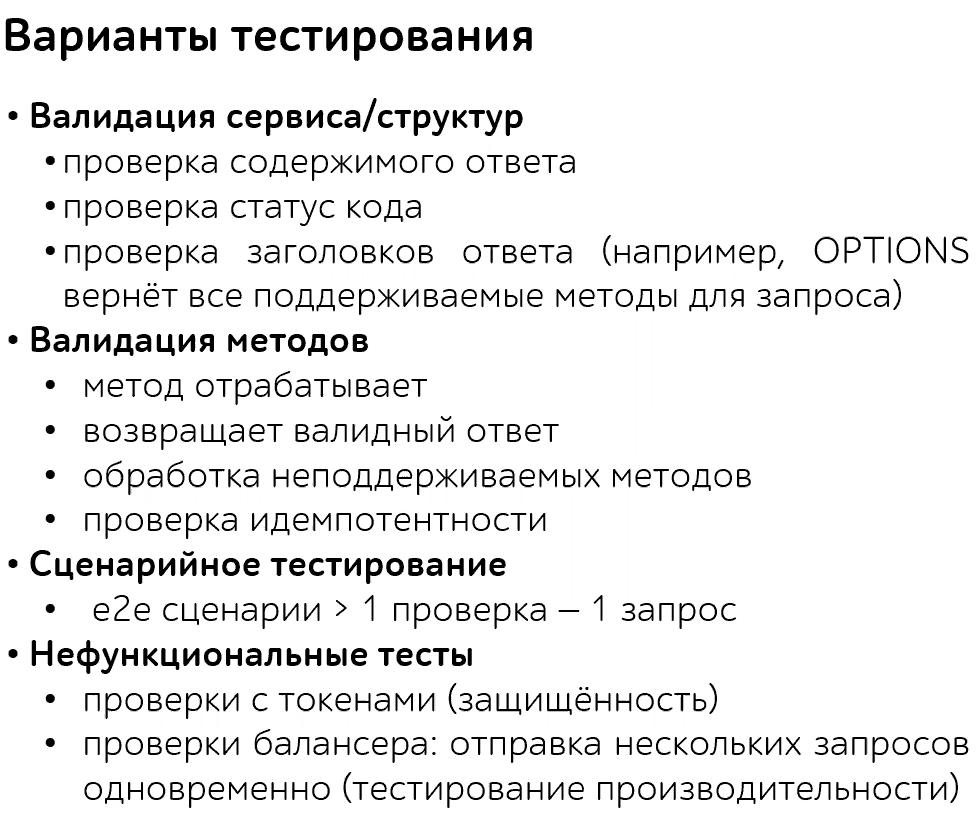


**Как тестировать**



Изменять параметры API запроса (Endpoint, Query parameters) – граничные значения, эквивалентные классы

Изменять заголовки

Изменять тело запроса (ровно так же как бы мы тестировали в ручную. Каждый параметр в теле воспринимать как отдельный элемент)

Тестировать бизнес логику

Смотреть код ответа сервера:

* Если пришла 400 ошибка, то проверять правильно ли мы передаем данные. Нет ли у нас расширений которые блокируют передачу запроса и тд. В общем проверять свою часть.

Смотреть тело ответа сервера

Смотреть заголовки ответа сервера

Если в теле ответа есть ссылки на другие ресурсы – это **Hypermedia API**. Перейти на эти ссылки и посмотреть ссылаются ли они на что то. Ссылаются ли они на изначальный ендпоинт

Ищи бизнес проблемы. Узнай какую проблему бизнеса решает этот API. Публичный API и тот который будет использован внутренней командой разные вещи. Если найдена проблема которая блокирует API делать то ради чего он был создан – это важный баг. Узнаем кто будет пользоваться прилагой, строим персональность этого человека, и на основе нее тестируем.:

1. Identify at least two key stakeholders for the API. Do this by asking the question "who wants (or wanted) this API built?" Write down these stakeholder names.
2. If possible, talk to those stakeholders and ask them what they think the purpose of this API should be and why they want to build it. Write down their answers.
3. Create preferably two (but at least one) personas that list out the kinds of people that you think will be using the API. What skill level do they have? What work are they trying to accomplish? How will your API help them?
4. Write down what problem(s) you think the API will solve.
5. Now, take all the information that you have gathered and look through it. Distill it down into two or three sentences that explain the purpose of the API.

**API Tests**

**Contract Testing** - use to verify that your API is meeting a given contract, which is usually specified in some sort of specification language such as OpenAPI.

**Integration** or **Workflow** - where you are testing that you can accomplish a certain workflow. This kind of testing would typically involve calling multiple endpoints. For example, you might POST something to create a new item and then call another endpoint to verify that that item has been correctly created, and is also accessible in other parts of the system. Another example would be to GET an object such as a shopping cart and then use information in that response to GET some related information, such as details about the products in the cart.

**Endpoint tests** - call an endpoint with various kinds of inputs and verify the results. One important thing to consider here isn't so much a type of API testing as it is a style of test that should be run. You should check some negative test scenarios. What happens if you put in bad values or incorrect data? Like any good testing, API tests should not just consider what happens

**Security testing** - it is an important style of testing to consider in APIs. APIs expose a lot of power and flexibility to end users (and that is partly why many companies make them), but in doing so, it is easy to miss things that could allow user access to things they should not have. Many security breaches in recent history have come from problems in APIs. We test different auth flows (Bearer Token, API Keys, …) and what user with different privileges can and can’t do.

**Performance testing** - Postman isn't really designed as a performance testing tool, but you can get some basic performance data out of it.

**Checking UI** – after making changes to an Endpoint via Postman, we should check UI that these changes indeed happened.