# Documentation API Scala

Description des tables

#### User

Id 🡪 Identifiant de l’utilisateur (**auto-incrémenté**)

name 🡪 Nom de l’utilisateur, **varchar(80)**

firstname 🡪 Prénom de l’utilisateur, **varchar(80)**

birthdate 🡪 Date de naissance, **format : AAAA-MM-JJ**

email 🡪 Email de l’utilisateur, **varchar(80)**

password 🡪 MDP de l’utilisateur, **varchar(80)**

#### Destination

Id 🡪 Identifiant de la destination (**auto-incrémenté**)

name 🡪 Nom de la destination, **varchar(80)**

description 🡪 Description textuelle de la destination, **varchar(200)**

audio 🡪 Lien de l’audio-guide, **varchar(200)**

coordX 🡪 Cordonnées en X de la destination, **float(10)**

coordY 🡪 Cordonnées en Y de la destination, **float(10)**

Picture **🡪** Url de l’image, **varchar(6000)**

Url 🡪 Url de la caméra liée au monument, **varchar(6000)**

#### Travel

Id\_destination 🡪 Identifiant de la destination, **smallInt(9)**

Id\_user 🡪 Identifiant de l’utilisateur, **smallInt(9)**

hasVisited 🡪Booléen indiquant si l’utilisateur a visité la destination

#### WeatherInformation

beginning 🡪 Date et heure du début de l’intervalle de 15 minutes, **LocalDateTime**

ending 🡪 Date et heure de la fin de l’intervalle de 15 minutes, **LocalDateTime**

temperature 🡪 Température en degrés, **float(5) NULLABLE**

humidity 🡪 Pourcentage d’humidité, **float(5) NULLABLE**

luminosity 🡪 Pourcentage de luminosité, **float(5) NULLABLE**

sound 🡪 Niveau sonore, **float(5) NULLABLE**

id\_destination 🡪 Identifiant de la destination, **smallInt(9)**

Description des contrôleurs

## UserController

### getUsers()

URL 🡪 /user/getUsers

IN 🡪 /

OUT 🡪 La liste des utilisateurs de la BD

### getUser()

METHODE 🡪 POST

URL 🡪 /user/getUser

IN 🡪 Body de la requête : Email & mot de passe

OUT 🡪 Les informations de l’utilisateur identifié par son adresse email unique



### deleteUser(id : Int)

URL 🡪 /user/deleteUser/{ID du user}

IN 🡪 Identifiant de l’utilisateur

OUT 🡪 /

### updateUserPassword()

URL 🡪 /user/updateUserPassword

IN 🡪 Body de la requête : Email & mot de passe

OUT 🡪 /



### saveUser()

URL 🡪 /user/save

IN 🡪 Body de la requête : Toutes les informations de l’utilisateur (sauf l’ID qui est auto-incrémenté)

OUT 🡪 /



## DestinationController

### getDestinations()

URL 🡪 /destination/getDestinations

IN 🡪 /

OUT 🡪 La liste des destinations de la BD

### setSelectedDestination() ATTENTION c’est http PUT

URL 🡪 /destination/setSelectedDestination?idDestination={destination sélectionnée}&idStop={id de l’arrêt TEC}

IN 🡪 idStop : destination sélectionnée  
 idDestination : id de l’arrêt TEC (sur lequel la tablette se trouve donc)

OUT 🡪 http 200 ou 404

### getSelectedDestination()

URL 🡪 /destination/getSelectedDestination?chipNr={id de la puce qui a été scannée sur l’arrêt de bus}

IN 🡪 chipNr : id de la puce qui a été scannée sur l’arrêt de bus

OUT 🡪 la destination actuellement sélectionnée ou http 204 No content si rien n’est sélectionné si la donnée est dépassée.

### getDestination(id : Int)

URL 🡪 /destination/getDestination/{ID de la destination}

IN 🡪 Identifiant de la destination

OUT 🡪 Les informations de la destination

### getDestinationAround

URL 🡪 /destination/getDestinationsAround?coordX= ???&coordY=??? (et facultativement : &maxDistanceInKm= ??? par défaut c’est 0.5)

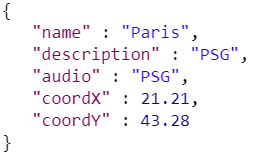
OUT 🡪 L’ensemble des destinations se trouvant dans un certain cercle de maxDistanceInKm de rayon aux alentours des coordonnées passée en arguments

### saveDestination()

URL 🡪 /destination/save

IN 🡪 Body de la requête : Toutes les informations de la destination (sauf l’ID qui est auto-incrémenté)

OUT 🡪 /



## TravelController

### getTravels(idUser : Int)

URL 🡪 /travel/getTravels/{ID du user}

IN 🡪 Identifiant de l’utilisateur

OUT 🡪 Une liste d’identifiants de destinations auquel l’utilisateur s’est déjà rendue

getTravelsVisitedAround(idUser: Int, coordX:Double, coordY:Double, maxDistanceInKm: Double)

URL 🡪 / travel/getTravelsAround?idUser=?&coordX=?&coordY=?

IN 🡪 Identifiant de l’utilisateur – Coordonnées X & Y - MaxDistanceInKm= ??? par défaut c’est 0.5

OUT 🡪 Une liste d’identifiants de destinations auquel l’utilisateur s’est déjà rendue en fonction de leurs proximité déterminée par maxDistance

### getTravelsListUser (id\_destination : Int)

URL 🡪 /travel/getTravelsListUser/{ID de la destination}

IN 🡪 Identifiant de la destination

OUT 🡪 Une liste d’identifiants d’utilisateurs qui ont déjà visité le monument

### getInfoVisited (id\_user : Int, id\_destination : Int)

URL 🡪 /travel/getInfoVisited?id\_user={ID de l’utilisateur}&id\_destination={ID de la destination}

IN 🡪 Identifiant du user et de la destination

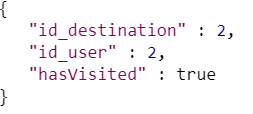
OUT 🡪 Booléen indiquant si l’utilisateur à visité le lieu ou non

### saveTravel()

URL 🡪 /travel/save

IN 🡪 Body de la requête : Toutes les informations d’un Travel

OUT 🡪 /



## WeatherController

### getWeathers(id : Int)

URL 🡪 /weather/getWeathers/{ID de la destination}

IN 🡪 Identifiant de la destination

OUT 🡪 Toutes les informations de temps reliée à cette destination

### getMostRecentWeather(id: Int)

URL 🡪 /weather/getMostRecentWeather/{ID de la destination}

IN 🡪 Identifiant de la destination

OUT 🡪 L’information de temps la plus récente sur cette destination

### saveWeather()

URL 🡪 /weather/saveWeather

IN 🡪 Body de la requête : id\_destination (int), luminosity(double), temperature (double), humidity (double)

OUT 🡪 /

### saveSound()

URL 🡪 /weather/saveSound

IN 🡪 Body de la requête : id\_destination (int), sound(double)

OUT 🡪 /