TECHPARK SÜDTIROL / ALTO ADIGE

NOI A.G. / S.p.A.
Roberto Cavaliere
r.cavaliere@noi.bz.it
T +39 0471 066 676

Bolzano, 16.06.2021

Preliminary market consultation and contextual request for quote

Project and CUP: ***D39G18000040002- MENTOR – INTERREG ITALY-SWITZERLAND***

Dear supplier,

NOI SpA intends to initiate a preliminary market consultation pursuant to art. 20 of LP no. 16/2015 and art. 40 of Directive 2014/24/EC for the implementation of several software components ("**Data Collector**"), for the integration of different datasets to be integrated in the Open Data Hub that can support the development of the Mobility-as-a-Service (MaaS) vision in South Tyrol. These components have the function to retrieve on a real-time basis the data made available by different data provides through machine-readable interfaces (API) and integrate it in the Open Data Hub with the other already available datasets. More information and specifications about the aforementioned project and this market request are described in more detail in the Annex.

NOI SpA invites all interested economic operators to participate by filling in an expression of interest, **also in the form of a quote**, in relation to the products and requirements described in the Annex.

DEADLINE FOR THE DELIVERY OF THE QUOTE (30.06.2021):
The quote is to be sent exclusively by e-mail to r.cavaliere@noi.bz.it

Best regards
Roberto Cavaliere

ANNEX: Details of the preliminary market consultation

1. The MENTOR project

The MENTOR project is a project financed by the **Interreg-V-A Italy-Switzerland** programme, coordinated by the Municipality of Merano and implemented in collaboration with **NOI Techpark**, **SASA**, the **Municipality of Brig-Glis** in the Canton of Valais and **Postauto**.

The aim of the project is to demonstrate a concept of "**Mobility-as-a-Service**" (MaaS) in the two pilot municipalities, which are representative of the Alpine environment. MaaS is currently one of the main drivers of technological innovation in mobility and is based on the idea of being able to combat the use of the private car with integrated packages of sustainable mobility services, which the user can easily use, book and pay.

The demonstration is carried out on three axes of intervention:

- **Pilot testing of new mobility services**, designed to be integrated with the public transport offer, which in the vision of the project partners must be the backbone of a MaaS ecosystem. Specifically, following services will be tested:
  - **Merano**: car pooling service, bike sharing service, on-demand service
  - **Brig-Glis**: on-demand transport service

  Pilot testing of MaaS tools, aimed at making access to these services as simple as possible:
  - **Merano:** evolution of the pilot application mobility.meran.eu. In particular, a real-time inter-modal routing function is under development, so that people can get a possibly sustainable travel option for each possible travel search from A to B.
- **Demonstration of automated mobility services**, aimed at creating a high acceptance by local travellers to use this new generation of vehicles. In particular, a first demonstration of small self-driving shuttles on predefined, traffic-free routes in Merano and Brig-Glis was carried out in 2019.

The project started in December 2019 with an expected duration of 3 years. More information on the MENTOR project is available at the following links: [1] [2].

1. "On-demand service" dataset

In the scope of the MENTOR service a pilot experiment with an on-demand service called "Callbus" is currently carried out. It is about a transportation service conceived to be a complementary service to both public transportation and taxi. The service is offered in a particular residential area of Merano, characterized by no public transportation due to very narrow streets and limited mobility demand, and aims to connect the people living in this area with other points of interest in Merano, including other nodes of the public transportation offer. All the details of the service can be found at the following page: https://www.comune.merano.bz.it/callbus

From a technological point of view, the service is implemented through a modern ICT solution based on a back-end system and an APP, one configured for the end users in order to make booking and receive real-time information, and one configured for the drivers so to get all detailed information about the service which has to be given. The APP can be downloaded here:

- **iOS**: https://apps.apple.com/it/app/callbus/id1565363821
- **Android**: https://play.google.com/store/apps/details?id=lu.cube4t8.callbus&gl=IT

The exchange of information takes place through an API, which is going to be used also to integrate relevant service information also in the Open Data Hub. Calls be made through HTTP GET, and are formatted as JSON. The end-point is documented at this link. An authentication mechanism with username / password is foreseen. In particular, the following data are going to be integrated:

- **stops / polygon**: the service is made up of so-called "pick-up / drop-off points", i.e. "virtual" bus stops where on-demand trips can start or end. A polygon is also defined: this is an area which defines the geographical area in which all pick-up / drop-off points are located. Stops outside this polygon are also foreseen, but they just represent possible origin / destinations of a trip. This means that a trip must always end or begin inside this polygon. These elements are simply modelled as an object with several attributes together with their geometry. The reference methods are "stops" and "polygon", respectively. A "stop" is characterized by the following fields:
    - **id**
    - **title**: the name of the stop (in Italian and German language)
    - **reference**: a code related to the stop
    - **type**: a label classifying the type of stop according to predefined values (e.g. "BUS_STOP")
    - **geoAddress**: a data structure characterized by the following attributes:
        - **id**
        - **title**
        - **category**
        - **position**: a data structure characterized by:
            - **type**: a label classifying the type of geometry provided (e.g. "Point")
            - **coordinates**: an array of values, formatted as [long, lat]
        - **streetAddress**: a data structure characterized by:
            - **state**: the name of the region
            - **county**: a code related to the region
            - **city**
            - **district**

        - **street**: the address associated to the position
        - **country**
        - **contrycode**
        - **housenumber**
        - **postalcode**
    - **groups**: a data structure characterized (id, name) which associate the stop to a certain group
    - **region**: a data structure characterized (id, name) which associate the stop to a certain region

  On the other side, a "polygon" is characterized by the following fields:

    - **id**
    - **name**
    - **description**
    - **active**: a boolean which indicates if the polygon is active or not
    - **borders**: a data structure characterized by the following attributes:
        - **type**: "Feature"
        - **properties**
        - **geometry**: a data structure characterized by:
            - **type**: a label classifying the type of geometry provided (e.g. "Polygon")
            - **coordinates**: an array of values, formatted as [long, lat], which describe the border of the polygon
    - **region**: a data structure characterized (id, name) which associate the polygon to a certain region

- **route and vehicle**: passengers can book a seat on a certain journey up to a certain time interval (currently 30 minutes) before the journey starts. This time interval is needed in order to allow the transportation service provider to properly organize the service and maximize the efficiency of the "pooling", i.e. the ability to match together the mobility demand of different people which can be slightly different both in space (choice of origin / destination) and in time (different timing needs). At the end what is computed is simply a route, i.e. an ordered list of pick-up / drop-off points that are served, coupled with their travel times. The reference method for the retrieval of this data is still under development and will be integrated under "activities". Through this method it will also be possible to see the real-time position of the vehicle. Following fields are going to be available:

    - **id**: the journey ID
    - **updatedAt**: timestamp related to last update of the data record
    - **state**: a label classifying the state of the journey (e.g. "PUBLISHED")
    - **plannedStartAt**: planned departure time (first stop)
    - **plannedEndAt**: planned arrival time (last stop)
    - **startedAt**: planned departure time (first stop)
    - **bus**
        - **licensePlateNumber**: plate number of the vehicle
        - **type**: a label classifying the type of the vehicle
        - **operator**: the reference operator carrying out the service
        - **capacityMax**: the maximum capacity of the vehicle
        - **capacityUsed**: the current level of occupancy of the vehicle
    - **latestPosition**
        - **recordTime**: timestamp related to the last GPS position recording
        - **position**: a data structure characterized by:
            - **type**: a label classifying the type of geometry provided (e.g. "Point")
            - **coordinates**: an array of values, formatted as [long, lat], which provides the real-time position of the vehicle
        - **heading**: information related to the heading provided by the GPS location system

        - **accuracyMeters**: information related to the positioning accuracy provided by the GPS location system
    - **itineraryDone**: data structure containing the information of the service already executed, structured as itineraryRemaining, see below.
    - **itineraryRemaining**: data structure containing the information of the service to be executed, structured as follows:
        - **type**: a label classifying the type of stop served (e.g. “ACTIVITY_HALT_PRIVATE”). If “ROUTE” it is followed by the field routeEncoded which contains the geometry of the path. The routes are formatted according the encoded polyline algorithm format (for more details ee: https://developers.google.com/maps/documentation/utilities/polylinealgorithm)
        - **time**: data structure containing all timing information related to the stop to be served:
            - **arrivalEarly**: timestamp related to the expected arrival time plus anticipation flexibility
            - **arrivalExpected**: timestamp related to the expected arrival time
            - **arrivalLate**: timestamp related to the expected arrival time plus delay flexibility
            - **arrivalAt**: timestamp related to the effective arrival time
            - **departureEarly**: timestamp related to the expected departure time plus anticipation flexibility
            - **departureExpected**: timestamp related to the departure arrival time
            - **departureLate**: timestamp related to the expected departure time plus delay flexibility
            - **departureAt**: timestamp related to the effective departure time
        - **stop**: data structure containing the information of to the stop to be served:
            - **id**: stop ID
            - **type**: a label classifying the type of stop served (e.g. “ON_DEMAND_BUS_STOP”)
            - **title**: a further description of the stop
            - **reference**: a reference of the region in which the stop is located
            - **position**: a data structure characterized by:
                - **type**: a label classifying the type of geometry provided (e.g. “Point”)
                - **coordinates**: an array of values, formatted as [long, lat]
            - **streetAddress**: a data structure containing the address information of the stop
        - **dropOffCapacities**: indicate the number of passengers that can be dropped off
        - **pickUpCapacities**: indicate the number of passengers that can be picked up

- **vehicle**: additional data related to a certain vehicle are available through the method “vehicle”. A “vehicle” is characterized by the following fields:
    - **licensePlateNumber**
    - **type**: vehicle type
    - **operators**: a data structure characterized by the following attributes:
        - **id**
        - **name**
        - **externalId**
        - **email**
        - **phoneNumber**
        - **address**
    - **position**: a data structure characterized by:
        - **type**: a label classifying the type of geometry provided (e.g. “Point”)
        - **coordinates**: an array of values, formatted as [long, lat], which provides the real-time position of the vehicle
    - **region**: a data structure characterized (id, name) which associate the polygon to a certain region
    - **capacityUsed**: indicates the number of passenger on board

- **capacityMax**: a data structure characterized by:
  - **SEAT_ADULT**: the maximum capacity of the vehicle
  - **ELECTRIC_WHEELCHAIR_XL**: the number of “extra-large” electric wheelchairs admitted on board
  - **BIKE**: the number of bikes admitted on board
  - **PET**: the number of pets admitted on board
  - **LUGGAGE**: the number of luggage admitted on board
  - **ELECTRIC_WHEELCHAIR**: the number of electric wheelchairs admitted on board
  - **WEIGHT**: the maximum weight admitted on board
  - **SEAT_CHILD**: the maximum number of seats for children
  - **WHEELCHAIR**: the number of wheelchairs admitted on board
  - **FOLDABLE_WHEELCHAIR**: the number of foldable wheelchairs admitted on board
  - **ELECTRIC_WHEELCHAIR_L**: the number of “large” electric wheelchairs admitted on board

2. “Bike counters” dataset

The outbreak of the COVID-19 pandemic has contributed to a further development of the cycling mobility almost everywhere. People have experienced how the majority of trips, in particular in the denser urban areas, can be efficiently covered with (e)-bikes. Based on that, a new challenge is starting to face out: the proper management and control not only of streets for motorized traffic, but also of dedicated cycling infrastructures.

In South Tyrol the major cycling infrastructures in the rural areas outside the main urban centers are managed by specific public organizations that provide common services to certain municipalities they represent, known in Italy as “*Comunità Comprensoriali*”. During the last years almost all these organizations have started to invest in a monitoring system for the cycling infrastructures, in particular sensors that are able to count and in some cases classify the cycling traffic passing at some relevant measuring point. At present there is about a dozen of sensors in the whole Province of Bolzano.

Most of these sensors are provided by the international company Eco-Counter (https://www.eco-counter.com/) and are typically collected in a back-end system once a day – so in this case it is about a “nearly” real-time dataset. This current situation could be however improved in the future, if needed. The back-end system makes available the data for 3rd parties through an API, which is documented at the following links:

https://apiadmin.eco-counter-tools.com/store/apis/info?name=API&version=1.0&provider=admin
http://eco-test2.com/apidoc/wso2/apidoc.html

3. “Bike chargers” dataset

An additional tool for promoting cycling mobility is the availability of dedicated infrastructures for simplifying parking and – in case of e-bikes – charging operations. One the above aforementioned organization, in particular the “*Comunità Comprensoriale Salto -Sciliar*” has recently made an investment for about 30 e-bike charging stations, in which people with an e-bike can have the possibility to charge the battery of his travel mean.

These stations are provided by the local company Ecospazio, located in Trentino, and are managed by a back-end system in real-time. Such back-end system can also integrate with other 3rd party system via API, which is currently implemented with two methods:

- **method 1**: retrieval of all available stations, characterized by their identifiers. Following attributes are returned:
    - **id**
    - **address**
    - **name**
    - **state** (mapped as a function of pre-defined states such as “READY”)
    - **lat** (latitude), expressed according the WGS84 coordinate reference system
    - **long** (longitude), expressed according the WGS84 coordinate reference system
- **method 2**: retrieval of the metadata and real-time data of a certain station.
    - **totalBays**
    - **freeBay**
    - **availableVehicles**
    - **bays**: each bay is characterized by the following information
        - **label**: bay name, formatted as [“name”]:bay ID
        - **state**.
        - **charger**: indicates if a charging operation is possible (expressed as boolean)
        - **use:** indicate how the bay can be used, according to pre-defined states such as “PRIVATE” (parking allowed only for certain users)
        - **vehicle**: structure that characterize the vehicle which is currently charging (structure = NULL if no vehicle parked here). Associated attributes are:
            - **batteryState**
            - **code** (vehicle identifier)
            - **vehicleType**: structure which characterize the type of vehicle
                - **name**: vehicle type, according to pre-defined values
                - **electric**: boolean which indicates if the vehicle is electric or not

The methods can be called via HTTP GET, and the response are formatted as JSON. The end-point is https://maas.ecospazio.it/api/v1. The access is secured by an authentication mechanism based on header token.

4. “Traffic events” dataset

The Province of Bolzano and in particular its regional Traffic Information Centre makes available as open data the information related to traffic events such as accidents, roadworks, closures, etc., that are related not only to streets destined to motorized traffic but also to the cycling infrastructures. The data provided by this API is the one shown on the web application available at https://verkehr.provinz.bz.it/

Basically, two methods are available:

- **Roadworks / closures**: http://geoservices.retecivica.bz.it/geoserver/p_bz-traffic/ows?service=WFS&version=1.0.0&request=GetFeature&typeName=p_bz-traffic:ConstructionSites_Roadblocks&maxFeatures=5000&outputFormat=application/json&srsName=EPSG:4326
- **Traffic information:** http://geoservices.retecivica.bz.it/geoserver/p_bz-traffic/ows?service=WFS&version=1.0.0&request=GetFeature&typeName=p_bz-traffic:TrafficInformation&maxFeatures=5000&outputFormat=application/json&srsName=EPSG:4326

The structure of the response is however pretty identical. Each “event” is characterized by a certain geometry (a point) and a certain number of attributes in the form of texts and ID which characterize the type of event. The fields

“BEGIN_DATE” and “END_DATE” allow to distinguish if the event is planned (current timestamp precedent to BEGIN_DATE), currently on-going (current timestamp between the two dates) or complete (current timestamp posterior to END_DATE).

5. Integration specification in the Open Data Hub

The model of the Open Data Hub (Mobility) is quite simple, and is mainly made up of three levels:

- **station**: refers to a concept of "station", i.e. a fixed/mobile installation that collects and transmits data (e.g. measuring station);
- **type**: refers to a concept of measured parameter, which can be associated with a particular sensor installed in a measuring station (e.g. temperature sensor in a meteorological station);
- **measurement**: is the measurement value which is associated univocally to a type and a station.

All following indications are preliminary inputs for better defining the mapping work that needs to be implemented with the requested components. All detailed design choices are going to be made during the implementation work.

“On-demand service” dataset

In order to avoid modifying the fundamental core structure of Open Data Hub, the approach in this case is to consider all elements of the service (stops, polygon, route, vehicle) as a particular “station” which does not have types and measurements, but just characterized by rich metadata, which can also include a complex geometry like in the case of the polygon or complex data structures such as the route with the list of stops to be served.

“Bike counter” dataset

In this case the integration is quite simple. Each bike counter has to be modeled as a station which provides data related to one or more “count” types (e.g. in case the classification of specific vehicle categories is provided). It is currently unclear if a detail per direction is available, in case this is possible a “station” has to be foreseen for each couple (counter, direction) – which means two “stations” have to be foreseen for each bike counter.

“Bike chargers” dataset

The integration proposal is essentially the same as that already implemented for the bike sharing service in the city of Bolzano, which characterize both the bike sharing station and the individual bays as "stations", providing an appropriate parental relationship between the two types of stations (1 bike sharing station associated to N “bays” stations). This link is evident when looking at the structure of the data exposed by the Open Data Hub.

https://mobility.api.opendatahub.bz.it/v2/flat,node/BikesharingStation/*/latest

https://mobility.api.opendatahub.bz.it/v2/flat,node/Bicyclestationbay/*/latest

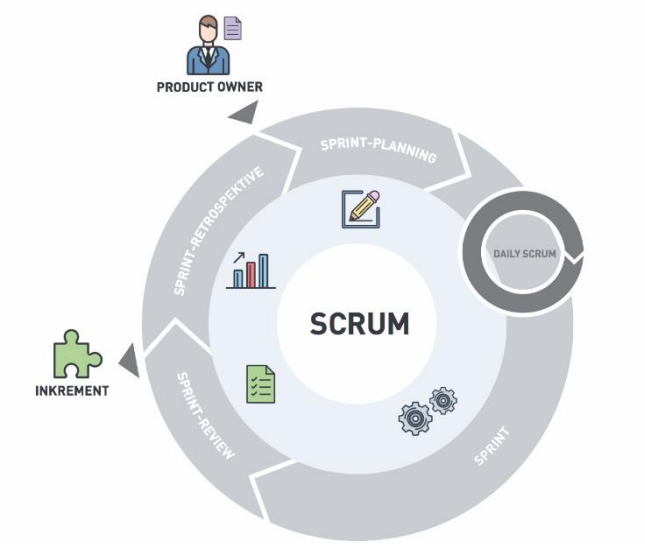
“Traffic events” dataset

Similarly, as for the “on-demand service” dataset, the approach is to consider also in this case an “event” as a particular “station” which does not have types and measurements. A “station” in the Open Data Hub is already charac-

terized by a field called “active” which indicates if it is currently live, which can be used by third parties to distinguish if an event is currently on-going or not. The Open Data Hub is already in condition to properly manage and store all variations of the metadata associated to a station, which can be useful to understand how an event has evolved during the time.

6. Modalities and guidelines for the development

The development of the activities covered by this market survey will follow the agile method (scrum). Two weeks sprint sessions are scheduled, unless otherwise agreed during the kick-off meeting with the core team of NOI S.p.A.



The software development will take place in three phases/environments:

- **development environment**: this environment is on supplier’s infrastructure and is used during the development of the software components;
- **testing environment**: on infrastructure made available from NOI Techpark. This environment is used in order to test the new working versions of the software components. For the publication of the new versions a Continuous Integration (Jenkins) pipeline will be developed by the NOI team. For this reason, the new versions of the code will have to be “committed” to a dedicated Git Repository according to the instructions provided by the team of the NOI Techpark;
- **production environment**: on infrastructure made available from NOI Techpark. After the testing phase, as soon as the software produced is considered sufficiently stable, the software will be integrated in the production environment. Also, this process is managed automatically with Continuous Integration pipelines.

To coordinate the project NOI S.p.A. will use a Kanban Board in Github. Each functionality or issue will be described by NOI S.p.A. in Github and put on the Kanban Board. The Kanban Board will have the following columns:

- **Backlog**: contains all issues that are on hold and have to be discussed during the next sprint meeting with the supplier;

- **ToDo**: contains all issues that have to be concluded in the actual sprint;
- **In Progress**: contains all issues where the is working on;
- **To Review**: contains all issues where NOI Techpark has to make some reviews and that has to be reviewed during the sprint meeting.

All issues in the Kanban, but the one in Backlog, have to be assigned to the user that has to make the next step (e.g. the issues in ToDo will be assigned to the developer who has to develop the functionality, the issue in ToReview will be assigned to the tester, etc.). The supplier will have access to the project Kanban board and will have to check it regularly.

In order to allow the NOI S.p.A. team to properly review and test the code, for each issue in the ToDo lane the service provider has to send a pull request to the development Branch of the repository at least 5 working days before the sprint meeting.

In order to allow a better integration with the systems already in use by NOI Techpak it is required to implement all software components, where possible, using the technologies that are already in use by the Open Data Hub project. These technologies are described in the technical documentation, available at the following link:

https://docs.opendatahub.bz.it/

The source code has to be uploaded to the Git repositories provided by NOI Techpark. During the upload the service provider has to take particular attention to the following aspects:

- do not commit usernames or passwords. NOI Techpark uses Jenkins technology to build the code which implements password ingestion based on special keywords in the source code;
- well document the code describing at least:
    - the general architecture of the system;
    - the list of the licences of all the libraries used;
    - the installation process;
    - all other useful information for people who want to fork or install and use the project.

As Open Data Hub we created some boilerplate repositories for the most common project type (es. Java project, Web Component, .Net Core project, etc.). In case you are starting a new project from scratch, before starting your project please look for the boilerplate that best fits your project and use it to initialize your repository

While you are documenting your code please consider that the official language of the Open Data Hub is English. So, the entire documentation, including the comments in the code, has to be in English. Moreover, you have to observe the following guidelines:

- use the right boilerplate of the README.md if exists;
- use only markdown or text (no binaries, no PDF, etc.);
- should be so detailed that a third person, without any connection to the developers can setup the project, run it and develop it further;
- Java Doc and similar tools for other languages should be as complete as possible;
- add the author tags incl. emails;
- README.md should be a good description of the project and should also have a usage instruction (boilerplate does not consider that). Mainly because tools like **npm** use it as homepage for each project

In general, the documentation of the project (e.g. readme file, license file, etc.) should be done in order to allow third parties developers, who don't know anything about the project, to understand the whole project and also replicate, install or modify it without the need to get in contact with NOI S.p.A.

Therefore, the documentation (README.md) should include also:

- a short description that allows the user to understand the overall goal and functionalities of the project;
- Longer and detailed description that includes also:
  - description of the different parts of the repository/application;
  - description of different parts of the project (also other repositories, if existing, and a link to them) and how this application is part of the overall project;
  - external services/code/framework/software that are used including their licence and copyright information;
- detailed development setup instructions (including testing);
- detailed deployment setup instructions

In respect to the licensing and copyright information, the service provider has to follow the guidelines defined by the Reuse project:

https://reuse.software/

The service provider must provide code where the Reuse linter passes without errors and the licenses must be all compatible with each other.

As mentioned above the service provider, before each sprint meeting, will deliver the source code by making a Pull Request to the Development Branch of the repository Git provided by NOI S.p.A. at the beginning of the project. In general, the service provider has to observe the following guidelines to make the pull requests:

- at the beginning of each sprint the service provider will open a Pull Request (PR) with a prefix [WIP];
- during the sprint the service provider has to regularly push the commits to that PR in order to allow NOI S.p.A. to monitor the status of the project (additional information are available under https://opendatahub.readthedocs.io/en/latest/contributors.html);
- at the end of the sprint (at least 5 days before the sprint meeting) the service provider will close and send the Pull Request.

NOI S.p.A. will analyze the Pull Request before the meeting and eventually send feedback to the service provider. The minimal requirements for a Pull Request to get accepted are:

- the documentation must exist and be as complete as possible in respect to the status of the project
- commits must not contain credentials or any other sensible data
- contributions (e.g. documentation, comments, etc.) must be in english
- merge conflicts must be resolved by the contributor
- all Continuous Integration verifications must pass
- Pull request branches should possibly have a linear history, that is, they should not contain merge commits

During the development cycles the pull request comments and in general the issues and the dedicated Kanban board on Github (original repository) must be tracked by the service provider. The discussion about issues, pull requests,

and other specific comments on the code development will be managed on Git in the project repository and NOT through email. That also involves moving user stories to the corresponding column in the Kanban and assigning them to the right user.

These paragraphs contain some guidelines that the service provider should follow while implementing the project:

- commits should contain a single thing/feature, not be too big and specially they should not be a combination of unrelated features or bug-fixes;
- each commit must be described: present tense and active (e.g. "Add logging to commons" not "commons will get logging now" and not "Added logging").

For the deployment of the project NOI S.p.A. will use its CI/CD infrastructure, for this reason it is important that the service provider includes in the documentation of the project the information about how the application should be deployed or updated by a CD pipeline. Therefore, the documentation should point out the following things:

- What parameters must be configured? Which ones are secrets and which are not?
- What services must be used? (e.g.. PostgreSQL database, S3, ..)
- What steps must be made to package the application/project so that it can be copied to the server?
- What steps must be made on the server after deploying? (ex. database migrations executing with special command)
- What must be adjusted on the server only once? (ex. cron job, shared folder)

All projects should include unit tests and the minimal requirements for the service provider are:

- setup a test infrastructure;
- write unit tests to cover the most important features;
- the minimal test coverage should be 20%;
- tests should mainly cover own business logic (even if minimal) and not third party API's / libraries

Finally, a test driven development is appreciated.

In case that within the project it is foreseen also the development or the change of APIs, the service provider should observe the following guidelines:

- all API calls must be documented in the README.md;
- Swagger UI should be used;
- in case of errors the API should return to the consumer valid and descriptive error messages;
- the API should be RESTful, if possible, but, in case of need, other formats will be considered. In case of non RESTful APIs the service provider should present to NOI S.p.A. enough documentation to allow NOI S.p.A. to decide whether to go on with the new technology or stick to RESTful;
- the API must include also:
    - Response codes,
    - HTTP methods,
    - validity errors,
    - logging: JSON format for production and plain-text for local development written to stdout

In case that the project foresees Access Control List management, the service provider should observe the following guidelines:

- every login to a webapp needs ACL;
- the passwords must be complex enough to be secure;
- Oauth 2.0 standard is required
- Session management for webapps should be present, logout after an inactivity time (the length of the inactivity time depends on the single projects and has to be agreed with NOI S.p.A.)

As an Access Management tool NOI S.p.A. uses Keycloak (https://www.keycloak.org/) instance. More details are available at the following links:

https://docs.opendatahub.bz.it/en/latest/guidelines/authentication.html

https://docs.opendatahub.bz.it/en/latest/guidelines/authentication.html#authentication-to-internal-infrastructure

NOI S.p.A. is using Docker (https://www.docker.com/) to automate the deployment of the application and we strongly recommend to:

- use docker for local development;
- keep local docker setup, staging and production as similar as possible (these will be provided and updated by the NOI S.p.A. team).
- use environmental variables to configure different stages (i.e., .env files)

7. Timing of implementation

Following indicative milestones are foreseen for the development work requested:

- **Milestone 1**: completion of the implementation of “On-Demand” Data Collector (end of July)
- **Milestone 2**: completion of the implementation of “Traffic Events” Data Collector (end of August)
- **Milestone 3**: completion of the implementation of “Bike Counters” Data Collector (end of September)
- **Milestone 4**: completion of the implementation of “Bike Chargers” Data Collector (end of October)

These milestones are however indicative, and primarily aim to highlight the priority given to the different Data Collectors. Proposals for improvements of such reference plan are obviously acceptable and shall be presented according to the indications provided in the following chapter. Alternative proposals that worsen this proposal can be also accepted, but the development work must be completed at the end of November at latest.

8. Contents and evaluation of proposals / quotes

The quotes will be evaluated according to the criteria summarized in the table below.

Criteria	Points
1. Technical evaluation	**70**
1.1 Technical proposal for the implementation of the Data Collectors	20
1.2 Temporal planning and ability to complete the required developments in the shortest possible time	30
1.3 References and overall experience in the requested software developments	20
2. Economical quote	**30**

As far as **criterion 1.1** is concerned, it is expected to receive some technical indications about how Data Collectors are going to be implemented (e.g. with which programming language, how is the source code going to be structured, e.g.).

As far as **criterion 1.2** is concerned, it is expected to receive some concrete indications about when and how (e.g. with which personnel resources) each of the milestones indicated in chapter 7 are going to be fulfilled. The temporal plan should take in consideration the bi-weekly time frame in which developments' sprints are going to be organized.

As far as **criterion 1.3** is concerned, it is expected to receive a CV of the personnel staff to be involved in this activity and up to 3 project references about similar development activities in which this staff (or alternatively the company) was involved. Such project references should have taken place from 1.1.2018 onwoards.

As far as the **economical quote** is concerned, it is expected to receive a cost indication for the implementation of each Data Collector. The points are going to be assigned according to the following formulas:

$$C_i = \frac{O_{min}}{O_i}$$

$$PE_i = C_i * P_{max}$$

where:

- O_i is the economical quote of the *i*-th proposal.
- O_{min} is the economical quote of the best quote (i.e. with the lowest price)
- C_i is the coefficient associated to the *i*-th proposal
- P_{max} is the maximum number of points related to the economical quote (i.e. 30 points)
- PE_i is the number of points associated to the *i*-th proposal.

The technical evaluation and economical quote shall be provided in a document that should not exceed 15 pages of documentation. Any collaboration with other companies and the presence of any subcontracts must be explicitly indicated.

NOI S.p.A reserves the right to activate a cooperation with several economic operators if it considers it functional and efficient from a technical and economic point of view. Therefore, it is possible to present a quote also for just a subset of the datasets described in this document. In this case, the comparison of the quotes will be limited to the solely Data Collectors that are included in the proposal.

9. Invoicing procedures

The invoicing of the activities concluded by the supplier will be sent to NOI S.p.A via electronic invoice only after the outputs produced have been successfully tested by NOI S.p.A. Before to proceed with the testing of the outputs, the supplier must provide to NOI S.p.A.:

- the entire documentation;
- if code development is planned, the code must be uploaded to the Git repository provided by NOI S.p.A;

- in the case of multimedia contents (e.g. photos, videos, illustrations, documents), the service provider has to upload it on specific platforms (e.g. Vimeo, Flickr, etc.) and provide the source files or open versions through appropriate file hosting services indicated by NOI S.p.A.

All invoices must include that the transaction is subject to the Split Payment discipline as mentioned in the art.17-ter del DPR 633/197 and must be issued exclusively in electronic format (Unique Office code: T04ZHR3).

10. Transfer of rights

Where the creation of material subject to proprietary rights, including copyrights, sui generis data rights, and related rights, including solely of photographs, industrial design, all rights of economic exploitation arising from achieved results are reserved to NOI S.p.A., excepting those expressly excluded when the order is placed.

Further, if the material includes a software development project, all source code from libraries or other modules used in the realisation of an assignment and belonging to a third party must be released under an Open Source license (opensource.org/ licenses) in a manner compatible with the scope of the "outbound" software license, without requirement for adaptation, addition, cancellation or requests for permission from third parties on the part of NOI S.p.A. In the absence of any expressly indicated license, the terms of the GPL v3 or AGPL v3 (depending on the project type) license shall apply. The use of material belonging to third parties must be expressly declared at the time of the quote, or be easily and immediately understandable from the description of the project. In the event that code is developed during the realisation of this assignment, NOI S.p.A. will initiate a Git repository on which the supplier must develop and publish the source code.

If the material consists of data, creative works (drawings, literary works, cinematographic works, figurative art, photographs), industrial design or other material which are subject in whole or in part to the proprietary rights of a third party, the use of such material is permitted provided it is licensed under conditions compatible with the license under which said material will be published, if indicated. If no license is indicated, the material will be subject to conditions compatible with the Creative Commons Zero (CC0) license.