elastic

Web API and client generation using OpenAPI specification

Enrico Zimuel, *Principal Software Engineer*

May 20, 2022

phpday

Enrico Zimuel

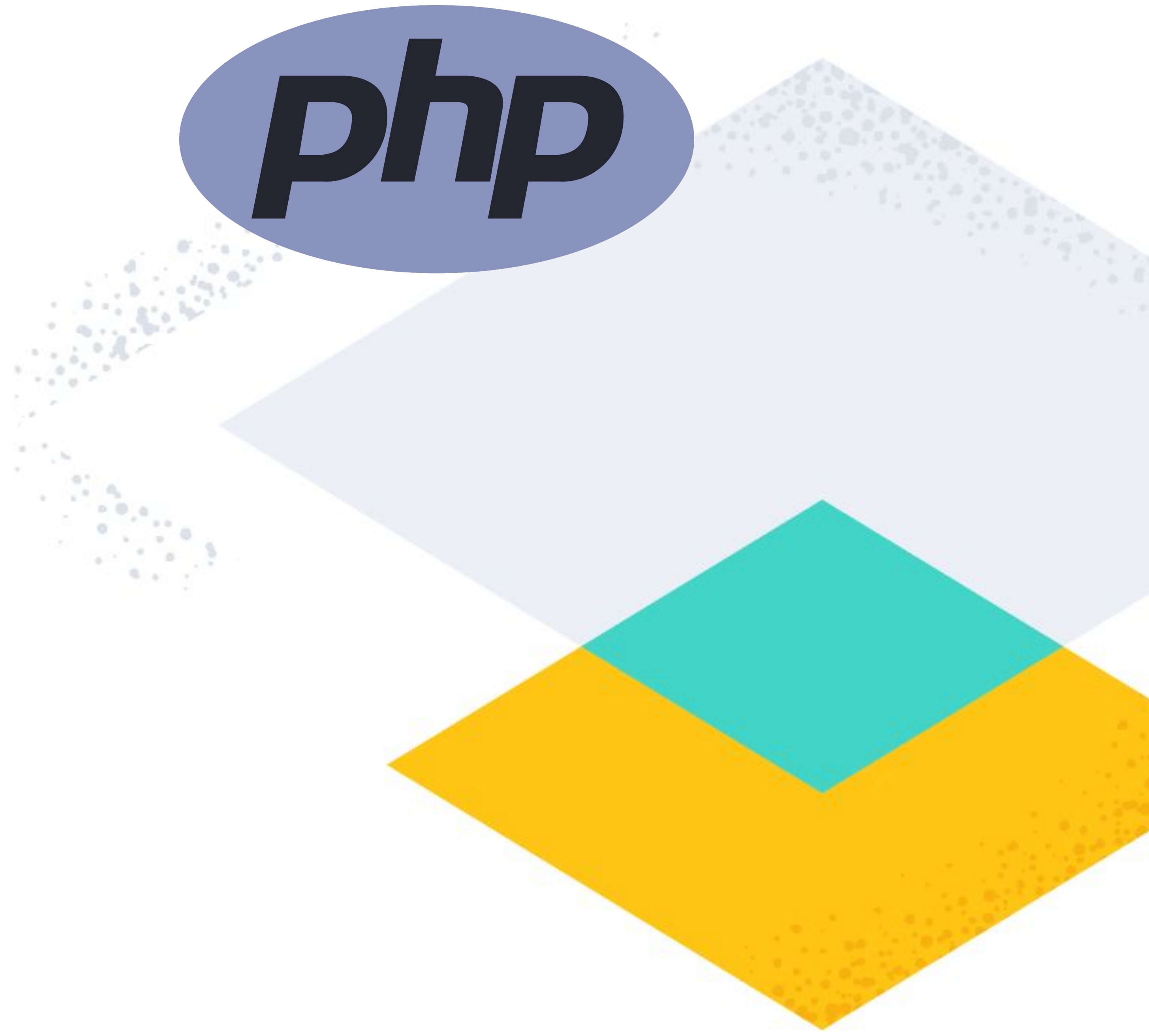
Principal Software Engineer

- Developer since 1996
- Principal Software Engineer @ Elastic
- ex Zend Framework core developer @ Zend
- ex Research Programmer @ UvA
- PHP FIG core committee member
- TEDx and international speaker +110 conferences
- Author of the book Sviluppare in PHP 7 by Tecniche Nuove, 2019
- Adjunct Professor at University of Turin and ITS ICT Piemonte
- Co-founder PUG Torino

elastic

Summary

- OpenAPI specification
- PHP Tools
- Web API client
- Generating a web client from OpenAPI
  - elasticsearch-php
  - enterprise-search-php



OpenAPI

- The **OpenAPI Specification** (OAS) defines a standard, programming language-agnostic interface description for HTTP APIs, which allows both humans and computers to discover and understand the capabilities of a service without requiring access to source code
- Focus on **RESTful API**
- Last version **3.1.0** (released 2021-02-15)

OpenAPI (2)

- Previously known as the **Swagger Specification**, it became a separate project in 2016, overseen by the **OpenAPI Initiative** an open-source collaboration project of the **Linux Foundation**
- Use JSON (or YAML 1.2)
- Official site: openapis.org

A standard for describing web API

- Applications implemented based on OpenAPI interface files can automatically generate documentation of methods, parameters and models
- This helps keep the documentation, client libraries, and source code in sync
- The goal is to have an automated system for:
  - **Annotate** the server side code
  - Generate the **API documentation**
  - Generate a **client library**

OAS: root document object

- **openapi***: string
- **info***: Info obj
- **servers**: [Server obj]
- **paths***: Paths obj
- **components**: Components obj
- **security**: [Security required obj]
- **tags**: [Tag obj]
- **externalDocs**: External doc obj

* = required

Example

```yaml
openapi: 3.0.3
info:
 title: Sample API
 description: Here the description
 version: 0.1.0
servers:
 - url: http://api.example.com/v1
   description: Optional server description, e.g. Main (production) server
 - url: http://staging-api.example.com
   description: Optional server description, e.g. Internal staging server for testing
paths:
 /users:
   get:
     summary: Returns a list of users.
     description: Optional extended description in CommonMark or HTML.
     responses:
       '200':    # status code
         description: A JSON array of user names
         content:
           application/json:
             schema:
               type: array
               items:
                 type: string
```

Request and response body

```yaml
responses:
 '200':
   description: A User object
   content:
     application/json:
       schema:
         type: object
         properties:
           id:
             type: integer
             description: The user ID.
           username:
             type: string
             description: The user name.
```

elastic

Reference: $ref

```yaml
responses:
 '200':
   description: A User object
   content:
     application/json:
       schema:
         $ref: '#/components/schemas/User'
```

```yaml
components:
 schemas:
   User:
     type: object
     properties:
       id:
         type: integer
         description: The user ID.
       username:
         type: string
         description: The user name.
```

PHP Tools

zircote/swagger-php

- zircote/swagger-php
- Generate interactive OpenAPI documentation for your RESTful API using doctrine annotations

```
/**
 * @OA\Info(title="My First API", version="0.1")
 */


/**
 * @OA\Get(
 *     path="/api/users",
 *     @OA\Response(response="200", description="An example resource")
 *     @OA\Response(response="401", description="Not allowed")
 * )
 */
```

Attributes

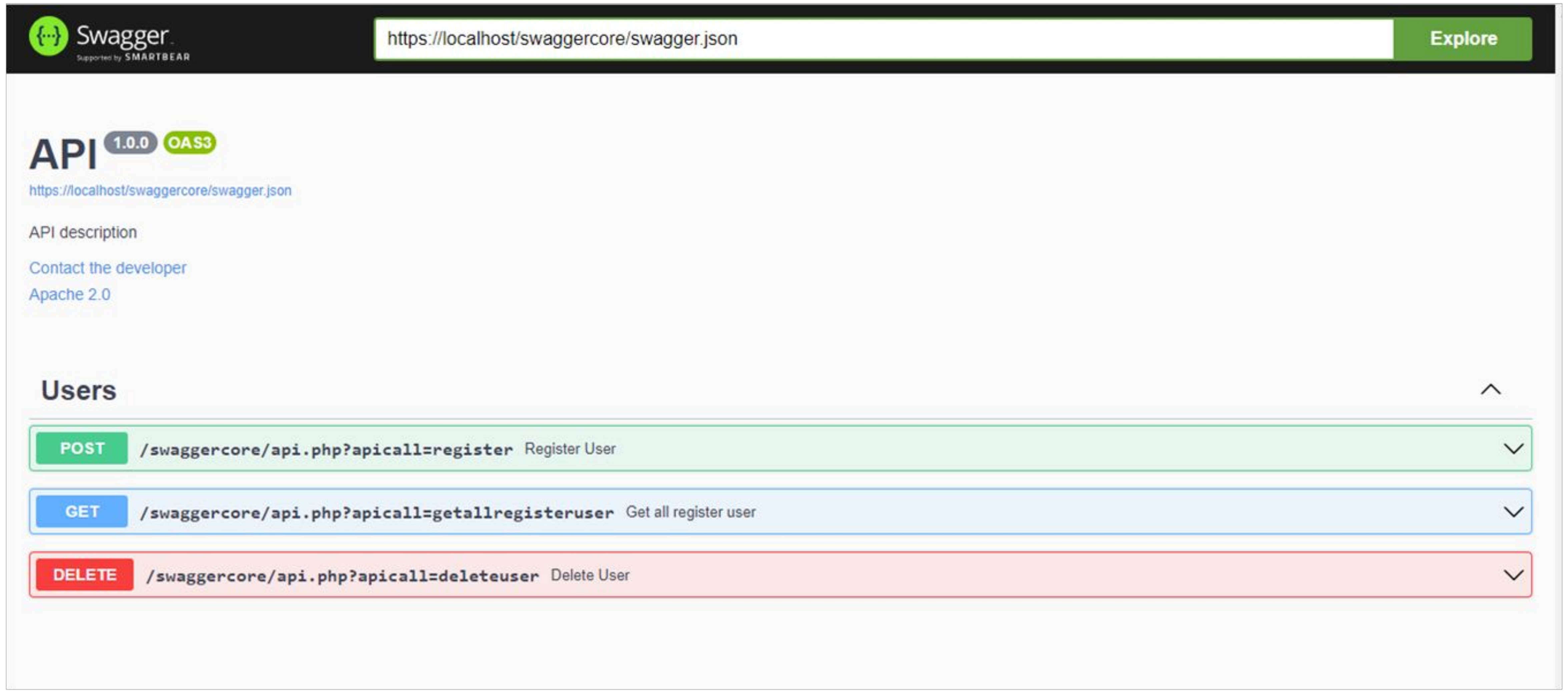
- Swagger-php uses also PHP 8 attributes

```
#[OA\Get(
   path: '/api/users',
   responses: [
       new OA\Response(response: 200, description: 'An example resource'),
       new OA\Response(response: 401, description: 'Not allowed'),
   ]
)]
public function users() { /* ... */ }
```

Generate API documentation

- Generate always-up-to-date documentation:

```
$openapi = \OpenApi\Generator::scan(['/path/to/project']);
header('Content-Type: application/x-yaml');
echo $openapi->toYaml();
```



API 1.0.0 OAS3

https://localhost/swaggercore/swagger.json

API description

Contact the developer

Apache 2.0

Users

POST /swaggercore/api.php?apicall=register Register User

GET /swaggercore/api.php?apicall=getallregisteruser Get all register user

DELETE /swaggercore/api.php?apicall=deleteuser Delete User

openapi-psr7-validator

- thephpleague/openapi-psr7-validator
- This package can validate PSR-7 messages against OpenAPI (3.0.x) specifications expressed in YAML or JSON

```
use League\OpenAPIValidation\PSR7\OperationAddress;
use League\OpenAPIValidation\PSR7\ValidatorBuilder;

$jsonFile = 'openapi.json';
$validator = (new ValidatorBuilder)->fromJsonFile($jsonFile)->getRequestValidator();
// validate a PSR-7 $request
$match = $validator->validate($request);

$validator = (new ValidatorBuilder)->fromJsonFile($jsonFile)->getResponseValidator();
$operation = new OperationAddress('/some/operation', 'post');
// validate a PSR-7 $response
$validator->validate($operation, $response);
```

elastic

Swagger Editor Validate Github Action

- char0n/swagger-editor-validate
- GitHub Action to validate OpenAPI (OAS) definition file using Swagger Editor

```yaml
on: [push]

jobs:
 test_swagger_editor_validator_remote:
   runs-on: ubuntu-latest
   name: Swagger Editor Validator Remote

   steps:
     - uses: actions/checkout@v2
     - name: Validate OpenAPI definition
       uses: char0n/swagger-editor-validate@v1
       with:
         definition-file: examples/openapi-2-0.yaml
```

Nette PHP Generator

- nette/php-generator
- Generator of PHP code
- Object oriented architecture
- Supports the latest PHP 8.1 features: enum, attributes, etc
- PSR-12 compliant

elastic

Nette PHP Generator: example

```php
$class = new Nette\PhpGenerator\ClassType('Demo');

$class->setFinal()
    ->setExtends(ParentClass::class)
    ->addImplement(Countable::class)
    ->addComment("Description of class.\nSecond line\n")
    ->addComment('@property-read Nette\Forms\Form $form');

// to generate PHP code simply cast to string or use echo:
echo $class;
```

```php
/**
* Description of class.
* Second line
*
* @property-read Nette\Forms\Form $form
*/
final class Demo extends ParentClass implements Countable
{
}
```

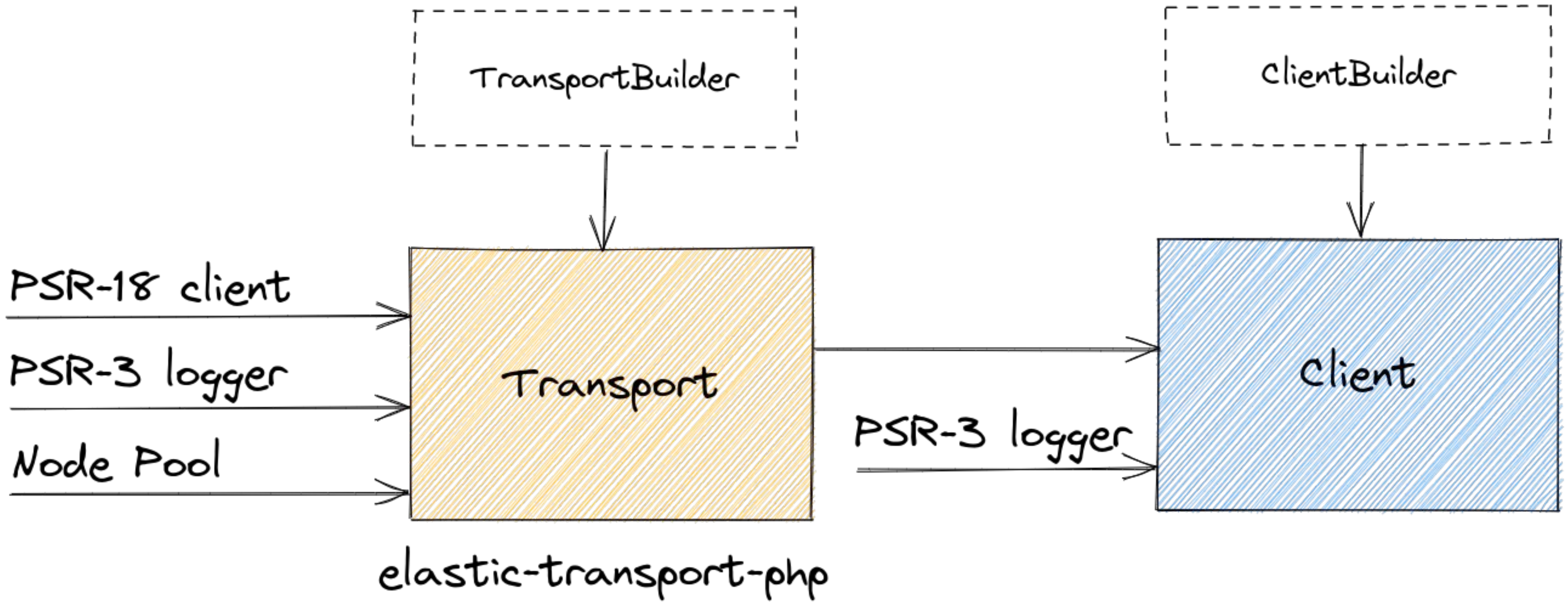
Web API client

Goals

- Generate code from OpenAPI specification
- Use PSR standards:
  - PSR-7 for HTTP messages
  - PSR-18 for HTTP client
  - PSR-3 for logging
- Consume the API endpoint as functions of a class
- Simplify the serialization and deserialization of the body request and response

elastic-transport-php

- elastic/elastic-transport-php
- A PSR-18 wrapper client for Elastic products



enterprise-search-php

- elastic/enterprise-search-php
- PHP client for Enterprise Search, App Search and Workplace Search
- Supports PSR standards
- Request types (object with schema from OpenAPI spec)
- Response as object, array, string

Example: enterprise-search-php

```php
use Elastic\EnterpriseSearch\Client;
use Elastic\EnterpriseSearch\AppSearch\Request;

$client = new Client([
   'host' => 'http://localhost:3002',
   'app-search' => [ 'token' => '<insert here the API-KEY>' ]
]);
$appSearch = $client->appSearch();
// Create a 'test' engine
$result = $appSearch->createEngine(
   new Request\CreateEngine(
       new Schema\Engine('test')
   )
);
var_dump($result->name); // test
var_dump($result->document_count); // 0, since just created
```

Create Engine API

```
POST <URL>/api/as/v1/engines
```

If you don’t have OpenAPI spec

- No OpenAPI spec in place
- The server API is a black box (eg. no PHP code)
- You can record the HTTP request and response (e.g. use the test environment of the API endpoints to store the HTTP traffic)
- You can write an OpenAPI custom generator or use an existing tool (e.g. Swagger Inspector, Akita)

elastic

Elasticsearch

- elastic/elasticsearch
- **Elasticsearch** is a distributed, multitenant-capable full-text search engine with an HTTP web interface and schema-free JSON documents
- Written in Java, very big project (64’263 commit)
- Dual license: Server Side Public License, Elastic
- ≈ 400 API endpoints
- API test suite already in place (custom YAML format)

elastic

YAML tests of Elasticsearch

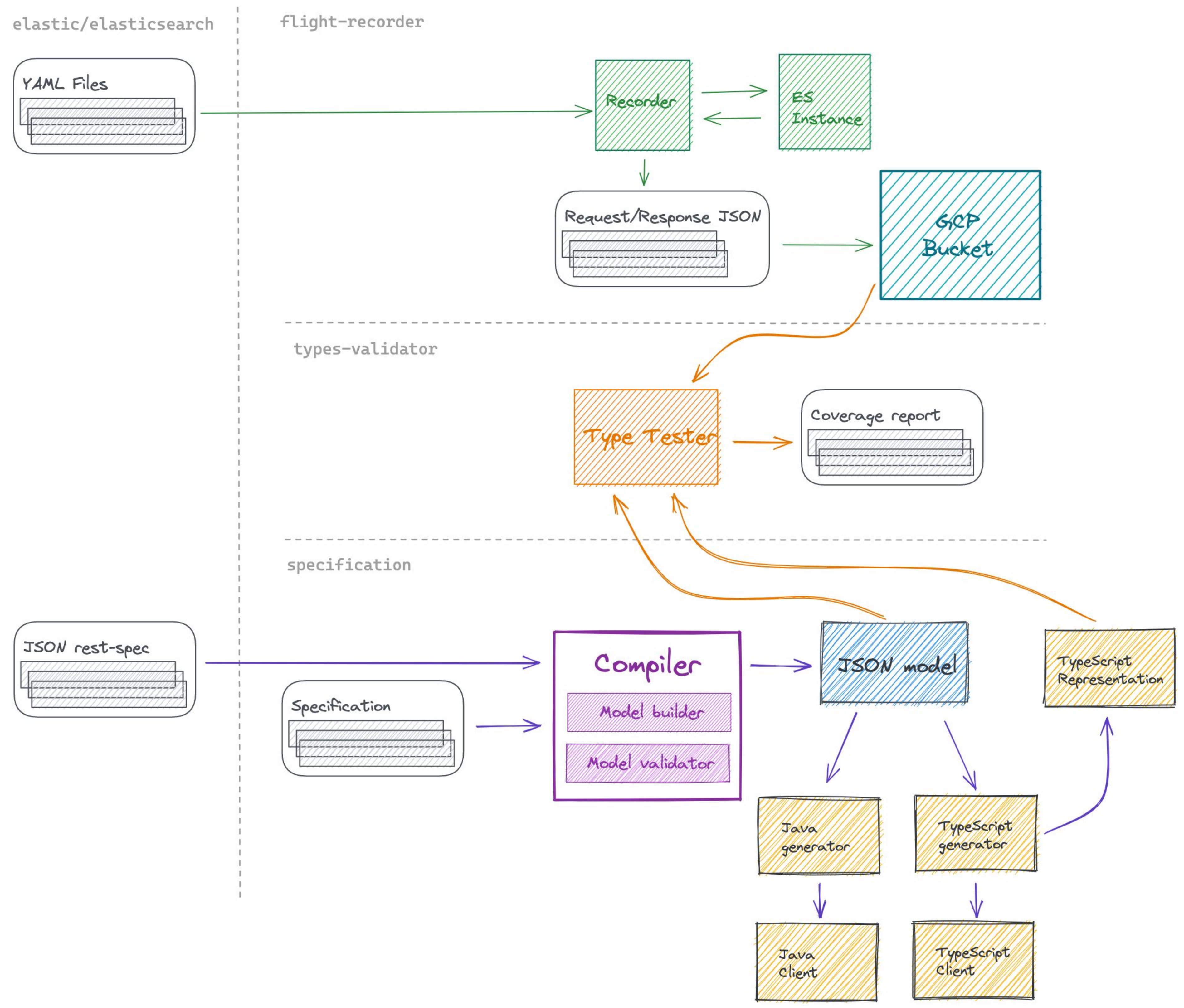
/rest-api-spec/src/yamlRestTest/resources/rest-api-spec/test

```
---
"Help":
 - skip:
     version: " - 7.3.99"
     reason:  "is_write_index is shown in cat.aliases starting version 7.4.0"
 - do:
     cat.aliases:
       help: true
 - match:
     $body: |
       /^  alias            .+   \n
           is_write_index   .+   \n
       $/
```

Elasticsearch clients

- Library clients in different languages
- Need: automate the code generation using API spec
- Elastic supports libraries for:
    - Go
    - Java
    - Javascript
    - .NET
    - Perl
    - PHP
    - Python/Eland
    - Ruby/Ruby on Rails
    - Rust

Elasticsearch specification



PHP client for Elasticsearch 8

- elastic/elasticsearch-php
- ≈ 70 million installations
- A brand new library based on PSR standards
- Ensure backward compatibility with v7
- Add new features: request/response types (coming!)

elastic

Elasticsearch-php: example

```php
$client = Elastic\Elasticsearch\ClientBuilder::create()
   ->setHosts(['localhost:9200'])
   ->build();
// Info API
$response = $client->info();

echo $response['version']['number']; // 8.2.0
echo $response->version->number; // 8.2.0
// $response is PSR-7 response
echo $response->getStatusCode(); // 200
echo (string) $response->getBody(); // Response body in JSON

var_dump($response->asArray());  // response body content as array
var_dump($response->asObject()); // response body content as object
var_dump($response->asString()); // response body as string (JSON)
var_dump($response->asBool());   // true if HTTP response code between 200 and 300
```

@generated

- **@generated** is a new PSR-19 tag
- Used to denote a class or a function that has been generated using an automation script
- This tag should be used to warn to not change the code, since the change will be overwritten by the automation script

```
/**
* @generated This file is generated, please do not edit
*/
```

Metadata

- Using a code generator you can add interesting metadata (phpdoc, attributes) to PHP code

```
/**
    * Shows information about …
    *
    * @see https://www.elastic.co/guide/en/elasticsearch/reference/master/cat-alias.html
    * @param array{
    *     name: list, //  A comma-separated list of alias names to return
    *     format: string, // a short version of the Accept header, e.g. json, yaml
    * } $params
    *
    * @throws MissingParameterException if a required parameter is missing
    *
    * @return Elasticsearch|Promise
    */
   public function aliases(array $params = [])
```

References

- Satish A. How we can create documentation for API while development in simple way Swagger (PHP), Medium, 2021
- Dev Drawer, Create REST API Documentation with Swagger UI using PHP, Youtube video
- Phil Sturgeon, Creating OpenAPI from HTTP Traffic, APIs you won't hate
- Swagger OpenAPI tools
- OpenAPI specification
- Elasticsearch PHP client documentation
- E. Zimuel, Introducing the new PHP client for Elasticsearch 8, Elastic, blog, 2022

Thanks!

For more information:

https://www.elastic.co/