BYM

Generated by Doxygen 1.9.8

Глава 1

Namespace Index

1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Глава 2

Hierarchical Index

2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Глава 3

Class Index

3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Глава 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

Глава 5

Namespace Documentation

5.1 bym Namespace Reference

Namespaces

- namespace BYM_project

5.2 bym.BYM_project Namespace Reference

Namespaces

- namespace main_app

5.3 bym.BYM_project.main_app Namespace Reference

Namespaces

- namespace admin
- namespace apps
- namespace cart
- namespace config
- namespace conversions_functions
- namespace forms
- namespace functions_backend
- namespace models
- namespace parser
- namespace recommendation_system
- namespace search
- namespace serializers
- namespace tests
- namespace views

5.4 bym.BYM_project.main_app.admin Namespace Reference

5.5 bym.BYM_project.main_app.apps Namespace Reference

Classes

- class MainAppConfig

5.6 bym.BYM_project.main_app.cart Namespace Reference

Classes

- class Cart

  Класс для работы с карзиной пользователья

5.7 bym.BYM_project.main_app.config Namespace Reference

Variables

- webdriver_name = os.getenv("webdriver_name")
- list urls_lavka

5.7.1 Variable Documentation

5.7.1.1 urls_lavka

list bym.BYM_project.main_app.config.urls_lavka

Initial value:

```
00001 = [
00002     'https://lavka.yandex.ru/10743/category/ovoshchi_griby_i_zelen',
00003     'https://lavka.yandex.ru/10743/category/Frukty_i_yagody',
00004     'https://lavka.yandex.ru/10743/category/milk_products',
00005     'https://lavka.yandex.ru/10743/category/cheese',
00006     'https://lavka.yandex.ru/10743/category/kefir_smetana_tvorog',
00007     'https://lavka.yandex.ru/10743/category/e64b861bd9a34ebc9103dbf15e2d2932',
00008     'https://lavka.yandex.ru/10743/category/molochnoe_dlya_detej',
00009     'https://lavka.yandex.ru/10743/category/hleb',
00010     'https://lavka.yandex.ru/10743/category/vipechka',
00011     'https://lavka.yandex.ru/10743/category/hlebcy',
00012     'https://lavka.yandex.ru/10743/category/beef_pork',
00013     'https://lavka.yandex.ru/10743/category/wurst_all',
00014     'https://lavka.yandex.ru/10743/category/ryba_i_moreprodukty',
00015     'https://lavka.yandex.ru/10743/category/zakuski_i_pashtety',
00016     'https://lavka.yandex.ru/10743/category/morozhenoe',
00017     'https://lavka.yandex.ru/10743/category/pelmeni_i_vareniki',
00018     'https://lavka.yandex.ru/10743/category/frozen_vegetables_and_berries',
00019     'https://lavka.yandex.ru/10743/category/deserty',
00020     'https://lavka.yandex.ru/10743/category/polufabrikaty',
00021     'https://lavka.yandex.ru/10743/category/553b84d49692448bb06588f9d694cbde',
00022     'https://lavka.yandex.ru/10743/category/b9c6c90166f1487888de39b933119747',
00023     'https://lavka.yandex.ru/10743/category/makarony_krupy_muka',
00024     'https://lavka.yandex.ru/10743/category/hlopya_i_myusli',
00025     'https://lavka.yandex.ru/10743/category/oils_sauces_spices',
00026     'https://lavka.yandex.ru/10743/category/coffee_and_cocoa',
00027     'https://lavka.yandex.ru/10743/category/tea',
00028     'https://lavka.yandex.ru/10743/category/91ba4698f47f450087e3e8de50d93a15'
00029 ]
```

Definition at line 8 of file config.py.

5.7.1.2 webdriver_name

bym.BYM_project.main_app.config.webdriver_name = os.getenv("webdriver_name")

Definition at line 6 of file config.py.

5.8 bym.BYM_project.main_app.conversions_functions Namespace Reference

Functions

- how_many_add (category_id, need_unit, have_unit, need, have)
  Расчет сколько штук товара необходимо купть на что бы полностью покрыть рецепт
- get_lavka_unit_id (name_unit)
  Исчет единицу измерения из магазина в базе данных, если нет создает

5.8.1 Function Documentation

5.8.1.1 get_lavka_unit_id()

bym.BYM_project.main_app.conversions_functions.get_lavka_unit_id (
name_unit)

Исчет единицу измерения из магазина в базе данных, если нет создает

Parameters

name_unit	string

Returns

UnitOfMeasure

Definition at line 47 of file conversions_functions.py.

5.8.1.2 how_many_add()

bym.BYM_project.main_app.conversions_functions.how_many_add (
category_id,
need_unit,
have_unit,
need,
have)

Расчет сколько штук товара необходимо купть на что бы полностью покрыть рецепт

Parameters

category↩ _id	int
need_unit	UnitOfMeasure - еденицы измерения продукта по рецепту (например стаканы)
have_unit	UnitOfMeasure - еденицы измерения товара из магазина (например граммы)
need	int - необходимое количество по рецепту (например 5, have_unit - стаканы)
have	int - количество в одном юните в магазине (например 500, have_unit-граммы)

Returns

kol int - сколько нужно заказать

Definition at line 6 of file conversions_functions.py.

5.9 bym.BYM_project.main_app.forms Namespace Reference

Classes

- class SearchDishForm

5.10 bym.BYM_project.main_app.functions_backend Namespace Reference

Functions

- start_session (request)
  Начало сессии с пользователем и создание его кортежа в бд по session_id.
- get_dish_types ()
  Запрос в бд на получение типов блюд и соответстующих им под типов
- get_particular_dish (dish_id)
  Возвращает конкретное блюдо, стадии его приготовления и необходимые игридиенты
- string_to_minutes (string)
  Возвращает количество минут в строке Варианты входа: 'n часов', 'n минут', 'n часов k минут'.
- convert_fraction (string)
  Возвращает float из строки с числом
- number_and_measure (string)
  Возвращает по строке количества продукта число + единицу измерения в начальной форме

5.10.1 Function Documentation

5.10.1.1 convert_fraction()

bym.BYM_project.main_app.functions_backend.convert_fraction (
string)

Возвращает float из строки с числом

Parameters

string	string

Returns

float

Definition at line 68 of file functions_backend.py.

5.10.1.2 get_dish_types()

bym.BYM_project.main_app.functions_backend.get_dish_types ( )

Запрос в бд на получение типов блюд и соответстующих им под типов

Returns

dict {type:[subtype,...]}

Definition at line 21 of file functions_backend.py.

5.10.1.3 get_particular_dish()

bym.BYM_project.main_app.functions_backend.get_particular_dish (
dish_id)

Возвращает конкретное блюдо, стадии его приготовления и необходимые игридиенты

Parameters

dish↩ _id	int

Returns

dish Dish

cooking_stages [CookingStage,...]

ingredients [{'name':string, amount:int}] - список ингридиентов - название категории их кол-во

Definition at line 32 of file functions_backend.py.

5.10.1.4 number_and_measure()

bym.BYM_project.main_app.functions_backend.number_and_measure (
string)

Возвращает по строке количества продукта число + единицу измерения в начальной форме

Parameters

string	string

Returns

float

string

Definition at line 79 of file functions_backend.py.

5.10.1.5 start_session()

bym.BYM_project.main_app.functions_backend.start_session (
request)

Начало сессии с пользователем и создание его кортежа в бд по session_id.

Parameters

request	HttpRequest

Definition at line 7 of file functions_backend.py.

5.10.1.6 string_to_minutes()

bym.BYM_project.main_app.functions_backend.string_to_minutes (
string)

Возвращает количество минут в строке Варианты входа: 'n часов', 'n минут', 'n часов k минут'.

Parameters

string	string

Returns

int

Definition at line 52 of file functions_backend.py.

5.11 bym.BYM_project.main_app.models Namespace Reference

Classes

- class Category

категории различных продуктов для приготовления
name – название категории (например: бакалея или помидоры)
supreme_category_id – надкатегория (например надкатегорией римских помидор являются помидоры)

- class Conversion

  коэффициенты для перевода единиц измерения
  coefficient – число на которое нужно умножить что бы перевести from_unit_id в to_unit_id
  category_id – категория для которой выполняется перевод (очевидно что коэффициент для перевода штук в граммы для помидоров и огурцов отличается)
  from_unit_id– единица измерения из которой выполняется перевод
  to_unit_id – единица измерения в которую выполняется перевод

- class CookingStage

  шаги для приготовления блюда
  description – текст того, что нужно сделать на этом шаге
  order – порядковый номер шага
  dish_id – блюдо, к которому относится этот шаг
  image – ссылка на изображение шага

- class Country

  Таблица с кухнями различных стран
  name – название кухни (например Русская кухня)

- class Dish

  информация о блюде
  name – названия блюда
  pers_num – кол-во порций
  cooking_time_minutes – время готовки в минутах
  calories – кол-во калорий в блюде
  proteins – кол0белки в блюде
  carbohydrates – углеводы в блюде
  fats – жиры
  country_id – страна из кухни которой это блюдо (например итальянская кухня)
  subtype_id – подтип блюда (например, фруктовые десерты)
  image – ссылка на изображения блюда

- class DishCategory

  информация о составе блюда.

- class Menu

  состав заказа
  amount– кол-во товара, которое было заказано
  dish_id – товар который был заказан
  order_id – заказ в котором этот товар был заказан

- class Order

  информация о заказах.

- class Product

  информация о конкретном товаре в магазине
  name – название товара
  price – цена
  amount – количество (в unit_id) товара которое в одной единице заказ (например 300, unit_id – граммы)
  link – ссылка на товар
  shop_id – магазин, из которого товар
  unit_id – единица измерения
  user_id – пользователь для которого был спаршен данный товар

- class ProductLog

информация о ранее заказанных товарах в магазине
name – название товара
price – цена
amount – количество (в unit_id) товара которое в одной единице заказ (например 300, unit_id – граммы)
quantity – количество единиц, которое было заказано
link – ссылка на товар
shop_id – магазин, из которого товар
order_id – заказ, в котором был куплен данный товар
unit_id – единица измерения

- class Shop

  информация о магазине
  name – название магазина
  link – ссылка на магазин

- class SubType

  таблица с подтипами блюд
  name – название подтипа (например фруктовые десерты)
  type_id – тип к которому относится подтип

- class Type

  таблица с типами блюд
  name – название типа (например Завтраки)

- class UnitOfMeasure

  единицы изменения
  name – название единицы измерения (например штуки или кг)
  supreme_unit_id – основная форма категории (например Supreme_unit_id у "штуки" это "штука")

- class User

  Информация о пользователе.

5.12 bym.BYM_project.main_app.parser Namespace Reference

Classes

- class Lavka

  Определяет класс используемый для парсинга лавки.

- class Parser

  Определяет базавый класс используемый для парсинга всех сайтов

Functions

- parse_by_address (user_id, city_street, number)

5.12.1 Function Documentation

5.12.1.1 parse_by_address()

```
bym.BYM_project.main_app.parser.parse_by_address (
        user_id,
        city_street,
        number )
```

Definition at line 16 of file parser.py.

5.13 bym.BYM_project.main_app.recommendation_system Namespace Reference

Classes

- class RecommendationSystem

5.14 bym.BYM_project.main_app.search Namespace Reference

Classes

- class SearchDish

  Класс реализует функционал поиска и сортировки блюд по строке поиска, фильтрам блюда, рекомендациям конткретному пользователю и указанному адресу
- class SearchProduct

5.15 bym.BYM_project.main_app.serializers Namespace Reference

Classes

- class UserSerializer

5.16 bym.BYM_project.main_app.tests Namespace Reference

5.17 bym.BYM_project.main_app.views Namespace Reference

Classes

- class ChangeCartApi
- class HomePageApi
- class ParseProductsApi
- class SearchDishApi
- class UserApiView

Functions

- HomePageView (request)
- Dish (request, id)
- FinalPage (request)
- my_view (request)
- update_counter (request)
- parse_adress (request)

5.17.1 Function Documentation

5.17.1.1 Dish()

bym.BYM_project.main_app.views.Dish (
request,
id)

Definition at line 44 of file views.py.

5.17.1.2 FinalPage()

bym.BYM_project.main_app.views.FinalPage (
request)

Definition at line 71 of file views.py.

5.17.1.3 HomePageView()

bym.BYM_project.main_app.views.HomePageView (
request)

Definition at line 18 of file views.py.

5.17.1.4 my_view()

bym.BYM_project.main_app.views.my_view (
request)

Definition at line 89 of file views.py.

5.17.1.5 parse_adress()

bym.BYM_project.main_app.views.parse_adress (
request)

Definition at line 104 of file views.py.

5.17.1.6 update_counter()

bym.BYM_project.main_app.views.update_counter (
request)

Definition at line 95 of file views.py.

Глава 6

Class Documentation

6.1 bym.BYM_project.main_app.cart.Cart Class Reference

Класс для работы с карзиной пользователья

Public Member Functions

- __init__ (self, user_id)
  Находит текущий Order по user_id.
- change_order (self, dish_id, add_flg)
  Изменяем корзину блюд пользователя - или добавить (+1), или убрать (-1)
- get_order_array (self)
  Возвращает текущую корзину блюд пользователя
- get_categories_list (self, slice)
  Выдает slice наиболие подходящих товаров в каждой категории

Static Public Member Functions

- get_categories (order_array)
  Раскладывает карзину пользователя (в виде блюд), на товары если в рецептах товары в разных единицах измерения приводит к единой

Public Attributes

- order_obj

6.1.1 Detailed Description

Класс для работы с карзиной пользователья

Definition at line 5 of file cart.py.

6.1.2 Constructor & Destructor Documentation

6.1.2.1 __init__()

bym.BYM_project.main_app.cart.Cart.__init__ (
self,
user_id)

Находит текущий Order по user_id.

Parameters

user↩ _id	

Definition at line 8 of file cart.py.

6.1.3 Member Function Documentation

6.1.3.1 change_order()

bym.BYM_project.main_app.cart.Cart.change_order (
self,
dish_id,
add_flg)

Изменяем корзину блюд пользователя - или добавить (+1), или убрать (-1)

Parameters

dish↩ _id	int
add_flg	bool - True прибавление, False вычитание

Definition at line 14 of file cart.py.

6.1.3.2 get_categories()

bym.BYM_project.main_app.cart.Cart.get_categories (
order_array) [static]

Раскладывает карзину пользователя (в виде блюд), на товары если в рецептах товары в разных единицах измерения приводит к единой

Parameters

order_array	[{"dish": Dish, "amount": int}, ...]

Returns

categories [{"category": Category, "amount": float, "unit_name": string}, ...]

Definition at line 45 of file cart.py.

6.1.3.3 get_categories_list()

bym.BYM_project.main_app.cart.Cart.get_categories_list (
self,
slice)

Выдает slice наиболие подходящих товаров в каждой категории

Parameters

slice	int

Returns

categories_list[{"category": string, "category_amount": int, "category_unit_name": string, "product_list": [Product, ...]}, ...]

Definition at line 103 of file cart.py.

6.1.3.4 get_order_array()

bym.BYM_project.main_app.cart.Cart.get_order_array (
self)

Возвращает текущую корзину блюд пользователя

Returns

order_array [{"dish": Dish, "amount": int}, ...]

Definition at line 32 of file cart.py.

6.1.4 Member Data Documentation

6.1.4.1 order_obj

bym.BYM_project.main_app.cart.Cart.order_obj

Definition at line 12 of file cart.py.

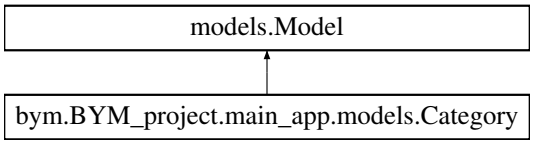
The documentation for this class was generated from the following file:

- cart.py

6.2 bym.BYM_project.main_app.models.Category Class Reference

категории различных продуктов для приготовления
name – название категории (например: бакалея или помидоры)
supreme_category_id – надкатегория (например надкатегорией римских помидор являются помидоры)

Inheritance diagram for bym.BYM_project.main_app.models.Category:



Classes

- class Meta

Static Public Attributes

- supreme_category
- null
- True
- blank
- on_delete
- name = models.CharField(max_length=70)

6.2.1 Detailed Description

категории различных продуктов для приготовления
name – название категории (например: бакалея или помидоры)
supreme_category_id – надкатегория (например надкатегорией римских помидор являются помидоры)

Definition at line 140 of file models.py.

6.2.2 Member Data Documentation

6.2.2.1 blank

bym.BYM_project.main_app.models.Category.blank [static]

Definition at line 150 of file models.py.

6.2.2.2 name

bym.BYM_project.main_app.models.Category.name = models.CharField(max_length=70) [static]

Definition at line 152 of file models.py.

6.2.2.3 null

bym.BYM_project.main_app.models.Category.null [static]

Definition at line 150 of file models.py.

6.2.2.4 on_delete

bym.BYM_project.main_app.models.Category.on_delete [static]

Definition at line 150 of file models.py.

6.2.2.5 supreme_category

bym.BYM_project.main_app.models.Category.supreme_category [static]

Definition at line 150 of file models.py.

6.2.2.6 True

bym.BYM_project.main_app.models.Category.True [static]

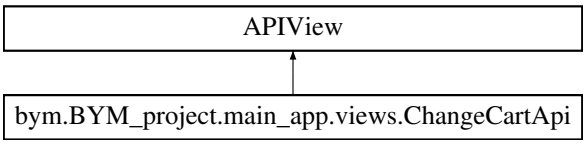
Definition at line 150 of file models.py.

The documentation for this class was generated from the following file:

- models.py

6.3 bym.BYM_project.main_app.views.ChangeCartApi Class Reference

Inheritance diagram for bym.BYM_project.main_app.views.ChangeCartApi:



Public Member Functions

- post (self, request)

6.3.1 Detailed Description

Definition at line 160 of file views.py.

6.3.2 Member Function Documentation

6.3.2.1 post()

bym.BYM_project.main_app.views.ChangeCartApi.post (
self,
request)

Definition at line 161 of file views.py.

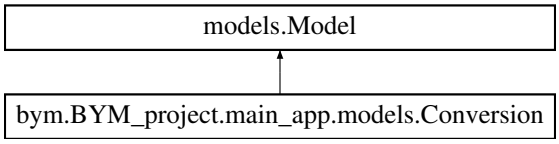
The documentation for this class was generated from the following file:

- views.py

6.4 bym.BYM_project.main_app.models.Conversion Class Reference

коэффициенты для перевода единиц измерения
coefficient – число на которое нужно умножить что бы перевести from_unit_id в to_unit_id
category_id – категория для которой выполняется перевод (очевидно что коэффициент для перевода штук в граммы для помидоров и огурцов отличается)
from_unit_id– единица измерения из которой выполняется перевод
to_unit_id – единица измерения в которую выполняется перевод

Inheritance diagram for bym.BYM_project.main_app.models.Conversion:



Classes

- class Meta

Static Public Attributes

- category
- Category
- on_delete
- from_unit = models.ForeignKey(UnitOfMeasure, related_name='from_unit', on_↩ delete=models.CASCADE)
- to_unit = models.ForeignKey(UnitOfMeasure, related_name='to_unit', null=True, blank=True, on_delete=models.CASCADE)
- coefficient = models.DecimalField(max_digits=4 + 4, decimal_places=4, default=1)

6.4.1 Detailed Description

коэффициенты для перевода единиц измерения
coefficient – число на которое нужно умножить что бы перевести from_unit_id в to_unit_id
category_id – категория для которой выполняется перевод (очевидно что коэффициент для перевода штук в граммы для помидоров и огурцов отличается)
from_unit_id– единица измерения из которой выполняется перевод
to_unit_id – единица измерения в которую выполняется перевод

Definition at line 170 of file models.py.

6.4.2 Member Data Documentation

6.4.2.1 category

bym.BYM_project.main_app.models.Conversion.category [static]

Definition at line 182 of file models.py.

6.4.2.2 Category

bym.BYM_project.main_app.models.Conversion.Category [static]

Definition at line 182 of file models.py.

6.4.2.3 coefficient

bym.BYM_project.main_app.models.Conversion.coefficient = models.DecimalField(max_digits=4 + 4, decimal_↩ places=4, default=1) [static]

Definition at line 186 of file models.py.

6.4.2.4 from_unit

bym.BYM_project.main_app.models.Conversion.from_unit = models.ForeignKey(UnitOfMeasure, related_↩ name='from_unit', on_delete=models.CASCADE) [static]

Definition at line 183 of file models.py.

6.4.2.5 on_delete

bym.BYM_project.main_app.models.Conversion.on_delete [static]

Definition at line 182 of file models.py.

6.4.2.6 to_unit

bym.BYM_project.main_app.models.Conversion.to_unit = models.ForeignKey(UnitOfMeasure, related_name='to_↩ unit', null=True, blank=True, on_delete=models.CASCADE) [static]

Definition at line 184 of file models.py.

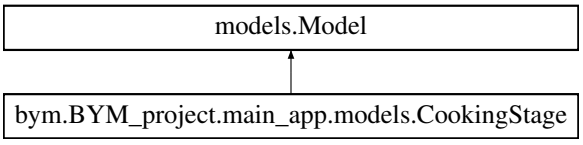
The documentation for this class was generated from the following file:

- models.py

6.5 bym.BYM_project.main_app.models.CookingStage Class Reference

шаги для приготовления блюда
description – текст того, что нужно сделать на этом шаге
order – порядковый номер шага
dish_id – блюдо, к которому относится этот шаг
image – ссылка на изображение шага

Inheritance diagram for bym.BYM_project.main_app.models.CookingStage:



Classes

- class Meta

Static Public Attributes

- dish
- Dish
- on_delete
- description = models.TextField()
- order = models.PositiveIntegerField()
- image = models.ImageField(null=True, blank=True, max_length=130)

6.5.1 Detailed Description

шаги для приготовления блюда
description – текст того, что нужно сделать на этом шаге
order – порядковый номер шага
dish_id – блюдо, к которому относится этот шаг
image – ссылка на изображение шага

Definition at line 121 of file models.py.

6.5.2 Member Data Documentation

6.5.2.1 description

bym.BYM_project.main_app.models.CookingStage.description = models.TextField() [static]

Definition at line 135 of file models.py.

6.5.2.2 dish

bym.BYM_project.main_app.models.CookingStage.dish [static]

Definition at line 133 of file models.py.

6.5.2.3 Dish

bym.BYM_project.main_app.models.CookingStage.Dish [static]

Definition at line 133 of file models.py.

6.5.2.4 image

bym.BYM_project.main_app.models.CookingStage.image = models.ImageField(null=True, blank=True, max_↩ length=130) [static]

Definition at line 137 of file models.py.

6.5.2.5 on_delete

bym.BYM_project.main_app.models.CookingStage.on_delete [static]

Definition at line 133 of file models.py.

6.5.2.6 order

bym.BYM_project.main_app.models.CookingStage.order = models.PositiveIntegerField() [static]

Definition at line 136 of file models.py.

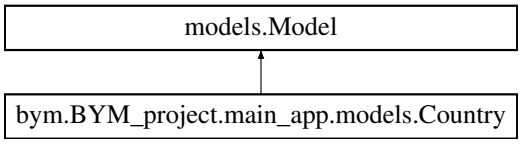
The documentation for this class was generated from the following file:

- models.py

6.6 bym.BYM_project.main_app.models.Country Class Reference

Таблица с кухнями различных стран
name – название кухни (например Русская кухня)

Inheritance diagram for bym.BYM_project.main_app.models.Country:



Classes

- class Meta

Static Public Attributes

- name
- max_length

6.6.1 Detailed Description

Таблица с кухнями различных стран
name – название кухни (например Русская кухня)

Definition at line 51 of file models.py.

6.6.2 Member Data Documentation

6.6.2.1 max_length

bym.BYM_project.main_app.models.Country.max_length [static]

Definition at line 60 of file models.py.

6.6.2.2 name

bym.BYM_project.main_app.models.Country.name [static]

Definition at line 60 of file models.py.

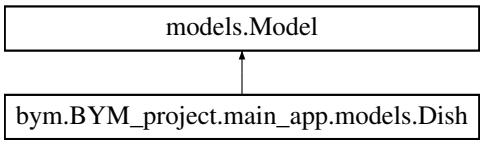
The documentation for this class was generated from the following file:

- models.py

6.7 bym.BYM_project.main_app.models.Dish Class Reference

информация о блюде
name – названия блюда
pers_num – кол-во порций
cooking_time_minutes – время готовки в минутах
calories – кол-во калорий в блюде
proteins – кол0белки в блюде
carbohydrates – углеводы в блюде
fats – жиры
country_id – страна из кухни которой это блюдо (например итальянская кухня)
subtype_id – подтип блюда (например, фруктовые десерты)
image – ссылка на изображения блюда

Inheritance diagram for bym.BYM_project.main_app.models.Dish:



Classes

- class Meta

Static Public Attributes

- country
- Country
- on_delete
- subtype = models.ForeignKey(SubType, on_delete=models.CASCADE)
- name = models.CharField(max_length=100)
- pers_num = models.PositiveIntegerField()
- cooking_time_minutes = models.PositiveIntegerField()
- image = models.ImageField(null=True, blank=True, max_length=130)
- calories = models.PositiveIntegerField()
- proteins = models.PositiveIntegerField()
- carbohydrates = models.PositiveIntegerField()
- fats = models.PositiveIntegerField()

6.7.1 Detailed Description

информация о блюде
name – названия блюда
pers_num – кол-во порций
cooking_time_minutes – время готовки в минутах
calories – кол-во калорий в блюде
proteins – кол0белки в блюде
carbohydrates – углеводы в блюде
fats – жиры
country_id – страна из кухни которой это блюдо (например итальянская кухня)
subtype_id – подтип блюда (например, фруктовые десерты)
image – ссылка на изображения блюда

Definition at line 90 of file models.py.

6.7.2 Member Data Documentation

6.7.2.1 calories

bym.BYM_project.main_app.models.Dish.calories = models.PositiveIntegerField() [static]

Definition at line 115 of file models.py.

6.7.2.2 carbohydrates

bym.BYM_project.main_app.models.Dish.carbohydrates = models.PositiveIntegerField() [static]

Definition at line 117 of file models.py.

6.7.2.3 cooking_time_minutes

bym.BYM_project.main_app.models.Dish.cooking_time_minutes = models.PositiveIntegerField() [static]

Definition at line 113 of file models.py.

6.7.2.4 country

bym.BYM_project.main_app.models.Dish.country [static]

Definition at line 108 of file models.py.

6.7.2.5 Country

bym.BYM_project.main_app.models.Dish.Country [static]

Definition at line 108 of file models.py.

6.7.2.6 fats

bym.BYM_project.main_app.models.Dish.fats = models.PositiveIntegerField() [static]

Definition at line 118 of file models.py.

6.7.2.7 image

bym.BYM_project.main_app.models.Dish.image = models.ImageField(null=True, blank=True, max_length=130) [static]

Definition at line 114 of file models.py.

6.7.2.8 name

bym.BYM_project.main_app.models.Dish.name = models.CharField(max_length=100) [static]

Definition at line 111 of file models.py.

6.7.2.9 on_delete

bym.BYM_project.main_app.models.Dish.on_delete [static]

Definition at line 108 of file models.py.

6.7.2.10 pers_num

bym.BYM_project.main_app.models.Dish.pers_num = models.PositiveIntegerField() [static]

Definition at line 112 of file models.py.

6.7.2.11 proteins

bym.BYM_project.main_app.models.Dish.proteins = models.PositiveIntegerField() [static]

Definition at line 116 of file models.py.

6.7.2.12 subtype

bym.BYM_project.main_app.models.Dish.subtype = models.ForeignKey(SubType, on_delete=models.CASCADE) [static]

Definition at line 109 of file models.py.

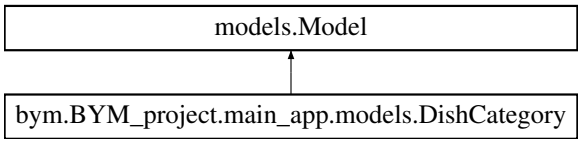
The documentation for this class was generated from the following file:

- models.py

6.8 bym.BYM_project.main_app.models.DishCategory Class Reference

информация о составе блюда.

Inheritance diagram for bym.BYM_project.main_app.models.DishCategory:



Classes

- class Meta

Static Public Attributes

- dish
- Dish
- on_delete
- category = models.ForeignKey(Category, on_delete=models.CASCADE)
- unit = models.ForeignKey(UnitOfMeasure, null=True, blank=True, on_delete=models.↩ CASCADE)
- amount = models.DecimalField(max_digits=4 + 4, decimal_places=4)

6.8.1 Detailed Description

информация о составе блюда.

amount – кол-во продукта в блюде
category_id – категория продукта
dish_id – блюдо к которому относится продукт
unit_id – единицы измерения в которых указано кол-во

Definition at line 189 of file models.py.

6.8.2 Member Data Documentation

6.8.2.1 amount

bym.BYM_project.main_app.models.DishCategory.amount = models.DecimalField(max_digits=4 + 4, decimal_↩ places=4) [static]

Definition at line 205 of file models.py.

6.8.2.2 category

bym.BYM_project.main_app.models.DishCategory.category = models.ForeignKey(Category, on_delete=models.↩ CASCADE) [static]

Definition at line 202 of file models.py.

6.8.2.3 dish

bym.BYM_project.main_app.models.DishCategory.dish [static]

Definition at line 201 of file models.py.

6.8.2.4 Dish

bym.BYM_project.main_app.models.DishCategory.Dish [static]

Definition at line 201 of file models.py.

6.8.2.5 on_delete

bym.BYM_project.main_app.models.DishCategory.on_delete [static]

Definition at line 201 of file models.py.

6.8.2.6 unit

bym.BYM_project.main_app.models.DishCategory.unit = models.ForeignKey(UnitOfMeasure, null=True, blank=True, on_delete=models.CASCADE) [static]

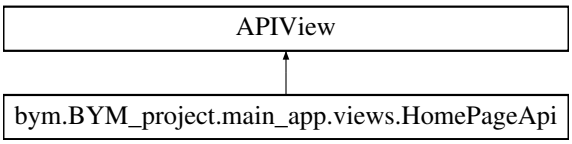
Definition at line 203 of file models.py.

The documentation for this class was generated from the following file:

- models.py

6.9 bym.BYM_project.main_app.views.HomePageApi Class Reference

Inheritance diagram for bym.BYM_project.main_app.views.HomePageApi:



Public Member Functions

- get (self, request)

6.9.1 Detailed Description

Definition at line 125 of file views.py.

6.9.2 Member Function Documentation

6.9.2.1 get()

bym.BYM_project.main_app.views.HomePageApi.get (
self,
request)

Definition at line 126 of file views.py.

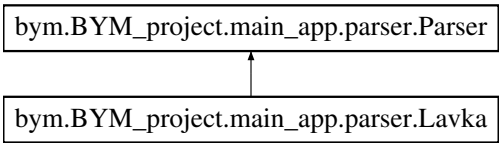
The documentation for this class was generated from the following file:

- views.py

6.10 bym.BYM_project.main_app.parser.Lavka Class Reference

Определяет класс используемый для парсинга лавки.

Inheritance diagram for bym.BYM_project.main_app.parser.Lavka:



Public Member Functions

- __init__ (self, user_id)

  Инициализирует класс Lavka.
- get_data_product (self, product)

  Выделяет из блока с информацией о продукте нужную и записывает в БД
- get_data_html (self, html)

  Разбивает html код на блоки по продуктам, асинхронно запускает их обработку
- pars (self, city_street, home)

  Парсит список продуктов с ценами яндекс лавки по определенному адресу Производит запись в базу данных

Public Member Functions inherited from bym.BYM_project.main_app.parser.Parser

- sql_script (self)

  Сохраняет в базу данных

Public Attributes

- url_main
- driver
- wait
- urls
- sl_tags

Public Attributes inherited from bym.BYM_project.main_app.parser.Parser

- user_obj
- options
- product_list
- shop_obj

6.10.1 Detailed Description

Определяет класс используемый для парсинга лавки.

Definition at line 56 of file parser.py.

6.10.2 Constructor & Destructor Documentation

6.10.2.1 __init__()

bym.BYM_project.main_app.parser.Lavka.__init__ (
self,
user_id)

Инициализирует класс Lavka.

Parameters

user↩ _id	int

Reimplemented from bym.BYM_project.main_app.parser.Parser (p. 55).

Definition at line 61 of file parser.py.

6.10.3 Member Function Documentation

6.10.3.1 get_data_html()

bym.BYM_project.main_app.parser.Lavka.get_data_html (
self,
html)

Разбивает html код на блоки по продуктам, асинхронно запускает их обработку

Parameters

html	string

Definition at line 107 of file parser.py.

6.10.3.2 get_data_product()

bym.BYM_project.main_app.parser.Lavka.get_data_product (
self,
product)

Выделяет из блока с информацией о продукте нужную и записывает в БД

Parameters

product	BeautifulSoup

Definition at line 79 of file parser.py.

6.10.3.3 pars()

bym.BYM_project.main_app.parser.Lavka.pars (
self,
city_street,
home)

Парсит список продуктов с ценами яндекс лавки по определенному адресу Производит запись в базу данных

Parameters

city_strit	string
home	string

Definition at line 123 of file parser.py.

6.10.4 Member Data Documentation

6.10.4.1 driver

bym.BYM_project.main_app.parser.Lavka.driver

Definition at line 73 of file parser.py.

6.10.4.2 sl_tags

bym.BYM_project.main_app.parser.Lavka.sl_tags

Definition at line 76 of file parser.py.

6.10.4.3 url_main

bym.BYM_project.main_app.parser.Lavka.url_main

Definition at line 67 of file parser.py.

6.10.4.4 urls

bym.BYM_project.main_app.parser.Lavka.urls

Definition at line 75 of file parser.py.

6.10.4.5 wait

bym.BYM_project.main_app.parser.Lavka.wait

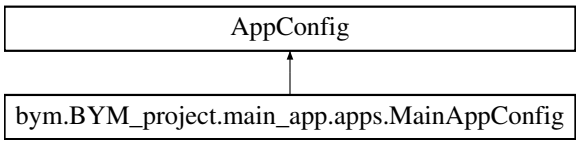
Definition at line 74 of file parser.py.

The documentation for this class was generated from the following file:

- parser.py

6.11 bym.BYM_project.main_app.apps.MainAppConfig Class Reference

Inheritance diagram for bym.BYM_project.main_app.apps.MainAppConfig:



Static Public Attributes

- str default_auto_field = 'django.db.models.BigAutoField'
- str name = 'main_app'

6.11.1 Detailed Description

Definition at line 4 of file apps.py.

6.11.2 Member Data Documentation

6.11.2.1 default_auto_field

str bym.BYM_project.main_app.apps.MainAppConfig.default_auto_field = 'django.db.models.BigAutoField' [static]

Definition at line 5 of file apps.py.

6.11.2.2 name

str bym.BYM_project.main_app.apps.MainAppConfig.name = 'main_app' [static]

Definition at line 6 of file apps.py.

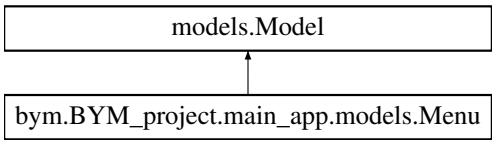
The documentation for this class was generated from the following file:

- apps.py

6.12 bym.BYM_project.main_app.models.Menu Class Reference

состав заказа
amount– кол-во товара, которое было заказано
dish_id – товар который был заказан
order_id – заказ в котором этот товар был заказан

Inheritance diagram for bym.BYM_project.main_app.models.Menu:



Classes

- class Meta

Static Public Attributes

- dish
- Dish
- on_delete
- order = models.ForeignKey(Order, on_delete=models.CASCADE)
- amount = models.PositiveIntegerField(default=1)

6.12.1 Detailed Description

состав заказа
amount– кол-во товара, которое было заказано
dish_id – товар который был заказан
order_id – заказ в котором этот товар был заказан

Definition at line 225 of file models.py.

6.12.2 Member Data Documentation

6.12.2.1 amount

bym.BYM_project.main_app.models.Menu.amount = models.PositiveIntegerField(default=1) [static]

Definition at line 239 of file models.py.

6.12.2.2 dish

bym.BYM_project.main_app.models.Menu.dish [static]

Definition at line 236 of file models.py.

6.12.2.3 Dish

bym.BYM_project.main_app.models.Menu.Dish [static]

Definition at line 236 of file models.py.

6.12.2.4 on_delete

bym.BYM_project.main_app.models.Menu.on_delete [static]

Definition at line 236 of file models.py.

6.12.2.5 order

bym.BYM_project.main_app.models.Menu.order = models.ForeignKey(Order, on_delete=models.CASCADE) [static]

Definition at line 237 of file models.py.

The documentation for this class was generated from the following file:

- models.py

6.13 bym.BYM_project.main_app.models.Category.Meta Class Reference

Static Public Attributes

- str verbose_name = 'Категория продукта'
- str verbose_name_plural = 'Категории продуктов'

6.13.1 Detailed Description

Definition at line 146 of file models.py.

6.13.2 Member Data Documentation

6.13.2.1 verbose_name

str bym.BYM_project.main_app.models.Category.Meta.verbose_name = 'Категория продукта' [static]

Definition at line 147 of file models.py.

6.13.2.2 verbose_name_plural

str bym.BYM_project.main_app.models.Category.Meta.verbose_name_plural = 'Категории продуктов' [static]

Definition at line 148 of file models.py.

The documentation for this class was generated from the following file:

- models.py

6.14 bym.BYM_project.main_app.models.Conversion.Meta Class Reference

Static Public Attributes

- str verbose_name = 'Преобразование единицы измерения'
- str verbose_name_plural = 'Преобразования единиц измерения'

6.14.1 Detailed Description

Definition at line 178 of file models.py.

6.14.2 Member Data Documentation

6.14.2.1 verbose_name

str bym.BYM_project.main_app.models.Conversion.Meta.verbose_name = 'Преобразование единицы измерения' [static]

Definition at line 179 of file models.py.

6.14.2.2 verbose_name_plural

str bym.BYM_project.main_app.models.Conversion.Meta.verbose_name_plural = 'Преобразования единиц измерения' [static]

Definition at line 180 of file models.py.

The documentation for this class was generated from the following file:

- models.py

6.15 bym.BYM_project.main_app.models.CookingStage.Meta Class Reference

Static Public Attributes

- str verbose_name = 'Этап приготовления блюда'
- str verbose_name_plural = 'Этапы приготовления блюд'

6.15.1 Detailed Description

Definition at line 129 of file models.py.

6.15.2 Member Data Documentation

6.15.2.1 verbose_name

str bym.BYM_project.main_app.models.CookingStage.Meta.verbose_name = 'Этап приготовления блюда' [static]

Definition at line 130 of file models.py.

6.15.2.2 verbose_name_plural

str bym.BYM_project.main_app.models.CookingStage.Meta.verbose_name_plural = 'Этапы приготовления блюд' [static]

Definition at line 131 of file models.py.

The documentation for this class was generated from the following file:

- models.py

6.16 bym.BYM_project.main_app.models.Country.Meta Class Reference

Static Public Attributes

- str verbose_name = 'Страна'
- str verbose_name_plural = 'Страны'

6.16.1 Detailed Description

Definition at line 56 of file models.py.

6.16.2 Member Data Documentation

6.16.2.1 verbose_name

str bym.BYM_project.main_app.models.Country.Meta.verbose_name = 'Страна' [static]

Definition at line 57 of file models.py.

6.16.2.2 verbose_name_plural

str bym.BYM_project.main_app.models.Country.Meta.verbose_name_plural = 'Страны' [static]

Definition at line 58 of file models.py.

The documentation for this class was generated from the following file:

- models.py

6.17 bym.BYM_project.main_app.models.Dish.Meta Class Reference

Static Public Attributes

- str verbose_name = 'Блюдо'
- str verbose_name_plural = 'Блюда'

6.17.1 Detailed Description

Definition at line 104 of file models.py.

6.17.2 Member Data Documentation

6.17.2.1 verbose_name

str bym.BYM_project.main_app.models.Dish.Meta.verbose_name = 'Блюдо' [static]

Definition at line 105 of file models.py.

6.17.2.2 verbose_name_plural

str bym.BYM_project.main_app.models.Dish.Meta.verbose_name_plural = 'Блюда' [static]

Definition at line 106 of file models.py.

The documentation for this class was generated from the following file:

- models.py

6.18 bym.BYM_project.main_app.models.DishCategory.Meta Class Reference

Static Public Attributes

- str verbose_name = 'Продукт в блюде'
- str verbose_name_plural = 'Продукты в блюдах'

6.18.1 Detailed Description

Definition at line 197 of file models.py.

6.18.2 Member Data Documentation

6.18.2.1 verbose_name

str bym.BYM_project.main_app.models.DishCategory.Meta.verbose_name = 'Продукт в блюде' [static]

Definition at line 198 of file models.py.

6.18.2.2 verbose_name_plural

str bym.BYM_project.main_app.models.DishCategory.Meta.verbose_name_plural = 'Продукты в блюдах' [static]

Definition at line 199 of file models.py.

The documentation for this class was generated from the following file:

- models.py

6.19 bym.BYM_project.main_app.models.Menu.Meta Class Reference

Static Public Attributes

- str verbose_name = 'Меню'
- str verbose_name_plural = 'Меню'

6.19.1 Detailed Description

Definition at line 232 of file models.py.

6.19.2 Member Data Documentation

6.19.2.1 verbose_name

str bym.BYM_project.main_app.models.Menu.Meta.verbose_name = 'Меню' [static]

Definition at line 233 of file models.py.

6.19.2.2 verbose_name_plural

str bym.BYM_project.main_app.models.Menu.Meta.verbose_name_plural = 'Меню' [static]

Definition at line 234 of file models.py.

The documentation for this class was generated from the following file:

- models.py

6.20 bym.BYM_project.main_app.models.Order.Meta Class Reference

Static Public Attributes

- str verbose_name = 'Заказ'
- str verbose_name_plural = 'Заказы'

6.20.1 Detailed Description

Definition at line 215 of file models.py.

6.20.2 Member Data Documentation

6.20.2.1 verbose_name

str bym.BYM_project.main_app.models.Order.Meta.verbose_name = 'Заказ' [static]

Definition at line 216 of file models.py.

6.20.2.2 verbose_name_plural

str bym.BYM_project.main_app.models.Order.Meta.verbose_name_plural = 'Заказы' [static]

Definition at line 217 of file models.py.

The documentation for this class was generated from the following file:

- models.py

6.21 bym.BYM_project.main_app.models.Product.Meta Class Reference

Static Public Attributes

- str verbose_name = 'Продукт из магазина'
- str verbose_name_plural = 'Продукты из магазинов'

6.21.1 Detailed Description

Definition at line 267 of file models.py.

6.21.2 Member Data Documentation

6.21.2.1 verbose_name

str bym.BYM_project.main_app.models.Product.Meta.verbose_name = 'Продукт из магазина' [static]

Definition at line 268 of file models.py.

6.21.2.2 verbose_name_plural

str bym.BYM_project.main_app.models.Product.Meta.verbose_name_plural = 'Продукты из магазинов' [static]

Definition at line 269 of file models.py.

The documentation for this class was generated from the following file:

- models.py

6.22 bym.BYM_project.main_app.models.ProductLog.Meta Class Reference

Static Public Attributes

- str verbose_name = 'Лог родукта из магазина'
- str verbose_name_plural = 'Логи родуктов из магазинов'

6.22.1 Detailed Description

Definition at line 293 of file models.py.

6.22.2 Member Data Documentation

6.22.2.1 verbose_name

str bym.BYM_project.main_app.models.ProductLog.Meta.verbose_name = 'Лог родукта из магазина' [static]

Definition at line 294 of file models.py.

6.22.2.2 verbose_name_plural

str bym.BYM_project.main_app.models.ProductLog.Meta.verbose_name_plural = 'Логи родуктов из магазинов' [static]

Definition at line 295 of file models.py.

The documentation for this class was generated from the following file:

- models.py

6.23 bym.BYM_project.main_app.models.Shop.Meta Class Reference

Static Public Attributes

- str verbose_name = 'Магазин'
- str verbose_name_plural = 'Магазины'

6.23.1 Detailed Description

Definition at line 248 of file models.py.

6.23.2 Member Data Documentation

6.23.2.1 verbose_name

str bym.BYM_project.main_app.models.Shop.Meta.verbose_name = 'Магазин' [static]

Definition at line 249 of file models.py.

6.23.2.2 verbose_name_plural

str bym.BYM_project.main_app.models.Shop.Meta.verbose_name_plural = 'Магазины' [static]

Definition at line 250 of file models.py.

The documentation for this class was generated from the following file:

- models.py

6.24 bym.BYM_project.main_app.models.SubType.Meta Class Reference

Static Public Attributes

- str verbose_name = 'Подтип блюда'
- str verbose_name_plural = 'Подтипы блюд'

6.24.1 Detailed Description

Definition at line 81 of file models.py.

6.24.2 Member Data Documentation

6.24.2.1 verbose_name

str bym.BYM_project.main_app.models.SubType.Meta.verbose_name = 'Подтип блюда' [static]

Definition at line 82 of file models.py.

6.24.2.2 verbose_name_plural

str bym.BYM_project.main_app.models.SubType.Meta.verbose_name_plural = 'Подтипы блюд' [static]

Definition at line 83 of file models.py.

The documentation for this class was generated from the following file:

- models.py

6.25 bym.BYM_project.main_app.models.Type.Meta Class Reference

Static Public Attributes

- str verbose_name = 'Тип бдюда'
- str verbose_name_plural = 'Типы блюд'

6.25.1 Detailed Description

Definition at line 68 of file models.py.

6.25.2 Member Data Documentation

6.25.2.1 verbose_name

str bym.BYM_project.main_app.models.Type.Meta.verbose_name = 'Тип бдюда' [static]

Definition at line 69 of file models.py.

6.25.2.2 verbose_name_plural

str bym.BYM_project.main_app.models.Type.Meta.verbose_name_plural = 'Типы блюд' [static]

Definition at line 70 of file models.py.

The documentation for this class was generated from the following file:

- models.py

6.26 bym.BYM_project.main_app.models.UnitOfMeasure.Meta Class Reference

Static Public Attributes

- str verbose_name = 'Единица измерения'
- str verbose_name_plural = 'Единицы измерения'

6.26.1 Detailed Description

Definition at line 161 of file models.py.

6.26.2 Member Data Documentation

6.26.2.1 verbose_name

str bym.BYM_project.main_app.models.UnitOfMeasure.Meta.verbose_name = 'Единица измерения' [static]

Definition at line 162 of file models.py.

6.26.2.2 verbose_name_plural

str bym.BYM_project.main_app.models.UnitOfMeasure.Meta.verbose_name_plural = 'Единицы измерения' [static]

Definition at line 163 of file models.py.

The documentation for this class was generated from the following file:

- models.py

6.27 bym.BYM_project.main_app.models.User.Meta Class Reference

Static Public Attributes

- str verbose_name = 'Пользователь'
- str verbose_name_plural = 'Пользователи'

6.27.1 Detailed Description

Definition at line 13 of file models.py.

6.27.2 Member Data Documentation

6.27.2.1 verbose_name

str bym.BYM_project.main_app.models.User.Meta.verbose_name = 'Пользователь' [static]

Definition at line 14 of file models.py.

6.27.2.2 verbose_name_plural

str bym.BYM_project.main_app.models.User.Meta.verbose_name_plural = 'Пользователи' [static]

Definition at line 15 of file models.py.

The documentation for this class was generated from the following file:

- models.py

6.28 bym.BYM_project.main_app.serializers.UserSerializer.Meta Class Reference

Static Public Attributes

- model = User
- str fields = '__all__'

6.28.1 Detailed Description

Definition at line 6 of file serializers.py.

6.28.2 Member Data Documentation

6.28.2.1 fields

str bym.BYM_project.main_app.serializers.UserSerializer.Meta.fields = '__all__' [static]

Definition at line 8 of file serializers.py.

6.28.2.2 model

bym.BYM_project.main_app.serializers.UserSerializer.Meta.model = User [static]

Definition at line 7 of file serializers.py.

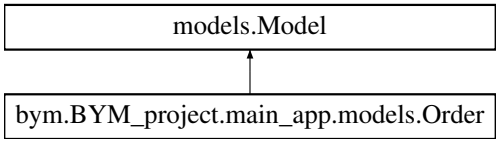
The documentation for this class was generated from the following file:

- serializers.py

6.29 bym.BYM_project.main_app.models.Order Class Reference

информация о заказах.

Inheritance diagram for bym.BYM_project.main_app.models.Order:



Classes

- class Meta

Static Public Attributes

- user
- User
- on_delete
- order_date = models.DateTimeField()
- current = models.BooleanField(default=True)

6.29.1 Detailed Description

информация о заказах.

order_date – дата заказа
current – статус (выполнен/ не выполнен)
user_id – пользователь совершивший заказ

Definition at line 208 of file models.py.

6.29.2 Member Data Documentation

6.29.2.1 current

bym.BYM_project.main_app.models.Order.current = models.BooleanField(default=True) [static]

Definition at line 222 of file models.py.

6.29.2.2 on_delete

bym.BYM_project.main_app.models.Order.on_delete [static]

Definition at line 219 of file models.py.

6.29.2.3 order_date

bym.BYM_project.main_app.models.Order.order_date = models.DateTimeField() [static]

Definition at line 221 of file models.py.

6.29.2.4 user

bym.BYM_project.main_app.models.Order.user [static]

Definition at line 219 of file models.py.

6.29.2.5 User

bym.BYM_project.main_app.models.Order.User [static]

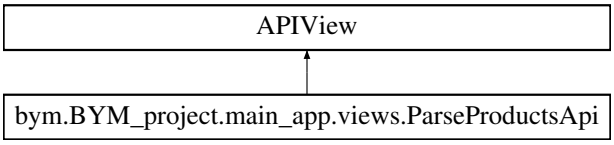
Definition at line 219 of file models.py.

The documentation for this class was generated from the following file:

- models.py

6.30 bym.BYM_project.main_app.views.ParseProductsApi Class Reference

Inheritance diagram for bym.BYM_project.main_app.views.ParseProductsApi:



Public Member Functions

- post (self, request)

6.30.1 Detailed Description

Definition at line 178 of file views.py.

6.30.2 Member Function Documentation

6.30.2.1 post()

bym.BYM_project.main_app.views.ParseProductsApi.post (
self,
request)

Definition at line 179 of file views.py.

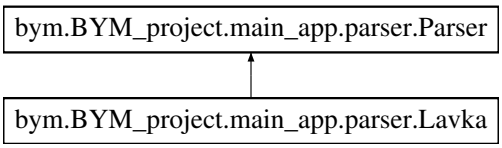
The documentation for this class was generated from the following file:

- views.py

6.31 bym.BYM_project.main_app.parser.Parser Class Reference

Определяет базавый класс используемый для парсинга всех сайтов

Inheritance diagram for bym.BYM_project.main_app.parser.Parser:



Public Member Functions

- __init__ (self, user_id, shop_id)
  Инициализирует класс Parser.
- sql_script (self)
  Сохраняет в базу данных

Public Attributes

- user_obj
- options
- product_list
- shop_obj

6.31.1 Detailed Description

Определяет базавый класс используемый для парсинга всех сайтов

Definition at line 23 of file parser.py.

6.31.2 Constructor & Destructor Documentation

6.31.2.1 __init__()

bym.BYM_project.main_app.parser.Parser.__init__ (
self,
user_id,
shop_id)

Инициализирует класс Parser.

Parameters

user_id	
shop↩ _id	

Reimplemented in bym.BYM_project.main_app.parser.Lavka (p. 37).

Definition at line 28 of file parser.py.

6.31.3 Member Function Documentation

6.31.3.1 sql_script()

bym.BYM_project.main_app.parser.Parser.sql_script (
self)

Сохраняет в базу данных

Definition at line 39 of file parser.py.

6.31.4 Member Data Documentation

6.31.4.1 options

bym.BYM_project.main_app.parser.Parser.options

Definition at line 35 of file parser.py.

6.31.4.2 product_list

bym.BYM_project.main_app.parser.Parser.product_list

Definition at line 36 of file parser.py.

6.31.4.3 shop_obj

bym.BYM_project.main_app.parser.Parser.shop_obj

Definition at line 37 of file parser.py.

6.31.4.4 user_obj

bym.BYM_project.main_app.parser.Parser.user_obj

Definition at line 34 of file parser.py.

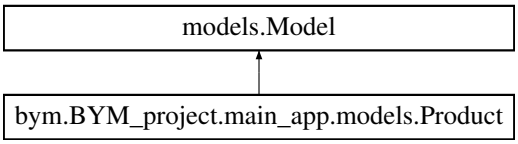
The documentation for this class was generated from the following file:

- parser.py

6.32 bym.BYM_project.main_app.models.Product Class Reference

информация о конкретном товаре в магазине
name – название товара
price – цена
amount – количество (в unit_id) товара которое в одной единице заказ (например 300, unit_id – граммы)
link – ссылка на товар
shop_id – магазин, из которого товар
unit_id – единица измерения
user_id – пользователь для которого был спаршен данный товар

Inheritance diagram for bym.BYM_project.main_app.models.Product:



Classes

- class Meta

Static Public Attributes

- user
- User
- on_delete
- unit = models.ForeignKey(UnitOfMeasure, on_delete=models.CASCADE)
- shop = models.ForeignKey(Shop, on_delete=models.CASCADE)
- name = models.CharField(max_length=100)
- price = models.PositiveIntegerField()
- amount = models.DecimalField(max_digits=4 + 4, decimal_places=4)
- link = models.URLField()

6.32.1 Detailed Description

информация о конкретном товаре в магазине
name – название товара
price – цена
amount – количество (в unit_id) товара которое в одной единице заказ (например 300, unit_id – граммы)
link – ссылка на товар
shop_id – магазин, из которого товар
unit_id – единица измерения
user_id – пользователь для которого был спаршен данный товар

Definition at line 256 of file models.py.

6.32.2 Member Data Documentation

6.32.2.1 amount

bym.BYM_project.main_app.models.Product.amount = models.DecimalField(max_digits=4 + 4, decimal_places=4) [static]

Definition at line 277 of file models.py.

6.32.2.2 link

bym.BYM_project.main_app.models.Product.link = models.URLField() [static]

Definition at line 278 of file models.py.

6.32.2.3 name

bym.BYM_project.main_app.models.Product.name = models.CharField(max_length=100) [static]

Definition at line 275 of file models.py.

6.32.2.4 on_delete

bym.BYM_project.main_app.models.Product.on_delete [static]

Definition at line 271 of file models.py.

6.32.2.5 price

bym.BYM_project.main_app.models.Product.price = models.PositiveIntegerField() [static]

Definition at line 276 of file models.py.

6.32.2.6 shop

bym.BYM_project.main_app.models.Product.shop = models.ForeignKey(Shop, on_delete=models.CASCADE) [static]

Definition at line 273 of file models.py.

6.32.2.7 unit

bym.BYM_project.main_app.models.Product.unit = models.ForeignKey(UnitOfMeasure, on_delete=models.↩ CASCADE) [static]

Definition at line 272 of file models.py.

6.32.2.8 user

bym.BYM_project.main_app.models.Product.user [static]

Definition at line 271 of file models.py.

6.32.2.9 User

bym.BYM_project.main_app.models.Product.User [static]

Definition at line 271 of file models.py.

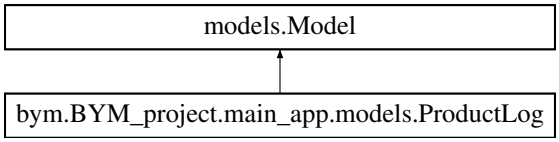
The documentation for this class was generated from the following file:

- models.py

6.33 bym.BYM_project.main_app.models.ProductLog Class Reference

информация о ранее заказанных товарах в магазине
name – название товара
price – цена
amount – количество (в unit_id) товара которое в одной единице заказ (например 300, unit_id – граммы)
quantity – количество единиц, которое было заказано
link – ссылка на товар
shop_id – магазин, из которого товар
order_id – заказ, в котором был куплен данный товар
unit_id – единица измерения

Inheritance diagram for bym.BYM_project.main_app.models.ProductLog:



Classes

- class Meta

Static Public Attributes

- order
- Order
- on_delete
- shop = models.ForeignKey(Shop, on_delete=models.CASCADE)
- unit = models.ForeignKey(UnitOfMeasure, on_delete=models.CASCADE)
- name = models.CharField(max_length=100)
- price = models.PositiveIntegerField()
- amount = models.DecimalField(max_digits=4 + 4, decimal_places=4)
- quantity = models.PositiveIntegerField()
- link = models.URLField()

6.33.1 Detailed Description

информация о ранее заказанных товарах в магазине
name – название товара
price – цена
amount – количество (в unit_id) товара которое в одной единице заказ (например 300, unit_id – граммы)
quantity – количество единиц, которое было заказано
link – ссылка на товар
shop_id – магазин, из которого товар
order_id – заказ, в котором был куплен данный товар
unit_id – единица измерения

Definition at line 281 of file models.py.

6.33.2 Member Data Documentation

6.33.2.1 amount

bym.BYM_project.main_app.models.ProductLog.amount = models.DecimalField(max_digits=4 + 4, decimal_places=4) [static]

Definition at line 303 of file models.py.

6.33.2.2 link

bym.BYM_project.main_app.models.ProductLog.link = models.URLField() [static]

Definition at line 305 of file models.py.

6.33.2.3 name

bym.BYM_project.main_app.models.ProductLog.name = models.CharField(max_length=100) [static]

Definition at line 301 of file models.py.

6.33.2.4 on_delete

bym.BYM_project.main_app.models.ProductLog.on_delete [static]

Definition at line 297 of file models.py.

6.33.2.5 order

bym.BYM_project.main_app.models.ProductLog.order [static]

Definition at line 297 of file models.py.

6.33.2.6 Order

bym.BYM_project.main_app.models.ProductLog.Order [static]

Definition at line 297 of file models.py.

6.33.2.7 price

bym.BYM_project.main_app.models.ProductLog.price = models.PositiveIntegerField() [static]

Definition at line 302 of file models.py.

6.33.2.8 quantity

bym.BYM_project.main_app.models.ProductLog.quantity = models.PositiveIntegerField() [static]

Definition at line 304 of file models.py.

6.33.2.9 shop

bym.BYM_project.main_app.models.ProductLog.shop = models.ForeignKey(Shop, on_delete=models.CASCADE) [static]

Definition at line 298 of file models.py.

6.33.2.10 unit

bym.BYM_project.main_app.models.ProductLog.unit = models.ForeignKey(UnitOfMeasure, on_delete=models.↩ CASCADE) [static]

Definition at line 299 of file models.py.

The documentation for this class was generated from the following file:

- models.py

6.34 bym.BYM_project.main_app.recommendation_system.↩ RecommendationSystem Class Reference

Public Member Functions

- __init__ (self)
- fit (self, X, y)
- predict (self, X, y)

6.34.1 Detailed Description

Definition at line 1 of file recommendation_system.py.

6.34.2 Constructor & Destructor Documentation

6.34.2.1 __init__()

bym.BYM_project.main_app.recommendation_system.RecommendationSystem.__init__ (
self)

Definition at line 2 of file recommendation_system.py.

6.34.3 Member Function Documentation

6.34.3.1 fit()

bym.BYM_project.main_app.recommendation_system.RecommendationSystem.fit (
self,
X,
y)

Definition at line 5 of file recommendation_system.py.

6.34.3.2 predict()

bym.BYM_project.main_app.recommendation_system.RecommendationSystem.predict (
self,
X,
y)

Definition at line 8 of file recommendation_system.py.

The documentation for this class was generated from the following file:

- recommendation_system.py

6.35 bym.BYM_project.main_app.search.SearchDish Class Reference

Класс реализует функционал поиска и сортировки блюд по строке поиска, фильтрам блюда, рекомендациям конткретному пользователю и указанному адресу

Public Member Functions

- None __init__ (self, types=None, subtypes=None, search_string='', address_flg=0, user_id=None, ratio=1, threshold=0)

  Инициализирует класс списком блюд, делая запрос в базу данных
- float score (self, dish)

  Сопосталяет каждому объекту класса Dish число, по которому будет осуществлена сортировка
- list get (self, slice)

  Сортирует список объектов класса Dish.
- float recommend_scorer (self, dish)

  Предсказывает вес нравящности для блюда и пользователя.
- float seq_match_scorer (self, dish)

  Сравнивает Dish.name и self.search_string с помощью SequenceMather()

Public Attributes

- threshold
- search_string
- address_flg
- ratio
- dish_list

6.35.1 Detailed Description

Класс реализует функционал поиска и сортировки блюд по строке поиска, фильтрам блюда, рекомендациям конткретному пользователю и указанному адресу

Definition at line 7 of file search.py.

6.35.2 Constructor & Destructor Documentation

6.35.2.1 __init__()

```
None bym.BYM_project.main_app.search.SearchDish.__init__ (
            self,
            types = None,
            subtypes = None,
            search_string = '',
            address_flg = 0,
            user_id = None,
            ratio = 1,
            threshold = 0 )
```

Инициализирует класс списком блюд, делая запрос в базу данных

Parameters

types=[]	list of int
subtypes=[]	list of int
search_string=''	str
adress_flg=0	int
user_id=None	int
ratio	float (во сколько раз рекомендательная система важнее совпадения строк)

Definition at line 13 of file search.py.

6.35.3 Member Function Documentation

6.35.3.1 get()

list bym.BYM_project.main_app.search.SearchDish.get (
self,
slice)

Сортирует список объектов класса Dish.

Returns

dish_list list of Dish

Definition at line 59 of file search.py.

6.35.3.2 recommend_scorer()

float bym.BYM_project.main_app.search.SearchDish.recommend_scorer (
self,
dish)

Предсказывает вес нравящности для блюда и пользователя.

Parameters

dish	Dish

Returns

weight double

Definition at line 70 of file search.py.

6.35.3.3 score()

float bym.BYM_project.main_app.search.SearchDish.score (
self,
dish)

Сопосталяет каждому объекту класса Dish число, по которому будет осуществлена сортировка

Parameters

dish	Dish

Returns

score double

Definition at line 48 of file search.py.

6.35.3.4 seq_match_scorer()

float bym.BYM_project.main_app.search.SearchDish.seq_match_scorer (
self,
dish)

Сравнивает Dish.name и self.search_string с помощью SequenceMather()

Parameters

dish	Dish

Returns

sm_score float

Definition at line 78 of file search.py.

6.35.4 Member Data Documentation

6.35.4.1 address_flg

bym.BYM_project.main_app.search.SearchDish.address_flg

Definition at line 30 of file search.py.

6.35.4.2 dish_list

bym.BYM_project.main_app.search.SearchDish.dish_list

Definition at line 39 of file search.py.

6.35.4.3 ratio

bym.BYM_project.main_app.search.SearchDish.ratio

Definition at line 35 of file search.py.

6.35.4.4 search_string

bym.BYM_project.main_app.search.SearchDish.search_string

Definition at line 29 of file search.py.

6.35.4.5 threshold

bym.BYM_project.main_app.search.SearchDish.threshold

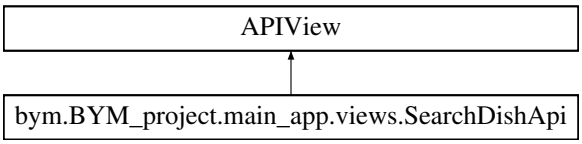
Definition at line 24 of file search.py.

The documentation for this class was generated from the following file:

- search.py

6.36 bym.BYM_project.main_app.views.SearchDishApi Class Reference

Inheritance diagram for bym.BYM_project.main_app.views.SearchDishApi:



Public Member Functions

- post (self, request)

6.36.1 Detailed Description

Definition at line 148 of file views.py.

6.36.2 Member Function Documentation

6.36.2.1 post()

bym.BYM_project.main_app.views.SearchDishApi.post (
self,
request)

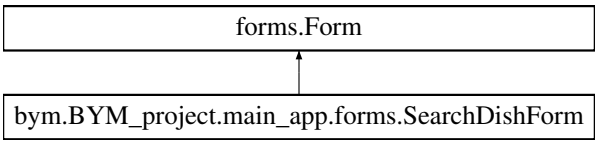
Definition at line 149 of file views.py.

The documentation for this class was generated from the following file:

- views.py

6.37 bym.BYM_project.main_app.forms.SearchDishForm Class Reference

Inheritance diagram for bym.BYM_project.main_app.forms.SearchDishForm:



Static Public Attributes

- search_string

6.37.1 Detailed Description

Definition at line 4 of file forms.py.

6.37.2 Member Data Documentation

6.37.2.1 search_string

bym.BYM_project.main_app.forms.SearchDishForm.search_string [static]

Initial value:
```
= forms.CharField(widget=forms.TextInput(attrs={
      "id": "search-input-id",
      "class": "i1ypu1yl",
      "value": "",
      "aria-label": "Поиск",
      "autocomplete": "off",
      "placeholder": "Найти блюдо"
  }))
```

Definition at line 5 of file forms.py.

The documentation for this class was generated from the following file:

- forms.py

6.38 bym.BYM_project.main_app.search.SearchProduct Class Reference

Public Member Functions

- None __init__ (self, category_id, user_id, threshold=0.7)
- float score (self, product)

  Сопосталяет каждому объекту класса Product число, по которому будет осуществлена сортировка
- list get (self)

  Сортирует список объектов класса Product.
- float seq_match_scorer (self, product)

  Сравнивает Product.name и self.category.name с помощью SequenceMather()
- Category get_category (self)

  Возвращает self.category.

Public Attributes

- category
- products
- threshold

6.38.1 Detailed Description

1. Загружаем из бд список всех спаршенных продуктов
2. Вызов функции SequenceMatch() (взвешиваем каждый product.name относительно category.↩ name по похожести)
3. Сортируем продукты по убыванию весов и эффективности покупки

Parameters

category↩ _id	int

Returns

products [[Product obj, ...], [Category.name, [image1, ...]]]]

Definition at line 95 of file search.py.

6.38.2 Constructor & Destructor Documentation

6.38.2.1 __init__()

```
None bym.BYM_project.main_app.search.SearchProduct.__init__ (
                self,
                category_id,
                user_id,
                threshold = 0.7 )
```

Parameters

category↩ _id	int

Definition at line 104 of file search.py.

6.38.3 Member Function Documentation

6.38.3.1 get()

```
list bym.BYM_project.main_app.search.SearchProduct.get (
                self )
```

Сортирует список объектов класса Product.

Returns

products list of Product

Definition at line 122 of file search.py.

6.38.3.2 get_category()

```
Category bym.BYM_project.main_app.search.SearchProduct.get_category (
            self )
```

Возвращает self.category.

Returns

category Category

Definition at line 149 of file search.py.

6.38.3.3 score()

```
float bym.BYM_project.main_app.search.SearchProduct.score (
            self,
            product )
```

Сопосталяет каждому объекту класса Product число, по которому будет осуществлена сортировка

Parameters

product	Product

Returns

score double

Definition at line 113 of file search.py.

6.38.3.4 seq_match_scorer()

```
float bym.BYM_project.main_app.search.SearchProduct.seq_match_scorer (
            self,
            product )
```

Сравнивает Product.name и self.category.name с помощью SequenceMather()

Parameters

product	Product

Returns

sm_score float

Definition at line 133 of file search.py.

6.38.4 Member Data Documentation

6.38.4.1 category

bym.BYM_project.main_app.search.SearchProduct.category

Definition at line 109 of file search.py.

6.38.4.2 products

bym.BYM_project.main_app.search.SearchProduct.products

Definition at line 110 of file search.py.

6.38.4.3 threshold

bym.BYM_project.main_app.search.SearchProduct.threshold

Definition at line 111 of file search.py.

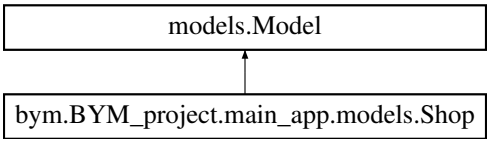
The documentation for this class was generated from the following file:

- search.py

6.39 bym.BYM_project.main_app.models.Shop Class Reference

информация о магазине
name – название магазина
link – ссылка на магазин

Inheritance diagram for bym.BYM_project.main_app.models.Shop:



Classes

- class Meta

Static Public Attributes

- name
- max_length
- link = models.URLField()

6.39.1 Detailed Description

информация о магазине
name – название магазина
link – ссылка на магазин

Definition at line 242 of file models.py.

6.39.2 Member Data Documentation

6.39.2.1 link

bym.BYM_project.main_app.models.Shop.link = models.URLField() [static]

Definition at line 253 of file models.py.

6.39.2.2 max_length

bym.BYM_project.main_app.models.Shop.max_length [static]

Definition at line 252 of file models.py.

6.39.2.3 name

bym.BYM_project.main_app.models.Shop.name [static]

Definition at line 252 of file models.py.

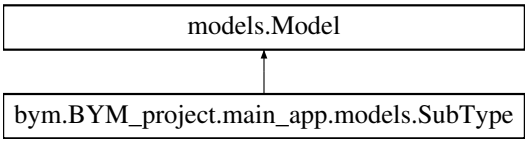
The documentation for this class was generated from the following file:

- models.py

6.40 bym.BYM_project.main_app.models.SubType Class Reference

таблица с подтипами блюд
name – название подтипа (например фруктовые десерты)
type_id – тип к которому относится подтип

Inheritance diagram for bym.BYM_project.main_app.models.SubType:



Classes

- class Meta

Static Public Attributes

- type
- Type
- on_delete
- name = models.CharField(max_length=30)

6.40.1 Detailed Description

таблица с подтипами блюд
name – название подтипа (например фруктовые десерты)
type_id – тип к которому относится подтип

Definition at line 75 of file models.py.

6.40.2 Member Data Documentation

6.40.2.1 name

bym.BYM_project.main_app.models.SubType.name = models.CharField(max_length=30) [static]

Definition at line 87 of file models.py.

6.40.2.2 on_delete

bym.BYM_project.main_app.models.SubType.on_delete [static]

Definition at line 85 of file models.py.

6.40.2.3 type

bym.BYM_project.main_app.models.SubType.type [static]

Definition at line 85 of file models.py.

6.40.2.4 Type

bym.BYM_project.main_app.models.SubType.Type [static]

Definition at line 85 of file models.py.

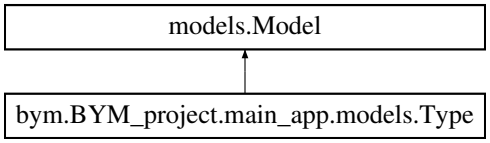
The documentation for this class was generated from the following file:

- models.py

6.41 bym.BYM_project.main_app.models.Type Class Reference

таблица с типами блюд
name – название типа (например Завтраки)

Inheritance diagram for bym.BYM_project.main_app.models.Type:



Classes

- class Meta

Static Public Attributes

- name
- max_length

6.41.1 Detailed Description

таблица с типами блюд
name – название типа (например Завтраки)

Definition at line 63 of file models.py.

6.41.2 Member Data Documentation

6.41.2.1 max_length

bym.BYM_project.main_app.models.Type.max_length [static]

Definition at line 72 of file models.py.

6.41.2.2 name

bym.BYM_project.main_app.models.Type.name [static]

Definition at line 72 of file models.py.

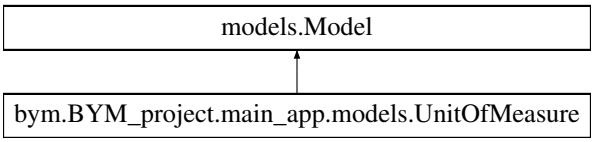
The documentation for this class was generated from the following file:

- models.py

6.42 bym.BYM_project.main_app.models.UnitOfMeasure Class Reference

единицы изменения
name – название единицы измерения (например штуки или кг)
supreme_unit_id – основная форма категории (например Supreme_unit_id у "штуки" это "штука")

Inheritance diagram for bym.BYM_project.main_app.models.UnitOfMeasure:



Classes

- class Meta

Static Public Attributes

- supreme_unit
- null
- True
- blank
- on_delete
- name = models.CharField(max_length=20)

6.42.1 Detailed Description

единицы изменения
name – название единицы измерения (например штуки или кг)
supreme_unit_id – основная форма категории (например Supreme_unit_id у "штуки" это "штука")

Definition at line 155 of file models.py.

6.42.2 Member Data Documentation

6.42.2.1 blank

bym.BYM_project.main_app.models.UnitOfMeasure.blank [static]

Definition at line 165 of file models.py.

6.42.2.2 name

bym.BYM_project.main_app.models.UnitOfMeasure.name = models.CharField(max_length=20) [static]

Definition at line 167 of file models.py.

6.42.2.3 null

bym.BYM_project.main_app.models.UnitOfMeasure.null [static]

Definition at line 165 of file models.py.

6.42.2.4 on_delete

bym.BYM_project.main_app.models.UnitOfMeasure.on_delete [static]

Definition at line 165 of file models.py.

6.42.2.5 supreme_unit

bym.BYM_project.main_app.models.UnitOfMeasure.supreme_unit [static]

Definition at line 165 of file models.py.

6.42.2.6 True

bym.BYM_project.main_app.models.UnitOfMeasure.True [static]

Definition at line 165 of file models.py.

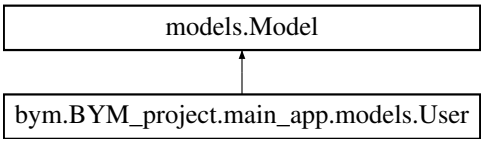
The documentation for this class was generated from the following file:

- models.py

6.43 bym.BYM_project.main_app.models.User Class Reference

Информация о пользователе.

Inheritance diagram for bym.BYM_project.main_app.models.User:



Classes

- class Meta

Static Public Member Functions

- authorization (user_id, login)

  Поиск пользователя по логину, если не найден то присвоить логин текущему user_id.
- get_user_id (request)

  Возвращаем id пользователя - залогиненного или выданного по session_id.

Static Public Attributes

- session
- Session
- on_delete
- CASCADE
- null
- True
- blank
- last_address = models.CharField(max_length=100, null= True, blank= True)
- nickname = models.CharField(max_length=50, null= True, blank= True)
- login = models.EmailField(null= True, blank= True)

6.43.1 Detailed Description

Информация о пользователе.

last_address – последний адрес, с которого пользователь заказывал
nickname – ник пользователья
login – его логин
session_id – id текущей сессии

Definition at line 5 of file models.py.

6.43.2 Member Function Documentation

6.43.2.1 authorization()

bym.BYM_project.main_app.models.User.authorization (
user_id,
login) [static]

Поиск пользователя по логину, если не найден то присвоить логин текущему user_id.

В противном случае вернуть новый user_id

Parameters

user↩ _id	int
login	string

Returns

user_id int

Definition at line 24 of file models.py.

6.43.2.2 get_user_id()

bym.BYM_project.main_app.models.User.get_user_id (request) [static]

Возвращаем id пользователя - залогиненного или выданного по session_id.

Parameters

request	HttpRequest

Returns

user_id int

Definition at line 39 of file models.py.

6.43.3 Member Data Documentation

6.43.3.1 blank

bym.BYM_project.main_app.models.User.blank [static]

Definition at line 17 of file models.py.

6.43.3.2 CASCADE

bym.BYM_project.main_app.models.User.CASCADE [static]

Definition at line 17 of file models.py.

6.43.3.3 last_address

bym.BYM_project.main_app.models.User.last_address = models.CharField(max_length=100, null= True, blank= True) [static]

Definition at line 19 of file models.py.

6.43.3.4 login

bym.BYM_project.main_app.models.User.login = models.EmailField(null= True, blank= True) [static]

Definition at line 21 of file models.py.

6.43.3.5 nickname

bym.BYM_project.main_app.models.User.nickname = models.CharField(max_length=50, null= True, blank= True) [static]

Definition at line 20 of file models.py.

6.43.3.6 null

bym.BYM_project.main_app.models.User.null [static]

Definition at line 17 of file models.py.

6.43.3.7 on_delete

bym.BYM_project.main_app.models.User.on_delete [static]

Definition at line 17 of file models.py.

6.43.3.8 session

bym.BYM_project.main_app.models.User.session [static]

Definition at line 17 of file models.py.

6.43.3.9 Session

bym.BYM_project.main_app.models.User.Session [static]

Definition at line 17 of file models.py.

6.43.3.10 True

bym.BYM_project.main_app.models.User.True [static]

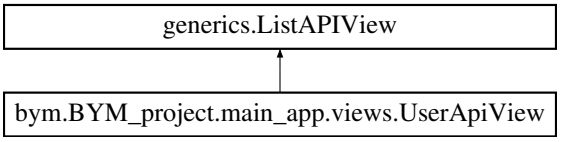
Definition at line 17 of file models.py.

The documentation for this class was generated from the following file:

- models.py

6.44 bym.BYM_project.main_app.views.UserApiView Class Reference

Inheritance diagram for bym.BYM_project.main_app.views.UserApiView:



Static Public Attributes

- queryset = User.objects.all()
- serializer_class = UserSerializer

6.44.1 Detailed Description

Definition at line 119 of file views.py.

6.44.2 Member Data Documentation

6.44.2.1 queryset

bym.BYM_project.main_app.views.UserApiView.queryset = User.objects.all() [static]

Definition at line 120 of file views.py.

6.44.2.2 serializer_class

bym.BYM_project.main_app.views.UserApiView.serializer_class = UserSerializer [static]

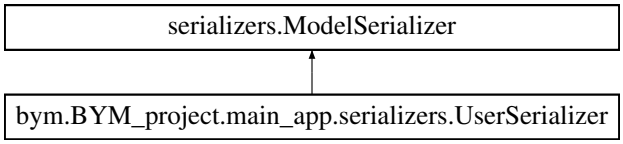
Definition at line 121 of file views.py.

The documentation for this class was generated from the following file:

- views.py

6.45 bym.BYM_project.main_app.serializers.UserSerializer Class Reference

Inheritance diagram for bym.BYM_project.main_app.serializers.UserSerializer:



Classes

- class Meta

6.45.1 Detailed Description

Definition at line 5 of file serializers.py.

The documentation for this class was generated from the following file:

- serializers.py

Глава 7

File Documentation

7.1 __init__.py File Reference

7.2 __init__.py

Go to the documentation of this file.

7.3 admin.py File Reference

Namespaces

- namespace bym
- namespace bym.BYM_project
- namespace bym.BYM_project.main_app
- namespace bym.BYM_project.main_app.admin

7.4 admin.py

Go to the documentation of this file.

```
00001 from django.contrib import admin
00002
00003 from main_app.models import User, Country, Type, SubType, Dish, CookingStage, Category,\
00004                     UnitOfMeasure, Conversion, DishCategory, \
00005                     Order, Menu, Shop, Product, ProductLog
00006
00007 admin.site.register(User)
00008 admin.site.register(Country)
00009 admin.site.register(Type)
00010 admin.site.register(SubType)
00011 admin.site.register(Dish)
00012 admin.site.register(CookingStage)
00013 admin.site.register(Category)
00014 admin.site.register(UnitOfMeasure)
00015 admin.site.register(Conversion)
00016 admin.site.register(DishCategory)
00017 admin.site.register(Order)
00018 admin.site.register(Menu)
00019 admin.site.register(Shop)
00020 admin.site.register(Product)
00021 admin.site.register(ProductLog)
```

7.5 apps.py File Reference

Classes

- class bym.BYM_project.main_app.apps.MainAppConfig

Namespaces

- namespace bym
- namespace bym.BYM_project
- namespace bym.BYM_project.main_app
- namespace bym.BYM_project.main_app.apps

7.6 apps.py

Go to the documentation of this file.

```
00001 from django.apps import AppConfig
00002
00003
00004 class MainAppConfig(AppConfig):
00005     default_auto_field = 'django.db.models.BigAutoField'
00006     name = 'main_app'
```

7.7 cart.py File Reference

Classes

- class bym.BYM_project.main_app.cart.Cart

  Класс для работы с карзиной пользователья

Namespaces

- namespace bym
- namespace bym.BYM_project
- namespace bym.BYM_project.main_app
- namespace bym.BYM_project.main_app.cart

7.8 cart.py

Go to the documentation of this file.

```
00001 from main_app.models import Dish, Menu, Category, DishCategory, Order, UnitOfMeasure, Conversion
00002 from main_app.search import SearchProduct
00003
00004
00005 class Cart:
00006     "'! Класс для работы с карзиной пользователья
00007     "'
00008     def __init__(self, user_id):
00009         "'! Находит текущий Order по user_id
00010         @param user_id
00011         "'
00012         self.order_obj = Order.objects.get(user__id=user_id, current=True)
00013
00014     def change_order(self, dish_id, add_flg):
00015         "'! Изменяем корзину блюд пользователя - или добавить (+1), или убрать (-1)
00016         @param dish_id int
00017         @param add_flg bool - True прибавление, False вычитание
00018         "'
00019         dish_obj = Dish.objects.get(pk=dish_id)
00020         menu_obj, created = Menu.objects.get_or_create(order=self.order_obj, dish=dish_obj)
00021         if not created:
00022             # amount = menu_obj.amount + 1 if add_flg else (-1)
00023             if add_flg == True:
00024                 amount = menu_obj.amount + 1
00025             else:
00026                 amount = menu_obj.amount - 1
00027             if amount <= 0:
00028                 Menu.objects.filter(order=self.order_obj, dish=dish_obj).delete()
00029             else:
00030                 Menu.objects.filter(order=self.order_obj, dish=dish_obj).update(amount=amount)
00031
00032     def get_order_array(self):
00033         "'! Возвращает текущую корзину блюд пользователя
00034         @return order_array [{"dish": Dish, "amount": int}, ...]
00035         "'
00036         # order_array = []
00037         # order = Menu.objects.filter(order=self.order_obj)
00038         # for x in order:
00039         #     order_array.append({"dish": x.dish, "amount": x.amount})
00040         # return order_array
00041
00042         return Menu.objects.filter(order=self.order_obj)
00043
00044     @staticmethod
00045     def get_categories(order_array):
00046         "'! Раскладывает карзину пользователя (в виде блюд), на товары если в рецептах товары в разных единицах
    измерения приводит к единой
00047         @param order_array [{"dish": Dish, "amount": int}, ...]
00048         @return categories [{"category": Category, "amount": float, "unit_name": string}, ...]
00049         "'
00050         category_dict = dict()
00051
00052         for x in order_array:
00053             dish_category_list = DishCategory.objects.filter(dish=x['dish'])
00054
00055             for dish_category_obj in dish_category_list:
00056                 category_id = dish_category_obj.category.id
00057                 number = float(dish_category_obj.amount) * x['amount']
00058                 unit_name = dish_category_obj.unit.supreme_unit.name
00059
00060                 if dish_category_obj.unit.supreme_unit.name == 'по вкусу':
00061                     continue
00062
00063                 if category_id in category_dict.keys():
00064                     if category_dict[category_id]['unit_name'] != unit_name:
00065                         if category_dict[category_id]['unit_name'] == ['г', 'мл']:
00066                             category_dict[category_id]['amount'] += number *
    float(Conversion.objects.get(category=dish_category_obj.category,
    from_unit=dish_category_obj.unit.supreme_unit).to_unit.coefficient)
00067                             continue
00068                         prev_unit = UnitOfMeasure.objects.get(name=category_dict[category_id]['unit_name'])
00069                         if dish_category_obj.unit.supreme_unit.name == 'г':
00070                             category_dict[category_id]['unit_name'] = 'г'
00071                         elif dish_category_obj.unit.supreme_unit.name == 'мл':
00072                             category_dict[category_id]['unit_name'] = 'мл'
00073                         else:
00074                             print(dish_category_obj.category.name, dish_category_obj.unit.supreme_unit.name)
00075                             category_dict[category_id]['unit_name'] =
    Conversion.objects.get(category=dish_category_obj.category,
    from_unit=dish_category_obj.unit.supreme_unit).to_unit.supreme_unit.name
00076
00077                         if prev_unit.name in ['г', 'мл']:
```

```
00078                         category_dict[category_id]['amount'] = category_dict[category_id]['amount']
00079                     else:
00080                         category_dict[category_id]['amount'] = category_dict[category_id]['amount'] *
      float(Conversion.objects.get(category=dish_category_obj.category, from_unit=prev_unit).coefficient)
00081
00082                     if dish_category_obj.unit.name in ['г', 'мл']:
00083                         category_dict[category_id]['amount'] += number
00084                     else:
00085                         category_dict[category_id]['amount'] += number *
      float(Conversion.objects.get(category=dish_category_obj.category,
      from_unit=dish_category_obj.unit.supreme_unit).coefficient)
00086                 else:
00087                     category_dict[category_id]['amount'] += number
00088             else:
00089                 category_dict[category_id] = {"amount": number, "unit_name": unit_name}
00090
00091         categories = []
00092         for key in category_dict.keys():
00093             category = Category.objects.get(id=key)
00094             amount = category_dict[key]['amount']
00095             amount = f'{float(amount):g}'
00096             unit_name = category_dict[key]['unit_name']
00097             if amount == 0:
00098                 amount = None
00099             categories.append({"category": category, "amount": amount, "unit_name": unit_name})
00100
00101         return categories
00102
00103     def get_categories_list(self, slice):
00104         '''! Выдает slice наиболие подходящих товаров в каждой категории
00105         @param slice int
00106         @return categories_list[{"category": string, "category_amount": int, "category_unit_name": string, "product_list":
      [Product, ...]}, ...]
00107         '''
00108         order_array = self.get_order_array()
00109         categories = Cart.get_categories(order_array)
00110
00111         categories_list = []
00112         for cat in categories:
00113             product_list = SearchProduct(cat["category"].id, self.order_obj.user.id, 0).get()[:slice]
00114             for product in product_list:
00115                 product.amount = f'{float(product.amount):g}'
00116
00117             if cat["amount"] is not None:
00118                 categories_list.append({"category": cat["category"].name, "category_amount": cat["amount"],
00119                                         "category_unit_name": cat["unit_name"], "product_list": product_list})
00120
00121         return categories_list
```

7.9 config.py File Reference

Namespaces

- namespace bym
- namespace bym.BYM_project
- namespace bym.BYM_project.main_app
- namespace bym.BYM_project.main_app.config

Variables

- bym.BYM_project.main_app.config.webdriver_name = os.getenv("webdriver_name")
- list bym.BYM_project.main_app.config.urls_lavka

7.10 config.py

Go to the documentation of this file.

```
00001 from dotenv import load_dotenv
00002 import os
00003
```

```
00004
00005 load_dotenv()
00006 webdriver_name = os.getenv("webdriver_name")
00007
00008 urls_lavka = [
00009     'https://lavka.yandex.ru/10743/category/ovoshchi_griby_i_zelen',
00010     'https://lavka.yandex.ru/10743/category/Frukty_i_yagody',
00011     'https://lavka.yandex.ru/10743/category/milk_products',
00012     'https://lavka.yandex.ru/10743/category/cheese',
00013     'https://lavka.yandex.ru/10743/category/kefir_smetana_tvorog',
00014     'https://lavka.yandex.ru/10743/category/e64b861bd9a34ebc9103dbf15e2d2932',
00015     'https://lavka.yandex.ru/10743/category/molochnoe_dlya_detej',
00016     'https://lavka.yandex.ru/10743/category/hleb',
00017     'https://lavka.yandex.ru/10743/category/vipechka',
00018     'https://lavka.yandex.ru/10743/category/hlebcy',
00019     'https://lavka.yandex.ru/10743/category/beef_pork',
00020     'https://lavka.yandex.ru/10743/category/wurst_all',
00021     'https://lavka.yandex.ru/10743/category/ryba_i_moreprodukty',
00022     'https://lavka.yandex.ru/10743/category/zakuski_i_pashtety',
00023     'https://lavka.yandex.ru/10743/category/morozhenoe',
00024     'https://lavka.yandex.ru/10743/category/pelmeni_i_vareniki',
00025     'https://lavka.yandex.ru/10743/category/frozen_vegetables_and_berries',
00026     'https://lavka.yandex.ru/10743/category/deserty',
00027     'https://lavka.yandex.ru/10743/category/polufabrikaty',
00028     'https://lavka.yandex.ru/10743/category/553b84d49692448bb06588f9d694cbde',
00029     'https://lavka.yandex.ru/10743/category/b9c6c90166f1487888de39b933119747',
00030     'https://lavka.yandex.ru/10743/category/makarony_krupy_muka',
00031     'https://lavka.yandex.ru/10743/category/hlopya_i_myusli',
00032     'https://lavka.yandex.ru/10743/category/oils_sauces_spices',
00033     'https://lavka.yandex.ru/10743/category/coffee_and_cocoa',
00034     'https://lavka.yandex.ru/10743/category/tea',
00035     'https://lavka.yandex.ru/10743/category/91ba4698f47f450087e3e8de50d93a15'
00036 ]
```

7.11 conversions_functions.py File Reference

Namespaces

- namespace bym
- namespace bym.BYM_project
- namespace bym.BYM_project.main_app
- namespace bym.BYM_project.main_app.conversions_functions

Functions

- bym.BYM_project.main_app.conversions_functions.how_many_add (category_id, need_unit, have_unit, need, have)

  Расчет сколько штук товара необходимо купть на что бы полностью покрыть рецепт
- bym.BYM_project.main_app.conversions_functions.get_lavka_unit_id (name_unit)

  Исчет единицу измерения из магазина в базе данных, если нет создает

7.12 conversions_functions.py

Go to the documentation of this file.

```
00001 import math
00002 from main_app.models import Conversion, UnitOfMeasure
00003 from django.core.mail import send_mail
00004
00005
00006 def how_many_add(category_id, need_unit, have_unit, need, have):
00007     '''! Расчет сколько штук товара необходимо купть на что бы полностью покрыть рецепт
00008     @param category_id int
00009     @param need_unit UnitOfMeasure - еденицы измерения продукта по рецепту (например стаканы)
00010     @param have_unit UnitOfMeasure - еденицы измерения товара из магазина (например граммы)
00011     @param need int - необходимое количество по рецепту (например 5, have_unit - стаканы)
00012     @param have int - количество в одном юните в магазине (например 500, have_unit-граммы)
00013     @return kol int - сколько нужно заказать
```

```
00014     '''
00015
00016     if need_unit == UnitOfMeasure.objects.get(pk=41) or have_unit == UnitOfMeasure.objects.get(pk=41):
00017         return -1
00018
00019     if need_unit == UnitOfMeasure.objects.get(pk=13):
00020         need = need * 1000
00021         need_unit = UnitOfMeasure.objects.get(pk=7)
00022     elif need_unit == UnitOfMeasure.objects.get(pk=16):
00023         need = need * 1000
00024         need_unit = UnitOfMeasure.objects.get(pk=1)
00025
00026     if have_unit == UnitOfMeasure.objects.get(pk=13):
00027         have = have * 1000
00028         need_unit = UnitOfMeasure.objects.get(pk=7)
00029     elif have_unit == UnitOfMeasure.objects.get(pk=16):
00030         have = have * 1000
00031         need_unit = UnitOfMeasure.objects.get(pk=1)
00032
00033     if have_unit == need_unit:
00034         pass
00035     elif have_unit == UnitOfMeasure.objects.get(pk=1) or have_unit == UnitOfMeasure.objects.get(pk=7):
00036         coef = Conversion(category=category_id, from_unit=need_unit, to_unit=have_unit).coefficient
00037         need = need * coef
00038     elif need_unit == UnitOfMeasure.objects.get(pk=1) or need_unit == UnitOfMeasure.objects.get(pk=7):
00039         coef = Conversion(category=category_id, from_unit=have_unit, to_unit=need_unit).coefficient
00040         need = need / coef
00041     else:
00042         return 0
00043     kol = math.ceil(need / have)
00044     return kol
00045
00046
00047 def get_lavka_unit_id(name_unit):
00048     '''! Исчет единицу измерения из магазина в базе данных, если нет создает
00049     @param name_unit string
00050     @return UnitOfMeasure
00051     '''
00052     units = UnitOfMeasure.objects.filter(name=name_unit)
00053     if units.exists():
00054         for x in units:
00055             return x.supreme_unit
00056     else:
00057         '''
00058         send_mail(
00059             'Subject here',
00060             'Here is the message.',
00061             'from@example.com',
00062             ['sdgssdfsdf@mail.ru'],
00063             fail_silently=False,
00064         )
00065         '''
00066         _, created = UnitOfMeasure.objects.get_or_create(supreme_unit=UnitOfMeasure.objects.get(pk=41),
      name=name_unit)
00067         return UnitOfMeasure.objects.get(pk=41)
```

7.13 forms.py File Reference

Classes

- class bym.BYM_project.main_app.forms.SearchDishForm

Namespaces

- namespace bym
- namespace bym.BYM_project
- namespace bym.BYM_project.main_app
- namespace bym.BYM_project.main_app.forms

7.14 forms.py

Go to the documentation of this file.

```
00001 from django import forms
00002
00003
00004 class SearchDishForm(forms.Form):
00005     search_string = forms.CharField(widget=forms.TextInput(attrs={
00006         "id": "search-input-id",
00007         "class": "i1ypu1yl",
00008         "value": "",
00009         "aria-label": "Поиск",
00010         "autocomplete": "off",
00011         "placeholder": "Найти блюдо"
00012     }))
```

7.15 functions_backend.py File Reference

Namespaces

- namespace bym
- namespace bym.BYM_project
- namespace bym.BYM_project.main_app
- namespace bym.BYM_project.main_app.functions_backend

Functions

- bym.BYM_project.main_app.functions_backend.start_session (request)

  Начало сессии с пользователем и создание его кортежа в бд по session_id.
- bym.BYM_project.main_app.functions_backend.get_dish_types ()

  Запрос в бд на получение типов блюд и соответстующих им под типов
- bym.BYM_project.main_app.functions_backend.get_particular_dish (dish_id)

  Возвращает конкретное блюдо, стадии его приготовления и необходимые игридиенты
- bym.BYM_project.main_app.functions_backend.string_to_minutes (string)

  Возвращает количество минут в строке Варианты входа: 'n часов', 'n минут', 'n часов k минут'.
- bym.BYM_project.main_app.functions_backend.convert_fraction (string)

  Возвращает float из строки с числом
- bym.BYM_project.main_app.functions_backend.number_and_measure (string)

  Возвращает по строке количества продукта число + единицу измерения в начальной форме

7.16 functions_backend.py

Go to the documentation of this file.

```
00001 from django.utils import timezone
00002 import re
00003 import unicodedata
00004 from main_app.models import User, Type, SubType, Dish, CookingStage, Order, DishCategory
00005
00006
00007 def start_session(request):
00008     '''! Начало сессии с пользователем и создание его кортежа в бд по session_id
00009     @param request HttpRequest
00010     '''
00011     if 'start_session' not in request.session:
00012         request.session['start_session'] = 'start'
00013         request.session.save()
00014
00015         user_obj = User(session_id=request.session.session_key)
00016         user_obj.save()
```

```
00017         order = Order(user=user_obj, order_date=timezone.now())
00018         order.save()
00019
00020
00021 def get_dish_types():
00022     '''! Запрос в бд на получение типов блюд и соответстующих им под типов
00023     @return dict {type:[subtype,...]}
00024     '''
00025     dic = {}
00026     types = Type.objects.all()
00027     for type in types:
00028         dic[type] = [subtype for subtype in SubType.objects.filter(type=type)]
00029     return dic
00030
00031
00032 def get_particular_dish(dish_id):
00033     '''! Возвращает конкретное блюдо, стадии его приготовления и необходимые игридиенты
00034     @param dish_id int
00035     @return dish Dish
00036     @return cooking_stages [CookingStage,...]
00037     @return ingredients [{'name':string, amount:int}] - список ингридиентов - название категории их кол-во
00038     '''
00039     dish = Dish.objects.get(pk=dish_id)
00040     cooking_stages = CookingStage.objects.filter(dish=dish)
00041     ingredients = []
00042     for x in DishCategory.objects.filter(dish=dish):
00043         if x.amount != 0:
00044             amount = ' '.join([f'{float(x.amount):g}', x.unit.name])
00045         else:
00046             amount = x.unit.name
00047         ingredients.append({'name': x.category.name, 'amount': amount})
00048
00049     return dish, cooking_stages, ingredients
00050
00051
00052 def string_to_minutes(string):
00053     '''! Возвращает количество минут в строке
00054     Варианты входа: 'n часов', 'n минут', 'n часов k минут'
00055     @param string string
00056     @return int
00057     '''
00058     nums = [int(x) for x in re.findall(r'\d+', string)]
00059     if 'час' in string:
00060         if 'мин' in string:
00061             return nums[0] * 60 + nums[1]
00062         else:
00063             return nums[0] * 60
00064     else:
00065         return nums[0]
00066
00067
00068 def convert_fraction(string):
00069     '''! Возвращает float из строки с числом
00070     @param string string
00071     @return float
00072     '''
00073     string = string.replace(',', '.')
00074     if len(string) == 1:
00075         return unicodedata.numeric(string)
00076     return float(string)
00077
00078
00079 def number_and_measure(string):
00080     '''! Возвращает по строке количества продукта число + единицу измерения в начальной форме
00081     @param string string
00082     @return float
00083     @return string
00084     '''
00085     words = string.split()
00086     if len(words) >= 2 and (words[0].isnumeric() or '.' in words[0] or ',' in words[0]):
00087         return convert_fraction(words[0]), ' '.join(words[1:])
00088     else:
00089         return 0.0, ' '.join(words)
```

7.17 models.py File Reference

Classes

- class bym.BYM_project.main_app.models.User

  Информация о пользователе.

- class bym.BYM_project.main_app.models.User.Meta
- class bym.BYM_project.main_app.models.Country

  Таблица с кухнями различных стран
  name – название кухни (например Русская кухня)

- class bym.BYM_project.main_app.models.Country.Meta
- class bym.BYM_project.main_app.models.Type

  таблица с типами блюд
  name – название типа (например Завтраки)

- class bym.BYM_project.main_app.models.Type.Meta
- class bym.BYM_project.main_app.models.SubType

  таблица с подтипами блюд
  name – название подтипа (например фруктовые десерты)
  type_id – тип к которому относится подтип

- class bym.BYM_project.main_app.models.SubType.Meta
- class bym.BYM_project.main_app.models.Dish

  информация о блюде
  name – названия блюда
  pers_num – кол-во порций
  cooking_time_minutes – время готовки в минутах
  calories – кол-во калорий в блюде
  proteins – кол0белки в блюде
  carbohydrates – углеводы в блюде
  fats – жиры
  country_id – страна из кухни которой это блюдо (например итальянская кухня)
  subtype_id – подтип блюда (например, фруктовые десерты)
  image – ссылка на изображения блюда

- class bym.BYM_project.main_app.models.Dish.Meta
- class bym.BYM_project.main_app.models.CookingStage

  шаги для приготовления блюда
  description – текст того, что нужно сделать на этом шаге
  order – порядковый номер шага
  dish_id – блюдо, к которому относится этот шаг
  image – ссылка на изображение шага

- class bym.BYM_project.main_app.models.CookingStage.Meta
- class bym.BYM_project.main_app.models.Category

  категории различных продуктов для приготовления
  name – название категории (например: бакалея или помидоры)
  supreme_category_id – надкатегория (например надкатегорией римских помидор являются помидоры)

- class bym.BYM_project.main_app.models.Category.Meta
- class bym.BYM_project.main_app.models.UnitOfMeasure

  единицы изменения
  name – название единицы измерения (например штуки или кг)
  supreme_unit_id – основная форма категории (например Supreme_unit_id у "штуки" это "штука")

- class bym.BYM_project.main_app.models.UnitOfMeasure.Meta
- class bym.BYM_project.main_app.models.Conversion

  коэффициенты для перевода единиц измерения
  coefficient – число на которое нужно умножить что бы перевести from_unit_id в to_unit_id
  category_id – категория для которой выполняется перевод (очевидно что коэффициент для перевода штук в граммы для помидоров и огурцов отличается)

from_unit_id– единица измерения из которой выполняется перевод
to_unit_id – единица измерения в которую выполняется перевод

- class bym.BYM_project.main_app.models.Conversion.Meta
- class bym.BYM_project.main_app.models.DishCategory

  информация о составе блюда.
- class bym.BYM_project.main_app.models.DishCategory.Meta
- class bym.BYM_project.main_app.models.Order

  информация о заказах.
- class bym.BYM_project.main_app.models.Order.Meta
- class bym.BYM_project.main_app.models.Menu

  состав заказа
  amount– кол-во товара, которое было заказано
  dish_id – товар который был заказан
  order_id – заказ в котором этот товар был заказан

- class bym.BYM_project.main_app.models.Menu.Meta
- class bym.BYM_project.main_app.models.Shop

  информация о магазине
  name – название магазина
  link – ссылка на магазин

- class bym.BYM_project.main_app.models.Shop.Meta
- class bym.BYM_project.main_app.models.Product

  информация о конкретном товаре в магазине
  name – название товара
  price – цена
  amount – количество (в unit_id) товара которое в одной единице заказ (например 300, unit_id – граммы)
  link – ссылка на товар
  shop_id – магазин, из которого товар
  unit_id – единица измерения
  user_id – пользователь для которого был спаршен данный товар

- class bym.BYM_project.main_app.models.Product.Meta
- class bym.BYM_project.main_app.models.ProductLog

  информация о ранее заказанных товарах в магазине
  name – название товара
  price – цена
  amount – количество (в unit_id) товара которое в одной единице заказ (например 300, unit_id – граммы)
  quantity – количество единиц, которое было заказано
  link – ссылка на товар
  shop_id – магазин, из которого товар
  order_id – заказ, в котором был куплен данный товар
  unit_id – единица измерения

- class bym.BYM_project.main_app.models.ProductLog.Meta

Namespaces

- namespace bym
- namespace bym.BYM_project
- namespace bym.BYM_project.main_app
- namespace bym.BYM_project.main_app.models

7.18 models.py

Go to the documentation of this file.

```
00001 from django.db import models
00002 from django.contrib.sessions.models import Session
00003
00004
00005 class User(models.Model):
00006     "'! Информация о пользователе. <br>
00007     last_address – последний адрес, с которого пользователь заказывал <br>
00008     nickname – ник пользователья <br>
00009     login – его логин <br>
00010     session_id – id текущей сессии <br>
00011     "'
00012
00013     class Meta:
00014         verbose_name = 'Пользователь'
00015         verbose_name_plural = 'Пользователи'
00016
00017     session = models.ForeignKey(Session, on_delete=models.CASCADE, null=True, blank=True)
00018
00019     last_address = models.CharField(max_length=100, null=True, blank=True)
00020     nickname = models.CharField(max_length=50, null=True, blank=True)
00021     login = models.EmailField(null=True, blank=True)
00022
00023     @staticmethod
00024     def authorization(user_id, login):
00025         "'! Поиск пользователя по логину, если не найден то присвоить логин текущему user_id.
00026          В противном случае вернуть новый user_id
00027         @param user_id int
00028         @param login string
00029         @return user_id int
00030         "'
00031         try:
00032             user = User.objects.get(login=login)
00033             user_id = user.pk
00034         except User.DoesNotExist:
00035             User.objects.filter(pk=user_id).update(login=login)
00036         return user_id
00037
00038     @staticmethod
00039     def get_user_id(request):
00040         "'! Возвращаем id пользователя - залогиненного или выданного по session_id
00041         @param request HttpRequest
00042         @return user_id int
00043         "'
00044         if request.user.is_authenticated:
00045             current_user = request.user
00046         else:
00047             current_user = User.objects.get(session_id=request.session.session_key)
00048         return current_user.id
00049
00050
00051 class Country(models.Model):
00052     "'! Таблица с кухнями различных стран <br>
00053     name – название кухни (например Русская кухня) <br>
00054     "'
00055
00056     class Meta:
00057         verbose_name = 'Страна'
00058         verbose_name_plural = 'Страны'
00059
00060     name = models.CharField(max_length=30)
00061
00062
00063 class Type(models.Model):
00064     "'! таблица с типами блюд <br>
00065     name – название типа (например Завтраки) <br>
00066     "'
00067
00068     class Meta:
00069         verbose_name = 'Тип бдюда'
00070         verbose_name_plural = 'Типы блюд'
00071
00072     name = models.CharField(max_length=30)
00073
00074
00075 class SubType(models.Model):
00076     "'! таблица с подтипами блюд <br>
00077     name – название подтипа (например фруктовые десерты) <br>
00078     type_id – тип к которому относится подтип <br>
00079     "'
00080
00081     class Meta:
00082         verbose_name = 'Подтип блюда'
```

```
00083         verbose_name_plural = 'Подтипы блюд'
00084
00085     type = models.ForeignKey(Type, on_delete=models.CASCADE)
00086
00087     name = models.CharField(max_length=30)
00088
00089
00090 class Dish(models.Model):
00091     '''! информация о блюде <br>
00092     name – названия блюда <br>
00093     pers_num – кол-во порций <br>
00094     cooking_time_minutes – время готовки в минутах <br>
00095     calories – кол-во калорий в блюде <br>
00096     proteins – кол0белки в блюде <br>
00097     carbohydrates – углеводы в блюде <br>
00098     fats – жиры <br>
00099     country_id – страна из кухни которой это блюдо (например итальянская кухня) <br>
00100     subtype_id – подтип блюда (например, фруктовые десерты) <br>
00101     image – ссылка на изображения блюда <br>
00102     '''
00103
00104     class Meta:
00105         verbose_name = 'Блюдо'
00106         verbose_name_plural = 'Блюда'
00107
00108     country = models.ForeignKey(Country, on_delete=models.CASCADE)
00109     subtype = models.ForeignKey(SubType, on_delete=models.CASCADE)
00110
00111     name = models.CharField(max_length=100)
00112     pers_num = models.PositiveIntegerField()
00113     cooking_time_minutes = models.PositiveIntegerField()
00114     image = models.ImageField(null=True, blank=True, max_length=130)
00115     calories = models.PositiveIntegerField()
00116     proteins = models.PositiveIntegerField()
00117     carbohydrates = models.PositiveIntegerField()
00118     fats = models.PositiveIntegerField()
00119
00120
00121 class CookingStage(models.Model):
00122     '''! шаги для приготовления блюда <br>
00123     description – текст того, что нужно сделать на этом шаге <br>
00124     order – порядковый номер шага <br>
00125     dish_id – блюдо, к которому относится этот шаг <br>
00126     image – ссылка на изображение шага <br>
00127     '''
00128
00129     class Meta:
00130         verbose_name = 'Этап приготовления блюда'
00131         verbose_name_plural = 'Этапы приготовления блюд'
00132
00133     dish = models.ForeignKey(Dish, on_delete=models.CASCADE)
00134
00135     description = models.TextField()
00136     order = models.PositiveIntegerField()
00137     image = models.ImageField(null=True, blank=True, max_length=130)
00138
00139
00140 class Category(models.Model):
00141     '''! категории различных продуктов для приготовления <br>
00142     name – название категории (например: бакалея или помидоры) <br>
00143     supreme_category_id – надкатегория (например надкатегорией римских помидор являются помидоры) <br>
00144     '''
00145
00146     class Meta:
00147         verbose_name = 'Категория продукта'
00148         verbose_name_plural = 'Категории продуктов'
00149
00150     supreme_category = models.ForeignKey('self', null=True, blank=True, on_delete=models.CASCADE)
00151
00152     name = models.CharField(max_length=70)
00153
00154
00155 class UnitOfMeasure(models.Model):
00156     '''! единицы изменения <br>
00157     name – название единицы измерения (например штуки или кг) <br>
00158     supreme_unit_id – основная форма категории (например Supreme_unit_id у "штуки" это "штука") <br>
00159     '''
00160
00161     class Meta:
00162         verbose_name = 'Единица измерения'
00163         verbose_name_plural = 'Единицы измерения'
00164
00165     supreme_unit = models.ForeignKey('self', null=True, blank=True, on_delete=models.CASCADE)
00166
00167     name = models.CharField(max_length=20)
00168
00169
```

```
00170 class Conversion(models.Model):
00171     '''! коэффициенты для перевода единиц измерения <br>
00172     coefficient – число на которое нужно умножить что бы перевести from_unit_id в to_unit_id <br>
00173     category_id – категория для которой выполняется перевод (очевидно что коэффициент для перевода штук в
      граммы для помидоров и огурцов отличается) <br>
00174     from_unit_id– единица измерения из которой выполняется перевод <br>
00175     to_unit_id – единица измерения в которую выполняется перевод <br>
00176     '''
00177 
00178     class Meta:
00179         verbose_name = 'Преобразование единицы измерения'
00180         verbose_name_plural = 'Преобразования единиц измерения'
00181 
00182     category = models.ForeignKey(Category, on_delete=models.CASCADE)
00183     from_unit = models.ForeignKey(UnitOfMeasure, related_name='from_unit', on_delete=models.CASCADE)
00184     to_unit = models.ForeignKey(UnitOfMeasure, related_name='to_unit', null=True, blank=True,
      on_delete=models.CASCADE)
00185 
00186     coefficient = models.DecimalField(max_digits=4 + 4, decimal_places=4, default=1)
00187 
00188 
00189 class DishCategory(models.Model):
00190     '''! информация о составе блюда. <br>
00191     amount – кол-во продукта в блюде <br>
00192     category_id – категория продукта <br>
00193     dish_id – блюдо к которому относится продукт <br>
00194     unit_id – единицы измерения в которых указано кол-во <br>
00195     '''
00196 
00197     class Meta:
00198         verbose_name = 'Продукт в блюде'
00199         verbose_name_plural = 'Продукты в блюдах'
00200 
00201     dish = models.ForeignKey(Dish, on_delete=models.CASCADE)
00202     category = models.ForeignKey(Category, on_delete=models.CASCADE)
00203     unit = models.ForeignKey(UnitOfMeasure, null=True, blank=True, on_delete=models.CASCADE)
00204 
00205     amount = models.DecimalField(max_digits=4 + 4, decimal_places=4)
00206 
00207 
00208 class Order(models.Model):
00209     '''! информация о заказах. <br>
00210     order_date – дата заказа <br>
00211     current – статус (выполнен/ не выполнен) <br>
00212     user_id – пользователь совершивший заказ <br>
00213     '''
00214 
00215     class Meta:
00216         verbose_name = 'Заказ'
00217         verbose_name_plural = 'Заказы'
00218 
00219     user = models.ForeignKey(User, on_delete=models.CASCADE)
00220 
00221     order_date = models.DateTimeField()
00222     current = models.BooleanField(default=True)
00223 
00224 
00225 class Menu(models.Model):
00226     '''! состав заказа <br>
00227     amount– кол-во товара, которое было заказано <br>
00228     dish_id – товар который был заказан <br>
00229     order_id – заказ в котором этот товар был заказан <br>
00230     '''
00231 
00232     class Meta:
00233         verbose_name = 'Меню'
00234         verbose_name_plural = 'Меню'
00235 
00236     dish = models.ForeignKey(Dish, on_delete=models.CASCADE)
00237     order = models.ForeignKey(Order, on_delete=models.CASCADE)
00238 
00239     amount = models.PositiveIntegerField(default=1)
00240 
00241 
00242 class Shop(models.Model):
00243     '''! информация о магазине <br>
00244     name – название магазина <br>
00245     link – ссылка на магазин <br>
00246     '''
00247 
00248     class Meta:
00249         verbose_name = 'Магазин'
00250         verbose_name_plural = 'Магазины'
00251 
00252     name = models.CharField(max_length=30)
00253     link = models.URLField()
00254 
```

```
00255
00256 class Product(models.Model):
00257     '''! информация о конкретном товаре в магазине <br>
00258     name – название товара <br>
00259     price – цена <br>
00260     amount – количество (в unit_id) товара которое в одной единице заказ (например 300, unit_id – граммы) <br>
00261     link – ссылка на товар <br>
00262     shop_id – магазин, из которого товар <br>
00263     unit_id – единица измерения <br>
00264     user_id – пользователь для которого был спаршен данный товар <br>
00265     '''
00266
00267     class Meta:
00268         verbose_name = 'Продукт из магазина'
00269         verbose_name_plural = 'Продукты из магазинов'
00270
00271     user = models.ForeignKey(User, on_delete=models.CASCADE)
00272     unit = models.ForeignKey(UnitOfMeasure, on_delete=models.CASCADE)
00273     shop = models.ForeignKey(Shop, on_delete=models.CASCADE)
00274
00275     name = models.CharField(max_length=100)
00276     price = models.PositiveIntegerField()
00277     amount = models.DecimalField(max_digits=4 + 4, decimal_places=4)
00278     link = models.URLField()
00279
00280
00281 class ProductLog(models.Model):
00282     '''! информация о ранее заказанных товарах в магазине <br>
00283     name – название товара <br>
00284     price – цена  <br>
00285     amount – количество (в unit_id) товара которое в одной единице заказ (например 300, unit_id – граммы) <br>
00286     quantity – количество единиц, которое было заказано <br>
00287     link – ссылка на товар <br>
00288     shop_id – магазин, из которого товар <br>
00289     order_id – заказ, в котором был куплен данный товар <br>
00290     unit_id – единица измерения <br>
00291     '''
00292
00293     class Meta:
00294         verbose_name = 'Лог родукта из магазина'
00295         verbose_name_plural = 'Логи родуктов из магазинов'
00296
00297     order = models.ForeignKey(Order, on_delete=models.CASCADE)
00298     shop = models.ForeignKey(Shop, on_delete=models.CASCADE)
00299     unit = models.ForeignKey(UnitOfMeasure, on_delete=models.CASCADE)
00300
00301     name = models.CharField(max_length=100)
00302     price = models.PositiveIntegerField()
00303     amount = models.DecimalField(max_digits=4 + 4, decimal_places=4)
00304     quantity = models.PositiveIntegerField()
00305     link = models.URLField()
```

7.19 parser.py File Reference

Classes

- class bym.BYM_project.main_app.parser.Parser

  Определяет базавый класс используемый для парсинга всех сайтов
- class bym.BYM_project.main_app.parser.Lavka

  Определяет класс используемый для парсинга лавки.

Namespaces

- namespace bym
- namespace bym.BYM_project
- namespace bym.BYM_project.main_app
- namespace bym.BYM_project.main_app.parser

Functions

- bym.BYM_project.main_app.parser.parse_by_address (user_id, city_street, number)

7.20 parser.py

Go to the documentation of this file.

```
00001 from selenium.webdriver import Chrome
00002 from selenium.webdriver.common.by import By
00003 from selenium.webdriver.common.keys import Keys
00004 from selenium.webdriver.support import expected_conditions as EC
00005 from selenium.webdriver.support.ui import WebDriverWait
00006 from selenium import webdriver
00007 from selenium.webdriver.chrome.service import Service as ChromeService
00008 from webdriver_manager.chrome import ChromeDriverManager
00009 from main_app.config import urls_lavka, webdriver_name
00010 from bs4 import BeautifulSoup
00011 import asyncio
00012 from main_app.models import Product, Shop, User, UnitOfMeasure
00013 from main_app.conversions_functions import get_lavka_unit_id
00014
00015
00016 def parse_by_address(user_id, city_street, number):
00017     a = Lavka(user_id)
00018     # asyncio.run(a.pars("Москва, Таллинская улица", "34"))
00019     asyncio.run(a.pars(city_street, number))
00020     a.sql_script()
00021
00022
00023 class Parser:
00024     '''!
00025     Определяет базавый класс используемый для парсинга всех сайтов
00026     '''
00027
00028     def __init__(self, user_id, shop_id):
00029         '''!
00030         Инициализирует класс Parser.
00031         @param user_id
00032         @param shop_id
00033         '''
00034         self.user_obj = User.objects.get(pk=user_id)
00035         self.options = webdriver.ChromeOptions()
00036         self.product_list = []
00037         self.shop_obj = Shop.objects.get(pk=shop_id)
00038
00039     def sql_script(self):
00040         '''! Сохраняет в базу данных
00041         '''
00042         print("-----START_SAVE_BD------")
00043
00044         for x in self.product_list:
00045             # if x["unit"] == 'шт.':
00046             #     надо поработать с get UnitOfMeasure
00047             unit_obj = get_lavka_unit_id(x['unit'])
00048             _, created = Product.objects.get_or_create(user=self.user_obj,
00049                                                        unit=unit_obj,
00050                                                        shop=self.shop_obj, name=x["name"],
00051                                                        price=x["price"], amount=x["amount"], link=x["link"])
00052
00053         print("-----END------")
00054
00055
00056 class Lavka(Parser):
00057     '''!
00058     Определяет класс используемый для парсинга лавки.
00059     '''
00060
00061     def __init__(self, user_id):
00062         '''!
00063         Инициализирует класс Lavka.
00064         @param user_id int
00065         '''
00066         super().__init__(user_id, 1)
00067         self.url_main = "https://lavka.yandex.ru"
00068         self.options.add_argument("--start-maximized")
00069         prefs = {"profile.managed_default_content_settings.images": 2}
00070         self.options.add_experimental_option("prefs", prefs)
00071         self.options.headless = True
00072         self.options.page_load_strategy = 'eager'
00073         self.driver = Chrome(options=self.options, service=ChromeService(ChromeDriverManager().install()))
00074         self.wait = WebDriverWait(self.driver, 100)
00075         self.urls = urls_lavka
00076         self.sl_tags = dict(product_block="p10zc8qs", product_name="l4t8cc8", product_link="p11oed5n",
00077                             discount_price="b1bvai5j", no_discount_price="c1jd7nwq", amount_and_unit="iks4ndv")
00078
00079     async def get_data_product(self, product):
00080         '''!
00081         Выделяет из блока с информацией о продукте нужную и записывает в БД
00082         @param product BeautifulSoup
```

```
00083         '''
00084         if (product.find(class_=self.sl_tags["product_name"]) is not None) and (
00085                 product.find(class_=self.sl_tags["product_link"]) is not None):
00086             name = product.find(class_=self.sl_tags["product_name"]).get_text().replace('', '')
00087             # print(name)
00088             if product.find(class_=self.sl_tags["discount_price"]) is not None:
00089                 price = int(
00090                     product.find(class_=self.sl_tags["discount_price"]).get_text().replace('p', '').replace(' ',
00091                                                                                              '').replace(
00092                         ' ', ''))
00093             else:
00094                 price = int(
00095                     product.find(class_=self.sl_tags["no_discount_price"]).get_text().replace('p', '').replace(' ',
00096                                                                                               '').replace(
00097                         ' ', ''))
00098             link = self.url_main + product.find(class_=self.sl_tags["product_link"])['href']
00099             am_un = product.find(class_=self.sl_tags["amount_and_unit"]).get_text()
00100             amount = float(am_un[:am_un.find(' ')].replace(',', '.'))
00101             unit = am_un[am_un.find(' ') + 1:]
00102             self.product_list.append({"name": name, "price": price, "link": link, "amount": amount, "unit": unit})
00103         else:
00104             # print("pass")
00105             pass
00106 
00107     async def get_data_html(self, html):
00108         '''!
00109         Разбивает html код на блоки по продуктам, асинхронно запускает их обработку
00110         @param html string
00111         '''
00112 
00113         soup = BeautifulSoup(html, 'html.parser')
00114 
00115         products = soup.find_all(class_=self.sl_tags["product_block"])
00116         tasks = []
00117         loop2 = asyncio.get_event_loop()
00118         for product in products:
00119             task = loop2.create_task(self.get_data_product(product))
00120             tasks.append(task)
00121         await asyncio.wait(tasks)
00122 
00123     async def pars(self, city_street, home):
00124         '''!
00125         Парсит список продуктов с ценами яндекс лавки по определенному адресу
00126         Производит запись в базу данных
00127         @param city_strit string
00128         @param home string
00129         '''
00130         print("pars_start")
00131         self.driver.get(self.url_main + "/10743/category/ovoshchi_griby_i_zelen")
00132         ukaz_adr = self.wait.until(EC.presence_of_element_located((By.XPATH,
    '//*[@id="root"]/div[2]/header/div[5]/button')))
00133         ukaz_adr.click()
00134         print("adr_start")
00135         self.wait.until(EC.presence_of_element_located((By.CLASS_NAME, 'c12fmzph')))
00136         await asyncio.sleep(1)
00137         kr = self.wait.until(EC.element_to_be_clickable(
00138             (By.XPATH, '/html/body/div[2]/div[3]/div/div/div/div[1]/div[2]/div[1]/div/button')))
00139         kr.click()
00140         adr_inp = self.driver.find_element(By.CLASS_NAME, 'i1645061')
00141         for lit in city_street:
00142             adr_inp.send_keys(lit)
00143             await asyncio.sleep(0.1)
00144         adr_inp.send_keys(' ')
00145         await asyncio.sleep(1)
00146         for lit in home:
00147             adr_inp.send_keys(lit)
00148             await asyncio.sleep(1)
00149         await asyncio.sleep(5)
00150         adr_inp.send_keys(Keys.DOWN)
00151         adr_inp.send_keys(Keys.ENTER)
00152         ok = self.wait.until(
00153             EC.element_to_be_clickable((By.XPATH, '/html/body/div[2]/div[3]/div/div/div/div[1]/div[2]/div[2]/button')))
00154         ok.click()
00155         self.wait.until(EC.presence_of_element_located((By.CLASS_NAME, 's17r5x1')))
00156         print("adr_finish")
00157         k = 6
00158 
00159         for j in range(0, k):
00160             tasks = []
00161             loop1 = asyncio.get_event_loop()
00162 
00163             i = j
00164             tabi = 1
00165             while i < len(self.urls):
00166                 self.driver.execute_script(f"window.open('{self.urls[i]}', 'tab{tabi}');")
00167                 i = i + k
00168                 tabi = tabi + 1
```

```
00169             i = j
00170             tabi = 1
00171             while i < len(self.urls):
00172                 self.driver.switch_to.window(f'tab{tabi}')
00173                 html = self.driver.page_source
00174                 task = loop1.create_task(self.get_data_html(html))
00175                 tasks.append(task)
00176                 i = i + k
00177                 tabi = tabi + 1
00178             await asyncio.wait(tasks)
00179 
00180         self.driver.quit()
00181 
00182 # a = Lavka()
00183 # asyncio.run(a.pars("Москва, Таллинская улица","34"))
```

7.21 recommendation_system.py File Reference

Classes

- class bym.BYM_project.main_app.recommendation_system.RecommendationSystem

Namespaces

- namespace bym
- namespace bym.BYM_project
- namespace bym.BYM_project.main_app
- namespace bym.BYM_project.main_app.recommendation_system

7.22 recommendation_system.py

Go to the documentation of this file.

```
00001 class RecommendationSystem:
00002     def __init__(self):
00003         pass
00004 
00005     def fit(self, X, y):
00006         return self
00007 
00008     def predict(self, X, y):
00009         pass
```

7.23 search.py File Reference

Classes

- class bym.BYM_project.main_app.search.SearchDish

  Класс реализует функционал поиска и сортировки блюд по строке поиска, фильтрам блюда, рекомендациям конткретному пользователю и указанному адресу
- class bym.BYM_project.main_app.search.SearchProduct

Namespaces

- namespace bym
- namespace bym.BYM_project
- namespace bym.BYM_project.main_app
- namespace bym.BYM_project.main_app.search

7.24 search.py

Go to the documentation of this file.

```
00001 from main_app.models import Dish, Category, Product
00002 from difflib import SequenceMatcher
00003 from django.db.models import Q
00004 from main_app.recommendation_system import RecommendationSystem
00005
00006
00007 class SearchDish:
00008     '''!
00009     Класс реализует функционал поиска и сортировки блюд по строке поиска, фильтрам блюда,
00010     рекомендациям конткретному пользователю и указанному адресу
00011     '''
00012
00013     def __init__(self, types=None, subtypes=None, search_string='', address_flg=0, user_id=None, ratio=1,
      threshold=0) -> None:
00014         '''!
00015         Инициализирует класс списком блюд, делая запрос в базу данных
00016         @param types=[] list of int
00017         @param subtypes=[] list of int
00018         @param search_string='' str
00019         @param adress_flg=0 int
00020         @param user_id=None int
00021         @param ratio float (во сколько раз рекомендательная система важнее совпадения строк)
00022         '''
00023
00024         self.threshold = threshold
00025         if subtypes is None:
00026             subtypes = []
00027         if types is None:
00028             types = []
00029         self.search_string = search_string
00030         self.address_flg = address_flg
00031
00032         if ratio < 0:
00033             raise ValueError("Ratio is less than 0")
00034         else:
00035             self.ratio = ratio
00036
00037         if subtypes.__len__():
00038             if types.__len__():
00039                 self.dish_list = Dish.objects.filter(Q(subtype__id__in=subtypes) | Q(subtype__type__id__in=types))
00040             else:
00041                 self.dish_list = Dish.objects.filter(subtype__id__in=subtypes)
00042         else:
00043             if types.__len__():
00044                 self.dish_list = Dish.objects.filter(subtype__type__id__in=types)
00045             else:
00046                 self.dish_list = Dish.objects.all()
00047
00048     def score(self, dish) -> float:
00049         '''!
00050         Сопосталяет каждому объекту класса Dish число, по которому будет осуществлена сортировка
00051         @param dish Dish
00052         @return score double
00053         '''
00054         sm_score = self.seq_match_scorer(dish)
00055         rec_score = self.recommend_scorer(dish)
00056         score = sm_score / (self.ratio + 1) + rec_score * self.ratio / (self.ratio + 1)
00057         return score
00058
00059     def get(self, slice) -> list:
00060         '''!
00061         Сортирует список объектов класса Dish
00062         @return dish_list list of Dish
00063         '''
00064         new = sorted(self.dish_list, key=self.score, reverse=True)
00065         for i in range(len(new)):
00066             if self.score(new[i]) < self.threshold:
00067                 return new[:min(i, slice)]
00068         return new[:slice]
00069
00070     def recommend_scorer(self, dish) -> float:
00071         '''! Предсказывает вес нравящности для блюда и пользователя.
00072         @param dish Dish
00073         @return weight double
00074         '''
00075
00076         return 0
00077
00078     def seq_match_scorer(self, dish) -> float:
00079         '''!
00080         Сравнивает Dish.name и self.search_string с помощью SequenceMather()
00081         @param dish Dish
```

```
00082         @return sm_score float
00083         '''
00084         if dish is None:
00085             return 0
00086         if not dish.name or dish.name is None:
00087             return 0
00088         if not self.search_string:
00089             return 0
00090         s = SequenceMatcher(a=self.search_string, b=dish.name)
00091         sm_score = s.ratio()
00092         return sm_score
00093
00094
00095 class SearchProduct:
00096     '''!
00097     1. Загружаем из бд список всех спаршенных продуктов
00098     2. Вызов функции SequenceMatch() (взвешиваем каждый product.name относительно category.name по похожести)
00099     3. Сортируем продукты по убыванию весов и эффективности покупки
00100     @param category_id int
00101     @return products [[Product obj, ...], [Category.name, [image1, ...]]]]
00102     '''
00103
00104     def __init__(self, category_id, user_id, threshold=0.7) -> None:
00105         '''!
00106         @param category_id int
00107         '''
00108
00109         self.category = Category.objects.get(id=category_id)
00110         self.products = Product.objects.filter(user__id=user_id)
00111         self.threshold = threshold
00112
00113     def score(self, product) -> float:
00114         '''!
00115         Сопосталяет каждому объекту класса Product число, по которому будет осуществлена сортировка
00116         @param product Product
00117         @return score double
00118         '''
00119         sm_score = self.seq_match_scorer(product)
00120         return sm_score
00121
00122     def get(self) -> list:
00123         '''!
00124         Сортирует список объектов класса Product
00125         @return products list of Product
00126         '''
00127         new = sorted(self.products, key=self.score, reverse=True)
00128         for i in range(len(new)):
00129             if self.score(new[i]) < self.threshold:
00130                 return new[:i]
00131         return new
00132
00133     def seq_match_scorer(self, product) -> float:
00134         '''!
00135         Сравнивает Product.name и self.category.name с помощью SequenceMather()
00136         @param product Product
00137         @return sm_score float
00138         '''
00139         if product is None:
00140             return 0
00141         if not product.name or product.name is None:
00142             return 0
00143         if not self.category.name or self.category.name is None:
00144             return 0
00145         s = SequenceMatcher(a=self.category.name, b=product.name)
00146         sm_score = s.ratio()
00147         return sm_score
00148
00149     def get_category(self) -> Category:
00150         '''!
00151         Возвращает self.category
00152         @return category Category
00153         '''
00154         return self.category
```

7.25 serializers.py File Reference

Classes

- class bym.BYM_project.main_app.serializers.UserSerializer
- class bym.BYM_project.main_app.serializers.UserSerializer.Meta

Namespaces

- namespace bym
- namespace bym.BYM_project
- namespace bym.BYM_project.main_app
- namespace bym.BYM_project.main_app.serializers

7.26 serializers.py

Go to the documentation of this file.

```
00001 from rest_framework import serializers
00002 from .models import User
00003
00004
00005 class UserSerializer(serializers.ModelSerializer):
00006     class Meta:
00007         model = User
00008         fields = '__all__'
```

7.27 tests.py File Reference

Namespaces

- namespace bym
- namespace bym.BYM_project
- namespace bym.BYM_project.main_app
- namespace bym.BYM_project.main_app.tests

7.28 tests.py

Go to the documentation of this file.

```
00001 from django.test import TestCase
00002
00003 # Create your tests here.
```

7.29 views.py File Reference

Classes

- class bym.BYM_project.main_app.views.UserApiView
- class bym.BYM_project.main_app.views.HomePageApi
- class bym.BYM_project.main_app.views.SearchDishApi
- class bym.BYM_project.main_app.views.ChangeCartApi
- class bym.BYM_project.main_app.views.ParseProductsApi

Namespaces

- namespace bym
- namespace bym.BYM_project
- namespace bym.BYM_project.main_app
- namespace bym.BYM_project.main_app.views

Functions

- bym.BYM_project.main_app.views.HomePageView (request)
- bym.BYM_project.main_app.views.Dish (request, id)
- bym.BYM_project.main_app.views.FinalPage (request)
- bym.BYM_project.main_app.views.my_view (request)
- bym.BYM_project.main_app.views.update_counter (request)
- bym.BYM_project.main_app.views.parse_adress (request)

7.30 views.py

Go to the documentation of this file.

```
00001 from django.shortcuts import render
00002 from main_app.search import SearchDish
00003 from main_app.functions_backend import get_particular_dish, start_session
00004 from main_app.cart import Cart
00005 from main_app.models import User
00006 from main_app.parser import parse_by_address
00007 from django.http import JsonResponse, HttpResponseRedirect
00008 from django.views.decorators.csrf import csrf_exempt
00009 from rest_framework import generics
00010 from .models import User
00011 from .serializers import UserSerializer
00012 from rest_framework.views import APIView
00013 from rest_framework.response import Response
00014 from django.core import serializers
00015 import json
00016
00017
00018 def HomePageView(request):
00019     start_session(request)
00020
00021     user_id = User.get_user_id(request)
00022     cart_obj = Cart(user_id)
00023
00024     # if request.method == 'POST':
00025     #     action = request.POST.get("action")
00026     #     if action == "click_mest":
00027     #         parse_by_address(user_id)
00028
00029     try:
00030         search = request.GET.get('search')
00031     except:
00032         search = None
00033
00034     search_dish = SearchDish(search_string=search)
00035     object_list = search_dish.get(slice=12)
00036
00037     cart = cart_obj.get_order_array()
00038
00039     return render(request, 'landing_page.html', {'object_list': object_list,
00040                                                  'cart': cart,
00041                                                  'search': search})
00042
00043
00044 def Dish(request, id):
00045     start_session(request)
00046
00047     user_id = User.get_user_id(request)
00048     cart_obj = Cart(user_id)
00049
00050     # if request.method == 'POST':
00051     #     action = request.POST.get("action")
00052     #     if action == "click_mest":
00053     #         parse_by_address(user_id)
00054
00055     if request.method == 'GET':
00056         value = request.GET.get("add_dish")
00057         print("Action: ", value)
00058         if value == "Добавить в меню":
00059             cart_obj.change_order(id, True)
00060
00061     dish, cooking_stages, ingredients = get_particular_dish(id)
00062
00063     cart = cart_obj.get_order_array()
00064
00065     return render(request, 'page_of_dish.html', {'dish': dish,
```

```
00066                                           'cart': cart,
00067                                           'cooking_stages': cooking_stages,
00068                                           'ingredients': ingredients})
00069
00070
00071 def FinalPage(request):
00072     start_session(request)
00073
00074     user_id = User.get_user_id(request)
00075     cart_obj = Cart(user_id)
00076
00077     # if request.method == 'POST':
00078     #     action = request.POST.get("action")
00079     #     if action == "click_mest":
00080     #         parse_by_address(user_id)
00081
00082     categories_list = cart_obj.get_categories_list(slice=4)
00083
00084     return render(request, 'second_checkout.html', {'CatProd_list': categories_list})
00085
00086
00087
00088
00089 def my_view(request):
00090     count = request.POST.get('count', 0)
00091     #передать значение счётчика на бэк
00092     return render(request, 'vue.html')
00093
00094 @csrf_exempt
00095 def update_counter(request):
00096     if request.method == 'POST':
00097         count = int(request.POST.get('count', 0))
00098         # Do something with the count value here
00099         return JsonResponse({'success': True})
00100     else:
00101         return JsonResponse({'success': False, 'error': 'Invalid request method'})
00102
00103 # Функция ajax запроса нажатия на кнопку "Таллинская 34"
00104 def parse_adress(request):
00105     if request.method == 'POST':
00106         user_id = User.get_user_id(request)
00107         print("parser went to lavka")
00108         parse_by_address(user_id)
00109         return JsonResponse({'success': True})
00110     else:
00111         return JsonResponse({'success': False, 'error': 'Invalid request method'})
00112
00113
00114 # class UserApiView(APIView):
00115 #     def get(self, request):
00116 #         return Response({"})
00117
00118
00119 class UserApiView(generics.ListAPIView):
00120     queryset = User.objects.all()
00121     serializer_class = UserSerializer
00122
00123
00124 # для тестов
00125 class HomePageApi(APIView):
00126     def get(self, request):
00127         start_session(request)
00128
00129         user_id = User.get_user_id(request)
00130         cart_obj = Cart(user_id)
00131
00132         try:
00133             search = request.GET.get('search')
00134         except:
00135             search = None
00136
00137         search_dish = SearchDish(search_string=search)
00138         object_list = search_dish.get(slice=12)
00139
00140         cart = cart_obj.get_order_array()
00141
00142         return Response({'cart': json.loads(serializers.serialize('json', cart)),
00143                          'object_list': json.loads(serializers.serialize('json', object_list)),
00144                          'search': search,
00145                          'shit' : 'sheeesh'})
00146
00147 # возвращает на frontend json с блюдами по поисковой строке
00148 class SearchDishApi(APIView):
00149     def post(self, request):
00150         search_string = request.data['searchString']
00151         if search_string == '':
00152             search_string = None
```

```
00153
00154         search_dish = SearchDish(search_string=search_string)
00155         object_list = search_dish.get(slice=12)
00156
00157         return Response({'object_list': json.loads(serializers.serialize('json', object_list))})
00158
00159 # изменяет корзину блюд пользователя и возвращает ее на frontend в виде json
00160 class ChangeCartApi(APIView):
00161     def post(self, request):
00162         cart_change = request.data['cartChange']
00163
00164         # start_session(request)
00165         # user_id = User.get_user_id(request)
00166         cart_obj = Cart(1)
00167
00168         if cart_change['dish_pk'] != 0:
00169             cart_obj.change_order(cart_change['dish_pk'], cart_change['add_flg'])
00170         order_array = cart_obj.get_order_array()
00171
00172         cart = []
00173         for x in order_array:
00174             cart.append({'dish_pk': x.dish.pk, 'name': x.dish.name, 'amount': x.amount})
00175
00176         return Response(cart)
00177
00178 class ParseProductsApi(APIView):
00179     def post(self, request):
00180         address = request.data['address']
00181         # user_id = User.get_user_id(request)
00182         # print("parser went to lavka")
00183         city_street = ', '.join([address['city'], address['street']])
00184         number = address['number']
00185         parse_by_address(1, city_street, number)
00186
00187         return Response({'cringe': 'shit'})
```

Предметный указатель