# CERC persistence reference manual

***Release 0.1.0.2***

**CERC Next-Generation Cities**

**Nov 29, 2023**

# CONTENTS:

CHAPTER

ONE

# CERC PERSISTENCE' REFERENCE MANUAL

## 1.1 Authors

- Guillermo Gutierrez Morote
- Pilar Monsalvete Alvarez de Uribarri

## 1.2 Contributors

- Seyedehrabeeh Hosseinihaghighi
- Milad Aghamohamadnia
- Peter Yefi
- Koa Wells
- Sanam Dabirian
- Soroush Samareh Abolhassani

## 1.3 About the PERSISTENCE

This document contains the essential documentation for the CERC PERSISTENCE, a set of classes, factories, and helpers that simplifies the research at urban scale in multiples domains; these components are designed around three central axes, **extensibility**, **code clarity** and **consistency** as we intend to allow domain experts to perform urban scale simulations with multiple programs and enrich the city from several data sources. PERSISTENCE is composed of four 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

2 directories, 18 files
```

## 1.5 Model Classes

### 1.5.1 Application

**class Application**(*name*, *description*, *application_uuid*)
**Inherit: _DynamicAttributesType, Mapper\ [:obj:Any]]**
A model representation of an application

### 1.5.2 City

**class City**(*pickle_path*, *name*, *scenario*, *application_id*, *user_id*, *hub_release*)
**Inherit: _DynamicAttributesType, Mapper\ [:obj:Any]]**
A model representation of a city

### 1.5.3 CityObject

**class CityObject**(*city_id*, *building*)
**Inherit: _DynamicAttributesType, Mapper\ [:obj:Any]]**
A model representation of an application

### 1.5.4 SimulationResults

**class SimulationResults**(*name*, *values*, *city_id=None*, *city_object_id=None*)
**Inherit: _DynamicAttributesType, Mapper\ [:obj:Any]]**
A model representation of an application

### 1.5.5 User

**class User**(*name*, *password*, *role*, *application_id*)
**Inherit: _DynamicAttributesType, Mapper\ [:obj:Any]]**
A model representation of a city

### 1.5.6 UserRoles

**class UserRoles**(*value*)
**Inherit: Enum**
User roles enum

## 1.6 Repository Classes

### 1.6.1 Application

**class Application**(*name*, *description*, *application_uuid*)
**Inherit: _DynamicAttributesType, Mapper\ [:obj:Any]]**
A model representation of an application

### 1.6.2 City

**class City**(*pickle_path*, *name*, *scenario*, *application_id*, *user_id*, *hub_release*)
**Inherit: _DynamicAttributesType, Mapper\ [:obj:Any]]**
A model representation of a city

### 1.6.3 CityObject

**class CityObject**(*city_id*, *building*)
**Inherit: _DynamicAttributesType, Mapper\ [:obj:Any]]**
A model representation of an application

### 1.6.4 SimulationResults

**class SimulationResults**(*name*, *values*, *city_id=None*, *city_object_id=None*)
**Inherit: _DynamicAttributesType, Mapper\ [:obj:Any]]**
A model representation of an application

### 1.6.5 User

**class User**(*name*, *password*, *role*, *application_id*)
**Inherit: _DynamicAttributesType, Mapper\ [:obj:Any]]**
A model representation of a city

## 1.7 Classes

### 1.7.1 Configuration

**class Configuration**(*db_name: str*, *dotenv_path: str*, *app_env='TEST'*)
Configuration class to hold common cerc_persistence configuration

### 1.7.2 DB Control

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

**buildings_info**(*request_values*, *city_id*) → [CityObject'>]
Retrieve the buildings info from the database

**Parameter** request_values Building names

**Parameter** city_id 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: cerc_persistence.models.user.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: cerc_persistence.repositories.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: cerc_persistence.models.user.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*)
Creates a Persistence database structure

### 1.7.4 Repository

**class Repository**(*db_name*, *dotenv_path: str*, *app_env='TEST'*)
Base repository class to establish db connection

CHAPTER

TWO

# 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

# INDEX