INWT Statistics

AP7 Abschlussbericht

Schnittstellen

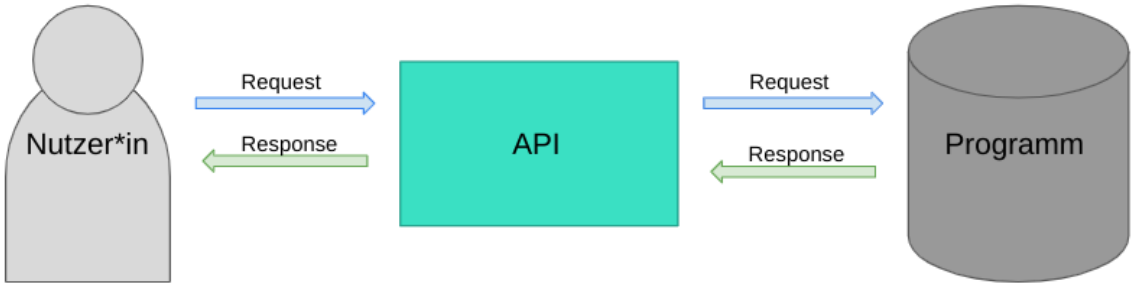
Im Folgenden beschreiben wir die API für die externe und interne Verwendung der Schadstoffprognosen. Der Zugriff auf externe Datenquellen für die Modellierung wurde bereits im AP3-Abschlussbericht beschrieben.

Inhaltsverzeichnis

API

Zunächst wird erklärt, was eine API ist und welche Technologie für die Umsetzung der API verwendet wird.

Bei einer API (**A**pplication **P**rogramming **I**nterface) handelt es sich um eine Schnittstelle, die die Kommunikation mit und den generellen Zugang zu einem Programm ermöglicht. Eine oft genutzte Analogie zur Verdeutlichung des Konzepts einer API ist der Restaurantbesuch: Die Nutzer*in stellt den Gast im Restaurant dar. Die API fungiert als Kellner*in und dient als Kommunikationsbrücke zwischen Gast und Küche. Das Programm selbst ist die Küche bzw. die Köch*in.



Mit dieser Analogie kann auch die Frage nach der Berechtigung für die API beantwortet werden: Würden alle Gäste - ohne Speisekarte, ohne Kellner*in - direkt in die Küche laufen, um Essen und Getränke zu bestellen, entstünde im besten Falle ein großes Durcheinander. Die Kellner*in kann in Verbindung mit der Speisekarte die Anweisungen für die Küche in das richtige Format bringen und Wünsche, die nicht zu erfüllen sind, ablehnen und dafür sorgen, dass die Gäste die richtige Bestellung erhalten.

Im Falle der Luftschadstoffprognose wird mit der API eine Datenquelle, in Form der Prognosen, erreichbar gemacht. Der Zugang über eine Schnittstelle hat gegenüber der direkten Bereitstellung der Daten (z. B. in Form von Dateien oder dem Zugang zu einer Datenbank) mehrere Vorteile:

- **Standard-Format:** Die Daten sind standardisiert und liegen damit immer im gleichen Format vor. Auch die Abfrage der Daten ist standardisiert und dokumentiert.
- **Öffentliche Bereitstellung:** Die Daten können relativ einfach der Allgemeinheit zur Verfügung gestellt werden.
- **Sicherheit und Flexibilität:** Diese wird gewährleistet durch die Trennung von Client und Server, weil Infrastruktur und Prozesse abgekoppelt sind vom Client - dem öffentlichen Zugang. Die Trennung ermöglicht es außerdem, unabhängig voneinander Änderungen an den Systemen durchzuführen.

REST API

Bei einer REST API handelt es sich um eine Schnittstelle, der ein bestimmtes Schema zugrunde liegt. Neben REST gibt es auch das SOAP-Protokoll, bei dem es sich um ein Standardprotokoll für Netzwerkkommunikation handelt. Wie die Bezeichnung "Protokoll" vermuten lässt, ist es strenger und umfangreicher in der Umsetzung als ein Schema oder Paradigma. Das REST-Schema hingegen kann relativ flexibel umgesetzt werden und führt zu schlankeren und schnelleren Lösungen als eine Umsetzung mit SOAP; es entwickelt sich zunehmend zum Standard für öffentliche APIs. REST steht für **Re**presentational **S**tate **T**ransfer und ist eine Spezifikation des bekannten HTTP (**H**yper**t**ext **T**ransfer **P**rotocol), das für das Aufrufen von Internetseiten genutzt wird. Damit ist REST im Prinzip ein auf HTTP basierendes Regelwerk für den Austausch von Daten im Netz. Dies umfasst unter anderem:

- **Client-Server-Model:** Die Daten sind getrennt von der Schnittstelle, d. h. die API und in diesem Fall die Datenbank stellen zwei verschiedene Software-Systeme dar.
- **Zustandslosigkeit:** Die Kommunikation zwischen Client und Server soll zustandslos erfolgen. D. h. jede Anfrage (Request) beinhaltet alle benötigten Informationen zur Verarbeitung durch den Server. Damit gibt es keine Sessions oder Cookies, auf die zurückgegriffen werden kann.
- **Einheitliche Schnittstelle:** Der Zugriff auf die API erfolgt immer über die entsprechenden HTTP-Standardmethoden, z. B. GET, POST, PUT, etc.

FastAPI

Wie im oberen Abschnitt beschrieben, handelt es sich bei der Bezeichnung REST API um ein theoretisches Schema. Umgesetzt wurde die REST API für die Luftschadstoffprognose mit dem Python 3.7+ Framework FastAPI[1].

Zugriff

Die API für die Schadstoffprognosen ist unter

- https://api.fairq.inwt-statistics.de verfügbar.
  Eine graphische Benutzeroberfläche und Dokumentation findet sich unter
- https://api.fairq.inwt-statistics.de/docs.

Über die Dokumentation können ebenfalls alle Endpunkte mit Daten abgefragt werden. Es ist auch möglich, die API über gängige Kommandozeilentools wie cURL abzufragen. Der Zugriff ist aktuell noch mit einem Nutzernamen und einem Passwort gesichert, die der SenUMVK separat mitgeteilt wurden. Perspektivisch wird ein Zugang für die Öffentlichkeit ohne Authentifizierung angestrebt. Dieser öffentliche Zugang erfolgt spätestens nach Veröffentlichung der Daten über die Digitale Platform Stadtverkehr (DPS). Der öffentliche Zugang wird sowohl für die Endpunkte als auch für die oben beschriebene graphische Benutzeroberfläche und Dokumentation eingerichtet.

Dokumentation der Endpunkte

Die API verfügt über drei Endpunkte, die Prognosen zurückgeben: *stations*, *streets* und *grid*. Diese sind entsprechend via

- https://api.fairq.inwt-statistics.de/stations
- https://api.fairq.inwt-statistics.de/streets
- https://api.fairq.inwt-statistics.de/grid

zu erreichen. An allen drei Endpunkten werden die Luftschadstoffprognosen für die nächsten 93 Stunden (Erstellung der Vorhersage um 4 Uhr CET) oder für die nächsten

[1] fastapi.tiangolo.com

106 Stunden (Erstellung der Vorhersage um 15 Uhr CET) geliefert. Es steht immer nur die aktuellste Vorhersage zur Verfügung. Im Regelfall sind die Daten spätestens 2 Stunden nach Erstellung der Vorhersage über die API verfügbar, d. h. um 7 Uhr bzw. um 17 Uhr ist die jeweils neueste Vorhersage verfügbar.

Neben den oben genannten Endpunkten sind drei weitere Endpunkte geplant, die im Rahmen von AP6 umgesetzt werden. In der Diskussion sind unter anderem ein Endpunkt, der die Prognosen auf Tages- und LOR-Ebene aggregiert, sowie ein noch näher zu spezifizierender Endpunkt für den erlaubten Anteil an KFZ (Schwellenwert), um Luftschadstoff-Grenzwerte einzuhalten.

Parameter

Damit nicht immer alle Daten über die API abgerufen werden müssen, wird die Nutzung von Parametern bei den Endpunkten *streets* und grid angeboten. Dabei werden per Default immer die ersten 1000 Zeilen zurückgegeben. Um alle Daten dieser beiden Endpunkte abzufragen, können die Pfadparameter *skip* und *limit* genutzt werden. Mit *skip* wird angegeben, wie viele Zeilen übersprungen werden sollen, mit *limit*, wie viele Zeilen insgesamt zurückgegeben werden sollen. Um also die ersten 1000 Zeilen bei *streets* zu überspringen und die darauffolgenden 1000 Zeilen zurück zu geben, würde die URL folgendermaßen aussehen:

- https://api.fairq.inwt-statistics.de/streets?skip=1000&limit=1000

Oder um bei *grid* die ersten 2000 Zeilen zu überspringen und die darauffolgenden 1000 Zeilen zurück zu geben:

- https://api.fairq.inwt-statistics.de/grid?skip=2000&limit=1000

Endpunkt /stations

Über den *stations*-Endpunkt werden die Prognosen an den 18 Messstationen geliefert. Im Folgenden wird anhand eines Beispiels erklärt, welche Felder geliefert werden. Das Beispiel im json-Format findet sich auch unter “Datenformat: geoJSON”:

- geometry

    - type: Point
        - Typ der Geometrie. Point bedeutet: Der Standort jeder Messstation ist durch einen Punkt in Berlin definiert.
    - coordinates
        - Koordinaten der Messstation
    - crs
        - Coordinate-Reference-System: EPSG Spezifikation der Koordinaten
- properties
    - station_id
        - ID der Messstation
    - date_time_forecast_iso8601
        - Zeitpunkt der Prognoseerstellung
    - forecast_range_iso8601
        - Zeitlicher Bereich in dem sich die Prognosen in den nachfolgenden Arrays bewegen nach iso8601 Format: R[AnzahlZeitschritte]/[YYYY-MM-DDTHH:mm:ss.fffffZ/PT1H]
        - Beispiel: R93/2022-11-29T03:00:00.000000Z/PT1H
            - R93 steht für die Anzahl der Zeitschritte in dem Array
            - Datum und Uhrzeit in Zeitzone Z = zero meridian (UTC)
            - PT1H: stündlicher Abstand der Vorhersagen
    - no2
        - Ein Array mit 93 Stickstoffdioxid-Prognosen gerundet auf eine Nachkommastelle. Der erste Wert ist die Vorhersage zum Zeitpunkt der Erstellung der Vorhersage (“Nowcast”), gefolgt von den 92 Vorhersagen (bis 02. Dezember 00:00 Uhr CET). Fehlende Werte sind mit -999 kodiert.
    - pm10
        - Dieselbe Struktur wie für NO_2, allerdings mit den Vorhersagen für PM_{10}
    - pm2.5
        - Dieselbe Struktur wie für NO_2, allerdings mit den Vorhersagen für $PM_{2.5}$

Endpunkt /streets

Über den *streets*-Endpunkt werden Prognosen entlang der 9355 Straßen in Berlin mit den StEP-Klassen I-IV sowie 0 zurückgeliefert.
Im Vergleich zu dem *stations*-Endpunkt bestehen folgende Unterschiede:

- geometry
  - type: LineString
    - Typ der Geometrie. Die Linien des Straßenabschnitts ergeben sich durch eine Anfangs- und Endkoordinate, deren Wert unter coordinates angegeben ist..
  - coordinates
    - Anfangs-, sowie Endkoordinate des Straßenabschnitts.
- properties
  - element_nr
    - ID des Straßenabschnitts aus dem FIS-broker mit dem Namen “element_nr”

Endpunkt /grid

Der *grid*-Endpunkt liefert die Prognosen in einem 50x50m^2-Gitternetz im gesamten Stadtgebiet von Berlin. Hierfür werden 360.765 Gitterzellen definiert, deren Mittelpunkte innerhalb von Berlin liegen. **Vorsicht**: Dieser Endpunkt liefert mehrere GB Daten pro Vorhersage zurück. Im Vergleich zu dem *stations*-Endpunkt bestehen folgende Unterschiede:

- geometry
  - type: Point
    - das Koordinatenpaar entspricht dem Mittelpunkt der 50x50 m^2 Gitterzelle
- properties
  - id
    - die ID der Gitterzelle (zwischen 1 und 695.686)

Validierung der API-Endpunkte

Um die Qualität und Stabilität der Endpunkte zu gewährleisten, arbeiten wir mit Redash und Pydantic. Redash ist ein Monitoring-System, das die Daten in der Datenbank überwacht. Pydantic ist eine Python-Bibliothek, auf der FastAPI basiert und welche es erlaubt, den Input und Output der API genau zu definieren.

Redash

Redash (https://redash.io/) ist eine Software, mit Hilfe derer man basierend auf einer Datenbank Dashboards erstellen und Alarme definieren kann.
Redash wird in diesem Projekt verwendet, um beispielsweise die Schadstoffprognosen kontinuierlich auf Plausibilität und zufriedenstellende Prognosegüte zu prüfen und für die SenUMVK auch vor der Anbindung der API schnell zugänglich zu machen. Die Dashboards wurden ausführlicher in den Abschlussberichten zu AP4 (Schwellenwerte) und AP5 (Modellevaluation) beschrieben.
Außerdem wurden umfangreiche Alarme definiert, um eine kontinuierliche Prognoseverfügbarkeit sicherzustellen. Dies umfasst folgende Checks:

- Stimmt die Anzahl der prognostizierten Werte? (D.h. Reichen die Prognosen genau so weit in die Zukunft wie gewünscht?)
- Werden die Prognosen regelmäßig zu den geplanten Zeitpunkten in die Datenbank geschrieben?
- Liegen die Prognosen in einem plausiblen Bereich zwischen 0 µg/m³ und 100 µg/m³?

Alle Alarme wurden jeweils für die Prognosen an den Messstationen, auf dem Grid und an den Straßen eingerichtet. Schlägt ein Alarm an, werden mehrere Personen von INWT über ein Chat-System informiert und können zeitnah reagieren, um die Ursache zu identifizieren und zu beseitigen oder sich ggf. mit der SenUMVK über eine Lösung abzustimmen (z.B. bei Wegfall einer Datenquelle).

Pydantic

Mit der Python-Bibliothek Pydantic kann, in Abgrenzung zu Unit-Tests, in FastAPI eine Realtime-Validierung des Inputs (Request) und des Outputs (Response) stattfinden. Bei Unit-Tests wird in der Regel der Programm-Code auf Funktionsfähigkeit in verschiedenen Szenarien getestet, die dafür realitätsnahe Beispieldaten enthalten. Um aber nicht nur die einzelnen Komponenten zu testen, sondern explizit die Daten, mit denen die Komponenten arbeiten, können Schemata in Pydantic erstellt werden. Diese Schemata lassen sich flexibel definieren und an verschiedene Szenarien anpassen. Jede Response / jeder Request muss dann im Live-Betrieb genau in das ihr/ihm zugewiesene Schema passen. Ist dies nicht der Fall, wird eine entsprechende Fehlermeldung angezeigt.

Beispiel: Request Schema

Als Beispiel können die Pfadparameter *skip* und *limit* herangezogen werden. Diese wurden in einem Pydantic-Schema als Integer (ganze Zahlen) hinterlegt und eine erlaubte Spanne definiert. Gibt man hier eine Zahl außerhalb dieser Spanne ein oder übergibt hier Buchstaben an die Parameter (z. B. *?skip=zehn*) so erhält man eine aussagekräftige Fehlermeldung:

```
{"detail":[{"loc":["query","skip"],"msg":"value is not a valid integer","type":"type_error.integer"}]}
```

Beispiel: Response Schema

Die drei API-Endpunkte *stations*, *streets* und *grid* geben Daten im geoJSON-Format zurück. Dafür werden die Daten von der tabellarischen Form in der Datenbank in das geoJSON Format transformiert. Teil des geoJSON-Formats sind Koordinaten einer bestimmten Form, beispielsweise Punkt, Linie oder Polygon. Beim *stations*-Endpunkt werden immer Punkt-Koordinaten erwartet, die wie folgt aussehen:

```
{ "type": "Point",
    "coordinates": [388061,5822699]
}
```

D. h. für diesen Endpunkt kontrolliert das definierte Schema, dass der Typ ein “Point” ist und dass die Koordinaten eine Liste mit Integer-Werten sind. Würde hier der Typ “Line” oder “Polygon” übergeben, oder in den Koordinaten etwas anderes stehen, käme ebenfalls ein Fehler, der auf das entsprechende Verhalten hinweist.

Datenformat: geoJSON

JSON (**J**ava**S**cript **O**bject **N**otation) ist ein Format, das sowohl für Menschen als auch für Maschinen lesbar ist. REST APIs nutzen typischerweise JSON als Datenformat. Von JSON abgeleitet hat sich das geoJSON-Format, das speziell für die Darstellung von geografischen Daten entwickelt wurde. Zu den geografischen Typen von geoJSON gehören:

- Punkte, z. B. Adressen und Orte,
- Linien, z. B. Straßen, Routen und Grenzen,
- Polygone, z. B. Länder, Landkreise & Stadtteile,
- sowie Sammlungen dieser Typen.

Innerhalb des geoJSON-Formats werden Koordinaten normalerweise als Längen- und Breitengraden (WGS84) angegeben. Auf Wunsch der SenUMVK wurde für die Luftschadstoffprognose abweichend davon die X-, Y-Notation (EPSG:25833) gewählt.

Bei der Luftschadstoffprognose werden Sammlungen von geo-Features (*FeatureCollection*) zurückgegeben. Das wird im *type* auf der obersten Ebene definiert. Danach folgt die Liste mit den *features*. Die einzelnen Inhalte der features wurden im Abschnitt zur Dokumentation der Endpunkte erläutert. Hier ein Beispiel des *stations*-Endpunktes:

```
{
  "type":"FeatureCollection",
  "features":[
    {
      "type":"Feature",
      "geometry":{
        "type":"Point",
        "coordinates":[
          388061,
          5822699
        ],
        "crs":"EPSG:25833"
      },
      "properties":{
        "station_id":"010",
        "date_time_forecast_iso8601":"2022-11-28T03:00:00+00:00",
        "forecast_range_iso8601":"R93/2022-11-28T03:00:00.000000Z/PT1H",
        "no2":[
          17.5,
          18.0,
          19.8,
          …
        ],
        "pm10":[
          20.0,
          21.7,
          20.8,
          …
        ],
        "pm2.5":[
          16.7,
          16.5,
          16.8,
          …
        ]
      }
    }
  ]
}
```


Lizenzierung der Daten

Die Daten der Luftschadstoffprognosen unterliegen - wie die Verkehrsdaten - folgender Lizenz von GovData:

- https://www.govdata.de/dl-de/by-2-0

Diese Lizenz erlaubt unter Nennung der Quelle ausdrücklich die kommerzielle und nicht-kommerzielle Nutzung, und die Daten dürfen

1. vervielfältigt, ausgedruckt, präsentiert, verändert, bearbeitet sowie an Dritte übermittelt werden;
2. mit eigenen Daten und Daten Anderer zusammengeführt und zu selbständigen neuen Datensätzen verbunden werden;
3. in interne und externe Geschäftsprozesse, Produkte und Anwendungen in öffentlichen und nicht öffentlichen elektronischen Netzwerken eingebunden werden.

Bei der Nutzung der Daten muss folgende Quelle angegeben werden:
INWT Statistics GmbH, **F**orecasting **Air Q**uality, https://api.fairq.inwt-statistics.de/docs#/

Hosting der API

Die API wird auf Servern von Hetzner (https://www.hetzner.com) in Nürnberg gehostet.
Zum aktuellen Zeitpunkt ist die API noch über Authentifizierung mit Benutzername und Passwort abgesichert. In Zukunft wird die API öffentlich verfügbar sein.
Die Verbindung zur API wird über eine verschlüsselte SSL/TLS Verbindung hergestellt, die mit einem letsencrypt-Zertifikat abgesichert ist.
Damit die API einerseits gegen Angriffe abgesichert ist (z. B. DDoS-Attacken) und andererseits auch viele Anfragen performant beantworten kann, werden die Anfragen an die API aus einem Cache beantwortet. Das bedeutet, dass die Ergebnisse, die die API zurückgeben muss, in einer Art Zwischenspeicher abgelegt werden. Dieser wird regelmäßig mit Daten aus der Datenbank aktualisiert. Dadurch müssen die Ergebnisse nicht bei jeder erneuten API-Anfrage zeitaufwändig zusammengestellt werden.

Hosting von Redash

Der Zugriff auf Redash ist über eine Authentifizierung mit Benutzername und Passwort abgesichert. Außerdem ist der Zugriff nur über Server der Senatsverwaltung und aus dem Büro von INWT möglich.
Links zu den freigegebenen Dashboards:

- FAirQ - Neueste Prognosen: Grid
  https://dashboard.fairq.inwt-statistics.de/public/dashboards/HM7ja2d352pdgg9dyvc2n3tTmR64Kwbny1WsP1cA
- FAirQ - Neueste Prognosen: Messstationen
  https://dashboard.fairq.inwt-statistics.de/public/dashboards/i3HRjPOoXiyUesmRCIvVGubj4pykozIpwcLm3FxG
- FAirQ - Messstationen: Überblick
  https://dashboard.fairq.inwt-statistics.de/public/dashboards/dSmNjFyCwdhLehhkLXE04zOgdSnwkMfBEhNiyceT
- FAirQ - Messstation: Details
  https://dashboard.fairq.inwt-statistics.de/public/dashboards/E8Veq6sejnrM3BI9DAkISsdmtivKpNhaHBebUztm
- FAirQ - Modellgüte nach Prognosehorizont
  https://dashboard.fairq.inwt-statistics.de/public/dashboards/JeiFHuRvVXHwarhfvdEXpkteDTxm7FYl2SLLrVKl