О валидации данных. Может можно обойтись без нее?

FastAPI тесно связан с библиотекой Pydantic, занимающейся в основном валидацией данных. Для написания качественного API мы должны настроить валидацию данных на входе, то есть при получении запроса от пользователя, и на выходе, то есть при отправке ответа пользователю.

> Зачем нужна валидация данных? О каких данных идет речь?

Если мы ожидаем от клиента (здесь и далее — программы, использующей наш API) конкретные параметры с определенными типами, будет правильно отдать клиенту ошибку, если он забудет передать один из обязательных параметров или передаст параметр не того типа. Причем очень важно говорить клиенту, где именно он ошибся, чтобы тот смог быстро исправить свою ошибку и передать верные параметры внутри запроса. Pydantic отлично справляется со всеми этими задачами.

Если мы обещаем клиенту, что тот будет получать список словариков с определенными ключами, например,

```
[
  {
    "id": 1,
    "name": "Артём"
  },
  {
    "id": 2,
    "name": "Sebastian"
  }
]
```

согласитесь, будет не очень хорошо, если мы вместо списка словариков отправим словарик, или список словариков внутри списка, или любую другую структуру данных, которую клиент не ожидает получить. Для целей валидации выходных данных FastAPI также использует мощь Pydantic, чтобы убедиться, что API отдает данные нужной структуры с нужными типами данных.

Работа с входными данными

Входные данные проверяются в два этапа: сперва проверяется, указаны ли все обязательные параметры запроса, а затем проверяется их тип. Эндпоинт с такой проверкой выглядит следующим образом:

```python
@router.get("/{location}")
def get_hotels(
    location: str,
    date_from: date,
    date_to: date,
):
    ....
```

Данная функция создает эндпоинт вида `localhost:8000/{location}`, где вместо `{location}` может быть, например, Алтай, тогда получится эндпоинт вида `localhost:8000/Алтай`. Причем при обращении к эндпоинту пользователю API нужно будет указать дополнительно параметры `date_from` и `date_to`.

FastAPI при получении запроса от пользователя с помощью библиотеки Pydantic сперва проверяет, что параметр `location` указан в параметрах пути, а `date_from` и `date_to` — в параметрах запроса. Далее Pydantic пытается привести все полученные параметры к указанным нами типам. Если ему не удается это сделать, вы увидите ошибку с кодом 422 и подробным ответом, указывающим какой из параметров не удалось привести к нужному типу.

Работа с выходными данными

Пример ниже показывает 2 возможных способа указания Pydantic схемы, по которой будет происходить валидация и сериализация данных (сериализация — это приведение объекта из одного формата в другой, например Python объекта в JSON, десериализация — обратный этому процесс). В примере Pydantic попробует привести переменную `hotels` к схеме `SchemaHotel` и если все удастся, то клиент получит свои данные. В противном случае клиент увидит ошибку сервера.

```python
@router.get("/{location}", response_model=list[SchemaHotel])  # 1 способ
def get_hotels(
    location: str,
    date_from: date,
    date_to: date,
) -> list[SchemaHotel]:  # 2 способ
    ...
    hotels = [<HotelsModel 1>, <HotelsModel 2>]
    ...
    return hotels
```

Первый способ более явный, в то время как второй способ более лаконичный. Выбирайте понравившийся вам, но, пожалуйста, не используйте их одновременно :)

Ошибка валидации данных в FastAPI

Важно запомнить вид ошибки валидации данных и ее код `422`, так как они очень часто будут встречаться на вашем пути. Ниже пример типичного ответа от Pydantic при ошибке валидации данных:

```
{
    "detail": [
        {
            "loc": [  # <-- loc = location, то есть указание, где произошла
ошибка
                "path",  # <-- ошибка в параметре пути
                "user_id"  # <-- название параметра, на котором произошла
ошибка
            ],
            "msg": "value is not a valid integer",  # <-- приятное глазу
сообщение, что user_id - не целое число
            "type": "type_error.integer"
        }
    ]
}
```

Пример ошибки валидации данных, которую вы будете видеть в терминале