|  |  |
| --- | --- |
| Название класса | Описание класса |
| [KBS](#kbs) | Данный класс служит для составление, обработки и создания удобного вида данных для передачи функции, отвечающей за нормализацию данных. Класс оборачивает данные в списки и реализует карту с геометрией. Также класс служит для пространственного анализа данных в сфере туризма, в частности, для оценки условий размещения бизнеса в зависимости от различных факторов, таких как близость маршрутов, наличие достопримечательностей и природоохранные ограничения. |
| [ConnectionBD](#ConnectionBD) | Этот класс отвечает за взаимодействие с базой данных MySQL, обработку и фильтрацию данных, необходимых для пространственного анализа туристических объектов. |
| [ReviewProcessor](#ReviewProcessor) | Данный класс обрабатывает текстовые отзывы, извлекает ключевые плюсы и минусы. |
| [DataUpdater](#DataUpdater) | Класс обновляет информацию в базе данных, улучшая анализ пользовательских отзывов. |
| [ZoningOlkhon](#Zoning) | Загружает геоданные (GeoJSON) с информацией о зонировании острова Ольхон.  Преобразует систему координат данных в EPSG:4326 (географическая система координат).  Получает координаты острова Ольхон через OSMnx для центрирования карты.  Создает карту с помощью folium и добавляет на нее полигоны, представляющие различные зоны.  Назначает каждой зоне цвет из заранее определенной схемы (color\_map).  Добавляет всплывающие подсказки (tooltip) с информацией о рекомендуемых видах туризма в данной зоне.  Возвращает интерактивную карту. |
| [Density](#Density) | Класс предназначен для визуализации плотности объектов на карте с использованием сетки H3. Он позволяет анализировать и отображать различные объекты, такие как места общественного питания, точки размещения и достопримечательности. |
| [Analyze](#analyze) | Класс отвечает за нормализацию данных и реализации анализа с помощью функции определяющей корреляцию данных |
| [Report](#report) | Данный класс генерирует отчет на основе критериев и результатов. |
| [App](#app) | Класс App управляет созданием карт с фильтрацией данных по типу бизнеса, генерирует карты зон и плотности объектов, а также предоставляет API для обновления карт через веб-интерфейс с использованием Flask. |
| Form | Интерфейс Form определяет метод display\_data(), который выводит данные формы. Этот метод не принимает аргументов и не возвращает значений, просто отображая информацию, связанную с объектом формы. |

**Атрибуты KBS**

|  |  |  |  |
| --- | --- | --- | --- |
| **Атрибут** | **Тип данных** | **Возможные значения** | **Описание** |
| type\_object | str | "общественное питание", "места размещения" | Тип бизнеса, для которого выполняется анализ. |
| size\_poligon | int | 7, 8, 9 | Уровень детализации H3-ячейки (меньшее значение — более крупные ячейки, большее — более детальная сетка). |

**Операции KBS**

|  |  |  |
| --- | --- | --- |
| **Операция** | **Параметры** | **Описание** |
| load\_routes() -> GeoDataFrame | *Нет параметров* | Загружает маршруты из файла routes\_Baikal.geojson и возвращает их в виде GeoDataFrame. |
| calculate\_distance\_to\_routes(polygon: Polygon, routes\_gdf: GeoDataFrame) -> float | - polygon: Polygon — полигон H3, для которого вычисляется расстояние.  - routes\_gdf: GeoDataFrame — данные о маршрутах. | Вычисляет минимальное расстояние от центра полигона до ближайшего маршрута. |
| calculate\_landmarks\_within\_radius(polygon: Polygon, df\_landmark: DataFrame, radius: float) -> int | - polygon: Polygon — полигон H3, для которого проводится расчет.  - df\_landmark: DataFrame — данные о достопримечательностях (столбцы lat, lng).  - radius: float — радиус поиска в метрах. | Подсчитывает количество достопримечательностей в пределах указанного радиуса от центра полигона. |
| calculate\_degree\_landshaft\_zone(polygon: Polygon, gdfVec: GeoDataFrame) -> int | - polygon: Polygon — анализируемый полигон.  - gdfVec: GeoDataFrame — данные о природоохранных зонах. | Возвращает уровень природоохранных ограничений на территории полигона (1 — строгие, 2 — средние, 3 — нет ограничений). |
| create\_geometry(df: DataFrame, size\_poligon: int, full\_hex: DataFrame) -> DataFrame | - df: DataFrame — данные о бизнес-объектах (содержит координаты lat, lng).  - size\_poligon: int — уровень детализации H3.  - full\_hex: DataFrame — полная сетка ячеек H3. | Создает геометрические объекты H3, группирует объекты по ячейкам, считает количество объектов в каждой ячейке. |
| calculate\_kbs\_risk\_assessment(risk\_factors: List[Tuple[float, float]], obj\_hex1: DataFrame) -> DataFrame | - risk\_factors: List[Tuple[float, float]] — список весов для расчета КБС.  - obj\_hex1: DataFrame — данные с геометрией и характеристиками объектов. | Рассчитывает коэффициент благоприятствования с учетом факторов риска. |
| filter\_type(gdf: GeoDataFrame, business: str) -> str | - gdf: GeoDataFrame — границы территории анализа.  - business: str — выбранный тип бизнеса ("общественное питание", "места размещения"). | Фильтрует данные по выбранному типу бизнеса, строит карту КБС и сохраняет в kbs\_map.html. |

**Атрибуты ConnectionBD**

|  |  |  |  |
| --- | --- | --- | --- |
| **Атрибут** | **Тип данных** | **Возможные значения** | **Описание** |
| connection | Connection | Экземпляр соединения с БД | Подключение к базе данных MySQL через pymysql. |

**Операции ConnectionBD**

|  |  |  |
| --- | --- | --- |
| **Операция** | **Параметры** | **Описание** |
| execute(query: str) -> ResultSet | - query: str — SQL-запрос. | Выполняет SQL-запрос и возвращает результат. |
| select\_pl() -> list | *Нет параметров* | Извлекает данные о местах размещения (placement\_location\_olkhon\_test), включая ID, координаты, название, преимущества, недостатки, минимальную цену и общий рейтинг. |
| select\_caterings() -> list | *Нет параметров* | Извлекает данные об объектах общественного питания (catering\_olkhon), включая ID, координаты, название, тип, преимущества, недостатки, среднюю цену, кухню и рейтинг. |
| select\_sights() -> list | *Нет параметров* | Извлекает данные о достопримечательностях (sights\_olkhon), включая ID, координаты, название и тип. |
| filter\_data() -> list | *Нет параметров* | Фильтрует данные объектов по географическим границам острова Ольхон. |

**Атрибуты ReviewProcessor**

|  |  |  |  |
| --- | --- | --- | --- |
| **Атрибут** | **Тип данных** | **Возможные значения** | **Описание** |
| review | str | Текстовый отзыв | Отзыв, который требуется обработать. |
| positive\_words | Set[str] | Набор положительных слов | Список слов, указывающих на положительные характеристики (например, "вкусный", "уютный"). |
| negative\_words | Set[str] | Набор отрицательных слов | Список слов, относящихся к негативным аспектам (например, "грязный", "дорого"). |

**Операции ReviewProcessor**

|  |  |  |
| --- | --- | --- |
| **Операции** | **Параметры** | **Описание** |
| review\_processin(review: str, negative\_words: Set[str], positive\_words: Set[str]) -> (List[str], List[str]) | - review: str — отзыв. - negative\_words: Set[str] — негативные слова. - positive\_words: Set[str] — позитивные слова. | Анализирует отзыв, используя токенизацию, стемминг и определение частей речи. Возвращает списки плюсов и минусов. |

**Атрибуты DataUpdater**

|  |  |  |  |
| --- | --- | --- | --- |
| **Атрибут** | **Тип данных** | **Возможные значения** | **Описание** |
| connection | Connection | Экземпляр соединения с БД | Подключение к базе данных MySQL через pymysql. |

**Операции DataUpdater**

|  |  |  |
| --- | --- | --- |
| **Операции** | **Параметры** | **Описание** |
| add\_cons\_pros(query\_select\_text: str, connection: pymysql.Connection, table\_review: str) -> None | - query\_select\_text: str — SQL-запрос для выборки отзывов. - connection: pymysql.Connection — соединение с базой данных. - table\_review: str — таблица с отзывами. | Обрабатывает отзывы и обновляет соответствующие плюсы и минусы в базе данных. |
| get\_top\_phrases(text\_list: List[str], top\_n: int = 3, min\_count: int = 2) -> str | - text\_list: List[str] — список текстов плюсов или минусов. - top\_n: int (по умолчанию 3) — количество часто встречающихся фраз. - min\_count: int (по умолчанию 2) — минимальное количество повторений фразы. | Выбирает наиболее часто встречающиеся фразы из отзывов и возвращает их в виде строки. |
| update\_pros\_cons(connection: pymysql.Connection, table: str, table\_r: str, id: str)：void | - connection: pymysql.Connection — соединение с базой данных. - table: str — таблица с объектами (например, catering\_olkhon). - table\_r: str — таблица с отзывами (например, reviews\_caterings). - id: str — идентификатор объекта. | Анализирует отзывы, выбирает часто встречающиеся плюсы и минусы и обновляет таблицу с объектами. |

#### ****Атрибуты ZoningOlkhon****

|  |  |  |
| --- | --- | --- |
| **Атрибут** | **Тип данных** | **Описание** |
| gdfVec | GeoDataFrame | Геоданные, содержащие информацию о зонировании острова Ольхон. |

#### ****Операции ZoningOlkhon****

|  |  |  |
| --- | --- | --- |
| **Метод** | **Параметры** | **Описание** |
| zone\_olkhon() -> Map | - | Создает интерактивную карту зонирования острова Ольхон с информацией о рекомендуемых видах туризма. |

#### ****Атрибуты класса**** Density

|  |  |  |
| --- | --- | --- |
| **Атрибут** | **Тип данных** | **Описание** |
| type\_object | str | Тип объекта (например, рестораны, отели, достопримечательности). |
| size\_poligon | int | Размер ячеек сетки H3 для агрегации объектов. |
| zoom | int | Уровень приближения карты при загрузке. |
| type\_business | str | категория бизнеса : места размещения, общественное питание |
| gdf | GeoDataFrame | Геодатафрейм с границами территории для анализа. |

#### ****Методы класса**** Density

|  |  |  |
| --- | --- | --- |
| **Метод** | **Возвращаемый тип** | **Описание** |
| create\_geometry(df, size\_poligon, full\_hex) | DataFrame | Создает геометрические объекты сетки H3, агрегирует количество объектов и формирует DataFrame с полигонами. |
| get\_color(value, min\_val, max\_val) | str | Определяет цвет полигона в зависимости от плотности объектов, используя логарифмическую шкалу. |
| add\_legend(map\_object) | void | Добавляет легенду на карту для пояснения цветового кодирования плотности. |
| main(df, gdf) | Map | Формирует интерактивную карту плотности объектов, добавляет полигоны и легенду. |
| markers\_obj(map, df) | Map | Добавляет маркеры объектов на карту с всплывающими подсказками. |
| density\_map\_function(gdf, type\_obj, type\_business, price, rating, kitchen) | str | Фильтрует данные по заданным параметрам и вызывает основную функцию построения карты. |

#### ****Методы Analyze****

|  |  |  |
| --- | --- | --- |
| **Метод** | **Возвращаемый тип** | **Описание** |
| minmax\_normalize\_data(data) | DataFrame | Выполняет нормализацию данных методом минимакс (Min-Max Scaling). |
| corr\_data(data) | DataFrame | Вычисляет корреляцию между переменными в переданных данных. |

#### ****Атрибуты Report****

|  |  |  |
| --- | --- | --- |
| **Атрибут** | **Тип данных** | **Описание** |
| report\_title | String | Заголовок отчета. |
| criteria | Dictionary<String, Any> | Критерии, используемые в анализе. |
| results | Dictionary<String, Any> | Результаты анализа. |

#### ****Методы Report****

|  |  |  |
| --- | --- | --- |
| **Метод** | **Возвращаемый тип** | **Описание** |
| generate\_report() | void | Генерирует отчет на основе критериев и результатов. |

**Атрибуты App**

|  |  |  |
| --- | --- | --- |
| **Атрибут** | **Тип данных** | **Описание** |
| business\_type | str | Тип бизнеса, используемый для фильтрации объектов, отображаемых на карте. |
| type\_map | None | Атрибут, который не используется в коде, возможно предназначен для хранения типа карты или других данных, связанных с картой. |

**Методы App**

|  |  |  |
| --- | --- | --- |
| **Метод** | **Возвращаемый тип** | **Описание** |
| create\_map(business\_type) | str | Создаёт карту для выбранного типа бизнеса, фильтруя данные с помощью функции filter\_type. Возвращает путь к файлу с картой, который сохраняется в директории static. |
| create\_report(bussiness\_type, criteria) | None | Метод, который должен генерировать отчёт, но в данном коде не реализован. |