Project Description:

Application for browsing and searching recipes (mainly for main course dishes, soups and desserts). Source of recipes:

https://www.themealdb.com/api.php

About technologies and libraries used

JavaFX

Project is JavaFX application with views stored as .FXML files.

com.googlecode.json-simple

Used for parsing jsons retrieved from mealdb API. See MealdbAPI package.

com.h2database.h2

Used to store application data about its user and his favorite meals. Used in embedded mode file 'default.mv.db' stored in main folder. Configuration in /resources/app.properties file.

jakarta.persistence-api + org.hibernate-core

Used for writing and reading data from application. See Repository and POJOs packages. Configuration in /resources/META-INF/persistence.xml file.

org.apache.logging.log4j

Used for logging. Logs stored in /log folder (and also directed to standard output). Configuration in /resources/log4j2.xml file.

com.google.common.eventbus

Used for communication between View_Controllers.

About patterns used

Singleton

Used In

- UserProxy – class controlling access to currently logged in user. Because there can be only one currently logged in, I decided to use singleton pattern.
- EventBusFactory – class containing main event bus in application ,because application only uses single event bus, I decided to use singleton pattern.

Proxy

Used in UserProxy – class controlling access to currently logged in user.

Event-Subscriber

Mainly used in EventBus and also utilized in few more classes like for example in IngredientsController for restricting ingredients table in response to user input.

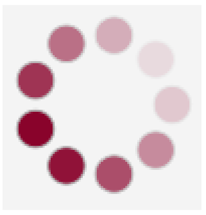
Custom graphic components

LoadingBar

Used as loading animation for every class inheriting after MealsController. When loading listview of meals. Also used in DashboardController in bigger version but only so it would be easy to see in all its glory.

Class extends AnchorPane and we can customize colors, size and speed of animation.

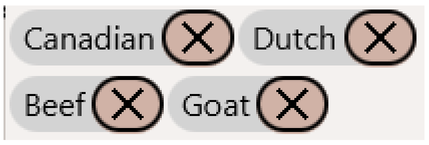
LoadingCircles



Used as loading animation in SearchController (search drawer) and IngredientsController (get inspired) after clicking search. Also used in DashboardController but only so it would be easy to see in all its glory.

Class extends AnchorPane and we can customize size of wheel, circles, amount of circles, speed of turning and speed of changing colors (it goes thru all rainbow colors)

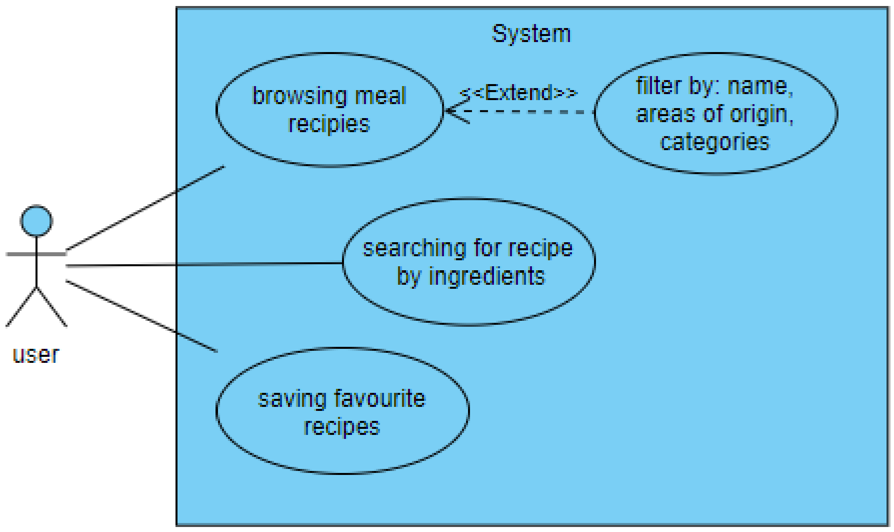
FilterPills



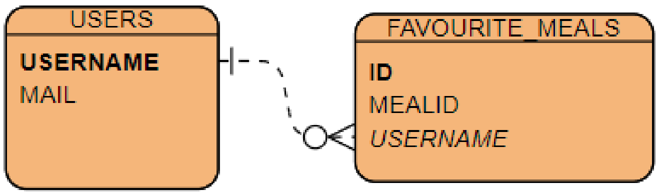
With FilterPillsDisplay and FilterWindow classes used to store and change search vector in SearchController and IngredientsController.

FilterPill extends HBox, FilterPillsDisplay extends FlowPane, and FilterWindow extends Stage. We can customize component only with CSS.

Use Case Diagram


System
browsing meal recipies
<<Extend>>
filter by: name, areas of origin, categories
searching for recipe by ingredients
saving favourite recipes
user

Application Embedded Database Diagram



API description

https://rapidapi.com/thecocktaildb/api/themealdb/

Application utilize 11/13 endpoints (there is two identical endpoints on list)

Access to API is handled by classes inheriting form MealGetter and SearchEngine in MealsdbAPI package:

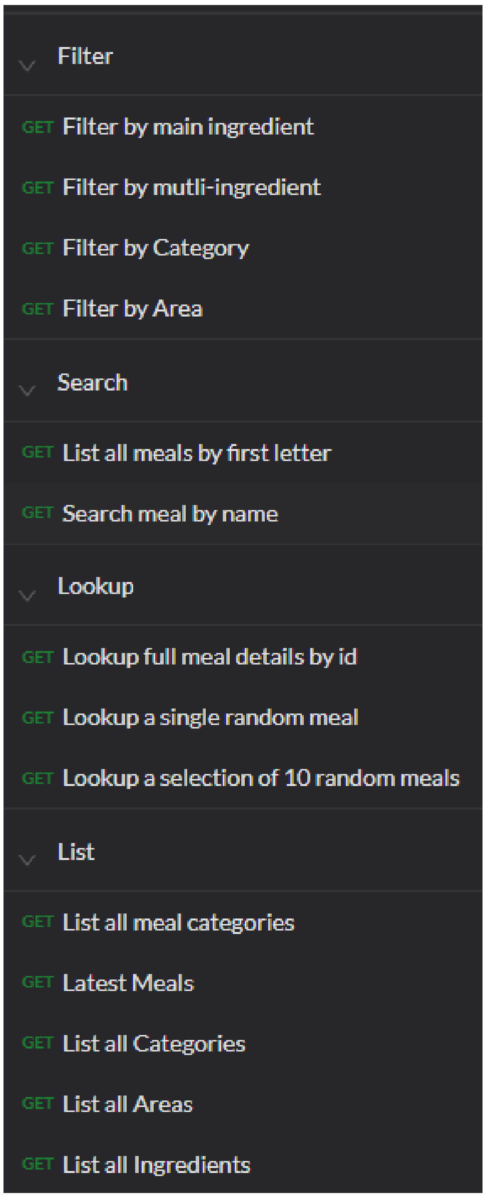
MealGetter uses endpoints listed below:

- Lookup a single random meal
- Lookup full meal details by id
- List all Categories
- List all Areas
- List all Ingredients
- List all meals by first letter (while getting all meals)

SearchEngine uses endpoints listed below:

- Search meal by name
- Filter by main ingredient
- Filter by multi-ingredient
- Filter by Area
- Filter by Category

More about MealGetter and SearchEngine in section “Packages descriptions”.



Filter

GET Filter by main ingredient

GET Filter by mutli-ingredient

GET Filter by Category

GET Filter by Area

Search

GET List all meals by first letter

GET Search meal by name

Lookup

GET Lookup full meal details by id

GET Lookup a single random meal

GET Lookup a selection of 10 random meals

List

GET List all meal categories

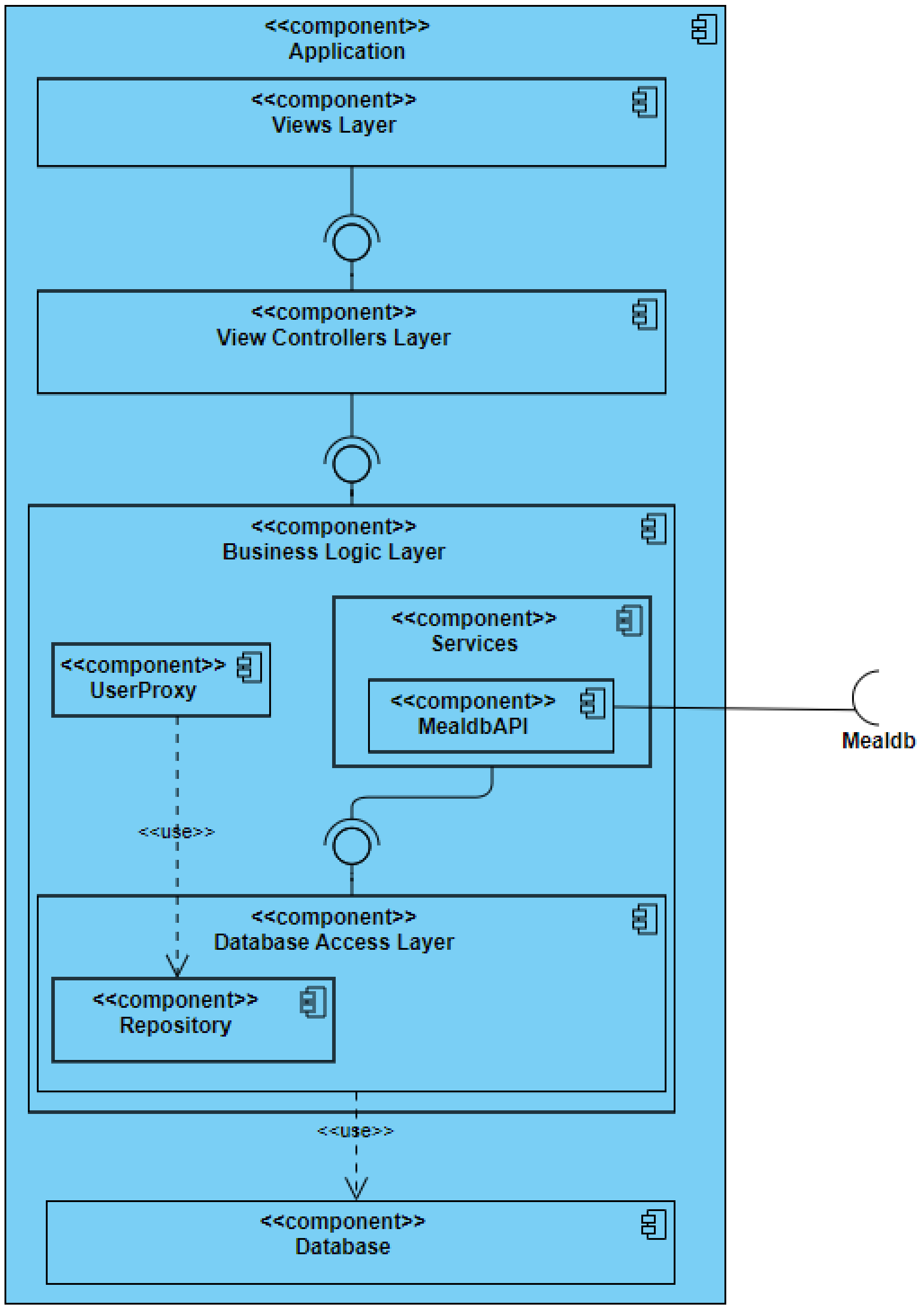
GET Latest Meals

GET List all Categories

GET List all Areas

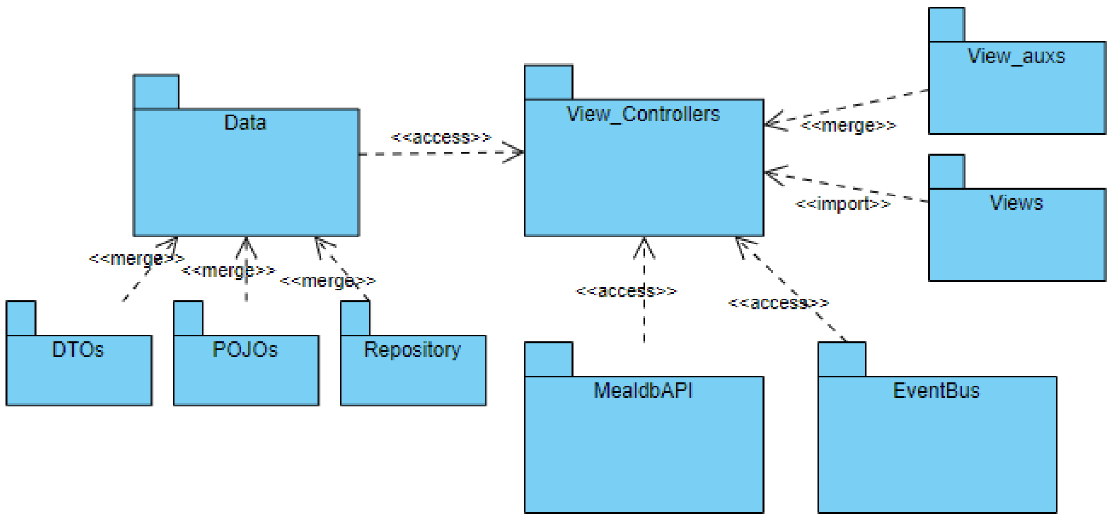
GET List all Ingredients

Component Diagram



<<component>> Application
<<component>> Views Layer
<<component>> View Controllers Layer
<<component>> Business Logic Layer
<<component>> Services
<<component>> UserProxy
<<component>> MealdbAPI
Mealdb
<<use>>
<<component>> Database Access Layer
<<component>> Repository
<<use>>
<<component>> Database

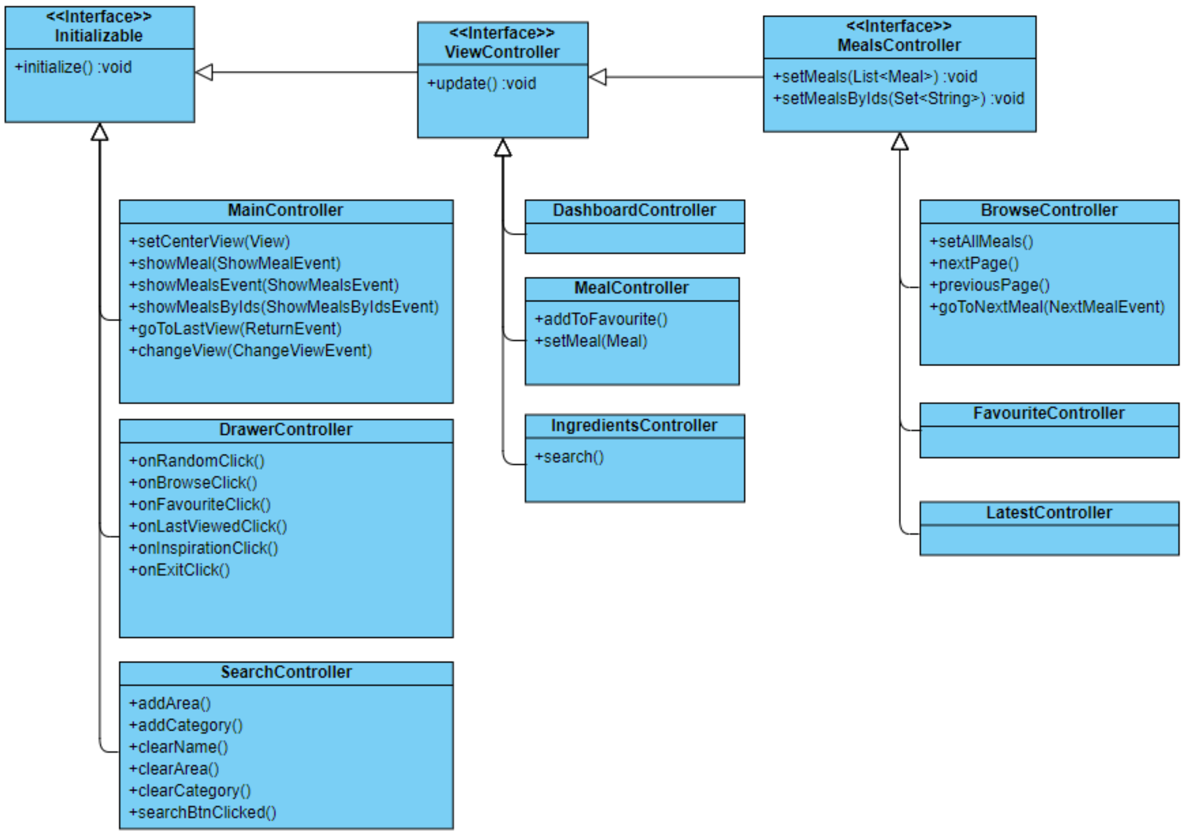
Package diagram



Packages descriptions

View_Controllers

Contains all Controllers classes.



MainController controls behavior of application and correctness of what view should be displayed. Rest of Controllers controls user’s actions in appropriate views.

View_auxs

Contains classes used by View_Controllers like:

- FilterWrap, FilterWindow, FilterPillsDisplay – classes responsible for displaying and controlling filter pills – more in section "Custom graphic components".
- MealCellFactory – class responsible for displaying meals in list like in BrowseController.
- LoadingBar, LoadingCircles – classes responsible for displaying loading animations – more in section "Custom graphic components"

Views

Contains FXML files, used by View_Controllers

EventBus

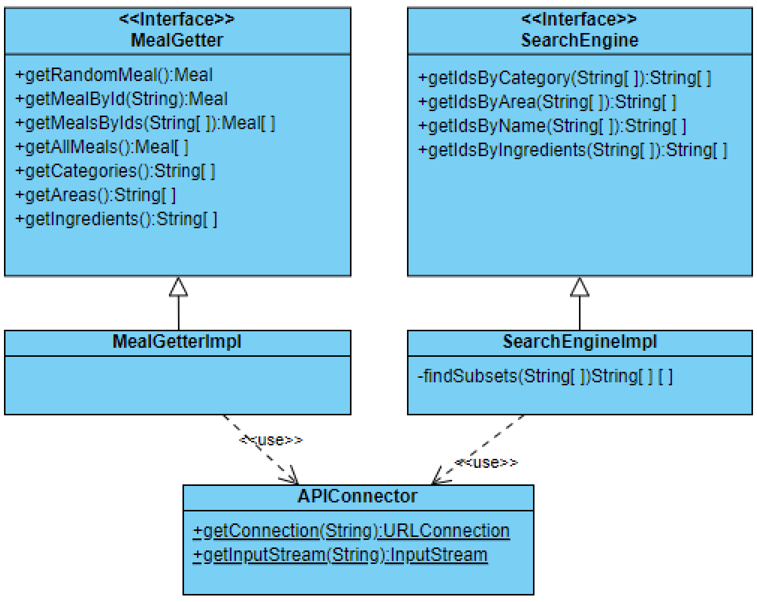
Contains class EventBusFactory – Singleton class, containing com.google.common.eventbus. EventBus – used for communication in Application. And Event Classes used for that communication:

- ShowMealsEvent
- ShowMealEvent
- ShowMealsByIdsEvent
- ChangeViewEvent
- NextMealEvent
- LoadingFinshedEvent
- OpenSearchDrawerEvent
- ReturnEvent
- AddToFavouriteEvent

MealdbApi

Contains Classes responsible for communicating with Mealdb API

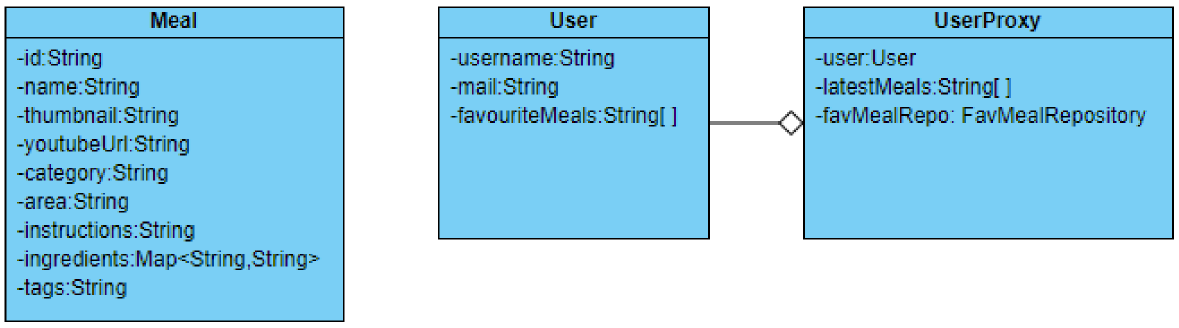
- MealGetter - retrieve meal(meals) by given id(ids) and retrieve list of all areas, categories, ingredients available.
- SearchEngine – retrieve meals ids by given search vectors
- APIConnector – contains static functions returning URLConnection and InputStream for given url


<<Interface>>
MealGetter
+getRandomMeal():Meal
+getMealById(String):Meal
+getMealsByIds(String[ ]):Meal[ ]
+getAllMeals():Meal[ ]
+getCategories():String[ ]
+getAreas():String[ ]
+getIngredients():String[ ]
<<Interface>>
SearchEngine
+getIdsByCategory(String[ ]):String[ ]
+getIdsByArea(String[ ]):String[ ]
+getIdsByName(String[ ]):String[ ]
+getIdsByIngredients(String[ ]):String[ ]
MealGetterImpl
SearchEngineImpl
-findSubsets(String[ ])String[ ] [ ]
<<use>>
<<use>>
APIConnector
+getConnection(String):URLConnection
+getInputStream(String):InputStream

Data

Contains classes responsible for storing data in application.

DTOs



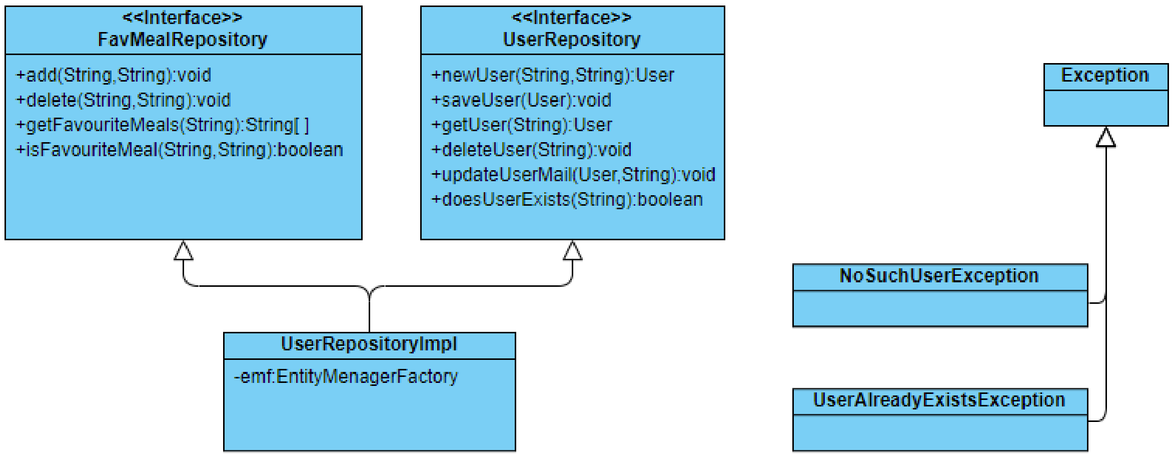
UserProxy is a singleton class that controls access to currently logged-in user. Favorite meals are acquired from database via FavMealRepository interface during logging in the user.

POJOs

Contains DTO classes used for inserting data into database like FavouriteMealPojo and UserPojo. And clas responsible for mapping User class to POJOs and POJOs to User.

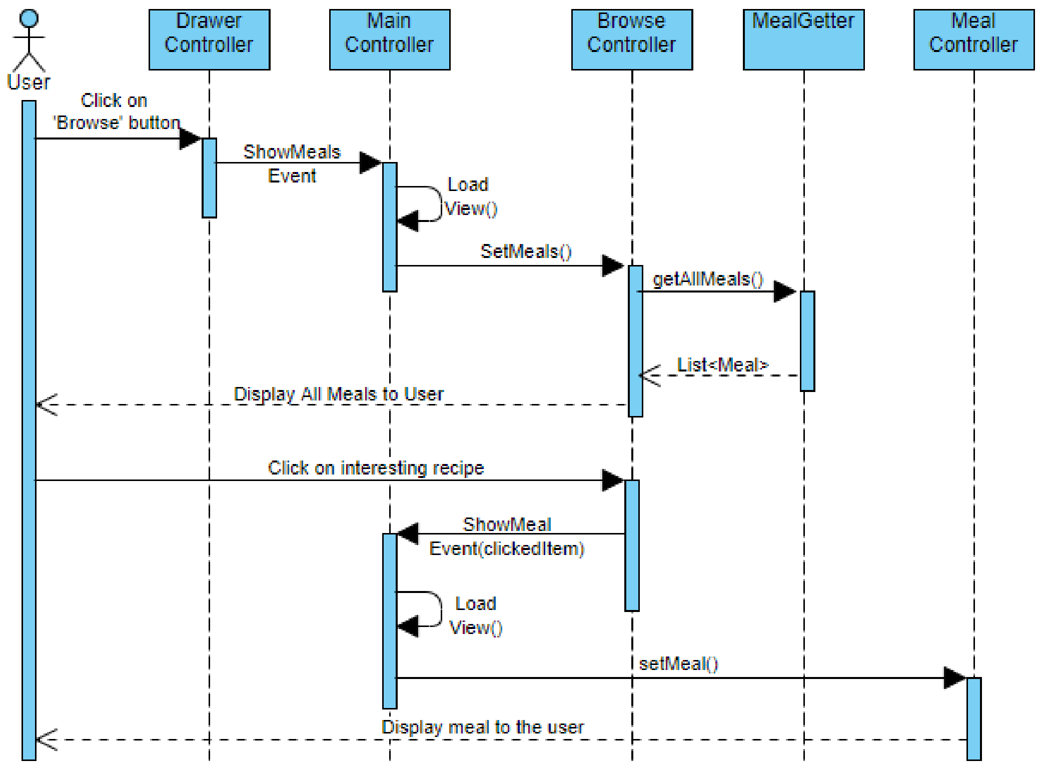
Repository

Contains classes responsible for saving and reading data into/from database

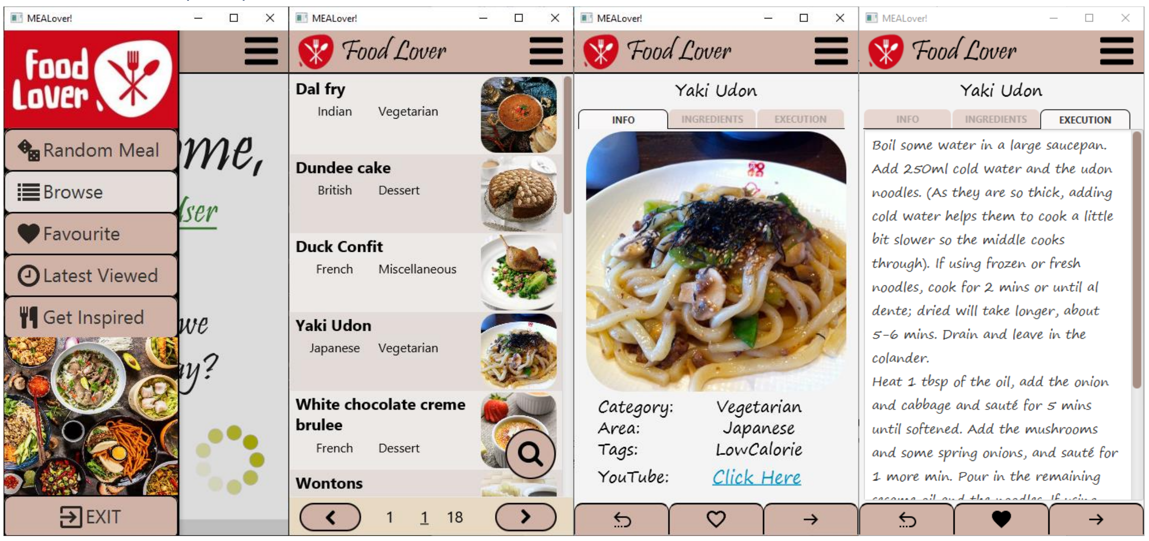

<<Interface>>
FavMealRepository
+add(String,String):void
+delete(String,String):void
+getFavouriteMeals(String):String[ ]
+isFavouriteMeal(String,String):boolean
<<Interface>>
UserRepository
+newUser(String,String):User
+saveUser(User):void
+getUser(String):User
+deleteUser(String):void
+updateUserMail(User,String):void
+doesUserExists(String):boolean
UserRepositoryImpl
-emf:EntityMenagerFactory
Exception
NoSuchUserException
UserAlreadyExistsException

Sequence Diagrams

Browsing Recipes:



Seen from users perspective:



Dal fry
Indian Vegetarian

Dundee cake
British Dessert

Duck Confit
French Miscellaneous

Yaki Udon
Japanese Vegetarian

White chocolate creme brulee
French Dessert

Wontons

Yaki Udon

Category:	Vegetarian
Area:	Japanese
Tags:	LowCalorie
YouTube:	Click Here

Yaki Udon

Boil some water in a large saucepan. Add 250ml cold water and the udon noodles. (As they are so thick, adding cold water helps them to cook a little bit slower so the middle cooks through). If using frozen or fresh noodles, cook for 2 mins or until al dente; dried will take longer, about 5-6 mins. Drain and leave in the colander.
Heat 1 tbsp of the oil, add the onion and cabbage and sauté for 5 mins until softened. Add the mushrooms and some spring onions, and sauté for 1 more min. Pour in the remaining

Notes:

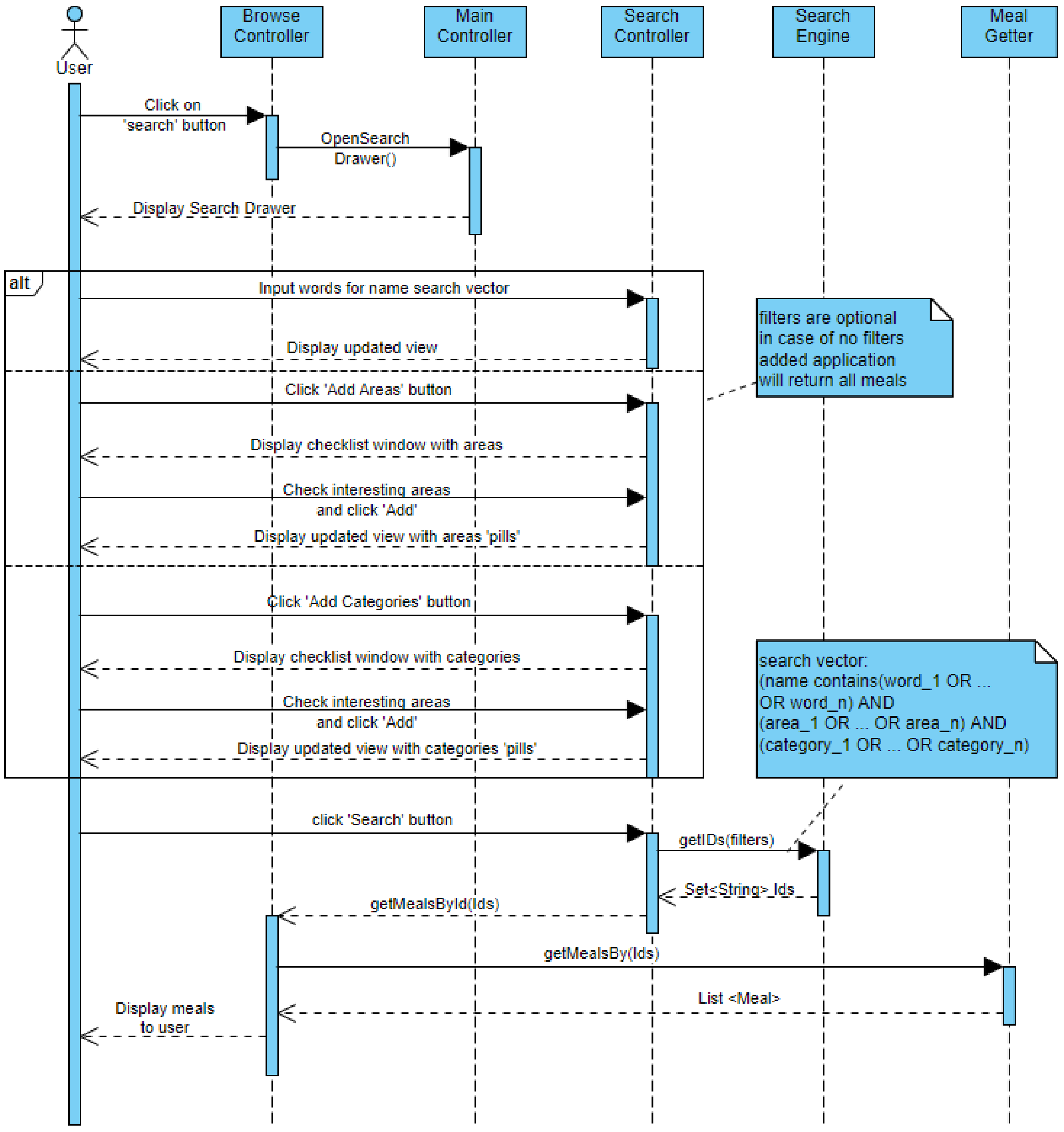
1) By pressing middle action button with heart outline you can add meal to favorite meals -> meal is instantly saved into database as favorite meal of currently logged user and added to list in UserProxy – singleton class that controls access to currently logged-in user. (it will also change heart outline to filled heart).
2) By pressing right action button with arrow pointing to the right you can go to next meal in the list (here it would be “White chocolate crème brulee”) .

3) By pressing left action button with arrow going back you will return to previous view.
4) You can see required ingredients and steps of execution after pressing appropriate tabs below meal name.
5) YouTube link currently doesn't work due to problems with implementing javafx.scene.web package and unavailability of substitutes (paywall, Swing exclusive library)

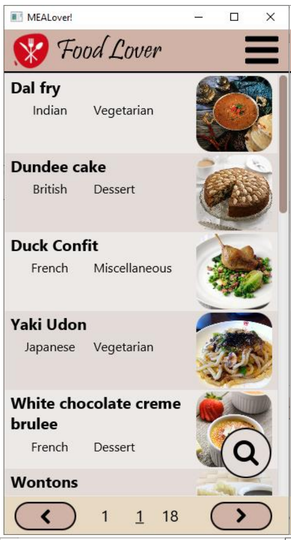
It works similarly for every other change of View like:

1) Showing random meal – the difference is instead of getAllMeals() it utilizes getRandomMeal() function in MealGetter()

Filtering browsing list by name, areas of origin or categories:


User
Browse Controller
Main Controller
Search Controller
Search Engine
Meal Getter
Click on 'search' button
OpenSearch Drawer()
Display Search Drawer
alt
Input words for name search vector
Display updated view
Click 'Add Areas' button
Display checklist window with areas
Check interesting areas and click 'Add'
Display updated view with areas 'pills'
Click 'Add Categories' button
Display checklist window with categories
Check interesting areas and click 'Add'
Display updated view with categories 'pills'
filters are optional in case of no filters added application will return all meals
search vector: (name contains(word_1 OR ... OR word_n) AND (area_1 OR ... OR area_n) AND (category_1 OR ... OR category_n)
click 'Search' button
getIDs(filters)
Set<String> Ids
getMealsById(Ids)
getMealsBy(Ids)
List <Meal>
Display meals to user

Seen from users perspective:


MEALover!

Food Lover

Dal fry — Indian — Vegetarian

Dundee cake — British — Dessert

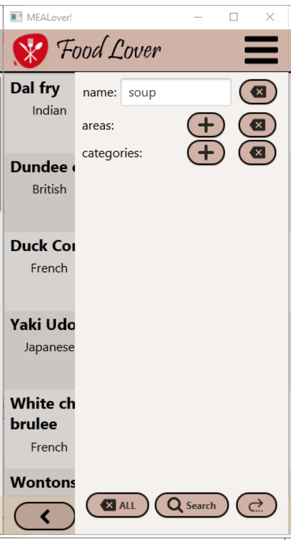
Duck Confit — French — Miscellaneous

Yaki Udon — Japanese — Vegetarian

White chocolate creme brulee — French — Dessert

Wontons

1 1 18

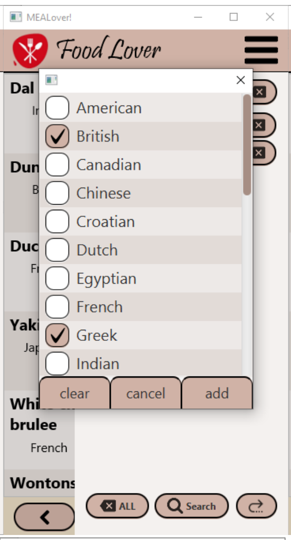

MEALover!

Food Lover

name: soup

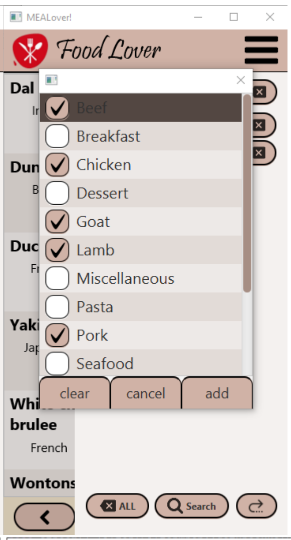
areas:

categories:


MEALover!

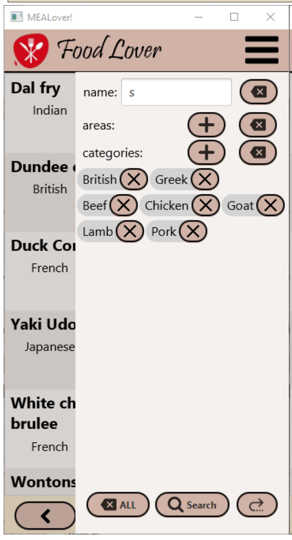
Food Lover

American ☐
British ✓
Canadian ☐
Chinese ☐
Croatian ☐
Dutch ☐
Egyptian ☐
French ☐
Greek ✓
Indian ☐


MEALover!

Food Lover

Beef ✓
Breakfast ☐
Chicken ✓
Dessert ☐
Goat ✓
Lamb ✓
Miscellaneous ☐
Pasta ☐
Pork ✓
Seafood ☐


MEALover!

Food Lover

name: s

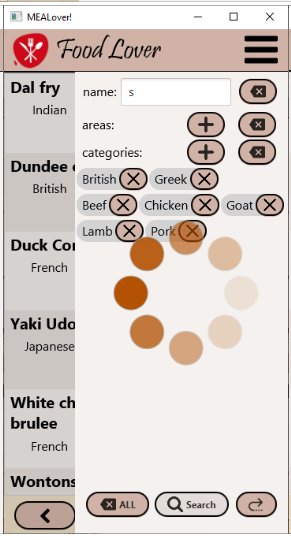
areas:

categories:

British Greek

Beef Chicken Goat

Lamb Pork


MEALover!

Food Lover

name: s

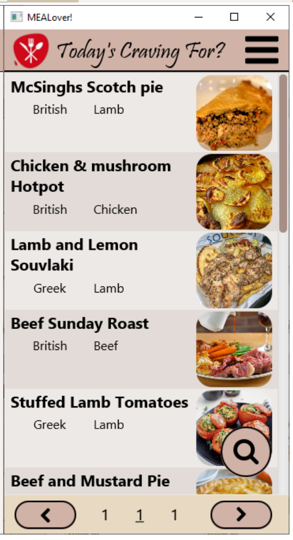
areas:

categories:

British Greek

Beef Chicken Goat

Lamb Pork


MEALover!

Today's Craving For?

McSinghs Scotch pie — British — Lamb

Chicken & mushroom Hotpot — British — Chicken

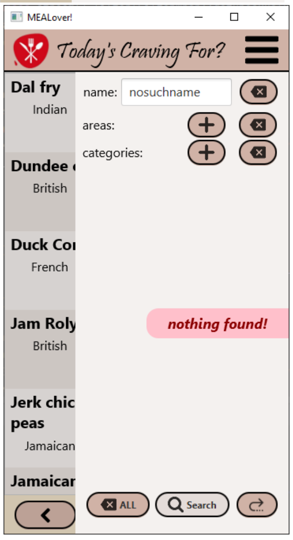
Lamb and Lemon Souvlaki — Greek — Lamb

Beef Sunday Roast — British — Beef

Stuffed Lamb Tomatoes — Greek — Lamb

Beef and Mustard Pie

1 1 1


MEALover!

Today's Craving For?

name: nosuchname

areas:

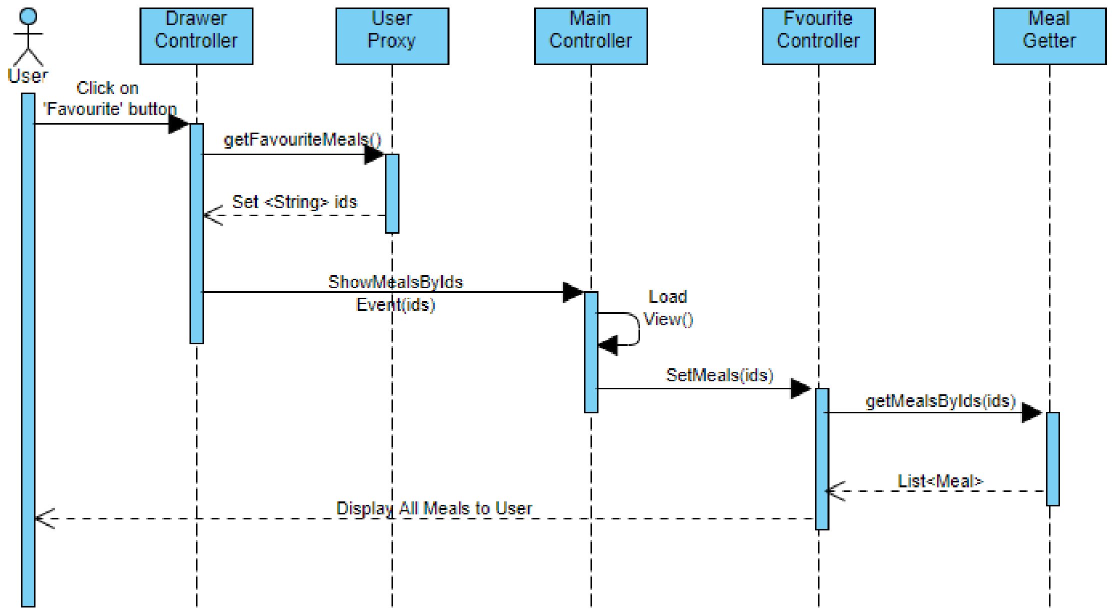
categories:

nothing found!

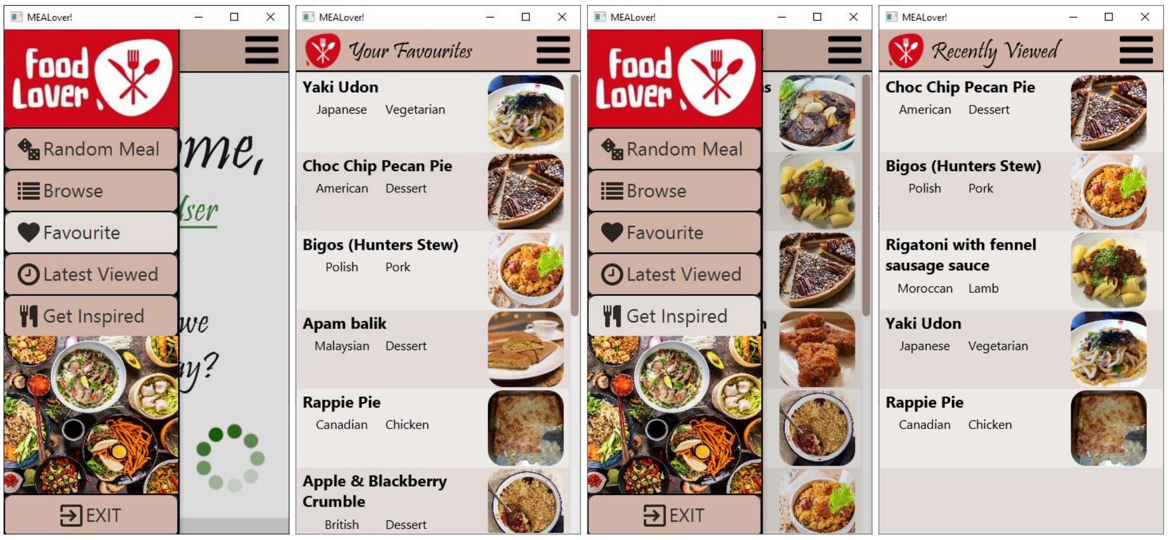
Notes:

2) As mentioned, search vector is (word_1 OR … OR word_n) AND (area_1 OR … OR area_n) AND (category_1 OR … category_n) – 'word_n' is word in field described as name separated form other words by at least one whitespace character.
3) Alternatively if search engine found nothing that fits requirements it displays proper panel ejecting form right side of the screen (last screen)

Browsing (and saving) favorite recipes (and recently viewed recipes)



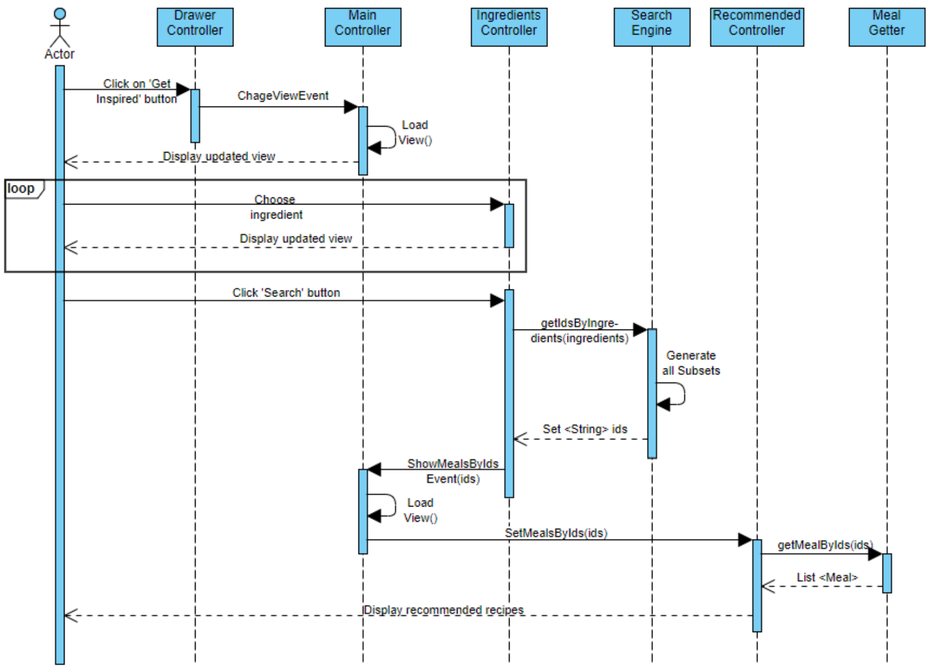
Seen from users perspective:



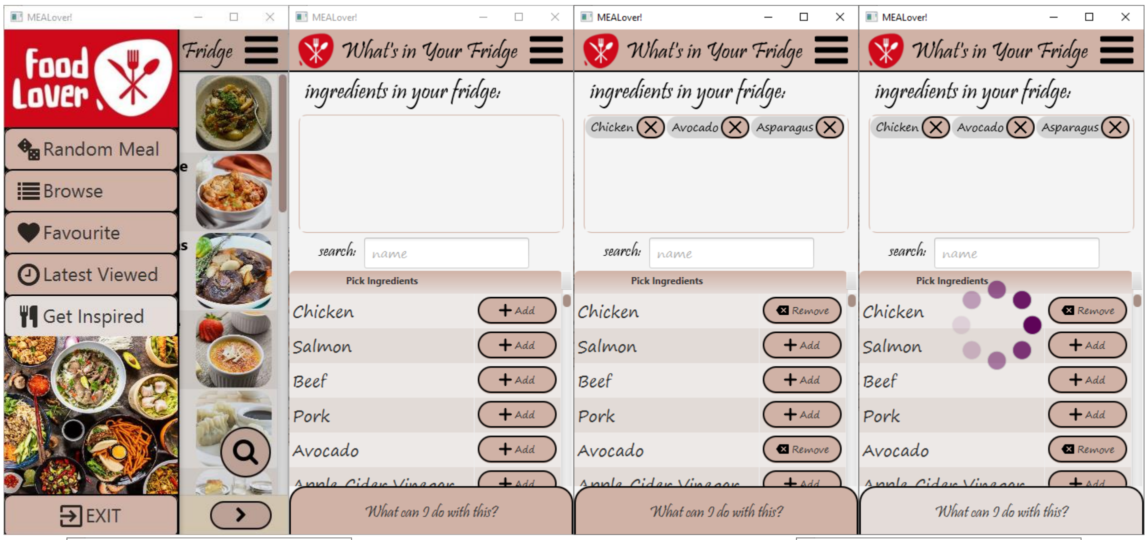
Notes:

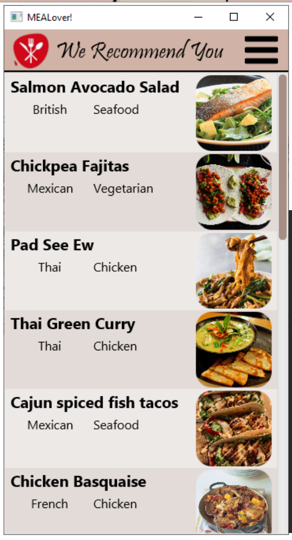
1) List of favorite and recently viewed meals are both stored as sets of meals ids (Strings) in UserProxy – singleton class that controls access to currently logged-in user. Favorite meals are acquired from database via FavMealRepository interface during logging user. Latest viewed recipes are also stored as set of ids in UserProxy but are not stored in database.

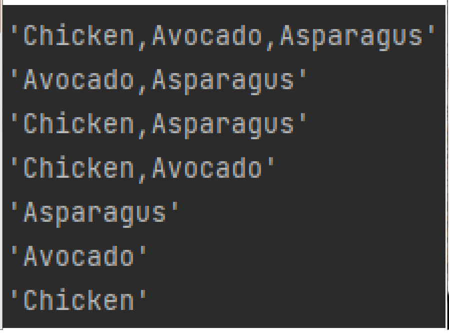
Search for recipe by ingredients



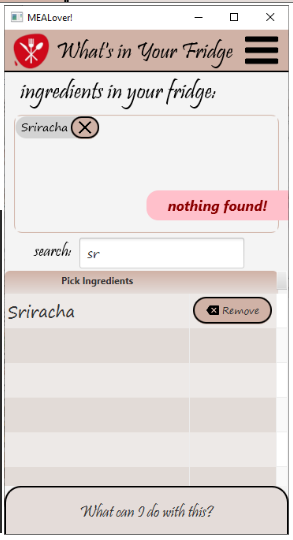
Seen from the user perspective







```
'Chicken,Avocado,Asparagus'
'Avocado,Asparagus'
'Chicken,Asparagus'
'Chicken,Avocado'
'Asparagus'
'Avocado'
'Chicken'
```



Notes:

1) For now, for easier implementation the role of RecommendedController is fulfilled by FavouriteController (its violation of Single Responsibility rule of SOLID, because now FavouriteController has two reasons to change) – in near future it should be split accordingly.
2) To multiply number of recipes found, SearchEngine first produces all possibly subsets of ingredients array and sorts them to increase probability of recommending proper recipe (see before last screen).
3) In case of incapability of finding recipe IngredientsController displays proper panel ejecting form right side of the screen (last screen)