# **TP: London 1888**

**Introduction**

You will need these tools to complete this TP:

- Node.js and NPM

- Visual Studio Code

- Postman to call your API

- Docker or MySQL version 5.7 installed

**Presentation**

This session will be dedicated to your TP. Coefficient for this one will be very high, so complete it with a lot of attention! **Due date is Saturday, November 28**. We accept groups up to **4** people, no solo project.

In this TP, you will have to:

* Write an API with 4 actions
* Manage a MySQL connection
* Write tests

An advice: you have two weeks for this TP, but we will start talking about React next week and you will have another TP to work on, so don’t take your time too much.

**Provided files**

In addition to this subject, we will provide you:

* a basic environment for your project
* a sql file to help you create the table if you do not have access to Docker.

The documentation for your container is [here](https://hub.docker.com/r/flotorel/london1888).

**Database specifications**

Url: 0.0.0.0

Username: root

Password: root

Port: 3306 (MySQL default port number)

Database name: Db\_London1888

Table name: LondonCitizen

Content:

* id: int (generated primary key)
* name: varchar(100)
* posX: int
* posY: int
* isVictim: boolean (1 or 0)

You should not modify the structure of the database. You can access it using *mysql2* package or *TypeORM* (recommended)

**Project**

London, 1888. England is the light of Europe, an example to follow. And social inequality at its finest…

The district of Whitechapel is the home for the poor, the face of London that is not supposed to be seen. But someone will show the world what kind of misery lives in there, in a very brutal way.

Jack the Ripper.

The very first famous *serial killer* in history. He will never be arrested. He will not even be identified…

For this project, you will try to help Scotland Yard to identify Jack the Ripper once and for all *(spoiler: we won’t be able to…)*

1. **API & Database**

You must implement these 4 actions inside your API:

1. Declare a normal citizen

Url: <http://localhost:8081/citizen/:name/:posX/:posY>

Method: Post

Parameters:

* Name: the name of the citizen (string)
* posX and posY: coordinates of the citizen (int, positive or negative)

What it does: declares a new Citizen in the database who is not a victim. You do not need to check if he already exists or if the name is empty

Response: a 200 response with the inserted citizen as an object (with the right id)

1. Declare a victim

Url: <http://localhost:8081/victim/:name/:posX/:posY>

Method: Post

Parameters:

* Name: the name of the citizen (string)
* posX and posY: coordinates of the victim (int, positive or negative)

What it does: declares a new Citizen in the database who is a victim. You do not need to check if he already exists or if the name is empty, but you need to check if a victim has already been declared: there can be only one at a time!

Response: a 200 response with the inserted victim as an object (with the right id) if it is fine, or a 409 response (conflict) if there is already a victim in the database.

1. Find the killer!

Url: [http://localhost:8081/getJack](http://localhost:8081/getJack/:name/:posX/:posY)

Method: Get

Parameters: None

What it does: Returns the closest citizen from the victim’s location. This formula returns the distance between points A and B: . You can get the coordinates from your citizen and your victim: they have been specified as *posX* and *posY* parameters previously. If there is no victim, no normal citizen or if the closest distance between the victim and the citizens is equal for at least two citizens, you must return an error.

Response: a 200 response with the closest citizen as an object if possible; a 404 response (not found) if there is no victim or citizen; a 409 response (conflict) if there are at least two citizens at the same distance from the victim (if they are the closest)

1. Get rid of evidences

Url: <http://localhost:8081/evidences>

Method: Delete

Parameters: None

What it does: Resets the table LondonCitizen. Everything in it is erased. You can truncate it or delete its content, it doesn’t matter.

Response: a 204 response with nothing.

1. **Tests**

The “Jack the Ripper” case may be a failure, but you will still need to test your application anyway! I know, life is unfair…

You must test all the specified cases at least once, without updating the database at all.

**What you should NOT do in ANY case!**

**Any of these will give you an unconditional 0:**

- using any library which is not installed by default

- impossibility for us to install your project and run your tests using “npm install”, “npm start” and “npm test”

- sending us the node\_modules folder

- sending us a project containing red tests

- sending us a project without new tests

**This time, we will show no mercy!**

**How to check your project**

Check your API: use Postman, do not hesitate to try weird cases, because we will.

Check your tests: to check if your tests do not modify your database, you can just stop your Docker container (or your local MySQL database). The tests should be able to run anyway.

Check what you send to us: take the zip file you want to send to us, extract its content in a completely different, empty, folder and run commands *npm i*, *npm start* and *npm test*. Everything should work as expected.

If everything is fine, you will not have 0 ☺

Remember: it’s better to send a partially finished project that a completed one which does not compile…

**What you should send to us**

You just need to send your project content using WeTransfer. Be sure to check everything!

# **Good luck! Next session, it’s time to talk about React! ☺**