SWE 4x	Übung zu Softwareentwicklung mit modernen Plattformen 4	SS 2015, Übung 6

Abgabetermin: SA in der KW 23

Gr. 1, E. Pitzer
Gr. 2, F. Gruber-Leitner

Name Roman Lumetsberger **Aufwand in h** 12

Punkte _____ **Kurzzeichen Tutor / Übungsleiter** _____ / _____

Wie angekündigt wollen wir in den restlichen Übungen – also von der vorliegenden 6. bis zur 8. Übung – anhand *einer* Aufgabenstellung verschiedene Möglichkeiten der Java-Klassenbibliothek ausloten. Damit Sie die weiteren Übungen meistern können, sollten Sie von Anfang an mitmachen.

CaaS – Campina as a Service: Online-Menübestellung

Kurzbeschreibung

Unsere Haus-und-Hof-Mensa möchte für die Studierenden ein besonderes Service anbieten: Um die Wartezeit auf das Mittagsmenü zu minimieren, kann bereits am Vormittag das Wunschmenü online reserviert werden und kommt dann pünktlich heiß auf den Tisch.

Dafür wird die Erstellung einer Software in Auftrag gegeben, die im Wesentlichen folgende Anforderungen erfüllen soll:

Menüverwaltung

- Wartung der aktuellen Speisekarte
- Benutzerverwaltung (hinzufügen, aktualisieren, sperren von Benutzern)
- Übersicht über getätigte Bestellungen

Online-Bestellung

- Login mit Benutzername und Passwort
- Bestellung eines Menüs für den aktuellen Tag

Menüverwaltung

Die Menüverwaltung steht dem Restaurantbetreiber zur Verfügung und ermöglicht ihm, seine aktuelle Speisekarte zu warten. Die Speisekarte besteht aus verschiedenen Bereichen („Vegetarische Gerichte“, „Aus der Pfanne und vom Grill“, „Fischspezialitäten“ etc.). Für die Online-Bestellung werden nur die Hauptspeisen eingetragen, alle anderen Speisen (Desserts, etc.) können ohnehin rasch serviert werden und brauchen nicht berücksichtigt zu werden. Jede Speise besteht aus der Beschreibung („Wiener Schnitzel mit Petersilkartoffeln“) und einem Preis. Um auch vorübergehende Angebote abbilden zu können, sollen Speisen optional mit einem Beginn- und Endedatum versehen werden können.

Möchten Studierende das Online-Service nutzen, müssen sie sich beim Restaurantbetreiber persönlich melden. Dieser kann dann in der Menüverwaltung neue Benutzer anlegen (Vorname, Nachname, Benutzername, Passwort). Wenn ein Benutzer mehrmals ein Menü bestellt, ohne es abzuholen, so möchte der Betreiber den Benutzer auch wieder sperren können.

In einer Übersichtsanzeige sieht das Restaurant alle Bestellungen des aktuellen Tages (gereiht nach der Uhrzeit, zu der die Benutzer das Menü serviert bekommen wollen) und kann so punktgenau kochen.

Online-Bestellungsplattform

Nach der Anmeldung mit den erhaltenen Zugangsdaten erscheint die aktuelle Speisekarte. Der Benutzer kann aus der Speisekarte eine Hauptspeise auswählen und muss eine Uhrzeit angeben, zu der die Speise fertig sein soll. Optional kann er auch Sonderwünsche bekannt geben – einen „Gruß *an* die Küche“ quasi.

Die Online-Plattform ist allerdings jeden Tag nur bis 11:00 geöffnet, anschließend werden keine Bestellungen mehr angenommen.

Ausbaustufe 1: CaaS-FX (24 Punkte)

Der Restaurantbetreiber möchte vor der endgültigen Auftragserteilung der Online-Plattform zunächst einen Prototyp für die Menüverwaltung sehen.

Entwickeln Sie daher mit Hilfe von JavaFx für die Menüverwaltung einen funktionstüchtigen, ausbaufähigen Benutzeroberflächen-Prototyp. Versuchen Sie eine möglichst intuitiv zu verwendende Benutzeroberfläche zu entwerfen. Setzen Sie dafür Ihr in UEN erworbenes Wissen ein. Trennen Sie den Entwurfs- vom Implementierungsprozess, indem Sie Ihre Benutzeroberfläche zunächst mit Mockups (Grobentwurf der Benutzeroberfläche) modellieren. Fügen Sie die Mockups zu Ihrer Systemdokumentation hinzu.

Trennen Sie den Code zur Realisierung der grafischen Benutzeroberfläche vom Code zur Repräsentation der Daten Ihrer Anwendung (Benutzer, Menüs, Bestellungen etc.). Durch diese Maßnahme wird Ihre Anwendung einfach erweiterbar, was Ihnen in der nächsten Ausbaustufe zugutekommen sollte. Die Verwendung von FXML und Werkzeugen zum Design der grafischen Benutzeroberfläche (JavaFX Scene Builder) ist nicht erlaubt.

Ihr Prototyp soll es ermöglichen, alle Fenster und Dialoge der Benutzeroberfläche zu öffnen (und diese auch wieder zu schließen). Die Anwendung muss die Eingaben aber noch nicht über die Programmlaufzeit hinaus speichern können. Als Ersatz dafür können Sie mit hart codierten Daten arbeiten. Der Benutzeroberflächen-Prototyp muss folgende Funktionen abdecken:

- Benutzer hinzufügen/sperren, Benutzerdaten aktualisieren
- Speisekarte: Bereiche erstellen/löschen
- Speisekarte: Hauptspeisen hinzufügen/löschen
- Anzeige der heutigen Bestellungen

Beachten Sie, dass die Implementierung der anderen Systemkomponenten erst in weiteren Ausbaustufen gefordert ist. Die genauen technischen Anforderungen an diese Komponenten werden im weiteren Verlauf der Übung bekannt gegeben.

1 CaaS – Campina as a Service

1.1 Lösungsidee

Da bei dieser Aufgabe ein GUI Prototyp gefordert ist, ist es besonders wichtig die Business-Logik nicht mit der GUI zu vermischen.
Ein Ansatz dies zu erreichen ist es, die Business-Logik in eigene Klassen auszulagern.
Diese können dann als Singleton verwendet werden.
Weiters sollte die GUI mit eigenen Objekten arbeiten um nicht von der Struktur der Datenhaltung abhängig zu sein.

1.1.1 Businesslogik

Alle benötigten Business-Methoden werden in eigene Klassen ausgelagert. Diese implementieren das *Singleton* Entwurfsmuster.

1.1.2 ViewModel

Die Daten für die GUI werden in eigenen *ViewModel* Klassen gekapselt. Dadurch ist man unabhängig von der externen Struktur.

1.1.3 Pages

Die Oberfläche kann mit Pages umgesetzt werden. Eine Page wird im mittleren Bereich des Hauptfensters angezeigt.
Über die Menübuttons kann zwischen den Pages gewechselt werden. (siehe Mockups)

1.1.4 Dialoge

Zum Editieren und Anlegen von Elementen können Dialoge verwendet werden.

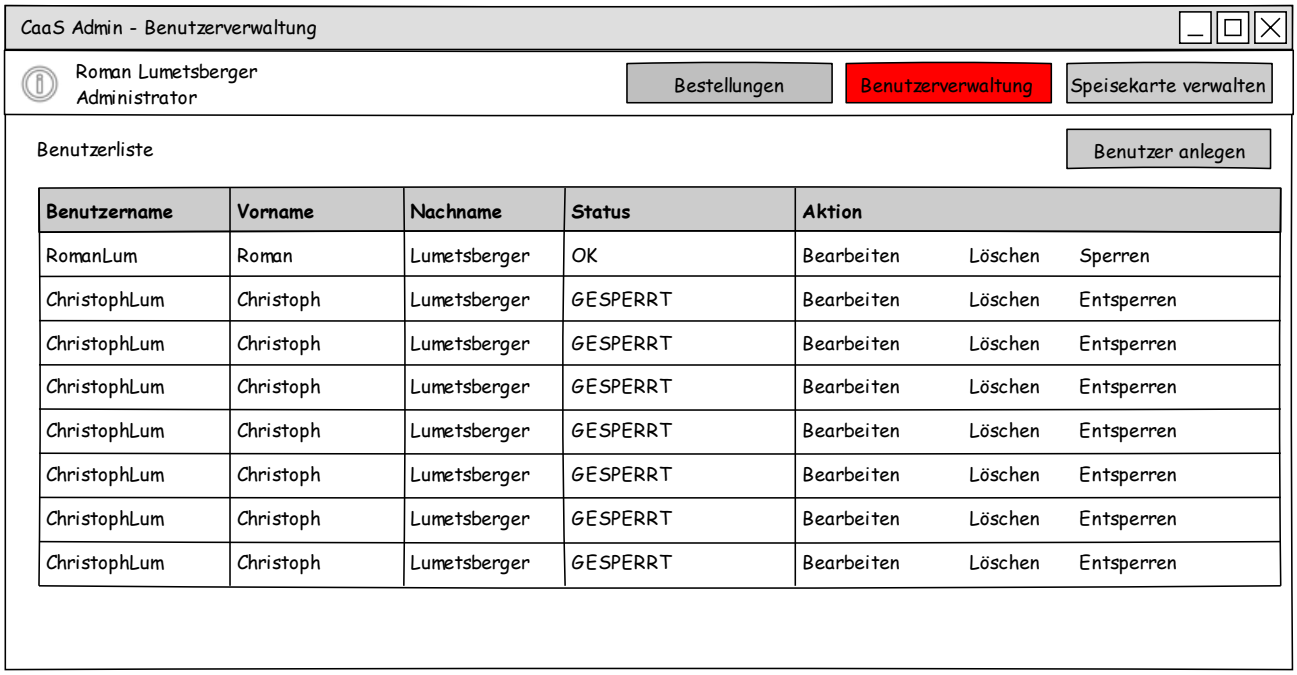
1.1.5 Design

Das Design sollte mit CSS gesteuert werden können, d. h. im Code sollten nur die benötigten Elemente angelegt und CSS-Klassen vergeben werden. Diese Klassen können dann in der CSS Datei nach Belieben geändert werden.

GUI

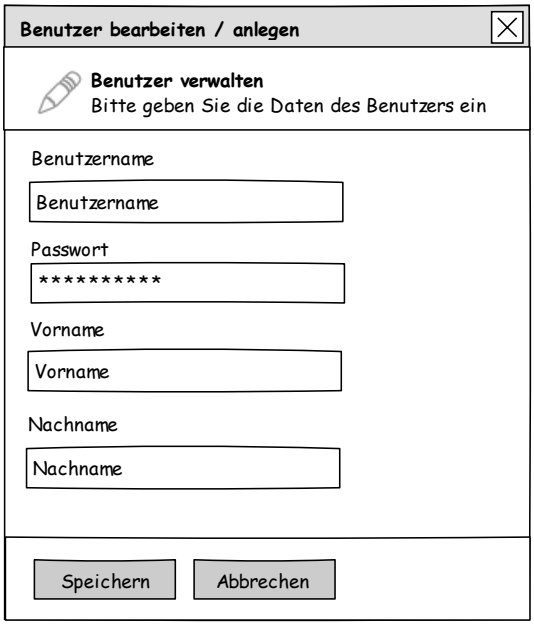
Exported at: Thu Jun 04 2015 16:30:29 GMT+0200

CaaS - Benutzerverwaltung

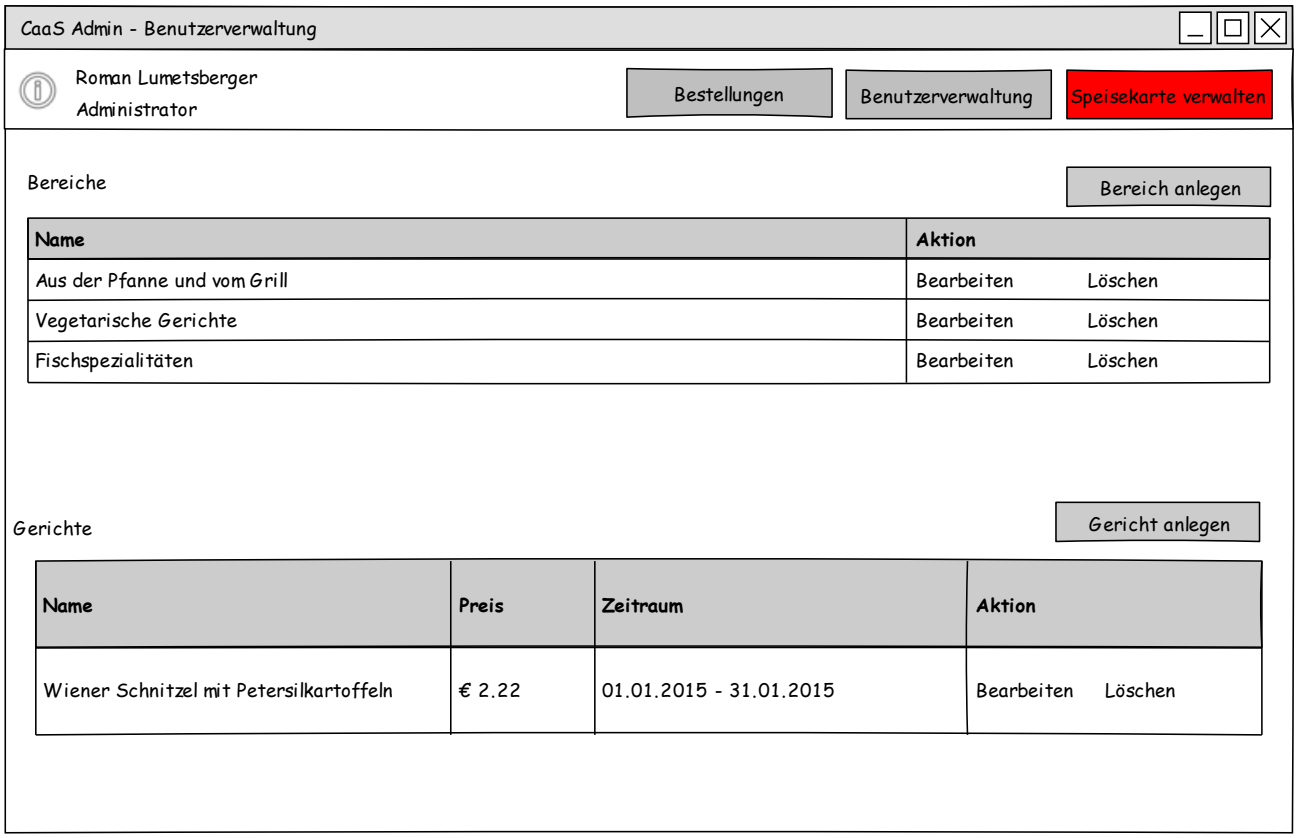


Benutzername	Vorname	Nachname	Status	Aktion		
RomanLum	Roman	Lumetsberger	OK	Bearbeiten	Löschen	Sperren
ChristophLum	Christoph	Lumetsberger	GESPERRT	Bearbeiten	Löschen	Entsperren
ChristophLum	Christoph	Lumetsberger	GESPERRT	Bearbeiten	Löschen	Entsperren
ChristophLum	Christoph	Lumetsberger	GESPERRT	Bearbeiten	Löschen	Entsperren
ChristophLum	Christoph	Lumetsberger	GESPERRT	Bearbeiten	Löschen	Entsperren
ChristophLum	Christoph	Lumetsberger	GESPERRT	Bearbeiten	Löschen	Entsperren
ChristophLum	Christoph	Lumetsberger	GESPERRT	Bearbeiten	Löschen	Entsperren
ChristophLum	Christoph	Lumetsberger	GESPERRT	Bearbeiten	Löschen	Entsperren

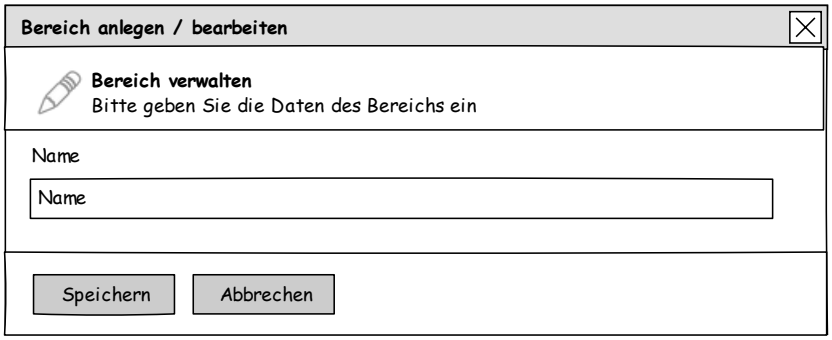
CaaS - Benutzerverwaltung II

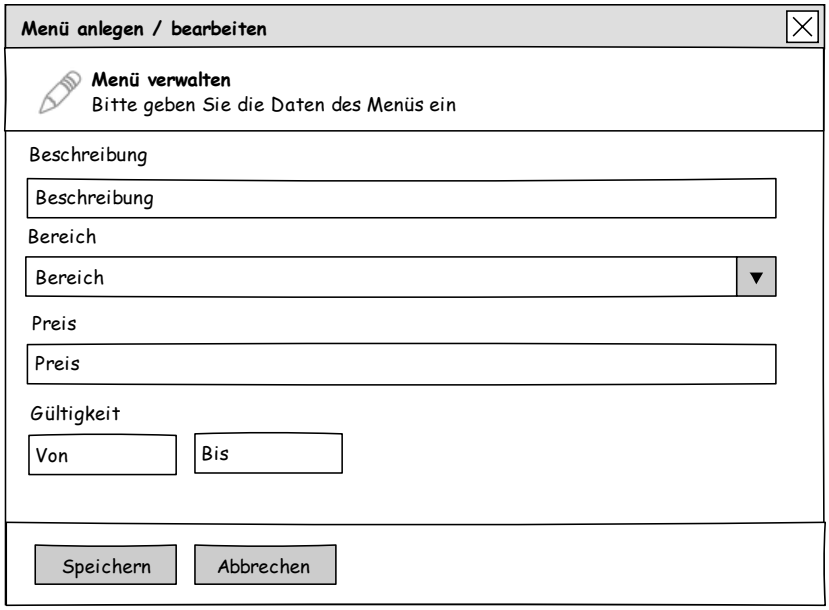


CaaS - Speisekarte verwalten

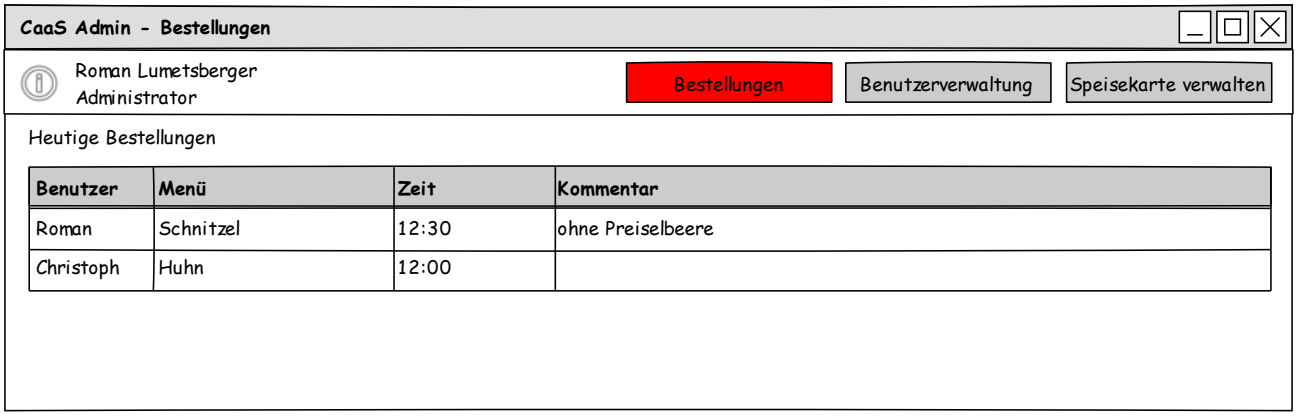


CaaS - Speisekarte verwalten II





Caas - Bestellungen



1.2 Sourcecode - Java

Main.java

```
1  package at.lumetsnet.caas;
2
3  import java.util.logging.LogManager;
4
5  import javafx.application.Application;
6  import javafx.stage.Stage;
7  import at.lumetsnet.caas.gui.MainWindow;
8  import at.lumetsnet.caas.model.User;
9
10 public class Main extends Application {
11
12   @Override
13   public void start(Stage stage) throws Exception {
14     User user = new User();
15     user.setFirstName("Roman");
16     user.setLastName("Lumetsberger");
17
18     MainWindow wnd = new MainWindow(stage, user);
19     wnd.show();
20
21   }
22
23   public static void main(String[] args) {
24     LogManager.getLogManager().reset();
25     launch(args);
26
27   }
28
29 }
```

MenuService.java

```
1  package at.lumetsnet.caas.business;
2
3  import java.time.LocalDate;
4  import java.util.ArrayList;
5  import java.util.Collection;
6
7  import at.lumetsnet.caas.model.Menu;
8  import at.lumetsnet.caas.model.MenuCategory;
9
10 /***
11  * Mock menu business logic class
12  * Used as singleton, should be replaced with
```

```
13  * real logic class
14  * @author romanlum
15  *
16  */
17 public class MenuService {
18
19   ArrayList<MenuCategory> categories;
20   ArrayList<Menu> menus;
21
22   private static MenuService instance = null;
23
24   private MenuService() {
25     categories = new ArrayList<>();
26     categories.add(new MenuCategory(0, "Vegetarisch"));
27     categories.add(new MenuCategory(1, "Fleisch und Fisch"));
28
29     menus = new ArrayList<>();
30     menus.add(new Menu(0, "Wienerschnitzel", 1000, LocalDate.now(),
31         LocalDate.now(), categories.get(0)));
32     menus.add(new Menu(1, "Gericht 2", 20000, LocalDate.now(), LocalDate
33         .now(), categories.get(1)));
34   }
35
36   /***
37    * Singleton instance class
38    * @return
39    */
40   public static MenuService getInstance() {
41     if (instance == null) {
42       instance = new MenuService();
43     }
44     return instance;
45   }
46
47   /***
48    * Fetches all categories
49    * @return
50    */
51   public Collection<MenuCategory> getAllCategories() {
52     return categories;
53   }
54
55   /***
56    * Deletes the category with the given id
57    * @param id
58    */
59   public void deleteCategory(long id) {
60     categories.removeIf(x -> x.getId() == id);
61   }
```

```
62
63    /***
64     * Deletes the menu with the given id
65     * @param id
66     */
67    public void deleteMenu(long id) {
68      menus.removeIf(x -> x.getId() == id);
69    }
70
71    /***
72     * Saves or adds the category
73     * @param data
74     */
75    public void saveOrUpdateCategory(MenuCategory data) {
76      Util.saveOrUpdate(data, categories);
77    }
78
79    /***
80     * Saves or adds teh menu
81     * @param data
82     */
83    public void saveOrUpdateMenu(Menu data) {
84      Util.saveOrUpdate(data, menus);
85    }
86
87    /***
88     * Fetches all menus
89     * @return
90     */
91    public Collection<Menu> getAllMenus() {
92      return menus;
93    }
94
95  }
```

OrderService.java

```
1  package at.lumetsnet.caas.business;
2
3  import java.time.LocalDateTime;
4  import java.util.ArrayList;
5  import java.util.Collection;
6  import java.util.List;
7
8  import at.lumetsnet.caas.model.Menu;
9  import at.lumetsnet.caas.model.MenuCategory;
10 import at.lumetsnet.caas.model.Order;
11 import at.lumetsnet.caas.model.User;
12
```

```
13 /***
14  * Mock order business logic class
15  * Used as singleton, should be replaced with
16  * real logic class
17  * @author romanlum
18  *
19  */
20
21 public class OrderService {
22
23   private static OrderService instance = null;
24
25   private OrderService() {
26
27   }
28
29   /**
30      * singleton instance
31      */
32   public static OrderService getInstance() {
33     if (instance == null) {
34       instance = new OrderService();
35     }
36     return instance;
37   }
38
39   /***
40    * Fetches the orders of today
41    * @return
42    */
43   public Collection<Order> getTodaysOrders() {
44     List<Order> orders = new ArrayList<Order>();
45
46     User user = new User(1, "romanlum", "", "Roman", "Lumetsberger", false,
47         true);
48
49     Menu menu = new Menu(1, "Wienerschnitzel mit Pommes", 0, null, null,
50         new MenuCategory());
51
52     LocalDateTime time = LocalDateTime.now();
53
54     orders.add(new Order(1, menu, user,time ,
55         "ohne Preiselbeeren"));
56     time = time.minusHours(1);
57     user = new User(1, "christophlum", "", "Christoph", "Lumetsberger", false,
58         true);
59     orders.add(new Order(1, menu, user,time,
60         "mit Preiselbeeren"));
61     time = time.minusMinutes(1);
```

```
62      user = new User(1, "romanlum", "", "Moe", "Sislec", false,
63          true);
64      orders.add(new Order(1, menu, user,time,
65          "normal"));
66      return orders;
67    }
68  }
```

UserService.java

```
1  package at.lumetsnet.caas.business;
2
3  import java.util.ArrayList;
4  import java.util.Collection;
5  import java.util.Optional;
6
7  import at.lumetsnet.caas.model.User;
8
9  /***
10  * Mock user business logic class
11  * Used as singleton, should be replaced with
12  * real logic class
13  * @author romanlum
14  *
15  */
16 public class UserService {
17
18   ArrayList<User> users;
19
20   private static UserService instance = null;
21
22   private UserService() {
23     users = new ArrayList<>();
24     users.add(new User(0, "admin", "admin", "Roman", "Lumetsberger", false,
25         true));
26     users.add(new User(1, "romanlum", "password", "Roman", "Lumetsberger",
27         false, true));
28     users.add(new User(2, "christophlum", "password", "Christoph",
29         "Lumetsberger", true, false));
30     users.add(new User(3, "moe", "password", "Moe", "Sislec", false, true));
31
32   }
33
34   /**
35      * Singleton instance class
36      */
37   public static UserService getInstance() {
38     if (instance == null) {
39       instance = new UserService();
```

```
40      }
41      return instance;
42    }
43
44    /***
45     * Fetches all users
46     * @return
47     */
48    public Collection<User> getAllUsers() {
49      return users;
50    }
51
52    /***
53     * delete the user
54     * @param id
55     */
56    public void deleteUser(long id) {
57      users.removeIf(x -> x.getId() == id);
58    }
59
60    /***
61     * Toggles the locked state of a user
62     * @param id
63     */
64    public void toggleLockState(long id) {
65      Optional<User> result = users.stream().filter(x -> x.getId() == id)
66          .findFirst();
67      if (result.isPresent()) {
68        result.get().setLocked(!result.get().isLocked());
69      }
70    }
71
72    /***
73     * Saves or adds the user
74     * @param userModel
75     */
76    public void saveOrUpdate(User userModel) {
77      Util.saveOrUpdate(userModel, users);
78    }
79  }
```

Util.java

```
1  package at.lumetsnet.caas.business;
2
3  import java.util.ArrayList;
4  import java.util.Optional;
5  import java.util.OptionalLong;
6
```

```java
import at.lumetsnet.caas.model.Entity;

/***
 * Utility class used for mock services
 * @author romanlum
 *
 */
public class Util {

  /***
   * Adds or replaces an object in the given list
   * @param model
   * @param list
   */
  public static <T extends Entity> void saveOrUpdate(T model,
      ArrayList<T> list) {

    if (model.getId() == -1) {
      // new item
      long max = list.stream().mapToLong(x -> x.getId()).max().orElse(-1);
      model.setId(max + 1);
      list.add(model);
    } else {
      // replace old one
      Optional<T> old = list.stream()
          .filter(x -> x.getId() == model.getId()).findAny();
      list.set(list.indexOf(old.get()), model);

    }

  }

}
```

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
 *
```

```
14  * @param <S>
15  * @param <T>
16  */
17 public class ActionTableCell<S, T> extends TableCell<S, T> {
18   protected HBox box = new HBox();
19   protected Button editButton = new Button("Bearbeiten");
20   protected Button deleteButton = new Button("Löschen");
21
22   protected Consumer<T> editAction;
23   protected Consumer<T> deleteAction;
24
25   public ActionTableCell(Consumer<T> editAction, Consumer<T> deleteAction) {
26     this.editAction = editAction;
27     this.deleteAction = deleteAction;
28
29     box.getStyleClass().add("table-command-container");
30     editButton.getStyleClass()
31         .addAll("table-command", "table-command-edit");
32     deleteButton.getStyleClass().addAll("table-command",
33         "table-command-delete");
34     box.getChildren().addAll(editButton, deleteButton);
35   }
36
37   @Override
38   protected void updateItem(T entity, boolean empty) {
39
40     if (entity != null) {
41       editButton.setOnAction(x -> {
42         if (editAction != null)
43           editAction.accept(entity);
44
45       });
46       deleteButton.setOnAction(x -> {
47         if (deleteAction != null)
48           deleteAction.accept(entity);
49       });
50       setGraphic(box);
51     } else {
52       setGraphic(null);
53     }
54   }
55 }
```

AmountTableCell.java

```
1 package at.lumetsnet.caas.gui;
2
3 import javafx.scene.control.TableCell;
4
```

```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("");
    }
  }
}
```

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 {
```

```
24        setText("");
25      }
26    }
27  }
```

MainWindow.java

```
1  package at.lumetsnet.caas.gui;
2
3  import java.util.HashMap;
4
5  import javafx.scene.Node;
6  import javafx.scene.Scene;
7  import javafx.scene.control.Button;
8  import javafx.scene.control.Label;
9  import javafx.scene.layout.BorderPane;
10 import javafx.scene.layout.HBox;
11 import javafx.scene.layout.Pane;
12 import javafx.scene.layout.Priority;
13 import javafx.scene.layout.VBox;
14 import javafx.stage.Stage;
15 import javafx.stage.Window;
16 import at.lumetsnet.caas.gui.pages.ManageMenusPage;
17 import at.lumetsnet.caas.gui.pages.ManageUsersPage;
18 import at.lumetsnet.caas.gui.pages.OrdersPage;
19 import at.lumetsnet.caas.gui.pages.Showable;
20 import at.lumetsnet.caas.model.User;
21
22 /***
23  * Main windows class
24  * Creates all pages and manages page switching
25  * @author romanlum
26  *
27  */
28 public class MainWindow {
29
30   private enum VIEW_TYPE {
31     ADMIN_ORDERS, ADMIN_USERS, ADMIN_MENUS
32   }
33
34   private User user;
35   private Stage stage = new Stage();
36   private HashMap<VIEW_TYPE, Pane> pages;
37   private BorderPane container;
38
39   private Button adminOrdersViewButton;
40   private Button adminUsersViewButton;
41   private Button adminMenusViewButton;
42
```

```
43  public MainWindow(Window owner, User user) {
44    this.user = user;
45    container = new BorderPane();
46    container.getStyleClass().add("window-container");
47    container.setTop(createTopMenu());
48
49    pages = createPages();
50
51    Scene scene = new Scene(container);
52    scene.getStylesheets().add(
53        getClass().getResource("css/main-window.css").toExternalForm());
54
55    stage.initOwner(owner);
56    stage.setScene(scene);
57    stage.setTitle("CaaS");
58    //start with 1024/768
59    stage.setWidth(1024);
60    stage.setHeight(768);
61    switchView(VIEW_TYPE.ADMIN_ORDERS);
62
63  }
64
65  private HashMap<VIEW_TYPE, Pane> createPages() {
66    HashMap<VIEW_TYPE, Pane> pages = new HashMap<>();
67    pages.put(VIEW_TYPE.ADMIN_ORDERS, new OrdersPage());
68    pages.put(VIEW_TYPE.ADMIN_USERS, new ManageUsersPage());
69    pages.put(VIEW_TYPE.ADMIN_MENUS, new ManageMenusPage());
70
71    return pages;
72  }
73
74  public void show() {
75
76    stage.show();
77  }
78
79  private Node createTopMenu() {
80    HBox topContainer = new HBox();
81    topContainer.getStyleClass().add("nav-container");
82
83    Label iconLabel = new Label();
84    iconLabel.setId("nav-top-icon");
85    VBox descBox = new VBox();
86
87    Label userNameLabel = new Label();
88    userNameLabel.getStyleClass().add("nav-username");
89    userNameLabel.setText(user.getFirstName() + " " + user.getLastName());
90
91    Label descLabel = new Label();
```

```
92      descLabel.getStyleClass().add("nav-role");
93      descLabel.setText("Administrator");
94
95      descBox.getChildren().add(userNameLabel);
96      descBox.getChildren().add(descLabel);
97
98      HBox commandBox = new HBox();
99      commandBox.getStyleClass().add("nav-command-container");
100
101     adminOrdersViewButton = new Button();
102     adminOrdersViewButton.getStyleClass().add("nav-command");
103     adminOrdersViewButton.setText("Bestellungen");
104     adminOrdersViewButton
105         .setOnAction(x -> switchView(VIEW_TYPE.ADMIN_ORDERS));
106     commandBox.getChildren().add(adminOrdersViewButton);
107
108     adminUsersViewButton = new Button();
109     adminUsersViewButton.getStyleClass().add("nav-command");
110     adminUsersViewButton.setText("Benutzerverwaltung");
111     adminUsersViewButton
112         .setOnAction(x -> switchView(VIEW_TYPE.ADMIN_USERS));
113     commandBox.getChildren().add(adminUsersViewButton);
114
115     adminMenusViewButton = new Button();
116     adminMenusViewButton.getStyleClass().add("nav-command");
117     adminMenusViewButton.setText("Speisekarte verwalten");
118     adminMenusViewButton
119         .setOnAction(x -> switchView(VIEW_TYPE.ADMIN_MENUS));
120     commandBox.getChildren().add(adminMenusViewButton);
121
122     HBox.setHgrow(commandBox, Priority.ALWAYS);
123
124     topContainer.getChildren().add(iconLabel);
125     topContainer.getChildren().add(descBox);
126     topContainer.getChildren().add(commandBox);
127     return topContainer;
128   }
129
130   /***
131    * Switches the view
132    * @param type
133    */
134   private void switchView(VIEW_TYPE type) {
135     adminOrdersViewButton.getStyleClass().remove("nav-command-selected");
136     adminUsersViewButton.getStyleClass().remove("nav-command-selected");
137     adminMenusViewButton.getStyleClass().remove("nav-command-selected");
138
139     Button selected = null;
140     switch (type) {
```

```
141     case ADMIN_USERS:
142       selected = adminUsersViewButton;
143 
144       break;
145     case ADMIN_ORDERS:
146       selected = adminOrdersViewButton;
147       break;
148     case ADMIN_MENUS:
149       selected = adminMenusViewButton;
150       break;
151     }
152     if (selected != null)
153       selected.getStyleClass().add("nav-command-selected");
154 
155     Pane page = pages.get(type);
156     if (page instanceof Showable)
157       ((Showable) page).show();
158     container.setCenter(page);
159 
160   }
161 
162 }
```

ManageUserActionCell.java

```
1  package at.lumetsnet.caas.gui;
2  
3  import java.util.function.Consumer;
4  
5  import javafx.scene.control.Button;
6  import at.lumetsnet.caas.viewmodel.UserViewModel;
7  
8  /***
9   * TableCell used for displaying additional lock/unlock
10  * button inside the TableView
11  * @author romanlum
12  *
13  * @param <S>
14  * @param <T>
15  */
16 public class ManageUserActionCell<S, T> extends ActionTableCell<S, T> {
17 
18   protected Button lockButton = new Button("Sperren");
19   protected Consumer<T> lockAction;
20 
21   public ManageUserActionCell(Consumer<T> editAction,
22       Consumer<T> deleteAction, Consumer<T> lockAction) {
23     super(editAction, deleteAction);
24     this.lockAction = lockAction;
```

```
25      box.getChildren().addAll(lockButton);
26    }
27
28    @Override
29    protected void updateItem(T entity, boolean empty) {
30
31      if (entity != null) {
32        lockButton.setOnAction(x -> {
33          if (lockAction != null)
34            lockAction.accept(entity);
35
36        });
37        if (getTableRow() != null && getTableRow().getItem() != null) {
38          Object model = getTableRow().getItem();
39          if (model instanceof UserViewModel) {
40            lockButton.getStyleClass().removeIf(
41                x -> x.startsWith("table-command"));
42            if (((UserViewModel) model).getLockedProperty().get()) {
43              lockButton.setText("Entsperren");
44              lockButton.getStyleClass().addAll("table-command",
45                  "table-command-unlock");
46            } else {
47              lockButton.setText("Sperren");
48              lockButton.getStyleClass().addAll("table-command",
49                  "table-command-lock");
50            }
51          }
52        }
53
54      }
55      super.updateItem(entity, empty);
56    }
57
58  }
```

Util.java

```
1  package at.lumetsnet.caas.gui;
2
3  import java.util.ArrayList;
4  import java.util.Collection;
5
6  import javafx.beans.property.ObjectProperty;
7  import javafx.beans.property.StringProperty;
8  import javafx.beans.value.ChangeListener;
9  import javafx.beans.value.ObservableValue;
10 import javafx.collections.ObservableList;
11 import javafx.scene.Node;
12 import javafx.scene.control.ComboBox;
```

```
13 import javafx.scene.control.Label;
14 import javafx.scene.control.PasswordField;
15 import javafx.scene.control.TextField;
16
17 /***
18  * Common gui utils
19  * @author romanlum
20  *
21  */
22 public class Util {
23
24   /***
25    * Creates a form label
26    * @param text
27    * @return
28    */
29   public static Node getFormLabel(String text) {
30     Label lab = new Label();
31     lab.setText(text);
32     lab.getStyleClass().add("form-label");
33     return lab;
34   }
35
36   /***
37    * Creates a text field form and label
38    * @param text
39    * @param property
40    * @return
41    */
42   public static Collection<Node> getTextFieldForm(String text,
43       StringProperty property) {
44
45     Collection<Node> nodes = new ArrayList<Node>();
46     nodes.add(getFormLabel(text));
47
48     TextField field = new TextField();
49     field.getStyleClass().add("form-textfield");
50     field.textProperty().bindBidirectional(property);
51     nodes.add(field);
52     return nodes;
53   }
54
55   /***
56    * Creates a password form field and label
57    * @param text
58    * @param property
59    * @return
60    */
61   public static Collection<Node> getPasswordFieldForm(String text,
```

```
62        StringProperty property) {
63
64      Collection<Node> nodes = new ArrayList<Node>();
65      nodes.add(getFormLabel(text));
66
67      PasswordField field = new PasswordField();
68      field.getStyleClass().add("form-textfield");
69      field.textProperty().bindBidirectional(property);
70      nodes.add(field);
71      return nodes;
72    }
73
74    /***
75     * Creates a combo box form and label
76     * @param text
77     * @param property
78     * @param data
79     * @return
80     */
81    public static <T> Collection<Node> getComboboxForm(String text,
82        ObjectProperty<T> property, ObservableList<T> data) {
83
84      Collection<Node> nodes = new ArrayList<Node>();
85      nodes.add(getFormLabel(text));
86
87      ComboBox<T> field = new ComboBox<>();
88      field.getStyleClass().add("form-combobox");
89      field.setItems(data);
90      field.getSelectionModel().selectedItemProperty()
91          .addListener(new ChangeListener<T>() {
92            @Override
93            public void changed(ObservableValue<? extends T> item,
94                T arg1, T arg2) {
95              property.setValue(item.getValue());
96
97            }
98          });
99      field.selectionModelProperty().get().select(property.get());
100     nodes.add(field);
101     return nodes;
102   }
103
104 }
```

Dialog.java

```
1 package at.lumetsnet.caas.gui.dialogs;
2
3 import javafx.scene.Node;
```

```
4  import javafx.scene.Scene;
5  import javafx.scene.control.Label;
6  import javafx.scene.layout.BorderPane;
7  import javafx.scene.layout.HBox;
8  import javafx.scene.layout.Pane;
9  import javafx.scene.layout.VBox;
10 import javafx.stage.Modality;
11 import javafx.stage.Stage;
12 import javafx.stage.StageStyle;
13 import javafx.stage.Window;
14
15 /***
16  * Dialog base class used for all dialogs
17  * @author romanlum
18  *
19  */
20 public abstract class Dialog {
21
22   protected Stage dialogStage = new Stage();
23
24   protected final void createGui(Window owner, String title,
25       String description) {
26     BorderPane layout = new BorderPane();
27     layout.getStyleClass().add("dialog-container");
28     layout.setTop(createTopPane(title, description));
29     layout.setCenter(createContentPane());
30     layout.setBottom(createBottomPane());
31
32     Scene dialogScene = new Scene(layout);
33     dialogScene.getStylesheets().add(
34         getClass().getResource("../css/dialog.css").toExternalForm());
35
36     dialogStage.initOwner(owner);
37     dialogStage.setScene(dialogScene);
38     dialogStage.setTitle(title);
39     dialogStage.initModality(Modality.WINDOW_MODAL);
40     dialogStage.initStyle(StageStyle.UTILITY);
41     dialogStage.setResizable(false);
42   }
43
44   protected Node createTopPane(String title, String description) {
45     HBox box = new HBox();
46     box.getStyleClass().add("dialog-top-container");
47
48     Label iconLabel = new Label();
49     iconLabel.setId("dialog-icon");
50     VBox descBox = new VBox();
51
52     Label titleLabel = new Label();
```

```
53      titleLabel.getStyleClass().add("dialog-title");
54      titleLabel.setText(title);
55
56      Label descLabel = new Label();
57      descLabel.getStyleClass().add("dialog-description");
58      descLabel.setText(description);
59
60      descBox.getChildren().add(titleLabel);
61      descBox.getChildren().add(descLabel);
62
63      box.getChildren().add(iconLabel);
64      box.getChildren().add(descBox);
65      return box;
66    }
67
68    protected Pane createBottomPane() {
69      HBox box = new HBox();
70      box.getStyleClass().add("dialog-bottom-container");
71      return box;
72
73    }
74
75    /***
76     * Content used for the dialog
77     * @return
78     */
79    protected abstract Node createContentPane();
80
81  }
```

ErrorDialog.java

```
1  package at.lumetsnet.caas.gui.dialogs;
2
3  import javafx.scene.Node;
4  import javafx.scene.control.Button;
5  import javafx.scene.layout.Pane;
6  import javafx.stage.Window;
7
8  /***
9   * Own dialog class used because
10  * javafx does not have dialogs before update 40
11  * :(
12  * @author romanlum
13  *
14  */
15 public class ErrorDialog extends Dialog {
16
17   public ErrorDialog(Window owner) {
```

```
18
19      createGui(owner, "Fehler", "Bitte korrigieren Sie Ihre Eingaben.");
20
21      dialogStage
22          .getScene()
23          .getStylesheets()
24          .add(getClass().getResource("../css/error-dialog.css")
25              .toExternalForm());
26    }
27
28    @Override
29    protected Pane createBottomPane() {
30      Pane pane = super.createBottomPane();
31      Button button = new Button();
32      button.setText("OK");
33      button.getStyleClass().add("command-button");
34      button.setOnAction(x -> okCommand());
35      pane.getChildren().add(button);
36      return pane;
37    }
38
39    private void okCommand() {
40      dialogStage.close();
41    }
42
43    @Override
44    protected Node createContentPane() {
45      return null;
46    }
47
48    public boolean show() {
49      dialogStage.showAndWait();
50      return true;
51    }
52
53    public static boolean show(Window owner) {
54      ErrorDialog dialog = new ErrorDialog(owner);
55      return dialog.show();
56
57    }
58  }
```

ManageEntityDialog.java

```
1  package at.lumetsnet.caas.gui.dialogs;
2
3  import javafx.beans.property.BooleanProperty;
4  import javafx.beans.property.SimpleBooleanProperty;
5  import javafx.scene.control.Button;
```

```
6  import javafx.scene.layout.Pane;
7  import at.lumetsnet.caas.viewmodel.Validatable;
8
9  /***
10  * Base dialog used for editing a viewmodel entity
11  * @author romanlum
12  *
13  * @param <T>
14  */
15 public abstract class ManageEntityDialog<T> extends Dialog {
16
17   protected boolean canceled = false;
18   protected T viewModel;
19   protected BooleanProperty validationErrorProperty = new SimpleBooleanProperty();
20
21   /***
22    * Shows the dialog and waits for exit
23    * returns if the action was canceled
24    * @return
25    */
26   public boolean show() {
27     dialogStage.showAndWait();
28     return canceled;
29   }
30
31   /***
32    * Save action
33    */
34   protected abstract void saveCommand();
35
36   /***
37    * Cancel action
38    * Default: closes the dialog and sets canceled flag
39    */
40   protected void cancelCommand() {
41     canceled = true;
42     dialogStage.close();
43
44   }
45
46   /***
47    * Validates the viewmodel if it is Validatable
48    * Shows error dialog if validation fails
49    * @return true if the viewmodel is not validatable
50    *         true on successfull validation
51    *         otherwise false
52    */
53   protected boolean validate() {
54     if (!(viewModel instanceof Validatable)) {
```

```
55        return true;
56      }
57
58      boolean result = ((Validatable) viewModel).validate();
59      if (!result) {
60        ErrorDialog.show(dialogStage);
61      }
62      return result;
63    }
64
65    @Override
66    protected Pane createBottomPane() {
67      Pane pane = super.createBottomPane();
68      Button button = new Button();
69      button.setText("Speichern");
70      button.getStyleClass().add("command-button");
71      button.setOnAction(x -> saveCommand());
72      pane.getChildren().add(button);
73
74      button = new Button();
75      button.setText("Abbrechen");
76      button.getStyleClass().add("command-button");
77      button.setOnAction(x -> cancelCommand());
78      pane.getChildren().add(button);
79
80      return pane;
81    }
82
83  }
```

ManageMenuCategoryDialog.java

```
1  package at.lumetsnet.caas.gui.dialogs;
2
3  import javafx.scene.Node;
4  import javafx.scene.layout.VBox;
5  import javafx.stage.Window;
6  import at.lumetsnet.caas.business.MenuService;
7  import at.lumetsnet.caas.gui.Util;
8  import at.lumetsnet.caas.model.MenuCategory;
9  import at.lumetsnet.caas.viewmodel.MenuCategoryViewModel;
10
11 /***
12  * Edit/Add dialog for MenuCategories
13  * @author romanlum
14  *
15  */
16 public class ManageMenuCategoryDialog extends
17     ManageEntityDialog<MenuCategoryViewModel> {
```

```
18
19    public ManageMenuCategoryDialog(Window owner, MenuCategory data) {
20      viewModel = new MenuCategoryViewModel(data);
21      createGui(owner, "Bereich verwalten",
22          "Bitte geben Sie die Daten des Bereichs ein.");
23
24      dialogStage
25          .getScene()
26          .getStylesheets()
27          .add(getClass().getResource(
28              "../css/manage-menu-category-dialog.css")
29              .toExternalForm());
30    }
31
32    @Override
33    protected Node createContentPane() {
34
35      VBox box = new VBox();
36      box.getStyleClass().add("form-container");
37      box.getChildren().addAll(
38          Util.getTextFieldForm("Name", viewModel.getNameProperty()));
39
40      return box;
41    }
42
43    @Override
44    protected void saveCommand() {
45      if (validate()) {
46        //Saves the entity
47        MenuService.getInstance().saveOrUpdateCategory(
48            viewModel.toCategoryModel());
49        dialogStage.close();
50      }
51
52    }
53
54    /***
55     * Static method for showing the dialog
56     * @param owner
57     * @param model
58     * @return
59     */
60    public static boolean show(Window owner, MenuCategory model) {
61      ManageMenuCategoryDialog dialog = new ManageMenuCategoryDialog(owner,
62          model);
63      return dialog.show();
64
65    }
66  }
```

ManageMenuDialog.java

```
1  package at.lumetsnet.caas.gui.dialogs;
2
3  import javafx.collections.FXCollections;
4  import javafx.collections.ObservableList;
5  import javafx.scene.Node;
6  import javafx.scene.control.DatePicker;
7  import javafx.scene.layout.HBox;
8  import javafx.scene.layout.VBox;
9  import javafx.stage.Window;
10 import at.lumetsnet.caas.business.MenuService;
11 import at.lumetsnet.caas.gui.Util;
12 import at.lumetsnet.caas.model.Menu;
13 import at.lumetsnet.caas.viewmodel.MenuCategoryViewModel;
14 import at.lumetsnet.caas.viewmodel.MenuViewModel;
15
16 /***
17  * Edit/Add dialog for menus
18  * @author romanlum
19  *
20  */
21 public class ManageMenuDialog extends ManageEntityDialog<MenuViewModel> {
22
23   public ManageMenuDialog(Window owner, Menu menu) {
24     viewModel = new MenuViewModel(menu);
25     createGui(owner, "Hauptspeise verwalten",
26         "Bitte geben Sie die Daten der Hauptspeise ein.");
27
28     dialogStage
29         .getScene()
30         .getStylesheets()
31         .add(getClass().getResource("../css/manage-menu-dialog.css")
32             .toExternalForm());
33   }
34
35   @Override
36   protected Node createContentPane() {
37
38     VBox box = new VBox();
39     box.getStyleClass().add("form-container");
40     box.getChildren().addAll(
41         Util.getTextFieldForm("Beschreibung",
42             viewModel.getDescriptionProperty()));
43     box.getChildren().addAll(
44         Util.getComboboxForm("Bereich",
45             viewModel.getCategoryProperty(), getMenuCategories()));
```

```
46
47      box.getChildren().addAll(
48          Util.getTextFieldForm("Preis",
49              viewModel.getPriceAsStringProperty()));
50      DatePicker picker = new DatePicker();
51      box.getChildren().add(Util.getFormLabel("Zeitraum"));
52
53      HBox rangeBox = new HBox();
54      rangeBox.setSpacing(10);
55
56      picker.valueProperty().bindBidirectional(viewModel.getBeginProperty());
57      picker.getStyleClass().add("form-datepicker");
58      rangeBox.getChildren().add(picker);
59
60      picker = new DatePicker();
61      picker.getStyleClass().add("form-datepicker");
62      picker.valueProperty().bindBidirectional(viewModel.getEndProperty());
63      rangeBox.getChildren().add(picker);
64      box.getChildren().add(rangeBox);
65      return box;
66    }
67
68    private ObservableList<MenuCategoryViewModel> getMenuCategories() {
69      ObservableList<MenuCategoryViewModel> data = FXCollections
70          .observableArrayList();
71      MenuService.getInstance().getAllCategories()
72          .forEach(x -> data.add(new MenuCategoryViewModel(x)));
73      return data;
74    }
75
76    @Override
77    protected void saveCommand() {
78      if (validate()) {
79        //Saves the entity
80        MenuService.getInstance().saveOrUpdateMenu(viewModel.toMenuModel());
81        dialogStage.close();
82      }
83
84    }
85
86    /***
87     * Static method for showing the dialog
88     * @param owner
89     * @param model
90     * @return
91     */
92    public static boolean show(Window owner, Menu model) {
93      ManageMenuDialog dialog = new ManageMenuDialog(owner, model);
94      return dialog.show();
```

```
95
96    }
97  }
```

ManageUserDialog.java

```
1  package at.lumetsnet.caas.gui.dialogs;
2
3  import javafx.scene.Node;
4  import javafx.scene.layout.VBox;
5  import javafx.stage.Window;
6  import at.lumetsnet.caas.business.MenuService;
7  import at.lumetsnet.caas.business.UserService;
8  import at.lumetsnet.caas.gui.Util;
9  import at.lumetsnet.caas.model.User;
10 import at.lumetsnet.caas.viewmodel.UserViewModel;
11
12 public class ManageUserDialog extends ManageEntityDialog<UserViewModel> {
13
14   public ManageUserDialog(Window owner, User user) {
15     viewModel = new UserViewModel(user);
16     createGui(owner, "User verwalten",
17         "Bitte geben Sie die Daten des Benutzers ein.");
18
19     dialogStage
20         .getScene()
21         .getStylesheets()
22         .add(getClass().getResource("../css/manage-user-dialog.css")
23             .toExternalForm());
24   }
25
26   @Override
27   protected Node createContentPane() {
28
29     VBox box = new VBox();
30     box.getStyleClass().add("form-container");
31     box.getChildren().addAll(
32         Util.getTextFieldForm("Benutzername",
33             viewModel.getUserNameProperty()));
34     box.getChildren().addAll(
35         Util.getPasswordFieldForm("Passwort",
36             viewModel.getPasswordProperty()));
37     box.getChildren().addAll(
38         Util.getTextFieldForm("Vorname",
39             viewModel.getFirstNameProperty()));
40     box.getChildren().addAll(
41         Util.getTextFieldForm("Nachname",
42             viewModel.getLastNameProperty()));
43
```

```
44      return box;
45    }
46
47    @Override
48    protected void saveCommand() {
49      if (validate()) {
50        //Saves the entity
51        UserService.getInstance().saveOrUpdate(viewModel.toUserModel());
52        dialogStage.close();
53      }
54    }
55
56    /***
57     * Static method for showing the dialog
58     * @param owner
59     * @param model
60     * @return
61     */
62    public static boolean show(Window owner, User model) {
63      ManageUserDialog dialog = new ManageUserDialog(owner, model);
64      return dialog.show();
65
66    }
67  }
```

ManageMenusPage.java

```
1  package at.lumetsnet.caas.gui.pages;
2
3  import java.util.Optional;
4
5  import javafx.scene.Node;
6  import javafx.scene.control.Button;
7  import javafx.scene.control.Label;
8  import javafx.scene.control.TableColumn;
9  import javafx.scene.control.TableView;
10 import javafx.scene.layout.HBox;
11 import javafx.scene.layout.Priority;
12 import javafx.scene.layout.VBox;
13 import at.lumetsnet.caas.gui.ActionTableCell;
14 import at.lumetsnet.caas.gui.AmountTableCell;
15 import at.lumetsnet.caas.gui.dialogs.ManageMenuCategoryDialog;
16 import at.lumetsnet.caas.gui.dialogs.ManageMenuDialog;
17 import at.lumetsnet.caas.viewmodel.ManageMenusPageViewModel;
18 import at.lumetsnet.caas.viewmodel.MenuCategoryViewModel;
19 import at.lumetsnet.caas.viewmodel.MenuViewModel;
20
21 /***
22  * View page used for managing menus
```

```
23  * Uses a ManageMenusPageViewModel for business logic operations
24  * @author romanlum
25  *
26  */
27 public class ManageMenusPage extends VBox implements Showable {
28
29   TableView<MenuCategoryViewModel> categoryTable;
30   TableView<MenuViewModel> menuTable;
31   ManageMenusPageViewModel viewModel;
32
33   public ManageMenusPage() {
34
35     viewModel = new ManageMenusPageViewModel();
36     getStyleClass().add("page-content-container");
37
38     HBox topPane = new HBox();
39     topPane.getChildren().add(createTitle("Bereiche"));
40     HBox commandContainer = new HBox();
41     commandContainer.getStyleClass().add("page-command-container");
42     commandContainer.getChildren().add(createAddCategoryButton());
43     HBox.setHgrow(commandContainer, Priority.ALWAYS);
44
45     topPane.getChildren().add(commandContainer);
46     categoryTable = createCategoryTableView();
47     categoryTable.setItems(viewModel.getCategoryList());
48
49     HBox menuTopPane = new HBox();
50     menuTopPane.getChildren().add(createTitle("Hauptspeisen"));
51     HBox menuCommandContainer = new HBox();
52     menuCommandContainer.getStyleClass().add("page-command-container");
53     menuCommandContainer.getChildren().add(createAddMenuButton());
54     HBox.setHgrow(menuCommandContainer, Priority.ALWAYS);
55
56     menuTopPane.getChildren().add(menuCommandContainer);
57     menuTable = createMenuTableView();
58     menuTable.setItems(viewModel.getMenuList());
59
60     getChildren().addAll(topPane, categoryTable, menuTopPane, menuTable);
61
62   }
63
64   private Node createAddCategoryButton() {
65     Button button = new Button("Bereich anlegen");
66     button.getStyleClass().add("page-command");
67     button.setOnAction(x -> addCategoryCommand());
68     return button;
69   }
70
71   private Node createAddMenuButton() {
```

```
72      Button button = new Button("Hauptspeise anlegen");
73      button.getStyleClass().add("page-command");
74      button.setOnAction(x -> addMenuCommand());
75      return button;
76    }
77
78    private Label createTitle(String title) {
79      Label titleLabel = new Label(title);
80      titleLabel.getStyleClass().add("page-title");
81      return titleLabel;
82    }
83
84    private TableView<MenuCategoryViewModel> createCategoryTableView() {
85      TableView<MenuCategoryViewModel> table = new TableView<>();
86      table.getStyleClass().add("table");
87      table.setColumnResizePolicy(TableView.CONSTRAINED_RESIZE_POLICY);
88
89      TableColumn<MenuCategoryViewModel, String> column = new TableColumn<>(
90          "Name");
91      column.setCellValueFactory(x -> x.getValue().getNameProperty());
92      table.getColumns().add(column);
93
94      TableColumn<MenuCategoryViewModel, Number> actionColumn = new TableColumn<>(
95          "Aktion");
96      actionColumn.setCellValueFactory(x -> x.getValue().getIdProperty());
97      actionColumn.setCellFactory(p -> new ActionTableCell<>(
98          x -> editCategoryCommand(x), x -> deleteCategoryCommand(x)));
99
100     table.getColumns().add(actionColumn);
101     return table;
102   }
103
104   private TableView<MenuViewModel> createMenuTableView() {
105     TableView<MenuViewModel> table = new TableView<>();
106     table.getStyleClass().add("table");
107     table.setColumnResizePolicy(TableView.CONSTRAINED_RESIZE_POLICY);
108
109     TableColumn<MenuViewModel, String> column = new TableColumn<>("Name");
110     column.setCellValueFactory(x -> x.getValue().getDescriptionProperty());
111     table.getColumns().add(column);
112
113     TableColumn<MenuViewModel, Number> priceColumn = new TableColumn<>(
114         "Preis");
115     priceColumn.setCellValueFactory(x -> x.getValue().getPriceProperty());
116     priceColumn.setCellFactory(x -> new AmountTableCell<MenuViewModel>());
117     table.getColumns().add(priceColumn);
118
119     column = new TableColumn<>("Zeitraum");
120     column.setCellValueFactory(x -> x.getValue().getUsageRangeProperty());
```

```
121     table.getColumns().add(column);
122
123     TableColumn<MenuViewModel, Number> actionColumn = new TableColumn<>(
124         "Aktion");
125     actionColumn.setCellValueFactory(x -> x.getValue().getIdProperty());
126     actionColumn.setCellFactory(p -> new ActionTableCell<>(
127         x -> editMenuCommand(x), x -> deleteMenuCommand(x)));
128
129     table.getColumns().add(actionColumn);
130     return table;
131   }
132
133   private void deleteCategoryCommand(Number x) {
134     viewModel.deleteCategoryCommand(x);
135
136   }
137
138   private void editCategoryCommand(Number id) {
139     Optional<MenuCategoryViewModel> model = categoryTable.getItems()
140         .stream().filter(x -> x.getIdProperty().get() == (long) id)
141         .findFirst();
142
143     if (model.isPresent()) {
144       boolean result = ManageMenuCategoryDialog.show(getScene()
145           .getWindow(), model.get().toCategoryModel());
146       if (!result) {
147         viewModel.updateCategories();
148       }
149     }
150   }
151
152   private void deleteMenuCommand(Number x) {
153     viewModel.deleteMenuCommand(x);
154
155   }
156
157   private void editMenuCommand(Number id) {
158     Optional<MenuViewModel> model = menuTable.getItems().stream()
159         .filter(x -> x.getIdProperty().get() == (long) id).findFirst();
160
161     if (model.isPresent()) {
162       boolean result = ManageMenuDialog.show(getScene().getWindow(),
163           model.get().toMenuModel());
164       if (!result) {
165         viewModel.updateMenus();
166       }
167     }
168   }
169
```

```
170   private void addCategoryCommand() {
171     boolean result = ManageMenuCategoryDialog.show(getScene().getWindow(),
172         null);
173     if (!result) {
174       viewModel.updateCategories();
175     }
176   }
177
178   private void addMenuCommand() {
179     boolean result = ManageMenuDialog.show(getScene().getWindow(), null);
180     if (!result) {
181       viewModel.updateMenus();
182     }
183   }
184
185   public void show() {
186     viewModel.updateCategories();
187     viewModel.updateMenus();
188   }
189 }
```

ManageUsersPage.java

```
1  package at.lumetsnet.caas.gui.pages;
2
3  import java.util.Optional;
4
5  import javafx.scene.Node;
6  import javafx.scene.control.Button;
7  import javafx.scene.control.Label;
8  import javafx.scene.control.TableColumn;
9  import javafx.scene.control.TableView;
10 import javafx.scene.layout.HBox;
11 import javafx.scene.layout.Priority;
12 import javafx.scene.layout.VBox;
13 import at.lumetsnet.caas.gui.ManageUserActionCell;
14 import at.lumetsnet.caas.gui.dialogs.ManageUserDialog;
15 import at.lumetsnet.caas.viewmodel.ManageUsersPageViewModel;
16 import at.lumetsnet.caas.viewmodel.UserViewModel;
17
18 /***
19  * View page used for managing users
20  * Uses a ManageUsersPageViewModel for business logic operations
21  * @author romanlum
22  *
23  */
24 public class ManageUsersPage extends VBox implements Showable {
25
26   TableView<UserViewModel> table;
```

```
27  ManageUsersPageViewModel viewModel;
28
29  public ManageUsersPage() {
30
31    viewModel = new ManageUsersPageViewModel();
32    getStyleClass().add("page-content-container");
33
34    HBox topPane = new HBox();
35    topPane.getChildren().add(createTitle("Benutzerliste"));
36    HBox commandContainer = new HBox();
37    commandContainer.getStyleClass().add("page-command-container");
38    commandContainer.getChildren().add(createAddButton());
39    HBox.setHgrow(commandContainer, Priority.ALWAYS);
40
41    topPane.getChildren().add(commandContainer);
42    table = createTableView();
43    table.setItems(viewModel.getUserList());
44    getChildren().addAll(topPane, table);
45
46  }
47
48  private Node createAddButton() {
49    Button button = new Button("User anlegen");
50    button.getStyleClass().add("page-command");
51    button.setOnAction(x -> addUserCommand());
52    return button;
53  }
54
55  private Label createTitle(String title) {
56    Label titleLabel = new Label(title);
57    titleLabel.getStyleClass().add("page-title");
58    return titleLabel;
59  }
60
61  private TableView<UserViewModel> createTableView() {
62    TableView<UserViewModel> table = new TableView<>();
63    table.getStyleClass().add("table");
64    table.setColumnResizePolicy(TableView.CONSTRAINED_RESIZE_POLICY);
65
66    TableColumn<UserViewModel, String> column = new TableColumn<>(
67        "Benutzername");
68    column.setCellValueFactory(x -> x.getValue().getUserNameProperty());
69    table.getColumns().add(column);
70
71    column = new TableColumn<>("Vorname");
72    column.setCellValueFactory(x -> x.getValue().getFirstNameProperty());
73    table.getColumns().add(column);
74
75    column = new TableColumn<>("Nachname");
```

```
76      column.setCellValueFactory(x -> x.getValue().getLastNameProperty());
77      table.getColumns().add(column);
78
79      column = new TableColumn<>("Status");
80      column.setCellValueFactory(x -> x.getValue().getLockedStringProperty());
81      table.getColumns().add(column);
82
83      TableColumn<UserViewModel, Number> actionColumn = new TableColumn<>(
84          "Aktion");
85      actionColumn.setCellValueFactory(x -> x.getValue().getIdProperty());
86      actionColumn.setCellFactory(p -> new ManageUserActionCell<>(
87          x -> editCommand(x), x -> deleteCommand(x),
88          x -> toggleLockStateCommand(x)));
89
90      table.getColumns().add(actionColumn);
91      return table;
92    }
93
94    private void toggleLockStateCommand(Number x) {
95      viewModel.toggleLockStateCommand(x);
96    }
97
98    private void deleteCommand(Number x) {
99      viewModel.deleteCommand(x);
100
101   }
102
103   private void editCommand(Number id) {
104     Optional<UserViewModel> model = table.getItems().stream()
105         .filter(x -> x.getIdProperty().get() == (long) id).findFirst();
106
107     if (model.isPresent()) {
108       //call edit dialog
109       boolean result = ManageUserDialog.show(getScene().getWindow(),
110           model.get().toUserModel());
111       if (!result) {
112         viewModel.update();
113       }
114     }
115   }
116
117   private void addUserCommand() {
118     //call add dialog
119     boolean result = ManageUserDialog.show(getScene().getWindow(), null);
120     if (!result) {
121       viewModel.update();
122     }
123   }
124
```

```
125   public void show() {
126     viewModel.update();
127   }
128 }
```

OrdersPage.java

```
1  package at.lumetsnet.caas.gui.pages;
2
3  import javafx.scene.control.Label;
4  import javafx.scene.control.TableColumn;
5  import javafx.scene.control.TableColumn.SortType;
6  import javafx.scene.control.TableView;
7  import javafx.scene.layout.VBox;
8  import at.lumetsnet.caas.viewmodel.OrderViewModel;
9  import at.lumetsnet.caas.viewmodel.OrdersPageViewModel;
10
11 /***
12  * View page used for showing the orders
13  * Uses a OrdersPageViewModel for business logic operations
14  * @author romanlum
15  *
16  */
17 public class OrdersPage extends VBox implements Showable {
18
19   TableView<OrderViewModel> table;
20   OrdersPageViewModel viewModel;
21
22   public OrdersPage() {
23
24     viewModel = new OrdersPageViewModel();
25     getStyleClass().add("page-content-container");
26
27     Label title = createTitle("Heutige Bestellungen");
28     getChildren().add(title);
29
30     table = createTableView();
31     table.setItems(viewModel.getOrders());
32     getChildren().add(table);
33
34   }
35
36   private Label createTitle(String title) {
37     Label titleLabel = new Label(title);
38     titleLabel.getStyleClass().add("page-title");
39     return titleLabel;
40   }
41
42   private TableView<OrderViewModel> createTableView() {
```

```
43      TableView<OrderViewModel> table = new TableView<>();
44      table.getStyleClass().add("table");
45      table.setColumnResizePolicy(TableView.CONSTRAINED_RESIZE_POLICY);
46      TableColumn<OrderViewModel, String> column = new TableColumn<>(
47          "Benutzer");
48      column.setCellValueFactory(x -> x.getValue().getUserNameProperty());
49      table.getColumns().add(column);
50
51      column = new TableColumn<>("Gericht");
52      column.setCellValueFactory(x -> x.getValue().getMenuProperty());
53      table.getColumns().add(column);
54
55      column = new TableColumn<>("Zeit");
56      column.setCellValueFactory(x -> x.getValue().getTimeProperty());
57      column.setSortType(SortType.ASCENDING);
58      table.getColumns().add(column);
59      table.getSortOrder().clear();
60      table.getSortOrder().add(column);
61
62      column = new TableColumn<>("Kommentar");
63      column.setCellValueFactory(x -> x.getValue().getCommentProperty());
64
65      table.getColumns().add(column);
66
67      return table;
68    }
69
70    public void show() {
71      viewModel.update();
72      table.sort();
73    }
74  }
```

Showable.java

```
1  package at.lumetsnet.caas.gui.pages;
2
3  /***
4   * Simple interface used for marking pages
5   * as showable
6   * @author romanlum
7   *
8   */
9  public interface Showable {
10
11   public void show();
12 }
```

Entity.java

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

/***
 * Base data entity
 * @author romanlum
 *
 */
public class Entity {
  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;
  }
}
```

Menu.java

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

import java.time.LocalDate;

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

public class Menu extends Entity {

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

  public Menu() {
```

```
21  }
22
23  /**
24   * @param id
25   * @param description
26   * @param price
27   * @param begin
28   * @param end
29   * @param category
30   */
31  public Menu(long id, String description, long price, LocalDate begin,
32      LocalDate end, MenuCategory category) {
33    this.id = id;
34    this.description = description;
35    this.price = price;
36    this.begin = begin;
37    this.end = end;
38    this.category = category;
39  }
40
41  /**
42   * @return the description
43   */
44  public String getDescription() {
45    return description;
46  }
47
48  /**
49   * @param description
50   *            the description to set
51   */
52  public void setDescription(String description) {
53    this.description = description;
54  }
55
56  /**
57   * @return the price in the smallest currency unit
58   *
59   */
60  public long getPrice() {
61    return price;
62  }
63
64  /**
65   * @param price
66   *            the price to set
67   *            this price is set in the smallest
68   *            currency unit
69   */
```

```
70    public void setPrice(long price) {
71      this.price = price;
72    }
73
74    /**
75     * @return the begin
76     */
77    public LocalDate getBegin() {
78      return begin;
79    }
80
81    /**
82     * @param begin
83     *            the begin to set
84     */
85    public void setBegin(LocalDate begin) {
86      this.begin = begin;
87    }
88
89    /**
90     * @return the end
91     */
92    public LocalDate getEnd() {
93      return end;
94    }
95
96    /**
97     * @param end
98     *            the end to set
99     */
100   public void setEnd(LocalDate end) {
101     this.end = end;
102   }
103
104   /**
105    * @return the category
106    */
107   public MenuCategory getCategory() {
108     return category;
109   }
110
111   /**
112    * @param category
113    *            the category to set
114    */
115   public void setCategory(MenuCategory category) {
116     this.category = category;
117   }
118
```

```
119 }
```

MenuCategory.java

```
1  package at.lumetsnet.caas.model;
2
3  /***
4   * MenuCategory data entity
5   * @author romanlum
6   *
7   */
8
9  public class MenuCategory extends Entity {
10
11   private String name;
12
13   public MenuCategory() {
14   }
15
16   /**
17    * @param id
18    * @param name
19    */
20   public MenuCategory(long id, String name) {
21     this.id = id;
22     this.name = name;
23   }
24
25   /**
26    * @return the name
27    */
28   public String getName() {
29     return name;
30   }
31
32   /**
33    * @param name
34    *             the name to set
35    */
36   public void setName(String name) {
37     this.name = name;
38   }
39
40 }
```

Order.java

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

import java.time.LocalDateTime;

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

public class Order extends Entity {

  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
   */
  public void setMenu(Menu menu) {
    this.menu = menu;
```

```
50    }
51
52    /**
53     * @return the user
54     */
55    public User getUser() {
56      return user;
57    }
58
59    /**
60     * @param user
61     *            the user to set
62     */
63    public void setUser(User user) {
64      this.user = user;
65    }
66
67    /**
68     * @return the comment
69     */
70    public String getComment() {
71      return comment;
72    }
73
74    /**
75     * @param comment
76     *            the comment to set
77     */
78    public void setComment(String comment) {
79      this.comment = comment;
80    }
81
82    /**
83     * @return the time
84     */
85    public LocalDateTime getTime() {
86      return time;
87    }
88
89    /**
90     * @param time
91     *            the time to set
92     */
93    public void setTime(LocalDateTime time) {
94      this.time = time;
95    }
96
97  }
```

User.java

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

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

public class User extends Entity {

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

  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, boolean isAdmin) {
    this.id = id;
    this.userName = userName;
    this.password = password;
    this.firstName = firstName;
    this.lastName = lastName;
    this.locked = locked;
    this.isAdmin = isAdmin;
  }

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

```

```
48    /**
49     * @param userName
50     *            the userName to set
51     */
52    public void setUserName(String userName) {
53      this.userName = userName;
54    }
55
56    /**
57     * @return the password
58     */
59    public String getPassword() {
60      return password;
61    }
62
63    /**
64     * @param password
65     *            the password to set
66     */
67    public void setPassword(String password) {
68      this.password = password;
69    }
70
71    /**
72     * @return the firstName
73     */
74    public String getFirstName() {
75      return firstName;
76    }
77
78    /**
79     * @param firstName
80     *            the firstName to set
81     */
82    public void setFirstName(String firstName) {
83      this.firstName = firstName;
84    }
85
86    /**
87     * @return the lastName
88     */
89    public String getLastName() {
90      return lastName;
91    }
92
93    /**
94     * @param lastName
95     *            the lastName to set
96     */
```

```
97    public void setLastName(String lastName) {
98      this.lastName = lastName;
99    }
100
101   /**
102    * @return the locked
103    */
104   public boolean isLocked() {
105     return locked;
106   }
107
108   /**
109    * @param locked
110    *             the locked to set
111    */
112   public void setLocked(boolean locked) {
113     this.locked = locked;
114   }
115
116   /**
117    * @return the isAdmin
118    */
119   public boolean isAdmin() {
120     return isAdmin;
121   }
122
123   /**
124    * @param isAdmin
125    *             the isAdmin to set
126    */
127   public void setAdmin(boolean isAdmin) {
128     this.isAdmin = isAdmin;
129   }
130
131 }
```

Validatable.java

```
1  package at.lumetsnet.caas.viewmodel;
2
3  /***
4   * Interface used for validating entity view models
5   * @author romanlum
6   *
7   */
8  public interface Validatable {
9
10   /***
11    * validates the entity
```

```
12     * @return true on successfull validation otherwise false
13     */
14   boolean validate();
15 }
```

ManageMenusPageViewModel.java

```
1  package at.lumetsnet.caas.viewmodel;
2
3  import java.util.Collection;
4
5  import javafx.collections.FXCollections;
6  import javafx.collections.ObservableList;
7  import at.lumetsnet.caas.business.MenuService;
8  import at.lumetsnet.caas.model.Menu;
9  import at.lumetsnet.caas.model.MenuCategory;
10
11 /***
12  * ViewModel (logic) class for ManageMenusPage
13  * @author romanlum
14  *
15  */
16 public class ManageMenusPageViewModel {
17
18   private ObservableList<MenuCategoryViewModel> categories;
19   private ObservableList<MenuViewModel> menus;
20
21   public ManageMenusPageViewModel() {
22     categories = FXCollections.observableArrayList();
23     menus = FXCollections.observableArrayList();
24   }
25
26   public void updateCategories() {
27     Collection<MenuCategory> data = MenuService.getInstance()
28         .getAllCategories();
29     categories.clear();
30     if (data != null) {
31       data.forEach(x -> categories.add(new MenuCategoryViewModel(x)));
32     }
33   }
34
35   public void updateMenus() {
36     Collection<Menu> data = MenuService.getInstance().getAllMenus();
37     menus.clear();
38     if (data != null) {
39       data.forEach(x -> menus.add(new MenuViewModel(x)));
40     }
41   }
42
```

```
43   public ObservableList<MenuCategoryViewModel> getCategoryList() {
44     return categories;
45   }
46 
47   public ObservableList<MenuViewModel> getMenuList() {
48     return menus;
49   }
50 
51   public void deleteCategoryCommand(Number userId) {
52     MenuService.getInstance().deleteCategory((long) userId);
53     updateCategories();
54 
55   }
56 
57   public void deleteMenuCommand(Number id) {
58     MenuService.getInstance().deleteMenu((long) id);
59     updateMenus();
60 
61   }
62 
63 }
```

ManageUsersPageViewModel.java

```
1  package at.lumetsnet.caas.viewmodel;
2 
3  import java.util.Collection;
4 
5  import javafx.collections.FXCollections;
6  import javafx.collections.ObservableList;
7  import at.lumetsnet.caas.business.UserService;
8  import at.lumetsnet.caas.model.User;
9 
10 /***
11  * ViewModel (logic) class for ManageUsersPage
12  * @author romanlum
13  *
14  */
15 
16 public class ManageUsersPageViewModel {
17 
18   private ObservableList<UserViewModel> users;
19 
20   public ManageUsersPageViewModel() {
21     users = FXCollections.observableArrayList();
22   }
23 
24   public void update() {
25     Collection<User> data = UserService.getInstance().getAllUsers();
```

```
26      users.clear();
27      if (data != null) {
28        data.forEach(x -> users.add(new UserViewModel(x)));
29      }
30    }
31
32    public ObservableList<UserViewModel> getUserList() {
33      return users;
34    }
35
36    public void deleteCommand(Number userId) {
37      UserService.getInstance().deleteUser((long) userId);
38      update();
39
40    }
41
42    public void toggleLockStateCommand(Number userId) {
43      UserService.getInstance().toggleLockState((long) userId);
44      update();
45    }
46  }
```

MenuCategoryViewModel.java

```
1  package at.lumetsnet.caas.viewmodel;
2
3  import javafx.beans.property.LongProperty;
4  import javafx.beans.property.SimpleLongProperty;
5  import javafx.beans.property.SimpleStringProperty;
6  import javafx.beans.property.StringProperty;
7  import at.lumetsnet.caas.model.MenuCategory;
8
9  /***
10  * ViewModel wrapper for MenuCategory entity
11  * Uses javafx properties for databinding
12  * @author romanlum
13  *
14  */
15 public class MenuCategoryViewModel implements Validatable {
16
17    private LongProperty idProperty;
18    private StringProperty nameProperty;
19
20    public MenuCategoryViewModel(MenuCategory input) {
21      if (input == null) {
22        input = new MenuCategory();
23        input.setId(-1);
24      }
25      idProperty = new SimpleLongProperty(input.getId());
```

```
26      nameProperty = new SimpleStringProperty(input.getName());
27
28    }
29
30    public MenuCategory toCategoryModel() {
31      MenuCategory result = new MenuCategory();
32      result.setId(idProperty.get());
33      result.setName(nameProperty.get());
34      return result;
35
36    }
37
38    public boolean validate() {
39      return nameProperty.get() != null && !nameProperty.get().isEmpty();
40    }
41
42    @Override
43    public String toString() {
44      return getNameProperty().get();
45    }
46
47    /**
48     * @return the id
49     */
50    public LongProperty getIdProperty() {
51      return idProperty;
52    }
53
54    /**
55     * @return the userNameProperty
56     */
57    public StringProperty getNameProperty() {
58      return nameProperty;
59    }
60
61  }
```

MenuViewModel.java

```
1  package at.lumetsnet.caas.viewmodel;
2
3  import java.time.LocalDate;
4  import java.time.format.DateTimeFormatter;
5
6  import javafx.beans.property.LongProperty;
7  import javafx.beans.property.ObjectProperty;
8  import javafx.beans.property.SimpleLongProperty;
9  import javafx.beans.property.SimpleObjectProperty;
10 import javafx.beans.property.SimpleStringProperty;
```

```
11 import javafx.beans.property.StringProperty;
12 import at.lumetsnet.caas.model.Menu;
13
14 /***
15  * ViewModel wrapper for Menu entity
16  * Uses javafx properties for databinding
17  * @author romanlum
18  *
19  */
20
21 public class MenuViewModel implements Validatable {
22
23   private LongProperty idProperty;
24   private StringProperty descriptionProperty;
25   private LongProperty priceProperty;
26   private ObjectProperty<LocalDate> beginProperty;
27   private ObjectProperty<LocalDate> endProperty;
28   private ObjectProperty<MenuCategoryViewModel> categoryProperty;
29   private StringProperty usageRangeProperty;
30   private StringProperty priceAsStringProperty;
31
32   public MenuViewModel(Menu data) {
33     if (data == null) {
34       data = new Menu();
35       data.setId(-1);
36     }
37     idProperty = new SimpleLongProperty(data.getId());
38     descriptionProperty = new SimpleStringProperty(data.getDescription());
39     priceProperty = new SimpleLongProperty(data.getPrice());
40     beginProperty = new SimpleObjectProperty<LocalDate>(data.getBegin());
41     endProperty = new SimpleObjectProperty<LocalDate>(data.getEnd());
42     categoryProperty = new SimpleObjectProperty<>(
43         new MenuCategoryViewModel(data.getCategory()));
44     if (data.getBegin() != null && data.getEnd() != null) {
45       usageRangeProperty = new SimpleStringProperty(data.getBegin()
46           .format(DateTimeFormatter.ISO_LOCAL_DATE)
47           + " - "
48           + data.getEnd().format(DateTimeFormatter.ISO_LOCAL_DATE));
49     } else {
50       usageRangeProperty = new SimpleStringProperty("");
51     }
52     priceAsStringProperty = new SimpleStringProperty(""
53         + (data.getPrice() / (double) 100));
54   }
55
56   public Menu toMenuModel() {
57     Menu menu = new Menu();
58     menu.setId(idProperty.get());
59     menu.setBegin(beginProperty.get());
```

```
60      menu.setEnd(endProperty.get());
61      menu.setDescription(descriptionProperty.get());
62
63      long amount = (long) (Double.parseDouble(priceAsStringProperty.get()) * 100);
64      menu.setPrice(amount);
65      menu.setCategory(categoryProperty.get().toCategoryModel());
66      return menu;
67
68    }
69
70    /**
71     * @return the idProperty
72     */
73    public LongProperty getIdProperty() {
74      return idProperty;
75    }
76
77    /**
78     * @return the descriptionNameProperty
79     */
80    public StringProperty getDescriptionProperty() {
81      return descriptionProperty;
82    }
83
84    /**
85     * @return the priceProperty
86     */
87    public LongProperty getPriceProperty() {
88      return priceProperty;
89    }
90
91    /**
92     * @return the beginProperty
93     */
94    public ObjectProperty<LocalDate> getBeginProperty() {
95      return beginProperty;
96    }
97
98    /**
99     * @return the endProperty
100    */
101   public ObjectProperty<LocalDate> getEndProperty() {
102     return endProperty;
103   }
104
105   /**
106    * @return the categoryProperty
107    */
108   public ObjectProperty<MenuCategoryViewModel> getCategoryProperty() {
```

```
109     return categoryProperty;
110   }
111
112   /**
113    * @return the usageRange
114    */
115   public StringProperty getUsageRangeProperty() {
116     return usageRangeProperty;
117   }
118
119   /**
120    * @return the priceAsStringProperty
121    */
122   public StringProperty getPriceAsStringProperty() {
123     return priceAsStringProperty;
124   }
125
126   @Override
127   public boolean validate() {
128     boolean val = descriptionProperty.get() != null
129         && !descriptionProperty.get().isEmpty()
130         && categoryProperty.get() != null
131         && categoryProperty.get().getIdProperty().get() != -1;
132
133     if (!val)
134       return val;
135     // extended validation of price
136     try {
137       Double.parseDouble(priceAsStringProperty.get());
138
139     } catch (NumberFormatException ex) {
140       return false;
141     }
142     return true;
143   }
144
145 }
```

OrdersPageViewModel.java

```
1 package at.lumetsnet.caas.viewmodel;
2
3 import java.util.Collection;
4
5 import javafx.collections.FXCollections;
6 import javafx.collections.ObservableList;
7 import at.lumetsnet.caas.business.OrderService;
8 import at.lumetsnet.caas.model.Order;
9
```

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

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;
```

```
21
22    public OrderViewModel(Order order) {
23      userNameProperty = new SimpleStringProperty(order.getUser()
24          .getFirstName() + " " + order.getUser().getLastName());
25      menuProperty = new SimpleStringProperty(order.getMenu()
26          .getDescription());
27      timeProperty = new SimpleStringProperty(order.getTime().format(
28          DateTimeFormatter.ofPattern("HH:mm")));
29      commentProperty = new SimpleStringProperty(order.getComment());
30    }
31
32    /**
33     * @return the userNameProperty
34     */
35    public StringProperty getUserNameProperty() {
36      return userNameProperty;
37    }
38
39    /**
40     * @return the menuProperty
41     */
42    public StringProperty getMenuProperty() {
43      return menuProperty;
44    }
45
46    /**
47     * @return the timeProperty
48     */
49    public StringProperty getTimeProperty() {
50      return timeProperty;
51    }
52
53    /**
54     * @return the commentProperty
55     */
56    public StringProperty getCommentProperty() {
57      return commentProperty;
58    }
59
60  }
```

UserViewModel.java

```
1  package at.lumetsnet.caas.viewmodel;
2
3  import javafx.beans.property.BooleanProperty;
4  import javafx.beans.property.LongProperty;
5  import javafx.beans.property.SimpleBooleanProperty;
6  import javafx.beans.property.SimpleLongProperty;
```

```
7  import javafx.beans.property.SimpleStringProperty;
8  import javafx.beans.property.StringProperty;
9  import at.lumetsnet.caas.model.User;
10
11 /***
12  * ViewModel wrapper for User entity
13  * Uses javafx properties for databinding
14  * @author romanlum
15  *
16  */
17
18 public class UserViewModel implements Validatable {
19
20   private LongProperty idProperty;
21   private StringProperty userNameProperty;
22   private StringProperty passwordProperty;
23   private StringProperty firstNameProperty;
24   private StringProperty lastNameProperty;
25   private BooleanProperty lockedProperty;
26   private BooleanProperty isAdminProperty;
27
28   public UserViewModel(User user) {
29     if (user == null) {
30       user = new User();
31       user.setId(-1);
32     }
33     idProperty = new SimpleLongProperty(user.getId());
34     userNameProperty = new SimpleStringProperty(user.getUserName());
35     passwordProperty = new SimpleStringProperty(user.getPassword());
36     firstNameProperty = new SimpleStringProperty(user.getFirstName());
37     lastNameProperty = new SimpleStringProperty(user.getLastName());
38     lockedProperty = new SimpleBooleanProperty(user.isLocked());
39     isAdminProperty = new SimpleBooleanProperty(user.isAdmin());
40
41   }
42
43   public User toUserModel() {
44     User result = new User();
45     result.setId(idProperty.get());
46     result.setUserName(userNameProperty.get());
47     result.setPassword(passwordProperty.get());
48     result.setFirstName(firstNameProperty.get());
49     result.setLastName(lastNameProperty.get());
50     result.setLocked(lockedProperty.get());
51     result.setAdmin(isAdminProperty.get());
52     return result;
53
54   }
55
```

```
56  /**
57   * @return the id
58   */
59  public LongProperty getIdProperty() {
60    return idProperty;
61  }
62
63  /**
64   * @return the userNameProperty
65   */
66  public StringProperty getUserNameProperty() {
67    return userNameProperty;
68  }
69
70  /**
71   * @return the passwordProperty
72   */
73  public StringProperty getPasswordProperty() {
74    return passwordProperty;
75  }
76
77  /**
78   * @return the firstNameProperty
79   */
80  public StringProperty getFirstNameProperty() {
81    return firstNameProperty;
82  }
83
84  /**
85   * @return the lastNameProperty
86   */
87  public StringProperty getLastNameProperty() {
88    return lastNameProperty;
89  }
90
91  /**
92   * @return the lockedProperty
93   */
94  public BooleanProperty getLockedProperty() {
95    return lockedProperty;
96  }
97
98  /**
99   * @return the lockedProperty as string
100  */
101 public StringProperty getLockedStringProperty() {
102   return new SimpleStringProperty(lockedProperty.get() ? "Gesperrt"
103       : "Ok");
104 }
```

```
105
106   /**
107    * @return the isAdminProperty
108    */
109   public BooleanProperty getIsAdminProperty() {
110     return isAdminProperty;
111   }
112
113   @Override
114   public boolean validate() {
115     return userNameProperty.get() != null
116         && !userNameProperty.get().isEmpty()
117         && passwordProperty.get() != null
118         && !passwordProperty.get().isEmpty()
119         && firstNameProperty.get() != null
120         && !firstNameProperty.get().isEmpty()
121         && lastNameProperty.get() != null
122         && !lastNameProperty.get().isEmpty();
123
124   }
125
126 }
```

1.3 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;
}
```

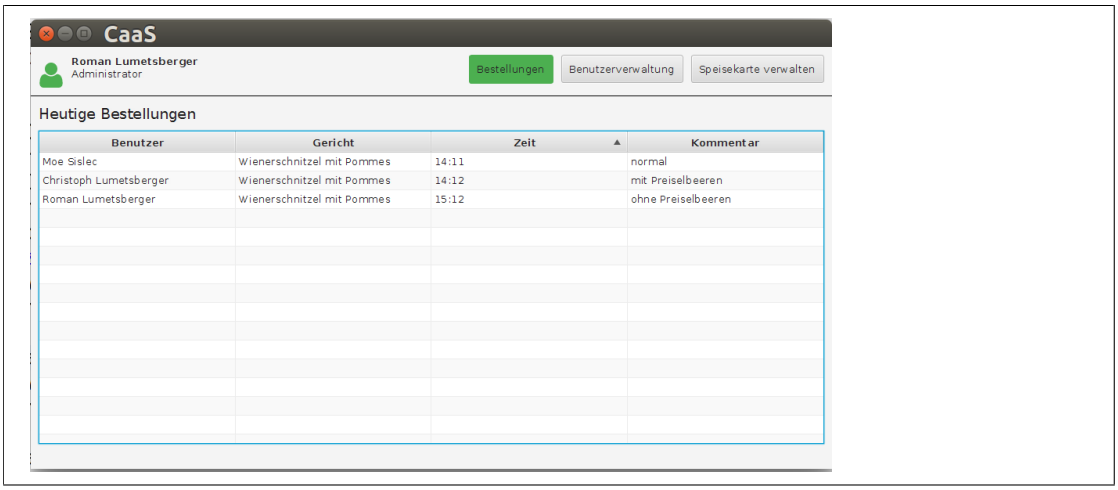
manage-menu-dialog.css

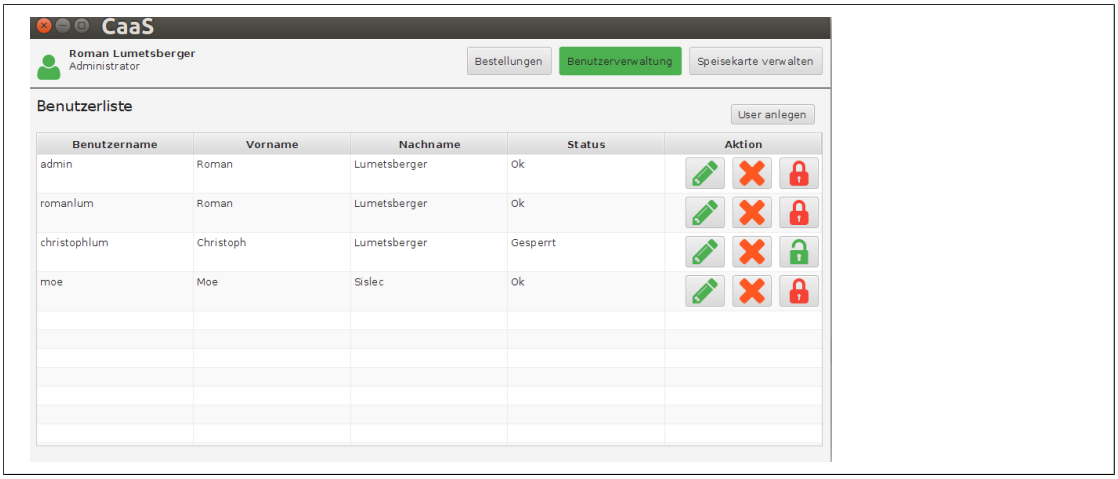
```
/* dialog icon */
#dialog-icon {
  -fx-padding: 10 0 0 0;
  -fx-graphic: url('ic-edit.png');
  -fx-alignment: baseline-center;
  -fx-content-display: center;
}
```

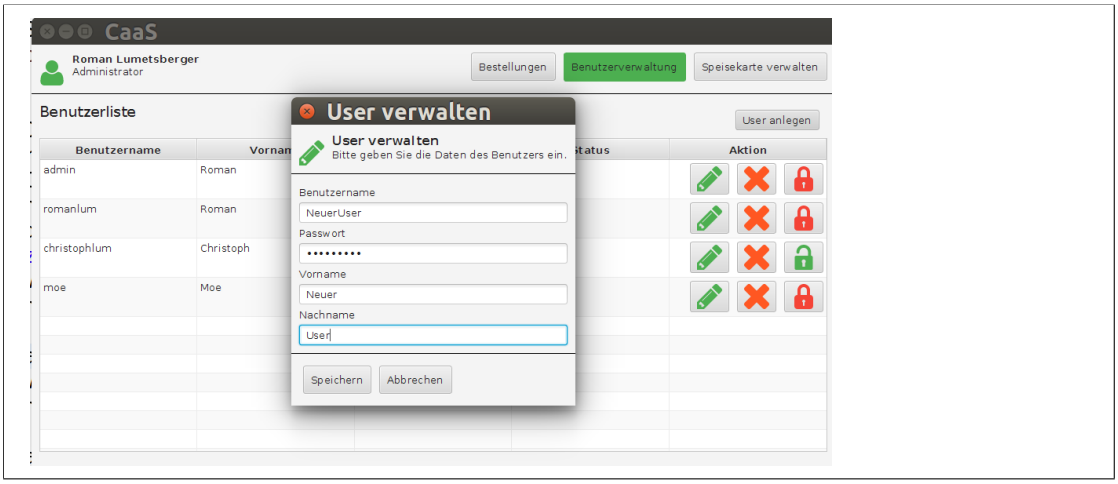
manage-user-dialog.css

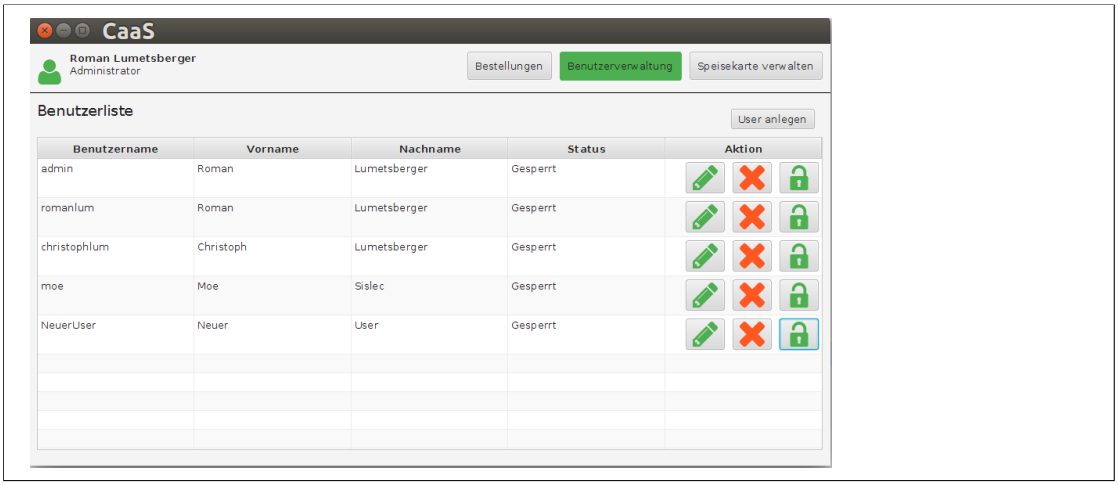
```
/* dialog icon */
#dialog-icon {
  -fx-padding: 10 0 0 0;
  -fx-graphic: url('ic-edit.png');
  -fx-alignment: baseline-center;
  -fx-content-display: center;
}
```

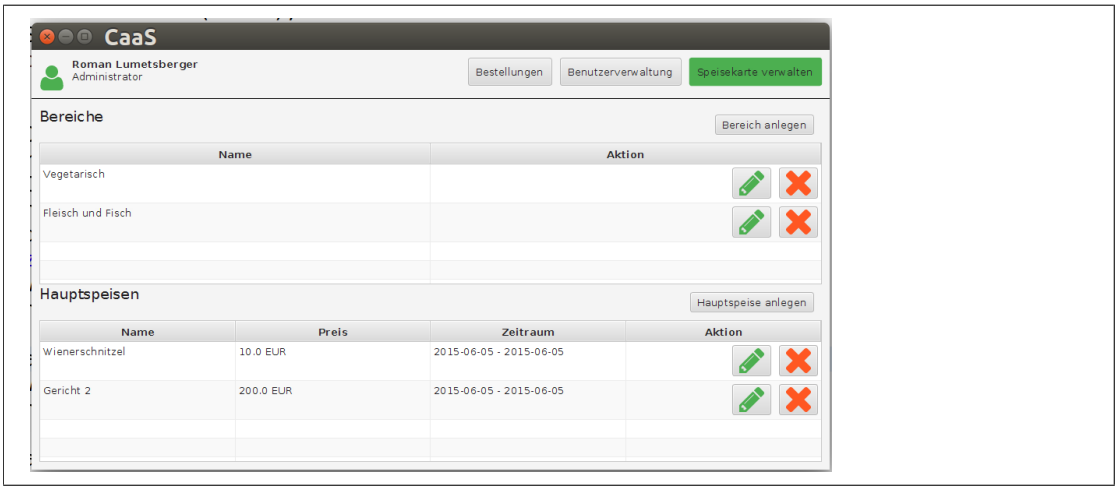
1.4 Test - Screenshots

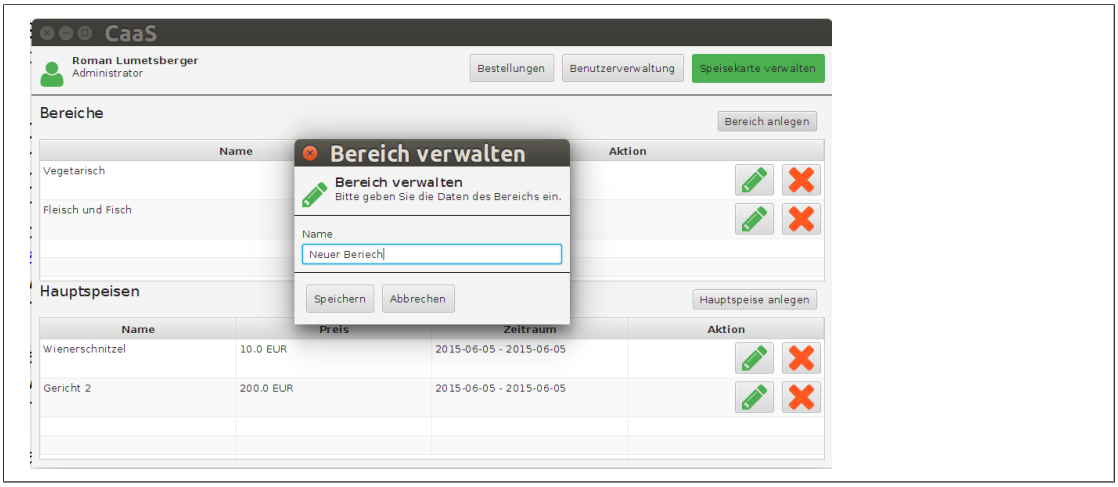


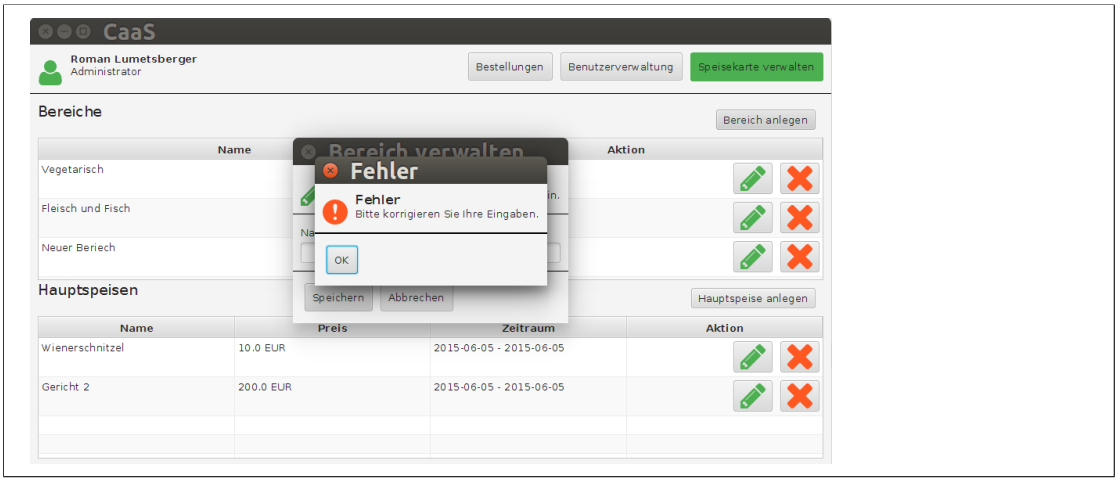


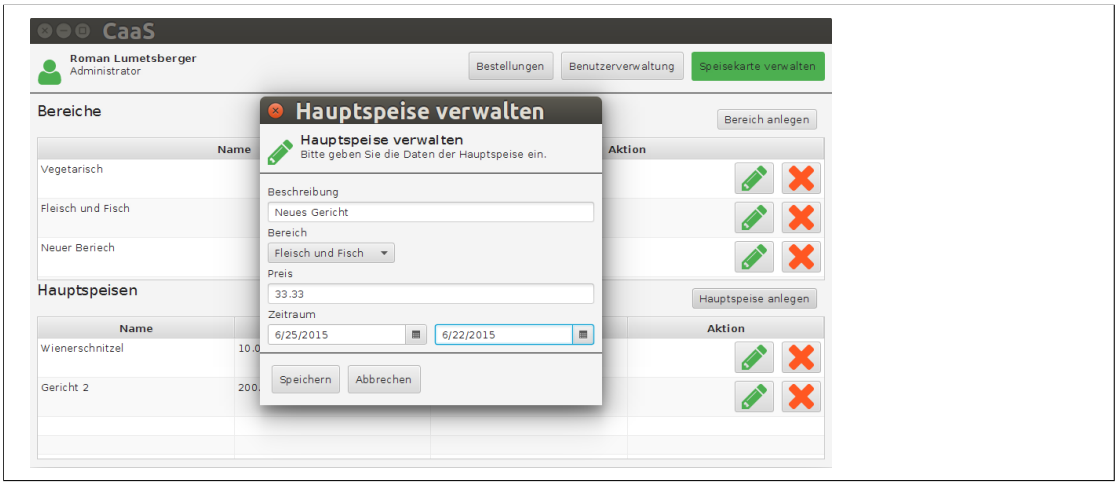



CaaS
Roman Lumetsberger
Administrator
Bestellungen
Benutzerverwaltung
Speisekarte verwalten
Benutzerliste
User anlegen
Benutzername
Vorname
Nachname
Status
Aktion
admin
Roman
Lumetsberger
Gesperrt
romanlum
Roman
Lumetsberger
Gesperrt
christophlum
Christoph
Lumetsberger
Gesperrt
moe
Moe
Sislec
Gesperrt
NeuerUser
Neuer
User
Gesperrt


CaaS
Roman Lumetsberger
Administrator
Bestellungen
Benutzerverwaltung
Speisekarte verwalten
Bereiche
Bereich anlegen
Name
Aktion
Vegetarisch
Fleisch und Fisch
Hauptspeisen
Hauptspeise anlegen
Name
Preis
Zeitraum
Aktion
Wienerschnitzel
10.0 EUR
2015-06-05 - 2015-06-05
Gericht 2
200.0 EUR
2015-06-05 - 2015-06-05


CaaS
Roman Lumetsberger
Administrator
Bestellungen
Benutzerverwaltung
Speisekarte verwalten
Bereiche
Bereich anlegen
Name
Aktion
Vegetarisch
Fleisch und Fisch
Bereich verwalten
Bereich verwalten
Bitte geben Sie die Daten des Bereichs ein.
Name
Neuer Beriech
Speichern
Abbrechen
Hauptspeisen
Hauptspeise anlegen
Name
Preis
Zeitraum
Aktion
Wienerschnitzel
10.0 EUR
2015-06-05 - 2015-06-05
Gericht 2
200.0 EUR
2015-06-05 - 2015-06-05


CaaS
Roman Lumetsberger
Administrator
Bestellungen
Benutzerverwaltung
Speisekarte verwalten
Bereiche
Bereich anlegen
Name
Aktion
Vegetarisch
Fleisch und Fisch
Neuer Beriech
Fehler
Fehler
Bitte korrigieren Sie Ihre Eingaben.
OK
Speichern
Abbrechen
Hauptspeisen
Hauptspeise anlegen
Name
Preis
Zeitraum
Aktion
Wienerschnitzel
10.0 EUR
2015-06-05 - 2015-06-05
Gericht 2
200.0 EUR
2015-06-05 - 2015-06-05


CaaS
Roman Lumetsberger
Administrator
Bestellungen
Benutzerverwaltung
Speisekarte verwalten
Bereiche
Bereich anlegen
Hauptspeise verwalten
Hauptspeise verwalten
Bitte geben Sie die Daten der Hauptspeise ein.
Beschreibung
Neues Gericht
Bereich
Fleisch und Fisch
Preis
33.33
Zeitraum
6/25/2015
6/22/2015
Speichern
Abbrechen
Name
Aktion
Vegetarisch
Fleisch und Fisch
Neuer Beriech
Hauptspeisen
Hauptspeise anlegen
Wienerschnitzel
Gericht 2