Karlsruhe Institute of Technology

Mensa-App

Applikation für Mitteilung und kollektiven Austausch von Speiseplaninformationen in universitären Gastronomieeinrichtungen

PSE: Implementierung

Praxis der Softwareentwicklung
Sommersemester 2023

Alexander Albers, Peer Booken, Elena Häußler,
Alexander Kutschera, Jonatan Ziegler

28. Juli 2023

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Einleitung

Dieses Dokument dient zur Dokumentation der Implementierungsphase der Mensa-App. Dazu wird im Folgenden auf das allgemeine Vorgehen und Verzögerungen im Ablauf eingegangen. Anschließend werden Änderungen zum Entwurfsdokument dokumentiert. Auch wird beschrieben, welche Muss- und Kannkriterien aus dem Pflichtenheft implementiert wurden. Zuletzt werden einige Daten ausgewertet, die durch GitHub und YouTrack erhoben wurden, sowie die Anzahl der Tests von verschiedenen Komponenten dargestellt.

Im folgenden Dokument sind Änderungen zum Pflichtenheft farbig hervorgehoben. Dabei ist neu hinzugefügtes grün, Geändertes orange und entferntes rot hinterlegt. Zusätzlich sind einige Erklärungen zu den Änderungen blau hinterlegt.

2 Vorgehen

Im Folgenden wird unser Vorgehen zur Organisation des Projektes beschrieben.

2.1 Git

Zur Versionskontrolle und Änderungsnachverfolgung haben wir Github[1] verwendet.
Wir haben mit Branches gearbeitet, um die Arbeit besser parallelisieren zu können. Dafür hatten wir neben dem main-Branch noch jeweils einen für das Front- und das Backend. Von diesen Branches haben wir dann für jede Komponente Branches erstellt. Bevor wir die Änderungen des Branches dann auf den jeweils darunterliegenden Branch gemergt haben, haben wir den gesamten Branch reviewt. Dabei haben wir auf folgende Punkte geachtet:

- Dokumentation
- Logging[2]
- Unittests
- Formatierung
- keine Warnungen vom Compiler bzw. von den verwendeten Codeanalyse-Tools[3]
- Änderungen in diesem Dokument eintragen

2.2 YouTrack

Zum Zeitmanagement, also Arbeitszeitplanung und -erfassung haben wir YouTrack[4] verwendet. Die Plattform bietet zudem noch die Möglichkeit Tickets zu erstellen, die wir zur Organisation und Arbeitsteilung verwendet haben. Außerdem haben wir dort Dateien hochgeladen, die zur internen Koordination dienen. Dazu gehören Besprechungsprotokolle, Absprachen und Dokumentation.

2.3 Meetings

Während der Implementierungsphase haben wir uns in der Gruppe mindestens einmal die Woche in Person getroffen, um uns untereinander über den neuesten Stand des Projektes auszutauschen. Dazu kamen nach zahlreiche Online-Meetings.

[1] `https://github.com/`
[2] Nur im backend
[3] Im Backend wurde dafür Clippy (`https://doc.rust-lang.org/stable/clippy/index.html`) verwendet
[4] `https://www.jetbrains.com/youtrack/`

3 Zeitplan

Das folgende Kapitel zeigt den Zeitplan der Implementierung und dessen Verzüge.

3.1 Zu Beginn der Implementierung

Aufgrund der wöchentlichen Meetings konnten Komponenten einer oder mehreren Wochen zugeteilt werden, sodass sie immer zu einem Meeting besprochen werden konnten. In der Folgenden Tabelle sind keine Aufgaben in der letzten Woche zugeteilt, da zu Beginn der Implementierung eine Woche als Puffer geplant war.

	Woche 1 (ab 26.06.23)	Woche 2 (ab 03.07.23)	Woche 3 (ab 10.07.23)	Woche 4 (ab 17.07.23)	Woche 5 (ab 24.07.23)
Backend					
Interfaces	X				
Paket: util	X				
Datenbankmigration	X				
Komponente: GraphQL	X				
Komponente: MealplanManagement		X			
Komponente: SwKaParser		X			
Komponente: Scheduler		X			
Komponente: Command		X			
Komponente: ImageReview			X		
Komponente: Datenbank			X		
Komponente: FlickrApi			X		
Komponente: Mail			X		
Packet: Startup				X	
Frontend					
Datenklassen	X				
Error-Handling	X				
Interfaces	X				
Core Widgets	X				
Logic	X	X	X		
GraphQLServerAccess		X			
SharedPreferenceAccess		X			
Filter Widgets		X			
Detail-View Widgets		X			
MealPlan Widgets		X			
SQLiteDatabaseAccess			X		
Settings Widgets			X		
Image Widgets			X		
Favorites Widgets			X		
Abgabe Implementierung					X

3.2 Zum Ende der Implementierung

Durch falsche Einschätzungen und unvorhersehbaren Problemen, hat sich der der Zeitplan während der Implementierung an manchen Stellen geändert.

- Die geplante Puffer-Woche am Ende der Implementierung hat sich bezahlt gemacht und wurde vollständig ausgenutzt für das Nachholen von zurückgebliebenen Implementierungen.
- Komponenten der `Model-View` des Frontends beanspruchten mehr Zeit als erwartet und drängten Komponenten der `View` weiter nach hinten.
- Der `MensaParser` hat trotz seinem sehr hoch gesetztem Aufwand länger benötigt als erwartet. Dadurch rutschten Komponenten wie `ImageReview`, `Database` oder `FlickrApi` weiter nach hinten.
- Die `Datenbank` des Backends hatte einen viel größeren Testaufwand als erwartet. Neben dem Zeitdruck zur Abgabe in der selben Woche, musste mehr Zeit pro Tag investiert werden um dessen Funktionalität gewährleisten zu können.

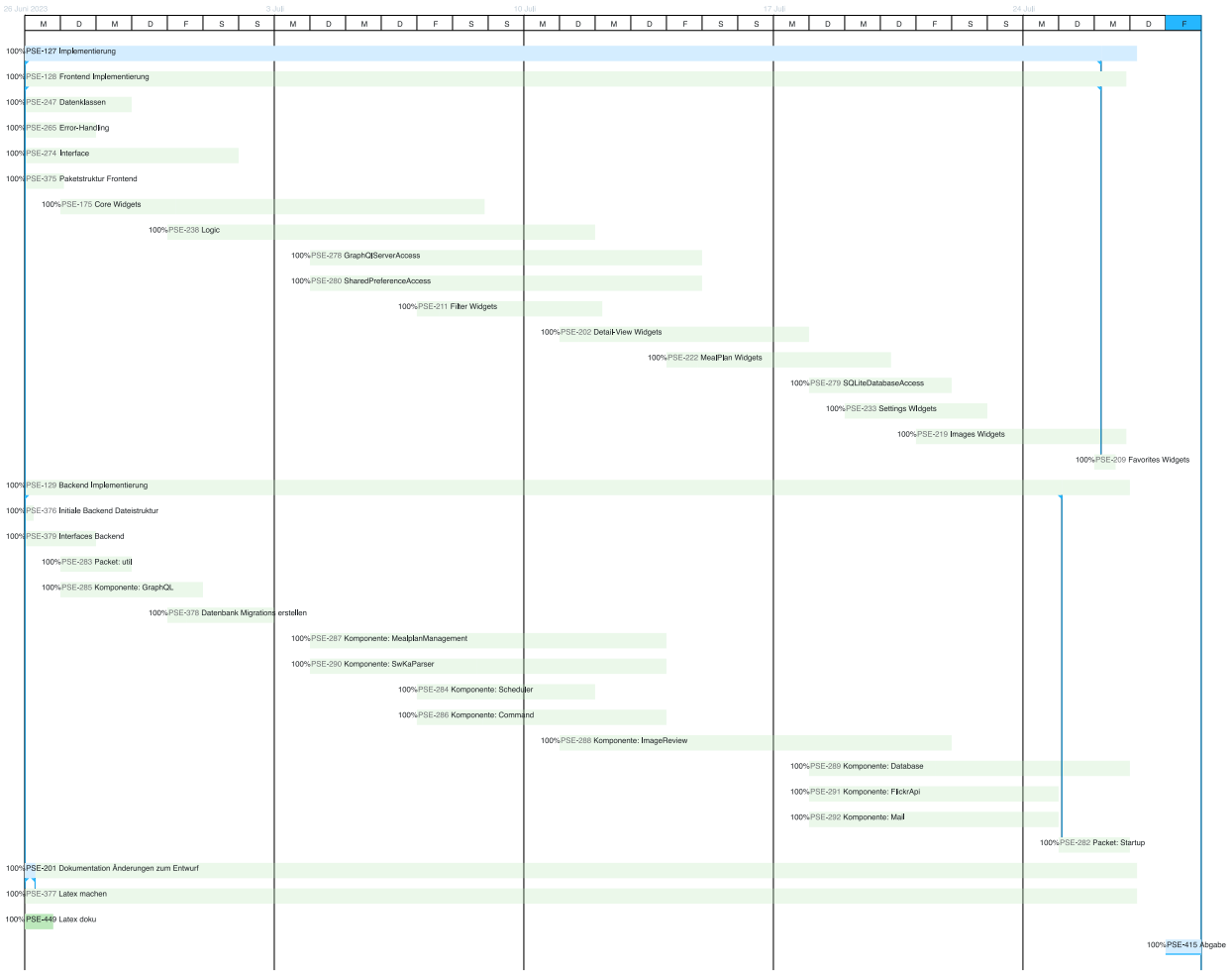

Abbildung 1: Zeitplan am Ende der Implementierung

4 Flussdiagramme

Der folgende Absatz zeigt die Flussdiagramme des Frontends sowie des Backends. Aus den jeweiligen Diagrammen lässt sich das Arbeitsverhalten der beiden Parteien deuten. Legende:

- Grün: Abgeschlossen
- Gelb: Zu besprechen
- Orange: in Bearbeitung
- Rot: Offen

Im Allgemeinen wurden Tickets zu Beginn der Bearbeitung von `offen` auf `in Bearbeitung` gesetzt. Nach der Implementierung wurden die Tickets auf `zu besprechen` gesetzt. Wenn dann alle Tickets einer Komponente fertig implementiert waren, wurde ein Pull-request auf GitHub gestartet. Nachdem dieser reviewed und gemergt wurde, wurden alle zugehörigen Tickets auf `abgeschlossen` gesetzt. Daher gibt es teilweise große Sprünge in der Anzahl an abgeschlossenen Tickets. Im Frontend fallen die Sprünge kleiner aus, da dort öfter und kleinere Pull-requests gestartet wurden.
Zu Beginn der Implementierungsphase wurden alle Datenklassen und Strukturen in kurzer Zeit abgeschlossen. Bis zur letzten Woche ist die Abschlussrate pro Woche geringer, da die Komponenten mehr Besprechnungsaufwand und einen höheren Arbeitsaufwand haben.

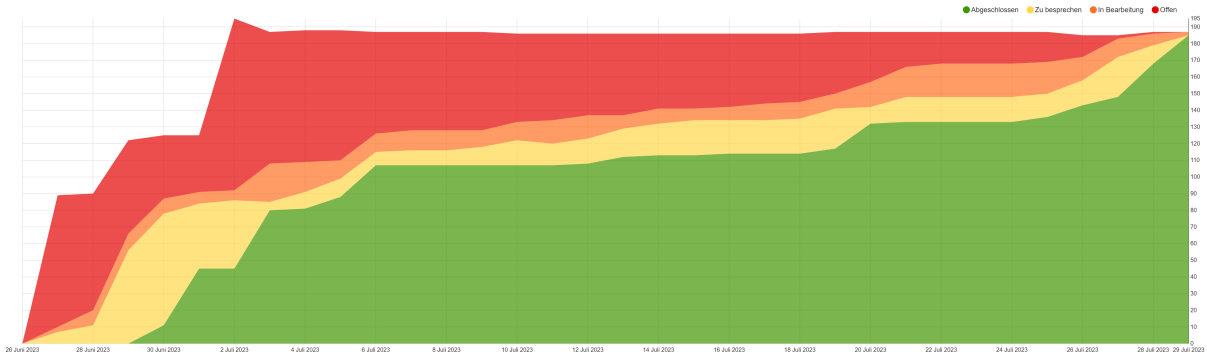

Abbildung 2: Flowdiagramm für das gesamte Projekt

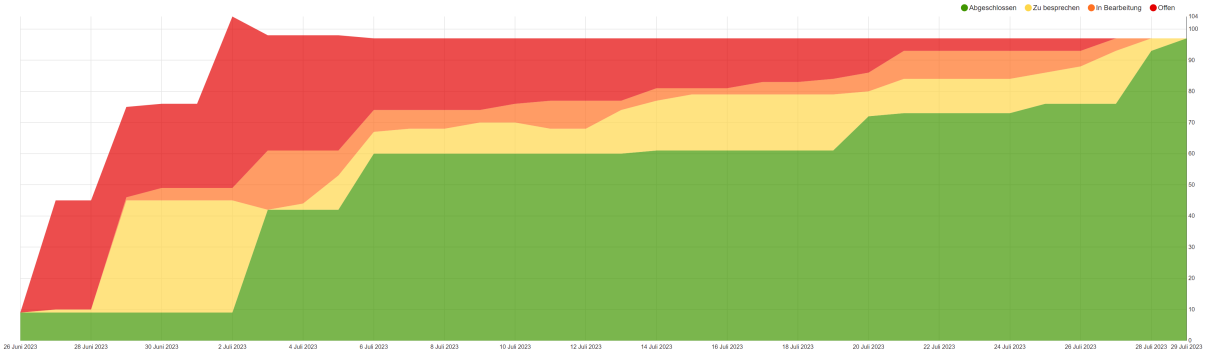

Abbildung 3: Flussdiagramm für das Backend

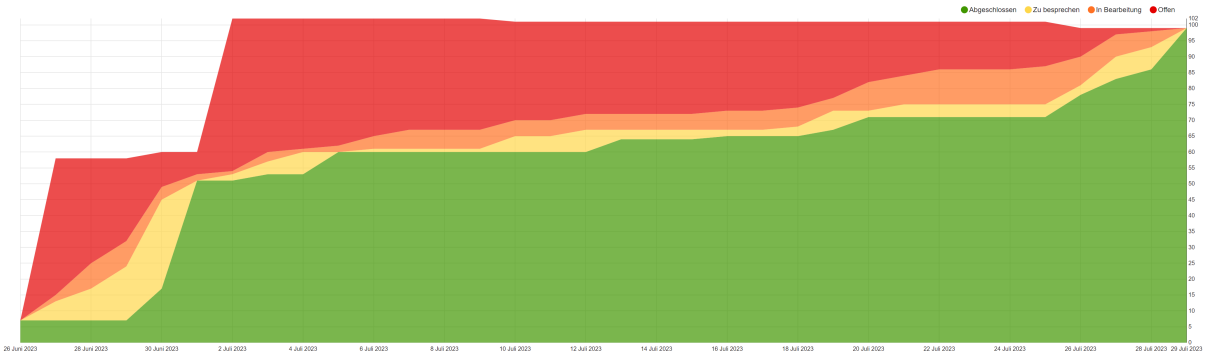

Abbildung 4: Flussdiagramm für das Frontend

5 Git Statistiken

Das folgende Diagramm zeigt die Aktivität auf unserem Git Repository nach Wochentag.

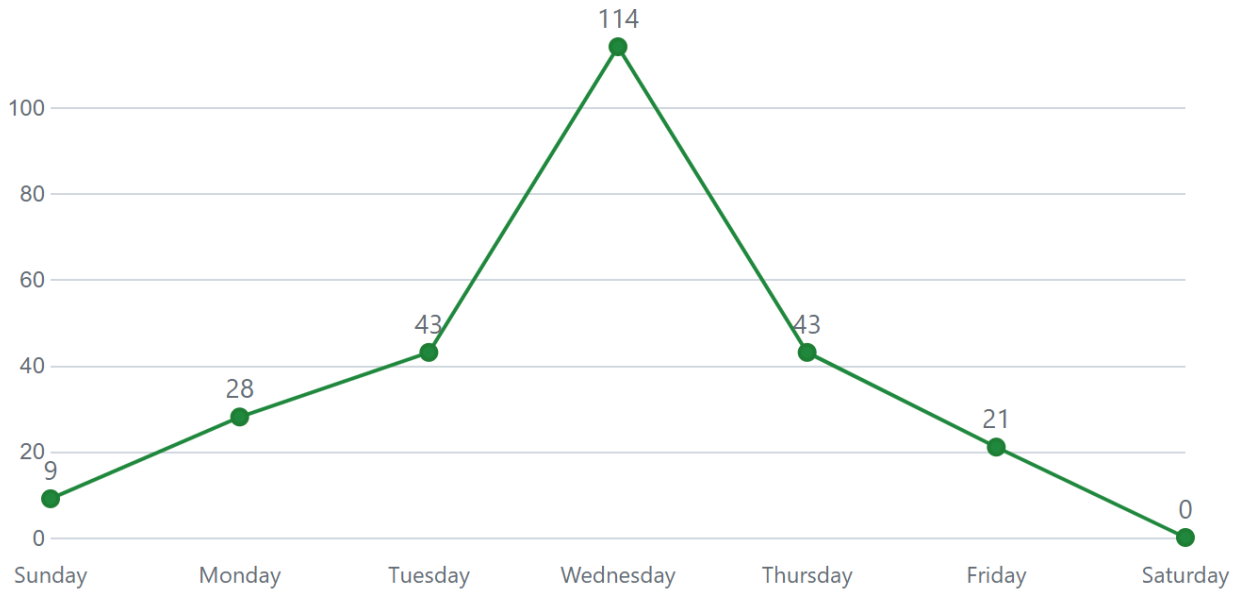

Abbildung 5: Commits auf das Git Repository nach Wochentag

6 Änderungen zum Entwurf

Im Folgenden wird angegeben, welche Muss- und Kannkriterien implementiert wurden.

6.1 Musskriterien

Es wurden alle Musskriterien implementiert. Das umfasst:

MK01 Gerichte anzeigen

- zu einem bestimmten Tag bis zu einem Monat im Voraus
- für alle Mensen des Studierendenwerk Karlsruhe
- inklusive ihrer Allergene und Zusatzstoffe

MK02 Preise für die verschiedenen Preisklassen anzeigen

MK03 Gerichte nach Gerichtstyp und Allergene filtern

MK04 Bilder zu Gerichten verlinken und anzeigen

MK05 Von Nutzern verlinkte Bilder melden

6.2 Kannkriterien

Es wurden nicht alle Kannkriterien implementiert. Es wurden alle Kannkriterien implemnetiert, die im Entwurf modelliert wurden.

Implementierte Kannkriterien

Folgende Kriterien wurden modelliert und implementiert.

KK01 Lightmode / Darkmode der App

KK02 Gerichte favorisieren

KK03 Gerichte mit ein bis fünf Sternen bewerten

KK04 Bilder zu Gerichten als hilfreich oder nicht hilfreich bewerten

KK05 Häufigkeitsklassen der Gerichte anzeigen

KK08 Gerichte nach Preis, Bewertung, Favoriten und Häufigkeitsklassen filtern

KK09 Gerichte nach Linie, Bewertung, Preis und Häufigkeitsklassen sortieren

Nichtimplementierte Kannkriterien

Folgende Kriterien wurden weder modelliert noch implementiert.

KK06 Benachrichtigung für favorisierte Gerichte

KK07 Favoriten und Einstellungen im- und exportieren

7 Änderungen zum Entwurf: Frontend

ℹ Im Frontend wurde bei den meisten Widgets kein Konstruktor angegeben. Die meisten Widgets benötigen allerdings einen Konstruktor. Dies ist im Laufe der Implementierung aufgefallen und wurde dementsprechend hinzugefügt. Auch wurden einzelne Widgets (Icons ausgenommen) hinzugefügt oder entfernt, da sie benötigt / nicht benötigt wurden.

ℹ Außerdem wurde bei der Implementierung festgestellt, dass es in einigen Fällen von Vorteil ist, wenn man festlegt, um welche Exceptions es sich handlen kann, wenn etwas fehlschlägt. Darum kam es hier zu einigen Änderungen. Das verursacht auch Änderungen in der Signatur aller Methoden, die `Result` als Rückgabewert haben.

7.1 View-Core

MensaApp

Typ: `class` extends `StatelessWidget`
Paket: `view/core`
Beschreibung: Dieses Widget zeigt die gesamte Mensa-App an.
Methoden:

\+ `MensaApp(key: Key, delegate: FlutterI18nDelegate) «create»`
Konstruktor für das MensaApp Widget

MainPage

Typ: `class` extends `StatefulWidget`
Paket: `view/core`
Beschreibung: Dieses Widget zeigt den Rahmen für die Hauptansichten an.

MensaNavigationBar

Typ: `class` extends `StatelessWidget`
Paket: `view/core`
Beschreibung: Dieses Widget zeigt die Menüleiste in der MainPage an.

MensaAppBar

Typ: `class` extends `StatelessWidget`
Paket: `view/core`
Beschreibung: Dieses Widget zeigt die obere App-Leiste mit der aktuell gewählten Mensa an.
Methoden:

\+ `MensaAppBar(key: Key, optional buttom: PreferredSizeWidget, optional appBarHeight: double, child: Widget) «create»`
Konstruktor für das MensaAppBar Widget

MealGrid

Typ: `class` extends `StatelessWidget`

Paket: `view/core/meal-view-format`

Beschreibung: Dieses Widget zeigt Gerichte im Galerie-Modus an.

Methoden:

\+ `MealGrid(key: Key, mealPlans: List<MealPlan>) «create»`
Konstruktor für das MealGrid Widget

MealGridLine

Typ: `class` extends `StatelessWidget`

Paket: `view/core/meal-view-format`

Beschreibung: Dieses Widget zeigt eine Mensa-Linie mit ihren Gerichten des MealGrid an.

Methoden:

\+ `MealGridLine(key: Key, mealPlan: MealPlan) «create»`
Konstruktor für das MealGridLine Widget

MealGridEntry

Typ: `class` extends `StatelessWidget`

Paket: `view/core/meal-view-format`

Beschreibung: Dieses Widget zeigt ein Gericht in einer MealGridLine an.

Methoden:

\+ `MealGridEntry(key: Key, meal: Meal, width: double) «create»`
Konstruktor für das MealGridEntry Widget

MealList

Typ: `class` extends `StatelessWidget`

Paket: `view/core/meal-view-format`

Beschreibung: Dieses Widget zeigt Gerichte im Listen-Modus an.

Methoden:

\+ `MealList(key: Key, mealPlans: List<MealPlan>) «create»`
Konstruktor für das MealList Widget

MealListLine

Typ: `class` extends `StatelessWidget`

Paket: `view/core/meal-view-format`

Beschreibung: Dieses Widget zeigt eine Mensa-Linie mit ihren Gerichten der MealList an.

Methoden:

\+ `MealGridLine(key: Key, mealPlan: MealPlan) «create»`
Konstruktor für das MealGridLine Widget

MealListEntry

Typ: `class` extends `StatelessWidget`
Paket: `view/core/meal-view-format`
Beschreibung: Dieses Widget zeigt ein Gericht in einer MealListLine an.
Methoden:

\+ `MealListEntry(key: Key, meal: Meal) «create»`
Konstruktor für das MealListEntry Widget

MealPreviewImage

Typ: `class` extends `StatelessWidget`
Paket: `view/core/information-display`
Beschreibung: Dieses Widget zeigt ein Vorschaubild eines Gerichts für MealGridEntry oder MealListEntry an.
Methoden:

\+ `MealPreviewImage(key: Key, meal: Meal, optional displayFavorite: boolean, optional enableUploadButton: boolean, optional enable FavoriteButton: boolean, optinoal borderRadius: BorderRadius, optional height: double, optional width: double, optional onUploadButtonPressed: Function, ooptional onImagePressed: Function) «create»`
Konstruktor für das MealPreviewImage Widget

MealRating

Typ: `class` extends `StatelessWidget`
Paket: `view/core/information-display`
Beschreibung: Dieses Widget zeigt eine Sterne-Bewertung eines Gerichts an.

MealMainEntry

Typ: `class` extends `StatelessWidget`
Paket: `view/core/information-display`
Beschreibung: Dieses Widget zeigt eine Beschreibung eines Gerichts inklusive Name, Gerichtstyp und Preis an.
Methoden:

\+ `MealMainEntry(key: Key, meal: Meal) «create»`
Konstruktor für das MealMainEntry Widget

MealSideEntry

Typ: `class` extends `StatelessWidget`
Paket: `view/core/information-display`
Beschreibung: Dieses Widget zeigt eine Beschreibung einer Beilage inklusive Name, Gerichtstyp und Preis an.

Methoden:

+ `MealSideEntry(key: Key, side: Side) «create»`
  Konstruktor für das MealSideEntry Widget

MensaRatingInput

Typ: `class` extends `StatelessWidget`

Paket: `view/core/input-components`

Beschreibung: Dieses Widget zeigt Schaltflächen zum Wählen einer Sternebewertung an.

Methoden:

+ `MensaRatingInput(key: Key, onChanged: function(int), value: double, optional max: int, optional color: Color, optional disabled: bool, optional size: double) «create»`
  Konstruktor für das MensaRatingInput Widget

MensaTextField

Typ: `class` extends `StatelessWidget`

Paket: `view/core/input-components`

Beschreibung: Dieses Widget zeigt Textfeld an.

Methoden:

+ `MensaTextField(key: Key, controller: TextEditingController) «create»`
  Konstruktor für das MensaTextField Widget

MensaCheckbox

Typ: `class` extends `StatelessWidget`

Paket: `view/core/selection-components`

Beschreibung: Dieses Widget zeigt eine Checkbox an.

Methoden:

+ `MensaCheckbox(key: Key, onChanged: Function(boolean), value: boolean, label: String) «create»`
  Konstruktor für das MensaCheckbox Widget

MensaSlider

Typ: `class` extends `StatelessWidget`

Paket: `view/core/selection-components`

Beschreibung: Dieses Widget zeigt einen Slider an.

Methoden:

+ `MensaSlider(key: Key, onChanged: Function(couble), value: double, optional min: double, optional max: double, optional devisions: int) «create»`
  Konstruktor für das MensaSlider Widget

MensaToggle

Typ: `class` extends `StatelessWidget`
Paket: `view/core/selection-components`
Beschreibung: Dieses Widget zeigt einen Schalter an.
Methoden:

+ `MensaToggle(key: Key, onChanged: Function(boolean), value: boolean, label: String) «create»`
  Konstruktor für das MensaToggle Widget

MensaDropdown

Typ: `class` extends `StatelessWidget`
Paket: `view/core/selection-components`
Beschreibung: Dieses Widget zeigt eine Dropdown-Liste an.
Methoden:

+ `MensaDropdown<T>(key: Key, onChanged: Function(T), value: T, items: List<MensaDropdownEntry<T>», optional backgroundColor: Color) «create»`
  Konstruktor für das MensaDropdown Widget

MensaDropdownEntry

Typ: `class` extends `StatelessWidget`
Paket: `view/core/selection-components`
Beschreibung: Dieses Widget zeigt einen Eintrag einer Dropdown-Liste an.
Methoden:

+ `MensaDropdownEntry<T>(key: Key, value: T, label: String) «create»`
  Konstruktor für das MensaDropdownEntry Widget

MensaIconButton

Typ: `class` extends `StatelessWidget`
Paket: `view/core/buttons`
Beschreibung: Dieses Widget zeigt eine Schaltfläche mit Icon an.
Methoden:

+ `MensaIconButton(optional key: Key, onPressed: Function(), icon: Icon) «create»`
  Konstruktor für das MensaIconButton

MensaButton

Typ: `class` extends `StatelessWidget`
Paket: `view/core/buttons`
Beschreibung: Diess Widget zeigt eine Schaltfläche an.
Methoden:

+ `MensaButton(optional key: Key, onPressed: Function(), optional onLongPressed: Function(), text: String) «create»`
  Konstruktor für das MensaButton

MensaTapable

Typ: `class` extends `StatelessWidget`
Paket: `view/core/buttons`
Beschreibung: Diess Widget zeigt eine Schaltfläche an.
Methoden:

+ `MensaTapable(optional key: Key, child: Widget, onTap: Function(), optional color: Color, optional onLongPressed: Function()) «create»`
  Konstruktor für das MensaTapable

MensaLink

Typ: `class` extends `StatelessWidget`
Paket: `view/core/buttons`
Beschreibung: Dieses Widget zeigt eine Schaltfläche mit einem Link zu einer Webseite an.
Methoden:

+ `MensaLink(optional key: Key, onPressed: Function(), optional onLongPressed: Function(), text: String) «create»`
  Konstruktor für das MensaLink

MensaCtaButton

Typ: `class` extends `StatelessWidget`
Paket: `view/core/buttons`
Beschreibung: Dieses Widget zeigt eine Schaltfläche für eine Aktion an.
Methoden:

+ `MensaCtaButton(optional key: Key, onPressed: Function(), optional onLongPressed: Function(), text: String) «create»`
  Konstruktor für das MensaCtaButton

MensaDialog

Typ: `class` extends `StatelessWidget`
Paket: `view/core/dialogs`
Beschreibung: Dieses Widget zeigt einen allgemeinen Dialog an, der nicht den gesamten Bildschirm einnimmt.
Methoden:

+ `MensaDialog(optional key: Key, title: String, optional content: Widget, optional actions: Widget) «create»`
  Konstruktor für das MensaDialog

MensaFullscreenDialog

Typ: `class` extends `StatelessWidget`
Paket: `view/core/dialogs`
Beschreibung: Dieses Widget zeigt einen allgemeinen Dialog an, der den gesamten Bildschirm einnimmt.
Methoden:

+ `MensaDialog(optional key: Key, optional appBar: PreferredSizeWidget, optional content: Widget, optional actions: Widget)` «create»
  Konstruktor für das MensaDialog

7.2 View-Icons

Es wurde festgestellt, dass für die Icons je eine Klasse benötigt wird. Darum wird (aufgrund der großen Anzahl an Icons) eine dementsprechend große Menge an Klassen hinzugefügt.

CanteenClosedExceptionIcon

Typ: `class` extends `StatelessWidget`
Paket: `view/core/icons/exceptions`
Beschreibung: Dieses Klasse zeigt das Icon für eine geschlossene Mensa an.
Methoden:

+ `CanteenClosedExceptionIcon(key: Key, size: double, color: Color)` «create»
  Konstruktor für das CanteenClosedExceptionIcon.

ErrorExceptionIcon

Typ: `class` extends `StatelessWidget`
Paket: `view/core/icons/exceptions`
Beschreibung: Dieses Klasse zeigt das Icon für keine Verbindung zum Server an.
Methoden:

+ `ErrorExceptionIcon(key: Key, size: double, color: Color)` «create»
  Konstruktor für das ErrorExceptionIcon.

FilterExceptionIcon

Typ: `class` extends `StatelessWidget`
Paket: `view/core/icons/exceptions`
Beschreibung: Dieses Klasse zeigt das Icon für keine Gerichte, die dem Filter entsprechen an.
Methoden:

+ `FilterExceptionIcon(key: Key, size: double, color: Color)` «create»
  Konstruktor für das FilterExceptionIcon.

NoDataExceptionIcon

Typ: `class` extends `StatelessWidget`
Paket: `view/core/icons/exceptions`
Beschreibung: Dieses Klasse zeigt das Icon dafür an, dass es zum aktuellen Zeitpunkt keine Speiseplaninformationen auf dem Server gibt.
Methoden:

+ `NoDataExceptionIcon(key: Key, size: double, color: Color)` «create»
  Konstruktor für das NoDataExceptionIcon.

MealIcon

Typ: `class` extends `StatelessWidget`
Paket: `view/core/icons`
Beschreibung: Dieses Klasse zeigt das entsprechende Icon für den Übergebenen Gerichtstypen an.
Methoden:

+ `MealIcon(width: double, height: double, foodType: FoodType)` «create»
  Konstruktor für das MealIcon

MealBeefAwIcon

Typ: `class` extends `StatelessWidget`
Paket: `view/core/icons`
Beschreibung: Dieses Klasse zeigt das Icon für Rindfleisch aus artgerechter Tierhaltung an.
Methoden:

+ `MealBeefAwIcon(width: double, height: double)` «create»
  Konstruktor für das MealBeefAwIcon

MealBeefIcon

Typ: `class` extends `StatelessWidget`
Paket: `view/core/icons`
Beschreibung: Dieses Klasse zeigt das Icon für Rindfleisch an.
Methoden:

+ `MealBeefIcon(width: double, height: double)` «create»
  Konstruktor für das MealBeefIcon

MealFishIcon

Typ: `class` extends `StatelessWidget`
Paket: `view/core/icons`
Beschreibung: Dieses Klasse zeigt das Icon für Fisch an.
Methoden:

`+ MealFishIcon(width: double, height: double) «create»`
Konstruktor für das MealFishIcon

MealPorkAwIcon

Typ: `class` extends `StatelessWidget`
Paket: `view/core/icons`
Beschreibung: Dieses Klasse zeigt das Icon für Schweinefleisch aus artgerechter Tierhaltung an.
Methoden:

`+ MealPorkAwIcon(width: double, height: double) «create»`
Konstruktor für das MealPorkAwIcon

MealPorkIcon

Typ: `class` extends `StatelessWidget`
Paket: `view/core/icons`
Beschreibung: Dieses Klasse zeigt das Icon für Schweinefleisch an.
Methoden:

`+ MealPorkIcon(width: double, height: double) «create»`
Konstruktor für das MealPorkIcon

MealVeganIcon

Typ: `class` extends `StatelessWidget`
Paket: `view/core/icons`
Beschreibung: Dieses Klasse zeigt das Icon für ein veganes Gericht an.
Methoden:

`+ MealVeganIcon(width: double, height: double) «create»`
Konstruktor für das MealVeganIcon

MealVegetarianIcon

Typ: `class` extends `StatelessWidget`
Paket: `view/core/icons`
Beschreibung: Dieses Klasse zeigt das Icon für ein vegetarisches Gericht an.
Methoden:

+ `MealVegetarianIcon(width: double, height: double) «create»`
  Konstruktor für das MealVegetarianIcon

IAllergenIcon

Typ: `class` extends `StatelessWidget`
Paket: `view/core/icons/allergens`
Beschreibung: Dieses Klasse dient als abstrakte Klasse für alle Standardklasse von welcher die einzelnen AllergenIcons erben können.
Methoden:

+ `AllergenIcon(width: double, height: double, color: Color) «create»`
  Konstruktor für das AllergenIcon
+ `getColor(): Color`
  Gibt die Farbe des Icons zurück.
+ `getHight(): double`
  Gibt die Höhe des Widgets zurück
+ `getWidth(): double`
  Gibt die Breide des Widgets zurück

AllergenIcon

Typ: `class` extends `IAllergenIcon`
Paket: `view/core/icons/allergens`
Beschreibung: Dieses Klasse dient als Standardklasse von welcher die einzelnen Allergen Icons erben können.
Methoden:

+ `AllergenIcon(width: double, height: double, color: Color) «create»`
  Konstruktor für das AllergenIcon

AllergenAlmondsIcon

Typ: `class` extends `IAllergenIcon`
Paket: `view/core/icons/allergens`
Beschreibung: Dieses Klasse zeigt das Icon für Mandeln an.
Methoden:

`+ AllergenAlmondsIcon(width: double, height: double, color: Color)` «create»
Konstruktor für das AllergenAlmondsIcon

AllergenBarleyIcon

Typ: `class` extends `IAllergenIcon`
Paket: `view/core/icons/allergens`
Beschreibung: Dieses Klasse zeigt das Icon für Gerste an.
Methoden:

`+ AllergenBarleyIcon(width: double, height: double, color: Color)` «create»
Konstruktor für das AllergenBarleyIcon

AllergenBrazilNutsIcon

Typ: `class` extends `IAllergenIcon`
Paket: `view/core/icons/allergens`
Beschreibung: Dieses Klasse zeigt das Icon für Paranüsse an.
Methoden:

`+ AllergenBrazilNutsIcon(width: double, height: double, color: Color)` «create»
Konstruktor für das AllergenBrazilNutsIcon

AllergenCashewsIcon

Typ: `class` extends `IAllergenIcon`
Paket: `view/core/icons/allergens`
Beschreibung: Dieses Klasse zeigt das Icon für Cashewnüsse an.
Methoden:

`+ AllergenCashewsIcon(width: double, height: double, color: Color)` «create»
Konstruktor für das AllergenCashewsIcon

AllergenCeleryIcon

Typ: `class` extends `IAllergenIcon`
Paket: `view/core/icons/allergens`
Beschreibung: Dieses Klasse zeigt das Icon für Sellerie an.
Methoden:

`+ AllergenCeleryIcon(width: double, height: double, color: Color)` «create»
Konstruktor für das AllergenCeleryIcon

AllergenCrustaceansIcon

Typ: `class` extends `IAllergenIcon`
Paket: `view/core/icons/allergens`
Beschreibung: Dieses Klasse zeigt das Icon für Krebstiere an.
Methoden:

`+ AllergenCrustaceansIcon(width: double, height: double, color: Color)` «create»
Konstruktor für das AllergenCrustaceansIcon

AllergenEggsIcon

Typ: `class` extends `IAllergenIcon`
Paket: `view/core/icons/allergens`
Beschreibung: Dieses Klasse zeigt das Icon für Eier an.
Methoden:

`+ AllergenEggsIcon(width: double, height: double, color: Color)` «create»
Konstruktor für das AllergenEggsIcon

AllergenFishIcon

Typ: `class` extends `IAllergenIcon`
Paket: `view/core/icons/allergens`
Beschreibung: Dieses Klasse zeigt das Icon für Mandeln an.
Methoden:

`+ AllergenFishIcon(width: double, height: double, color: Color)` «create»
Konstruktor für das AllergenFishIcon

AllergenGelatineIcon

Typ: `class` extends `IAllergenIcon`
Paket: `view/core/icons/allergens`
Beschreibung: Dieses Klasse zeigt das Icon für Gelatine an.
Methoden:

`+ AllergenGelatineIcon(width: double, height: double, color: Color)`
«create»
Konstruktor für das AllergenGelatineIcon

AllergenHazelnutsIcon

Typ: `class` extends `IAllergenIcon`
Paket: `view/core/icons/allergens`
Beschreibung: Dieses Klasse zeigt das Icon für Haselnüsse an.
Methoden:

`+ AllergenHazelnutsIcon(width: double, height: double, color: Color)`
«create»
Konstruktor für das AllergenHazelnutsIcon

AllergenKamutIcon

Typ: `class` extends `IAllergenIcon`
Paket: `view/core/icons/allergens`
Beschreibung: Dieses Klasse zeigt das Icon für Kamut an.
Methoden:

`+ AllergenKamutIcon(width: double, height: double, color: Color)`
«create»
Konstruktor für das AllergenKamutIcon

AllergenLoafIcon

Typ: `class` extends `IAllergenIcon`
Paket: `view/core/icons/allergens`
Beschreibung: Dieses Klasse zeigt das Icon für tierisches Lab an.
Methoden:

`+ AllergenLoafIcon(width: double, height: double, color: Color)`
«create»
Konstruktor für das AllergenLoafIcon

AllergenLupinIcon

Typ: `class` extends `IAllergenIcon`
Paket: `view/core/icons/allergens`
Beschreibung: Dieses Klasse zeigt das Icon für Lupine an.
Methoden:

`+ AllergenLupinIcon(width: double, height: double, color: Color)`
«create»
Konstruktor für das AllergenLupinIcon

AllergenMacadamiaIcon

Typ: `class` extends `IAllergenIcon`
Paket: `view/core/icons/allergens`
Beschreibung: Dieses Klasse zeigt das Icon für Macadamianüsse an.
Methoden:

`+ AllergenMacadamiaIcon(width: double, height: double, color: Color)`
«create»
Konstruktor für das AllergenMacadamiaIcon

AllergenMilkIcon

Typ: `class` extends `IAllergenIcon`
Paket: `view/core/icons/allergens`
Beschreibung: Dieses Klasse zeigt das Icon für Milch an.
Methoden:

`+ AllergenMilkIcon(width: double, height: double, color: Color)`
«create»
Konstruktor für das AllergenMilkIcon

AllergenMolluscsIcon

Typ: `class` extends `IAllergenIcon`
Paket: `view/core/icons/allergens`
Beschreibung: Dieses Klasse zeigt das Icon für Weichtiere an.
Methoden:

`+ AllergenMolluscsIcon(width: double, height: double, color: Color)`
«create»
Konstruktor für das AllergenMolluscsIcon

AllergenMustardIcon

Typ: `class` extends `IAllergenIcon`
Paket: `view/core/icons/allergens`
Beschreibung: Dieses Klasse zeigt das Icon für Senf an.
Methoden:

```
+ AllergenMustardIcon(width: double, height: double, color: Color)
  «create»
```

Konstruktor für das AllergenMustardIcon

AllergenOatsIcon

Typ: `class` extends `IAllergenIcon`
Paket: `view/core/icons/allergens`
Beschreibung: Dieses Klasse zeigt das Icon für Hafer an.
Methoden:

```
+ AllergenOatsIcon(width: double, height: double, color: Color)
  «create»
```

Konstruktor für das AllergenOatsIcon

AllergenPeanutsIcon

Typ: `class` extends `IAllergenIcon`
Paket: `view/core/icons/allergens`
Beschreibung: Dieses Klasse zeigt das Icon für Erdnüsse an.
Methoden:

```
+ AllergenPeanutsIcon(width: double, height: double, color: Color)
  «create»
```

Konstruktor für das AllergenPeanutsIcon

AllergenPecansIcon

Typ: `class` extends `IAllergenIcon`
Paket: `view/core/icons/allergens`
Beschreibung: Dieses Klasse zeigt das Icon für Pekannüsse an.
Methoden:

```
+ AllergenPecansIcon(width: double, height: double, color: Color)
  «create»
```

Konstruktor für das AllergenPecansIcon

AllergenPistachiosIcon

Typ: `class` extends `IAllergenIcon`
Paket: `view/core/icons/allergens`
Beschreibung: Dieses Klasse zeigt das Icon für Pistazien an.
Methoden:

`+ AllergenPistachiosIcon(width: double, height: double, color: Color) «create»`
Konstruktor für das AllergenPistachiosIcon

AllergenRyeIcon

Typ: `class` extends `IAllergenIcon`
Paket: `view/core/icons/allergens`
Beschreibung: Dieses Klasse zeigt das Icon für Roggen an.
Methoden:

`+ AllergenRyeIcon(width: double, height: double, color: Color) «create»`
Konstruktor für das AllergenRyeIcon

AllergenSesameIcon

Typ: `class` extends `IAllergenIcon`
Paket: `view/core/icons/allergens`
Beschreibung: Dieses Klasse zeigt das Icon für Sesam an.
Methoden:

`+ AllergenSesameIcon(width: double, height: double, color: Color) «create»`
Konstruktor für das AllergenSesameIcon

AllergenSoyaIcon

Typ: `class` extends `IAllergenIcon`
Paket: `view/core/icons/allergens`
Beschreibung: Dieses Klasse zeigt das Icon für Soja an.
Methoden:

`+ AllergenSoyaIcon(width: double, height: double, color: Color) «create»`
Konstruktor für das AllergenSoyaIcon

AllergenSpeltIcon

Typ: `class` extends `IAllergenIcon`
Paket: `view/core/icons/allergens`
Beschreibung: Dieses Klasse zeigt das Icon für Dinkel an.
Methoden:

`+ AllergenSpeltIcon(width: double, height: double, color: Color)`
«create»
Konstruktor für das AllergenSpeltIcon

AllergenSulphiteIcon

Typ: `class` extends `IAllergenIcon`
Paket: `view/core/icons/allergens`
Beschreibung: Dieses Klasse zeigt das Icon für Sulfite an.
Methoden:

`+ AllergenSulphiteIcon(width: double, height: double, color: Color)`
«create»
Konstruktor für das AllergenSulphiteIcon

AllergenWalnutsIcon

Typ: `class` extends `IAllergenIcon`
Paket: `view/core/icons/allergens`
Beschreibung: Dieses Klasse zeigt das Icon für Walnüsse an.
Methoden:

`+ AllergenWalnutsIcon(width: double, height: double, color: Color)`
«create»
Konstruktor für das AllergenWalnutsIcon

AllergenWheatIcon

Typ: `class` extends `IAllergenIcon`
Paket: `view/core/icons/allergens`
Beschreibung: Dieses Klasse zeigt das Icon für Weizen an.
Methoden:

`+ AllergenWheatIcon(width: double, height: double, color: Color)`
«create»
Konstruktor für das AllergenWheatIcon

NavigationArrowDownIcon

Typ: `class` extends `StatelessWidget`
Paket: `view/core/icons/navigation`
Beschreibung: Dieses Klasse zeigt das Icon für Pfeil nach oben an.
Methoden:

```
+ NavigationArrowDownIcon(key: Key, size: double, color: Color)
  «create»
```

Konstruktor für das NavigationArrowDownIcon.

NavigationArrowLeftIcon

Typ: `class` extends `StatelessWidget`
Paket: `view/core/icons/navigation`
Beschreibung: Dieses Klasse zeigt das Icon für Pfeil nach unten an.
Methoden:

```
+ NavigationArrowLeftIcon(key: Key, size: double, color: Color)
  «create»
```

Konstruktor für das NavigationArrowLeftIcon.

NavigationArrowRightIcon

Typ: `class` extends `StatelessWidget`
Paket: `view/core/icons/navigation`
Beschreibung: Dieses Klasse zeigt das Icon für Pfeil nach rechts an.
Methoden:

```
+ NavigationArrowRightIcon(key: Key, size: double, color: Color)
  «create»
```

Konstruktor für das NavigationArrowRightIcon.

NavigationFavoritesIcon

Typ: `class` extends `StatelessWidget`
Paket: `view/core/icons/navigation`
Beschreibung: Dieses Klasse zeigt das Icon für einen Favoriten an.
Methoden:

```
+ NavigationFavoritesIcon(key: Key, size: double, color: Color)
  «create»
```

Konstruktor für das NavigationFavoritesIcon.

NavigationFilterOutlinedIcon

Typ: `class` extends `StatelessWidget`
Paket: `view/core/icons/navigation`
Beschreibung: Dieses Klasse zeigt das Icon für einen Filter an.
Methoden:

```
+ NavigationFilterOutlinedIcon(key: Key, size: double, color: Color)
  «create»
```

Konstruktor für das NavigationFilterOutlinedIcon.

NavigationGridOutlinedIcon

Typ: `class` extends `StatelessWidget`
Paket: `view/core/icons/navigation`
Beschreibung: Dieses Klasse zeigt das Icon für die Speiseplanansicht als Grid an.
Methoden:

```
+ NavigationGridOutlinedIcon(key: Key, size: double, color: Color)
  «create»
```

Konstruktor für das NavigationGridOutlinedIcon.

NavigationListOutlinedIcon

Typ: `class` extends `StatelessWidget`
Paket: `view/core/icons/navigation`
Beschreibung: Dieses Klasse zeigt das Icon für die Speiseplanansicht als Liste an.
Methoden:

```
+ NavigationListOutlinedIcon(key: Key, size: double, color: Color)
  «create»
```

Konstruktor für das NavigationListOutlinedIcon.

NavigationMealPlanIcon

Typ: `class` extends `StatelessWidget`
Paket: `view/core/icons/navigation`
Beschreibung: Dieses Klasse zeigt das Icon für die Speiseplanansicht an.
Methoden:

```
+ NavigationMealPlanIcon(key: Key, size: double, color: Color)
  «create»
```

Konstruktor für das NavigationMealPlanIcon.

NavigationSettingsIcon

Typ: `class` extends `StatelessWidget`
Paket: `view/core/icons/navigation`
Beschreibung: Dieses Klasse zeigt das Icon für die Einstellungen an.
Methoden:

```
+ NavigationSettingsIcon(key: Key, size: double, color: Color)
  «create»
```
Konstruktor für das NavigationSettingsIcon.

NavigationCloseIcon

Typ: `class` extends `StatelessWidget`
Paket: `view/core/icons/navigation`
Beschreibung: Dieses Klasse zeigt das Icon für die Einstellungen an.
Methoden:

```
+ NavigationCloseIcon(key: Key, size: double, color: Color) «create»
```
Konstruktor für das NavigationCloseIcon.

FilterRestoreIcon

Typ: `class` extends `StatelessWidget`
Paket: `view/core/icons/allergens`
Beschreibung: Dieses Klasse zeigt das Icon für Mandeln an.
Methoden:

```
+ FilterRestoreIcon(optional key: Key, optional size: double,
  optional color: Color) «create»
```
Konstruktor für das FilterRestoreIcon

SortAscendingIcon

Typ: `class` extends `StatelessWidget`
Paket: `view/core/icons/allergens`
Beschreibung: Dieses Klasse zeigt das Icon für Mandeln an.
Methoden:

```
+ SortAscendingIcon(optional key: Key, optional size: double,
  optional color: Color) «create»
```
Konstruktor für das SortAscendingIcon

SortDescendingIcon

Typ: `class` extends `StatelessWidget`
Paket: `view/core/icons/allergens`
Beschreibung: Dieses Klasse zeigt das Icon für Mandeln an.
Methoden:

+ `SortDescendingIcon(optional key: Key, optional size: double, optional color: Color) «create»`
  Konstruktor für das SortDescendingIcon

ThumbUpOutlinedIcon

Typ: `class` extends `StatelessWidget`
Paket: `view/core/icons/navigation`
Beschreibung: Dieses Klasse zeigt das Icon für die Einstellungen an.
Methoden:

+ `ThumbUpOutlinedIcon(key: Key, size: double, color: Color) «create»`
  Konstruktor für das ThumbUpOutlinedIcon.

ImageReportIcon

Typ: `class` extends `StatelessWidget`
Paket: `view/core/icons/navigation`
Beschreibung: Dieses Klasse zeigt das Icon für die Einstellungen an.
Methoden:

+ `ImageReportIcon(key: Key, size: double, color: Color) «create»`
  Konstruktor für das ImageReportIcon.

ThumbDownFilledIcon

Typ: `class` extends `StatelessWidget`
Paket: `view/core/icons/navigation`
Beschreibung: Dieses Klasse zeigt das Icon für die Einstellungen an.
Methoden:

+ `ThumbDownFilledIcon(key: Key, size: double, color: Color) «create»`
  Konstruktor für das ThumbDownFilledIcon.

ThumbDownOutlinedIcon

Typ: `class` extends `StatelessWidget`
Paket: `view/core/icons/navigation`
Beschreibung: Dieses Klasse zeigt das Icon für die Einstellungen an.
Methoden:

+ `ThumbDownOutlinedIcon(key: Key, size: double, color: Color)` «create»
  Konstruktor für das ThumbDownOutlinedIcon.

ThumbUpFilledIcon

Typ: `class` extends `StatelessWidget`
Paket: `view/core/icons/navigation`
Beschreibung: Dieses Klasse zeigt das Icon für die Einstellungen an.
Methoden:

+ `ThumbUpFilledIcon(key: Key, size: double, color: Color)` «create»
  Konstruktor für das ThumbUpFilledIcon.

7.3 View-Detailansicht

DetailsPage

Typ: `class` extends `StatefulWidget`
Paket: `view/detail-view`
Beschreibung: Dieses Widget zeigt die Detailansicht eines Gerichts.
Methoden:

+ `MensaAppBar(key: Key, meal: Meal, optional line: Line)` «create»
  Konstruktor für das MensaAppBar Widget

MealAccordion

Typ: `class` extends `StatefulWidget`
Paket: `view/detail-view`
Beschreibung: Dieses Widget zeigt Informationen zu einem Gericht und seinen Beilagen, sowie Bezeichnung, Gerichtstyp und Preis. Dieses Widget kann ausgeklappt werden, um zusätzlich MealAccordionInfo anzuzeigen.
Methoden:

+ `MealAccordion(key: Key, isExpanded: bool, optional mainEntry: MealMainEntry, optional sideEntry: MealSideEntry, info: MealaccordionInof, opitonal onTap: Function, optional backgroundColor: Color, optinal expandedColor: Color)` «create»
  Konstruktor für das MealAccordion Widget

MealAccordionInfo

Typ: `class` extends `StatelessWidget`
Paket: `view/detail-view`
Beschreibung: Dieses Widget zeigt Allergene und Zusatzstoffe zu einem Gericht an.
Methoden:

+ `MealAccordionInfo(key: Key, allergens: List<Allergen>, additives: List<Additive>) «create»`
  Konstruktor für das MealAccordionInfo Widget

RatingsOverview

Typ: `class` extends `StatelessWidget`
Paket: `view/detail-view`
Beschreibung: Dieses Widget zeigt eine Übersicht über die Bewertungen eines Gerichts an. Hierzu zählt die durchschnittlich Abgegebende Bewertung, sowie die Gesamtzahl aller abgegebenen Bewertungen.
Methoden:

+ `RatingsOverview(key: Key, meal: Meal, optional backgroundColor: Color) «create»`
  Konstruktor für das RatingsOverview Widget

MealRatingDialog

Typ: `class` extends `StatelessWidget`
Paket: `view/detail-view`
Beschreibung: Dieses Widget zeigt einen Dialog zum Abgeben einer Bewertung für ein Gericht an.
Methoden:

+ `MealRatingDialog(key: Key, meal: Meal) «create»`
  Konstruktor für das MealRatingDialog Widget

UploadImageDialog

Typ: `class` extends `StatelessWidget`
Paket: `view/detail-view`
Beschreibung: Dieses Widget zeigt einen Dialog zum Verlinken eines Bildes.
Methoden:

+ `UploadImageDialog(key: Key, meal: Meal) «create»`
  Konstruktor für das UploadImageDialog Widget

7.4 View-Favoritenansicht

Favorites

Typ: `class` extends `StatelessWidget`
Paket: `view/mealplan`
Beschreibung: Dieses Widget zeigt die Favoritenansicht mit einer Liste der favorisierten Gerichte an.
Methoden:

+ `Favorites(key: Key) «create»`
Konstruktor für das Favorites Widget

7.5 View-Filter

FilterDialog

Typ: `class` extends `StatefulWidget`
Paket: `view/filter`
Beschreibung: Dieses Widget zeigt einen Dialog zum Wählen der Filtereinstellungen an.
Methoden:

+ `FilterDialog<T>(optinoal key: Key) «create»`
Konstruktor für die FilterDialog

MensaButtonGroup<T>

Typ: `class` extends `StatelessWidget`
Paket: `view/filter`
Beschreibung: Dieses Widget zeigt eine Gruppe von Schaltflächen zum Wählen von Optionen an.
Methoden:

+ `MensaButtonGroup<T>(key: Key, value: T, onChanged: Function(T), entries: List<MensaButtonGroupEntry<T>) «create»`
Konstruktor für die MensaButtonGroup

MensaButtonGroupEntry

Typ: `class`
Paket: `view/filter`
Beschreibung: Dieses Widget stellt eine Schaltfläche in MensaButtonGroup an.
Attribute:

+ `value: T`
Der Wert der Option

Methoden:

+ `MensaButtonGroupEntry<T>(title: String, value: T) «create»`
Konstruktor für den MensaButtonGroupEntry

+ `build(context: BuildContext, selected: bool, onChanged: void Function(T))`
Gibt das Widget zum anzeigen zurück

MensaFilterIconCheckboxGroup

Typ: `class` extends `StatelessWidget`
Paket: `view/filter`

Beschreibung: Diese Widget zeigt eine Gruppe von wählbaren Optionen an. Diese Optionen werden über ein Icon dargstellt.
Methoden:

+ `MensaFilterIconCheckboxGroup<T>(key: Key, entries: List<MensaFilterIconCheckbox<T>», selectedValue: T, onChanged: Function(List<T>)) «create»`
Konstruktor für die MensaFilterIconCheckboxGroup

MensaFilterIconCheckbox

Typ: `class` extends `StatelessWidget`
Paket: `view/filter`
Beschreibung: Dieses Widget zeigt eien Schaltfläche in MensaFilterCheckboxGroup an. Die Schaltfläche zeigt ein Icon sowie einen Text zur Option an.
Methoden:

+ `MensaFilterIconCheckbox<T>(icon: IAllergenIcon, text: String, value: T) «create»`
Konstruktor für die MensaFilterIconCheckbox

MensaSortSelect<T>

Typ: `class` extends `StatelessWidget`
Paket: `view/filter`
Beschreibung: Dieses Widget zeigt Optionen zum Wählen einer Sortierung an.
Methoden:

+ `MensaSortSelect<T>(optional key: Key, entries: List<MensaSortSelectEntry<T>», selectedEntry: T, sortDirection: SortDirection, onEntrySelected: Function(T), onSortDirectionSelected: Fuction(SortDirection)) «create»`
Konstruktor für die MensaSortSelect

MensaSortSelectEntry

Typ: `class`
Paket: `view/filter`
Beschreibung: Dieses Widget zeigt eine Sortieroption in MensaSortSelect an.
Attribute:

+ `value: T`
Der Wert der Option
+ `label: String`
Was für den Wert angezeigt werden soll.

Methoden:

+ `MensaSortSelect<T>(value: T, label: String) «create»`
Konstruktor für die MensaSortSelect

7.6 View-Bilderdialog

MealImageDialog

Typ: `class` implements `StatefulWidget`

Paket: `view/images`

Beschreibung: Dieses Widget ist ein Dialog und öffnet sich durch einen Klick auf das Bild in der DetailsPage. In dieser Ansicht kann man durch die angezeigten Bilder wechseln, Bilder bewerten und Bilder melden.

Methoden:

\+ `MealImageDialog<T>(optinoal key: Key, image: ImageData) «create»`
Konstruktor für die MealImageDialog

ImageReportDialog

Typ: `class` implements `StatefulWidget`

Paket: `view/images`

Beschreibung: Dieses Widget ist ein Dialog und öffnet sich durch das Melden eines Bildes im MealImage-Dialog. Nach der Bestätigung des ImageReportDialogs, befindet sich der Nutzer wieder im MealImageDialog. Dieser Dialog besteht aus mehreren Widgets des Cores.

Methoden:

\+ `MealImageDialog<T>(optinoal key: Key, images: List<ImageData>) «create»`
Konstruktor für die MealImageDialog

7.7 View-Speiseplanansicht

MealPlanView

Typ: `class` extends `StatefulWidget`

Paket: `view/mealplan`

Beschreibung: Dieses Widget zeigt die Speiseplanansicht an.

Methoden:

\+ `MealPlanView(key: Key) «create»`
Konstruktor für das MealPlanView Widget

MealPlanList

Typ: `class` extends `StatelessWidget`

Paket: `view/mealplan`

Beschreibung: Dieses Widget zeigt den Speiseplan als Liste an.

MealPlanGrid

Typ: `class` extends `StatelessWidget`

Paket: `view/mealplan`

Beschreibung: Dieses Widget zeigt den Speiseplan als Galerie an.

MealPlanToolbar

Typ: `class` extends `StatelessWidget`

Paket: `view/mealplan`

Beschreibung: Dieses Widget zeigt die Toolbar zum Wechseln des Anzeigeformats und des Datums an.

Methoden:

+ `MealPlanToolbar(optional key: Key, optional toolBarHeigth: double, child: Widget) «create»`
  Konstruktor für das MealPlanToolbar Widget

MealPlanDateSelect

Typ: `class` extends `StatelessWidget`

Paket: `view/mealplan`

Beschreibung: Dieses Widget zeigt einen Dialog zur Auswahl des Datums an.

Methoden:

+ `MealPlanDateSelect(optinoal key: Key, date: DateTime, onDateChange: Function(DateTime) «create»`
  Konstruktor für das MealPlanDateSelect Widget

MealPlanError

Typ: `class` extends `StatelessWidget`

Paket: `view/mealplan`

Beschreibung: Dieses Widget zeigt an, dass keine Speiseplandaten aufgrund von Verbindungsfehlern vorhanden sind.

Methoden:

+ `MealPlanError(key: Key) «create»`
  Konstruktor für das MealPlanError Widget

MealPlanClosed

Typ: `class` extends `StatelessWidget`

Paket: `view/mealplan`

Beschreibung: Dieses Widget zeigt an, dass die Mensa geschlossen hat.

Methoden:

+ `MealPlanClosed(key: Key) «create»`
  Konstruktor für das MealPlanClosed Widget

MealPlanNoData

Typ: `class` extends `StatelessWidget`

Paket: `view/mealplan`

Beschreibung: Dieses Widget zeigt an, dass die App (noch) keine Speiseplandaten der Mensa hat.

Methoden:

+ `MealPlanNoData(key: Key) «create»`
  Konstruktor für das MealPlanNoData Widget

MealPlanFilter

Typ: `class` extends `StatelessWidget`

Paket: `view/mealplan`

Beschreibung: Dieses Widget zeigt an, dass kein Gericht des Speiseplans den Filtern entspricht.

Methoden:

\+ `MealPlanFilter(key: Key) «create»`
Konstruktor für das MealPlanFilter Widget

MensaCanteenSelect

Typ: `class` extends `StatelessWidget`

Paket: `view/mealplan`

Beschreibung: Dieses Widget zeigt ein Dropdown-Menü verschiedener Mensen an.

Methoden:

\+ `MealPlanFilter(optional key: Key, availableCanteens: List<Canteen>, selectedCanteen: Canteen, onCanteenSelected: Fuction(Canteen)) «create»`
Konstruktor für das MealPlanFilter Widget

7.8 View-Einstellungsansicht

Settings

Typ: `class` implements `\markchange {StatelessWidget}`

Paket: `view/settings`

Beschreibung: Zeigt die Einstellungen an. Diese Oberfläche besteht aus (mehreren) SettingsSections und anderen Widgets.

Methoden:

\+ `Settings(key: Key) «create»`
Konstruktor für das Settings Widget

SettingsSection

Typ: `class` implements `StatelessWidget`

Paket: `view/settings`

Beschreibung: Dieses Widget bildet einen Abschnitt in der Settings-Oberfläche.

Methoden:

\+ `SettingsSection(optional key: Key, heading: String, children: List<Widget>) «create»`
Konstruktor für das SettingsSection Widget

SettingsDropdownEntry<T>

Typ: `class` implements `StatelessWidget`

Paket: `view/settings`

Beschreibung: Dieses Widget ist ein Eintrag in einem „Dropdown“ auf der Settings-Oberfläche.

Methoden:

+ `SettingsDropdownEntry(optional key: Key, onChanged: Fuction(T), value: T, items: List<MensaDropdownEntry<T>, heading: String)`
«create»
Konstruktor für das SettingsDropdownEntry Widget

SettingsToggleEntry

Typ: `class` implements `StatelessWidget`
Paket: `view/settings`
Beschreibung: Dieses Widget ist ein „Toggle-Button“ der für die Anforderungen der Settings-Oberfläche zugeschnitten ist.

7.9 Logic

Der Zugang zu den SharedPreferences wurde für die `FilterPreferences` und die `Canteen` in den IMealAccess verschoben, da diese hier benötigt werden. Außerdem wurden Getter vereinfacht, sodass sie besser aufzurufen sind.

IMealAccess

Typ: `interface` extends `ChangeNotifier`
Paket: `view-model/logic/meal`
Beschreibung: Dieses Interface erlaubt das Laden des Speiseplans oder einzelner Gerichte und Änderungen an Gerichten.
Methoden:

+ `getMealPlan(): Future<Result<List<MealPlan>, MealPlanException»`
  Eine Liste der verschiedenen Speisepläne der Linien werden zurückgegeben oder einen Fehler, falls diese nicht vorhanden sind bzw. nicht geladen werden konnten.
+ `getWholeFavorite(id: String): Future<Result<Meal, Exception»`
  Das Gericht mit der übergebenen ID wird zurückgegeben oder ein Fehler, falls dieses nicht vorhanden ist oder nicht geladen werden konnte.
+ `refreshMealplan(): Future<String>`
  Der gesamte Speiseplan wird beim Server neu angefragt und in der Datenbank und der Benutzeroberfläche aktualisiert. Falls dies nicht möglich ist, so wird der Text für eine Fehlermeldung zurückgegeben.
+ `updateMealRating(rating: int, meal: Meal): Future<String>`
  Die Bewertung deines Nutzers zu einem bestimmten Gericht wird mit dem Server synchronisiert. Falls dies nicht möglich ist, so wird der Text für eine Fehlermeldung zurückgegeben.
+ `switchToMealPlanView(): Future<void>`
  Die Methode aktualisiert die Daten bei geänderten Einstellungen oder Favoriten.
+ `changeFilterPreferences(preferences: FilterPreferences): Future<void>`
  Die übergebenen `FilterPreferences` werden gespeichert und die Widgets über die Änderung informiert.
+ `getFilterPreferences(): Future<FilterPreferences>`
  Gibt die vom Nutzer gespeicherten Filtereinstellungen zurück.
+ `resetFilterPreferences(): Future<void>`
  Die Filterkonfiguration wird auf Standart-Werte zurückgesetzt, sodass alle Gerichte angezeigt werden.
+ `getCanteen(): Future<Canteen>`
  Die zuletzt angezeigte Mensa wird zurückgegeben.
+ `getAvailableCanteens(): Future<List<Canteen»`
  Diese Methode gibt alle erreichbaren Mensen zurück.
+ `changeCanteen(canteen: Canteen): Future<void>`
  Die angezeigte Mensa wird auf die übergebene Mensa gewechselt.
+ `getDate(): DateTime`
  Der Tag, für den der Speiseplan angezeigt wird, wird gewechselt.

+ `changeDate(date: DateTime): Future<void>`
  Der Speiseplan für den übergebenen Tag wird angezeigt.
+ `activateFilter(): Future<void>`
  Die Filter werden aktiviert.
+ `deactivateFilter(): Future<void>`
  Die Filter werden deaktiviert.
+ `toggleFilter(): Future<void>`
  Die Filter werden getoggelt.

CombinedMealPlanAccess

Typ: `class` implements `IMealAccess`
Paket: `view-model/logic/meal`
Beschreibung: Weiterleitung der Anfragen von Daten zu Gerichten oder Speiseplänen an die Datenbank oder an den Server, wobei bei Fehlern eine temporäre Fehlermeldung ausgegeben wird. Die Listener des ChangeNotifiers werden über Änderungen in den Daten benachrichtigt.
Methoden:

+ `CombinedMealAccess(api: IServerAccess, database: IDatabaseAccess, preferences: ISharedPreferenceAccess): «create»`
  Konstruktor der einen kombinierten Zugriffspunkt auf den Speiseplan ermöglicht.

IFavoriteMealAccess

Typ: `interface` extends `ChangeNotifier`
Paket: `view-model/logic/favorite`
Beschreibung: Dieses Interface erlaubt das Laden und Ändern der Favoriten des Nutzers.
Methoden:

+ `addFavorite(meal: Meal): Future<void>`
  Das übergebene Gericht wird zu den Favoriten des Nutzers hinzugefügt.
+ `deleteFavorite(meal: Meal): Future<void>`
  Das übergebene Gericht wird aus den Favoriten des Nutzers entfernt.
+ `isFavorite(meal: Meal): Future<boolean>`
  Es wird `true` zurückgegeben, falls das übergebene Gericht ein Favorit ist.
+ `getFavorites(): Future<List<Meal»`
  Eine Liste aller vom Nutzer favorisierten Gerichte wird zurückgegeben.

FavoriteMealAccess

Typ: `class` implements `IFavoriteMealAccess`
Paket: `view-model/logic/favorite`
Beschreibung: Weiterleitung der Anfragen zu Favorien an die Datenbank. Die Listener des ChangeNotifiers werden über Änderungen in den Daten benachrichtigt.
Methoden:

+ `FavoriteMealAccess(IDatabaseAccess access): «create»`
  Konstruktor welcher einen Zugriffspunkt auf die in einer lokal gespeicherten Datenbank erstellt.

IPreferenceAccess

Typ: `interface` extends `ChangeNotifier`
Paket: `view-model/logic/preference`
Beschreibung: Dieses Interface erlaubt das Laden und Ändern von verschiedenen Einstellungen und Clientinformationen.
Methoden:

+ `getClientIdentifier(): String`
  Der Geräteidentifikator wird zurückgegeben.
+ `setClientIdentifier(identifier: String): Future<void>`
  Der übergebene Identifikator wird als Geräteidentifikator gespeichert.
+ `getFilterPreferences(): Future<FilterPreferences>`
  Gibt die vom Nutzer gespeicherten Filtereinstellungen zurück.
+ `setFilterPreferences(FilterPreference filter): Future<void>`
  Die übergebene Filterkonfiguration wird gespeichert.
+ `resetFilterPreferences(): Future<void>`
  Die Filterkonfiguration wird auf Standart-Werte zurückgesetzt, sodass alle Gerichte angezeigt werden.
+ `getCanteen(): Future<Canteen>`
  Die zuletzt angezeigte Mensa wird zurückgegeben.
+ `setCanteen(Canteen canteen): Future<void>`
  Die übergebene Mensa wird gespeichert.
+ `getColorScheme(): ColorScheme`
  Das gespeicherte Farbschema wird zurückgegeben.
+ `setColorScheme(ColorScheme scheme): Future<void>`
  Das übergebene Farbschema wird gespeichert.
+ `getPriceCategory(): PriceCategory`
  Die eingestellte Preisklasse wird zurückgegeben.
+ `setPriceCategory(PriceCategory category): Future<void>`
  Die übergebene Preisklasse wird gespeichert.
+ `getMealPlanFormat(): MealPlanFormat`
  Die zuletzt gewählte Darstellungsform der Speiseplanansicht wird zurückgegeben.
+ `setMealPlanFormat(MealPlanFormat format): Future<void>`
  Die übergebene Darstellungsform wird gespeichert.

PreferenceAccess

Typ: `class` implements `IPreferenceAccess`
Paket: `view-model/logic/preference`
Beschreibung: Weiterleitung der Anfragen zu Einstellungen oder Nutzerpräferenzen an den lokalen Speicher, wobei bei Fehlern auf voreingestellte Werte zurückgegriffen wird, welche dann im local Storage gespeichert werden. Die Listener des ChangeNotifiers werden über Änderungen in den Daten benachrichtigt.
Methoden:

+ `PreferenceAccess(access: ILocalStorage): «create»`
  Konstruktor welcher einen Zugriffspunkt auf die lokal gespeicherten Einstellungsdaten erstellt.

IImageAccess

ℹ Im `IImageAccess` wird überall ein String zurückgegeben, der im Entwurf nicht modelliert wurde. Das liegt daran, dass die Anzeige dieses Strings im View implementiert ist. Modelliert wurde sie im View-Model.

Typ: `interface` extends `ChangeNotifier`

Paket: `view-model/logic/image`

Beschreibung: Dieses Interface erlaubt das Verlinken, Bewerten und Melden von Bildern.

Methoden:

+	`linkImage(url:String, meal: Meal): Future<String>` Die übergebene URL wird serverseitig mit dem übergebenen Bild verknüpft. Falls dies nicht möglich ist, so wird der Text für eine Fehlermeldung zurückgegeben. Ansonsten wird der Text für eine Erfolgsmedlung zurückgegeben.
+	`upvoteImage(image: Image): Future<String>` Die Bewertung des übergebenen Bildes wird dem Server übermittelt. Falls dies nicht möglich ist, so wird der Text für eine Fehlermeldung zurückgegeben.
+	`downvoteImage(image: Image): Future<String>` Die Bewertung des übergebenen Bildes wird dem Server übermittelt. Falls dies nicht möglich ist, so wird der Text für eine Fehlermeldung zurückgegeben.
+	`deleteUpvote(image: Image): Future<String>` Die Entfernung der positiven Bewertung des übergebenen Bildes wird dem Server übermittelt. Falls dies nicht möglich ist, so wird der Text für eine Fehlermeldung zurückgegeben.
+	`deleteDownvote(image: Image): Future<String>` Die Entfernung der negativen Bewertung des übergebenen Bildes wird dem Server übermittelt. Falls dies nicht möglich ist, so wird der Text für eine Fehlermeldung zurückgegeben.
+	`reportImage(image: Image, reportReason: ReportCategory): Future<String>` Die Meldung des übergebenen Bildes wird dem Server übermittelt. Falls dies nicht möglich ist, so wird der Text für eine Fehlermeldung zurückgegeben. Ansonsten wird der Text für eine Erfolgsmedlung zurückgegeben.

ImageAccess

Typ: `class` implements `IImageAccess`

Paket: `view-model/logic/image`

Beschreibung: Weiterleitung der Anfragen zu Bildern an den Server, wobei bei Fehlern eine temporäre Fehlermeldung ausgegeben wird. Die Listener des ChangeNotifiers werden über Änderungen in den Daten benachrichtigt.

Methoden:

`+ ImageAccess(access: IServerAccess): «create»`
Konstruktor welcher einen Zugriffspunkt auf die Serverfunktionaliäten für Bilder erstellt.

7.10 Datatypes

ℹ Bei einigen Klassen wurde ein zweiter Konstruktor hinzugefügt, der eine Klasse desselben Typs und alle Attribute des eigentlichen Konstruktors optional entgegen nimmt. Dies dient dazu einfach einzelne Attribute auszutauschen. Da diese final sind, ist das nicht in demselben Objekt möglich.

Meal

Typ: `class`
Paket: `view-model/repository/data-classes/meal`
Beschreibung: Klasse, die ein Gericht modelliert.
Methoden:

`Meal(id: String, name: String, foodType: FoodType, price: Price, optional allergens: List<Allergen>, optional additives: List<Additive>, optional sides: List<Side>, optional individualRating: integer, optional numberOfRatings: integer, optional averageRating: float, lastServed: DateTime, optional`
+ `nextServed: DateTime, optional relativeFrequency: Frequency, optional images: List<Image>, numberOfOccurance: int, isFavorite: boolean) «create»`

Konstruktor, der alle Attribute initialisiert, wobei nicht alle Attribute zwangsläufig benötigt werden. Die Attribute, die nicht zwangsläufig benötigt werden, sind mit dem Wort `optional` gekennzeichnet.

`Meal.copy(meal: Meal, optional id: String, optional name: String, optional foodType: FoodType, optional price: Price, optional allergens: List<Allergen>, optional additives: List<Additive>, optional sides: List<Side>, optional individualRating: integer, optional numberOfRatings: integer, optional averageRating: float,`
+ `lastServed: DateTime, optional nextServed: DateTime, optional relativeFrequency: Frequency, optional images: List<Image>, numberOfOccurance: int, isFavorite: boolean) «create»`

Konstruktor, der alle Attribute initialisiert, wobei nicht alle Attribute zwangsläufig benötigt werden, sondern durch die des übergebenen Gerichts-Attributes ersetzt werden. Die Attribute, die nicht zwangsläufig benötigt werden, sind mit dem Wort `optional` gekennzeichnet.

+ `getId(): String`

Gibt den Identifikator des Gerichts zurück.

+ `getName(): String`

Gibt den Namen des Gerichts zurück.

+ `getFoodType(): FoodType`

Gibt die Art des Gerichts zurück.

+ `getPrice(): Price`

Gibt den Preis des Gerichts zurück.

+ `getAllergens(): List<Allergen>`
  Gibt alle Allergene des Gerichts zurück.
+ `getAdditives(): List<Additive>`
  Gibt alle Zusatzstoffe des Gerichts zurück.
+ `getSides(): List<Side>`
  Gibt alle Beilagen des Gerichts zurück.
+ `getIndividualRating(): integer`
  Gibt die Bewertung des Nutzers für das Gericht zurück.
+ `getNumberOfRatings(): integer`
  Gibt die Gesamtanzahl der Bewertungen des Gerichts zurück.
+ `getAverageRating(): float`
  Gibt die Durchschnittliche Bewertung des Gerichts zurück.
+ `getLastServed(): DateTime`
  Gibt das Datum des Tages zurück, an dem das Gericht zuletzt serviert wurde.
+ `getNextServed(): DateTime`
  Gibt das Datum des Tages zurück, an dem das Gericht das nächste Mal serviert wird.
+ `getRelativeFrequency(): Frequency`
  Gibt die Kategorie der relativen Häufigkeit zurück, mit dem das Gericht angeboten wird.
+ `getImages(): List<Image>`
  Gibt alle Bilder zurück, die zu dem Gericht gehören.
+ `getNumberOfOccurance(): int`
  Gibt zurück, wie oft das Gericht in den letzten drei Monaten angeboten wurde.
+ `isFavorite(): boolean`
  Gibt zurück, ob das Gericht ein Favorit ist.
+ `setFavorite()`
  Setzt den Wert `isFavorite` auf `true`.
+ `deleteFavorite()`
  Setzt den Wert `isFavorite` auf `false`.
+ `Map<String, dynamic> toMap()`
  Gibt eine Zuordnung von den Attributen als String zu den Werten zurück, die dafür genutzt wird, die Werte in der Datenbank zu speichern. Dabei werden alle Attribute in die Liste eingefügt, die initialisiert sind und in der Datenbank für Meal benötigt werden (siehe Datenbankschema).
+ `List<Map<String, dynamic» additiveToMap()`
  Gibt alle Zusatzstoffe in einer Map zurück, die in der Datenbank gespeichert werden kann.
+ `List<Map<String, dynamic» allergenToMap()`
  Gibt alle Allergene in einer Map zurück, die in der Datenbank gespeichert werden kann.

Side

Typ: `class`
Paket: `view-model/repository/data-classes/meal`
Beschreibung: Klasse, die eine Beilage modelliert.
Methoden:

+ `Side(id: String, name: String, foodType: FoodType, price: Price, allergens: List<Allergen>, additives: List<Additive>) «create»`
Konstruktor, der alle Attribute mit den übergebenen Werten initialisiert.

`Side.copy(side: Side, optional id: String, optional name: String, optional foodType: FoodType, optional price: Price, optional allergens: List<Allergen>, optional additives: List<Additive>)`
+ `«create»`
Konstruktor, der alle Attribute mit den übergebenen Werten, falls vorhanden, initialisiert. Nicht übergebene Werte werden aus der übergebenen Beilage übernommen. Die Attribute, die nicht zwangsläufig benötigt werden, sind mit dem Wort `optional` gekennzeichnet.

+ `getId(): String`
Gibt den Identifikator der Beilage zurück.

+ `getName(): String`
Gibt den Namen der Beilage zurück.

+ `getFoodType(): FoodType`
Gibt die Art der Beilage zurück.

+ `getPrice(): Price`
Gibt den Preis der Beilage zurück.

+ `getAllergens(): List<Allergen>`
Gibt alle Allergene der Beilage zurück.

+ `getAdditives(): List<Additive>`
Gibt alle Zusatzstoffe der Beilage zurück.

+ `Map<String, dynamic> toMap()`
Gibt eine Zuordnung von den Attributen als String zu den Werten zurück, die dafür genutzt wird, die Werte in der Datenbank zu speichern. Dabei werden alle Attribute in die Liste eingefügt, die in der Datenbank für Side benötigt werden (siehe Datenbankschema).

+ `List<Map<String, dynamic» additiveToMap()`
Gibt alle Zusatzstoffe in einer Map zurück, die in der Datenbank gespeichert werden kann.

+ `List<Map<String, dynamic» allergenToMap()`
Gibt alle Allergene in einer Map zurück, die in der Datenbank gespeichert werden kann.

Image

Typ: `class`
Paket: `view-model/repository/data-classes/meal`
Beschreibung: Klasse, die ein Bild modelliert
Methoden:

+ `Image(id: String, url: String, imageRank: float, positiveRating: integer, negativeRating: integer, optional individualRating: integer) «create»`
Konstruktor, der alle Attribute initialisiert, wobei nicht alle Attribute zwangsläufig benötigt werden. Die Attribute, die nicht zwangsläufig benötigt werden, sind mit dem Wort `optional` gekennzeichnet.

+ `getId(): String`
  Gibt den Identifikator des Bildes zurück.
+ `getUrl(): String`
  Gibt den Link zum Bild zurück.
+ `getImageRank(): float`
  Gibt den Bildrang des Bildes zurück.
+ `getIndividualRating(): integer`
  Gibt die Bewertung des Nutzers für das Bild zurück.
+ `getPositiveRating(): integer`
  Gibt die Anzahl an positiven Bewertungen des Bildes zurück.
+ `getNegativeRating(): integer`
  Gibt die Anzahl an negativen Bewertungen des Bildes zurück.
+ `upvote()`
  Das Upvote des Bildes wird gepseichert.
+ `downvote()`
  Das Downvote des Bildes wird gepseichert.
+ `deleteRating()`
  Die Bewertung des Bildes wird gelöscht.
+ `Map<String, dynamic> toMap()`
  Gibt eine Zuordnung von den Attributen als String zu den Werten zurück, die dafür genutzt wird, die Werte in der Datenbank zu speichern. Dabei werden alle Attribute in die Liste eingefügt, die in der Datenbank für Image benötigt werden (siehe Datenbankschema).

FilterPreferences

Typ: `class`
Paket: `view-model/repository/data-classes/filter`
Beschreibung: Klasse, die eine Filterkonfiguration modelliert.
Methoden:

+ `FilterPreferences(optional {category: List<FoodTypes>, allergens: List<Allergens>, price: integer, rating: float, onlyFavorites: boolean, frequencies: List<Frequency>, sortedBy: Sorting, ascending: boolean}) «create»`
  Erstellt eine Filterkonfiguration anhand der übergebenen Parameter.
+ `getCategories(): List<FoodType>`
  Gibt die Gerichtsarten zurück, nach denen gefiltert werden soll.
+ `setAllCategories(categories: List<FoodType>)`
  Setzt die Gerichtsarten fest, nach denen gefiltert werden soll.
+ `setAllCategories()`
  Setzt die Gerichtsarten fest, sodass alle Gerichte angezeigt werden.
+ `addMeatCategory(foodType: FoodType)`
  Fügt die übergebene Fleischkategorie den Gerichtsarten hinzu, nach denen gefiltert wird.

+ `removeMeatCategory(foodType: FoodType)`
Entfernt die übergebene Fleischkategorie aus den Gerichtsarten, nach denen gefiltert wird.

+ `setCategoriesVegetarian()`
Setzt die Gerichtsarten fest, sodass nur vegetarische Gerichte oder Gerichte ohne Klassifizierung angezeigt werden.

+ `setCategoriesVegan()`
Setzt die Gerichtsarten fest, sodass nur vegane Gerichte oder Gerichte ohne Klassifizierung angezeigt werden.

+ `addCategoryUnknown()`
Fügt die Klasse der unzugeordneten Gerichten den Gerichtsarten hinzu, nach denen gefiltert wird.

+ `removeCategoryUnknown()`
Entfernt die Klasse der unzugeordneten Gerichten aus den Gerichtsarten, nach denen gefiltert wird.

+ `getAllergens(): List<Allergen>`
Gibt die Allergene zurück, nach denen gefiltert werden soll.

+ `setAllergens(allergens: List<Allergen>)`
Setzt die Allergene fest, nach denen gefiltert werden soll.

+ `getPrice(): integer`
Gibt den Maximalpreis zurück, nach dem gefiltert werden soll.

+ `setPrice(price: integer)`
Setzt den Maximalpreis fest, nach dem gefiltert werden soll.

+ `getRating(): float`
Gibt die Mindestbewertung zurück, nach der gefiltert werden soll.

+ `setRating(rating: float)`
Setzt die Mindestbewertung, nach der gefiltert werden soll.

+ `getFrequency(): List<Frequency>`
Gibt an, nach welchen Häufigkeiten gefiltert werden soll.

+ `setFrequency(frequency: List<Frequency>)`
Setzt fest, nach welchen Häufigkeiten gefiltert werden soll.

+ `setNewFrequency()`
Setzt fest, dass nur neue Gerichte angezeigt werden sollen.

+ `setRareFrequency()`
Setzt fest, dass nur seltene Gerichte angezeigt werden sollen.

+ `setAllFrequency()`
Setzt fest, dass Gerichte aller Häufigkeiten angezeigt werden sollen.

+ `getOnlyFavorites(): boolean`
Gibt an, ob nach Favoriten gefiltert werden soll.

+ `setOnlyFavorites(): boolean`
Setzt, ob Favoriten angezeigt werden sollen.

+ `getSortedBy(): Sorting`
Gibt die Sortierung an, nach der die Ergebnisse sortiert werden sollen.

+ `setSortedBy(sorting: Sorting)`
Setzt die Sortierung fest, nach der die Ergebnisse sortiert werden sollen.

+ `getAscending(): boolean`
Gibt an, ob die Ergebnisse auf- oder absteigend sortiert werden sollen.

+ `setAscending(): boolean`
Setzt, ob auf- oder absteigend sortiert werden soll.

Line

Typ: `class`
Paket: `view-model/repository/data-classes/mealplan`
Beschreibung: Klasse, die eine Line in einer Mensa modelliert.
Methoden:

+ `Line(id: String, name: String, position: integer, canteen: Canteen) «create»`
Konstruktor, der alle Attribute mit den übergebenen Werten initialisiert.

+ `getId(): String`
Gibt den Identifikator der Linie zurück.

+ `getName(): String`
Gibt den Namen der Linie zurück.

+ `getCanteen(): Canteen`
Gibt die Mensa zurück, in der sich die Linie befindet.

+ `getPosition(): integer`
Gibt die Position der Linie zurück, an der die Linie auf der Webseite des Studierendenwerks angezeigt wurde.

+ `Map<String, dynamic> toMap()`
Gibt eine Zuordnung von den Attributen als String zu den Werten zurück, die dafür genutzt wird, die Werte in der Datenbank zu speichern. Dabei werden alle Attribute in die Liste eingefügt, die in der Datenbank für Line benötigt werden (siehe Datenbankschema).

MealPlan

Typ: `class`
Paket: `view-model/repository/data-classes/mealplan`
Beschreibung: Klasse, die den Speiseplan eines bestimmten Tages an einer bestimmten Linie enthält.
Methoden:

+ `MealPlan(date: DateTime, line: Line, isClosed: boolean, meals: Meals[]) «create»`
Konstruktor, der alle Attribute mit den übergebenen Werten initialisiert.

+ `MealPlan.copy(mealplan: MealPlan, optional date: DateTime, optional line: Line, optional isClosed: boolean, optional meals: Meals[]) «create»`
Konstruktor, der alle Attribute mit den übergebenen Werten, falls vorhanden, initialisiert. Falls keine Werte vorhanden sind, so werden die Werte des übergebenen Speiseplans verwendet. Die Attribute, die nicht zwangsläufig benötigt werden, sind mit dem Wort `optional` gekennzeichnet.

+ `getDate(): DateTime`
  Gibt das Datum des Speiseplans zurück.
+ `getLine(): Line`
  Gibt die Linie des Speiseplans zurück.
+ `isClosed(): boolean`
  Gibt zurück, ob die ausgewählte Mensa am ausgewählten Tag geöffnet ist
+ `getMeals(): List<Meal>`
  Gibt alle Gerichte zurück, die am ausgewählten Tag an der ausgewählten Linie serviert werden.
+ `Map<String, dynamic> toMap()`
  Gibt eine Zuordnung von den Attributen als String zu den Werten zurück, die dafür genutzt wird, die Werte in der Datenbank zu speichern. Dabei werden alle Attribute in die Liste eingefügt, die in der Datenbank für MealPlan benötigt werden (siehe Datenbankschema).

Canteen

Typ: `class`
Paket: `view_model/repository/data_classes/mealplan`
Beschreibung: Klasse die eine Mensa modelliert.
Methoden:

+ `Canteen(id: String, name: String) «create»`
  Konstruktor, der alle Attribute mit den übergebenen Werten initialisiert.
+ `getId(): String`
  Gibt den Identifikator der Mensa zurück.
+ `getName(): String`
  Gibt den Namen der Mensa zurück.
+ `Map<String, dynamic> toMap()`
  Gibt eine Zuordnung von den Attributen als String zu den Werten zurück, die dafür genutzt wird, die Werte in der Datenbank zu speichern. Dabei werden alle Attribute in die Liste eingefügt, die in der Datenbank für Line benötigt werden (siehe Datenbankschema).

Price

Typ: `class`
Paket: `view-model/repository/data-classes/meal`
Beschreibung: Klasse, die den Preis eines Gerichts oder einer Beilage enthält
Methoden:

+ `Price(student: integer, employee: integer, pupil: integer, guest: integer) «create»`
  Konstruktor, der alle Attribute mit den übergebenen Werten initialisiert.
+ `getPrice(category: PriceCategory): integer`
  Gibt den Preis in Abhängigkeit von der Preiskategorie zurück
+ `getGuest(): int`
  Gibt den Preis für Gäste zurück.

+ `getStudent(): int`
  Gibt den Preis für Studierende zurück.
+ `getPupil(): int`
  Gibt den Preis für Schüler zurück.
+ `getEmployee(): int`
  Gibt den Preis für Mitarbeitende zurück.
+ `Map<String, dynamic> toMap()`
  Gibt eine Zuordnung von den Attributen als String zu den Werten zurück, die dafür genutzt wird, die Werte in der Datenbank zu speichern. Dabei werden alle Attribute in die Liste eingefügt, die in der Datenbank für Price benötigt werden (siehe Datenbankschema).

MensaColorScheme

Typ: `enum`
Paket: `view_model/repository/data_classes/settings`
Beschreibung: Die Menge aller Farbschemen der Nutzeroberfläche.

MealDisplayFormat

Typ: `enum`
Paket: `view_model/repository/data_classes/settings`
Beschreibung: Eine Auswahl an Darstellungsmöglichkeiten der Gerichte in der Nutzeroberfläche.

ReportCategory

Typ: `enum`
Paket: `view_model/repository/data_classes/settings`
Beschreibung: Eine Menge aller Report-Kategorien, die für das Melden eines Bildes verwendet werden.

PriceCategory

Typ: `enum`
Paket: `view_model/repository/data_classes/settings`
Beschreibung: Eine Menge alle Preiseinstufungen, die als Filtermöglichkeit verwendet wird.

Frequency

Typ: `enum`
Paket: `view_model/repository/data_classes/filter`
Beschreibung: Menge aller Häufigkeitstypen.

Sorting

Typ: `enum`
Paket: `view_model/repository/data_classes/filter`
Beschreibung: Eine Auswahl aller Sortieroptionen.

Allergen

Typ: `enum`
Paket: `view_model/repository/data_classes/meal`
Beschreibung: Menge aller Allergene.

FoodType

Typ: `enum`
Paket: `view_model/repository/data_classes/meal`
Beschreibung: Menge aller Gerichtstypen.

Additive

Typ: `enum`
Paket: `view_model/repository/data_classes/meal`
Beschreibung: Menge aller Additive.

7.11 Error-Handling

Result<S, E extends Exception> (sealed)

Typ: `class`
Paket: `view-model/repository/error-handling`
Beschreibung: Durch den Result-Typen lässt sich entweder ein Success oder ein Failure zurückgeben. Was genau zurückgegeben wird, lässt sich durch einen Switch-Ausdruck feststellen.
Methoden:

`+ {abstract} Result() «create»`
Konstruktor für Result

Success<S, E extends Exception>

Typ: `class` extends `Result`
Paket: `view-model/repository/error-handling`
Beschreibung: Diese Klasse ist der Rückgabetyp, wenn kein Fehler auftritt und ein Result im Rückgabewert enthalten ist.
Attribute:

`+ value: S`
Rückgabewert bei Erfolg der Funktion

Methoden:

`+ Success(value: S) «create»`
Konstruktor, der `value` auf den übergebenen Wert setzt.

Failure<S, E extends Exception>

Typ: `class` extends `Result`
Paket: `view-model/repository/error-handling`
Beschreibung: Diese Klasse ist der Rückgabetyp, wenn ein Fehler auftritt und ein Result im Rückgabewert enthalten ist.
Attribute:

+ `exception: E`
  Exception, die bei Misserfolg auftritt.

Methoden:

+ `Failure(exception: E) «create»`
  Konstruktor, der `exception` auf die übergebene Exception setzt.

MealPlanException (sealed)

Typ: `class` extends `Exception`
Paket: `view-model/repository/error-handling`
Beschreibung: Diese Exception ist eine Generalisierung aller Exceptions, die im Zusmmenhang mit dem Laden von Speiseplänen auftreten können

ClosedCanteenException

Typ: `class` extends `MealPlanException`
Paket: `view-model/repository/error-handling`
Beschreibung: Diese Exception tritt auf, wenn die Mensa geschlossen ist.

NoDataException

Typ: `class` extends `MealPlanException`
Paket: `view-model/repository/error-handling`
Beschreibung: Diese Exception tritt auf, wenn für das gewählte Datum keine Daten vorhanden sind, da das Datum vor den Aufzeichnungen der Speisepläne oder mehr als vier Wochen in der Zukunft liegt.

FilteredMealException

Typ: `class` extends `MealPlanException`
Paket: `view-model/repository/error-handling`
Beschreibung: Diese Exception tritt auf, wenn kein Gericht des Speiseplans den ausgewählten Filtern entspricht.

NoConnectionException

Typ: `class` extends `MealPlanException`
Paket: `view-model/repository/error-handling`

Beschreibung: Diese Exception tritt auf, wenn lokal für ein Datum keine Daten gespeichert sind und keine Verbindung zum Server aufgebaut werden kann.

NoMealException

Typ: `class` extends `Exception`
Paket: `view-model/repository/error-handling`
Beschreibung: Diese Exception tritt auf, wenn ein Gericht nicht in der Datenbank vorhanden ist.

7.12 Interfaces zum Model

ℹ Es wurden Methoden für den Umgang mit Mensen hinzugefügt, die es einfacher machen, mit Mensen zu arbeiten, nachdem diese eine Klasse sind und im Entwurf als Enum modelliert waren.

ℹ Außerdem wurde festgestellt, dass man bei den SharedPreferences für Abfragen keine asynchronen Funktionen benötigt, was den Code in den Widgets vereinfacht.

IDatabaseAccess

Typ: `interface`
Paket: `view-model/repository/interface`
Beschreibung: Dieses Interface ermöglicht das Ändern von Daten in der Datenbank und das Laden von Daten aus der Datenbank.
Methoden:

+ `updateAll(mealplans: List<MealPlan>): Future<void>`
  Die gesamte Datenbank wird mit den übergebenen Speiseplänen aktualisiert.
+ `getMealPlan(date: DateTime, canteen: Canteen): Future<Result<List<MealPlan>, MealPlanException»`
  Eine Liste der Speisepläne für die verschiedenen Linien für einen Tag in einer Mensa werden zurückgegeben. Falls diese nicht vorhanden sind, wird eine entsprechende Exception zurückgegeben.
+ `getMealFavorite(id: String): Future<Result<Meal, Exception»`
  Gibt die Details des Gerichts mit dem übergebenen Identifikator zurück, falls vorhanden. Ansonsten wird die NoMealException zurückgegeben
+ `getFavoriteMealsLine(meal: Meal): Future<Line>`
  Gibt eine Line zurück, an der das Gericht angeboten wurde oder angeboten wird.
+ `getFavoriteMealsDate(meal: Meal): Future<DateTime>`
  Gibt einen Tag zurück, an dem das Gericht angeboten wurde oder angeboten wird.
+ `addFavorite(Meal meal): Future<void>`
  Das übergebene Gericht wird zu den Favoriten in der Datenbank hinzugefügt.
+ `deleteFavorite(Meal meal): Future<void>`
  Das übergebene Gericht wird aus den Favoriten in der Datenbank entfernt.
+ `getFavorites(): Future<List<Meal»`
  Eine Liste aller vom Nutzer favorisierten Gerichte aus der Datenbank wird zurückgegeben.
+ `getCanteenById(id: String): Future<Canteen>`
  Die Methode gibt die Mensa mit dem übergebenen Identifikator zurück.

ILocalStorage

Typ: `interface`

Paket: `view-model/repository/interface`

Beschreibung: Dieses Interface ermöglicht das Speichern und Laden von Daten mithilfe lokaler Speichermöglichkeiten außerhalb einer Datenbank. Ist ein Key, der abgefragt wird, nicht gespeichert, so wird ein Standartwert zurückgeben.

Methoden:

+ `getClientIdentifier(): String`
  Der Geräteidentifikator wird zurückgegeben.
+ `setClientIdentifier(identifier: String): Future<void>`
  Der übergebene Identifikator wird als Geräteidentifikator gespeichert.
+ `getFilterPreferences(): FilterPreference`
  Gibt die vom Nutzer gespeicherten Filtereinstellungen zurück.
+ `setFilterPreferences(filter: FilterPreference): Future<void>`
  Die übergebene Filterkonfiguration wird gespeichert.
+ `getCanteen(): String`
  Die zuletzt angezeigte Mensa wird zurückgegeben.
+ `setCanteen(canteen: String): Future<void>`
  Die übergebene Mensa wird gespeichert.
+ `getColorScheme(): ColorScheme`
  Das gespeicherte Farbschema wird zurückgegeben.
+ `setColorScheme(scheme: ColorScheme): Future<void>`
  Das übergebene Farbschema wird gespeichert.
+ `getPriceCategory(): PriceCategory`
  Die eingestellte Preisklasse wird zurückgegeben.
+ `setPriceCategory(category: PriceCategory): Future<void>`
  Die übergebene Preisklasse wird gespeichert.
+ `getMealPlanFormat(): MealPlanFormat`
  Die zuletzt Darstellungsform der Speiseplanansicht wird zurückgegeben.
+ `setMealPlanFormat(format: MealPlanFormat): Future<void>`
  Die übergebene Darstellungsform wird gespeichert.
+ `getLanguage(): Future<String>`
  Die gespeicherte Sprache wird übergeben.
+ `setLanguage(language: String): Future<void>`
  Die übergebene Sprache wird gespeichert.

IServerAccess

Typ: `interface`

Paket: `view-model/repository/interface`

Beschreibung: Dieses Interface ist eine interne Schnittstelle zur Kommunikation mit dem Server zum Datenaustausch.

Methoden:

`updateAll(): Future<Result<List<MealPlan>, MealPlanException»`
+ Die Speisepläne aller Mensen werden aktualisiert und in der Datenbank gespeichert. Falls dies nicht möglich ist, so wird `false` zurückgegeben.

`updateCanteen(canteen: Canteen, date: DateTime): Future<Result<List<MealPlan>, MealPlanException»`
+ Eine Liste der verschiedenen Speisepläne der Linien wird zurückgegeben oder einen Fehler, falls diese nicht vorhanden sind bzw. nicht geladen werden konnten.

`getMeal(meal: Meal, line: Line, date: DateTime): Future<Result<Meal, Exception»`
+ Das Gericht mit dem übergebenen ID wird zurückgegeben oder ein Fehler, falls dieses nicht vorhanden ist oder nicht geladen werden konnte.

+ `updateMealRating(rating: int, meal: Meal): Future<boolean>`
Die Bewertung deines Nutzers zu einem bestimmten Gericht wird mit dem Server synchronisiert. Falls dies nicht möglich ist, so wird `false` zurückgegeben.

+ `linkImage(url:String, meal: Meal): Future<boolean>`
Die übergebene URL wird serverseitig mit dem übergebenen Bild verknüpft. Falls dies nicht möglich ist, so wird `false` zurückgegeben.

+ `upvoteImage(image: Image): Future<boolean>`
Die Bewertung des übergebenen Bildes wird dem Server übermittelt. Falls dies nicht möglich ist, so wird `false` zurückgegeben.

+ `downvoteImage(image: Image): Future<boolean>`
Die Bewertung des übergebenen Bildes wird dem Server übermittelt. Falls dies nicht möglich ist, so wird `false` zurückgegeben.

+ `deleteUpvote(image: Image): Future<boolean>`
Die Entfernung der positiven Bewertung des übergebenen Bildes wird dem Server übermittelt. Falls dies nicht möglich ist, so wird `false` zurückgegeben.

+ `deleteDownvote(image: Image): Future<boolean>`
Die Entfernung der negativen Bewertung des übergebenen Bildes wird dem Server übermittelt. Falls dies nicht möglich ist, so wird `false` zurückgegeben.

+ `reportImage(image: Image, reportReason: ReportCategory): Future<boolean>`
Die Meldung des übergebenen Bildes wird dem Server übermittelt. Falls dies nicht möglich ist, so wird `false` zurückgegeben.

`getDefaultCanteen(): Future<Canteen>`
+ Diese Methode fragt die Standardmensa vom Server an.

`getCanteens(): Future<List<Canteen»`
+ Diese Methode fordert alle Mensen vom Server an.

7.13 Model

SharedPreferenceAccess

Typ: `class` implements `ILocalStorage`
Paket: `model/local-storage`
Beschreibung: Diese Klasse ist für den Zugriff auf Shared Preferences des Clients verantwortlich.
Methoden:

+ `SharedPreferenceAccess(pref: SharedPreferences) «create»`
  Dieser Konstruktor gibt eine Instanz der Klasse zurück.

SQLiteDatabaseAccess

Typ: `class` implements `IDatabaseAccess`
Paket: `model/database`
Beschreibung: Diese Klasse ist für den Zugriff auf die SQLite-Datenbank des Clients verantwortlich.
Methoden:

+ `factory SQLiteDatabaseAccess() «create»`
  Dieser Konstruktor gibt eine Instanz der Klasse zurück.

ℹ Für die SQLite Datenbank werden zusätzlich zu der oben aufgeführten Methode weitere Methoden benötigt. Dazu gehört Klassen, die genau die Struktur der Datenbank repräsentiert, sodass die Ergebnisse einer Abfrage dort direkt gespeichert werden können. Dort sind auch die Strings zum Erstellen der einzelnen Relationen und die Attribute der Relationen gespeichert. Außerdem gibt es eine weitere Klasse, die für die Transformation von den Datenbanknahen Typen zu den im `view-model` gespeicherten Typen zuständig ist.

GraphQlServerAccess

Typ: `class` implements `IServerAccess`
Paket: `model/api-server`
Beschreibung: Diese Klasse ist für die Kommunikation mit dem Server durch GraphQL verantwortlich.
Methoden:

+ `factory GraphQlServerAccess(address: String, apiKey: String, clientId: String) «create»`
  Dieser Konstruktor gibt eine Instanz der Klasse zurück.

ℹ Für GraphQl wurden zusätzlich zu den hier und im Entwurf angegebenen Klassen noch weitere Klassen für Mutations, Anfragen und das allgemeine Schema benötigt, die aufgrund ihrer Menge hier nicht aufgeführt werden.
Die Mutations haben Methoden, die das Konvertieren von und zu Json-Daten übernehmen, sowie Konstruktoren und einige andere Methoden.
Die Querys enthalten Klassen, die unter anderem die Struktur von Mensen, Speiseplänen, Gerichten, usw. des GraphQl Schemas enthalten.
Das Schema enthält eine Zuordnung der Enum-Werte des GraphQl-Schemas zu den Enum-Werten des Dart-Schemas.

8 Änderungen zum Entwurf: Backend

8.1 Paket `startup`

Server

Typ: `class`
Paket: `startup`
Beschreibung: Klasse, die die gesammelte Serverfunktion nach außen anbietet.
Methoden:

`+ {static} run(): Void | ServerError`
Führt den Server und alles, was dazugehört aus. Dabei werden zunächst Konfigurationen eingelesen und dann die Komponentenstruktur aufgebaut.

LogInfo

Typ: `class`
Paket: `startup`
Beschreibung: Klasse, die die gesammelte Serverfunktion nach außen anbietet.
Methoden:

`+ {static} run(): Void | ServerError`
Führt den Server und alles, was dazugehört aus. Dabei werden zunächst Konfigurationen eingelesen und dann die Komponentenstruktur aufgebaut.

Logger

Typ: `class`
Paket: `startup`
Beschreibung: Klasse zum Initialisieren des der Protokollierungsfunktion.
Methoden:

`+ {static} init(info: LogInfo)`
Initialisiert die Protokollierungsfunktionen. Protokollierungen passieren in jeder Komponente über einen globalen Kontext, sodass diese in den weiteren Klassenbeschreibungen nicht sichtbar sind.

ConfigReader

ℹ Weitere Konfigurationen wurden hinzugefügt um weniger fest-codierte Werte im Programm zu haben und somit mehr Flexibilität ohne erneutes Kompilieren zu erreichen.

Typ: `class`
Paket: `startup/config`
Beschreibung: Klasse zum Einlesen von Konfigurationen aus Umgebungsvariable.
Methoden:

+ `read_database_info(): DatabaseInfo | ServerError`
  Liest die Konfiguration für den Zugang zur Datenbank aus den Umgebungsvariablen und gibt diese zurück. Falls notwendige Konfigurationen nicht vorhanden sind, wird ein Fehler zurückgegeben.
+ `read_mail_info(): MailInfo | ServerError`
  Liest die Konfiguration für den Zugang zum Mailserver aus den Umgebungsvariablen und gibt diese zurück. Falls notwendige Konfigurationen nicht vorhanden sind, wird ein Fehler zurückgegeben.
+ `read_schedule_info(): ScheduleInfo | ServerError`
  Liest die Konfiguration für den Scheduler aus den Umgebungsvariablen und gibt diese zurück. Falls notwendige Konfigurationen nicht vorhanden sind, wird ein Fehler zurückgegeben.
+ `read_flickr_info(): FlickrInfo | ServerError`
  Liest die Konfiguration für den Zugang zur Flickr-API aus den Umgebungsvariablen und gibt diese zurück. Falls notwendige Konfigurationen nicht vorhanden sind, wird ein Fehler zurückgegeben.
+ `should_migrate(): Boolean`
  Prüft anhand von Programm-Argumenten ob Datenbank-Migration zur Erstellung der Relationen angewandt werden sollen.
+ `should_print_help(): Boolean`
  Prüft anhand von Programm-Argumenten ob Hilfetext angezeigt werden soll.
+ `read_log_info(): LogInfo | ServerError`
  Liest die Konfiguration für die Protokollierung aus den Umgebungsvariablen und gibt diese zurück. Falls notwendige Konfigurationen nicht vorhanden sind, wird ein Fehler zurückgegeben.
+ `read_swka_info(): SwKaInfo | ServerError`
  Liest die Konfiguration für den Zugriff auf die Mensa-Webseite des Studierendenwerks aus den Umgebungsvariablen und gibt diese zurück. Falls notwendige Konfigurationen nicht vorhanden sind, wird ein Fehler zurückgegeben.

ServerError

ℹ Weitere Fehlervarianten haben sich bei der tatsächlichen Implementierung und der Wahl der verwendeten Libraries ergeben.

Typ: `enum`
Paket: `startup`
Beschreibung: Enumerationstyp, der die verschiedenen Möglichkeiten darstellt, durch die es zu einem Fehler beim Ausführend des Servers kommen kann.
Varianten:

+ `MissingEnvVar`
  Eine notwendige Umgebungsvariable wurde nicht gesetzt.
+ `Mailerror`
  Fehler beim erstellen der Mail-Komponente
+ `CommandError`
  Fehler beim erstellen der Command-Komponente.
+ `ParseError`
  Fehler beim Erstellen der Mensa-Parse-Komponente.

+ `IoError`
  Ein-Ausgabe-Fehler
+ `ParseIntError`
  Fehler beim Umwandeln von Ganzzahlen.
+ `DataError`
  Fehler beim erstellen der Datenbank-Komponente.

8.2 Paket `util`

Allergen

Typ: `enum`
Paket: `util`
Beschreibung: Dieser Enumerationstyp stellt alle möglichen Allergene dar.

Additive

Typ: `enum`
Paket: `util`
Beschreibung: Dieser Enumerationstyp stellt alle möglichen Zusatzstoffe dar.

MealType

Typ: `enum`
Paket: `util`
Beschreibung: Dieser Enumerationstyp stellt alle möglichen Gerichtstypen dar.

ReportReason

Typ: `enum`
Paket: `util`
Beschreibung: Dieser Enumerationstyp stellt alle möglichen Meldegründe dar.

Price

Typ: `struct`
Paket: `util`
Beschreibung: Dieses Struct beinhaltet alle Preisklassen von Gerichten. Preise sind in Euro angegeben.
Attribute:

+ `price_student: Integer`
  Preis für Studenten.
+ `price_employee: Integer`
  Preis für Mitarbeiter.
+ `price_guest: Integer`
  Preis für Gäste.
+ `price_pupil: Integer`
  Preis für Schüler.

8.3 Komponente `command`

Command

Typ: `interface`
Paket: `interface/api_command`
Beschreibung: Schnittstelle für den Zugriff auf Befehle, die von einer API ausgelöst werden können.
Methoden:

+ `report_image(image_id: Uuid, reason: ReportReason, auth_info: AuthInfo): Void | CommandError`
  Befehl zum Melden eines Bildes. Dabei wird auch geprüft, ob das Bild schon automatisch versteckt werden soll.
+ `add_image_upvote(image_id: Uuid, auth_info: AuthInfo): Void | CommandError`
  Befehl zum Aufwerten eines Bildes. Gleichzeitige Abwertungen desselben Nutzers werden entfernt.
+ `add_image_downvote(image_id: Uuid, auth_info: AuthInfo): Void | CommandError`
  Befehl zu Abwerten eines Bildes. Gleichzeitige Aufwertungen desselben Nutzers werden entfernt.
+ `remove_image_upvote(image_id: Uuid, auth_info: AuthInfo): Void | CommandError`
  Befehl zum Entfernen einer Aufwertung eines Bildes, falls vorhanden.
+ `remove_image_downvote(image_id: Uuid, auth_info: AuthInfo): Void | CommandError`
  Befehl zum Entfernen einer Abwertung eines Bildes, falls vorhanden.
+ `add_image(meal_id: Uuid, image_url: String, auth_info: AuthInfo): Void | CommandError`
  Befehl zum Verlinken eines Bildes zu einem Gericht.
+ `set_meal_rating(meal_id: Uuid, rating: Integer, auth_info: AuthInfo): Void | CommandError`
  Befehl zum Bewerten eines Gerichts.

AuthInfo

Typ: `struct`
Paket: `interface/api_command`
Beschreibung: Struktur mit allen nötigen Informationen zur Authentifizierung eines Clients.
Attribute:

+ `client_id: Uuid`
  Identifikator des Clients.
+ `api_ident: String`
  Anfang des API-Schlüssels, um diesen zu identifizieren.
+ `hash: String`
  Hash aller Attribute des Befehls einschließlich des Befehlsnamen.

CommandError

Typ: `enum`
Paket: `interface/api_command`
Beschreibung: Enumerationstyp für mögliche Fehlschläge von Befehlen.
Varianten:

+ `BadAuth`
  Authentifizierungsinformationen nicht korrekt.
+ `NoAuth`
  Authentifizierungsinformationen fehlen
+ `InternalError`
  Sonstiger Fehler bei z.B. Zugriff auf Ressourcen.
+ `DataError`
  Es ist ein Fehler in der Datenschicht aufgetreten
+ `ImageHosterError`
  Es ist ein Fehler beim Bildhoster aufgetreten

CommandHandler

ℹ Im Commandhandler hat sich geändert, dass dieser im Konstruktor eine Funktion aufruft, die einen Fehler zurückgeben kann

Typ: `class` implements `Command`
Paket: `layer/logic/api_command`
Beschreibung: Eine Klasse, die das Ausführen von Befehlen erlaubt.
Methoden:

+ `{static} new(command_data: CommandDataAccess, admin_notification: AdminNotification, image_hoster: ImageHoster) «create»: CommandHandler | DataError`
  Erzeugt ein neues Objekt mit den übergebenen Zugängen zu Daten, Administratorbenachrichtigungen und Bildhoster. Wirft einen Fehler, wenn die API-Schlüssel nicht von `command_data` geholt werden konnten.

Authenticator

ℹ Die Methoden zum Authentifizieren von Befehlen wurden zusammengelegt, um weniger Redundanz und Duplikation zu beinhalten. Stattdessen wurden die teils unterscheidlichen Argumente in den Enummerationstyp verpackt.

Typ: `class`
Paket: `layer/logic/api_command/auth`
Beschreibung: Eine Klasse, die für das Authentifizieren von Befehlen zuständig ist.
Methoden:

+ `{static} new(api_keys: String[*]) «create»`
  Erzeugt ein neues Objekt, welches Anfragen authentifiziert und dabei nur die übergebenen API-Schlüssel als gültig erachtet.

+ `authn_image_command(auth_info: AuthInfo, image_id: Uuid, image_command_type: ImageCommandType): Void | CommandError`
Authentifiziert einen Bilderbefehl, indem überprüft wird, ob der Hash zum gegebenen API-Schlüssel korrekt ist.

+ `authn_meal_rating_command(auth_info: AuthInfo, meal_id: Uuid, rating: Integer): Void | CommandError`
Authentifiziert den Befehl zum Bewerten von Gerichten, indem überprüft wird, ob der Hash zum gegebenen API-Schlüssel korrekt ist.

+ `authn_add_image_command(auth_info: AuthInfo, meal_id: Uuid, url: String): Void | CommandError`
Authentifiziert den Befehl zum Hinzufügen eines Bildes, indem überprüft wird, ob der Hash zum gegebenen API-Schlüssel korrekt ist.

+ `authn_command(auth_info: AuthInfo, command_type: CommandType`
Authentifiziert Befehle indem überprüft wird, ob der Hash zum gegebenen API-Schlüssel korrekt ist. Die parameter der Befehle werden im `command_type` überreicht.

CommandType

Typ: `enum`
Paket: `layer/logic/api_command`
Beschreibung: Enumerationstyp für die verschiedenen Arten von Bildbefehlen.
Varianten:

+ `ReportImage`
Befehl zum Melden eines Bildes.
+ `AddUpvote`
Befehl zum Hinzufügen einer Aufwertung.
+ `AddDownvote`
Befehl zum Hinzufügen einer Abwertung.
+ `RemoveUpvote`
Befehl zum Entfernen einer Aufwertung.
+ `RemoveDownvote`
Befehl zum Entfernen einer Abwertung.

+ `SetRating`
Befehl zum Bewerten eines Gerichtes.

+ `AddImage`
Befehl zum hinzufügen eines Bildes zu einem Gericht.

8.4 Komponente `graphql`

GraphQLServer

Typ: `class`
Paket: `layer/trigger/graphql`
Beschreibung: Eine Klasse, die den Webserver für GraphQL-Anfragen steuert.

Methoden:

+ `{static} new(server_info: GraphQLServerInfo, data_access: RequestDataAccess, command: Command)`
  Erzeugt ein neues Objekt mit den übergebenen Zugängen zum Datenspeicher und der Logik für Befehle
+ `start()`
  Startet den GraphQL-Server. Dieser läuft im Hintergrund, bis er mit `shutdown()` gestoppt wird.
+ `shutdown()`
  Beendet den GraphQL-Server.

GraphQLServerInfo

Typ: `struct`
Paket: `layer/trigger/graphql`
Beschreibung: Struktur, welche Informationen enthält, welche zum starten des GraphQL-Servers nötig sind.
Attribute:

+ `port: Integer`
  Portnummer auf welcher der Server auf anfragen hört.

QueryRoot

Typ: `class`
Paket: `layer/trigger/graphql`
Beschreibung: Eine Klasse, die die GraphQL-Root-Query implementiert.
Methoden:

+ `get_canteens(): Canteen[*]`
  Behandlungsmethode für die `getCanteens` GraphQL-Anfrage.
+ `get_canteen(canteen_id: Uuid): Canteen[0..1]`
  Behandlungsmethode für die `getCanteen` GraphQL-Anfrage.
+ `get_meal(meal_id: Uuid, line_id: Uuid, date: Date): Meal[0..1]`
  Behandlungsmethode für die `getMeal` GraphQL-Anfrage.
+ `api_version(): String`
  Diese Methode dient zur Abfrage der aktuellen API-Version. Sie kann auch zum überprüfen der Erreichbarkeit des Servers verwendet werden.
+ `get_my_auth(): AuthInfo[0..1]`
  Diese Methode dient zur Abfrage der aktuellen Authentifizierungsinformationen (AuthInfo).

MutationRoot

Typ: `class`
Paket: `layer/trigger/graphql`
Beschreibung: Eine Klasse, die die GraphQL-Root-Mutation implementiert.
Methoden:

+ `add_image(meal_id: Uuid, image_url: String): Boolean`
  Behandlungsmethode für die `addImage` GraphQL-Anfrage.
+ `set_rating(meal_id: Uuid, rating: Integer): Boolean`
  Behandlungsmethode für die `setRating` GraphQL-Anfrage.
+ `add_upvote(image_id: Uuid): Boolean`
  Behandlungsmethode für die `addUpvote` GraphQL-Anfrage.
+ `remove_upvote(image_id: Uuid): Boolean`
  Behandlungsmethode für die `removeUpvote` GraphQL-Anfrage.
+ `add_downvote(image_id: Uuid): Boolean`
  Behandlungsmethode für die `addDownvote` GraphQL-Anfrage.
+ `remove_downvote(image_id: Uuid): Boolean`
  Behandlungsmethode für die `removeDownvote` GraphQL-Anfrage.
+ `report_image(image_id: Uuid, reason: ReportReason): Boolean`
  Behandlungsmethode für die `reportImage` GraphQL-Anfrage.

Canteen

Typ: `class`
Paket: `layer/trigger/graphql/types`
Beschreibung: Eine Klasse, die eine Mensa in GraphQL repräsentiert.
Attribute:

+ `id: Uuid`
  Attribut, auf welches über die GraphQL-API mit `id` zugegriffen werden kann.
+ `name: String`
  Attribut, auf welches über die GraphQL-API mit `name` zugegriffen werden kann.

Methoden:

+ `lines(): Line[*]`
  Bereitstellungsmethode für das `lines` GraphQL-Attribut.

Line

Typ: `class`
Paket: `layer/trigger/graphql/types`
Beschreibung: Eine Klasse, die eine Linie in GraphQL repräsentiert.
Attribute:

+ `id: Uuid`
  Attribut, auf welches über die GraphQL-API mit `id` zugegriffen werden kann.
+ `name: String`
  Attribut, auf welches über die GraphQL-API mit `name` zugegriffen werden kann.
+ `canteen_id: Uuid`
  Die Id der Mensa, an der die Linie sich befindet

Methoden:

+ `canteen(): Canteen`
  Übergibt die Kantine, in der diese Linie aufgeführt wird.
+ `meals(date: Date): Meal[*][0..1]`
  Behandlungsmethode für die `meals()` GraphQL-Anfrage.

Meal

Typ: `class`
Paket: `layer/trigger/graphql/types`
Beschreibung: Eine Klasse, die ein Hauptgericht in GraphQL repräsentiert.
Attribute:

+ `id: Uuid`
  Attribut, auf welches über die GraphQL-API mit `id` zugegriffen werden kann.
+ `name: String`
  Attribut, auf welches über die GraphQL-API mit `name` zugegriffen werden kann.
+ `ratings: Ratings`
  Attribut, auf welches über die GraphQL-API mit `ratings` zugegriffen werden kann.
+ `price: Price`
  Attribut, auf welches über die GraphQL-API mit `price` zugegriffen werden kann.
+ `meal_type: MealType`
  Attribut, auf welches über die GraphQL-API mit `mealType` zugegriffen werden kann.
+ `statistics: MealStatistics`
  Attribut, auf welches über die GraphQL-API mit `statistics` zugegriffen werden kann.
+ `date: Date`
  Das Datum an dem das Gericht angeboten wird
+ `line_id: Uuid`
  Die Linie, an dem das Gericht angeboten wird

Methoden:

+ `statistics(): MealStatistics`
  Bereitstellungsmethode für das `statistics` GraphQL-Attribut.
+ `allergens(): Allergen[*]`
  Bereitstellungsmethode für das `allergens` GraphQL-Attribut.
+ `additives(): Additive[*]`
  Bereitstellungsmethode für das `additives` GraphQL-Attribut.
+ `ratings(): Ratings`
  Bereitstellungsmethode für das `ratings` GraphQL-Attribut.
+ `images(): Image[*]`
  Bereitstellungsmethode für das `images` GraphQL-Attribut.
+ `sides(): Side[*]`
  Bereitstellungsmethode für das `sides` GraphQL-Attribut.
+ `line(): Line`
  Übergibt die Linie, in der dieses Gericht aufgeführt wird.

Side

Typ: `class`
Paket: `layer/trigger/graphql/types`
Beschreibung: Eine Klasse, die eine Beilage zu einem Gericht in GraphQL repräsentiert.
Attribute:

+ `id: Uuid`
  Attribut, auf welches über die GraphQL-API mit `id` zugegriffen werden kann.
+ `name: String`
  Attribut, auf welches über die GraphQL-API mit `name` zugegriffen werden kann.
+ `price: Price`
  Attribut, auf welches über die GraphQL-API mit `price` zugegriffen werden kann.
+ `meal_type: MealType`
  Attribut, auf welches über die GraphQL-API mit `mealType` zugegriffen werden kann.

Methoden:

+ `allergens(): Allergen[*]`
  Bereitstellungsmethode für das `allergens` GraphQL-Attribut.
+ `additives(): Additive[*]`
  Bereitstellungsmethode für das `additives` GraphQL-Attribut.

Image

Typ: `class`
Paket: `layer/trigger/graphql/types`
Beschreibung: Eine Klasse, die eine Bild in GraphQL repräsentiert.
Attribute:

+ `id: Uuid`
  Attribut, auf welches über die GraphQL-API mit `id` zugegriffen werden kann.
+ `url: String`
  Attribut, auf welches über die GraphQL-API mit `url` zugegriffen werden kann.
+ `rank: Real`
  Attribut, auf welches über die GraphQL-API mit `rank` zugegriffen werden kann.
+ `upvotes: Integer`
  Attribut, auf welches über die GraphQL-API mit `upvotes` zugegriffen werden kann.
+ `downvotes: Integer`
  Attribut, auf welches über die GraphQL-API mit `downvotes` zugegriffen werden kann.

Methoden:

+ `personal_upvote(): Boolean`
  Bereitstellungsmethode für das `personalUpvote` GraphQL-Attribut.
+ `personal_downvote(): Boolean`
  Bereitstellungsmethode für das `personalDownvote` GraphQL-Attribut.

Ratings

Typ: class
Paket: layer/trigger/graphql/types
Beschreibung: Eine Klasse, die eine die Übersicht zu Bewertungen eines Gerichts in GraphQL repräsentiert.
Attribute:

+ average_rating: Real
  Attribut, auf welches über die GraphQL-API mit averageRating zugegriffen werden kann.
+ ratings_count: Integer
  Attribut, auf welches über die GraphQL-API mit ratingsCount zugegriffen werden kann.
+ meal_id: Uuid
  Die Id von dem Meal, welches bewertet wurde

Methoden:

+ personal_rating(): Integer[0..1]
  Bereitstellungsmethode für das personalRating GraphQL-Attribut.

Price

Typ: class
Paket: layer/trigger/graphql/types
Beschreibung: Eine Klasse, die die Preise für ein Gericht in GraphQL repräsentiert.
Attribute:

+ student: Integer
  Attribut, auf welches über die GraphQL-API mit student zugegriffen werden kann.
+ employee: Integer
  Attribut, auf welches über die GraphQL-API mit employee zugegriffen werden kann.
+ guest: Integer
  Attribut, auf welches über die GraphQL-API mit guest zugegriffen werden kann.
+ pupil: Integer
  Attribut, auf welches über die GraphQL-API mit pupil zugegriffen werden kann.

MealStatistics

Typ: class
Paket: layer/trigger/graphql/types
Beschreibung: Eine Klasse, die Statistiken zu einem Gericht in GraphQL repräsentiert.
Attribute:

+ last_served: Date[0..1]
  Attribut, auf welches über die GraphQL-API mit lastServed zugegriffen werden kann.
+ next_served: Date[0..1]
  Attribut, auf welches über die GraphQL-API mit nextServed zugegriffen werden kann.
+ frequency: Real
  Attribut, auf welches über die GraphQL-API mit frequency zugegriffen werden kann.

`new: Boolean`

\+ Attribut, auf welches über die GraphQL-API mit `new` zugegriffen werden kann. Dieses markiert, ob ein Gericht neu ist und noch nie zuvor angeboten wurde.

AuthInfo

Typ: `class`

Paket: `layer/trigger/graphql/types`

Beschreibung: Eine Klasse, welche die angegebenen Authentifizierungsinformationen repräsentiert. Diese existiert hauptsächlich um zu überprüfen, ob die Authentifizeriungsinformationen korrekt übermittelt wurden.

Attribute:

\+ `client_id: Uuid`
Der eigene angegebene Client-Identifikator.

\+ `api_ident: String`
Identifikator des eigens angegebenen API-Schlüssels. Dabei sollen nur die ersten 10 Zeichen angegeben werden.

\+ `hash: String`
Der eigens angegebne Hash der Anfrage.

Allergen

Typ: `enum`

Paket: `layer/trigger/graphql`

Beschreibung: Enumerationstyp, der Allergene in GraphQL repräsentiert. Dieser Enumerationstyp muss aus Gründen der Unabhängigkeit der einzelnen Schichten und der Funktionsweise der GraphQL-API hier erneut definiert werden, obwohl ein gleicher Typ schon im Paket `util` existiert.

Additive

Typ: `enum`

Paket: `layer/trigger/graphql`

Beschreibung: Enumerationstyp, der Zusatzstoffe in GraphQL repräsentiert.Dieser Enumerationstyp muss aus Gründen der Unabhängigkeit der einzelnen Schichten und der Funktionsweise der GraphQL-API hier erneut definiert werden, obwohl ein gleicher Typ schon im Paket `util` existiert.

MealType

Typ: `enum`

Paket: `layer/trigger/graphql`

Beschreibung: Enumerationstyp, der den Typ eines Gerichts repräsentiert. Dieser Enumerationstyp muss aus Gründen der Unabhängigkeit der einzelnen Schichten und der Funktionsweise der GraphQL-API hier erneut definiert werden, obwohl ein gleicher Typ schon im Paket `util` existiert.

ReportReason

Typ: `enum`
Paket: `layer/trigger/graphql`
Beschreibung: Enumerationstyp, der den Grund für einen Meldeantrag eines Bildes repräsentiert. Dieser Enumerationstyp muss aus Gründen der Unabhängigkeit der einzelnen Schichten und der Funktionsweise der GraphQL-API hier erneut definiert werden, obwohl ein gleicher Typ schon im Paket `util` existiert.

8.5 Komponente `scheduler`

Scheduler

Typ: `class`
Paket: `layer/trigger/scheduling`
Beschreibung: Klasse zum Planen von regelmäßigen Ereignissen.
Methoden:

+ `new(info: ScheduleInfo, image_scheduling: ImageReviewScheduling, parse_scheduling: MensaParseScheduling) «create»`
  Erzeugt einen neuen Scheduler mit dem in `info` angegebenen Zeitplan und den in `scheduling` angegebenen Aktionen.
+ `start()`
  Startet den Scheduler. Dieser läuft im Hintergrund, bis er mit `shutdown()` gestoppt wird.
+ `shutdown()`
  Stoppt den Scheduler.

8.6 Komponente `image-review`

ImageReviewScheduling

Typ: `interface`
Paket: `interface/mensa_management`
Beschreibung: Ein Interface, dass das Ausführen von Bildüberprüfungen erlaubt.
Methoden:

+ `start_image_review()`
  Startet die Bildüberprüfung.

ImageReviewer

Typ: `class` implements `ImageReviewScheduling`
Paket: `layer/logic/image_review`
Beschreibung: Eine Klasse, die für das Überprüfen der Bilder steuert.
Methoden:

+ `{static} new(data_access: ImageReviewDataAccess, image_hoster: ImageHoster) «create»`
  Erzeugt ein neues Objekt mit den übergebenen Zugängen zu Datenspeicher und Bildhoster.

8.7 Komponente `mail`

ℹ In der Mail-Komponente hat sich nur geändert, dass MailError hinzugefügt wurde und MailInfo in die Komponente verschoben wurde. Außerdem fehlte die Id des Bildes im Report

AdminNotification

Typ: `interface`
Paket: `interface/admin_notification`
Beschreibung: Interface für die Benachrichtigung von Administratoren.
Methoden:

+ `notify_admin_image_report(info: ImageReportInfo)`
  Benachrichtigt den Administrator über ein neu gemeldetes Bild und die vom automatischen System vorgenommene Reaktion.

ImageReportInfo

Typ: `struct`
Paket: `interface/admin_notification`
Beschreibung: Struktur mit Informationen über einen Meldeantrag.
Attribute:

+ `reason: ReportReason`
  Grund für Meldung als ReportReason.
+ `image_got_hidden: Boolean`
  Ob das Bild schon vom automatischen System ausgeblendet wurde.
+ `image_id: Uuid`
  Eindeutiger Identifikator des gemeldeten Bildes.
+ `image_link: String`
  Link zum Bild.
+ `report_count: Integer`
  Anzahl der Meldeanträge zum Bild, inklusive dem Aktuellen.
+ `positive_raiting_count: Integer`
  Anzahl der positiven Bewertungen des gemeldeten Bildes.
+ `negative_raiting_count: Integer`
  Anzahl der negativen Bewertungen des gemeldeten Bildes.
+ `get_image_rank: Real`
  Bildrang des gemeldeten Bildes.

MailError

Typ: `enum`
Paket: `layer/data/mail/mail_sender`
Beschreibung: Enumerationstyp für mögliche Fehlschläge von Mails darstellt.
Varianten:

+ `AddressError`
  Fehler beim parsen von Mailadressen
+ `TemplateError`
  Fehler beim Lesen der Emailvorlage
+ `MailParseError`
  Fehler beim Erstellen der Email
+ `MailSendError`
  Fehler beim Senden der Email

MailSender

Typ: `class` implements `AdminNotification`
Paket: `layer/data/mail`
Beschreibung: Klasse für das senden von E-Mails an einen Administrator.
Methoden:

+ `{static} new(config: MailConfig): MailSender | MailError`
  Erzeugt ein neues Objekt mit gegebener Konfiguration zum Senden von E-Mails.

MailInfo

Typ: `struct`
Paket: `layer/data/mail`
Beschreibung: Struktur mit allen Informationen zum Verbinden mit einem Mainserver und senden von Emails an Administratoren.
Attribute:

+ `smtp_server: String`
  Domänenname für die Verbindung zu einem Mail-Server über das SMTP-Protokoll.
+ `smpt_port: Integer`
  Port, auf welchem der in `smpt_server` angegebene Mail-Server auf SMTP-Anfragen hört.
+ `username: String`
  Benutzername zur Verbindung mit dem Mail-Server.
+ `password: String`
  Passwort zur Verbindung mit dem Mail-Server.
+ `admin_email_address: String`
  E-Mail-Adresse eines Administrators, der bei Meldeanträgen benachrichtigt wird.

8.8 Komponente `database`

RequestDataAccess

`get_meals` hat eine Änderungen in den Rückgabewerten, da nun entschieden werden kann, ob es Gerichte gibt, keine Gerichte gibt oder wir haben noch keine Daten haben.

Typ: `interface`
Paket: `interface/persistent_data`
Beschreibung: Eine Schnittstelle für GraphQL-Datenbankanfragen.
Methoden:

+ `get_canteen(id: Uuid): Canteen[0..1] | DataError`
  Übergibt die Mensen mit der passenden Uuid aus der Datenbank.
+ `get_canteens(): Canteen[*] | DataError`
  Übergibt alle Mensen aus der Datenbank.
+ `get_meal(id: Uuid, line_id: Uuid, date: Date): Meal[0..1] | DataError`
  Übergibt die Gerichte, bei denen die Parameter übereinstimmen aus der Datenbank.
+ `get_meals(line_id:: Uuid, date: Date): Meal[*][0..1] | DataError`
  Übergibt alle Gerichte, die mit den Parametern übereinstimmen aus der Datenbank. Ein separater Null-Wert wird übergeben, wenn bisher keine Speiseplaninformationen bekannt sind.
+ `get_line(id: Uuid): Line[0..1] | DataError`
  Übergibt die Linie mit der passenden Uuid aus der Datenbank.
+ `get_lines(canteen_id: Uuid): Line[*] | DataError`
  Übergibt die Linien aus der Datenbank.
+ `get_sides(line_id: Uuid, date: Date): Side[*] | DataError`
  Übergibt alle Beilagen einer Linie zu einem bestimmten Tag.
+ `get_visible_images(meal_id: Uuid, client_id: Uuid[0..1]): Image[*] | DataError`
  Übergibt alle Bilder, die einem bestimmten Gericht und Nutzer zugeordnet sind. Bilder, die der Nutzer gemeldet hat, werden nicht übergeben.
+ `get_personal_rating(meal_id: Uuid, client_id: Uuid): Integer[0..1] | DataError`
  Übergibt die Bewertung des Nutzers zu einem Gericht.
+ `get_personal_upvote(image_id: Uuid, client_id: Uuid): Boolean | DataError`
  Prüft, ob das Bild von dem Nutzer aufgewertet wurde.
+ `get_personal_downvote(image_id: Uuid, client_id: Uuid): Boolean | DataError`
  Prüft, ob das Bild von dem Nutzer abgewertet wurde.
+ `get_allergens(food_id: Uuid): Allergen[*] | DataError`
  Übergibt alle Allergene des Gerichts. Bei dem Gericht kann es sich um ein Hauptgericht oder eine Beilage handeln.

`get_additives(food_id: Uuid): Additive[*] | DataError`
\+ Übergibt alle Additive des Gerichts. Bei dem Gericht kann es sich um ein Hauptgericht oder eine Beilage handeln.

CommandDataAccess

Typ: `interface`
Paket: `interface/persistent_data`
Beschreibung: Eine Schnittstelle für GraphQL-Manipulationen.
Methoden:

\+ `get_image_info(image_id: Uuid) : Image | DataError`
Gibt Informationen zu einem Bild zurück.
\+ `hide_image(image_id: Uuid) Void | DataError`
Markiert ein Bild als versteckt, sodass es anderen Nutzern nicht mehr angezeigt wird.
\+ `add_report(image_id: Uuid, client_id: Uuid, reason: ReportReason): Void | DataError`
Speichert einen Meldeantrag eines Nutzers zu einem Bild.
\+ `add_upvote(image_id: Uuid, user_id: Uuid): Void | DataError`
Speichert eine positive Bewertung des angegebenen Nutzers für das angegebene Bild. Eine mögliche negative Bewertung desselben Nutzers wird dabei überschrieben.
\+ `add_downvote(image_id: Uuid, user_id: Uuid): Void | DataError`
Speichert eine negative Bewertung des angegebenen Nutzers für das angegebene Bild. Eine mögliche positive Bewertung desselben Nutzers wird dabei überschrieben.
\+ `remove_upvote(image_id: Uuid, user_id: Uuid): Void | DataError`
Entfernt eine positive Bewertung des angegebenen Nutzers zum angegebenen Bild.
\+ `remove_downvote(image_id: Uuid, user_id: Uuid): Void | DataError`
Entfernt eine negative Bewertung des angegebenen Nutzers zum angegebenen Bild.
\+ `link_image(meal_id: Uuid, user_id: Uuid, image_hoster_id: String, url: String): Void | DataError`
Fügt ein Bild hinzu und verlinkt es mit dem Gericht hinter der UUID.
\+ `add_rating(meal_id: Uuid, user_id: Uuid, rating: Integer): Void | DataError`
Fügt einem Gericht eine Bewertung hinzu. Die Bewertung wird mit der UUID des Nutzers verlinkt.
\+ `get_api_keys(): ApiKey[*] | DataError`
Lädt alle API-Schlüssel aus der Datenbank und gibt diese zurück.

ImageReviewDataAccess

Typ: `interface`
Paket: `interface/persistent_data`
Beschreibung: Eine Schnittstelle zum Erhalten von Bild-Daten.
Methoden:

+ `get_images_for_date(number_of_images: Integer, date: Date): Image[*] | DataError`
Gibt die ersten `number_of_images` Bilder sortiert nach dem Bildrang zurück, die zu Gerichten gehören, die am gegebenen Tag angeboten werden.

+ `get_unvalidated_images_for_next_week(number_of_images: Integer): Image[*] | DataError`
Gibt die ersten `number_of_images` Bilder sortiert nach dem Bildrang zurück, die zu Gerichten gehören, die im Laufe der nächsten Woche angeboten werden und in der letzten Woche nicht schon überprüft wurden.

+ `get_old_images(number_of_images: Integer): Image[*] | DataError`
Gibt die ersten `number_of_images` Bilder sortiert nach dem Datum der letzten Überprüfung (aufsteigend) zurück, die in der letzten Woche nicht schon überprüft wurden.

+ `delete_image(id: Uuid): Void | DataError`
Entfernt ein Bild inklusive seiner Verweise.

+ `mark_as_checked(id: Uuid): Void | DataError`
Markiert alle Bilder mit den übergebenen UUIDs als überprüft.

MealplanManagementDataAccess

ℹ Es wurden die Rückgaben dieses Interfaces geändert, da Referenzen ausreichten, statt ganze Objekte zurückzugeben.

ℹ Bei der Funktion `get_similar_line` wird jetzt auch überprüft, ob die Linie der selben Kantine angehört. Ohne diese Überprüfung gäbe es Überschneidungen in der Datenbank.

ℹ Es wurden Funktionen erstellt, die Gerichte dem Speiseplan hinzufügen.

Typ: `interface`
Paket: `interface/persistent_data`
Beschreibung: Eine Schnittstelle für das Überprüfen von Relationen und das Einfügen von Strukturen. Die mealplanmanagement verwendet diese Schnittstelle.
Methoden:

+ `dissolve_relations(canteen_id: Uuid, date: Date) Void | DataError`
Löst alle Beziehungen der Kantine an gegebenen Tag zu dem Speiseplan. Ohne diese Auflösung kann der Speiseplan nicht fehlerfrei überarbeitet werden.

+ `get_similar_canteen(similar_name: String): Uuid[0..1] | DataError`
Übergibt die Kantine mit dem ähnlichsten Namen.

+ `get_similar_line(similar_name: String, canteen_id: Uuid): Uuid[0..1] | DataError`
Übergibt die Linie mit dem ähnlichsten Namen.

+ `get_similar_meal(similar_name: String, allergens: Allergen[*], additives: Additive[*]): Uuid[0..1] | DataError`
Übergibt das Gericht, das den ähnlichsten Namen, gleiche Allergene und gleiche Additive hat.

+ `get_similar_side(similar_name: String, allergens: Allergen[*], additives: Additive[*]): Uuid[0..1] | DataError`
  Übergibt die Beilagen, die den ähnlichsten Namen, gleiche Allergene und gleiche Additive hat.
+ `update_canteen(uuid: Uuid, name: String, position: Integer): Void | DataError`
  Überschreibt die Bezeichnung einer Mensa. Gibt die betroffene Mensa zurück.
+ `update_line(uuid: Uuid, name: String, position: Integer): Void | DataError`
  Überschreibt die Bezeichnung einer Linie. Gibt die betroffene Linie zurück.
+ `update_meal(uuid: Uuid, name: String): Void | DataError`
  Überschreibt die Bezeichnung eines Gerichts.
+ `update_side(uuid: Uuid, name: String): Void | DataError`
  Überschreibt die Bezeichnung einer Beilage.
+ `add_meal_to_plan(line_id: Uuid, date: Date, meal_id: Uuid, price: Price): Void | DataError`
  Fügt dem Speiseplan an gegeben Tag und gegebener Kantine dieses Gericht hinzu.
+ `add_side_to_plan(line_id: Uuid, date: Date, side_id: Uuid, price: Price): Void | DataError`
  Fügt dem Speiseplan an gegeben Tag und gegebener Kantine diese Beilage hinzu.
+ `insert_canteen(name: String, position: Integer): Uuid | DataError`
  Fügt eine neue Mensa der Datenbank hinzu.
+ `insert_line(canteen_id: Uuid, name: String, position: Integer): Uuid | DataError`
  Fügt eine neue Linie der Datenbank hinzu.
+ `insert_meal(name: String, type: MealType, allergens: Allergen[*], additives: Additive[*]): Uuid | DataError`
  Fügt ein neues Gericht der Datenbank hinzu.
+ `insert_side(name: String, type: MealType, allergens: Allergen[*], additives: Additive[*]): Uuid | DataError`
  Fügt eine neue Beilage der Datenbank hinzu.

DataAccessFactory

ℹ Der Zugriff auf `ApiKeyDataAccess` wurde entfernt, da dieses Interface inzwischen durch `CommandDataAccess` abgedeckt ist.

Typ: `class`
Paket: `layer/data/database`
Beschreibung: Diese Klasse dient zur Verteilung der unterschiedlichen Datenbankanfragen.
Methoden:

+ `{static} new(info: DatabaseInfo, should_migrate: bool) «create»`
  Erstellt ein Objekt zur Verteilung der Datenbankanfragen. Bei der Erstellung wird eine Verbindung zur

Datenbank hergestellt. Falls spezifiziert können Datenbank-Migrationen zum erzeugen der Relationen angewandt werden.

+ `get_api_key_data_access(): ApiKeyDataAccess`
  Übergibt ein Objekt, in dem sich alle Datenbankanfragen befinden, die ApiKeys betreffen.
+ `get_command_data_access(): CommandDataAccess`
  Übergibt ein Objekt, in dem sich alle Datenbankanfragen befinden, die GraphQL-Manipulationen betreffen.
+ `get_image_review_data_access(): ImageReviewDataAccess`
  Übergibt ein Objekt, in dem sich alle Datenbankanfragen befinden, die Bildervalidierung betreffen.
+ `get_mealplan_management_data_access(): MealplanManagementDataAccess`
  Übergibt ein Objekt, in dem sich alle Datenbankanfragen befinden, die den Mensa-Parser betreffen.
+ `get_request_data_access(): RequestDataAccess`
  Übergibt ein Objekt, in dem sich alle Datenbankanfragen befinden, die GraphQL-Objektanfragen betreffen.

PersistentRequestData

Typ: `class` implements `RequestDataAccess`
Paket: `layer/data/database`
Beschreibung: Diese Klasse implementiert alle Datenbankanfragen, die GraphQL-Objektanfragen betreffen.

PersistentCommandData

Typ: `class` implements `CommandDataAccess`
Paket: `layer/data/database`
Beschreibung: Diese Klasse implementiert alle Datenbankanfragen, die GraphQL-Manipulationen betreffen.

PersistentImageReviewData

Typ: `class` implements `ImageReviewDataAccess`
Paket: `layer/data/database`
Beschreibung: Diese Klasse implementiert alle Datenbankanfragen, die Bildervalidierung betreffen.

PersistentMealplanManagementData

Typ: `class` implements `MealplanManagementDataAccess`
Paket: `layer/data/database`
Beschreibung: Diese Klasse implementiert alle Datenbankanfragen, die den Mensa-Parser betreffen.

Canteen

Typ: `struct`
Paket: `interface/persistent_data/model`
Beschreibung: Diese Mensa-Struktur ist angelehnt an die Datenbankentitäten `Canteen`. Diese Struktur wird verwendet für Datenbankoperationen.
Attribute:

+ `id: Uuid`
  Id der Mensa
+ `name: String`
  Name der Mensa

Line

Typ: `struct`
Paket: `interface/persistent_data/model`
Beschreibung: Diese Linien-Struktur ist angelehnt an die Datenbankentitäten `Line`. Diese Struktur wird verwendet für Datenbankoperationen.
Attribute:

+ `id: Uuid`
  Id der Linie.
+ `name: String`
  Name der Linie.
+ `canteen_id: Uuid`
  UUID der zugehörigen Kantine.

Meal

ℹ Die einzelnen Preise sind im `util`-Paket im `Price`-Struct gebündelt.

Typ: `struct`
Paket: `interface/persistent_data/model`
Beschreibung: Diese Gericht-Struktur ist angelehnt an die Datenbankentitäten `Food`, `FoodAllergen` und `FoodAdditive`. Diese Struktur wird verwendet für Datenbankoperationen.
Attribute:

+ `id: Uuid`
  Id des Gerichts.
+ `name: String`
  Name des Gerichts.
+ `meal_type: MealType`
  Der Typ des Gerichts.
+ `price_student: Integer`
  Der Preis des Gerichts für Studenten.
+ `price_employee: Integer`
  Der Preis des Gerichts für Angestellte.
+ `price_guest: Integer`
  Der Preis des Gerichts für Gäste.
+ `price_pupil: Integer`
  Der Preis des Gerichts für Schüler.
+ `price: Price`
  Preise des Gerichts.

+ `last_served: Date[0..1]`
  Der Tag, an dem das Gericht zuletzt serviert wurde.
+ `next_served: Date[0..1]`
  Der Tag, an dem das Gericht als nächstes serviert wird.
+ `frequency: Integer`
  Anzahl, wie oft das Gericht in den Letzten drei Monaten angeboten wurde.
+ `new: Boolean`
  Ob das Gericht neu ist und zuvor noch nie angeboten wurde.
+ `rating_count: Integer`
  Die Anzahl aller Bewertungen des Gerichts.
+ `average_rating: Real`
  Das Durchschnittsbewertung des Gerichts.
+ `date: Date`
  Das Datum an dem das Gericht serviert wird oder wurde. Wird beispielsweise ein Gericht aus der Vergangenheit angefragt, zeigt dieses Datum an ob es das Gericht an diesem Tag gab.
+ `line_id: Uuid`
  UUID der zugehörigen Linie des Gerichts.

Side

ℹ Die einzelnen Preise sind im `util`-Paket im `Price`-Struct gebündelt.

Typ: `struct`
Paket: `interface/persistent_data/model`
Beschreibung: Diese Beilagen-Struktur ist angelehnt an die Datenbankentitäten `Food`, `FoodAllergen` und `FoodAdditive`. Diese Struktur wird verwendet für Datenbankoperationen.
Attribute:

+ `id: Uuid`
  Id der Beilage.
+ `name: String`
  Name der Beilage.
+ `meal_type: MealType`
  Der Typ der Beilage.
+ `price_student: Integer`
  Der Preis der Beilage für Studenten.
+ `price_employee: Integer`
  Der Preis der Beilage für Angestellte.
+ `price_guest: Integer`
  Der Preis der Beilage für Gäste.
+ `price_pupil: Integer`
  Der Preis der Beilage für Schüler.
+ `price: Price`
  Preise der Beilage.

Image

ℹ Dieses Struct übernimmt alle Attribute von `ImageInfo` und ersetzt dieses.

Typ: `struct`
Paket: `interface/persistent_data/model`
Beschreibung: Diese Bild-Struktur ist angelehnt an die Datenbankentität `Image`. Diese Struktur wird verwendet für Datenbankoperationen.
Attribute:

+ `id: Uuid`
  Id des Bildes.
+ `image_hoster_id: String`
  ImageHoster-Identifikator des Bildes.
+ `url: String`
  Direkter Link zum Bild auf der Hoster-Webseite.
+ `rank: Real`
  Rang des Bildes.
+ `upvotes: Integer`
  Menge der Upvotes des Bildes.
+ `downvotes: Integer`
  Menge der Downvotes des Bildes.
+ `approved: Boolean`
  Ab das Bild von einem Administrator als valide bestätigt wurde.
+ `upload_date: Date`
  Hochladedatum des Bildes.
+ `report_count: Integer`
  Anzahl der offenen Meldeanträge zu diesem Bild.

ImageInfo

Typ: `struct`
Paket: `interface/persistent_data/model`
Beschreibung: Diese Bild-Info-Struktur ist angelehnt an die Datenbankentität `ImageRating` und `ImageReport`. Diese Struktur wird verwendet für Datenbankoperationen.
Attribute:

+ `approved: Boolean`
  Ab das Bild von einem Administrator als valide bestätigt wurde.
+ `upload_date: Date`
  Hochladedatum des Bildes.
+ `report_count: Integer`
  Anzahl der offenen Meldeanträge zu diesem Bild.
+ `image_url: String`
  Direkter Web-Link zum Bild.
+ `positive_rating_count: Integer`
  Anzahl an positiven Bewertungen zu diesem Bild.
+ `negative_rating_count: Integer`
  Anzahl an negativen Bewertungen zu diesem Bild.
+ `image_rank: Real`
  Bildrang, nach dem Bilder geordnet sind.

DataError

Typ: `enum`
Paket: `interface/persistent_data`
Beschreibung: Enumerationstyp für mögliche Fehlschläge von Datenanfragen.
Varianten:

+ `NoSuchItem`
  Die angefragten Daten existieren nicht.
+ `InternalError`
  Fehler bei der Abfrage oder Verbindung mit der Datenbank.
+ `TypeConversionError`
  Fehler beim Konvertieren von Integern. Tritt meistens bei der Konvertierung von Postgresql (unsigned) integern in Rust (signed) auf.
+ `UnexpectedNullError`
  Es wurde ein Null-Objekt von der Datenbank zurückgegeben
+ `MigrateError`
  Felhler beim andwenden der Datenbank-Migration zum erstellen der Relationen.

ApiKey

Typ: `struct`
Paket: `interface/persistent_data/model`
Beschreibung: Enthält alle Informationen zu Api-Schlüsseln.
Attribute:

+ `key: String`
  Der Api-Schlüssel.
+ `description: String`
  Eine kurze Beschreibung des Api-Schlüssels.

8.9 Komponente `SwKaParser`

ℹ Diese Komponente hatte zum Zeitpunkt des Entwurfs noch keine Fehlerbehandlung. Das wurde nachgebessert mit dem Fehlerenum `ParseError`. Dadurch ändert sich der Rückgabewert von vielen Funktionen der Komponente.

MealplanParser

Typ: `interface`
Paket: `interface/mensa_parser`
Beschreibung: Diese Schnittstelle erlaubt das Lesen von Speiseplänen.
Methoden:

+ `parse(day: Date): Canteen[*] | ParseError`
  Beginnt den Parse-Vorgang der Mensa-Webseite für Gerichte, die an übergebenen Tag angeboten werden.
+ `parse_all(): (Date, Canteen[*])[*] | ParseError`
  Beginnt den Parse-Vorgang der Mensa-Webseite für Gerichte der nächsten vier Tage.

ParseError

Typ: `enum`
Paket: `interface/mensa_parser`
Beschreibung: Fehlertyp für auftretendes Fehlverhalten in der SwKaParser Komponente.
Attribute:

+ `InvalidHtmlDocument`
  Ein Knoten im HTML-Dokument konnte nicht gefunden werden. Das zu parsende Dokument ist ungültig.
+ `NoConnectionEstablished`
  Es konnte keine Verbindung zum Webserver aufgebaut werden.
+ `DecodeFailed`
  Die Http-Anfrage konnte nicht entschlüsselt werden.
+ `ClientBuilderFailed`
  Der Http-Client konnte nicht erstellt werden.

ParseCanteen

Typ: `struct`
Paket: `interface/mensa_parser/model`
Beschreibung: Die Mensa-Struktur, die alle Speiseplandaten einer Mensa beinhaltet, die ohne Vorverarbeitung ausgelesen werden können.
Attribute:

+ `name: String`
  Bezeichnung der Mensa.
+ `lines: ParseLine[*]`
  Alle zugehörigen Linien der Kantine.
+ `pos: Integer`
  Die Position der Mensa. Wird für die Anzeigereihenfolge benötigt

ParseLine

Typ: `struct`
Paket: `interface/mensa_parser/model`
Beschreibung: Die Linien-Struktur, die eine Bezeichnung sowie alle Gerichte beinhaltet, die ohne Vorverarbeitung ausgelesen werden können.
Attribute:

+ `name: String`
  Bezeichnung der Linie.
+ `dishes: Dish[*]`
  Alle zugehörigen Gerichte der Linie.

+ `pos: Integer`
Die Position der Linie. Wird für die Anzeigereihenfolge benötigt

Dish

Typ: `struct`
Paket: `interface/mensa_parser/model`
Beschreibung: Diese Struktur beinhaltet alle Daten zu einem Gericht, die der SwKa-Webseite entnommen werden können.
Attribute:

+ `name: String`
Bezeichnung des Gerichts.
+ `price: Price`
Preise des Gerichts.
+ `allergens: Allergen[*]`
Allergene des Gerichts.
+ `additives: Additive[*]`
Additive des Gerichts.
+ `meal_type: MealType`
Typ des Gerichts.
+ `env_score: Integer`
Der Umweltscore des Gerichts. Wird nicht verwendet
+ `student: Integer`
Preis des Gerichts für Studenten.
+ `employee: Integer`
Preis des Gerichts für Angestellte.
+ `guest: Integer`
Preis des Gerichts für Gäste.
+ `pupil: Integer`
Preis des Gerichts für Schüler.

SwKaInfo

ℹ Es wurde das `SwKaInfo` Struct erstellt, um alle Informationen für die Initialisierung zusammenzuführen. Der `Scheduler` initialisiert mit diesem Struct diese Komponente.

Typ: `struct`
Paket: `layer/data/swka_parser`
Beschreibung: Beinhaltet alle Informationen für das Inizialisieren der SwKaParser Komponente.
Methoden:

+ `base_url: String`
  Dieser String beinhaltet die Basis-Url die vom SwKaLinkCreator benötigt wird.
+ `valid_canteens: String[*]`
  Beinhaltet Teil-Urls, die zu allen validen Kantinen (= Kantinen die wir auswerten) führen, nachdem sie vom SwKaLinkCreator erstellt wurden.
+ `client_timeout: Integer`
  Anzahl an Millisekunden, die der Client im SwKaHtmlRequest wartet, sobald eine Verbindung zur Website aufgebaut wurde.
+ `client_user_agent: String`
  Beinhaltet der den User-Agend für den Client im SwKaHtmlRequest, sodass Anfragen an die SwKa-Website durchgeführt werden können.

SwKaParseManager

ℹ Die Funktion `get_html` wurde entfernt, da immer mehrere angefragt auf einmal gestellt werden. Somit wird nur `get_html_strings` verwendet.

Typ: `class` implements `MealplanParser`
Paket: `layer/data/swka_parser`
Beschreibung: Eine Implementierung eines MealplanParsers angepasst für die SwKa-Webseite des Studierendenwerks.
Methoden:

+ `{static} new(parse_info: ParseInfo): SwKaParseManager | ParseError`
  Erstellt eine Instanz. Beim Instanziieren werden alle benötigten Objekte für den Parse-Vorgang erstellt. Dazu gehört der SwKaHtmlRequest, der SwKaLinkCreator und der HTMLParser.

SwKaHtmlRequest

Typ: `class`
Paket: `layer/data/swka_parser`
Beschreibung: Diese Klasse baut eine Verbindung zu einer Webseite auf und lädt dessen HTML-Code.
Methoden:

+ `{static} new(client_timeout: Duration, client_user_agent: String): SwKaHtmlRequest | ParseError`
  Erstellt eine Instanz von SwKaHtmlRequest.

+ `get_html_strings(urls: String[*]): String[*] | ParseError`
Baut eine Verbindung zur Webseite mit der übergebenen URL auf. Lädt den HTML-Code der Webseite in einen String und gibt diesen zurück. Dies wird für alle Urls wiederholt. Tritt ein Fehler auf, wird ein Fehler zurückgegeben.

+ `get_html(url: String): String`
Baut eine Verbindung zur Webseite mit der übergebenen URL auf. Lädt den HTML-Code der Webseite in einen String und gibt diesen zurück. Ist keine Verbindung möglich, wird ein leerer String zurückgegeben.

SwKaLinkCreator

Typ: `class`

Paket: `layer/data/swka_parser`

Beschreibung: Diese Klasse baut spezifische Links für die Anfragen an die SwKa-Webseite. Jede Mensa des Studierendenwerks wird über eine andere URL abgefragt.

Methoden:

+ `{static} new(base_url: String, valid_canteens: String[*]) «create»`
Erstellt eine Instanz des SwKaLinkCreators.

+ `get_urls(date: Date): String[*]`
Erstellt alle URLs zu den Mensen der SwKa-Webseite an dem übergebenen Tag und gibt diese zurück.

+ `get_all_urls(): String[*]`
Erstellt alle URLs zu den Mensen der SwKa-Webseite der nächsten vier Tagen.

HTMLParser

Typ: `class`

Paket: `layer/data/swka_parser`

Beschreibung: Diese Klasse interpretiert einen HTML-String in eine Daten-Struktur.

Methoden:

+ `{static} new() «create»`
Erstellt eine Instanz des HTMLParsers.

+ `transform(html: String, position: Integer): (Date, ParseCanteen)[*] | ParseError`
Diese Funktion transformiert den übergebenen HTML-String in eine Liste von Tupeln aus Daten und Mensen. Die Mensa ist dabei jeweils dieselbe. Das Attribut position gibt an, an welcher Position die Mensa später angezeigt wird.

8.10 Komponente `MealplanManagement`

MensaParseScheduling

Typ: `interface`

Paket: `interface/mealplan_management`

Beschreibung: Schnittstelle für die Aktivierung eines Parse-Vorgangs der Mensa-Webseite.

Methoden:

+ `start_update_parsing()`
  Beginnt das Parse-Vorgang der Mensa-Webseite für die Gerichte des heutigen Tages.
+ `start_full_parsing()`
  Beginnt den Parse-Vorgang der Mensa-Webseite für die Gerichte der nächsten vier Wochen.

MealplanManager

Typ: `class` implements `MensaParseScheduling`
Paket: `layer/logic/mealplan_management`
Beschreibung: Implementiert MensaParseScheduling und dessen Funktionen. Die Klasse steuert die Verwaltung geparster Objekte, bevor sie in die Datenbank eingefügt werden.
Methoden:

+ `{static} new(db: MealplanManagementDataAccess, parser: MealplanParser) «create»`
  Erstellt eine neue Instanz eines MealplanManagers.

RelationResolver

Typ: `class`
Paket: `layer/logic/mealplan_management`
Beschreibung: Diese Klasse verwaltet die Relationen von bekannten und neuen Daten.
Methoden:

+ `{static} new(db: MealplanManagementDataAccess) «create»`
  Erstellt eine neue Instanz.
+ `resolve(canteen: ParseCanteen, day: Date): Void | DataError`
  Löst alle Probleme verursacht von Datenbankrelationen betroffener Objekte des Mensa-Parsers. Dazu gehört das Entscheiden, ob ein Gericht eine Beilage oder eine Hauptspeise ist. Sind Probleme durch Relationen behoben worden, so werden die Objekte in die Datenbank eingefügt.

9 Tests

9.1 Unit Tests im Frontend

Die Tests wurden mithilfe `mocktail` erstellt. Dafür wurden für die Datenbank, die Api und den Lokalen Speicher Mocks mit dieser Library erstellt.

9.1.1 View-Model

FavoriteMealAccess:

- test initialisation
- favorite check
  - is favorite
  - is not favorite
- test adding and removing meals
  - add meal already in meals
  - remove meal not in meals
  - add meal to meals (not in meals)
  - remove meal from meals (that is is in meals)

CombinedMealPlanAccess:

- initialization
  - simple initialization
- filter meals
  - allergens
    - change allergens er
    - change allergens er and sn
    - change allergens er, sn and kr
    - remove filter allergens
  - frequency
    - only new
    - only rare
    - all
  - favorites
    - only favorites
    - deactivate
    - activate
    - all
  - rating
    - set rating limit
    - no rating limit
  - food types
    - vegan
    - vegetarian

        - all
      - price
        - price limit student
        - price limit employee
- reset
  - reset filter preferences
- edge cases
  - closed canteen
  - first line closed
  - no data yet
  - no connection
  - all filered

PrefereenceAccess:

- initialization
  - client identifier
  - color scheme
  - meal plan format
  - price category
- test setters
  - set client identifier
  - set color scheme
  - set meal plan format
  - set price category
- initialization with non standard values
  - client identifier
  - color scheme
  - meal plan format
  - price category

ImageAccess:

- upvote
  - failed upvote
  - successful upvote
- delete upvote
  - failed request
  - successful request
- downvote
  - failed request
  - successful request
- delete delete
  - failed request
  - successful request

9.1.2 Model

SharedPreferences:

- client identifer test
- meal plan format test
- color scheme test
- price category test
- canteen test
- filter preferences
  - price
  - rating
  - onlyFavorite
  - sortedBy
  - ascending
  - frequency
  - allergens
  - categories

GraphQlServerAccess:

- remove downvote
- remove upvote
- add downvote
- add upvote
- link image
- report image
- rate meal
- date format
- update all
- update canteen
- meal form id
- get canteen

9.2 Unittests im Backend

Im Backend wurden die Komponenten durch folgende Unit-Tests getestet:

- `layer/data/database/command`
  - `test_remove_upvote`
  - `test_hide_image`
  - `test_get_api_keys`
  - `test_add_downvote`
  - `test_add_rating`
  - `test_link_image`
  - `test_add_report`
  - `test_add_upvote`
  - `test_remove_downvote`
  - `test_get_image_info`
- `layer/data/database/image_review`
  - `test_delete_image`
  - `test_mark_as_checked`
  - `test_get_old_images`
  - `test_get_unvalidated_images_for_next_week`
  - `test_get_images_for_date`
- `layer/data/database/mealplan_management`
  - `test_add_to_plan`
  - `test_dissolve_relations`
  - `test_update_canteen`
  - `test_insert_line`
  - `test_insert_canteen`
  - `test_get_similar_meal`
  - `test_get_similar_side`
  - `test_get_similar_line`
  - `test_get_similar_canteen`
  - `test_insert_food`
  - `test_update_line`
  - `test_update_food`
- `layer/data/database/request`
  - `test_get_canteens`
  - `test_get_line`
  - `test_get_canteen`
  - `test_get_allergens`
  - `test_get_additives`
  - `test_get_lines`
  - `test_get_personal_rating`
  - `test_get_personal_upvote`

  - test_get_visible_images
  - test_get_meals
  - test_get_personal_downvote
  - test_get_sides
  - test_get_meal
- layer/data/flickr_api/api_request
  - valid_request_sizes
  - valid_check_license_request
  - invalid_license_request
  - error_check_license_invalid_photo
  - valid_error_request
  - invalid_request_sizes
- layer/data/flickr_api/flickr_api_handler
  - invalid_determine_photo_id
  - empty_determine_photo_id
  - valid_determine_photo_id
- layer/data/flickr_api/json_parser
  - fallback_get_size
  - invalid_get_size
  - invalid_parse_error
  - valid_check_license
  - valid_get_size
  - valid_parse_error
- layer/data/mail/mail_sender
  - test_get_report
  - test_notify_admin_image_report
- layer/data/swka_parser/html_parser
  - test_1
  - test_canteen_closed
  - test_canteen_closed_de
  - test_invalid
  - test_mensa_moltke
  - test_no_mealplan_shown
  - test_no_meal_data
  - test_not_a_canteen
  - test_not_a_canteen_de
  - test_normal
- layer/data/swka_parser/swka_html_request
  - get_html_response_fail
  - get_html_response_no_fail
  - get_html_strings_response_fail

  - get_html_strings_response_no_fail
- layer/data/swka_parser/swka_link_creator
  - test_get_all_urls
  - test_get_urls
- layer/data/swka_parser/swka_parse_manager
  - sort_and_parse_canteens_with_invalid_urls
  - sort_and_parse_canteens_with_valid_urls
- layer/logic/api_command/auth/authenticator
  - test_auth_add_image
  - test_auth_image_add_down
  - test_auth_image_add_up
  - test_auth_image_remove_down
  - test_auth_image_remove_up
  - test_auth_rate_meal
  - test_auth_report_image
- layer/logic/api_command/command_handler
  - test_add_image
  - test_add_image_downvote
  - test_add_image_upvote
  - test_new
  - test_remove_image_downvote
  - test_remove_image_upvote
  - test_report_image
  - test_set_meal_rating
- layer/logic/image_review/image_reviewer
  - test_review_image_ok
  - test_review_image_throws_error_when_checked
  - test_review_nonexistent_image
  - test_review_nonexistent_image_delete_error
  - test_full_review
- layer/logic/mealplan_management/meal_plan_manager
  - valid_start_full_parsing
  - valid_start_update_parsing
  - layer/logic/mealplan_management/relation_resolver
  - resolve_empty_canteen
  - resolve_canteens
  - resolve_line_with_rand_dishes
  - test_average_calc
- layer/trigger/graphql
  - test_add_image
  - test_api_version

  - test_get_auth_info_correct
  - test_get_auth_info_null
  - test_get_specific_canteen
  - test_get_specific_meal
  - test_image_votes
  - test_recursive_line_canteen_ok
  - test_report_image
  - test_frondend_query
  - test_recursive_meal_line_ok
  - test_set_rating
  - test_complete_request
- layer/trigger/graphql/server
  - test_double_start
  - test_not_running
  - test_graphql
  - test_playground
- layer/trigger/graphql/util
  - test_auth_info_parsing
  - test_auth_info_parsing_client_only
  - test_read_static_header
- layer/trigger/scheduling/scheduler
  - test_double_start
  - test_not_running
  - test_scheduling

10 Datenbankschema Backend

10.1 Entity-Relationship-Model

Im Folgenden wird das Datenbankschema als ER-Model dargestellt, wobei dabei zu beachten ist, dass die Attribute der einzelnen Entitäten hier fehlen und im Relationenschema genauer beschrieben werden.

ℹ Bei der Datenbank im Backend hat sich hauptsächlich geändert, dass die Userrelation nicht mehr existiert, da die User UUIDs von der App generiert und nicht Zentral verwaltet werden. Dadurch fallen die Fremdschlüssel auf die UserId in mehreren anderen Relationen weg. Außerdem wird die Position der Mensa in der Datanbank gespeichert.

10.2 Relationenschema

Im Folgenden ist das Relationenschema mit den Attributen der jeweiligen Entitäten zu sehen. Dabei sind die Schlüssel unterstrichen und die Fremdschlüssel kursiv dargestellt. Die angegeben Typen entsprechen dabei den PostgreSQL-Typen.

User = (userID: UUID)
Food = (foodID: UUID, name: text, food_type: ENUM)
Meal = (foodID: UUID)

MealRating = (userID : UUID, *foodID*: UUID, rating: smallint)
Image = (imageID: UUID, userID : UUID, *foodID*: UUID, id: text, url: text, linkDate: date, lastVerifiedDate: date, approved: boolean, currentlyVisible: boolean)
ImageRating = (*imageID*: UUID, userID : UUID, rating: smallint)
ImageReport = (*imageID*: UUID, userID : UUID, reportDate: date, category: ENUM)

FoodAllergen = (*foodID*: UUID, allergen: ENUM)
FoodAdditive = (*foodID*: UUID, additive: ENUM)
FoodPlan = (*lineID*: UUID, *foodID*: UUID, date: date, priceStudent: Preis, priceEmployee: Preis, pricePupil: Preis, priceGuest: Preis)
Line = (lineID: UUID, *canteenID*: UUID, name: text, position: smallint)
Canteen = (canteenID: UUID, name: text, position: smallint)
APIKeys = (key: text, description: text)

10.3 Domains

Preis

Preis ist vom Typ `smallint` und wird aus Gründen der Erweiter- und Wartbarkeit als Domain modelliert.

10.4 Entitäten

Im Folgenden werden die einzelnen Entitäten genauer beschrieben. Dabei sind alle Attribute, die „ID“ im Name enthalten nur für Schlüssel relevant und ihr Wert hat bis auf die Identifikation keine weitere Bedeutung. Sie werden deswegen im weiteren Prozess nicht weiter beschreiben.

User

Diese Entität enthält alle Identifikatoren der Nutzer, die schon mit dem Sever kommuniziert haben. Die userID wird in der Datenbank zur Identifikation der Nutzer für Bewertungen und Bilder benutzt.

Food

Diese Entität enthält alle Gerichte und Beilagen, die vom Studierendenwerk angeboten werden.
Attribute:
name: Name des Gerichts
food_type: Gerichtstyp.

Meal

Diese Entität enthält den Identifikator aller Gerichte, da diese einige weiteren Eigenschaften haben als Beilagen und deswegen extra behandelt werden müssen.

MealRating

Diese Entität enthält alle Bewertungen, die Nutzer zu Gerichten abgegeben haben.
Attribute:
rating: vom Nutzer abgegebene Bewertung

Image

Diese Entität enthält Informationen zu einem Bild, das von einem Nutzer hochgeladen wurde.
Attribute:
id: von Flickr vergebender Identifikator
url: URL, die direkt zu dem Bild führt (und nicht zur Rahmenseite)
linkDate: Datum, an dem das Bild verlinkt wurde
last_verified_date: Datum, an dem das Bild zuletzt überprüft wurde
approved: gibt an, ob ein Administrator das Bild bereits überprüft hat
currently_visible: gibt an, ob ein Bild angezeigt werden darf - tritt nicht ein, wenn es zu oft gemeldet wurde

ImageRating

Diese Entität enthält alle Bewertungen, die Nutzer zu Bildern abgegeben haben.
Attribute:
rating: vom Nutzer abgegebene Bewertung

ImageReport

Diese Entität enthält alle Informationen zu den Meldeanträgen, die noch nicht von einem Administrator überprüft wurden. Sobald ein Administrator ein Bild überprüft hat, werden die zugehörigen Meldeanträge entfernt.
Attribute:
reportDate: Tag der Meldung
reason: Meldegrund, der vom Nutzer beim Melden eines Bildes angegeben wird.

FoodAllergen

Diese Entität enthält alle Allergene, die in Gerichten enthalten sind.
Attribute:
allergen: Allergen

FoodAdditive

Diese Entität enthält alle Zusatzstoffe, die in Gerichten enthalten sind.
Attribute:
additive: Zusatzstoff

FoodPlan

Diese Entität enthält den Speiseplan und die verschiedenen Preise.
Attribute:
serve_date: Datum des Speiseplaneintrags.
price_student: Preis für Studierende
price_employee: Preis für Mitarbeitende
price_pupil: Preis für Schüler
price_guest: Preis für Gäste

Line

Diese Entität enthält alle Linien der Mensen des Studierendenwerks.
Attribute:
name: Name der Linie
position: Position auf der die Linie auf der Webseite des Studierendenwerks angezeigt wird.

Canteen

Diese Entität enthält die verschiedenen Mensen, die es gibt.
Attribute:
name: Name der Mensa
position: Die position der Mensa in der Reihenfolge von Mensen

APIKey

Diese Entität enthält alle API Schlüssel.
Attribute:
key: API Schlüssel
description: Beschreibung des Schlüssels

11 Datenbankschema Client

Im folgenden alle Änderungen zum Datenbankschema des Clients.

11.1 Entity-Relationship-Model

Im Folgenden wird das Datenbankschema als ER-Model dargestellt, wobei dabei zu beachten ist, dass die Attribute der einzelnen Entitäten hier fehlen und im Relationenschema genauer beschrieben werden.

11.2 Relationenschema

Im Folgenden ist das Relationenschema mit den Attributen der jeweiligen Entitäten zu sehen. Dabei sind die Schlüssel unterstrichen und die Fremdschlüssel kursiv dargestellt. Die angegeben Typen entsprechen dabei den SQLite-Typen.

MealPlan = (mealplanID: TEXT, *lineID*: TEXT, date: TEXT, isClosed: BOOLEAN)
Line = (lineID: TEXT, *canteenID*: TEXT, name: TEXT, position: INT)
Canteen = (canteenID: TEXT, name: TEXT)

Meal = (mealID: TEXT, *mealplanID*: TEXT, name: TEXT, foodtype: ENUM, priceStudent: INT, priceEmployee: INT, pricePupil: INT, priceGuest: INT, individualRating: INT, numberOfRatings: INT, averageRating: DECIMAL(1,1), lastServed: TEXT, nextServed: TEXT, relativeFrequency: TEXT)

Side = (sideID: TEXT, *mealID*: TEXT, name: TEXT, foodtype: ENUM, priceStudent: INT, priceEmployee: INT, pricePupil: INT, priceGuest: INT)

Image = (imageId: TEXT, url: TEXT, *mealId*: TEXT, imageRank: REAL, positiveRatings: INT, negativeRatings: INT, individualRating: INT)
MealAdditive = (*mealID*: TEXT, additive: ENUM)
MealAllgergen = (*mealID*: TEXT, allergen: ENUM)

SideAdditive = (*sideID*: TEXT, additive: ENUM)
SideAllgergen = (*sideID*: TEXT, allergen: ENUM)

Favorites = (favoriteID: TEXT, *lineID*: TEXT, lastDate: TEXT, foodType: ENUM, priceStudent: INT, priceEmployee: INT, pricePupil: INT, priceGuest: INT)

11.3 Entitäten

Im Folgenden werden die einzelnen Entitäten, die sich geändert haben genauer beschrieben.

MealPlan

Diese Entität speichert den Speiseplan für die nächste Woche.
Attribute:
date: Datum des Speiseplaneintrags
isClosed: Besagt, ob die Linie an einem gewissen Tag offen ist, oder nicht
date: Datum des Speiseplaneintrags

Line

Diese Entität enthält alle Linien der Mensen des Studierendenwerks.
Attribute:
name: Name der Linie
position: Position auf der die Linie auf der Webseite des Studierendenwerks angezeigt wird.

Canteen

Diese Entität enthält die verschiedenen Mensen, die es gibt.
Attribute:
name: Name der Mensa

Änderungen: Meal

Diese Entität enthält alle Gerichte, die in der nächsten Woche angeboten werden.

Attribute:
name: Der Name des Gerichts
foodtype: Ein Gerichtstyp
priceStudent: Preis für Studierende
priceEmployee: Preis für Mitarbeitende
pricePupil: Preis für Schüler
priceGuest: Preis für Gäste
individualRating: Die Bewertung des Gerichts vom Nutzer
numberOfRatings: Die Gesamtanzahl an Bewertungen von diesem Gericht
averageRating: Die Durchschnittsbewertung von diesem Gericht
lastServed: Das Datum, an dem dieses Gericht zuletzt serviert wurde
nextServed: Das Datum, an dem dieses Gericht das nächste Mal serviert wird

Die `relativeFrequency` wird jetzt als TEXT und nicht mehr als ENUM abgespeichert.

relativeFrequency: Häufigkeit, wie oft ein Gericht in drei Monaten angeboten wird.

Side

Diese Entität enthält alle zwischengespeicherten Beilagen, die einem Gericht zugeordnet werden.
Attribute:
name: Der Name der Beilage
foodtype: Ein Gerichtstyp
priceStudent: Preis für Studierende
priceEmployee: Preis für Mitarbeitende
pricePupil: Preis für Schüler
priceGuest: Preis für Gäste

Image

ℹ Die Entität `Image` wurde auf die Entität der Backend Datenbank angepasst.

Diese Entität enthält alle Verweise zu den Bildern für Gerichte.
Attribute:
url: Der Link zum Bild
imageRank: Der Rank des jeweiligen Bildes bestimmt seine Sichtbarkeit für den Client.
positiveRatings: Anzahl aller positiven Bewertungen zu diesem Bild.
negativeRatings: Anzahl aller negativen Bewertungen zu diesem Bild.
individualRating: Beinhaltet einen Wert, der bestimmt, ob das Bild vom Client bewertet wurde und wenn wie das Bild vom Client bewertet wurde.

MealAdditive

Diese Entität enthält alle Zusatzstoffe, die in Gerichten enthalten sind.
Attribute:
additive: Einer der Zusatzstoffe

MealAllergen

Diese Entität enthält alle Allergene, die in Gerichten enthalten sind.
Attribute:
allergen: Einer der Allergene

SideAdditive

Diese Entität enthält alle Zusatzstoffe, die in Gerichten enthalten sind.
Attribute:
additive: Einer der Zusatzstoffe

SideAllergen

Diese Entität enthält alle Allergene, die in Gerichten enthalten sind.
Attribute:
allergen: Einer der Allergene

Favorites

Diese Entität speichert alle Favoriten des Nutzers.
Attribute:
lastDate: Das Datum, an dem das Gericht zuletzt angeboten wurde.
foodtype: Ein Gerichtstyp
priceStudent: Preis für Studierende
priceEmployee: Preis für Mitarbeitende
pricePupil: Preis für Schüler
priceGuest: Preis für Gäste