9:41
io.italia.it
Messaggi
Tutti
Scadenze
Fascicolo Sanitario Elettronico
11.25
E' disponibile il referto dell'esame effetuato il...
Leggi il referto
Anagrafe Nazionale della Popolazione ...
ieri
La sua Carta d'Identità Elettronica verrà consegnata Mercoledì 31 Marzo fra le 10.00 e...
Aggiungi un promemoria
Comune di Milano - Tributi
12/03
Avviso di pagamento della Tassa Rifiuti 2020
MAR 30
Paga € 199,00
messaggi
portafoglio
documenti
preferenze

Design Secure APIs

Technical specifications and tools from the API Italian Interoperability Framework

EuroPython 2021

Roberto Polli
API Ecosystem

DIPARTIMENTO
PER LA TRASFORMAZIONE
DIGITALE

EUROPYTHON
2021 Jul 26 - Aug 1 Online

Agenda

Enforce secure design practices and coherent interfaces for services provided by 20k+ agencies

- Digital Interfaces Challenges
- API Guidelines
- A roundup of useful standards on HTTP headers, content and authentication in REST APIs
- Open source Online Validator

THE CHALLENGE

Standardizing all public sector APIs

Guidelines can uniform APIs produced by thousands of service providers

60M People

+12k Public Agencies

+8k Cities

20 Regions

(∞ cultural heritage)

DIPARTIMENTO
PER LA TRASFORMAZIONE
DIGITALE

Secure and usable by design with API Guidelines

To achieve **reliable, secure** and **consistently designed services** Italy wrote API Guidelines and support tools

Risks

- **over-complexity:** bureaucratic, non-digital processes are mapped to convoluted APIs without a proper redesign
- **time-constrained engineering:** a restricted group of people addressing the above use-cases within a short deadline

Mitigations

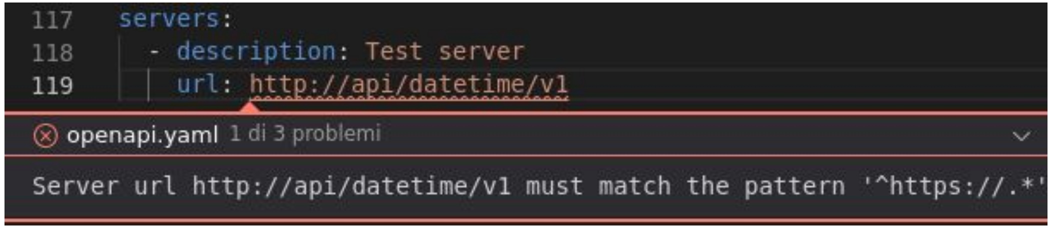
- **Interface Description Languages:** a *formal description of API* interactions, eg: OpenAPI (HTTP) and WSDL (SOAP).

- **API Guidelines:** to uniform the design and security of REST and SOAP services between 12k agencies, together with tools to help agencies and their suppliers in checking their design. *Engage with IETF communities!*

Security basics

Using OpenAPI3 simplifies a broad set of design checks, including some of the OWASP API Security top 10

- **HTTPS** - checks that all URLs in the spec use the https scheme

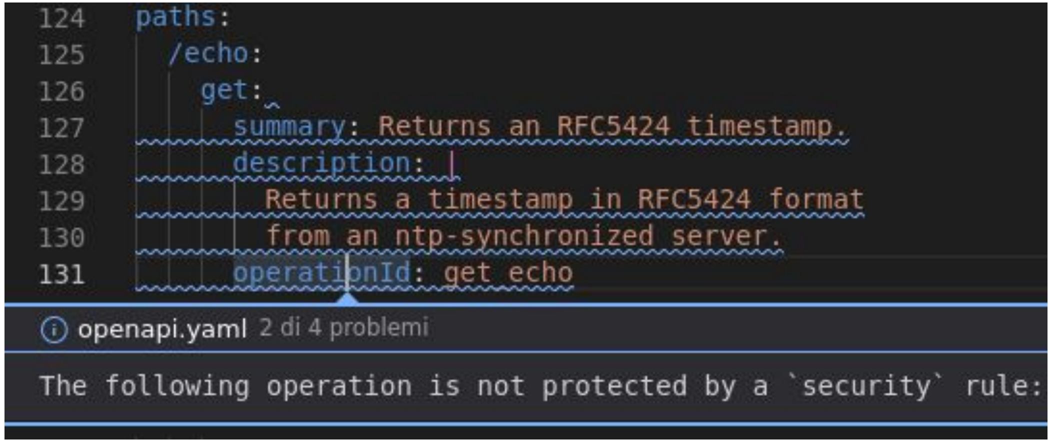


```
servers:
  - description: Test server
    url: http://api/datetime/v1
```

openapi.yaml 1 di 3 problemi

```
Server url http://api/datetime/v1 must match the pattern '^https://.*'
```

- **Authentication and authorization** - checks that every endpoint is properly protected



```
paths:
  /echo:
    get:
      summary: Returns an RFC5424 timestamp.
      description: |
        Returns a timestamp in RFC5424 format
        from an ntp-synchronized server.
      operationId: get_echo
```

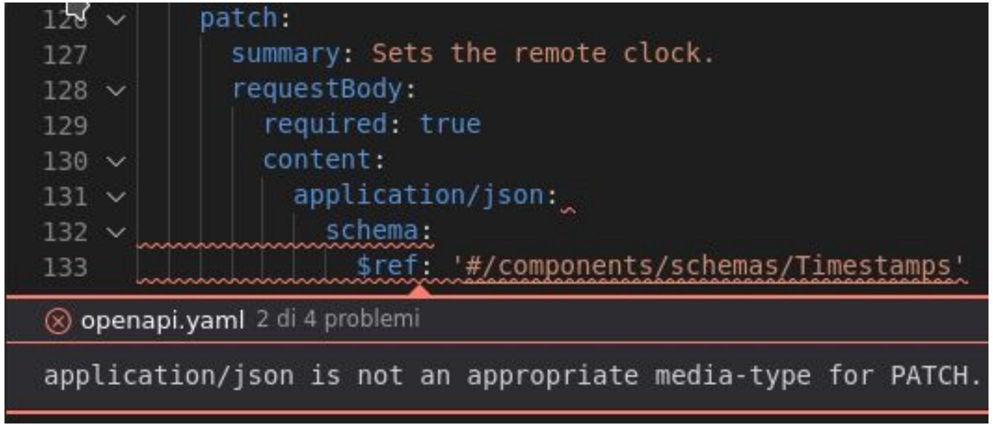
openapi.yaml 2 di 4 problemi

```
The following operation is not protected by a `security` rule:
```

Security basics

Using OpenAPI3 simplifies a broad set of design checks, including some of the OWASP API Security top 10

- **Use HTTP methods correctly** - for example checking that PATCH requests have a suitable media-type, eg. application/merge-patch RFC7386



```
    patch:
      summary: Sets the remote clock.
      requestBody:
        required: true
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/Timestamps'
```

openapi.yaml 2 di 4 problemi

```
application/json is not an appropriate media-type for PATCH.
```

- **RateLimit (OWASP API4:2019)** - define and enforce a coherent ratelimit framework such as draft-ietf-httpapi-ratelimit-headers

DIPARTIMENTO
PER LA TRASFORMAZIONE
DIGITALE

HTTP Headers

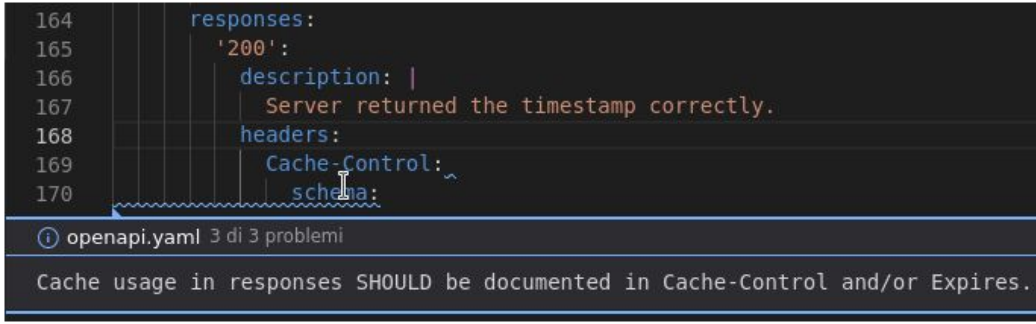
Not just adding or removing headers around!

- ✓ `Strict-Transport-Security`
- ✓ `X-XSS-Protection`
- ✓ `X-Content-Type-Options`
- ✓ `Content-Security-Policy`
- ✓ `X-Permitted-Cross-Domain-Policies`
- ✓ `Expect-CT`
- ✗ `Server`
- ✗ `X-Powered-By`

HTTP Headers

Document how you use Cache and Authorization requirements

→ **Cache-Control** - clarify in the specification how do you use cache

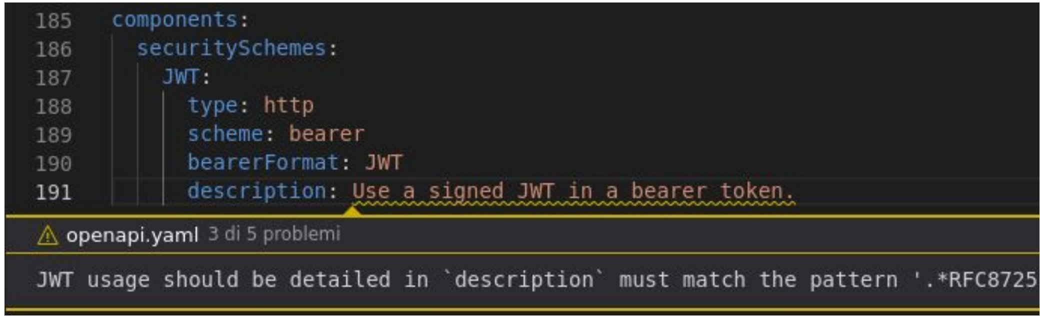


```
          responses:
            '200':
              description: |
                Server returned the timestamp correctly.
              headers:
                Cache-Control:
                  schema:
```

openapi.yaml 3 di 3 problemi

Cache usage in responses SHOULD be documented in Cache-Control and/or Expires.

→ **Authorization** - describe authentication and authorization headers and policies directly into the spec



```
components:
  securitySchemes:
    JWT:
      type: http
      scheme: bearer
      bearerFormat: JWT
      description: Use a signed JWT in a bearer token.
```

openapi.yaml 3 di 5 problemi

JWT usage should be detailed in `description` must match the pattern '.*RFC8725

HTTP Headers

Using secure serializers and parsers to manage headers reduces the attack surface of your APIs.

→ **Limit header parsing complexities** - eg. RFC8941 defines a safe serialization model for header values. There's a python library too!

```
from http_sfv import Dictionary

# Serializing a dictionary structured header
data = {"a": 1, "b": "two", "c": b'\x01\x02\x03'}
dict_header = Dictionary(data)
headers["Foo"] = {
"Foo": str(dict_header)
}

# Foo: a=1, b="two", c=:AQID:

# Parsing
dict_header = Dictionary()  # an empty header object
dict_header.parse(response.headers["Foo"])

assert dict_header["b"].value == "two"
```

Use interoperable subsets of JSON and XML

json-schema and XSD can model complex data-types, support nested structures and implementations have many nuances.

Some JSON hints:

- ➔ utf-8 only - RFC 8259
- ➔ encode floats/bigint as strings - rfc7493#section-2.2

```
>>> json.dumps({"1^1000": 1e1000})
'{"1^1000": Infinity}'
```

- ➔ beware of duplicate names - rfc7493#section-2.3

```
>>> json.loads('{"x": 1, "x": 2}')
{"x": 2}
```

- ➔ use strict parsers and don't truncate characters - consider that client and server will probably use different libraries. Custom parser may be less secure.

```
>>> custom_json_parser.loads('{"u": "ro\ud888ot"}')
{"u": "root"}
```

Read I-JSON specs RFC7493.

For XML, see the comprehensive OWASP XML Security Cheat Sheet

Limit item values and occurrencies.

Limit numbers and strings - in practice, every schema is limited in size. Start with reasonable values, then raise up.

Limit array sizes - arrays should be limited too

```
Uid:
  type: integer
  minimun: 0
  maximum: 1000

Title:
  type: string
  minLength: 5
  maxLength: 128
  pattern: '[a-zA-Z0-9 ]+'

Titles:
  type: array
  minItems: 1
  maxItems: 100
  uniqueItems: true
  items: {$ref: '#/Title'}
```

DIPARTIMENTO
PER LA TRASFORMAZIONE
DIGITALE

Constrain json-schema objects

Object validation is very tolerant

```
Account:                                ✓{"user": 123}
  type: object                          ✓{}
  properties:                           ✓{"name": ""}
    user: {$ref: "#/Uid"}
```

Specify required properties - or objects will always validate.

```
Account:
  type: object                         ✓{"user": 12}
  required: [user]                     ✓{
  properties:                             "user: 1
    user: {$ref: "#/Uid"}                 "infinite_list": [..]
                                         }
```

Disable additional properties - if you don't want unexpected fields

```
Account:
  type: object                           ✓{"user": 12}
  additionalProperties : false
  required: [user]                       ✓{
  properties:                               "user": 12,
    user:                                   "titles": []
      $ref: "#/Uid"                        }
    titles:
      $ref: "#/Title"
```

JWT and OAuth2

RFC8725 defines security best practices for JSON Web Tokens, and OAuth2 deprecated insecure flows.

JSON and XML flexibility increases their attack surface

JWT Best Current Practices (RFC8725) :

- use and verify appropriate algorithms, avoid substitution attacks
- use / validate the **aud**ience, **iss**uer and **sub**ject claims
- don't trust received claims

More JWT hints:

- limit temporal validity with **nbf** and **exp** claims
- add a token identifier (**jti** claim) to mitigate replay attacks
- don't use private keys associated to TLS certs to sign JWT to avoid cross-protocol attacks

OAuth2 hints:

- don't use "implicit" and "resource owner password" flows
- use "authorization code with PKCE" and "client credentials" with a jwt-bearer client_assertion_type (RFC7523)
- limit access token requests to specific resources using RFC8707

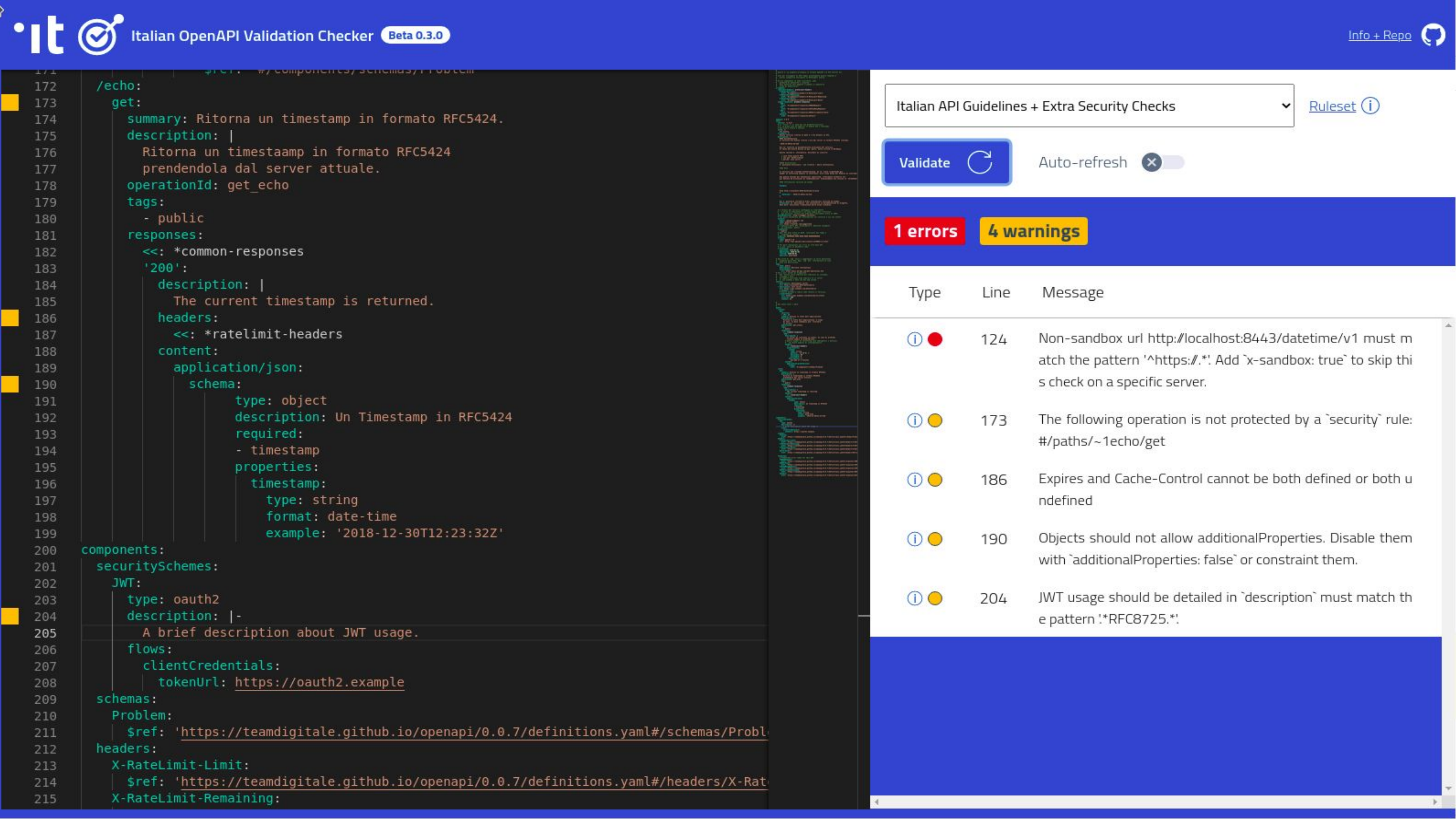
OpenAPI Checker

Guide implementers in checking the quality and security of APIs via the conformance with given rulesets, based on the *Spectral* open source tool.

APIs interactions and data schemas must be formally defined in "specification files" using an Interface Description Language. We can validate those files using automatic tools like italia/api-oas-checker!

- ➔ **security:** avoid common errors in API design (under-defined schemas, insecure methods, ...)
- ➔ **standards:** verify that Internet Standards are used correctly
- ➔ **usability:** the design is consistent with respect to the API domain and other usability rules (eg. field names, methods, ...)

[1] Spectral: https://github.com/stoplightio/spectral



```
171          $ref: '#/components/schemas/Problem
172  /echo:
173   get:
174    summary: Ritorna un timestamp in formato RFC5424.
175    description: |
176     Ritorna un timestaamp in formato RFC5424
177     prendendola dal server attuale.
178    operationId: get_echo
179    tags:
180     - public
181    responses:
182     <<: *common-responses
183     '200':
184      description: |
185       The current timestamp is returned.
186      headers:
187       <<: *ratelimit-headers
188      content:
189       application/json:
190        schema:
191           type: object
192           description: Un Timestamp in RFC5424
193           required:
194           - timestamp
195           properties:
196            timestamp:
197             type: string
198             format: date-time
199             example: '2018-12-30T12:23:32Z'
200 components:
201  securitySchemes:
202   JWT:
203    type: oauth2
204    description: |-
205     A brief description about JWT usage.
206    flows:
207     clientCredentials:
208      tokenUrl: https://oauth2.example
209  schemas:
210   Problem:
211    $ref: 'https://teamdigitale.github.io/openapi/0.0.7/definitions.yaml#/schemas/Probl
212  headers:
213   X-RateLimit-Limit:
214    $ref: 'https://teamdigitale.github.io/openapi/0.0.7/definitions.yaml#/headers/X-Rat
215   X-RateLimit-Remaining:
```

Italian API Guidelines + Extra Security Checks

Ruleset

Validate

Auto-refresh

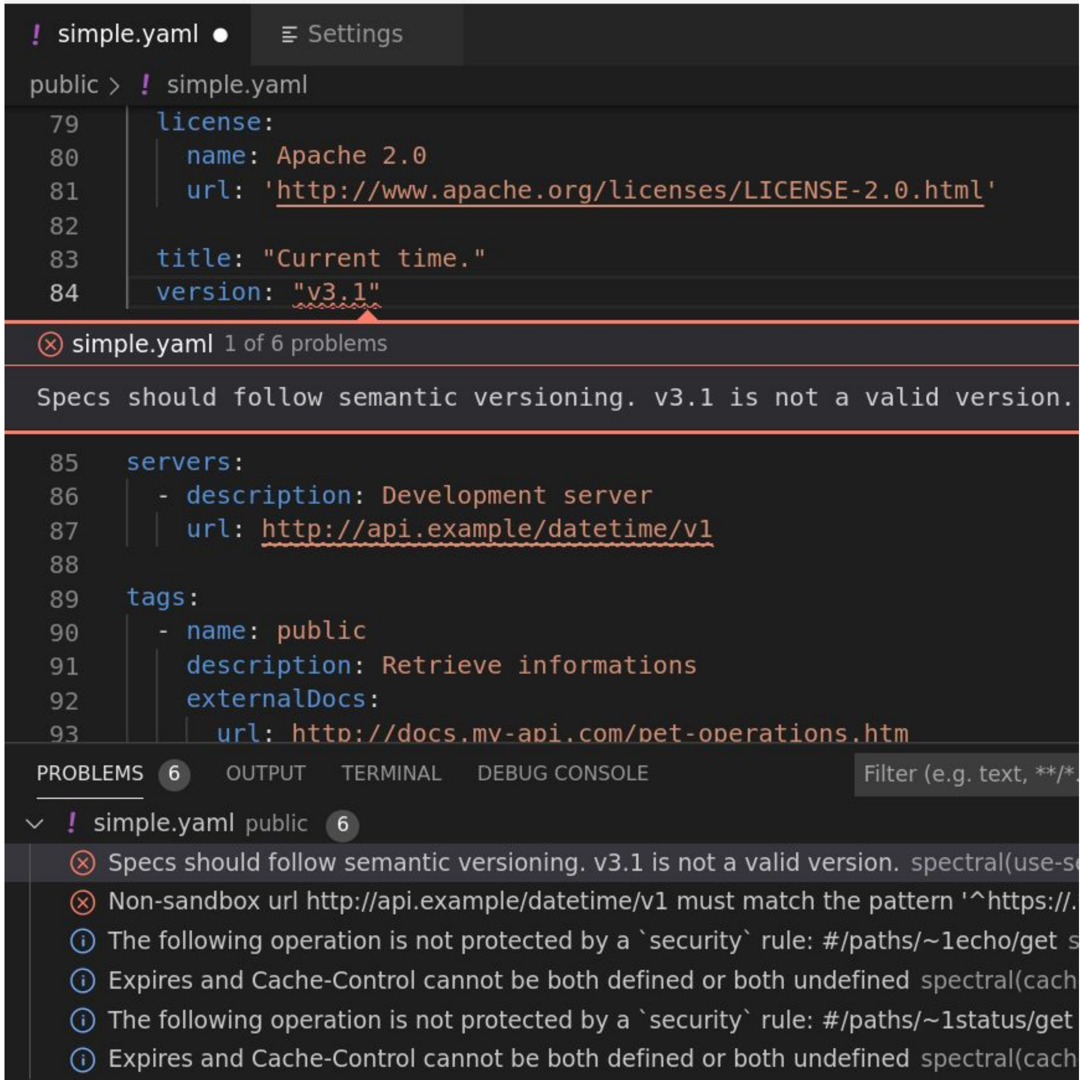
1 errors 4 warnings

Type	Line	Message
●	124	Non-sandbox url http://localhost:8443/datetime/v1 must match the pattern '^https://.*'. Add `x-sandbox: true` to skip this check on a specific server.
●	173	The following operation is not protected by a `security` rule: #/paths/~1echo/get
●	186	Expires and Cache-Control cannot be both defined or both undefined
●	190	Objects should not allow additionalProperties. Disable them with `additionalProperties: false` or constraint them.
●	204	JWT usage should be detailed in `description` must match the pattern '.*RFC8725.*'.

italia/api-oas-checker

Guideline support tools

Coherent and secure by design,
integrating checks in your IDE.



```
license:
  name: Apache 2.0
  url: 'http://www.apache.org/licenses/LICENSE-2.0.html'

title: "Current time."
version: "v3.1"
```

simple.yaml 1 of 6 problems

Specs should follow semantic versioning. v3.1 is not a valid version.

```
servers:
  - description: Development server
    url: http://api.example/datetime/v1

tags:
  - name: public
    description: Retrieve informations
    externalDocs:
      url: http://docs.my-api.com/pet-operations.htm
```

Specs should follow semantic versioning. v3.1 is not a valid version. spectral(use-s
Non-sandbox url http://api.example/datetime/v1 must match the pattern '^https://.
The following operation is not protected by a `security` rule: #/paths/~1echo/get s
Expires and Cache-Control cannot be both defined or both undefined spectral(cach
The following operation is not protected by a `security` rule: #/paths/~1status/get
Expires and Cache-Control cannot be both defined or both undefined spectral(cach

DIPARTIMENTO
PER LA TRASFORMAZIONE
DIGITALE

Next Steps

Involve communities, countries and API experts in the project

- **Usability:** improve the web interface, which is the showcase of the API Guidelines
- **Security:** create a community around the identification and implementation of more security rules
- **Coherence:** improve the coverage of the Italian API Guidelines and evolve the project together with the framework
- **Community**: synergies and contributions to related and underlying projects

Follow us

https://innovazione.gov.it/

@InnovazioneGov

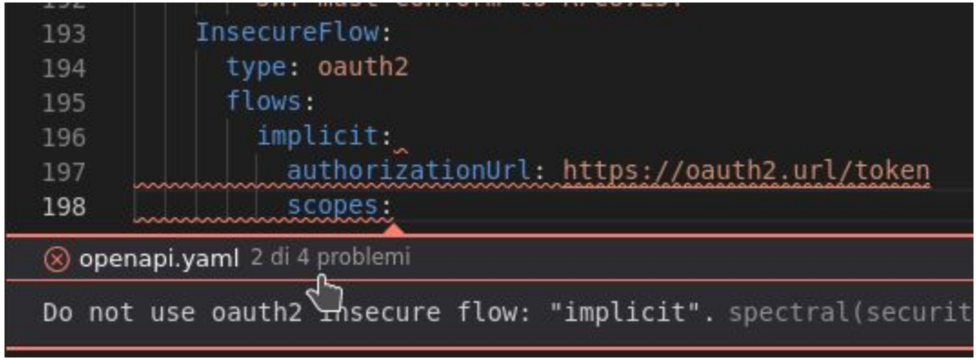
@DipartimentoTrasformazioneDigitale

@company/ministeroinnovazione/

Roberto Polli

- Email: roberto@teamdigitale.governo.it
- GitHub: ioggstream

DIPARTIMENTO
PER LA TRASFORMAZIONE
DIGITALE



```
    InsecureFlow:
      type: oauth2
      flows:
        implicit:
          authorizationUrl: https://oauth2.url/token
          scopes:
```

openapi.yaml 2 di 4 problemi

```
Do not use oauth2 insecure flow: "implicit". spectral(securit
```