

---

# **CERC persistence reference manual**

*Release 1.0.0.2*

## **CERC Next-Generation Cities**

**Apr 02, 2024**

## CONTENTS:

<b>1</b>	<b>CERC PERSISTENCE' reference manual</b>	<b>1</b>
1.1	Authors . . . . .	1
1.2	Contributors . . . . .	1
1.3	About the PERSISTENCE . . . . .	1
1.4	Folder structure . . . . .	2
1.5	Model Classes . . . . .	2
1.5.1	Application . . . . .	2
1.5.2	City . . . . .	2
1.5.3	CityObject . . . . .	2
1.5.4	SimulationResults . . . . .	3
1.5.5	User . . . . .	3
1.6	Repository Classes . . . . .	3
1.6.1	Application . . . . .	3
1.6.2	City . . . . .	4
1.6.3	CityObject . . . . .	5
1.6.4	SimulationResults . . . . .	6
1.6.5	User . . . . .	7
1.7	Classes . . . . .	8
1.7.1	Configuration . . . . .	8
1.7.2	DB Control . . . . .	8
1.7.3	DB Setup . . . . .	11
1.7.4	Repository . . . . .	11
<b>2</b>	<b>Additional Files</b>	<b>12</b>
2.1	Readme . . . . .	12
2.2	License . . . . .	12
2.3	Code of conduct . . . . .	12
2.4	How to contribute . . . . .	12
2.5	Coding style . . . . .	12
	<b>Index</b>	<b>13</b>

## CERC PERSISTENCE' REFERENCE MANUAL

### 1.1 Authors

- Peter Yefi
- Guillermo Gutierrez Morote

### 1.2 Contributors

- Ruben Sanchez

### 1.3 About the PERSISTENCE

This document contains the essential documentation for the cerc persistence package, a set of classes, that allows the permanent storage of a cec\_hub city. cerc persistence package is composed of two main components: **repositories** and **models**.

## 1.4 Folder structure

```

../cerc_persistence/cerc_persistence/
├── configuration.py
├── db_control.py
├── db_setup.py
├── __init__.py
├── models
│   ├── application.py
│   ├── city_object.py
│   ├── city.py
│   ├── __init__.py
│   ├── simulation_results.py
│   └── user.py
├── repositories
│   ├── application.py
│   ├── city_object.py
│   ├── city.py
│   ├── __init__.py
│   ├── simulation_results.py
│   └── user.py
├── repository.py
└── version.py

```

3 directories, 18 files

## 1.5 Model Classes

### 1.5.1 Application

```
class Application(*args: Any, **kwargs: Any)
```

**Inherit:** declarative\_base

Application(Models) class

### 1.5.2 City

```
class City(*args: Any, **kwargs: Any)
```

**Inherit:** declarative\_base

City(Models) class

### 1.5.3 CityObject

```
class CityObject(*args: Any, **kwargs: Any)
```

**Inherit:** declarative\_base

CityObject(Models) class

## 1.5.4 SimulationResults

**class** `SimulationResults(*args: Any, **kwargs: Any)`

**Inherit:** `declarative_base`

`SimulationResults`(Models) class

## 1.5.5 User

**class** `User(*args: Any, **kwargs: Any)`

**Inherit:** `declarative_base`

`User`(Models) class

## 1.6 Repository Classes

### 1.6.1 Application

**class** `Application(db_name, dotenv_path, app_env)`

**Inherit:** `Repository`

`Application`(Repository) class

**delete**(*application\_uuid: str*)

Deletes an application with the `application_uuid`

**Parameter** `application_uuid` The application uuid

**Returns** None

**get\_by\_uuid**(*application\_uuid: str*) → `Application`

Fetch Application based on the application uuid

**Parameter** `application_uuid` the application uuid

**Returns** Application with the provided `application_uuid`

**insert**(*name: str, description: str, application\_uuid: str*)

Inserts a new application

**Parameter** `name` Application name

**Parameter** `description` Application description

**Parameter** `application_uuid` Unique identifier for the application

**Returns** Identity id

**update**(*application\_uuid: str, name: str, description: str*)

Updates an application

**Parameter** `application_uuid` the application uuid of the application to be updated

**Parameter** `name` the application name

**Parameter** `description` the application description

**Returns** None

## 1.6.2 City

**class** `City`(*db\_name, dotenv\_path, app\_env*)

**Inherit:** `Repository`

`City`(`Repository`) class

**delete**(*city\_id: int*)

Deletes a `City` with the id

**Parameter** `city_id` the city id

**Returns** None

**get\_by\_user\_id\_and\_application\_id**(*user\_id, application\_id*) → [`City`'>]

Fetch city based on the user who created it

**Parameter** `user_id` the user id

**Parameter** `application_id` the application id

**Returns** `ModelCity`

**get\_by\_user\_id\_application\_id\_and\_scenario**(*user\_id, application\_id, scenario*) → [`City`'>]

Fetch city based on the user who created it

**Parameter** `user_id` the user id

**Parameter** `application_id` the application id

**Parameter** `scenario` simulation scenario name

**Returns** [`ModelCity`]

**insert**(*city: hub.city\_model\_structure.city.City, pickle\_path, scenario, application\_id, user\_id: int*)

Inserts a city

**Parameter** `city` The complete city instance

**Parameter** `pickle_path` Path to the pickle

param `scenario`: Simulation scenario name

**Parameter** `application_id` Application id owning the instance

**Parameter** `user_id` User id owning the instance

**Returns** Identity id

**update**(*city\_id: int, city: hub.city\_model\_structure.city.City*)

Updates a city name (other updates makes no sense)

**Parameter** `city_id` the id of the city to be updated

**Parameter** `city` the city object

**Returns** None

### 1.6.3 CityObject

**class** `CityObject`(*db\_name, dotenv\_path, app\_env*)

**Inherit:** `Repository`

`CityObject`(`Repository`) class

**building\_in\_cities\_info**(*name, cities*)

Fetch a city object based on name and city id

**Parameter** *name* city object name

**Parameter** *cities* city identifiers

**Returns** [`CityObject`] with the provided name or alias belonging to the city with id *city\_id*

**delete**(*city\_id: int, name: str*)

Deletes an application with the *application\_uuid*

**Parameter** *city\_id* The id for the city owning the city object

**Parameter** *name* The city object name

**Returns** None

**get\_by\_name\_or\_alias\_and\_city**(*name, city\_id*) → `OptionalCityObject`]

Fetch a city object based on name and city id

**Parameter** *name* city object name

**Parameter** *city\_id* a city identifier

**Returns** [`CityObject`] with the provided name or alias belonging to the city with id *city\_id*

**get\_by\_name\_or\_alias\_in\_cities**(*name, city\_ids*) → `CityObject`

Fetch a city object based on name and city ids

**Parameter** *name* city object name

**Parameter** *city\_ids* a list of city identifiers

**Returns** [`CityObject`] with the provided name or alias belonging to the city with id *city\_id*

**insert**(*city\_id: int, building: hub.city\_model\_structure.building.Building*)

Inserts a new city object

**Parameter** *city\_id* city id for the city owning this city object

**Parameter** *building* the city object (only building for now) to be inserted

return Identity id

**update**(*city\_id: int, building: hub.city\_model\_structure.building.Building*)

Updates an application

**Parameter** *city\_id* the city id of the city owning the city object

**Parameter** *building* the city object

**Returns** None

## 1.6.4 SimulationResults

**class** `SimulationResults`(*db\_name, dotenv\_path, app\_env*)

**Inherit:** `Repository`

`SimulationResults`(`Repository`) class

**delete**(*name: str, city\_id=None, city\_object\_id=None*)

Deletes an application with the application\_uuid

**Parameter** `name` The simulation results tool and workflow name

**Parameter** `city_id` The id for the city owning the simulation results

**Parameter** `city_object_id` the id for the city\_object owning these simulation results

**Returns** None

**get\_simulation\_results\_by\_city\_id\_city\_object\_id\_and\_names**(*city\_id, city\_object\_id, result\_names=None*) →  
[SimulationResults']

Fetch the simulation results based in the city\_id or city\_object\_id with the given names or all

**Parameter** `city_id` the city id

**Parameter** `city_object_id` the city object id

**Parameter** `result_names` if given filter the results

**Returns** [SimulationResult]

**get\_simulation\_results\_by\_city\_object\_id\_and\_names**(*city\_object\_id, result\_names=None*) →  
[SimulationResults']

Fetch the simulation results based in the city\_object\_id with the given names or all

**Parameter** `city_object_id` the city object id

**Parameter** `result_names` if given filter the results

**Returns** [SimulationResult]

**insert**(*name: str, values: str, city\_id=None, city\_object\_id=None*)

Inserts simulations results linked either with a city as a whole or with a city object

**Parameter** `name` results name

**Parameter** `values` the simulation results in json format

**Parameter** `city_id` optional city id

**Parameter** `city_object_id` optional city object id

**Returns** Identity id

**update**(*name: str, values: str, city\_id=None, city\_object\_id=None*)

Updates simulation results for a city or a city object

**Parameter** `name` The simulation results tool and workflow name

**Parameter** `values` the simulation results in json format

**Parameter** `city_id` optional city id

**Parameter** `city_object_id` optional city object id

**Returns** None



### 1.6.5 User

**class** `User(db_name, dotenv_path, app_env)`

**Inherit:** `Repository`

`User(Repository)` class

**delete**(*user\_id: int*)

Deletes a user with the id

**Parameter** `user_id` the user id

**Returns** None

**get\_by\_name\_and\_application**(*name: str, application\_id: int*) → *User*

Fetch user based on the email address

**Parameter** `name` Username

**Parameter** `application_id` User application name

**Returns** User matching the search criteria or None

**get\_by\_name\_application\_id\_and\_password**(*name: str, password: str, application\_id: int*) → *User*

Fetch user based on the name, password and application id

**Parameter** `name` Username

**Parameter** `password` User password

**Parameter** `application_id` Application id

**Returns** User

**get\_by\_name\_application\_uuid\_and\_password**(*name: str, password: str, application\_uuid: str*) → *User*

Fetch user based on the email and password

**Parameter** `name` Username

**Parameter** `password` User password

**Parameter** `application_uuid` Application uuid

**Returns** User

**insert**(*name: str, password: str, role: UserRoles, application\_id: int*)

Inserts a new user

**Parameter** `name` username

**Parameter** `password` user password

**Parameter** `role` user rol [Admin or Hub\_Reader]

**Parameter** `application_id` user application id

**Returns** Identity id

**update**(*user\_id: int, name: str, password: str, role: UserRoles*)

Updates a user

**Parameter** `user_id` the id of the user to be updated

**Parameter** `name` the name of the user

**Parameter** password the password of the user

**Parameter** role the role of the user

**Returns** None

## 1.7 Classes

### 1.7.1 Configuration

```
class Configuration(db_name: str, dotenv_path: str, app_env="TEST")
```

**Inherit:** :py:class:object

Configuration class to hold common persistence configuration

**property connection\_string**

Returns a connection string postgresql

**Returns** connection string

**property db\_name**

retrieve the configured database name

**property db\_user**

retrieve the configured username

### 1.7.2 DB Control

```
class DBControl(db_name, app_env, dotenv_path)
```

**Inherit:** :py:class:object

DBFactory class

```
add_simulation_results(name, values, city_id=None, city_object_id=None)
```

Add simulation results to the city or to the city\_object to the database

**Parameter** name simulation and simulation engine name

**Parameter** values simulation values in json format

**Parameter** city\_id city id or None

**Parameter** city\_object\_id city object id or None

```
application_info(application_uuid) → Application
```

Retrieve the application info for the given uuid from the database

**Parameter** application\_uuid the uuid for the application

**Returns** Application

```
building(name, user_id, application_id, scenario) → CityObject
```

Retrieve the building from the database

**Parameter** name Building name

**Parameter** user\_id User id

**Parameter** application\_id Application id

**Parameter** scenario Scenario

:

**building\_info**(*name, city\_id*) → *CityObject*

Retrieve the building info from the database

**Parameter** name Building name

**Parameter** city\_id City ID

**Returns** CityObject

**building\_info\_in\_cities**(*name, cities*) → *CityObject*

Retrieve the building info from the database

**Parameter** name Building name

**Parameter** cities [City ID]

**Returns** CityObject

**cities\_by\_user\_and\_application**(*user\_id, application\_id*) → [City']

Retrieve the cities belonging to the user and the application from the database

**Parameter** user\_id User id

**Parameter** application\_id Application id

**Returns** [City]

**create\_user**(*name: str, application\_id: int, password: str, role: UserRoles*)

Creates a new user in the database

**Parameter** name the name of the user

**Parameter** application\_id the application id of the user

**Parameter** password the password of the user

**Parameter** role the role of the user

**delete\_application**(*application\_uuid*)

Deletes a single application from the database

**Parameter** application\_uuid the id of the application to get

**delete\_city**(*city\_id*)

Deletes a single city from the database

**Parameter** city\_id the id of the city to get

**delete\_results\_by\_name**(*name, city\_id=None, city\_object\_id=None*)

Deletes city object simulation results from the database

**Parameter** name simulation name

**Parameter** city\_id if given, delete delete the results for the city with id city\_id

**Parameter** city\_object\_id if given, delete delete the results for the city object with id city\_object\_id

**delete\_user**(*user\_id*)

Delete a single user from the database

**Parameter** user\_id the id of the user to delete

**get\_by\_name\_and\_application**(*name: str, application: int*)

Retrieve a single user from the database

**Parameter** name username

**Parameter** application application accessing hub

**persist\_application**(*name: str, description: str, application\_uuid: str*)

Creates information for an application in the database

**Parameter** name name of application

**Parameter** description the description of the application

**Parameter** application\_uuid the uuid of the application to be created

**persist\_city**(*city: City, pickle\_path, scenario, application\_id: int, user\_id: int*)

Creates a city into the database

**Parameter** city City to be stored

**Parameter** pickle\_path Path to save the pickle file

**Parameter** scenario Simulation scenario name

**Parameter** application\_id Application id owning this city

**Parameter** user\_id User who create the city

return identity\_id

**results**(*user\_id, application\_id, request\_values, result\_names=None*) → Dict

Retrieve the simulation results for the given cities from the database

**Parameter** user\_id the user id owning the results

**Parameter** application\_id the application id owning the results

**Parameter** request\_values dictionary containing the scenario and building names to grab the results

**Parameter** result\_names if given, filter the results to the selected names

**update\_application**(*name: str, description: str, application\_uuid: str*)

Update the application information stored in the database

**Parameter** name name of application

**Parameter** description the description of the application

**Parameter** application\_uuid the uuid of the application to be created

**update\_city**(*city\_id, city*)

Update an existing city in the database

**Parameter** city\_id the id of the city to update

**Parameter** city the updated city object

**update\_user**(*user\_id: int, name: str, password: str, role: UserRoles*)

Updates a user in the database

**Parameter** user\_id the id of the user

**Parameter** name the name of the user

**Parameter** password the password of the user

**Parameter** role the role of the user

**user\_info**(*name, password, application\_id*) → *User*

Retrieve the user info for the given name and password and application\_id from the database

**Parameter** name the username

**Parameter** password the user password

**Parameter** application\_id the application id

**Returns** User

**user\_login**(*name, password, application\_uuid*) → *User*

Retrieve the user info from the database

**Parameter** name the username

**Parameter** password the user password

**Parameter** application\_uuid the application uuid

**Returns** User

### 1.7.3 DB Setup

**class DBSetup**(*db\_name, app\_env, dotenv\_path, admin\_password, application\_uuid*)

**Inherit:** `:py:class:object`

Creates a Persistence database structure

### 1.7.4 Repository

**class Repository**(*db\_name, dotenv\_path: str, app\_env='TEST'*)

**Inherit:** `:py:class:object`

Base repository class to establish db connection

## ADDITIONAL FILES

### 2.1 Readme

README.md

### 2.2 License

LICENSE.md

### 2.3 Code of conduct

CODE\_OF\_CONDUCT.md

### 2.4 How to contribute

CONTRIBUTING.md

### 2.5 Coding style

PYGUIDE.md

## A

add\_simulation\_results() (*cerc\_persistence.db\_control.DBControl* method), 8  
 Application (built-in class), 2, 3  
 application\_info() (*cerc\_persistence.db\_control.DBControl* method), 8

## B

building() (*cerc\_persistence.db\_control.DBControl* method), 8  
 building\_in\_cities\_info() (*cerc\_persistence.repositories.city\_object.CityObject* method), 5  
 building\_info() (*cerc\_persistence.db\_control.DBControl* method), 9  
 building\_info\_in\_cities() (*cerc\_persistence.db\_control.DBControl* method), 9

## C

cities\_by\_user\_and\_application() (*cerc\_persistence.db\_control.DBControl* method), 9  
 City (built-in class), 2, 4  
 CityObject (built-in class), 2, 5  
 Configuration (built-in class), 8  
 connection\_string (*cerc\_persistence.configuration* :property: Configuration property), 8  
 create\_user() (*cerc\_persistence.db\_control.DBControl* method), 9

## D

db\_name (*cerc\_persistence.configuration* :property: Configuration property), 8  
 db\_user (*cerc\_persistence.configuration* :property: Configuration property), 8  
 DBControl (built-in class), 8  
 DBSetup (built-in class), 11  
 delete() (*cerc\_persistence.repositories.application.Application* method), 3  
 delete() (*cerc\_persistence.repositories.city.City* method), 4  
 delete() (*cerc\_persistence.repositories.city\_object.CityObject* method), 5  
 delete() (*cerc\_persistence.repositories.simulation\_results.SimulationResults* method), 6  
 delete() (*cerc\_persistence.repositories.user.User* method), 7  
 delete\_application() (*cerc\_persistence.db\_control.DBControl* method), 9  
 delete\_city() (*cerc\_persistence.db\_control.DBControl* method), 9  
 delete\_results\_by\_name() (*cerc\_persistence.db\_control.DBControl* method), 9  
 delete\_user() (*cerc\_persistence.db\_control.DBControl* method), 9

## G

get\_by\_name\_and\_application() (*cerc\_persistence.db\_control.DBControl* method), 9  
 get\_by\_name\_and\_application() (*cerc\_persistence.repositories.user.User* method), 7  
 get\_by\_name\_application\_id\_and\_password() (*cerc\_persistence.repositories.user.User* method), 7  
 get\_by\_name\_application\_uid\_and\_password() (*cerc\_persistence.repositories.user.User* method), 7  
 get\_by\_name\_or\_alias\_and\_city() (*cerc\_persistence.repositories.city\_object.CityObject* method), 5  
 get\_by\_name\_or\_alias\_in\_cities() (*cerc\_persistence.repositories.city\_object.CityObject* method), 5  
 get\_by\_user\_id\_and\_application\_id() (*cerc\_persistence.repositories.city.City* method), 4  
 get\_by\_user\_id\_application\_id\_and\_scenario() (*cerc\_persistence.repositories.city.City* method), 4  
 get\_by\_uuid() (*cerc\_persistence.repositories.application.Application* method), 3  
 get\_simulation\_results\_by\_city\_id\_city\_object\_id\_and\_names() (*cerc\_persistence.repositories.simulation\_results.SimulationResults* method), 6  
 get\_simulation\_results\_by\_city\_object\_id\_and\_names() (*cerc\_persistence.repositories.simulation\_results.SimulationResults* method), 6

## I

insert() (*cerc\_persistence.repositories.application.Application* method), 3  
 insert() (*cerc\_persistence.repositories.city.City* method), 4  
 insert() (*cerc\_persistence.repositories.city\_object.CityObject* method), 5  
 insert() (*cerc\_persistence.repositories.simulation\_results.SimulationResults* method), 6  
 insert() (*cerc\_persistence.repositories.user.User* method), 7

## P

persist\_application() (*cerc\_persistence.db\_control.DBControl* method), 10  
 persist\_city() (*cerc\_persistence.db\_control.DBControl* method), 10

## R

Repository (built-in class), 11  
 results() (*cerc\_persistence.db\_control.DBControl* method), 10

## S

SimulationResults (built-in class), 3, 6

## U

update() (*cerc\_persistence.repositories.application.Application* method), 3  
 update() (*cerc\_persistence.repositories.city.City* method), 4  
 update() (*cerc\_persistence.repositories.city\_object.CityObject* method), 5  
 update() (*cerc\_persistence.repositories.simulation\_results.SimulationResults* method), 6  
 update() (*cerc\_persistence.repositories.user.User* method), 7  
 update\_application() (*cerc\_persistence.db\_control.DBControl* method), 10  
 update\_city() (*cerc\_persistence.db\_control.DBControl* method), 10  
 update\_user() (*cerc\_persistence.db\_control.DBControl* method), 10  
 User (built-in class), 3, 7  
 user\_info() (*cerc\_persistence.db\_control.DBControl* method), 10  
 user\_login() (*cerc\_persistence.db\_control.DBControl* method), 11