# **Bike3S configuration files:**

## Users\_configuration.json

Structure:

{

"initialUsers": [ //list of users

{ //NECESSARY PARAMETERS:

"position": { // position of appearance of user

"latitude": 40.4302937,

"longitude": -3.7069171

},

"destinationPlace": { // desired destination of user

"latitude": 40.4250361,

"longitude": -3.6837876

},

"timeInstant": 3, //instant from simulation start where user

//appears (in seconds)

"userType": { //the type of the user (java class with parameters)

"typeName": "USER\_UNINFORMED",

"parameters": {

"maxDistanceToRentBike" : 600

}

},

//OPTIONAL PARAMETERS:

"intermediatePosition": null, // intermediate destination of user

"cyclingVelocity": 4.8, // in meters per second

“walkingVelocity”: 1 // in meters per second

},

…

]

}

Any additional parameters in the json file are ignored!

## stations\_configuration.json

{

"stations": [ //list of stations

{ //necessary parameters:

"position": { // position of the station

"latitude": 40.4168961,

"longitude": -3.7024255

},

"bikes": 11, //number of bikes at start of simulation

"capacity": 24, //capacity of the station

"id": 1, //identifier

//not used parameters:

"name": "Puerta del Sol A",

"number": "1a"

},

…

]

}

Any additional parameters in the json file are ignored!

## global\_configuration.json

Fields:

"boundingBox": the area os service

"debugMode": false/true to generate or not debug information

"reservationTime": 1000, seconds a reservation is kept for users that do a reservation

"randomSeed": 10, for random generation of data

"totalSimulationTime": 86400, simulation time in seconds

"startDateTime": "2018-07-20T07:00:00", Date and time when the simulation starts

startDateTime is necessary to take the correct demand data if demand is used in recommendations

"graphManagerType": the manager for directions and routes that is used. Only one manager is implemented :

"graphManagerType": {

"typeName": "GRAPH\_HOPPER",

"parameters": {

"mapFile": absolut path to osm file (e.g. madrid.osm)",

"tempDirectory": "absolut path of temp directory (e.g. \.Bike3s"

}

},

"recommendationSystemType": the recommender system used for recommending statios to users. Corresponds to a java class

if the typeName for recommendationSystemType is none, no recommender is loaded

Example:

"recommendationSystemType": {

"typeName": "AVAILABLE\_RESOURCES",

"parameters": {

"maxDistanceRecommendation": 600

}

},

"demandManagerType": demand manager used if the recommendation system needs demand data

if the typeName for demandManagerType is none, no demand manager is loaded

Currently only one manager is implemented

Example:

"demandManagerType": {

"typeName": "FileBasedDemandManager",

"parameters": {

"demandDataFile": absolut path of the cvs file eg. ./demandDataMadrid0817\_0918.csv"

}

},

" fleetManagerType ": fleet manager used

if the typeName for fleetManagerType is none, no fleet manager is loaded

Currently only one manager is implemented

"fleetManagerType": {

"typeName": "FileBasedFleetManager",

"parameters": {

"EventFile": file with manager events e.g. ./managerevents.json"

}

}

## managerevents.json

the file with predefined events a fleetmanager can execute (e.g. moving a bike from one station to another)

{

"managingEvents": [

{

"timeInstant": 1800, //start time of the movement (seconds from

// simulation start)

"idEndStation": 78, //station where bike is put

"idInitialStation": 162, //station where bike is taken

"travelTime": 3195 //time of the movement (seconds)

},

…

]

}

## demandDataMadrid0817\_0918.csv

file with historic data in order to estimate demands

Station;dia;mes;año;hora;NumUsuariosCojenBiciEnEstacion;NumUsuariosReturnBiciAEstacion;Dia semana (lun=1)

1;1;8;2017;0;0;0;mar

2;1;8;2017;0;0;0;mar

3;1;8;2017;0;0;0;mar

4;1;8;2017;0;4;0;mar

5;1;8;2017;0;0;4;mar

6;1;8;2017;0;0;0;mar

## tests.json

File for executing different tests with the same user and station data, but different recommenders and usertypes

If tests are executed, for each test, all userstypes in the user configuration file are substituted with the usertype of the test.

Also the recommendationSystemType in the global configuration is substituted with the recommendationSystemType in the corresponding test.

Example:

{ "tests": [

{ "recommendationSystemType": {

"typeName": "AVAILABLE\_RESOURCES",

"parameters": {

"maxDistanceRecommendation": 600

}

},

"userType": {

"typeName": "USER\_UNINFORMED",

"parameters": {

"maxDistanceToRentBike" : 600

}

}

},

{ "recommendationSystemType": {

"typeName": "DEMAND\_cost\_prediction",

"parameters": {

"maxDistanceRecommendation": 600,

"minimumMarginProbability": 0.001,

"minProbBestNeighbourRecommendation": 0.5,

"desireableProbability": 0.8,

"penalisationfactorrent": 1,

"penalisationfactorreturn": 1,

"maxStationsToReccomend": 30,

"unsucesscostRent": 1000,

"unsucesscostReturn": 1000,

"squaredTimes": FALSE,

"PredictionNorm": 1,

"MaxCostValue":5000,

"additionalResourcesDesiredInProbability": 0,

"predictionWindow": 900

}

},

"userType": {

"typeName": "USER\_OBEDIENT",

"parameters": {

"maxDistanceToRentBike" : 600

}

}

}

]

}

# **Execution:**

Basically there are three execution types: simulation, visualization and configuration

## Configuration:

Frontend in node.js. Not very reliable at the moment. Executed from the front end as described in the old documentation.

## Visualization:

Frontend in node.js. Should work like described in the old documentation. But not 100% sure. It uses the history files generated by a simulation.

## Simulation:

In Java. The best is to execute from an IDE (e.g., netbeans).

1. ApplicationWithoutParameters for a single simulation

Change the parameters in the main method to adapt to your directories.

This will create three directories:

* History: with the simulation history in json files
* Analysis: files with analytical information of the simulation
* Debug: debug information (if in debug mode)

1. CompareTestApplication for comparing different recommendation systems

This will execute different experiments (with a test.json file) and compare the results.

Generates history, debug and analysis folders for each test and a final comparison file.