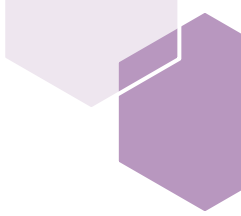




# Подходы к проектированию и реализации





## Цели урока:

- 
- 01 — узнать основные подходы к проектированию, в чем их различие и когда какой из подходов использовать
  - 02 — узнать про типы эмуляторов
  - 03 — научиться использовать полученные знания в работе
- 
- 

# Темы урока

01

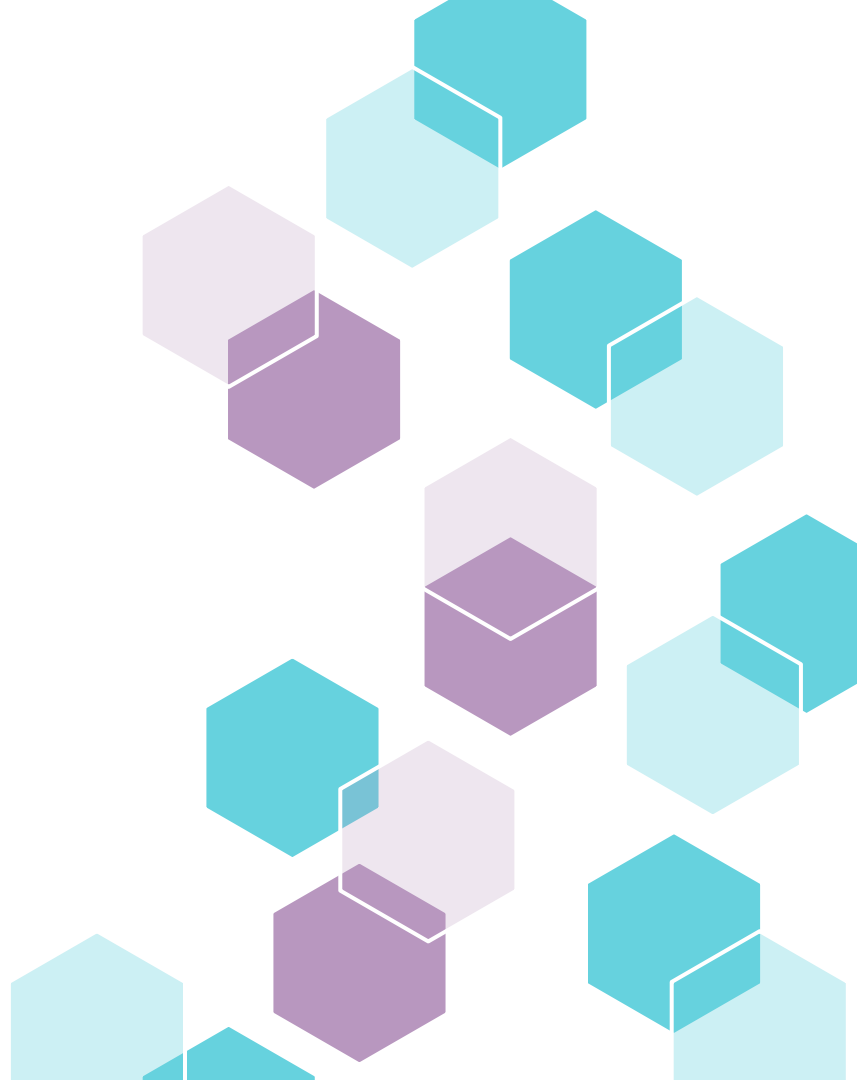
Подходы к  
проектированию  
методов


02

Эмуляция сервиса

—  
**01**

# **Подходы к проектированию методов**





При решении большинства задач, ориентированных на создание новой фичи или предоставление нового интерфейса взаимодействия с системой через API, можно оказаться в различных условиях, требующих разных **подходов к проектированию методов**

# Подходы к проектированию методов

Различают два основных

**Code First**

**API First**

## Code First



Вы создаете продукт, который пройдет путь от проектирования, реализации и документирования до презентации потенциальным потребителям, то вы действуете в условиях неопределенности количества и характеристик конечных потребителя



Ориентация на обобщенный образ "целевого потребителя",



Вы спроектируете "универсальное" решение



Никто ещё не зависит от вашего АПИ

# API First

Акцент смещается с универсальности вашего решения на учёт специфики потребителя

Корневая идея подхода заключается в том, что АПИ контракт не генерируется из кода, а создается аналитиком в полном соответствии с одним из стандартов - **OpenAPI/AsyncAPI/GraphQL/... etc.** - и это является первым шагом к реализации, до завершения которого ни поставщик, ни потребитель АПИ к реализации не приступают

*Стоит подчеркнуть, что реализация не может считаться завершенной, если не проведено её функциональное тестирование*





## Documentation First

Documentation First - это не существующее в реальности понятие  
Это формулировка, используемая в данном курсе для обозначения важной для понимания составляющей, необходимой для обеспечения возможности разработки middle слоя - ПО, непосредственно реализующего логику обработки запроса и отправки ответа потребителю согласно контракту

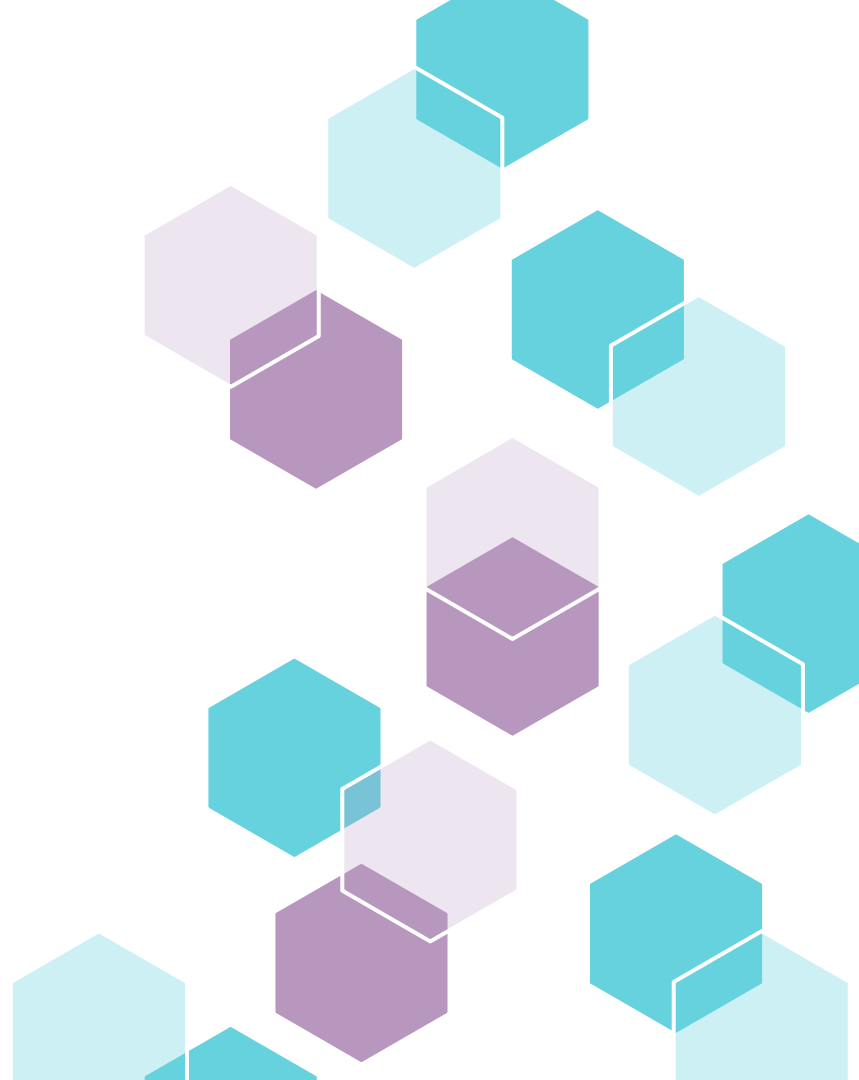
В зависимости от используемых фреймворков и в целом инфраструктуры, этот этап может отсутствовать


**Например**, в случае использования ORM - Object Relationship Model - то есть механизма, генерирующего SQL для работы с БД исходя из кода, написанного в middle слое

---

02

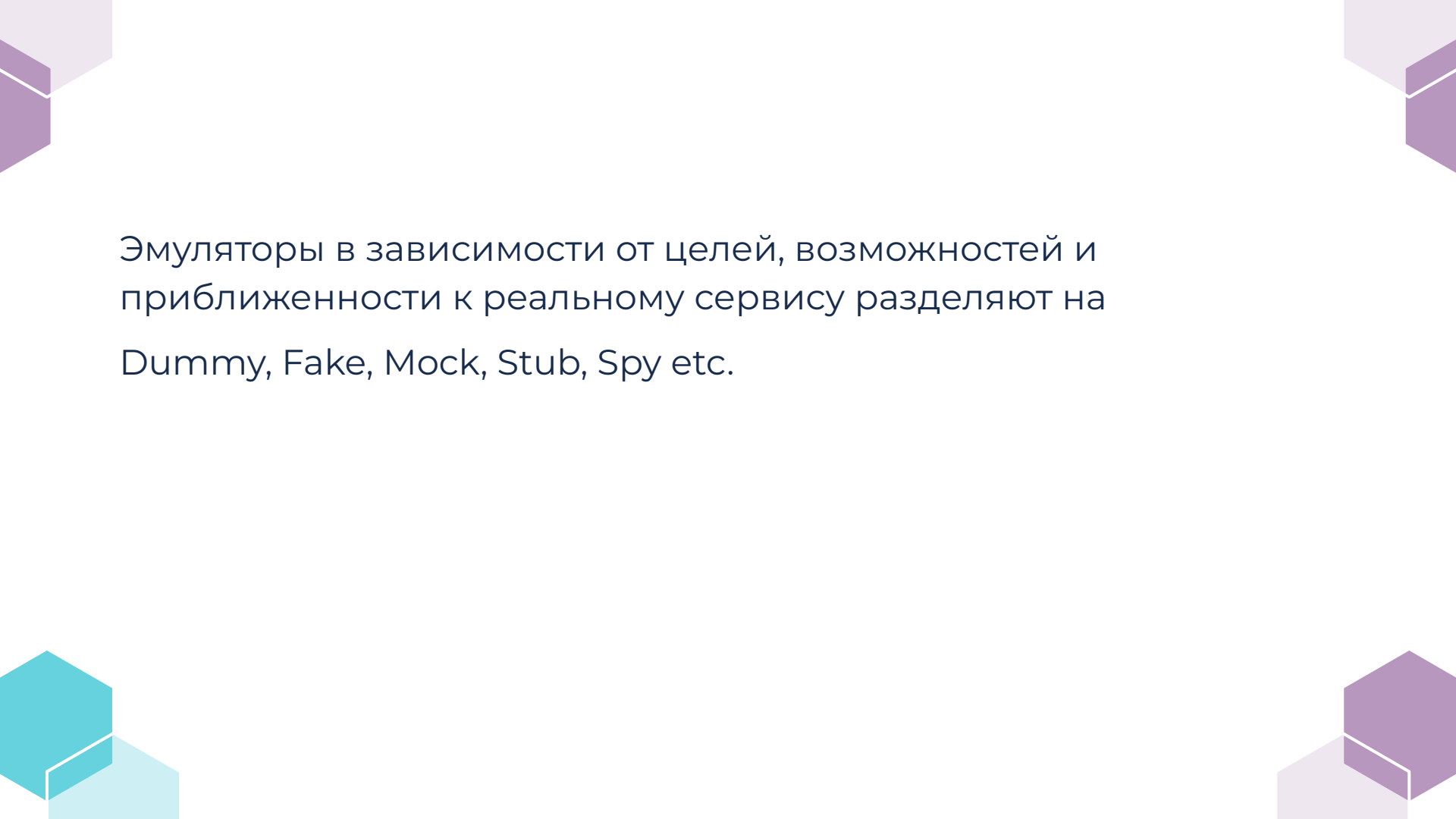
# Эмуляция сервиса





Задачу, которая решается эмуляцией сервиса, можно рассматривать как "получить соответствующий контракту ответ на запрос на некотором эндпоинте(url)"

Однажды этот эндпоинт будет предоставлен поставщиком, но пока этого не случилось - его необходимо эмулировать



Эмуляторы в зависимости от целей, возможностей и приближенности к реальному сервису разделяют на Dummy, Fake, Mock, Stub, Spy etc.

На практике же, аналитиками и тестировщиками чаще всего все это именуется одним общим термином - моки (**Mocks**)

Конкретизация типа эмуляции не существенна, **90% кейсов** эмуляции - это получить конкретный ответ из спеки, для проверки конкретного кейса

Следующий кейс = другой конкретный ответ



Оставшиеся **10%** - это проверки цепочки вызовов, кейсов с обработкой ошибок, отладки плавающих багов и т.д.

В эти 10% входит слишком много вариаций, чтобы охватить их в полноценный материал, поэтому мы сфокусируемся на 90% типовых случаев



## Типы эмуляторов

Информация размещена по ссылке, которая доступна в вашем личном кабинете на платформе

<https://medium.com/@matiasglessi/mock-stub-spy-and-other-test-doubles-a1869265ac47>

---

## В следующем уроке

Вы завершили урок по теме "Подходы к проектированию и реализации"

В следующем уроке вы изучите тему "SQL заглушки базовых методов"





**Выполните задание, которое размещено на платформе в вашем личном кабинете**

**Домашнее задание**