MenuService.getInstance().getAllCategories()
      .forEach(x -> data.add(new MenuCategoryViewModel(x)));
  return data;
}

@Override
protected void saveCommand() {
  if (validate()) {
    // Saves the entity
    MenuService.getInstance().saveOrUpdateMenu(viewModel.toMenuModel());
    dialogStage.close();
  }

  }

  /***
   * Static method for showing the dialog
   *
   * @param owner
   * @param model
   * @return
   */
  public static boolean show(Window owner, Menu model) {
    ManageMenuDialog dialog = new ManageMenuDialog(owner, model);
    return dialog.show();

  }
}
```

gui/dialogs/ManageUserDialog.java

```java
package at.lumetsnet.caas.gui.dialogs;

import javafx.scene.Node;
import javafx.scene.layout.VBox;
import javafx.stage.Window;
import at.lumetsnet.caas.business.UserService;
import at.lumetsnet.caas.gui.Util;
import at.lumetsnet.caas.model.User;
import at.lumetsnet.caas.viewmodel.UserViewModel;

public class ManageUserDialog extends ManageEntityDialog<UserViewModel> {

  public ManageUserDialog(Window owner, User user) {
    viewModel = new UserViewModel(user);
    createGui(owner, "User verwalten",
        "Bitte geben Sie die Daten des Benutzers ein.");

    dialogStage
        .getScene()
        .getStylesheets()
        .add(getClass().getResource("../css/manage-user-dialog.css")
            .toExternalForm());
  }

  @Override
  protected Node createContentPane() {

    VBox box = new VBox();
    box.getStyleClass().add("form-container");
    box.getChildren().addAll(
        Util.getTextFieldForm("Benutzername",
            viewModel.getUserNameProperty()));
    box.getChildren().addAll(
        Util.getPasswordFieldForm("Passwort",
            viewModel.getPasswordProperty()));
    box.getChildren().addAll(
        Util.getTextFieldForm("Vorname",
            viewModel.getFirstNameProperty()));
    box.getChildren().addAll(
        Util.getTextFieldForm("Nachname",
            viewModel.getLastNameProperty()));

    return box;
  }

  @Override
  protected void saveCommand() {
    if (validate()) {
      // Saves the entity
      UserService.getInstance().saveOrUpdate(viewModel.toUserModel());
      dialogStage.close();
    }
  }

  /***
   * Static method for showing the dialog
   *
   * @param owner
   * @param model
   * @return
   */
  public static boolean show(Window owner, User model) {
    ManageUserDialog dialog = new ManageUserDialog(owner, model);
    return dialog.show();

  }
}
```

gui/pages/ManageMenusPage.java

```
package at.lumetsnet.caas.gui.pages;

import java.util.Optional;

import javafx.scene.Node;
import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.scene.control.TableColumn;
import javafx.scene.control.TableView;
import javafx.scene.layout.HBox;
```

```java
import javafx.scene.layout.Priority;
import javafx.scene.layout.VBox;
import at.lumetsnet.caas.gui.ActionTableCell;
import at.lumetsnet.caas.gui.AmountTableCell;
import at.lumetsnet.caas.gui.dialogs.ManageMenuCategoryDialog;
import at.lumetsnet.caas.gui.dialogs.ManageMenuDialog;
import at.lumetsnet.caas.viewmodel.ManageMenusPageViewModel;
import at.lumetsnet.caas.viewmodel.MenuCategoryViewModel;
import at.lumetsnet.caas.viewmodel.MenuViewModel;

/***
 * View page used for managing menus Uses a ManageMenusPageViewModel for
 * business logic operations
 *
 * @author romanlum
 *
 */
public class ManageMenusPage extends VBox implements Showable {

  TableView<MenuCategoryViewModel> categoryTable;
  TableView<MenuViewModel> menuTable;
  ManageMenusPageViewModel viewModel;

  public ManageMenusPage() {

    viewModel = new ManageMenusPageViewModel();
    getStyleClass().add("page-content-container");

    HBox topPane = new HBox();
    topPane.getChildren().add(createTitle("Bereiche"));
    HBox commandContainer = new HBox();
    commandContainer.getStyleClass().add("page-command-container");
    commandContainer.getChildren().add(createAddCategoryButton());
    HBox.setHgrow(commandContainer, Priority.ALWAYS);

    topPane.getChildren().add(commandContainer);
    categoryTable = createCategoryTableView();
    categoryTable.setItems(viewModel.getCategoryList());

    HBox menuTopPane = new HBox();
    menuTopPane.getChildren().add(createTitle("Hauptspeisen"));
    HBox menuCommandContainer = new HBox();
    menuCommandContainer.getStyleClass().add("page-command-container");
    menuCommandContainer.getChildren().add(createAddMenuButton());
    HBox.setHgrow(menuCommandContainer, Priority.ALWAYS);

    menuTopPane.getChildren().add(menuCommandContainer);
    menuTable = createMenuTableView();
    menuTable.setItems(viewModel.getMenuList());

    getChildren().addAll(topPane, categoryTable, menuTopPane, menuTable);

  }

  private Node createAddCategoryButton() {
    Button button = new Button("Bereich anlegen");
    button.getStyleClass().add("page-command");
    button.setOnAction(x -> addCategoryCommand());
    return button;
  }

  private Node createAddMenuButton() {
    Button button = new Button("Hauptspeise anlegen");
    button.getStyleClass().add("page-command");
    button.setOnAction(x -> addMenuCommand());
    return button;
  }

  private Label createTitle(String title) {
    Label titleLabel = new Label(title);
    titleLabel.getStyleClass().add("page-title");
    return titleLabel;
  }

  private TableView<MenuCategoryViewModel> createCategoryTableView() {
    TableView<MenuCategoryViewModel> table = new TableView<>();
    table.getStyleClass().add("table");
    table.setColumnResizePolicy(TableView.CONSTRAINED_RESIZE_POLICY);

    TableColumn<MenuCategoryViewModel, String> column = new TableColumn<>(
        "Name");
    column.setCellValueFactory(x -> x.getValue().getNameProperty());
    table.getColumns().add(column);

    TableColumn<MenuCategoryViewModel, Number> actionColumn = new TableColumn<>(
        "Aktion");
    actionColumn.setCellValueFactory(x -> x.getValue().getIdProperty());
    actionColumn.setCellFactory(p -> new ActionTableCell<>(
        x -> editCategoryCommand(x), x -> deleteCategoryCommand(x)));

    table.getColumns().add(actionColumn);
    return table;
  }

  private TableView<MenuViewModel> createMenuTableView() {
    TableView<MenuViewModel> table = new TableView<>();
    table.getStyleClass().add("table");
    table.setColumnResizePolicy(TableView.CONSTRAINED_RESIZE_POLICY);

    TableColumn<MenuViewModel, String> column = new TableColumn<>("Name");
    column.setCellValueFactory(x -> x.getValue().getDescriptionProperty());
    table.getColumns().add(column);

    TableColumn<MenuViewModel, Number> priceColumn = new TableColumn<>(
        "Preis");
    priceColumn.setCellValueFactory(x -> x.getValue().getPriceProperty());
    priceColumn.setCellFactory(x -> new AmountTableCell<MenuViewModel>());
    table.getColumns().add(priceColumn);

    column = new TableColumn<>("Zeitraum");
    column.setCellValueFactory(x -> x.getValue().getUsageRangeProperty());
    table.getColumns().add(column);

    TableColumn<MenuViewModel, Number> actionColumn = new TableColumn<>(
        "Aktion");
    actionColumn.setCellValueFactory(x -> x.getValue().getIdProperty());
    actionColumn.setCellFactory(p -> new ActionTableCell<>(
        x -> editMenuCommand(x), x -> deleteMenuCommand(x)));

    table.getColumns().add(actionColumn);
    return table;
  }

  private void deleteCategoryCommand(Number x) {
    viewModel.deleteCategoryCommand(x);

  }

  private void editCategoryCommand(Number id) {
    Optional<MenuCategoryViewModel> model = categoryTable.getItems()
        .stream().filter(x -> x.getIdProperty().get() == (long) id)
        .findFirst();

    if (model.isPresent()) {
      boolean result = ManageMenuCategoryDialog.show(getScene()
          .getWindow(), model.get().toCategoryModel());
      if (!result) {
        viewModel.updateCategories();
      }
    }
  }

  private void deleteMenuCommand(Number x) {
    viewModel.deleteMenuCommand(x);

  }

```

```java
  private void editMenuCommand(Number id) {
    Optional<MenuViewModel> model = menuTable.getItems().stream()
        .filter(x -> x.getIdProperty().get() == (long) id).findFirst();

    if (model.isPresent()) {
      boolean result = ManageMenuDialog.show(getScene().getWindow(),
          model.get().toMenuModel());
      if (!result) {
        viewModel.updateMenus();
      }
    }
  }

  private void addCategoryCommand() {
    boolean result = ManageMenuCategoryDialog.show(getScene().getWindow(),
        null);
    if (!result) {
      viewModel.updateCategories();
    }
  }

  private void addMenuCommand() {
    boolean result = ManageMenuDialog.show(getScene().getWindow(), null);
    if (!result) {
      viewModel.updateMenus();
    }
  }

  public void show() {
    viewModel.updateCategories();
    viewModel.updateMenus();
  }
}
```

gui/pages/ManageUsersPage.java

```java
package at.lumetsnet.caas.gui.pages;

import java.util.Optional;

import javafx.scene.Node;
import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.scene.control.TableColumn;
import javafx.scene.control.TableView;
import javafx.scene.layout.HBox;
import javafx.scene.layout.Priority;
import javafx.scene.layout.VBox;
import at.lumetsnet.caas.gui.ManageUserActionCell;
import at.lumetsnet.caas.gui.dialogs.ManageUserDialog;
import at.lumetsnet.caas.viewmodel.ManageUsersPageViewModel;
import at.lumetsnet.caas.viewmodel.UserViewModel;

/***
 * View page used for managing users Uses a ManageUsersPageViewModel for
 * business logic operations
 *
 * @author romanlum
 *
 */
public class ManageUsersPage extends VBox implements Showable {

  TableView<UserViewModel> table;
  ManageUsersPageViewModel viewModel;

  public ManageUsersPage() {

    viewModel = new ManageUsersPageViewModel();
    getStyleClass().add("page-content-container");

    HBox topPane = new HBox();
    topPane.getChildren().add(createTitle("Benutzerliste"));
    HBox commandContainer = new HBox();
    commandContainer.getStyleClass().add("page-command-container");
    commandContainer.getChildren().add(createAddButton());
    HBox.setHgrow(commandContainer, Priority.ALWAYS);

    topPane.getChildren().add(commandContainer);
    table = createTableView();
    table.setItems(viewModel.getUserList());
    getChildren().addAll(topPane, table);

  }

  private Node createAddButton() {
    Button button = new Button("User anlegen");
    button.getStyleClass().add("page-command");
    button.setOnAction(x -> addUserCommand());
    return button;
  }

  private Label createTitle(String title) {
    Label titleLabel = new Label(title);
    titleLabel.getStyleClass().add("page-title");
    return titleLabel;
  }

  private TableView<UserViewModel> createTableView() {
    TableView<UserViewModel> table = new TableView<>();
    table.getStyleClass().add("table");
    table.setColumnResizePolicy(TableView.CONSTRAINED_RESIZE_POLICY);

    TableColumn<UserViewModel, String> column = new TableColumn<>(
        "Benutzername");
    column.setCellValueFactory(x -> x.getValue().getUserNameProperty());
    table.getColumns().add(column);

    column = new TableColumn<>("Vorname");
    column.setCellValueFactory(x -> x.getValue().getFirstNameProperty());
    table.getColumns().add(column);

    column = new TableColumn<>("Nachname");
    column.setCellValueFactory(x -> x.getValue().getLastNameProperty());
    table.getColumns().add(column);

    column = new TableColumn<>("Status");
    column.setCellValueFactory(x -> x.getValue().getLockedStringProperty());
    table.getColumns().add(column);

    TableColumn<UserViewModel, Number> actionColumn = new TableColumn<>(
        "Aktion");
    actionColumn.setCellValueFactory(x -> x.getValue().getIdProperty());
    actionColumn.setCellFactory(p -> new ManageUserActionCell<>(
        x -> editCommand(x), x -> deleteCommand(x),
        x -> toggleLockStateCommand(x)));

    table.getColumns().add(actionColumn);
    return table;
  }

  private void toggleLockStateCommand(Number x) {
    viewModel.toggleLockStateCommand(x);
  }

  private void deleteCommand(Number x) {
    viewModel.deleteCommand(x);

  }

  private void editCommand(Number id) {
    Optional<UserViewModel> model = table.getItems().stream()
        .filter(x -> x.getIdProperty().get() == (long) id).findFirst();

    if (model.isPresent()) {
      // call edit dialog
      boolean result = ManageUserDialog.show(getScene().getWindow(),
          model.get().toUserModel());
```

```java
      if (!result) {
        viewModel.update();
      }
    }
  }

  private void addUserCommand() {
    // call add dialog
    boolean result = ManageUserDialog.show(getScene().getWindow(), null);
    if (!result) {
      viewModel.update();
    }
  }

  public void show() {
    viewModel.update();
  }
}
```

gui/pages/OrdersPage.java

```java
package at.lumetsnet.caas.gui.pages;

import javafx.scene.control.Label;
import javafx.scene.control.TableColumn;
import javafx.scene.control.TableColumn.SortType;
import javafx.scene.control.TableView;
import javafx.scene.layout.VBox;
import at.lumetsnet.caas.viewmodel.OrderViewModel;
import at.lumetsnet.caas.viewmodel.OrdersPageViewModel;

/***
 * View page used for showing the orders Uses a OrdersPageViewModel for business
 * logic operations
 *
 * @author romanlum
 *
 */
public class OrdersPage extends VBox implements Showable {

  TableView<OrderViewModel> table;
  OrdersPageViewModel viewModel;

  public OrdersPage() {

    viewModel = new OrdersPageViewModel();
    getStyleClass().add("page-content-container");

    Label title = createTitle("Heutige Bestellungen");
    getChildren().add(title);

    table = createTableView();
    table.setItems(viewModel.getOrders());
    getChildren().add(table);

  }

  private Label createTitle(String title) {
    Label titleLabel = new Label(title);
    titleLabel.getStyleClass().add("page-title");
    return titleLabel;
  }

  private TableView<OrderViewModel> createTableView() {
    TableView<OrderViewModel> table = new TableView<>();
    table.getStyleClass().add("table");
    table.setColumnResizePolicy(TableView.CONSTRAINED_RESIZE_POLICY);
    TableColumn<OrderViewModel, String> column = new TableColumn<>(
        "Benutzer");
    column.setCellValueFactory(x -> x.getValue().getUserNameProperty());
    table.getColumns().add(column);

    column = new TableColumn<>("Gericht");
    column.setCellValueFactory(x -> x.getValue().getMenuProperty());
    table.getColumns().add(column);

    column = new TableColumn<>("Zeit");
    column.setCellValueFactory(x -> x.getValue().getTimeProperty());
    column.setSortType(SortType.ASCENDING);
    table.getColumns().add(column);
    table.getSortOrder().clear();
    table.getSortOrder().add(column);

    column = new TableColumn<>("Kommentar");
    column.setCellValueFactory(x -> x.getValue().getCommentProperty());

    table.getColumns().add(column);

    return table;
  }

  public void show() {
    viewModel.update();
    table.sort();
  }
}
```

gui/pages/Showable.java

```java
package at.lumetsnet.caas.gui.pages;

/***
 * Simple interface used for marking pages as showable
 *
 * @author romanlum
 *
 */
public interface Showable {

  public void show();
}
```

model/Entity.java

```java
package at.lumetsnet.caas.model;

import java.io.Serializable;

/***
 * Base data entity
 *
 * @author romanlum
 *
 */
public class Entity implements Serializable {
  /**
   *
   */
  private static final long serialVersionUID = 8439956639376975911L;
  protected long id;

  /**
   * @return the id
   */
  public long getId() {
    return id;
  }

  /**
   * @param id
   *            the id to set
   */
  public void setId(long id) {
    this.id = id;
  }
}
```

model/Menu.java

```java
package at.lumetsnet.caas.model;

import java.time.LocalDate;

/***
 * Menu data entity
 *
 * @author romanlum
 *
 */

public class Menu extends Entity {

  /**
   *
   */
  private static final long serialVersionUID = 2683509052388984064L;

  private String description;
  // price is stored in smallest currency unit!
  private long price;
  private LocalDate begin;
  private LocalDate end;
  private MenuCategory category;

  public Menu() {
  }

  /**
   * @param id
   * @param description
   * @param price
   * @param begin
   * @param end
   * @param category
   */
  public Menu(long id, String description, long price, LocalDate begin,
      LocalDate end, MenuCategory category) {
    this.id = id;
    this.description = description;
    this.price = price;
    this.begin = begin;
    this.end = end;
    this.category = category;
  }

  /**
```

```java
   * @return the description
   */
  public String getDescription() {
    return description;
  }

  /**
   * @param description
   *            the description to set
   */
  public void setDescription(String description) {
    this.description = description;
  }

  /**
   * @return the price in the smallest currency unit
   *
   */
  public long getPrice() {
    return price;
  }

  /**
   * @param price
   *            the price to set this price is set in the smallest currency
   *            unit
   */
  public void setPrice(long price) {
    this.price = price;
  }

  /**
   * @return the begin
   */
  public LocalDate getBegin() {
    return begin;
  }

  /**
   * @param begin
   *            the begin to set
   */
  public void setBegin(LocalDate begin) {
    this.begin = begin;
  }

  /**
   * @return the end
   */
  public LocalDate getEnd() {
    return end;
  }

  /**
   * @param end
   *             the end to set
   */
  public void setEnd(LocalDate end) {
    this.end = end;
  }

  /**
   * @return the category
   */
  public MenuCategory getCategory() {
    return category;
  }

  /**
   * @param category
   *             the category to set
   */
  public void setCategory(MenuCategory category) {
    this.category = category;
  }

  /**
   * @return the categoryId
   */
  public long getCategoryId() {
    return category != null ? category.getId() : -1;
  }

  /**
   * @param categoryId
   *             the categoryId to set
   */
  public void setCategoryId(long categoryId) {
    this.category = new MenuCategory();
    this.category.setId(categoryId);
  }

}
```

model/MenuCategory.java

```java
package at.lumetsnet.caas.model;

```

```java
/***
 * MenuCategory data entity
 *
 * @author romanlum
 *
 */

public class MenuCategory extends Entity {

  /**
   *
   */
  private static final long serialVersionUID = -6364790602658825179L;
  private String name;

  public MenuCategory() {
  }

  /**
   * @param id
   * @param name
   */
  public MenuCategory(long id, String name) {
    this.id = id;
    this.name = name;
  }

  /**
   * @return the name
   */
  public String getName() {
    return name;
  }

  /**
   * @param name
   *            the name to set
   */
  public void setName(String name) {
    this.name = name;
  }

}
```

model/Order.java

```java
package at.lumetsnet.caas.model;

import java.time.LocalDateTime;
```

```java

/***
 * Order data entity
 *
 * @author romanlum
 *
 */

public class Order extends Entity {

  /**
   *
   */
  private static final long serialVersionUID = -9188235335595584261L;
  private Menu menu;
  private User user;
  private LocalDateTime time;
  private String comment;

  public Order() {
  }

  /**
   * @param id
   * @param menu
   * @param user
   * @param time
   * @param comment
   */
  public Order(long id, Menu menu, User user, LocalDateTime time,
      String comment) {
    this.id = id;
    this.menu = menu;
    this.user = user;
    this.time = time;
    this.comment = comment;
  }

  /**
   * @return the menu
   */
  public Menu getMenu() {
    return menu;
  }

  /**
   * @param menu
   *            the menu to set
   */
```

```java
public void setMenu(Menu menu) {
  this.menu = menu;
}

/**
 * @return the user
 */
public User getUser() {
  return user;
}

/**
 * @param user
 *            the user to set
 */
public void setUser(User user) {
  this.user = user;
}

/**
 * @return the comment
 */
public String getComment() {
  return comment;
}

/**
 * @param comment
 *            the comment to set
 */
public void setComment(String comment) {
  this.comment = comment;
}

/**
 * @return the time
 */
public LocalDateTime getTime() {
  return time;
}

/**
 * @param time
 *            the time to set
 */
public void setTime(LocalDateTime time) {
  this.time = time;
}

```

```java
	/**
	 * @return the menuId
	 */
	public long getMenuId() {
		return menu != null ? menu.getId() : -1;
	}

	/**
	 * @param menuId
	 *            the menuId to set
	 */
	public void setMenuId(long menuId) {
		this.menu = new Menu();
		this.menu.setId(menuId);
	}

	/**
	 * @return the userId
	 */
	public long getUserId() {
		return user != null ? user.getId() : -1;
	}

	/**
	 * @param userId
	 *            the userId to set
	 */
	public void setUserId(long userId) {
		this.user = new User();
		this.user.setId(userId);
	}

}
```

model/User.java

```java
package at.lumetsnet.caas.model;

import java.io.Serializable;

/***
 * user data entity
 *
 * @author romanlum
 *
 */

public class User extends Entity implements Serializable {

```

```java
/**
 *
 */
private static final long serialVersionUID = -9121752719148285484L;

private String userName;
private String password;
private String firstName;
private String lastName;
private boolean locked;

public User() {
}

/**
 * @param id
 * @param userName
 * @param password
 * @param firstName
 * @param lastName
 * @param locked
 * @param isAdmin
 */
public User(long id, String userName, String password, String firstName,
    String lastName, boolean locked) {
  this.id = id;
  this.userName = userName;
  this.password = password;
  this.firstName = firstName;
  this.lastName = lastName;
  this.locked = locked;
}

/**
 * @return the userName
 */
public String getUserName() {
  return userName;
}

/**
 * @param userName
 *            the userName to set
 */
public void setUserName(String userName) {
  this.userName = userName;
}

/**
```

```
   * @return the password
   */
  public String getPassword() {
    return password;
  }

  /**
   * @param password
   *            the password to set
   */
  public void setPassword(String password) {
    this.password = password;
  }

  /**
   * @return the firstName
   */
  public String getFirstName() {
    return firstName;
  }

  /**
   * @param firstName
   *            the firstName to set
   */
  public void setFirstName(String firstName) {
    this.firstName = firstName;
  }

  /**
   * @return the lastName
   */
  public String getLastName() {
    return lastName;
  }

  /**
   * @param lastName
   *            the lastName to set
   */
  public void setLastName(String lastName) {
    this.lastName = lastName;
  }

  /**
   * @return the locked
   */
  public boolean isLocked() {
    return locked;
  }

  /**
   * @param locked
   *            the locked to set
   */
  public void setLocked(boolean locked) {
    this.locked = locked;
  }

}
```

rmi/RemoteServiceMain.java

```
package at.lumetsnet.caas.rmi;

import java.net.MalformedURLException;
import java.rmi.Naming;
import java.rmi.Remote;
import java.rmi.RemoteException;
import java.rmi.registry.LocateRegistry;
import java.rmi.server.UnicastRemoteObject;

import at.lumetsnet.caas.rmi.impl.MenuServiceImpl;
import at.lumetsnet.caas.rmi.impl.OrderServiceImpl;
import at.lumetsnet.caas.rmi.impl.UserServiceImpl;

/***
 * Rmi remote service class
 *
 * @author romanlum
 *
 */
public class RemoteServiceMain {

  public static void main(String[] args) {
    try {
      String host = "localhost:1099";
      if (args.length == 1) {
        host = args[0];
      }

      LocateRegistry.createRegistry(getPort(host));
      registerSerice(new UserServiceImpl(), host, "CaasUserService");
      registerSerice(new MenuServiceImpl(), host, "CaasMenuService");
      registerSerice(new OrderServiceImpl(), host, "CaasOrderService");

    } catch (Exception ex) {
      System.out.println("Could not register services");
```

```java
      ex.printStackTrace();
    }

  }

  /***
   * Creates and exports the service
   *
   * @param remote
   * @param host
   * @param name
   */
  public static void registerSerice(Remote remote, String host, String name) {
    Remote serviceStub;
    try {
      serviceStub = UnicastRemoteObject.exportObject(remote, 0);
      Naming.rebind("rmi://" + host + "/" + name, serviceStub);
      System.out.println(name + " available on port " + "rmi://" + host
          + "/" + name);
    } catch (RemoteException | MalformedURLException e) {
      System.out.println("Could not register service " + name);
      e.printStackTrace();
    }

  }

  /***
   * Gets the port from host:port string Defaults to 1099 (RMI default port)
   *
   * @param hostPort
   * @return
   */
  private static int getPort(String hostPort) {
    int idx = hostPort.lastIndexOf(':');
    if (idx == -1) {
      return 1099;
    } else {
      return Integer.parseInt(hostPort.substring(idx + 1));
    }
  }

}
```

rmi/impl/MenuServiceImpl.java

```java
package at.lumetsnet.caas.rmi.impl;

import java.util.Collection;

import at.lumetsnet.caas.dal.MenuCategoryDao;
import at.lumetsnet.caas.dal.MenuCategoryDaoJdbc;
import at.lumetsnet.caas.dal.MenuDao;
import at.lumetsnet.caas.dal.MenuDaoJdbc;
import at.lumetsnet.caas.model.Menu;
import at.lumetsnet.caas.model.MenuCategory;
import at.lumetsnet.caas.rmi.interfaces.RemoteMenuService;

/***
 * MenuService impl using jdbc
 *
 * @author romanlum
 *
 */
public class MenuServiceImpl extends ServiceImpl implements RemoteMenuService {

  private MenuDao menuDao;
  private MenuCategoryDao categoryDao;

  public MenuServiceImpl() {
    menuDao = new MenuDaoJdbc(CONNECTION_STRING, USER_NAME, PASSWORD);
    categoryDao = new MenuCategoryDaoJdbc(CONNECTION_STRING, USER_NAME,
        PASSWORD);
  }

  /***
   * Fetches all categories
   *
   * @return
   */
  public Collection<MenuCategory> getAllCategories() {
    return categoryDao.getAll();
  }

  /***
   * Deletes the category with the given id
   *
   * @param id
   */
  public void deleteCategory(long id) {
    categoryDao.delete(id);
  }

  /***
   * Deletes the menu with the given id
   *
   * @param id
   */
  public void deleteMenu(long id) {
    menuDao.delete(id);
  }

  /***
   * Saves or adds the category
   *
   * @param data
   */
  public void saveOrUpdateCategory(MenuCategory data) {
    categoryDao.saveOrUpdate(data);
  }

  /***
   * Saves or adds the menu
   *
   * @param data
   */
  public void saveOrUpdateMenu(Menu data) {
    menuDao.saveOrUpdate(data);
  }

  /***
   * Fetches all menus
   *
   * @return
   */
  public Collection<Menu> getAllMenus() {
    Collection<Menu> data = menuDao.getAll();
    data.stream().forEach(x -> {
      x.setCategory(categoryDao.get(x.getCategoryId()));
    });
    return data;

  }
}
```

rmi/impl/OrderServiceImpl.java

```java
package at.lumetsnet.caas.rmi.impl;

import java.rmi.RemoteException;
import java.time.LocalDate;
import java.util.Collection;

import at.lumetsnet.caas.dal.MenuDao;
import at.lumetsnet.caas.dal.MenuDaoJdbc;
import at.lumetsnet.caas.dal.OrderDao;
import at.lumetsnet.caas.dal.OrderDaoJdbc;
import at.lumetsnet.caas.dal.UserDao;
import at.lumetsnet.caas.dal.UserDaoJdbc;
import at.lumetsnet.caas.model.Order;
import at.lumetsnet.caas.rmi.interfaces.RemoteOrderService;

/***
 * Order service impl using jdbc
 *
 * @author romanlum
 *
 */
public class OrderServiceImpl extends ServiceImpl implements RemoteOrderService {
  private OrderDao orderDao;
  private MenuDao menuDao;
  private UserDao userDao;

  public OrderServiceImpl() {
    orderDao = new OrderDaoJdbc(CONNECTION_STRING, USER_NAME, PASSWORD);
    menuDao = new MenuDaoJdbc(CONNECTION_STRING, USER_NAME, PASSWORD);
    userDao = new UserDaoJdbc(CONNECTION_STRING, USER_NAME, PASSWORD);
  }

  @Override
  public Collection<Order> getTodaysOrders() throws RemoteException {
    Collection<Order> result = orderDao.getOrdersByDate(LocalDate.now());
    // fetch data for referenced objects
    result.stream().forEach(x -> {
      x.setMenu(menuDao.get(x.getMenuId()));
      x.setUser(userDao.get(x.getUserId()));
    });
    return result;
  }

}
```

rmi/impl/ServiceImpl.java

```java
package at.lumetsnet.caas.rmi.impl;

/***
 * Base class for all service impls
 *
 * @author romanlum
 *
 */
public class ServiceImpl {

  protected static final String CONNECTION_STRING = "jdbc:mysql://localhost/CaasDb";
  protected static final String USER_NAME = "root";
  protected static final String PASSWORD = null;

}
```

rmi/impl/UserServiceImpl.java

```java
package at.lumetsnet.caas.rmi.impl;

import java.rmi.RemoteException;
import java.util.Collection;

import at.lumetsnet.caas.dal.UserDao;
import at.lumetsnet.caas.dal.UserDaoJdbc;
import at.lumetsnet.caas.model.User;
import at.lumetsnet.caas.rmi.interfaces.RemoteUserService;

/***
 * User service impl using jdbc
 *
 * @author romanlum
 *
 */
public class UserServiceImpl extends ServiceImpl implements RemoteUserService {

  private UserDao dao;

  public UserServiceImpl() {
    dao = new UserDaoJdbc(CONNECTION_STRING, USER_NAME, PASSWORD);
  }

  /***
   * Fetches all users
   *
   * @return
   */
  public Collection<User> getAllUsers() throws RemoteException {
    return dao.getAll();
  }

  /***
   * delete the user
   *
   * @param id
   */
  public void deleteUser(long id) throws RemoteException {
    dao.delete(id);
  }

  /***
   * Toggles the locked state of a user
   *
   * @param id
   */
  public void toggleLockState(long id) throws RemoteException {
    User user = dao.get(id);
    if (user != null) {
      user.setLocked(!user.isLocked());
    }
    saveOrUpdate(user);
  }

  /***
   * Saves or adds the user
   *
   * @param userModel
   */
  public void saveOrUpdate(User userModel) throws RemoteException {
    dao.saveOrUpdate(userModel);
  }

}
```

rmi/interfaces/RemoteMenuService.java

```
package at.lumetsnet.caas.rmi.interfaces;

import java.rmi.Remote;
import java.rmi.RemoteException;
import java.util.Collection;

import at.lumetsnet.caas.model.Menu;
import at.lumetsnet.caas.model.MenuCategory;

public interface RemoteMenuService extends Remote {

  /***
   * Fetches all categories
   *
   * @return
   */
  public Collection<MenuCategory> getAllCategories() throws RemoteException;

  /***
   * Deletes the category with the given id
   *
   * @param id
   */
  public void deleteCategory(long id) throws RemoteException;

```

```java
  /***
   * Deletes the menu with the given id
   *
   * @param id
   */
  public void deleteMenu(long id) throws RemoteException;

  /***
   * Saves or adds the category
   *
   * @param data
   */
  public void saveOrUpdateCategory(MenuCategory data) throws RemoteException;

  /***
   * Saves or adds the menu
   *
   * @param data
   */
  public void saveOrUpdateMenu(Menu data) throws RemoteException;

  /***
   * Fetches all menus
   *
   * @return
   */
  public Collection<Menu> getAllMenus() throws RemoteException;
}
```

rmi/interfaces/RemoteOrderService.java

```java
package at.lumetsnet.caas.rmi.interfaces;

import java.rmi.Remote;
import java.rmi.RemoteException;
import java.util.Collection;

import at.lumetsnet.caas.model.Order;

public interface RemoteOrderService extends Remote {

  /***
   * Fetches the orders of today
   *
   * @return
   */
  public Collection<Order> getTodaysOrders() throws RemoteException;

}
```

rmi/interfaces/RemoteUserService.java

```java
package at.lumetsnet.caas.rmi.interfaces;

import java.rmi.Remote;
import java.rmi.RemoteException;
import java.util.Collection;

import at.lumetsnet.caas.model.User;

public interface RemoteUserService extends Remote {

	/***
	 * Fetches all users
	 *
	 * @return
	 */
	public Collection<User> getAllUsers() throws RemoteException;

	/***
	 * delete the user
	 *
	 * @param id
	 */
	public void deleteUser(long id) throws RemoteException;

	/***
	 * Toggles the locked state of a user
	 *
	 * @param id
	 */
	public void toggleLockState(long id) throws RemoteException;

	/***
	 * Saves or adds the user
	 *
	 * @param userModel
	 */
	public void saveOrUpdate(User userModel) throws RemoteException;
}
```

viewmodel/ManageMenusPageViewModel.java

```java
package at.lumetsnet.caas.viewmodel;

import java.util.Collection;

import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import at.lumetsnet.caas.business.MenuService;
import at.lumetsnet.caas.model.Menu;
import at.lumetsnet.caas.model.MenuCategory;

/***
 * ViewModel (logic) class for ManageMenusPage
 *
 * @author romanlum
 *
 */
public class ManageMenusPageViewModel {

  private ObservableList<MenuCategoryViewModel> categories;
  private ObservableList<MenuViewModel> menus;

  public ManageMenusPageViewModel() {
    categories = FXCollections.observableArrayList();
    menus = FXCollections.observableArrayList();
  }

  public void updateCategories() {
    Collection<MenuCategory> data = MenuService.getInstance()
        .getAllCategories();
    categories.clear();
    if (data != null) {
      data.forEach(x -> categories.add(new MenuCategoryViewModel(x)));
    }
  }

  public void updateMenus() {
    Collection<Menu> data = MenuService.getInstance().getAllMenus();
    menus.clear();
    if (data != null) {
      data.forEach(x -> menus.add(new MenuViewModel(x)));
    }
  }

  public ObservableList<MenuCategoryViewModel> getCategoryList() {
    return categories;
  }

  public ObservableList<MenuViewModel> getMenuList() {
    return menus;
  }

  public void deleteCategoryCommand(Number userId) {
    MenuService.getInstance().deleteCategory((long) userId);
    updateCategories();
    updateMenus();

  }

  public void deleteMenuCommand(Number id) {
    MenuService.getInstance().deleteMenu((long) id);
    updateMenus();

  }

}
```

viewmodel/ManageUsersPageViewModel.java

```java
package at.lumetsnet.caas.viewmodel;

import java.util.Collection;

import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import at.lumetsnet.caas.business.UserService;
import at.lumetsnet.caas.model.User;

/***
 * ViewModel (logic) class for ManageUsersPage
 *
 * @author romanlum
 *
 */

public class ManageUsersPageViewModel {

  private ObservableList<UserViewModel> users;

  public ManageUsersPageViewModel() {
    users = FXCollections.observableArrayList();
  }

  public void update() {
    Collection<User> data = UserService.getInstance().getAllUsers();
    users.clear();
    if (data != null) {
      data.forEach(x -> users.add(new UserViewModel(x)));
    }
  }

  public ObservableList<UserViewModel> getUserList() {
    return users;
  }

  public void deleteCommand(Number userId) {
    UserService.getInstance().deleteUser((long) userId);
    update();

  }

  public void toggleLockStateCommand(Number userId) {
    UserService.getInstance().toggleLockState((long) userId);
    update();
  }
}
```

viewmodel/MenuCategoryViewModel.java

```
package at.lumetsnet.caas.viewmodel;

import javafx.beans.property.LongProperty;
import javafx.beans.property.SimpleLongProperty;
import javafx.beans.property.SimpleStringProperty;
import javafx.beans.property.StringProperty;
import at.lumetsnet.caas.model.MenuCategory;

/***
 * ViewModel wrapper for MenuCategory entity Uses javafx properties for
 * databinding
 *
 * @author romanlum
 *
 */
public class MenuCategoryViewModel implements Validatable {

  private LongProperty idProperty;
  private StringProperty nameProperty;

  public MenuCategoryViewModel(MenuCategory input) {
    if (input == null) {
      input = new MenuCategory();
      input.setId(-1);
    }
    idProperty = new SimpleLongProperty(input.getId());
    nameProperty = new SimpleStringProperty(input.getName());

  }

  public MenuCategory toCategoryModel() {
    MenuCategory result = new MenuCategory();
    result.setId(idProperty.get());
    result.setName(nameProperty.get());
    return result;

  }

  public boolean validate() {
    return nameProperty.get() != null && !nameProperty.get().isEmpty();
  }

  @Override
  public String toString() {
    return getNameProperty().get();
  }

  /**
   * @return the id
   */
  public LongProperty getIdProperty() {
    return idProperty;
  }

  /**
   * @return the userNameProperty
   */
  public StringProperty getNameProperty() {
    return nameProperty;
  }

}
```

viewmodel/MenuViewModel.java

```
package at.lumetsnet.caas.viewmodel;

import java.time.LocalDate;
import java.time.format.DateTimeFormatter;

import javafx.beans.property.LongProperty;
import javafx.beans.property.ObjectProperty;
import javafx.beans.property.SimpleLongProperty;
import javafx.beans.property.SimpleObjectProperty;
import javafx.beans.property.SimpleStringProperty;
import javafx.beans.property.StringProperty;
import at.lumetsnet.caas.model.Menu;

/***
 * ViewModel wrapper for Menu entity Uses javafx properties for databinding
 *
 * @author romanlum
 *
 */

public class MenuViewModel implements Validatable {

  private LongProperty idProperty;
  private StringProperty descriptionProperty;
  private LongProperty priceProperty;
  private ObjectProperty<LocalDate> beginProperty;
  private ObjectProperty<LocalDate> endProperty;
  private ObjectProperty<MenuCategoryViewModel> categoryProperty;
  private StringProperty usageRangeProperty;
  private StringProperty priceAsStringProperty;

  public MenuViewModel(Menu data) {
    if (data == null) {
      data = new Menu();
      data.setId(-1);
    }
    idProperty = new SimpleLongProperty(data.getId());
    descriptionProperty = new SimpleStringProperty(data.getDescription());
    priceProperty = new SimpleLongProperty(data.getPrice());
    beginProperty = new SimpleObjectProperty<LocalDate>(data.getBegin());
    endProperty = new SimpleObjectProperty<LocalDate>(data.getEnd());
    categoryProperty = new SimpleObjectProperty<>(
        new MenuCategoryViewModel(data.getCategory()));
    if (data.getBegin() != null && data.getEnd() != null) {
      usageRangeProperty = new SimpleStringProperty(data.getBegin()
          .format(DateTimeFormatter.ISO_LOCAL_DATE)
          + " - "
          + data.getEnd().format(DateTimeFormatter.ISO_LOCAL_DATE));
    } else {
      usageRangeProperty = new SimpleStringProperty("");
    }
    priceAsStringProperty = new SimpleStringProperty(""
        + (data.getPrice() / (double) 100));
  }

  public Menu toMenuModel() {
    Menu menu = new Menu();
    menu.setId(idProperty.get());
    menu.setBegin(beginProperty.get());
    menu.setEnd(endProperty.get());
    menu.setDescription(descriptionProperty.get());

    long amount = (long) (Double.parseDouble(priceAsStringProperty.get()) * 100);
    menu.setPrice(amount);
    menu.setCategory(categoryProperty.get().toCategoryModel());
    return menu;

  }

  /**
   * @return the idProperty
   */
  public LongProperty getIdProperty() {
    return idProperty;
  }

  /**
   * @return the descriptionNameProperty
   */
  public StringProperty getDescriptionProperty() {
    return descriptionProperty;
  }

  /**
   * @return the priceProperty
   */
  public LongProperty getPriceProperty() {
    return priceProperty;
  }

  /**
   * @return the beginProperty
   */
  public ObjectProperty<LocalDate> getBeginProperty() {
    return beginProperty;
  }

  /**
   * @return the endProperty
   */
  public ObjectProperty<LocalDate> getEndProperty() {
    return endProperty;
  }

  /**
   * @return the categoryProperty
   */
  public ObjectProperty<MenuCategoryViewModel> getCategoryProperty() {
    return categoryProperty;
  }

  /**
   * @return the usageRange
   */
```

```java
  public StringProperty getUsageRangeProperty() {
    return usageRangeProperty;
  }

  /**
   * @return the priceAsStringProperty
   */
  public StringProperty getPriceAsStringProperty() {
    return priceAsStringProperty;
  }

  @Override
  public boolean validate() {
    boolean val = descriptionProperty.get() != null
        && !descriptionProperty.get().isEmpty()
        && categoryProperty.get() != null
        && categoryProperty.get().getIdProperty().get() != -1;

    if (!val)
      return val;
    // extended validation of price
    try {
      Double.parseDouble(priceAsStringProperty.get());

    } catch (NumberFormatException ex) {
      return false;
    }
    return true;
  }

}
```

viewmodel/OrdersPageViewModel.java

```java
package at.lumetsnet.caas.viewmodel;

import java.util.Collection;

import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import at.lumetsnet.caas.business.OrderService;
import at.lumetsnet.caas.model.Order;

/***
 * ViewModel (logic) class for OrdersPage
 *
 * @author romanlum
 *
 */
```

```java
public class OrdersPageViewModel {

  private ObservableList<OrderViewModel> orders;

  public OrdersPageViewModel() {
    orders = FXCollections.observableArrayList();
  }

  public void update() {
    Collection<Order> orderData = OrderService.getInstance()
        .getTodaysOrders();
    orders.clear();
    if (orderData != null) {
      orderData.forEach(x -> orders.add(new OrderViewModel(x)));
    }
  }

  public ObservableList<OrderViewModel> getOrders() {
    return orders;
  }
}
```

viewmodel/OrderViewModel.java

```java
package at.lumetsnet.caas.viewmodel;

import java.time.format.DateTimeFormatter;

import javafx.beans.property.SimpleStringProperty;
import javafx.beans.property.StringProperty;
import at.lumetsnet.caas.model.Order;

/***
 * ViewModel wrapper for order entity Uses javafx properties for databinding
 *
 * @author romanlum
 *
 */
public class OrderViewModel {

  private StringProperty userNameProperty;
  private StringProperty menuProperty;
  private StringProperty timeProperty;
  private StringProperty commentProperty;

  public OrderViewModel(Order order) {
    userNameProperty = new SimpleStringProperty(order.getUser()
        .getFirstName() + " " + order.getUser().getLastName());
    menuProperty = new SimpleStringProperty(order.getMenu()
        .getDescription());
    timeProperty = new SimpleStringProperty(order.getTime().format(
        DateTimeFormatter.ofPattern("HH:mm")));
    commentProperty = new SimpleStringProperty(order.getComment());
  }

  /**
   * @return the userNameProperty
   */
  public StringProperty getUserNameProperty() {
    return userNameProperty;
  }

  /**
   * @return the menuProperty
   */
  public StringProperty getMenuProperty() {
    return menuProperty;
  }

  /**
   * @return the timeProperty
   */
  public StringProperty getTimeProperty() {
    return timeProperty;
  }

  /**
   * @return the commentProperty
   */
  public StringProperty getCommentProperty() {
    return commentProperty;
  }

}
```

viewmodel/UserViewModel.java

```
package at.lumetsnet.caas.viewmodel;

import javafx.beans.property.BooleanProperty;
import javafx.beans.property.LongProperty;
import javafx.beans.property.SimpleBooleanProperty;
import javafx.beans.property.SimpleLongProperty;
import javafx.beans.property.SimpleStringProperty;
import javafx.beans.property.StringProperty;
import at.lumetsnet.caas.model.User;

/***
```

```java
 * ViewModel wrapper for User entity Uses javafx properties for databinding
 *
 * @author romanlum
 *
 */

public class UserViewModel implements Validatable {

  private LongProperty idProperty;
  private StringProperty userNameProperty;
  private StringProperty passwordProperty;
  private StringProperty firstNameProperty;
  private StringProperty lastNameProperty;
  private BooleanProperty lockedProperty;

  public UserViewModel(User user) {
    if (user == null) {
      user = new User();
      user.setId(-1);
    }
    idProperty = new SimpleLongProperty(user.getId());
    userNameProperty = new SimpleStringProperty(user.getUserName());
    passwordProperty = new SimpleStringProperty(user.getPassword());
    firstNameProperty = new SimpleStringProperty(user.getFirstName());
    lastNameProperty = new SimpleStringProperty(user.getLastName());
    lockedProperty = new SimpleBooleanProperty(user.isLocked());

  }

  public User toUserModel() {
    User result = new User();
    result.setId(idProperty.get());
    result.setUserName(userNameProperty.get());
    result.setPassword(passwordProperty.get());
    result.setFirstName(firstNameProperty.get());
    result.setLastName(lastNameProperty.get());
    result.setLocked(lockedProperty.get());
    return result;

  }

  /**
   * @return the id
   */
  public LongProperty getIdProperty() {
    return idProperty;
  }

  /**
```

```
   * @return the userNameProperty
   */
  public StringProperty getUserNameProperty() {
    return userNameProperty;
  }

  /**
   * @return the passwordProperty
   */
  public StringProperty getPasswordProperty() {
    return passwordProperty;
  }

  /**
   * @return the firstNameProperty
   */
  public StringProperty getFirstNameProperty() {
    return firstNameProperty;
  }

  /**
   * @return the lastNameProperty
   */
  public StringProperty getLastNameProperty() {
    return lastNameProperty;
  }

  /**
   * @return the lockedProperty
   */
  public BooleanProperty getLockedProperty() {
    return lockedProperty;
  }

  /**
   * @return the lockedProperty as string
   */
  public StringProperty getLockedStringProperty() {
    return new SimpleStringProperty(lockedProperty.get() ? "Gesperrt"
        : "Ok");
  }

  @Override
  public boolean validate() {
    return userNameProperty.get() != null
        && !userNameProperty.get().isEmpty()
        && passwordProperty.get() != null
        && !passwordProperty.get().isEmpty()
        && firstNameProperty.get() != null
        && !firstNameProperty.get().isEmpty()
        && lastNameProperty.get() != null
        && !lastNameProperty.get().isEmpty();

  }

}
```

viewmodel/Validatable.java

```
package at.lumetsnet.caas.viewmodel;

/***
 * Interface used for validating entity view models
 *
 * @author romanlum
 *
 */
public interface Validatable {

  /***
   * validates the entity
   *
   * @return true on successfull validation otherwise false
   */
  boolean validate();
}
```

1.4 Test - Sourcecode - Java

GenericDaoTest.java

```java
package at.lumetsnet.caas.test;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.SQLException;

import at.lumetsnet.caas.dal.DataAccessException;

public class GenericDaoTest {

  protected static final String CONNECTION_STRING = "jdbc:mysql://localhost/CaasDb";
  protected static final String USER_NAME = "root";
  protected static final String PASSWORD = null;

  public void cleanTable(String table) {
    try (Connection con = DriverManager.getConnection(CONNECTION_STRING,
        USER_NAME, PASSWORD)) {
      // clean table
      try (PreparedStatement stmt = con.prepareStatement(String.format(
          "Delete from `%s`", table))) {
        stmt.executeUpdate();
      }
    } catch (SQLException ex) {
      throw new DataAccessException(ex.getMessage());
    }
  }
}
```

MenuCategoryDaoTest.java

```java
package at.lumetsnet.caas.test;

import static org.junit.Assert.assertEquals;
import static org.junit.Assert.assertTrue;

import org.junit.Before;
import org.junit.Test;

import at.lumetsnet.caas.dal.MenuCategoryDaoJdbc;
import at.lumetsnet.caas.model.MenuCategory;

public class MenuCategoryDaoTest extends GenericDaoTest {

```

```java
private MenuCategoryDaoJdbc dao;

@Before
public void setUp() {
  cleanTable("Order");
  cleanTable("Menu");
  cleanTable("MenuCategory");
  cleanTable("User");

  dao = new MenuCategoryDaoJdbc(CONNECTION_STRING, USER_NAME, PASSWORD);
}

@Test
public void insertTest() {

  MenuCategory entity = new MenuCategory(-1, "category");
  dao.saveOrUpdate(entity);
  assertTrue(entity.getId() != -1);
}

@Test
public void updateTest() {
  MenuCategory entity = new MenuCategory(-1, "category");
  dao.saveOrUpdate(entity);
  entity.setName("newcat");
  dao.saveOrUpdate(entity);
  MenuCategory dbEntry = dao.get(entity.getId());
  assertEquals("newcat", dbEntry.getName());
}

@Test
public void getTest() {
  MenuCategory entity = new MenuCategory(-1, "category");
  dao.saveOrUpdate(entity);

  MenuCategory dbEntry = dao.get(entity.getId());
  assertEquals(entity.getName(), dbEntry.getName());

}

@Test
public void getAllTest() {
  MenuCategory entity = new MenuCategory(-1, "category");
  dao.saveOrUpdate(entity);
  entity = new MenuCategory(-1, "category II");
  dao.saveOrUpdate(entity);
  assertEquals(2, dao.getAll().size());
}

```

```java
  @Test
  public void deleteTest() {
    MenuCategory entity = new MenuCategory(-1, "category");
    dao.saveOrUpdate(entity);
    assertTrue(dao.get(entity.getId()) != null);
    dao.delete(entity.getId());
    assertTrue(dao.get(entity.getId()) == null);
  }

}
```

MenuDaoTest.java

```java
package at.lumetsnet.caas.test;

import static org.junit.Assert.assertEquals;
import static org.junit.Assert.assertTrue;

import java.time.LocalDate;

import org.junit.Before;
import org.junit.Test;

import at.lumetsnet.caas.dal.MenuCategoryDao;
import at.lumetsnet.caas.dal.MenuCategoryDaoJdbc;
import at.lumetsnet.caas.dal.MenuDao;
import at.lumetsnet.caas.dal.MenuDaoJdbc;
import at.lumetsnet.caas.model.Menu;
import at.lumetsnet.caas.model.MenuCategory;

public class MenuDaoTest extends GenericDaoTest {

  private MenuDao dao;
  private MenuCategoryDao categoryDao;

  private long catId;

  @Before
  public void setUp() {
    cleanTable("Order");
    cleanTable("Menu");
    cleanTable("MenuCategory");
    cleanTable("User");

    dao = new MenuDaoJdbc(CONNECTION_STRING, USER_NAME, PASSWORD);
    categoryDao = new MenuCategoryDaoJdbc(CONNECTION_STRING, USER_NAME,
        PASSWORD);
    MenuCategory cat = new MenuCategory(-1, "Cat1");
    categoryDao.saveOrUpdate(cat);
    catId = cat.getId();

  }

  @Test
  public void insertTest() {

    Menu entity = new Menu(-1, "desc", 1000, LocalDate.now(), LocalDate
        .now().plusDays(100), null);
    entity.setCategoryId(catId);
    dao.saveOrUpdate(entity);
    assertTrue(entity.getId() != -1);
  }

  @Test
  public void updateTest() {
    Menu entity = new Menu(-1, "desc", 1000, LocalDate.now(), LocalDate
        .now().plusDays(100), null);
    entity.setCategoryId(catId);
    dao.saveOrUpdate(entity);
    entity.setDescription("new desc");
    dao.saveOrUpdate(entity);
    Menu dbEntry = dao.get(entity.getId());
    assertEquals("new desc", dbEntry.getDescription());
  }

  @Test
  public void getTest() {
    Menu entity = new Menu(-1, "desc", 1000, LocalDate.now(), LocalDate
        .now().plusDays(100), null);
    entity.setCategoryId(catId);
    dao.saveOrUpdate(entity);

    Menu dbEntity = dao.get(entity.getId());
    assertEquals(entity.getBegin(), dbEntity.getBegin());
    assertEquals(entity.getCategoryId(), dbEntity.getCategoryId());
    assertEquals(entity.getDescription(), dbEntity.getDescription());
    assertEquals(entity.getEnd(), dbEntity.getEnd());
    assertEquals(entity.getPrice(), dbEntity.getPrice());

  }

  @Test
  public void getAllTest() {
    Menu entity = new Menu(-1, "desc", 1000, LocalDate.now(), LocalDate
        .now().plusDays(100), null);
    entity.setCategoryId(catId);
    dao.saveOrUpdate(entity);
    entity = new Menu(-1, "desc2", 2000, LocalDate.now(), LocalDate.now()
        .plusDays(100), null);
    entity.setCategoryId(catId);

    dao.saveOrUpdate(entity);
    assertEquals(2, dao.getAll().size());
  }

  @Test
  public void deleteTest() {
    Menu entity = new Menu(-1, "desc", 1000, LocalDate.now(), LocalDate
        .now().plusDays(100), null);
    entity.setCategoryId(catId);

    dao.saveOrUpdate(entity);
    assertTrue(dao.get(entity.getId()) != null);
    dao.delete(entity.getId());
    assertTrue(dao.get(entity.getId()) == null);
  }

}
```

OrderDaoTest.java

```
package at.lumetsnet.caas.test;

import static org.junit.Assert.assertEquals;
import static org.junit.Assert.assertTrue;

import java.time.LocalDate;
import java.time.LocalDateTime;
import java.util.Collection;

import org.junit.Before;
import org.junit.Test;

import at.lumetsnet.caas.dal.MenuCategoryDao;
import at.lumetsnet.caas.dal.MenuCategoryDaoJdbc;
import at.lumetsnet.caas.dal.MenuDao;
import at.lumetsnet.caas.dal.MenuDaoJdbc;
import at.lumetsnet.caas.dal.OrderDao;
import at.lumetsnet.caas.dal.OrderDaoJdbc;
import at.lumetsnet.caas.dal.UserDao;
import at.lumetsnet.caas.dal.UserDaoJdbc;
import at.lumetsnet.caas.model.Menu;
import at.lumetsnet.caas.model.MenuCategory;
import at.lumetsnet.caas.model.Order;
import at.lumetsnet.caas.model.User;

public class OrderDaoTest extends GenericDaoTest {
```

```java

  private OrderDao dao;
  private MenuDao menuDao;
  private MenuCategoryDao categoryDao;
  private UserDao userDao;

  private long menuId;
  private long userId;

  @Before
  public void setUp() {
    cleanTable("Order");
    cleanTable("Menu");
    cleanTable("MenuCategory");
    cleanTable("User");

    dao = new OrderDaoJdbc(CONNECTION_STRING, USER_NAME, PASSWORD);
    categoryDao = new MenuCategoryDaoJdbc(CONNECTION_STRING, USER_NAME,
        PASSWORD);
    menuDao = new MenuDaoJdbc(CONNECTION_STRING, USER_NAME, PASSWORD);
    userDao = new UserDaoJdbc(CONNECTION_STRING, USER_NAME, PASSWORD);
    MenuCategory cat = new MenuCategory(-1, "Cat1");
    categoryDao.saveOrUpdate(cat);

    Menu menu = new Menu(-1, "desc", 1000, LocalDate.now(), LocalDate.now()
        .plusDays(100), null);
    menu.setCategoryId(cat.getId());
    menuDao.saveOrUpdate(menu);
    menuId = menu.getId();

    User user = new User(-1, "username", "pass", "firstname", "lastname",
        false);
    userDao.saveOrUpdate(user);
    userId = user.getId();
  }

  @Test
  public void insertTest() {

    Order entity = new Order(-1, null, null, LocalDateTime.now(), "comment");
    entity.setMenuId(menuId);
    entity.setUserId(userId);
    dao.saveOrUpdate(entity);
    assertTrue(entity.getId() != -1);
  }

  @Test
  public void updateTest() {
    Order entity = new Order(-1, null, null, LocalDateTime.now(), "comment");
    entity.setMenuId(menuId);
    entity.setUserId(userId);
    dao.saveOrUpdate(entity);
    entity.setComment("new desc");
    dao.saveOrUpdate(entity);
    Order dbEntry = dao.get(entity.getId());
    assertEquals("new desc", dbEntry.getComment());
  }

  @Test
  public void getTest() {
    Order entity = new Order(-1, null, null, LocalDateTime.now(), "comment");
    // remove nano
    entity.setTime(entity.getTime().minusNanos(entity.getTime().getNano()));

    entity.setMenuId(menuId);
    entity.setUserId(userId);
    dao.saveOrUpdate(entity);

    Order dbEntity = dao.get(entity.getId());
    assertEquals(entity.getComment(), dbEntity.getComment());
    assertEquals(entity.getMenuId(), dbEntity.getMenuId());
    assertEquals(entity.getUserId(), dbEntity.getUserId());

    assertEquals(entity.getTime(), dbEntity.getTime());

  }

  @Test
  public void getAllTest() {
    Order entity = new Order(-1, null, null, LocalDateTime.now(), "comment");
    entity.setMenuId(menuId);
    entity.setUserId(userId);
    dao.saveOrUpdate(entity);
    entity = new Order(-1, null, null, LocalDateTime.now(), "comment2");
    entity.setMenuId(menuId);
    entity.setUserId(userId);

    dao.saveOrUpdate(entity);
    assertEquals(2, dao.getAll().size());
  }

  @Test
  public void deleteTest() {
    Order entity = new Order(-1, null, null, LocalDateTime.now(), "comment");
    entity.setMenuId(menuId);
    entity.setUserId(userId);
    dao.saveOrUpdate(entity);
    assertTrue(dao.get(entity.getId()) != null);
    dao.delete(entity.getId());

    assertTrue(dao.get(entity.getId()) == null);
  }

  @Test
  public void getOrdersByDateTest() {
    Order entity = new Order(-1, null, null, LocalDateTime.now(), "today");
    entity.setMenuId(menuId);
    entity.setUserId(userId);
    dao.saveOrUpdate(entity);
    entity = new Order(-1, null, null, LocalDateTime.now().plusDays(1),
        "other");
    entity.setMenuId(menuId);
    entity.setUserId(userId);
    dao.saveOrUpdate(entity);

    Collection<Order> result = dao.getOrdersByDate(LocalDate.now());
    assertEquals(1, result.size());
    assertEquals("today", result.stream().findFirst().orElse(null)
        .getComment());
  }

}
```

UserDaoTest.java

```java
package at.lumetsnet.caas.test;

import static org.junit.Assert.assertEquals;
import static org.junit.Assert.assertTrue;

import org.junit.Before;
import org.junit.Test;

import at.lumetsnet.caas.dal.UserDao;
import at.lumetsnet.caas.dal.UserDaoJdbc;
import at.lumetsnet.caas.model.User;

public class UserDaoTest extends GenericDaoTest {

  private UserDao dao;

  @Before
  public void setUp() {
    cleanTable("Order");
    cleanTable("Menu");
    cleanTable("MenuCategory");
    cleanTable("User");

    dao = new UserDaoJdbc(CONNECTION_STRING, USER_NAME, PASSWORD);
  }

  @Test
  public void insertTest() {

    User entity = new User(-1, "username", "pass", "firstname", "lastname",
        false);
    dao.saveOrUpdate(entity);
    assertTrue(entity.getId() != -1);
  }

  @Test
  public void updateTest() {
    User entity = new User(-1, "username", "pass", "firstname", "lastname",
        false);
    dao.saveOrUpdate(entity);
    entity.setFirstName("newFirstName");
    dao.saveOrUpdate(entity);
    User dbEntry = dao.get(entity.getId());
    assertEquals("newFirstName", dbEntry.getFirstName());
  }

  @Test
  public void getTest() {
    User entity = new User(-1, "username", "pass", "firstname", "lastname",
        false);
    dao.saveOrUpdate(entity);

    User dbUser = dao.get(entity.getId());
    assertEquals(entity.getUserName(), dbUser.getUserName());
    assertEquals(entity.getPassword(), dbUser.getPassword());
    assertEquals(entity.getFirstName(), dbUser.getFirstName());
    assertEquals(entity.getLastName(), dbUser.getLastName());
    assertEquals(entity.isLocked(), dbUser.isLocked());
  }

  @Test
  public void getAllTest() {
    User entity = new User(-1, "username", "pass", "firstname", "lastname",
        false);
    dao.saveOrUpdate(entity);
    entity = new User(-1, "username1", "pass", "firstname1", "lastname1",
        false);
    dao.saveOrUpdate(entity);
    assertEquals(2, dao.getAll().size());
  }

  @Test
  public void deleteTest() {
    User entity = new User(-1, "username", "pass", "firstname", "lastname",
        false);
    dao.saveOrUpdate(entity);
    assertTrue(dao.get(entity.getId()) != null);
    dao.delete(entity.getId());
    assertTrue(dao.get(entity.getId()) == null);
  }

}
```

1.5 Sourcecode - CSS

main-window.css

```
/* styles used for all pages and the main window */
.nav-container {
  -fx-border-width: 0 0 1 0;
  -fx-border-color: black;
  -fx-spacing: 10;
  -fx-padding: 10;
}

.nav-username {
  -fx-font: bold 10pt "Arial";
}

.nav-role {
  -fx-font: 10pt "Arial";
}

.nav-command-container {
  -fx-spacing: 10;
  -fx-alignment: baseline-right;
}

.nav-command {
  -fx-spacing: 10;
  -fx-padding: 10;
  -fx-font: 10pt "Arial";
}

.nav-command-selected {
  -fx-background-color: #4CAF50;
}

#nav-top-icon {
  -fx-padding: 10 0 0 0;
  -fx-graphic: url('ic-info.png');
  -fx-alignment: baseline-center;
  -fx-content-display: center;
}

.table {
  -fx-font: 10pt "Arial";
}

.table-command-container {
  -fx-alignment: baseline-right;
  -fx-spacing: 10;
}

.table-command {
  -fx-background-position: center;
  -fx-content-display:  graphic-only;
  -fx-min-height: 40px;
  -fx-min-width: 40px;
}
.table-command-edit {
  -fx-graphic: url('ic-edit.png');
}

.table-command-delete {
  -fx-graphic: url('ic-delete.png');
}
.table-command-lock {
  -fx-graphic: url('ic-lock.png');
}
.table-command-unlock {
  -fx-graphic: url('ic-unlock.png');
}

.page-content-container {
  -fx-padding: 10;
}

.page-title {
  -fx-padding: 0 0 10 0;
  -fx-font: 14pt Arial;
}

.page-command-container {
  -fx-padding: 10;
  -fx-alignment: baseline-right;
}
.page-command {
  -fx-font: 10pt "Arial";
}
```

dialog.css

```
/* Styles used in Dialogs */
.dialog-top-container {
  -fx-border-width: 0 0 1 0;
  -fx-border-color: black;
  -fx-spacing: 10;
  -fx-padding: 10;
}

.dialog-bottom-container {
  -fx-border-width: 1 0 0 0;
  -fx-border-color: black;
  -fx-spacing: 10;
  -fx-padding: 15;
}

.dialog-title {
  -fx-font: bold 12pt "Arial";
}

.dialog-description {
  -fx-font: 10pt "Arial";
}

.command-button {
  -fx-padding: 10;
  -fx-font: 10pt "Arial";
}

.form-container {
  -fx-padding: 10;
}

.form-label {
  -fx-padding: 5 0 5 0;
  -fx-font: 10pt "Arial";
}

.form-textfield {
  -fx-font: 10pt "Arial";
}

.form-combobox {
  -fx-font: 10pt "Arial";
}
```

```

.form-datepicker {
  -fx-font: 10pt "Arial";
}

.error-label {
  -fx-font: 10pt "Arial";
  -fx-text-fill: Red;
  -fx-padding: 10;
  -fx-content-display: center;
}
```

derror-ialog.css

```
/* styles used for error dialog */
#dialog-icon {
  -fx-padding: 10 0 0 0;
  -fx-graphic: url('ic-error.png');
  -fx-alignment: baseline-center;
  -fx-content-display: center;
}
```

manage-menu-category-dialog.css

```
#dialog-icon {
  -fx-padding: 10 0 0 0;
  -fx-graphic: url('ic-edit.png');
  -fx-alignment: baseline-center;
  -fx-content-display: center;
}
```

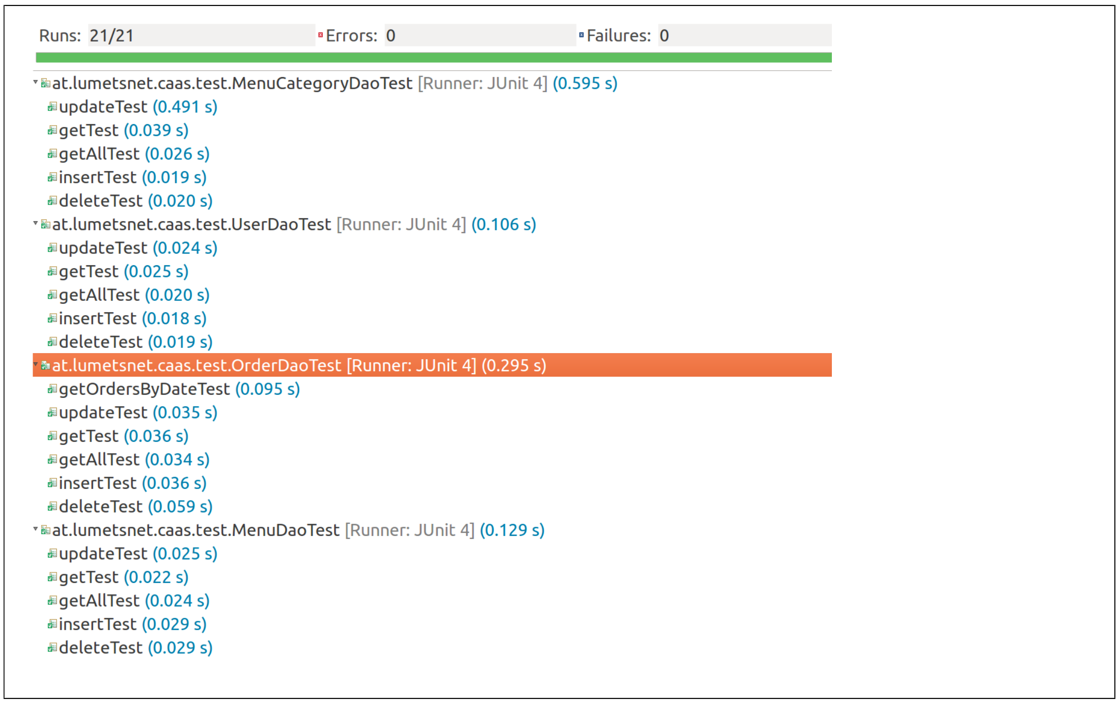
manage-menu-dialog.css

```
/* dialog icon */
#dialog-icon {
  -fx-padding: 10 0 0 0;
  -fx-graphic: url('ic-edit.png');
  -fx-alignment: baseline-center;
  -fx-content-display: center;
}
```

manage-user-dialog.css

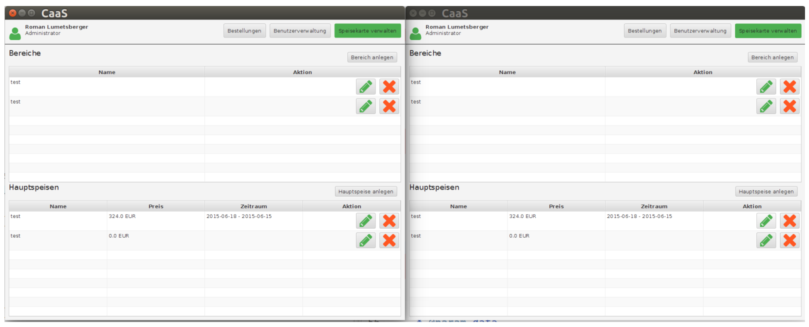
```
/* dialog icon */
#dialog-icon {
  -fx-padding: 10 0 0 0;
  -fx-graphic: url('ic-edit.png');
  -fx-alignment: baseline-center;
  -fx-content-display: center;
}
```

1.6 Screenshots



```
Runs: 21/21          Errors: 0          Failures: 0

at.lumetsnet.caas.test.MenuCategoryDaoTest [Runner: JUnit 4] (0.595 s)
  updateTest (0.491 s)
  getTest (0.039 s)
  getAllTest (0.026 s)
  insertTest (0.019 s)
  deleteTest (0.020 s)
at.lumetsnet.caas.test.UserDaoTest [Runner: JUnit 4] (0.106 s)
  updateTest (0.024 s)
  getTest (0.025 s)
  getAllTest (0.020 s)
  insertTest (0.018 s)
  deleteTest (0.019 s)
at.lumetsnet.caas.test.OrderDaoTest [Runner: JUnit 4] (0.295 s)
  getOrdersByDateTest (0.095 s)
  updateTest (0.035 s)
  getTest (0.036 s)
  getAllTest (0.034 s)
  insertTest (0.036 s)
  deleteTest (0.059 s)
at.lumetsnet.caas.test.MenuDaoTest [Runner: JUnit 4] (0.129 s)
  updateTest (0.025 s)
  getTest (0.022 s)
  getAllTest (0.024 s)
  insertTest (0.029 s)
  deleteTest (0.029 s)
```



Bereiche

Name	Aktion
test	
test	

Hauptspeisen

Name	Preis	Zeitraum	Aktion
test	324.0 EUR	2015-06-18 - 2015-06-15	
test	0.0 EUR		

Bereiche

Name	Aktion
test	
test	

Hauptspeisen

Name	Preis	Zeitraum	Aktion
test	324.0 EUR	2015-06-18 - 2015-06-15	
test	0.0 EUR		