Backend API Version 1.0 Specification

Initial Draft Version 1.0

Author: *jd*
Last reviewed: *January 02, 2023*

Table of Contents

1. Document Objectives and Purpose 1
2. Restrictions 1
3. Assumptions 2
4. Forecast Service API 2
4.1 /api/v1/models/{*model_id*}/forecast 2
4.2 Additional Documentation 3
5. Train Model Service API 3
5.1 /api/v1/models/{*model_type*}/train 3
5.2 Additional Documentation 3
6. Additional Services API 3
6.1 /api/v1/models 3
6.2 /api/v1/models/{*model_id*} 4
6.3 /api/v1/clients 5
6.4 /api/v1/clients 5
6.5 /api/v1/clients/{*client_id*} 6
6.6 /api/v1/clients/{*client_id*} 6
6.7 /api/v1/clients/{*client_id*}/models 6
6.8 /api/v1/clients/{*client_id*}/tasks 7
6.9 /api/v1/tasks/{*task_id*} 8
6.10 /api/v1/tasks/{*task_id*}/state 8
7. Annex A 9
7.1.1 Types of Models Supported (Available) 9
7.1.2 Task States 9
7.1.3 Detail 9
7.1.4 Input Data Structures for Forecasting 9
7.1.5 Input Data Structures for Training 10

1. Document Objectives and Purpose

This document contains the API specification for the web services provided by the *backend* that were developed, i.e., the F*orecast* and the T*rain Model* services.

2. Restrictions

R1 *The backend web services were developed using the Asynchronous Server Gateway Interface (ASGI) standard. Therefore, the backend services must be executed with an ASGI compatible (capable) web server.*

Commented [JD1]: Some restrictions may apply. This is the place where restrictions should be presented and discussed.

3. Assumptions

Commented [JD2]: Some assumptions may apply. This is the place where assumptions should be presented and discussed.

A1 *The communication between the backend and its clients is established using the HTTPS protocol.*
A2 *The backend is behind a reverse proxy that establishes, manages, and handles HTTPS communications with clients. The backend web services do not provide mechanisms to handle HTTPS communications.*
A3 *The backend web services run in specific IP addresses and ports that can be configured.*
A4 *The backend web services expect (and only accept) requests with a Bearer Token Authorization Header and inject the respective Bearer Token in the response.*
A5 *The backend and its clients exchange information using JSON.*
A6 *The backend web services assume that a request to obtain a forecast or to train a model, always correspond to one and only one task. If an execution plan has M >1 forecast or train model tasks, M requests must be performed. The backend does not guarantee the order in which requests are processed and results are returned to the client.*

4. Forecast Service API

The *Forecast Service* allows obtaining forecasts using a previously trained model. The models used are stored on the computer's hard disk.

4.1 /api/v1/models/{*model_id*}/forecast

Allows obtaining a forecast using a previously trained model identified with a model Id.

Type	***POST***
Usage Example	*https://localhost/api/v1/models/4/forecast*

Input

Path Parameters	*model_id*	*integer*	Identifier of the model to be used.
Body Schema	*client_id*	*string*	Identifier of the client making the request.
	model_input_data	*dictionary*	Input the model needs to compute a forecast (*See Section 7.1.4*).
	forecast_period	*integer*	Forecast period in hours.

Notes
- Since two models may need different inputs to compute a forecast, *model_input_data* is a *dictionary*.
- The forecast period can be obtained from the backend database.
- A client *μ* can only use a model trained by client *μ*.

Commented [JD3]: Remove this parameter?

Output

Body Schema	*forecast*	*List[integer]*	A list with the values predicted. The size of the list is equal to the forecast period specified.

Error Messages

In case an error occurs, an *HTTP Exception* will be raised and sent to the client.

status_code	*422*	Raised in response to an invalid request input.
detail	*List[Detail]*	A list (array) of objects of type *Detail* (*See Section 7.1.3*).

Example:

```
"detail": [
    {
        "loc": ["string", 0],
        "msg": "string",
        "type": "string"
    }
]
```

status_code	*404, 400, 500*	Other error codes.
detail	*string*	Error message, e.g., Client *μ* not found!

Notes
- Various *HTTP Exceptions* are used to report errors (if this is not ideal, errors can be reported using a different mechanism/different messages).

4.2 Additional Documentation

Use the following links (local access) to get more information about (or to interact with) the API:

- http://<*ip*>:<*port*>/docs
- http://<*ip*>:<*port*>/redoc
- http://<*ip*>:<*port*>/openapi.json

where *<ip>:<port>* refers to the IP and the port where the service is running (these links need internet access to work).

5. Train Model Service API

The Train Model *Service* allows training different types of machine learning models. The models trained are stored on the computer's hard disk and can be used later to obtain forecasts. Trained models are associated with one and only one client. A trained model can only be used by the client it is associated with.

5.1 /api/v1/models/{*model_type*}/train

Allows training a model of the type specified in the *model_type* path parameter variable if the type specified is available. A message is sent to the client informing that the task was accepted, or an error occurred. The model is trained in a background task. That is, a model may not immediately be available or trained.

Type	***POST***
Usage Example	*https://localhost/api/v1/models/Prophet/train*

Input

Path Parameters	*model_type*	*string*	Type of model to train (*See Section 7.1.1* for a list of the types of models available).
Body Schema	*client_id*	*string*	Identifier of the client making the request.
	model_name	*string*	A name for the model, e.g., *Schedules for Store A*.
	input_data	*dictionary*	Input needed to train the model (*See Section 7.1.5*).
	forecast_period	*integer*	Forecast period in hours that will be associated with the model.

Notes

- The *model_name* is something that makes sense to the client and is not used internally by the service to make decisions about how the model will be stored or identified.
- Since different inputs may be needed to train different models, the input data used for training is expected to be encapsulated in a *dictionary* data type (*See Section 7.1.5* for more details).

Output

Body Schema	*detail*	*string*	If the task is accepted: "detail": "1".
	task_id	*integer*	The task id.

Error Messages

The same as in *Section 4.1*.

5.2 Additional Documentation

See Section 4.2.

6. Additional Services API

6.1 /api/v1/models

List of supported (available) models for training.

Type	***GET***
Usage Example	*https://localhost/api/v1/models*

Input

No input

Output

Body Schema	*models*	*List[dictionary]*	A list containing information about the models available (*See Section 7.1.1*). A *dictionary* has the following structure:

```
{
   "model_type":  string,
   "model_name": string
}
```

Example:

```
"models": [
   {
      "model_type": "MLPRegressor",
      "model_name": "MLPRegressor"
   },
   {
      "model_type": "Prophet",
      "model_name": " Prophet "
   }
]
```

Error Messages

6.2 /api/v1/models/{*model_id*}

Details of a model previously trained.

Type	***GET***
Usage Example	*https://localhost/api/v1/models/1*

Input

Path Parameters	*model_id*	*integer*	The unique model identifier.

Output

Body Schema	*id*	*integer*	Model unique identifier.
	type	*string*	Model type.
	model_name	*string*	Model name defined by the client.
	time_trained	*string*	A string with the *timestamp* when the model was trained.
	metrics	*string*	A *csv* string with metrics collected during the training phase.
	forecast_period	*integer*	Forecast period associated with the model.
	train_params	*string*	A *csv* string with custom train parameters used in the training phase.
	html_report	*string*	Html report created during the training phase.

Example:

```
"id":               21,
"type":             "MLPRegressor",
"model_name":       "Sales Store A",
"time_trained":     "2022-12-25T17:30:18",
"metrics":          "rmse:10.996;mae:4.960;r2_score:0.958;",
"forecast_period":  720,
"train_params":     "max_number_iterations:10;",
"html_report":      "<div>…</div>"
```

Error Messages

Error	**Message**
model not found	"*status_code*": 404, "*detail*": "Model {*model_id*} not found!"

Discussion

- Should this API also expect (consider) the *client_id*?
- The output of this API will most likely change to contain an html report instead.

6.3 /api/v1/clients

List of 'registered' (known) clients.

Type	***GET***
Usage Example	*https://localhost/api/v1/clients*

Input

No input

Output

Body Schema	*clients*	*List[dictionary]*	A list containing information about the known (registered) clients. A *dictionary* has the following structure:

```
{
    "client_pkey": integer,
    "id":          string,
    "culture":     string,
    "is_active":   boolean
}
```

Example:

```
"clients":[
    {
        "client_pkey": 1,
        "id": "AB19GF",
        "culture": "es-ES",
        "is_active": true
    },
    {
        "client_pkey": 2,
        "id": "AF52HY",
        "culture": "en-EN",
        "is_active": true
    }
]
```

Error Messages

6.4 /api/v1/clients

Creates (registers) a new client.

Type	***PUT***
Usage Example	*https://localhost/api/v1/clients*

Input

Body Schema	*id*	*string*	Client unique identifier used when making requests.
	culture	*string*	The client's culture (language).
	is_active	*boolean*	Indicates if the client is active.

Output

Body Schema	"*detail*": "*1*"	The client was created (registered).

Error Messages

Error	**Message**
A client with the same id already exists	"*status_code*": *404*, "*detail*": "Client *{client.id}* already exists!"

6.5 /api/v1/clients/{*client_id*}

Updates the parameters associated with client *client_id*.

Type	***POST***
Usage Example	*https://localhost/api/v1/clients/AR21TY*

Input

Path Parameters	*client_id*	*string*	The client's unique identifier.
Body Schema	*culture*	*string*	The client's culture (language).
	is_active	*boolean*	Indicates if the client is active.

Output

Body Schema	"*detail*"*:* "*1*"	The client parameters where updated.

Error Messages

Error	**Message**
Client not found	"*status_code*"*: 404,* "*detail*"*:* "Client *{client_id}* not found!"

6.6 /api/v1/clients/{*client_id*}

Details of the client *client_id*.

Type	***GET***
Usage Example	*https://localhost/api/v1/clients/AR21TY*

Input

Path Parameters	*client_id*	*string*	The client's unique identifier.

Output

Body Schema	*id*	*string*	The client's unique identifier.
	culture	*string*	The client's culture (language).
	is_active	*boolean*	Indicates if the client is active.
	client_pkey	*integer*	The client's internal unique identifier.

***Example*:**

```
"id": "ZO52WE",
"culture": "us-US",
"is_active": true,
"client_pkey": 8
```

Error Messages

Error	**Message**
Client not found	"*status_code*"*: 404,* "*detail*"*:* "Client *{client_id}* not found!"

6.7 /api/v1/clients/{*client_id*}/models

List of models associated with the client *client_id*.

Type	***GET***
Usage Example	*https://localhost/api/v1/clients/AR21TY/models*

Input

Path Parameters	*client_id*	*string*	The client's unique identifier.

Output

Body Schema	*models*	*List[dictionary]*	A list containing information about the *models* associated with a client. A *dictionary* has the following structure:

```
{
    "id":           integer
    "type":         string
    "model_name":        string
    "time_trained":      string
    "metrics":           string
    "forecast_period":   integer
    "train_params":      string
    "html_report":       string
}
```

***Example*:**

```
"models": [
   {
      "id": 7,
      "type": "MLPRegressor",
      "model_name": "Sales Store A",
      "time_trained": "2022-12-21T15:18:22",
      "metrics": "rmse:9.75;mae:7.36;r2_score:0.96;",
      "forecast_period": 720,
      "train_params": "max_number_iterations:10;",
      "html_report": "<div>…</div>"
   },
   {
      "id": 20,
      "type": "HistGradientBoostingRegressor",
      "model_name": "Schedules Store B",
      "time_trained": "2022-12-25T00:58:28",
      "metrics": "rmse:11.05;mae:5.00;r2_score:0.95;",
      "forecast_period": 720,
      "train_params": "",
      "html_report": "<div>…</div>"
   }
]
```

Error Messages

Error	**Message**
Client not found	"*status_code*"*: 404,* "*detail*"*:* "Client *{client_id}* not found!"

6.8 /api/v1/clients/{*client_id*}/tasks

List of training tasks requested by client *client_id*.

Type	***GET***
Usage Example	*https://localhost/api/v1/clients/AR21TY/tasks*

Input

Path Parameters	*client_id*	*string*	The client's unique identifier.

Output

Body Schema	*tasks*	*List[dictionary]*	A list containing information about the training *tasks* requested by the client. A *dictionary* has the following structure:

```
{
    "id":              integer
    "time_created":    string
    "time_started":    string
    "time_finished":   string
    "model_type":      string
    "state":           string
}
```

***Example*:**

```
"tasks":[
   {
      "id": 7,
      "time_created": "2022-12-21T15:17:59",
      "time_started": "2022-12-21T15:17:59",
      "time_finished": "2022-12-21T15:18:22",
      "model_type": "MLPRegressor",
      "state": "Finished"
   },
   {
      "id": 12,
      "time_created": "2022-12-23T16:39:26",
      "time_started": "2022-12-23T16:39:26",
      "time_finished": null,
      "model_type": "MLPRegressor",
      "state": "Error"
   }
]
```

Error Messages

Error	**Message**
Client not found	"*status_code*": *404*, "*detail*": "Client *{client_id}* not found!"

6.9 /api/v1/tasks/{*task_id*}

Details of training task *task_id*.

Type	***GET***
Usage Example	*https://localhost/api/v1/tasks/7*

Input

Path Parameters	*task_id*	*integer*	The task unique identifier.

Output

Body Schema	*id*	*integer*	The task unique identifier.
	time_created	*string*	Timestamp indicating when the task was created.
	time_started	*string*	Timestamp indicating when the task started.
	time_finished	*string*	Timestamp indicating when the task finished.
	model_type	*string*	The model type trained in this task.
	state	*string*	The task state.

Example:

```
"id":            7,
"time_created":  "2022-12-21T15:17:59",
"time_started":  "2022-12-21T15:17:59",
"time_finished": "2022-12-21T15:18:22",
"model_type":    "MLPRegressor",
"state":         "Finished"
```

Error Messages

Error	**Message**
Task not found	"*status_code*": *404*, "*detail*": "Task *{task_id}* not found!"

Discussion

- Should this API also expect (consider) the *client_id*?

6.10 /api/v1/tasks/{*task_id*}/state

State of training task *task_id*. Tasks can have several states. See Section 7.1.2 for a list and description of the states that a task can be on (assume).

Type	***GET***
Usage Example	*https://localhost/api/v1/tasks/7/state*

Input

Path Parameters	*task_id*	*integer*	The task unique identifier.

Output

Body Schema	*state*	*string*	The task state.
	html_report	*string*	The corresponding html report. The report is only sent if the state of the task is *Finished*.

***Example*:**

"*state*":	"Finished"
"*html_report*":	"<div>…</div>"

Error Messages

Error	**Message**
Task not found	"*status_code*": *404*, "*detail*": "Task *{task_id}* not found!"

Discussion

- Should this API also expect (consider) the *client_id*?

7. Annex A

7.1.1 Types of Models Supported (Available)

Model	**Type (model_type)**	**Toolkit (Python)**
MLPRegressor	*MLPRegressor*	sklearn
HistGradientBoostingRegressor	*HistGradientBoostingRegressor*	sklearn
Prophet	*Prophet*	prophet

7.1.2 Task States

State	**Value (used internally)**
Created	*Created*
Pending	*Pending*
Executing	*Executing*
Finished	*Finished*
Error	*Error*

7.1.3 Detail

Detail data structure.

loc	*List[integer|string]*	Location of the error *List[integer] or List[string]*.
msg	*string*	Message associated with the error.
type	*string*	Type of error.

7.1.4 Input Data Structures for Forecasting

Model	**Input Data Structure**

MLPRegressor	*model_input_data = List[integer]* List of known values (previous observations). This list must have a size equal to the *forecast_period* specified in the forecast request.
HistGradientBoostingRegressor	A *dictionary* of the form: *model_input_data = {* "*Month*"*: List[integer],* "*Day*"*: List[integer],* "*Hour*"*: List[integer]* *}* In the current implementation, the *List[integer] objects* in the *dictionary*, must have a size equal to the *forecast_period* specified in the forecast request.
Prophet	A *dictionary* with the form: *model_input_data = {* "*ds*"*: List[string]* *}* where *ds* is a list of *strings* representing timestamps with the format YYYY-MM-DD HH:MM:SS, e.g., 2019-04-13 09:15:00.

Commented [JD4]: Needs confirmation.

Commented [JD5]: Can be optimized in the future.

Commented [JD6R5]: Also needs confirmation.

Important Note:

- In the case of the *HistGradientBoostingRegressor* model, the labels used in the *dictionary*, i.e., *Month*, *Day*, and *Hour*, must be presented exactly in the same way as the labels used during the training phase, or an error will be raised by the model.

7.1.5 Input Data Structures for Training

Model	**Input Data Structure**
MLPRegressor *HistGradientBoostingRegressor* *Prophet*	*input_data = {* "*ds*"*: List[string],* "*y*"*: List[integer]* *}* where *ds* is a list of *strings* representing timestamps with the format YYYY-MM-DD HH:MM:SS, e.g., 2019-04-13 09:15:00, and *y* are the observations corresponding to the timestamps. An implicit *1-1* correspondence between the elements of the two lists, *ds* and *y*, is expected, i.e., the two lists must have the same size. In general, the size of the input data for training should be significantly bigger than the *forecast_period* specified. These details, however, will not be discussed in this document.

Commented [JD7]: Discuss in more detail in another document?