ChatGPT

Класс/Интерфейс	Модуль	Назначение	Методы

`PipelineBase` (ABC)	`src/bioetl/core/pipeline_base.py`	Абстрактный базовый класс табличного ETL-пайплайна. Оркеструет стадии `extract→transform→validate→write`, обеспечивая единый интерфейс, логирование, валидацию (через Pandera) и атомарную запись результатов [1] [2]. Также централизованно управляет ресурсами (например, закрывает подключения) и поддерживает `dry_run` режим без записи [3] [4].	`run(outpu`
			`dry_run: b`
			`RunResult`
			пайплайна (
			`transform`
			необходимо
			времени и р
			результат за
			[6].
			`ex`
			`pd.DataFra`
			сырые данн
			API, файлы и
			DataFrame
			`pd.DataFra`
			Абстрактны
			трансформа
			обогащение
			валидацией
			[8].
			`va`
			`pd.DataFra`
			структуру да
			Pandera-схем
			свойство ли
			DataFrame, в
			отсутствии с
			упорядочить
			добавить хэ
			строк [9] [10]
			`pd.DataFra`
			`WriteResul`
			заданным ко
			записывает
			Parquet по р
			успешной за
			запуска (нап
			информацие
			количестве
			[12].
			`re`
			`client: An`
			внешний ре
			реестре для
			[13].
			`cl`
			Закрывает в
			ресурсы, вы
			`dispose()`
			но не преры

Класс/Интерфейс	Модуль	Назначение	Методы
`StageABC`	`src/bioetl/core/pipeline/base.py`	Абстракция отдельной стадии пайплайна. Определяет интерфейс выполнения шага и освобождения ресурсов, обеспечивая изоляцию логики этапа и возможность композиции этапов внутри пайплайна [2].	`run(contex` логику стади данные и во передачи сл реализация этапа. `c` ресурсы, исп закрывает с завершения использовать менеджер (в
`PipelineHookABC`	`src/bioetl/core/pipeline/hook.py`	Интерфейс хуков конвейера для мониторинга. Получает события начала и окончания стадий, а также ошибки, не вмешиваясь в логику, и позволяет подключить логирование, метрики или другую обработку событий пайплайна [19].	`on_stage_s` `None`: Вызы выполнения логировани этапа. `c` `StageABC)` успешном за фиксировать выполнения `StageABC,` Вызывается – позволяет реализовать
`CLICommandABC`	`src/bioetl/core/pipeline/cli_command.py`	Интерфейс для интеграции пайплайна с CLI. Определяет контракты для регистрации CLI-команды и запуска пайплайна с переданными параметрами конфигурации, обеспечивая единообразный способ запуска из командной строки [19].	`register_c` Регистриру в CLI-прило `click` / `typ` контейнер команд. `PipelineCo` Инициализи данной конф `PipelineBa` выполнения

Класс/Интерфейс	Модуль	Назначение	Методы
`SourceClientABC`	`src/bioetl/core/pipeline/source/client.py`	Интерфейс клиента внешнего источника данных. Описывает unified API для извлечения данных из внешних систем (REST API, баз данных и т.п.), скрывая детали HTTP/DB запросов. Поддерживает одиночные и пагинированные выборки данных [20] [21] .	`fetch_one` `Record | N` возвращает или `None` , [23] . `fe` `ClientRequ` Возвращает полученным разбиение [25] . `it` `ClientRequ` Возвращает данных (каж записей и и [26] [27] . `me` `Any]` : Предо (например, маршруты) Закрывает с клиента, есл сессии HTTP

Класс/Интерфейс	Модуль	Назначение	Методы
`BaseExternalDataClient`	`src/bioetl/core/pipeline/source/client.py`	Базовый класс для клиентов внешних API, реализующий `SourceClientABC`. Содержит общую логику пагинации и сбор данных из REST API (циклический запрос страниц, объединение параметров и т.п.) [32]. Конкретные клиенты (ChEMBL, PubChem и др.) наследуются от этого класса, переопределяя при необходимости специфичные части.	`fetch_one` `Record | N` возвращает страницы ре ничего не по [33]. `fe` `ClientRequ` Итеративно используя `i` постранично [25]. `it` `ClientRequ` Выполняет з результатов параметр сл (`page_para` [34]. `me` `Any]`: Возвр (базовый UR [29]. `cl` используем `RequestsH` определен м
`RateLimiterABC`	`src/bioetl/core/pipeline/source/client.py`	Интерфейс ограничителя частоты запросов. Обеспечивает соблюдение лимитов нагрузки при обращении к внешнему API (например, не более N запросов в секунду), чтобы избежать превышения квот или блокировок [20].	`acquire()` приостанавл пока не наст следующего заданным л время). внутренние (например, окна времен

Класс/Интерфейс	Модуль	Назначение	Методы
`RetryPolicyABC`	`src/bioetl/core/pipeline/source/client.py`	Интерфейс политики повторных попыток запроса. Определяет, в каких случаях и сколько раз повторять запрос при ошибках, а также с каким интервалом (включая стратегию экспоненциального backoff) [20] .	`should_re` `attempt: i` следует ли в при данном попытке (во повторить).< `-> float` : E секундах) пе исходя из но реализации экспоненциа
`RequestBuilderABC`	`src/bioetl/core/pipeline/source/request_builder.py`	Интерфейс построения запросов к внешнему источнику. Отвечает за формирование объекта запроса (`ClientRequest`) на основе входных параметров бизнес-логики (например, идентификаторов, фильтров), инкапсулируя логику формирования URL и параметров для клиента [20] .	`build_requ` `Any], page` `-> ClientR` объект запр (`ClientReq` параметры страницы. Р API (маршру формирует з знания дета
`ResponseParserABC`	`src/bioetl/core/pipeline/source/parser.py`	Интерфейс парсера отклика от внешнего источника. Определяет преобразование сырых данных ответа (обычно JSON) в стандартные записи (словарь `Record`) для последующей обработки в пайплайне [20] . Отделяет логику разбора и нормализации ответа API от логики его получения.	`parse(raw_` `Iterator[R` (например, J итеративно записи `Rec` извлечение типов и стру позволяет о стримингом
`PaginatorABC`	`src/bioetl/core/pipeline/source/parser.py`	Интерфейс логики пагинации для разбора ответов. Описывает способ извлечения списка элементов и маркера продолжения из полученного ответа API, абстрагируя различия в форматах пагинации у разных источников [20] .	`extract_it` `list[Recor` ответа списо страницы. Н вложенный JSON-ответе элемент в `R` формата.<br `-> str | N` маркер или страницу (на `cursor` , но заданному к дальнейших

Класс/Интерфейс	Модуль	Назначение	Методы
`TransformerABC`	`src/bioetl/core/pipeline/processing/transform.py`	Интерфейс трансформации данных. Определяет детерминированные преобразования входного DataFrame без взаимодействия с внешними системами – например, очистку значений, переименование колонок, приведение типов – перед дальнейшим обогащением и валидацией [43].	`transform` `pd.DataFra` чисто функц изменяя вне новый DataF данными. Га одинаковом идентичный (детерминир
`LookupEnricherABC`	`src/bioetl/core/pipeline/processing/enrich.py`	Интерфейс обогащения данных с помощью внешних справочных данных (lookup). Отвечает за добавление в таблицу недостающей информации из внешних источников/словарей (например, подтягивает данные по внешним ключам) [43]. Использует поставщика побочных данных (`SideInputProviderABC`) для доступа к справочникам.	`enrich(df` `SideInputP` `pd.DataFra` DataFrame, и данные из добавлять н значения ид данными из новый DataF информацие
`SideInputProviderABC`	`src/bioetl/core/pipeline/processing/enrich.py`	Интерфейс поставщика сторонних (побочных) данных для обогащения. Абстрагирует доступ к справочникам, кэшам или внешним источникам, которые используются в процессах обогащения (lookup) для получения дополнительной информации [43].	`get_data(` Предоставля заданному к Например, м словарь с да относящими таблице, есл может осуще файла, БД, А подготовлен
`DeduplicatorABC`	`src/bioetl/core/pipeline/processing/dedup.py`	Интерфейс для выявления и удаления дубликатов в данных. Определяет контракт консолидации записей с одинаковыми бизнес-ключами и устранения повторов перед загрузкой данных, обеспечивая идемпотентность пайплайна [43].	`deduplicat` `pd.DataFra` записи в Dat определенн совпадения DataFrame, и (при необхо информации стратегии сл использует вычисления хэшировани `MergeStra` дубликатов.

Класс/Интерфейс	Модуль	Назначение	Методы
`BusinessKeyDeriverABC`	`src/bioetl/core/pipeline/processing/dedup.py`	Интерфейс вычисления бизнес-ключа записи. Бизнес-ключ – человечески осмысленный уникальный идентификатор сущности (например, комбинация нескольких полей), используемый для определения дубликатов независимо от технических первичных ключей [44] [45].	`derive_key` Вычисляет б (словаря) на подмножест Например, д ключом мож название+вы строку, одно бизнес-сущн
`HasherABC`	`src/bioetl/core/pipeline/processing/dedup.py`	Интерфейс хеширования записи. Определяет способ вычисления детерминированного хэша (например, MD5, SHA) по содержимому записи для целей идентификации и сравнения записей [46] [45]. Хэши применяются для отслеживания изменений и проверки идемпотентности.	`hash(reco` Вычисляет к контрольны Гарантирует содержанию хэш. Реализа хешировани строка).
`MergeStrategyABC`	`src/bioetl/core/pipeline/processing/dedup.py`	Интерфейс стратегии слияния дубликатов. Определяет, как несколько записей, признанных дубликатами, объединяются в одну “консолидированную” запись перед загрузкой, чтобы информация не терялась [47].	`merge(rec` `Record`: Об дубликатов реализация записей бра предпочесть максимальн объединять запись, соде данные.
`SchemaProviderABC`	`src/bioetl/core/pipeline/validation/schema.py`	Интерфейс поставщика схемы данных. Предоставляет описание структуры и типов данных, ожидаемых на этапе валидации, абстрагируя конкретную реализацию схем (Pandera, Pydantic и т.п.) от остальной логики пайплайна [45].	`get_schema` объект схем `pa.DataFra` аналог из др данных, обр Конкретные схему прогр получать из

Класс/Интерфейс	Модуль	Назначение	Методы
`ValidatorABC`	`src/bioetl/core/pipeline/validation/validator.py`	Интерфейс валидатора данных. Обобщает применение схемы и правил качества данных (DQ) к результирующему DataFrame, формируя отчет о найденных нарушениях. Позволяет отделить проверку данных от бизнес-логики пайплайна [45] .	`validate(d` Validation на соответст правилам ка `SchemaProv` и набор `DQ` проверок (на обязательно результата в сведения о п проверках (н ошибок) [45]
`DQRuleABC`	`src/bioetl/core/pipeline/validation/dq_rules.py`	Интерфейс правила качества данных (Data Quality Rule). Определяет отдельное проверочное правило, не выражаемое лишь схемой (например, межколоночные зависимости, допустимые диапазоны значений, уникальность и пр.). Выделение DQ-правил позволяет гибко расширять проверки качества без изменения схемы [48] .	`check(df: list[DQIss` качества к D обнаруженн пустой списо проверку. Ка проблему (н допустимого включает сс Реализация проверять д результаты.
`PathStrategyABC`	`src/bioetl/core/pipeline/output/path_strategy.py`	Интерфейс стратегии формирования путей вывода. Определяет, как строятся пути и имена файлов для результатов пайплайна, чтобы обеспечить детерминизм и организованность вывода (например, единообразное именование по сущности, источнику, дате) [49] .	`make_path( str, base_` Генерирует для данного название па запуска и ба может добав подпапки по входных зна одинаковый
`WriterABC`	`src/bioetl/core/pipeline/output/writer.py`	Интерфейс компонента записи данных. Отвечает за непосредственную запись итогового DataFrame на диск (в файл) с обеспечением атомарности и соблюдением форматов. Инкапсулирует детали сортировки, выбора формата (CSV/Parquet) и предотвращения частично записанных результатов [49] .	`write(df: > None` : Зап DataFrame в должна: отсо столбцам (за пайплайна), расширение `.parquet`) имя и затем атомарности записи долж

Класс/Интерфейс	Модуль	Назначение	Методы
`MetadataWriterABC`	`src/bioetl/core/pipeline/output/writer.py`	Интерфейс записи метаданных запуска. Отвечает за сохранение вспомогательной информации о выполнении пайплайна – статистики и параметров запуска – в отдельный файл (например, `meta.yaml` рядом с данными) [49] .	`write_meta` `meta_path:` метаданные файл. Реали необходиму идентифика время начал этапов, коли флаг `dry_r` файла, хэш-н человекочит YAML или JS
`ConfigResolverABC`	`src/bioetl/core/pipeline/utils/config.py`	Интерфейс загрузки и разрешения конфигурации пайплайна. Инкапсулирует логику чтения конфигурационных файлов (например, YAML), слияния с дефолтами, применения переменных окружения и других преобразований, обеспечивая единый способ получения настроек для запуска пайплайна [51] .	`resolve(co` `PipelineCo` пайплайна п Реализация конфигурац `<config_na` возвращать Также может секретов и з `SecretProv` валидации с проверки.
`SecretProviderABC`	`src/bioetl/core/pipeline/utils/config.py`	Интерфейс доступа к секретам (пароли, токены API и др.). Предоставляет абстракцию над хранилищем чувствительных данных (например, переменными окружения, Vault, AWS Secrets Manager), позволяя безопасно подключать секреты в конфигурацию пайплайна [51] .	`get_secret` Возвращает имени/ключ источник (на окружения с обращение и обеспечив напрямую в может генер
`CacheABC`	`src/bioetl/core/pipeline/utils/cache.py`	Интерфейс кэша для промежуточных данных. Предоставляет унифицированный доступ к механизму сохранения и повторного использования данных между запусками или между этапами (например, кэширование результатов запросов к API, чтобы не дублировать вызовы) [51] .	`get(key: s` из кэша по к такой запис `value: Any` заданный об (перезапись существовал определять размера и п

Класс/Интерфейс	Модуль	Назначение	Методы
`LoggerAdapterABC`	`src/bioetl/core/pipeline/utils/logging.py`	Интерфейс адаптера логирования. Обеспечивает унифицированный способ структурированного логирования событий пайплайна, не зависящий от конкретной библиотеки логирования. Позволяет привязывать контекст (например, `run_id`, имя пайплайна) и логировать события пайплайна консистентно [51].	`bind(**fie` `LoggerAdap` логгер/адап контекстным `run_id` для сообщений) `**kwargs:` информацио названием/с полями данн значение в `str, **kwa` событие уро фиксирован ситуаций, со данными.
`TracerABC`	`src/bioetl/core/pipeline/utils/tracing.py`	Интерфейс трассировки выполнения. Позволяет интегрировать пайплайн с системами трассировки (distributed tracing) или собственным механизмом отслеживания выполнения этапов. Предоставляет методы для обозначения начала/конца важных операций, что упрощает отладку и мониторинг производительности [51].	`start_span` Отмечает на или этапа с з трассировоч например, с распределен трассировке `None`: Отме операции. И продолжите Реализация `start_span` контексту.
`ErrorPolicyABC`	`src/bioetl/core/pipeline/utils/error.py`	Интерфейс политики обработки ошибок. Определяет стратегию реагирования на исключения, возникающие во время выполнения пайплайна: продолжать выполнение, пропускать текущий элемент, повторять попытку или прерывать пайплайн [52]. Позволяет отделить логику обработки сбоев от основного кода пайплайна.	`on_error(s` `Exception)` оркестратор на стадии `s` `ErrorActio` дальше (нап текущую заг конвейер). Р исключения этапа и друг политики.

Класс/Интерфейс	Модуль	Назначение	Методы
`ProgressReporterABC`	`src/bioetl/core/pipeline/utils/progress.py`	Интерфейс агрегированного репортинга прогресса пайплайна. Собирает и регистрирует статистику по ходу выполнения: количество успешно обработанных записей, количество отброшенных/невалидных, причины отбраковки и пр. – без влияния на сам процесс обработки [45] [53]. Разделяет сбор статистики от бизнес-логики, предоставляя единый источник правды о ходе выполнения.	`on_stage_` Фиксирует н (например, логирует ста этапа). `str) -> No` обработанн (вызывается обработки, валидатором указанием причины).<b `str) -> No` стадии, окон (время, коли отброшенны и сохранены

[1] [3] [4] [5] [6] [7] [8] [9] [10] [11] [12] [13] [14] [15] [16] [17] [18] [50] PipelineBase_description.md
https://github.com/SatoryKono/bioactivity_data_acquisition/blob/98f17f432532cddd56d0a07fd4796b37c54677ec/_docs/PipelineBase_description.md

[2] [20] [43] [44] [45] [46] [49] [51] architecture_plan.md
https://github.com/SatoryKono/bioactivity_data_acquisition/blob/98f17f432532cddd56d0a07fd4796b37c54677ec/_docs/architecture_plan.md

[19] [47] [48] [52] abstract_objects_implementation_plan.md
https://github.com/SatoryKono/bioactivity_data_acquisition/blob/98f17f432532cddd56d0a07fd4796b37c54677ec/_docs/abstract_objects_implementation_plan.md

[21] [22] [23] [24] [25] [26] [27] [28] [29] [30] [31] [32] [33] [34] [35] [36] [37] [38] [39] [40] [41] [42] client_abc.py
https://github.com/SatoryKono/ChEMBL_data_acquisition/blob/14d2538df65b6d02dac4cb4703ee7ba34b501e32/src/bioetl/clients/base/client_abc.py

[53] ProgressReporterABC.md
https://github.com/SatoryKono/bioactivity_data_acquisition/blob/98f17f432532cddd56d0a07fd4796b37c54677ec/_docs/ProgressReporterABC.md