Designers

Developers

Community lab
Developers Italia e Designers Italia

25 maggio 2022

DIPARTIMENTO
PER LA TRASFORMAZIONE
DIGITALE

Progettiamo e sviluppiamo servizi pubblici, insieme!

DTD @IUAV 2022

DIPARTIMENTO
PER LA TRASFORMAZIONE
DIGITALE

COMMUNITY LAB

Scopri e condividi gli strumenti della community!

Perché i *community lab?*

5 appuntamenti per condividere conoscenze e soluzioni per migliorare i servizi pubblici digitali.

A chi sono dedicati?

➔ Alle figure tecniche della community, in particolare a progettisti, sviluppatori open source e maintainer PA, di sue forme aggregative, in house e fornitori.

Dove puoi trovare maggiori informazioni:

Pubblicheremo di volta in volta gli appuntamenti su siti di Developers Italia e Designers Italia, sui profili social e su Slack Developers.

Il format

➔ presentazione dello strumento, vantaggi, istruzioni per l'uso ed esempi pratici di utilizzo;

➔ spazio per la discussione.

Presentazioni - 35’

Progettiamo API REST per la Pubblica Amministrazione

DIPARTIMENTO
PER LA TRASFORMAZIONE
DIGITALE

Ciao a tutti,

Sono **Roberto Polli, API Expert del Dipartimento per la trasformazione digitale.**

Mi occupo di **standardizzare in modo semplice e sicuro** i servizi pubblici digitali creati dalle 20k pubbliche amministrazioni.

Oggi vi parlerò di API
In particolare:

- come progettare API interoperabili
- usare API Checker per adeguarle alle Linee Guida Agid su API e Sicurezza

Prerequisiti: progettisti software con esperienza su REST e/o SOAP

DIPARTIMENTO
PER LA TRASFORMAZIONE
DIGITALE

Perché lo ritengo uno strumento indispensabile nello sviluppo di servizi pubblici digitali?

Le Linee guida di interoperabilità sono uno strumento fondamentale per **progettare API semplici, efficienti e sicure**.
Sono ispirate alle più recenti best practice, ed hanno richiesto un lungo lavoro.

Volevamo rendere più fruibile tutta questa conoscenza: non c'è cosa più difficile infatti che progettare software semplice.

API Checker permette di **verificare sin dalla progettazione** il rispetto di criteri di **interoperabilità e sicurezza**.

API Defibrillatori

Progettiamo un'ipotetica API che permette alla popolazione di localizzare i defibrillatori e agli enti di gestirne l'elenco.

Il workshop mostra una **strategia di progettazione semplificata**. Affronteremo un'altra volta temi fondamentali quali:

1. caching e requisiti di robustezza (paginazione, rate limit) per limitare l'uso delle risorse infrastrutturali;
2. profili di autorizzazione per evitare il transito di informazioni eccedenti;
3. l'integrazione con le piattaforme previste dal PNRR: il Catalogo API PDND e il Catalogo nazionale dati per la semantica.

Provate ad aggiungere i punti 1 e 2 a casa e condividere le vostre implementazioni sul canale #api di https://slack.developers.italia.it

Partiamo col progettare un'API che permette:

- alle persone di localizzare i defibrillatori
- agli enti di gestirne l'elenco

Progettare API

Modellare utenti, obiettivi, dati e operazioni con gli API Canvas

Mappiamo **chi, che cosa, come e perché** su una griglia (API Canvas) ispirata a The design of Web APIs by Arnaud Lauret per

1. *Definire* il servizio e i suoi casi d'uso;
2. *Individuare* risorse e azioni secondo un approccio REST;
3. *Descrivere* risorse ed azioni in OpenAPI 3.

Qui il canvas usato per il workshop e l'OpenAPI finale (in swagger editor e nel API Checker).

Iniziamo con le informazioni di catalogo:

```
titolo: Defibrillatori

sottotitolo: Mostra i defibrillatori in un'area
e trova quello più vicino
```

Business Canvas - Defibrillatori

Whos	Whats	Hows	Inputs	Outputs	Goals
Chi sono gli utenti?	**Cosa possono fare?**	**Come lo fanno?**	**Che dati inviano? Da dove li prendono?**	**Cosa ottengono dall'API? Come lo usano?**	**Che obiettivi hanno?**
Soggetti privati	Pubblicare informazioni sui defibrillatori in una determinata area	Strumenti per la gestione di mappe	Id. dell'area individuato dal richiedente	Una lista di defibrillatori con le relative coordinate GPS e il loro stato (disponibile/non disponibile).	Pubblicare l'elenco dei defibrillatori su mappe o su schermi (hotel, bus, metro, ...)
Open data / civic hacker / giornalisti / enti	Monitorare la distribuzione dei DAE in un territorio.	Strumenti di data science: bulk download o client specifici.	Identificativo dell'area e timestamp: individuati dal richiedente	Una lista di defibrillatori con le relative coordinate GPS e il loro stato (disponibile/non disponibile).	Informare il pubblico sulla distribuzione dei defibrillatori. Valutare la copertura del servizio di defibrillatori.
Cittadini	Localizzare il defibrillatore più vicino / localizzare i defibrillatori in un'area	Via web/mobile apps	Coordinate del richiedente: preso dall'applicazione.	Una lista di defibrillatori con le relative coordinate GPS e il loro stato (disponibile/non disponibile).	Individuare i defibrillatori più vicini
Operatori della sanità	Localizzare il defibrillatore più vicino / localizzare i defibrillatori in un'area	Via web/mobile apps	Coordinate GPS: selezionate da una mappa. Indirizzo: indicato dall'operatore.	Una lista di defibrillatori con le relative coordinate GPS e il loro stato (disponibile/non disponibile).	Individuare i defibrillatori più vicini
Operatori della sanità	Gestire le informazioni di un defibrillatore	Via web/mobile apps. Via sistemi automatici di notifica (IoT).	Informazioni sul defibrillatore, inserite da un soggetto	Stato della modifica	Gestire lo stato dei defibrillatori.
Utenti avanzati dell'API	Recuperare lo stato del servizio	Via strumenti specifici / dashboard	-	Lo stato dell'API	Monitorare lo stato del servizio, pubblicare metriche sullo stato del servizio.

API Canvas: risorse, proprietà e verbi

Whos	Whats	Hows	Inputs	Outputs	Goals
Chi sono gli utenti?	**Cosa possono fare?**	**Come lo fanno?**	**Che dati inviano? Da dove li prendono?**	**Cosa ottengono dall'API? Come lo usano?**	**Che obiettivi hanno?**
utente non autenticato	Consulta defibrillatori	Elenca **defibrillatori**	area amministrativa, identificata da un vocabolario controllato	Una lista di defibrillatori con le coordinate GPS e il loro stato (disponibile/non disponibile).	Lettura del dataset
			coordinate GPS: del richiedente prese dal device,dell'indirizzo, prese da un'API di geocoding esterna; raggio.		
			area amministrativa, poligono		
		Mostra **defibrillatore**	identificativo defibrillatore	Informazioni defibrillatore, mostrato nell'applicazione	
operatore	Gestisci defibrillatori	Aggiungi **defibrillatore**	informazioni defibrillatore	Informazioni defibrillatore, mostrato nell'applicazione	Modifica del dataset
operatore		Modifica **defibrillatore**	identificativo e informazioni defibrillatore	Informazioni defibrillatore, mostrato nell'applicazione	
client-IoT		Modifica **defibrillatore** (stato)	identificativo e stato defibrillatore	Informazioni defibrillatore, verificato dal sistema automatico	
operatore		Cancella **defibrillatore**	identificativo defibrillatore	Conferma eliminazione	
utente non autenticato	Consulta lo stato del servizio	Mostra **stato del servizio**	-	Stato dell'API	Lettura stato del servizio

API Defibrillatori - user, operation, request, response

Whos	Whats	Hows	Inputs	Outputs	Goals
security / scopes	**operationId**	**method**	**parameters / requestBody**	**status code / content**	**tags**
public	lista_defibrillatori	GET /defibrillatori	area_amministrativa (codice istat), paginazione	200 + Defibrillatori geo+json 4xx|5xx + Problem problem+json	publish
			latitude, longitude in formato WGS84, raggio, paginazione		
			area_amministrativa (codice istat), paginazione, poligono in formato geojson		
			area_amministrativa (codice istat), bulk	200 + Defibrillatori text/csv	
	mostra_defibrillatore	GET /defibrillatori/{id}	id defibrillatore	200 + Defibrillatore geo+json 401|5xx + Problem problem+json	publish, manage
operatore	crea_defibrillatore	POST /defibrillatori	Defibrillatore geo+json	201 + Defibrillatore geo+json 401|5xx + Problem problem+json	manage
operatore	modifica_defibrillatore	PATCH /defibrillatori/{id}	id, json-patch+json variazione defibrillatore	200 + Defibrillatore geo+json 401|5xx + Problem problem+json	
client-IoT			id, json-patch+json stato defibrillatore	200 + Defibrillatore geo+json 401|403|5xx + Problem problem+json	
operatore	elimina_defibrillatore	DELETE /defibrillatori/{id}	id defibrillatore	204 + No content 401|5xx + Problem problem+json	
public	mostra_status	GET /status	-	200 + Problem problem+json 5xx + Problem problem+json	monitoring

API Defibrillatori - workshop goals

Whos	Whats	Hows	Inputs	Outputs	Goals
security / scopes	operationId	method	parameters / requestBody	status code / content	tags
public	**lista_defibrillatori**	**GET /defibrillatori**	**area_amministrativa (codice istat), x**	**200 + Defibrillatori geo+json 4xx|5xx + Problem problem+json**	**publish**
public	**lista_defibrillatori**	**GET /defibrillatori**	**latitude, longitude in formato WGS84, raggio**	**200 + Defibrillatori geo+json 4xx|5xx + Problem problem+json**	**publish**
public	lista_defibrillatori	GET /defibrillatori	area_amministrativa (codice istat), paginazione, poligono in formato geojson	200 + Defibrillatori geo+json 4xx|5xx + Problem problem+json	publish
public	lista_defibrillatori	GET /defibrillatori	area_amministrativa (codice istat), bulk	200 + Defibrillatori text/csv	publish
public	mostra_defibrillatore	GET /defibrillatori/{id}	id defibrillatore	200 + Defibrillatore geo+json 401|5xx + Problem problem+json	publish, manage
operatore	crea_defibrillatore	POST /defibrillatori	Defibrillatore geo+json	201 + Defibrillatore geo+json 401|5xx + Problem problem+json	manage
operatore	modifica_defibrillatore	PATCH /defibrillatori/{id}	id, json-patch+json variazione defibrillatore	200 + Defibrillatore geo+json 401|5xx + Problem problem+json	manage
client-IoT	modifica_defibrillatore	PATCH /defibrillatori/{id}	id, json-patch+json stato defibrillatore	200 + Defibrillatore geo+json 401|403|5xx + Problem problem+json	manage
operatore	elimina_defibrillatore	DELETE /defibrillatori/{id}	id defibrillatore	204 + No content 401|5xx + Problem problem+json	manage
public	**mostra_status**	**GET /status**	**-**	**200 + Problem problem+json 5xx + Problem problem+json**	**monitoring**

Descrivere API

Una volta individuati gli obiettivi dell'API, la descriviamo in formato OpenAPI

Descrivere un'API permette raffinare la sua progettazione, coinvolgere collaboratori nella sua stesura e utilizzare strumenti di revisione automatici come API Checker che aiutano a progettare API sicure.

Definiamo ora:

- i metadati #/info
- schemi dati #/components/schemas
- autenticazione #/components/securitySchemes
- le operazioni #/paths

Nel lab di oggi ci concentriamo sulle operazioni:

- get_status
- get_defibrillatori

Metadati

Popolate prima la sezione #/info, in particolare i campi title, x-summary e description.

Saranno utili a chiarire gli obiettivi del servizio!

```
openapi: 3.0.2  # Valida online
info:
  version: "0.0.1"
  title: Defibrillatori
  x-summary: Individua i defibrillatori sul territorio
  description: |
    ## Defibrillatori

    Questa API permette a tutti di elencare
    e localizzare i defibrillatori presenti in un'area.
    Inoltre permette al personale autorizzato di gestire
    le informazioni in elenco

    E' utile perche'...
  termsOfService: 'https://tos.example.it'
  contact:
    email: roberto@teamdigitale.governo.it
    name: Dipartimento Trasformazione Digitale
    url: 'https://innovazione.gov.it'
paths: {}
```

Metadati

Usiamo i #/tags per raggruppare le varie operazioni.

Possiamo associarli ai "Goal" presenti nel canvas.

I server devono usare https!
Possiamo marcare quelli in sviluppo con x-sandbox.

```
openapi: 3.0.2  # Valida online
info: ...
tags:
  -   name: publish
      description: >-
        Pubblica le informazioni sui defibrillatori
  -   name: manage
      description: >-
        Permetti agli enti di gestire i defibrillatori
  -   name: monitor
      description: >-
        Monitoraggio del servizio
servers:
  -   url: https://api.example/defibrillatori/v0
      description: production
  -   url: /defibrillatori/v0
      description: development
      x-sandbox: true
paths: {}
```

Autenticazione

Il Canvas individua due tipologie di utenti: "non autenticato" e "operatore". Possono essere mappate tramite "scopes" o tramite altri meccanismi.

L'accesso pubblico va indicato direttamente nel campo `security` delle singole operazioni usando un oggetto vuoto `{}`.

```
openapi: 3.0.1
components:  # Valida online
  securitySchemes:
    OAuthToken:
      type: oauth2
      description: |-
        L'API utilizza per gli amministratori degli
        access token emessi dal server indicato.
        Le operazioni che non richiedono autenticazione
        usano lo schema `{}`.
      flows:
        clientCredentials:
          tokenUrl: https://as.example/token
          scopes:
            "operatore": >-
              Gestisci i defibrillatori.
```

Schemi delle risorse

Il Canvas individua queste risorse:

- Problem (messaggi d'errore)
- Defibrillatore (singolo record)
- Defibrillatori (elenco)
- Area Amministrativa
- Coordinate WGS84 e Raggio

Erogheremo i dati in formato GeoJSON

```
components:  # Valida online con profilo Security
  schemes:
    Defibrillatore:
      type: object
      additionalProperties: false
      properties:
        id: {type: string, format: uuid, ...}
        status: {type: string, enum: [attivo, non-attivo]}
        gestore: {type: string, description: "codice IPA"}
    DefibrillatoreGeo:
      type: object
      description: Un oggetto GeoJSon Feature
      properties:
        type: ...
        geometry: ...
        properties:
          $ref: "#/components/schemas/Defibrillatore"
```

Schemi delle risorse - Esempi

Associate ad ogni schema degli esempi

- aiutano a capire meglio
- servono a verificare l'implementazione

Tool come Connexion e Prism usano gli esempi per creare dei server mock

```
components:
  schemes:
    DefibrillatoreGeo:
      type: object
      description: Un oggetto GeoJSon Feature
      properties:
        type: ...
        geometry: ...
        properties:
          $ref: "#/components/schemas/Defibrillatore"
    example:
      type: Feature
      geometry:
       type: Point
       coordinates: [ 40.123, 14.123]
      properties:
        id: 7411c337-2f9a-4f6b-911d-d1b814ea4c57
        status: attivo
        gestore: C_E472
```

API Operations

La prima operation dell'API è associata al path `/status` e serve a testare il funzionamento del servizio

Le chiavi dell'oggetto `responses` sono tutte stringhe, e `default` indica un errore generico.

Lo schema della risposta è basato su RFC7807

```yaml
paths:
  /status:
    get:
      summary: Mostra lo stato dell'API.
      operationId: mostra_status
      description: Ritorna OK se l'API funziona.
      responses:
        "200":
          description: L'API funziona
          content:
            application/problem+json:
              schema:
                $ref: "#/components/schemas/Problem"
        "default":
          $ref: "#/components/responses/default"
```

API Operations

Questa operazione risponde al goal di pubblicazione dei dati

Non serve autenticazione

Ricerca i defibrillatori per:

- o area amministrativa
- o alla tripla latitudine, longitudine e raggio

Ritorna un GeoJSON

```
paths:              # Valida online
  /defibrillatori:
    get:
      operationId: lista_defibrillatori
      tags: [publish]  # Pubblica i dati
      security: [{}]   # Nessuna autenticazione
      parameters:
      # o un'area amministrativa
      - $ref: "#/components/parameters/AreaAmministrativa"
      # o coordinate e raggio
      - $ref: "#/components/parameters/Latitudine"
      - $ref: "#/components/parameters/Longitudine"
      - $ref: "#/components/parameters/Radius"
      responses:
        "200":
          description: Elenco dei defibrillatori
          content:
            application/geo+json:
              schema:
                $ref:"#/components/schemas/DefibrillatoriGeo"
```

Esercizi

Usando il validatore, aggiungete queste feature all'OAS

1. Supporto per il profilo "sistema-IoT" che può modificare il solo "stato" di un defibrillatore in #/components/securitySchemes/OAuthToken
2. L'operation modifica_defibrillatore che ritorna anche uno status code appropriato se un "sistema-IoT" prova a fare una modifica non autorizzata
3. Gli header di caching
4. Gli header di rate limit e le altre considerazioni indicate nella sezione "Robustezza" delle Linee guida
5. Meccanismi di paginazione nelle richieste e nelle risposte
6. Un parametro di ricerca basato su un poligono geojson

Q&A - 15’ Domande e risposte

Conclusioni

- **Scopri API Checker per progettare API REST:** https://italia.github.io/api-oas-checker/
- **Vuoi saperne di più? Leggi il nostro articolo:** https://medium.com/developers-italia/openapi-checker-il-verificatore-delle-interfacce-digitali-api-1d50b978c8c5
- **Aiutaci a migliorare**: https://ec.europa.eu/eusurvey/runner/community-lab-25-05
- **Prossimi appuntamenti:** 29/06 - community lab - Scopriamo Docker e i container
- **Resta aggiornato sui nostri canali per saperne di più**

Designers

Developers

Rimaniamo in contatto

- **Chat** *slack.developers.italia.it*
- **Siti web** *developers.italia.it* & *designers.italia.it*
- **Mail** *contatti@{developers, designers}.italia.it*
- **Twitter** *@developersITA* e *@DesignersITA*
- **Blog** *medium.com/@Developers_Italia* e *medium.com/designers-italia*

DIPARTIMENTO PER LA TRASFORMAZIONE DIGITALE