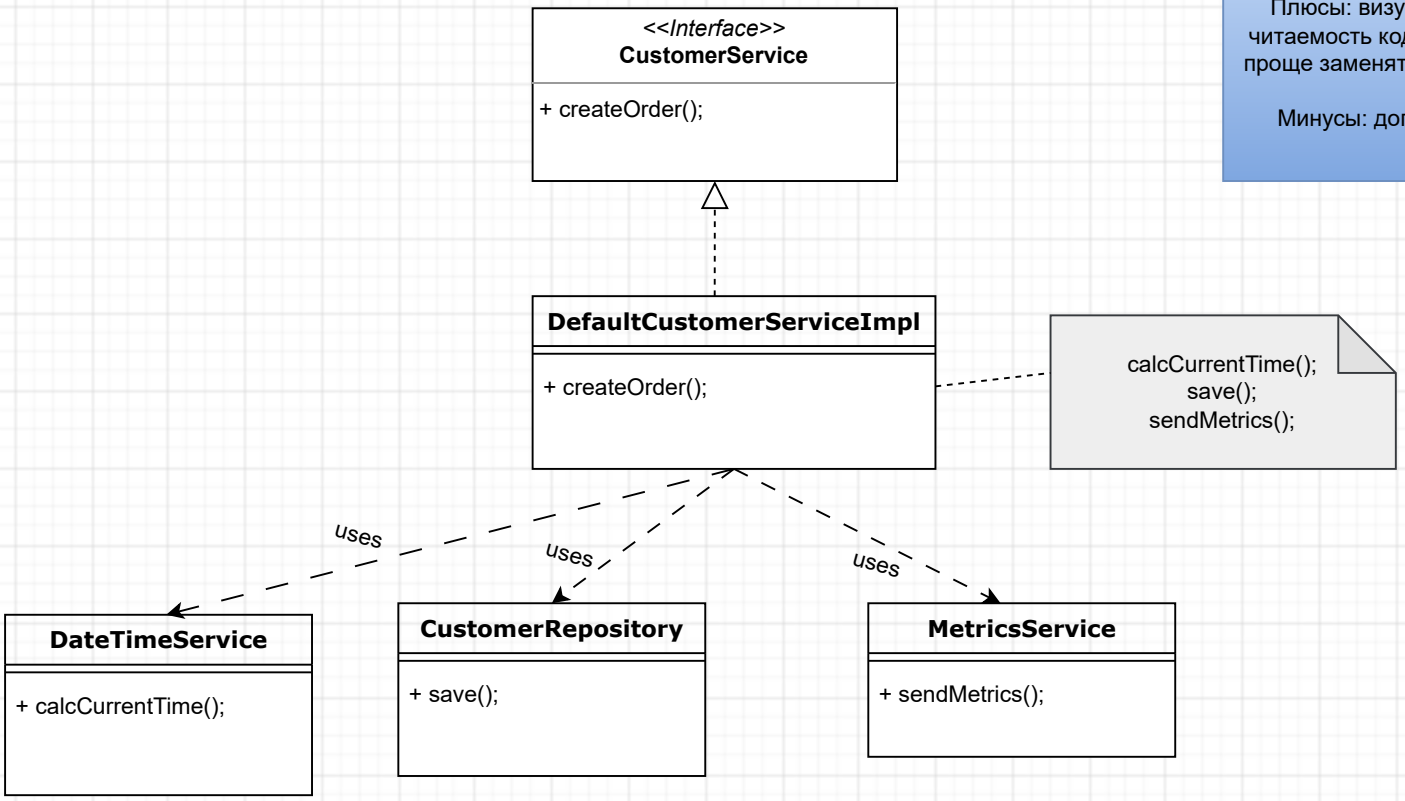
Сервис REST API
Паттерн Фасад, в данном случае скрывает детали имплементации некоторого метода API, для примера создания какого-то заказа.

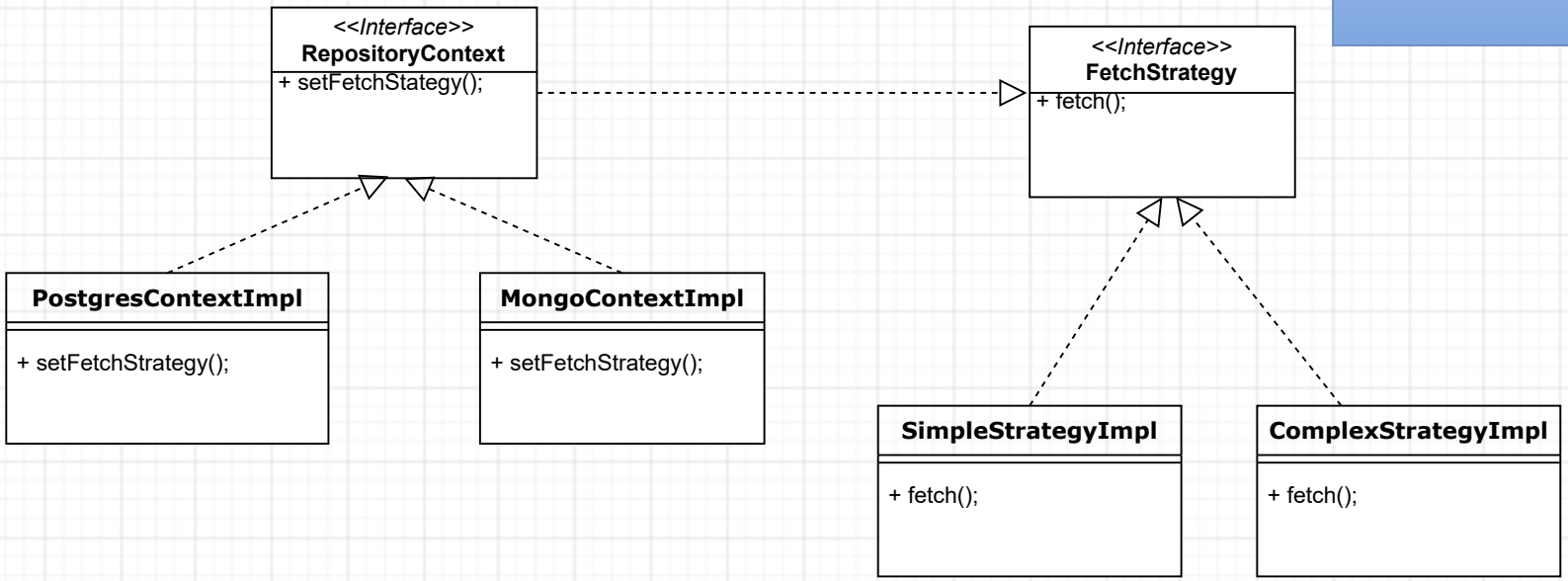
Плюсы: визуально скрывает сложность и повышает читаемость кода, через использование DI/IOC позволит проще заменять вариант реализации при изменившихся требованиях.
Минусы: дополнительные слои абстракции, может затруднять понимание



Сервис REST API

Паттерн Стратегия. Предположим, что у нас есть несколько сценариев работы с репозиториями, в зависимости от того, к какой БД мы хотим подключаться, и какие нам нужны оптимизации. С помощью стратегии мы сможем выбрать нужную БД и нужный вариант оптимизации.

Плюсы: "мобильность" в выборе нужных сценариев работы, ускорение процесса добавления новых вариантов реализации.
Минусы: потребитель данной стратегии должен будет следить за корректностью "конфигурирования" желаемой стратегиии.



Сервис REST API

Паттерн Строитель. Предположим, нам нужно конфигурировать разные конечные варианты экземпляров классов Order, на основании переданных через API входных значений

Плюсы: изолирует код создания заказа от остальной логики, позволяет создавать итеративно создавать продукт и следить за корректностью его создания.
Минусы: паттерн сильно завязан на структуру создаваемого объекта.