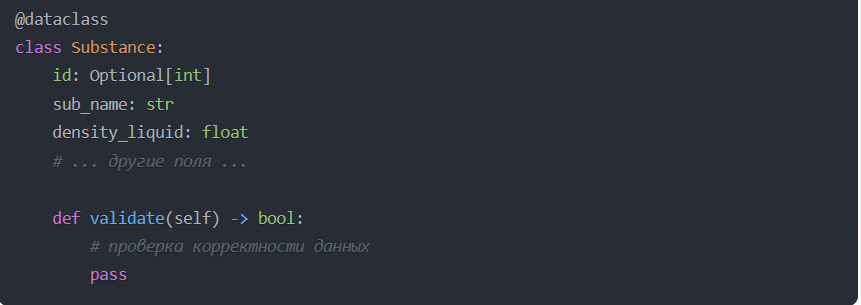


Разберем по порядку иерархию вызовов. Начнем с example\_usage.py, который является точкой входа:

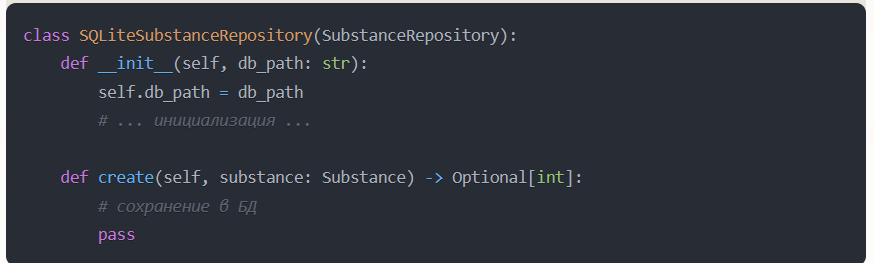
1. example\_usage.py вызывает (импортирует):
   * domain/models/substance.py (для создания объекта Substance)
   * infrastructure/database/sqlite/substance\_repository.py (для работы с БД)

Рассмотрим второй уровень - это наши основные классы, которые импортируются в example\_usage.py:

1. domain/models/substance.py - это класс, описывающий модель данных вещества:

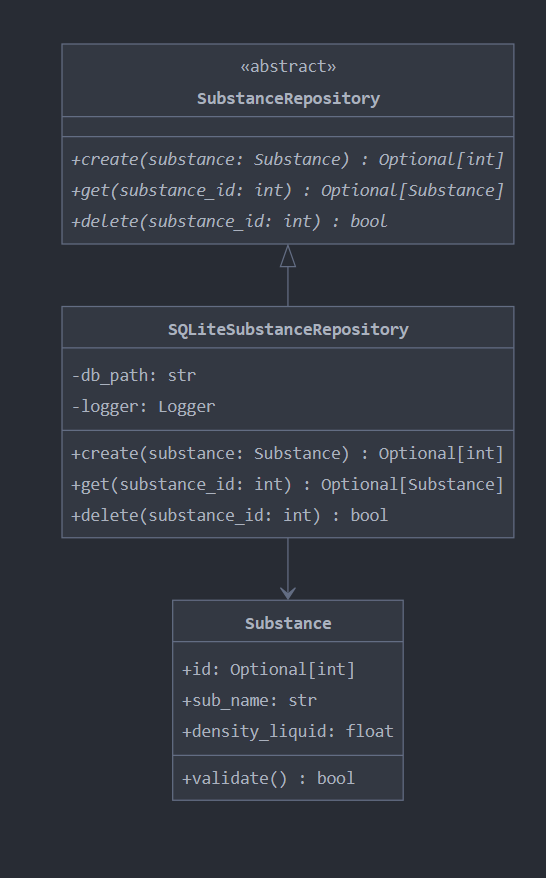


1. infrastructure/database/sqlite/substance\_repository.py - это класс, реализующий работу с БД:



Важный момент: SQLiteSubstanceRepository наследуется от абстрактного класса SubstanceRepository (который находится в domain/repositories/substance\_repository.py).

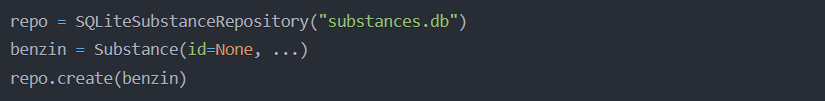
Схематично это выглядит так:



То есть:

1. Substance - это просто структура данных с валидацией
2. SubstanceRepository - это интерфейс, определяющий методы для работы с веществами
3. SQLiteSubstanceRepository - это конкретная реализация для SQLite

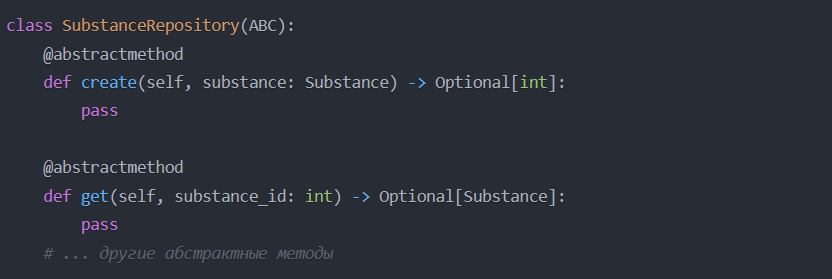
Когда в example\_usage.py мы делаем:



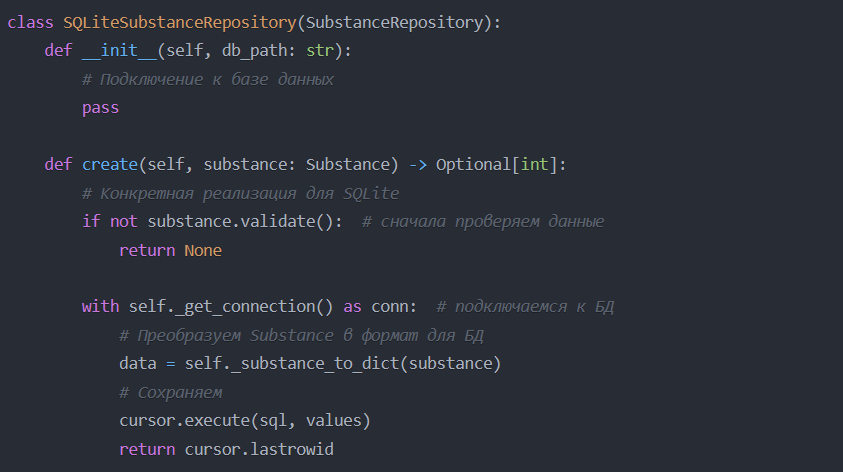
Мы создаём экземпляр SQLiteSubstanceRepository, который знает как работать с SQLite базой данных, создаём объект Substance с данными и используем репозиторий для сохранения этих данных в БД.

Давайте разберем следующий уровень - абстрактный класс SubstanceRepository и его реализацию.

1. domain/repositories/substance\_repository.py - это абстрактный базовый класс:



1. infrastructure/database/sqlite/substance\_repository.py - реализация для SQLite:



Когда происходит сохранение вещества, процесс такой:

1. example\_usage.py создает объект Substance
2. Вызывает метод create у SQLiteSubstanceRepository
3. SQLiteSubstanceRepository:
   * Проверяет валидность данных через Substance.validate()
   * Преобразует объект Substance в формат для БД
   * Подключается к БД
   * Выполняет SQL-запрос
   * Возвращает ID созданной записи

Преимущество такого подхода в том, что если нам понадобится другая база данных (например, PostgreSQL), мы просто создадим новый класс PostgreSQLSubstanceRepository, который будет реализовывать те же методы, но работать уже с PostgreSQL.

