|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Класс** | **Атрибуты** | **Назначение атрибутов** | **Операции** | **Назначение операций** |
| **KBS** | type\_object: str | Тип объекта (например, место размещение.) | load\_routes() | Загрузка маршрутов в виде GeoDataFrame |
|  | size\_poligon: int | Размер полигона для анализа | calculate\_distance\_to\_routes(polygon, routes\_gdf): float | Рассчитывает расстояние до маршрутов |
|  |  |  | calculate\_landmarks\_within\_radius(polygon, df\_landmark, radius): int | Подсчитывает достопримечательности в заданном радиусе |
|  |  |  | calculate\_degree\_landshaft\_zone(polygon, gdfVec): int | Функция для определения степени природоохранных ограничений по пересечению |
|  |  |  | create\_geometry(df, size\_poligon, full\_hex): DataFrame | Создает геометрические объекты |
|  |  |  | calculate\_kbs\_risk\_assessment(risk\_factors, obj\_hex1): DataFrame | Оценивает риски |
|  |  |  | filter\_type(gdf, business): str | Фильтрует объекты по типу бизнеса |
| **ConnectionDB** | connection: Connection | Соединение с базой данных | execute(query: str): ResultSet | Выполняет SQL-запрос |
|  |  |  | select\_pl(): list | Выбирает данные по достопримечательностям |
|  |  |  | select\_caterings(): list | Выбирает данные по заведениям питания |
|  |  |  | select\_sights(): list | Выбирает данные по туристическим объектам |
|  |  |  | filter\_data(): list | Фильтрует данные |
| **ReviewProcessor** | review: str | Текст отзыва | processReview(review: str): (List[str], List[str]) | функция для обработки тональности отзывов |
|  | positive\_words: Set[str] | Набор положительных слов |  |  |
|  | negative\_words: Set[str] | Набор отрицательных слов |  |  |
| **DataUpdater** | connection: Connection | Соединение с базой данных | addProsCons(reviews: List[(int, str)], table\_name: str): void | Добавляет положительные и отрицательные аспекты в БД |
|  |  |  | updateTopPhrases(reviews: DataFrame, table\_name: str, column\_name: str): void | Обновляет топ-фразы на основе отзывов |
| **ZoningOlkhon** | gdfVec: GeoDataFrame | Геоданные зоны Ольхона | zone\_olkhon(): Map | Создает карту зонирования Ольхона |
| **Density** | type\_object: str | Тип анализируемого объекта | create\_geometry(df, size\_poligon, full\_hex): DataFrame | Генерация геометрии полигонов |
|  | size\_poligon: int | Размер полигона | get\_color(value, min\_val, max\_val): str | Определяет цветовую схему |
|  | zoom: int | Масштаб карты | add\_legend(map\_object): void | Добавляет легенду на карту |
|  | type\_business: str | Тип бизнеса | main(df, gdf): Map | Генерирует карту плотности |
|  | gdf: GeoDataFrame | Геоданные | markers\_obj(map, df): Map | Добавляет маркеры объектов на карту |
| **Analyze** |  |  | minmax\_normalize\_data(data): DataFrame | Нормализация данных по методу Min-Max |
|  |  |  | corr\_data(data): DataFrame | Анализ корреляции данных |
| **Report** | report\_title: String | Заголовок отчета | generate\_report(): void | Генерирует отчет по анализу |
|  | criteria: Dictionary<String, Any> | Критерии анализа |  |  |
|  | results: Dictionary<String, Any> | Результаты анализа |  |  |
| **App** | business\_type: str | Тип бизнеса | create\_map(business\_type, type\_map) | Создает карту с данными |
|  | type\_map | Тип карты (например, плотность, маршруты) | create\_report(business\_type, criteria) | Создает отчет по бизнес-анализу |
| **Form (интерфейс)** |  |  | display\_data(): void | Отображает данные на интерфейсе |