**Projekt: Mosti**

Software Architektur Spezifikation

(Software Architecture Document)

[Dokumentstruktur basiert auf RUP „Software Architecture Document“]

# Dokumentinformationen

## Änderungsgeschichte

|  |  |  |  |
| --- | --- | --- | --- |
| *Datum* | *Version* | *Änderung* | *Autor* |
| 5.5.2016 | 1.0 | Erstellung der Software Architektur Spezifikation | Team 11 |
| 29.5.2016 | 2.0 | Aktualisierte Packagebeschreibung | Team 11 |
| 6.7.2016 | 3.0 | Endabgabe und Aktualisierte Packagebeschreibungen | Team 11 |
|  |  |  |  |

## Inhalt

[1 Dokumentinformationen 2](#_Toc455739329)

[1.1 Änderungsgeschichte 2](#_Toc455739330)

[1.2 Inhalt 3](#_Toc455739331)

[2 Einführung (Introduction) 6](#_Toc455739332)

[2.1 Definitionen und Abkürzungen (Definitions, Acronyms, Abbreviations) 6](#_Toc455739333)

[2.2 Referenzen (References) 6](#_Toc455739334)

[2.3 Übersicht (Overview) 6](#_Toc455739335)

[3 Architektonische Darstellung (Architectural Representation) 7](#_Toc455739336)

[4 Architektonische Ziele & Einschränkungen (Architectural Goals and Constraints) 7](#_Toc455739337)

[5 Logische Architektur (Logical View) 8](#_Toc455739338)

[5.1 Übersicht (Overview) 8](#_Toc455739339)

[5.2 Design Pakete (Architecturally Significant Design Packages) 10](#_Toc455739340)

[5.2.1 Package GUI 10](#_Toc455739341)

[5.2.1.1 Beschreibung des Package 10](#_Toc455739342)

[5.2.1.2 Schnittstellen 10](#_Toc455739343)

[5.2.2 Package Domain-Logik 10](#_Toc455739344)

[5.2.2.1 Beschreibung des Package 10](#_Toc455739345)

[5.2.2.2 Schnittstellen 10](#_Toc455739346)

[5.2.2.3 Operationen 10](#_Toc455739347)

[5.2.3 Package Persistenz 10](#_Toc455739348)

[5.2.3.1 Beschreibung des Package 10](#_Toc455739349)

[5.2.3.2 Schnittstellen 10](#_Toc455739350)

[5.3 Java-Pakete und deren Architektur-Design 11](#_Toc455739351)

[5.3.1 Package gui.account 11](#_Toc455739352)

[5.3.1.1 Beschreibung des Packages 11](#_Toc455739353)

[5.3.1.2 Diagramme 11](#_Toc455739354)

[5.3.1.3 Schnittstellen 11](#_Toc455739355)

[5.3.2 Package logik.account 12](#_Toc455739356)

[5.3.2.1 Beschreibung des Packages 12](#_Toc455739357)

[5.3.2.2 Diagramme 12](#_Toc455739358)

[5.3.2.3 Schnittstellen 12](#_Toc455739359)

[5.3.3 Package gui.administratorverwaltung 13](#_Toc455739360)

[5.3.3.1 Beschreibung des Packages 13](#_Toc455739361)

[5.3.3.2 Diagramme 13](#_Toc455739362)

[5.3.3.3 Schnittstellen 13](#_Toc455739363)

[5.3.4 Package logik.administratorverwaltung 14](#_Toc455739364)

[5.3.4.1 Beschreibung des Packages 14](#_Toc455739365)

[5.3.4.2 Diagramme 14](#_Toc455739366)

[5.3.4.3 Schnittstellen 14](#_Toc455739367)

[5.3.5 Package gui.dienstleistungsverwaltung 15](#_Toc455739368)

[5.3.5.1 Beschreibung des Packages 15](#_Toc455739369)

[5.3.5.2 Diagramme 15](#_Toc455739370)

[5.3.5.3 Schnittstellen 15](#_Toc455739371)

[5.3.6 Package logik.dienstleistungenverwaltung 16](#_Toc455739372)

[5.3.6.1 Beschreibung des Packages 16](#_Toc455739373)

[5.3.6.2 Diagramme 16](#_Toc455739374)

[5.3.6.3 Schnittstellen 16](#_Toc455739375)

[5.3.7 Package gui.kassenfunktion 17](#_Toc455739376)

[5.3.7.1 Beschreibung des Packages 17](#_Toc455739377)

[5.3.7.2 Diagramme 17](#_Toc455739378)

[5.3.7.3 Schnittstellen 17](#_Toc455739379)

[5.3.8 Package gui.kundenverwaltung 18](#_Toc455739380)

[5.3.8.1 Beschreibung des Packages 18](#_Toc455739381)

[5.3.8.2 Diagramme 18](#_Toc455739382)

[5.3.8.3 Schnittstellen 18](#_Toc455739383)

[5.3.9 Package logik.kundenverwaltung 19](#_Toc455739384)

[5.3.9.1 Beschreibung des Packages 19](#_Toc455739385)

[5.3.9.2 Diagramme 19](#_Toc455739386)

[5.3.9.3 Schnittstellen 19](#_Toc455739387)

[5.3.10 Package gui.mitarbeiterverwaltung 20](#_Toc455739388)

[5.3.10.1 Beschreibung des Packages 20](#_Toc455739389)

[5.3.10.2 Diagramme 20](#_Toc455739390)

[5.3.10.3 Schnittstellen 20](#_Toc455739391)

[5.3.11 Package logik.mitarbeiterverwaltung 21](#_Toc455739392)

[5.3.11.1 Beschreibung des Packages 21](#_Toc455739393)

[5.3.11.2 Diagramme 21](#_Toc455739394)

[5.3.11.3 Schnittstellen 21](#_Toc455739395)

[5.3.12 Package gui.produktverwaltung 22](#_Toc455739396)

[5.3.12.1 Beschreibung des Packages 22](#_Toc455739397)

[5.3.12.2 Diagramme 22](#_Toc455739398)

[5.3.12.3 Schnittstellen 22](#_Toc455739399)

[5.3.13 Package logik.produktverwaltung 23](#_Toc455739400)

[5.3.13.1 Beschreibung des Packages 23](#_Toc455739401)

[5.3.13.2 Diagramme 23](#_Toc455739402)

[5.3.13.3 Schnittstellen 23](#_Toc455739403)

[5.3.14 Package gui.schichtverwaltung 24](#_Toc455739404)

[5.3.14.1 Beschreibung des Packages 24](#_Toc455739405)

[5.3.14.2 Diagramme 24](#_Toc455739406)

[5.3.14.3 Schnittstellen 24](#_Toc455739407)

[5.3.15 Package logik.schichtverwaltung 25](#_Toc455739408)

[5.3.15.1 Beschreibung des Packages 25](#_Toc455739409)

[5.3.15.2 Diagramme 25](#_Toc455739410)

[5.3.15.3 Schnittstellen 25](#_Toc455739411)

[5.3.16 Package gui.terminplanung 26](#_Toc455739412)

[5.3.16.1 Beschreibung des Packages 26](#_Toc455739413)

[5.3.16.2 Diagramme 26](#_Toc455739414)

[5.3.16.3 Schnittstellen 26](#_Toc455739415)

[5.3.17 Package logik.terminplanung 27](#_Toc455739416)

[5.3.17.1 Beschreibung des Packages 27](#_Toc455739417)

[5.3.17.2 Diagramme 27](#_Toc455739418)

[5.3.17.3 Schnittstellen 27](#_Toc455739419)

[5.3.18 Package gui.trester 28](#_Toc455739420)

[5.3.18.1 Beschreibung des Packages 28](#_Toc455739421)

[5.3.18.2 Diagramme 28](#_Toc455739422)

[5.3.18.3 Schnittstellen 28](#_Toc455739423)

[5.3.19 Package logik.trester 29](#_Toc455739424)

[5.3.19.1 Beschreibung des Packages 29](#_Toc455739425)

[5.3.19.2 Diagramme 29](#_Toc455739426)

[5.3.19.3 Schnittstellen 29](#_Toc455739427)

[5.3.20 Package gui.verkauf 30](#_Toc455739428)

[5.3.20.1 Beschreibung des Packages 30](#_Toc455739429)

[5.3.20.2 Diagramme 30](#_Toc455739430)

[5.3.20.3 Schnittstellen 30](#_Toc455739431)

[5.3.21 Package logik.verkaufsverwaltung 31](#_Toc455739432)

[5.3.21.1 Beschreibung des Packages 31](#_Toc455739433)

[5.3.21.2 Diagramme 31](#_Toc455739434)

[5.3.21.3 Schnittstellen 31](#_Toc455739435)

[5.3.22 Package persistenz 32](#_Toc455739436)

[5.3.22.1 Beschreibung des Packages 32](#_Toc455739437)

[5.3.22.2 Diagramme 32](#_Toc455739438)

[5.3.22.3 Schnittstellen 32](#_Toc455739439)

[5.3.23 Package Main 33](#_Toc455739440)

[5.3.23.1 Beschreibung des Packages 33](#_Toc455739441)

[5.3.23.2 Diagramme 33](#_Toc455739442)

[5.3.23.3 Schnittstellen 33](#_Toc455739443)

[6 Datenspeicherung (Data View) 34](#_Toc455739444)

[7 Größen und Leistung (Size and Performance) 36](#_Toc455739445)

# Einführung (Introduction)

## Definitionen und Abkürzungen (Definitions, Acronyms, Abbreviations)

Vgl. separates Glossary-Dokument im Repository, Dateiname: Glossary.docx

## Referenzen (References)

Buch: Craig Larman: UML 2 und Patterns angewendet (2005)

## Übersicht (Overview)

Im folgenden Teil dieses Dokuments gehen wir auf die architektonische Darstellung unserer Software ein und beschreiben, welche Ziele und Einschränkungen damit verbunden sind. Außerdem möchten wir insbesondere auf die logische Architektur (logical view) und die Datenspeicherung (data view) eingehen, da diese Views bei unserer Software mehr zum Tragen kommen als andere. Abschließend wird noch beschrieben, welche Größen und Leistungen Mosti umfassen soll.

# Architektonische Darstellung (Architectural Representation)

Ein System lässt sich durch verschiedene architektonische Sichten darstellen. So gibt es beispielsweise Sichten für Logik, Daten, Sicherheit, Implementierung oder auch für Use Cases.

In diesem Dokument möchten wir die Architektur der Mosti-Software mit folgenden Sichten beschreiben:

Logische Sicht (logical view):

Diese Sicht soll die Struktur der Software widerspiegeln. Dies betrifft hier vor allem die Implementierung. So wird beispielsweise beschrieben, wie Packages und Klassen gegliedert bzw. organisiert werden oder auch wie Klassen anhand bestimmter Funktionen interagieren können. Wir werden bei unserer Software das Drei-Schichten-Modell an, welches später genauer erläutert wird.

Daten-Sicht (data view):

Diese Sicht soll einen Überblick darüber geben, wie wir Daten erhalten und verwalten. Bei Mosti setzen wir das mithilfe einer Datenbank um. Die Software hat dabei jederzeit Zugriff auf die Daten, die in der Datenbank gespeichert sind.

*Anmerkung:*

Eine physikalische Sicht ist bei Mosti derzeit nicht relevant, weil die Software so geplant ist, dass sie von einem Anwender nur auf einem bestimmten Rechner benutzt wird und nicht mit anderen Rechnern oder Systemen kommuniziert.

# Architektonische Ziele & Einschränkungen (Architectural Goals and Constraints)

Wir haben uns für eine Drei-Schichten-Architektur entschieden, welche sich aus den Schichten Präsentation (GUI-Schicht), Logik und Datenerhaltung (Persiszenz-Schicht) zusammensetzt. Dabei sollen zwischen den Schichten nur Abhängigkeiten von oben nach unten bestehen. Durch diese Reduktion von Abhängigkeiten können Änderungen in einzelnen Schichten vorgenommen werden, ohne das gesamte System ändern zu müssen. Da wir einen iterativen Software-Entwicklungs-Prozess durchlaufen, können nahezu jederzeit Änderungen in der Implementierung gefordert sein, da sich z. B. nochmals die Anforderungen geändert haben. Es kann auch beispielsweise eine andere Präsentation der Daten gewünscht sein – dann sind nur Änderungen in der sogenannten Präsentations- bzw. GUI-Schicht erforderlich. In den anderen beiden Schichten müssen dabei keinerlei Änderungen vorgenommen werden, da diese sich unterhalb der GUI-Schicht befinden und somit nicht von dieser abhängig sind. Ein weiterer Grund, das Drei Schichten Modell anzuwenden, ist die Tatsache, dass im Mosti-Projekt u. a. Teilfunktionalitäten mit spezifischen GUIs auftauchen, an welchen unterschiedliche Teammitglieder arbeiten. Durch das Schichtenmodell ist eine bestimmte Struktur bzgl. der Implementierung vorgegeben, wodurch andere Teammitglieder einen besseren Überblick und schnelleres Verständnis für Bereiche, die sie nicht selbst implementiert haben, erlangen sollen.

# Logische Architektur (Logical View)

Bei der Entwicklung von Mosti verfolgen wir das Drei-Schichten-Modell. Dabei unterscheiden wir folgende Schichten:

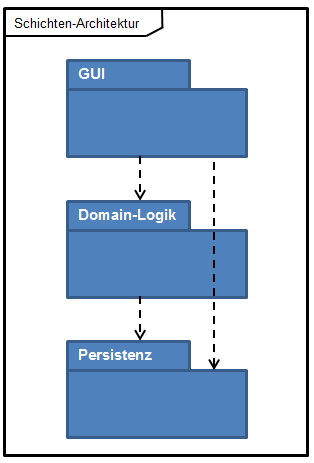
* GUI-Schicht
* Domain-Logik-Schicht
* Persistenz-Schicht (oder auch Datenerhaltungsschicht)

Dieses Modell erläutern wir im folgenden Punkt (5.1 Überblick) noch genauer.

Bezüglich der Package-Struktur streben wir generell an, ähnliche Klassen in einem Package zu gliedern (hohe Kohäsion) und andererseits, dass wenige Abhängigkeiten zwischen den Packages bestehen (geringe Kopplung).

## Übersicht (Overview)

Das Drei-Schichten-Modell für unsere Software lässt sich wie folgt darstellen:



Die Schicht GUI umfasst die grafische Benutzeroberfläche, also die Darstellung der Daten für den Nutzer.

Die Domain-Logik-Schicht ist in dieser Darstellung die Schnittstelle zwischen der GUI-Schicht und der Persistenz-Schicht. Darin werden die spezifischen Anforderungen bzw. Funktionalitäten umgesetzt.

In der Persistenz-Schicht wird die Datenerhaltung geregelt.

Aus dem Diagramm ist zu entnehmen, dass die Abhängigkeit zwischen den Schichten von oben nach unten verläuft. So kann die GUI-Schicht nur auf die Domain-Logik-Schicht und die Persistenz-Schicht zugreifen und die Domain-Logik-Schicht wiederum nur auf die Persistenz-Schicht. So können Domain-Logik-Schicht und Persistenz-Schicht nicht auf die GUI-Schicht zugreifen und machen somit eine unabhängige Entwicklung möglich.

Im Folgenden werden die einzelnen Schichten noch spezifischer dargelegt.

## Design Pakete (Architecturally Significant Design Packages)

In diesem Überpunkt gehen wir nochmals auf die drei einzelnen Schichten ein, die wir hier als Design-Pakete bezeichnen. Hierbei ist zu erwähnen, dass es sich dabei nicht um die von uns erstellten Java-Packages handelt. Da es bei Mosti verschiedene Teilbereiche gibt (z. B. Lagerverwaltung oder Kundenverwaltung), wurden diese jeweils zusammenhängenden Klassen in einem Java-Package zusammengefasst und sind innerhalb dieses Teilbereiches/Java-Packages in die Schichten unterteilt. Diese spezifischen Unterteilungen führen wir im Punkt 5.3 auf.

### Package GUI

#### Beschreibung des Package

Das Package GUI umfasst die graphische Benutzeroberfläche und somit u. a. die Präsentation von Daten, die über die Domain-Logik-Schicht zur Verfügung stehen. Bei Mosti arbeitet der Benutzer selbst ausschließlich über die GUI-Schicht, um Daten zu erfassen, abzurufen oder um bestimmte Funktionen auszuführen. Somit ist es wichtig, dass in diesem Package auch Benutzereingaben erkannt und weitergeleitet werden.

#### Schnittstellen

Die GUI-Schicht hat Zugang zur Domain-Logik-Schicht und zur Persistenz-Schicht. Jedoch kann von anderen Schichten nicht auf die GUI-Schicht zugegriffen werden.

### Package Domain-Logik

#### Beschreibung des Package

Das Package Domain-Logik umfasst die Logik, welche hinter den Funktionalitäten der Software liegt. Somit ist darin geregelt, wie bestimmte Daten verarbeitet werden. Bezogen auf Mosti ist beispielsweise die Funktionalität, welche die Kassenfunktion betrifft, von hoher Bedeutung.

#### Schnittstellen

Die Domain-Logik-Schicht hat Zugang auf die darunterliegende Persistenz-Schicht. Auf die Domain-Logik-Schicht zugreifen kann hingegen nur die GUI-Schicht.

#### Operationen

### Package Persistenz

#### Beschreibung des Package

Im Package Persistenz wird die Datenerhaltung geregelt. Da wir zur Speicherung von Daten eine Datenbank benutzen, soll dieses Package ermöglichen, dass eine Verbindung zur Datenbank aufgebaut wird, man auf die Daten zugreifen und Datenänderungen tätigen kann.

#### Schnittstellen

Die Schnittstelle zwischen unserer Software und der Datenbank wird innerhalb der Persistenz-Schicht gewährleistet. Zugang auf die Persistenz-Schicht haben die Domain-Logik-Schicht und die GUI-Schicht.

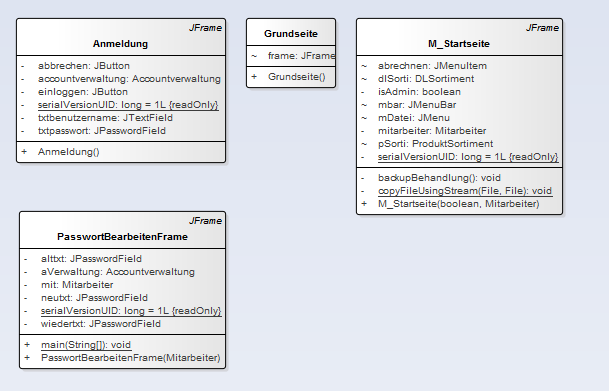
## Java-Pakete und deren Architektur-Design

### Package gui.account

#### Beschreibung des Packages

Das Package *gui.account* umfasst alle Klassen, mit denen die Anmeldung in der Mosti-Software graphisch dargestellt werden. Zusätzlich enthalten sind Klassen zur Editierung des Passwortes und der Hauptframe der Software.

#### Diagramme

****

#### Schnittstellen

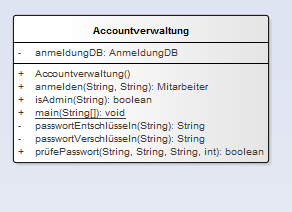
Das Package *gui*.*account* besitzt Schnittstellen zum *logik.account,* da hier die Eingaben des Benutzers überprüft und weiterverarbeitet werden.

### Package logik.account

#### Beschreibung des Packages

Das Package *logik.account* umfasst alle Aspekte, die sich mit der Anmeldung und Authentifizierung in der Mosti-Software befassen. Zusätzlich werden hier die Passwörter der Mitarbeiter verschlüsselt.

#### Diagramme

****

#### Schnittstellen

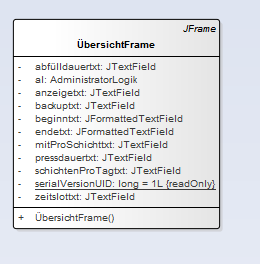
Das Package *logik.account* besitzt Schnittstellen zum Package *gui*.*mitarbeiterverwaltung*, da mit Anlegen eines Mitarbeiters auch sogleich ein Account für diesen erstellt wird. Des Weiteren besteht eine Schnittstelle zum Package *persistenz*, um die eingegebenen Passwörter mit denen in der Datenbank abzugleichen.

### Package gui.administratorverwaltung

#### Beschreibung des Packages

Das Package *gui.account* umfasst alle Klassen zur optischen Aufbereitung der Konfigurations- oder Administratorwerte.

#### Diagramme

****

#### Schnittstellen

Das Package *gui.administratorverwaltung* besitzt seine Schnittstellen zum Package *logik.administratorverwaltung,* von welchem es seine Daten erhält und zum Package *gui.account,* über welches der Benutzer zum selbigen Package gelangt.

### Package logik.administratorverwaltung

#### Beschreibung des Packages

Das Package *logik*.*administratorverwaltung* enthält die Klassen, die für administratorspezifischen Funktionalitäten zuständig sind.

#### Diagramme



#### Schnittstellen

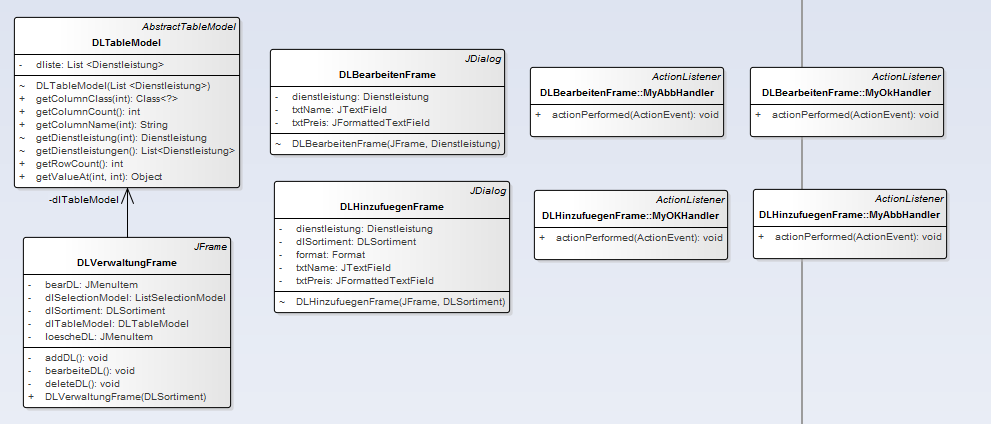
Schnittstellen existieren zu den Packages *gui.administratorverwaltung*  sowie zum Package *persistenz,*  um die in der Datenbank gespeicherten Daten nach ihrer Aufbereitung korrekt anzeigen zu können.

### Package gui.dienstleistungsverwaltung

#### Beschreibung des Packages

Dieses Package enthält alle Klassem zur Darstellung der Dienstleistungsverwaltung. Die darzustellenden Informationen erhält dieses Package vom Package *logik.dienstleistung.*

#### Diagramme



#### Schnittstellen

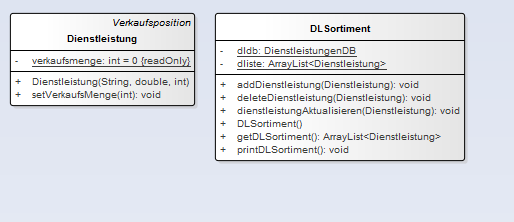
Die wichtigste Schnittstelle des *gui.dienstleistung*-Packages besteht zum *logik.dienstleistung,* um die Informationen über die darstellenden Werte zu erhalten.

### Package logik.dienstleistungenverwaltung

#### Beschreibung des Packages

Das Package *logik.dienstleistungenverwaltung* umfasst alle Aspekte zur Erstellung und Bearbeitung, sowie zum Löschen von einzelnen Dienstleistungen. Das Package bearbeitet die von der Datenbank geladenen Daten und stellt sie dem Package *gui.dienstleistung* zur Verfügung.

#### Diagramme



#### Schnittstellen

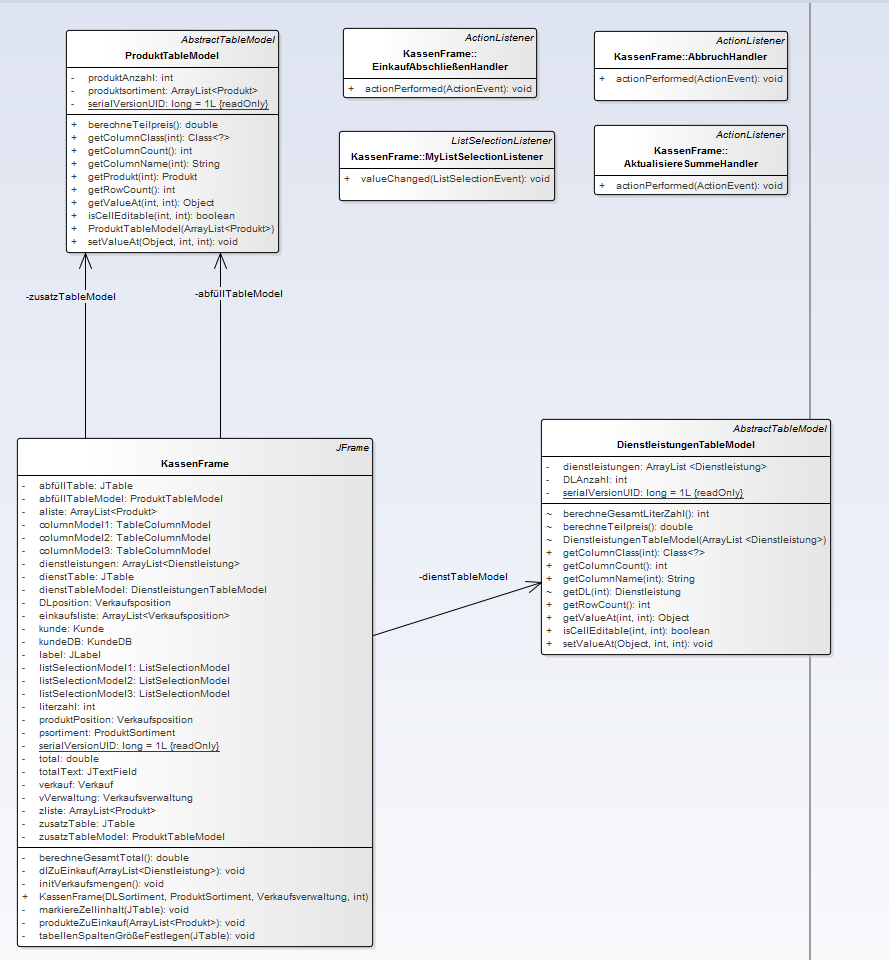
Das Package *logik*.*dienstleistungenverwaltung* besitzt Schnittstellen zum Package *gui.kassenfunktion*, in welcher die vorhandenen Dienstleistungen zum Verkauf präsentiert werden. Eine weitere Schnittstelle existiert zum Package *gui.account*, welches die Startseite der Software beinhaltet. Auf der Startseite kann der Benutzer dann die Verwaltung der Dienstleistungen aufrufen. Außerdem existieren Schnittstellen zu den Packages *gui.dienstleistungsverwaltung* und *persistenz*, um die Dienstleistungen zu speichern und zu präsentieren.

### Package gui.kassenfunktion

#### Beschreibung des Packages

Das Package *gui.kassenfunktion* umfasst alle Aspekte zur Präsentation von der Kasse, getätigten Einkäufen und Abrechungen.

#### Diagramme



#### Schnittstellen

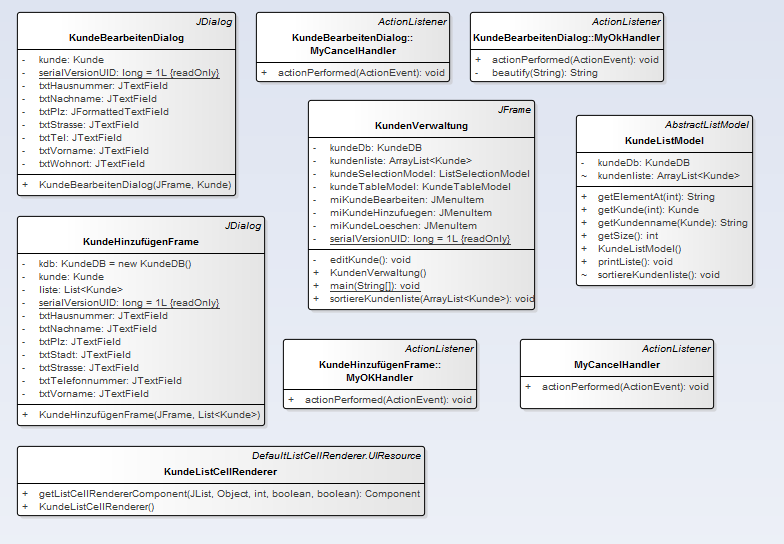
Das Package *gui.kassenfunktion* hat die Schnittstellen *logik.dienstleistung*, und *logik.produktverwaltung,* von welchen es die anzuzeigenden Daten erhält. Zusätzlich existiert eine Schnittstelle zum Package *logik.mitarbeiterverwaltung,* um die Einkäufe im Nachhinein mit einem Kunden in Verbindung bringen zu können.

### Package gui.kundenverwaltung

#### Beschreibung des Packages

Das Package *gui.kundenverwaltung* beinhaltet alle Klassen, um die Mosterei-Kunden und seine spezifischen Daten anzuzeigen und zu editieren.

#### Diagramme

****

#### Schnittstellen

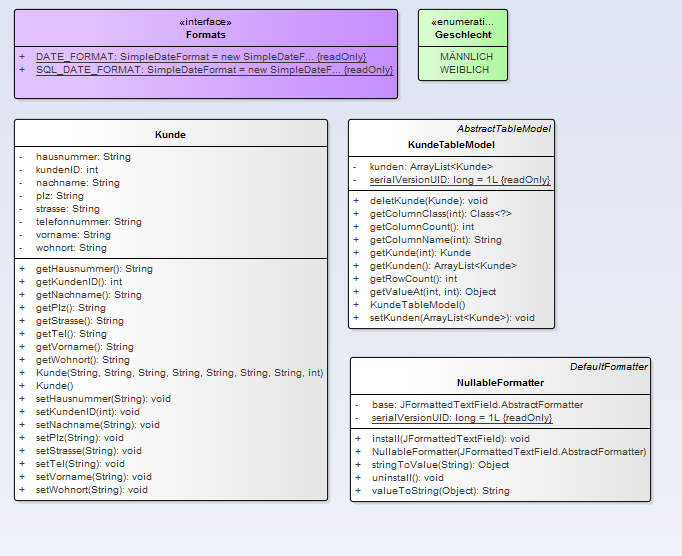
Schnittstellen existieren mit den Packages *logik.kundenverwaltung* und *persistenz,* um Kundendaten von der Datenbank zu laden oder zu speichern und in aufbearbeiteter Form anzuzeigen.

### Package logik.kundenverwaltung

#### Beschreibung des Packages

Das Package *logik.kundenverwaltung* beinhaltet alle Klassen zur Verwaltung der Mosterei-Kunden und seiner spezifischen Kundendaten.

#### Diagramme

****

#### Schnittstellen

Um Kunden mit einzelnen Termin verknüpfen zu können und um die Einkäufe mit den Kunden zu verbinden, besitzt das Package *logik*.*kundenverwaltung*  eine Schnittstelle sowohl zu *logik.terminplanung*  als auch zu *gui*.*kassenfunktion.*

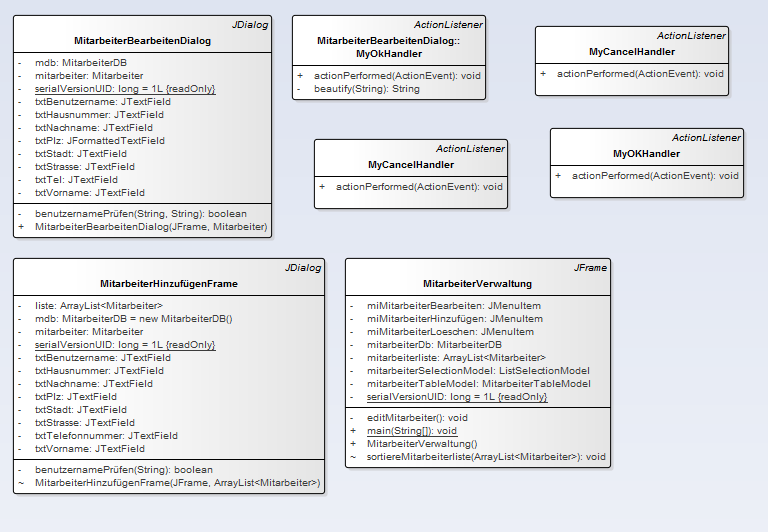
### Package gui.mitarbeiterverwaltung

#### Beschreibung des Packages

Das Package *gui.mitarbeiterverwaltung* enthält die Klassen, mit denen die optische Verwaltung von Mitarbeitern und ihren Daten möglich ist.

.

#### Diagramme

****

#### Schnittstellen

Eine Schnittstelle des Packages *gui*.*mitarbeiterverwaltung* besteht zum Package *logik.mitarbeiterverwaltung,* da hier die Informationen über die anzuzeigenden Daten gespeichert sind.

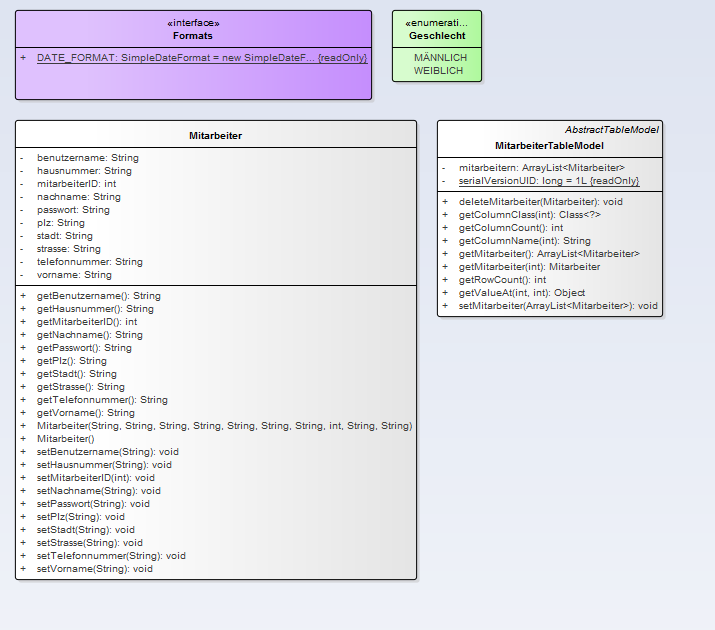
### Package logik.mitarbeiterverwaltung

#### Beschreibung des Packages

Das Package *logik.mitarbeiterverwaltung* beinhaltet Klassen zur Verwaltung (Mitarbeiter hinzufügen, bearbeiten löschen) der Mosterei-Mitarbeiter.

.

#### Diagramme

****

#### Schnittstellen

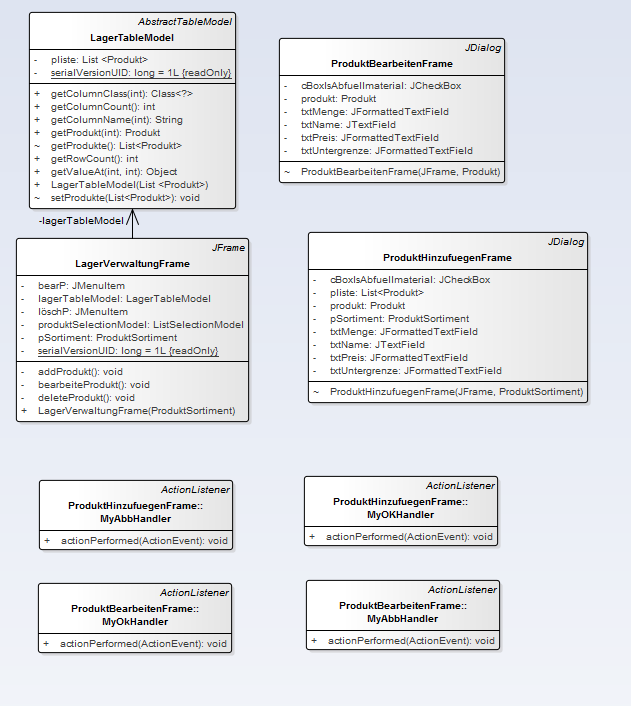
Eine Schnittstelle des Packages *logik*.*mitarbeiterverwaltung* besteht zum Package *logik.account,* da jeder Account mit einem konkreten Mitarbeiter verknüpft wird.

### Package gui.produktverwaltung

#### Beschreibung des Packages

Das Package *gui.produktverwaltung* umfasst alle Aspekte zur Präsentation von Lager und angelegter Produkt sowie Frames zum Hinzufügen bzw. Bearbeiten von Produkten.

#### Diagramme



#### Schnittstellen

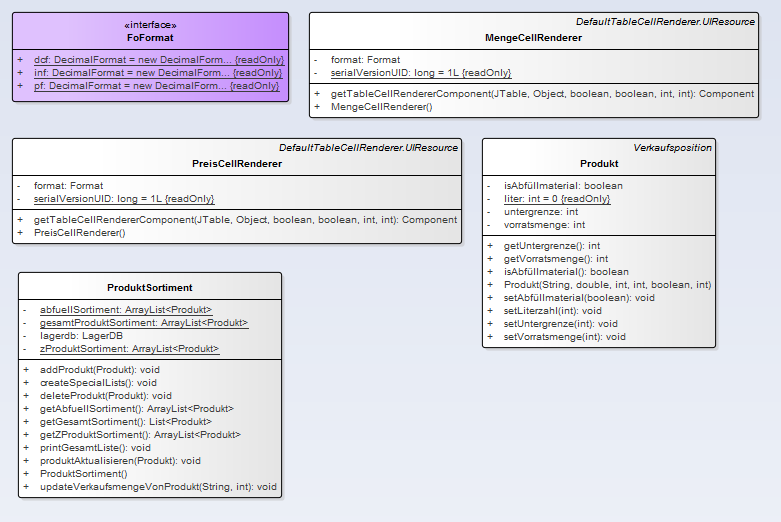
Das Package *gui.produktverwaltung* hat die Schnittstellen zu *logik.produktverwaltung,* von welchem es die anzuzeigenden Informationen erhält.

### Package logik.produktverwaltung

#### Beschreibung des Packages

Das Package logik.produktverwaltung umfasst alle Aspekte der Logischen Ausführungen der Lager- und Produktverwaltung.

#### Diagramme



#### Schnittstellen

Das Package *logik*.*produktverwaltung* besitzt Schnittstellen zu *logik.verkaufsverwaltung*. und *persistenz*.

### Package gui.schichtverwaltung

#### Beschreibung des Packages

In diesem Package sind diejenigen Klassen enthalten, die für die Präsentation der Schichten und des täglichen Schichtplan zuständig sind. Mitarbeiter und Administrator bekommen hier jeweils unterschiedliche Ansichten, die einmal nur eingesehen und einmal auch editiert werden können.

#### Diagramme

#### Schnittstellen

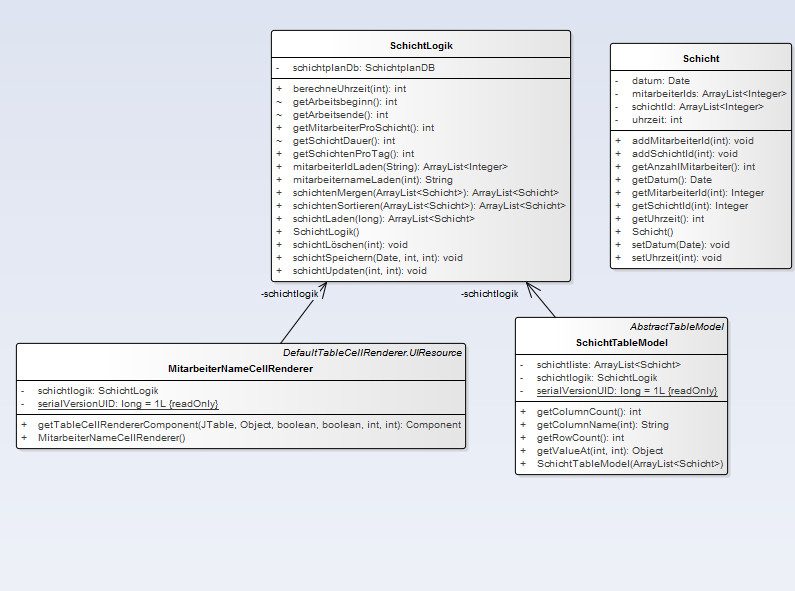
Das Package *gui.schichtverwaltung* besitzt Schnittstellen zum Package *logik*.*schichtverwaltung,* von welchem es die anzuzeigenden Informationen erhält. Eine weitere Schnittstelle existiert zum Package *gui.account,* da der Benutzer über die Startseite zur Bearbeitung bzw. zur Einsicht des Schichtplans gelangt.

### Package logik.schichtverwaltung

#### Beschreibung des Packages

Im Package *logik.schichtverwaltung* werden die in der Datenbank gespeicherten Informationen über die Schichten aufbereitet und dem Package *gui.schichtverwaltung* zur Präsentation übergeben.

#### Diagramme

****

#### Schnittstellen

Eine Schnittstelle existiert zum Package *gui.schichtverwaltung,* welches die Daten dem Benutzer zur Einsicht und zum Editieren zur Verfügung stellt.

### Package gui.terminplanung

#### Beschreibung des Packages

Das Package *gui.terminplanung* enthält alle Klassen, die zur Anzeige der Terminplanung benötigt werden.

#### Diagramme

#### Schnittstellen

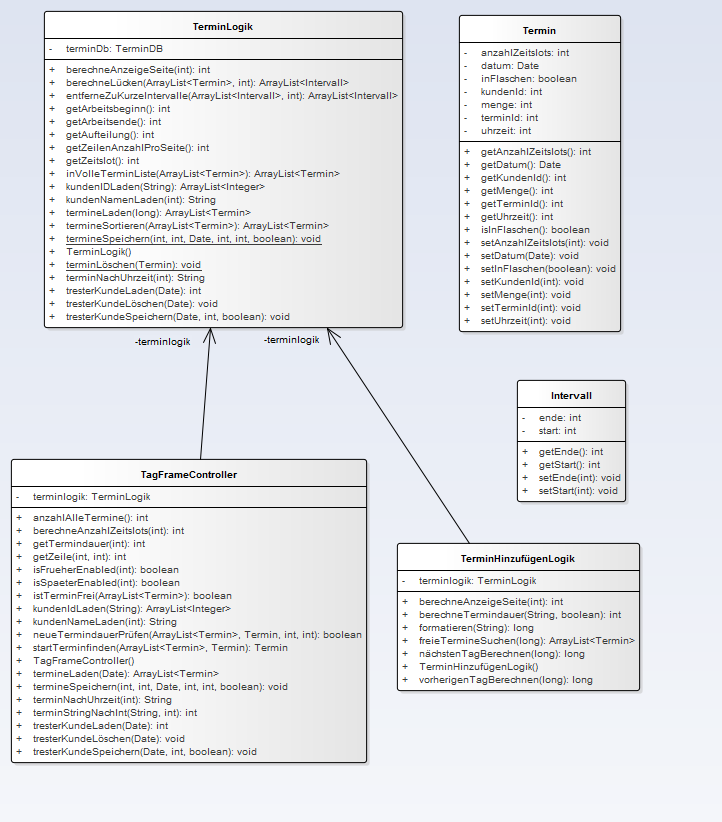
Das Package *gui.terminplanung* besitzt lediglich eine Schnittstelle zum Package *logik.terminplanung,* da hier die Informationen über die darzustellenden Werte enthalten sind.

### Package logik.terminplanung

#### Beschreibung des Packages

Das Package *logik*.*terminplanung* umfasst alle Aspekte der Planung und Speicherung von Terminen. Die hier aufgeführten Klassen managen den Umgang mit den aus der Datenbank herausgelesenen Termindaten und stellen sie den GUI-Klassen zur Anzeige bereit.

#### Diagramme

****

#### Schnittstellen

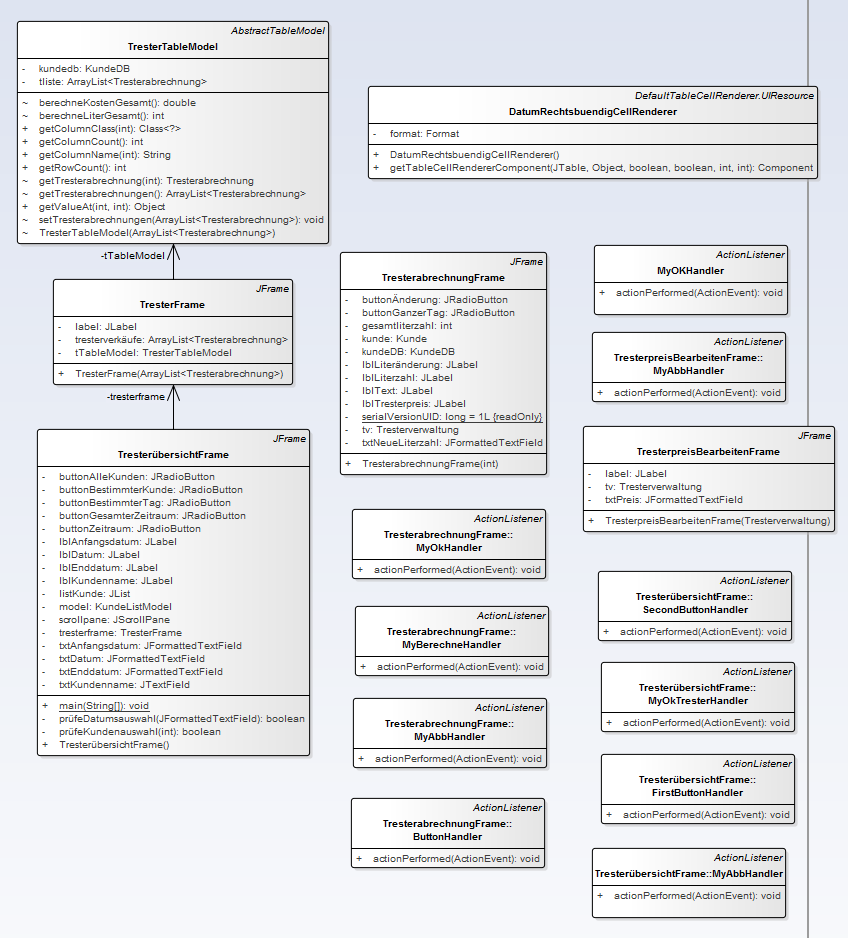
Das Package *logik.terminplanung* besitzt Schnittstellen zum Package *gui.kassenfunktion*, von welcher aus die Kundentermine mit den tatsächlichen Termindurchführung verknüpft werden. Eine weitere Schnittstelle existiert zum Package *gui.account*, welches die Startseite der Software beinhaltet und den Benutzer zwischen den einzelnen Modulen navigiert. Zur Verwaltung der Termine existiert des Weitern eine Schnittstelle *zu gui.terminplanung* und *persistenz.*

### Package gui.trester

#### Beschreibung des Packages

Das Package *gui.trester* enthält die Klassen, in welcher der Benutzer den Preis für den Trester einsehen und verwaltet kann und des Weiteren Abrechnungen für den Trester erstellen kann.

#### Diagramme



#### Schnittstellen

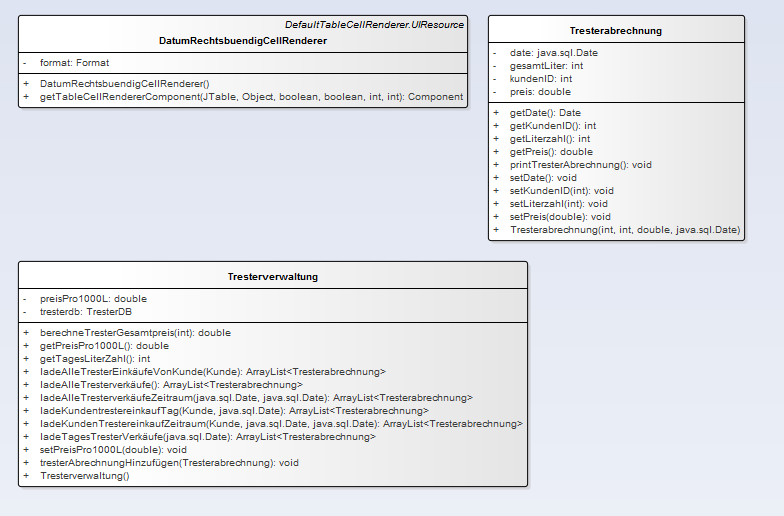
Es besteht eine Schnittstelle zum Package *logik.trester*, in welcher die Logik der Tresterverwaltung bzw. –abrechnung enthält.

### Package logik.trester

#### Beschreibung des Packages

Das Package *logik.trester* enthält die Logik zur Verwaltung des Tresterpreises, sowie zur Erstellung einer Tresterabrechnung.

#### Diagramme



#### Schnittstellen

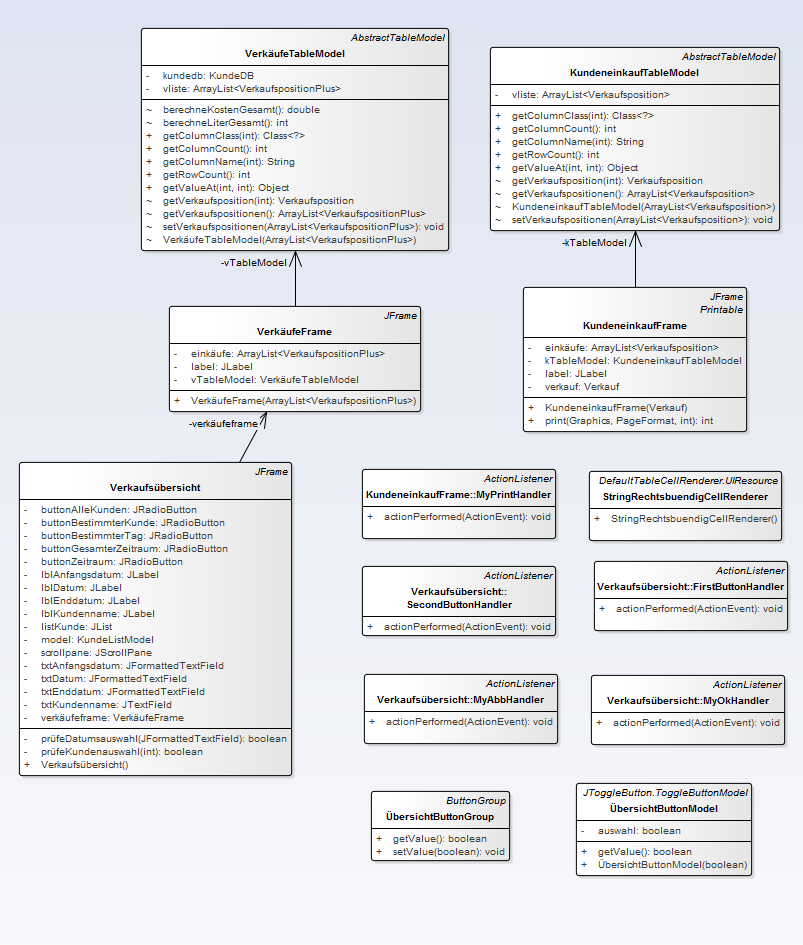
Eine Schnittstelle besteht zu Package *persistenz,* um die aktuellen Tresterpreise aus der Datenbank abfragen zu können und ggf. Änderungen speichern zu können.

### Package gui.verkauf

#### Beschreibung des Packages

Das Package *gui.verkauf* enthält Klassen zur Darstellung von Verkäufen, welche sich jeweils aus einzelnen Verkaufspositionen zusammensetzen. Außerdem ist eine Klasse vorhanden, mit welcher der Benutzer diverse Bedingungen zu dieser Auflistung festlegen kann (z. B. bestimmter Kunde, bestimmter Zeitraum).

#### Diagramme

****

#### Schnittstellen

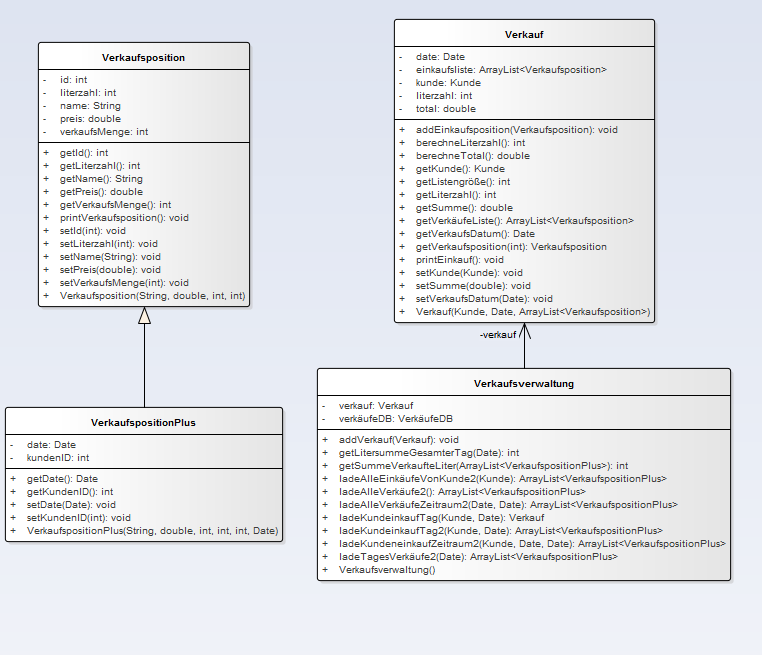
Das Package *gui.verkauf* besitzt Schnittstellen zum Package logik.*verkaufsverwaltung,* in welchem beispielsweise definiert ist, was eine Verkaufsposition ist.

### Package logik.verkaufsverwaltung

#### Beschreibung des Packages

Das Package *logik.verkaufsverwaltung* enthält Klassen, um einen Verkauf zu definieren, welcher aus u. a. aus mehreren Verkaufspositionen aufgebaut ist. Außerdem besitzt es eine Klasse, welche Funktionen der Datenbank, welche die Verkäufe verwaltet, aufruft.

#### Diagramme



#### Schnittstellen

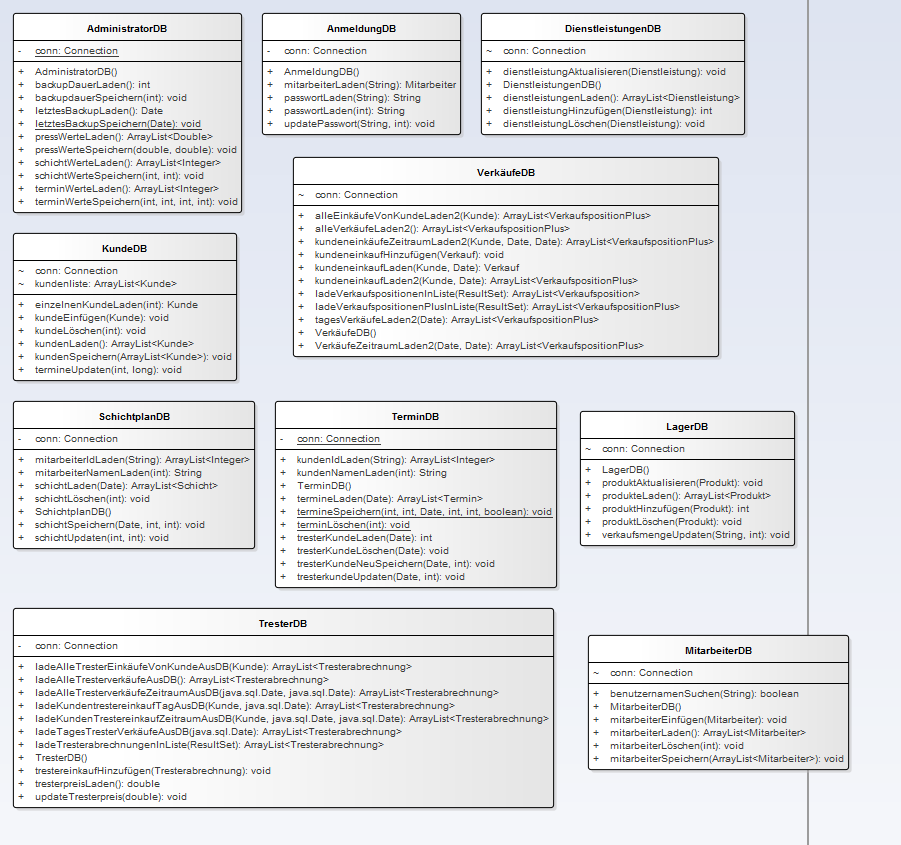
Das Package *logik.verkaufsverwaltung* besitzt Schnittstellen zum Package p*ersistenz,* in welchem der Zugriff auf die Datenbank geregelt ist. Außerdem besteht eine Schnittstellen zum Package *logik*.*kundenverwaltung*. um Einkäufe mit den Kunden in Verbindung zu bringen.

### Package persistenz

#### Beschreibung des Packages

Das Package *persistenz* enthält alle Klassen, die sich um das Laden und Speichern von Daten in die Datenbank kümmern.

#### Diagramme



#### Schnittstellen

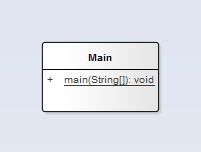
Schnittstellen bestehen zu verschiedenen Logikschichten der anderen Module, z.B. *logik.dienstleistungsverwaltung, logik.terminplanung, logik.produktverwaltung* usw.

### Package Main

#### Beschreibung des Packages

Das Package *main* enthält die Hauptfunktion, von der aus die Software gestartet wird.

#### Diagramme



#### Schnittstellen

Die wichtigste Schnittstelle des *main*-Packages besteht zum Package *gui*.*account,* über welchen die Software durch Anmeldung gestartet werden kann.

# Datenspeicherung (Data View)

Bei Mosti werden die erfassten Daten in einer Datenbank gespeichert. Die Verbindung wird mithilfe eines JDBC-Treibers aufgebaut, der innerhalb der Open-Source Implementierung UCanAccess auf eine Microsoft Access-Datenbank zugreift.

Aufbau der Datenbank:

*adminwerte:*

Enthält Konfigurationswerte, die der Mosterei-Inhaber bei Bedarf ändern kann

Tabellenspalten: ID, AnzeigeAufteilung, Arbeitsbeginn, Arbeitsende, Zeitslotlänge, MitarbeiterProSchicht, SchichtenProTag, Pressdauer, Abfülldauer, backup (beschreibt die Zeitspanne innerhalb welcher regelmäßig ein Backup erstellt werden soll), letzesBackup

*dienstleistungen:*

Enthält Informationen über die in der Mosterei angebotenen Dienstleistungen.

Tabellenspalten: ID, DLName, PreisProLiter

*kunden:*

Enthält die Informationen, die über den Kunden abgespeichert werden.

Tabellenspalten: ID, Vorname, Nachname, hausnummer, PLZ, Wohnort, Telefonnummer, Straße

*mitarbeiter:*

Enthält alle Informationen, die über den Mitarbeiter und seinen Account abgespeichert werden

Tabellenspalten: ID, Vorname, Nachname, Strasse, Hausnummer, PLZ, Stadt, Telefonnummer, Benutzername, Passwort (verschlüsselt)

*produkte:*

Enthält Informationen über alle in der Mosterei angebotenen Produkte.

Tabellenspalten: ID, ProduktName, Preis, Vorratsmenge, Untergrenze, istAbfuellMaterial

*schichtplan:*

Enthält die Daten zu den vergebenen Schichten und dem Schichtplan.

Tabellenspalten: ID, datum, Mitarbeiter, uhrzeit

*termine:*

Enthält alle erstellten Termine und ihre Belegung mit KundenId

TabellenSpalten: ID, Datum, AnzahlZeitslots, Beginn, KundenId, Menge, Flaschen

*tresterpreis:*

Enthält den momentanen Preis des Tresters.

Tabellenspalten: ID, PreisPro1000L

*trestertermine:*

Enthält den Kunden, der am Ende des Tages den Trester abholt.

Tabellenspalten: ID, Datum, KundenId

*tresterverkäufe:*

Enthält Daten über alle bereits abgeschlossenen Tresterverkäufe.

Tabellenspalten: ID, kundenid, tresterliste, tresterpreis, tresterverkaufsdatum

*verkäufe*:

Enthält Informationen über alle bisher abgeschlossenen Verkäufe.

Tabellenspalten: ID, verkaufsposition, verkaufsmenge, literzahl, verkaufsdatum, einzelpreis, kundenid

# Größen und Leistung (Size and Performance)

Die Mosti-Software soll bis zu 5000 Kundeneinträge verwalten können.

Die hohe Anzahl an Datensätzen soll die Leistung der Software nicht merklich dezimieren, sodass alle Funktionen der Software innerhalb von 10 Sekunden ausgeführt werden.