**Bug Reporter System**

The purpose of this document is show the architecture, repository and api call definitions and examples.

# Use Stories

**User Story #1**

As an Operator of "Bug-Reporter-System" I would to retrieve the list of all the bugs registered So I can evaluate how many bugs were tracked during software development

**User Story #2**

As an Operator of "Bug-Reporter-System" I would to retrieve a subset of bug list based on fulltext query search So I can evaluate how many specific bugs were tracked during software development

**User Story #3**

As an Operator of "Bug-Reporter-System" I would to register a new software bug So I can track all the bugs during software development

# Architecture

# 

# Repository

Source code of this application is hosted on Github, following repository url

https://github.com/supermanu87/bug-reporter-system

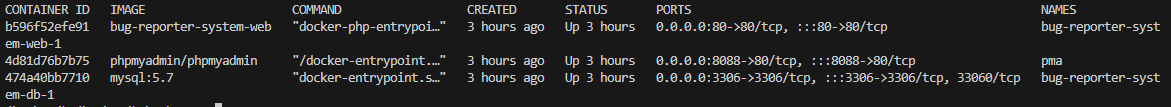
# Environments

This application could be launched in development environment with docker-compose, with following command

docker-compose up

This docker configuration bootstraps the full application including database and phpmyadmin client.

In absense of runtime errors, this is the list of containers started

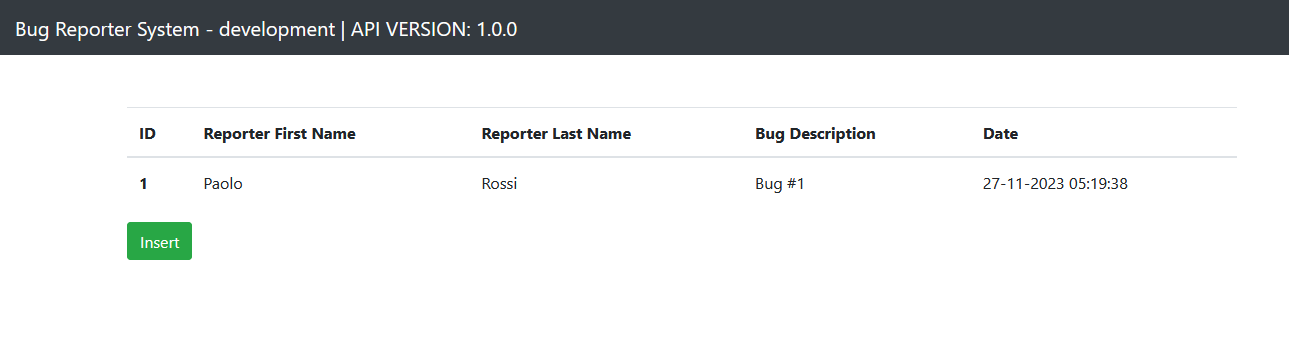


**bug-reporter-system-web**: it’s the web application based on Codeigniter 4.4.3 Framework

**phpmyadmin/phpmyadmin**: it’s the database web client

**mysql:5.7**: it’s the database server engine

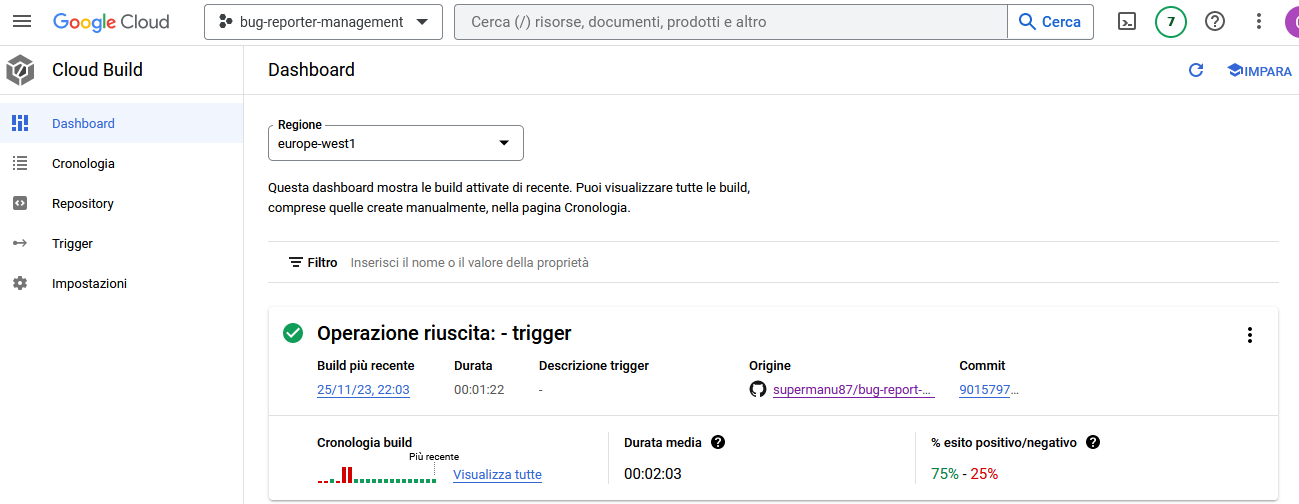
Once all the containers are started the application is reachable to: <http://localhost>

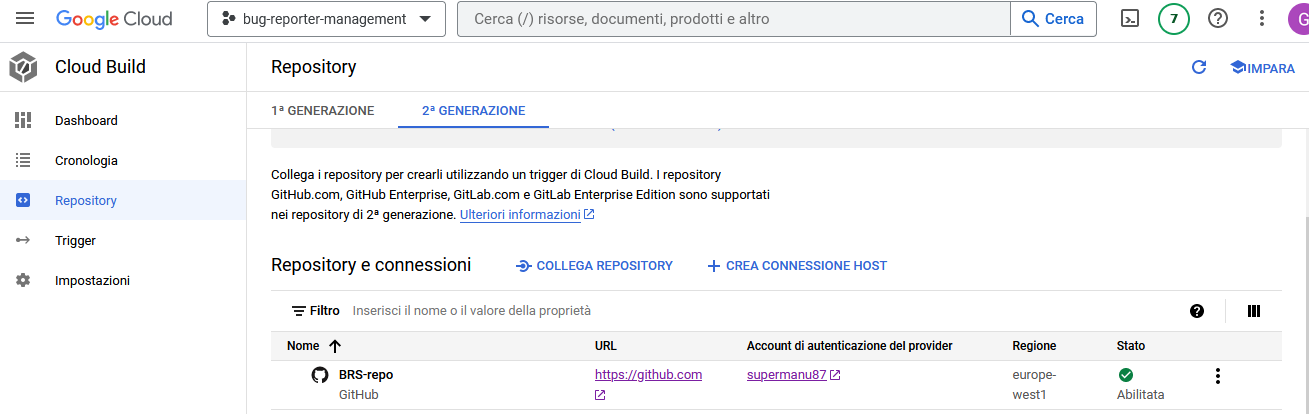


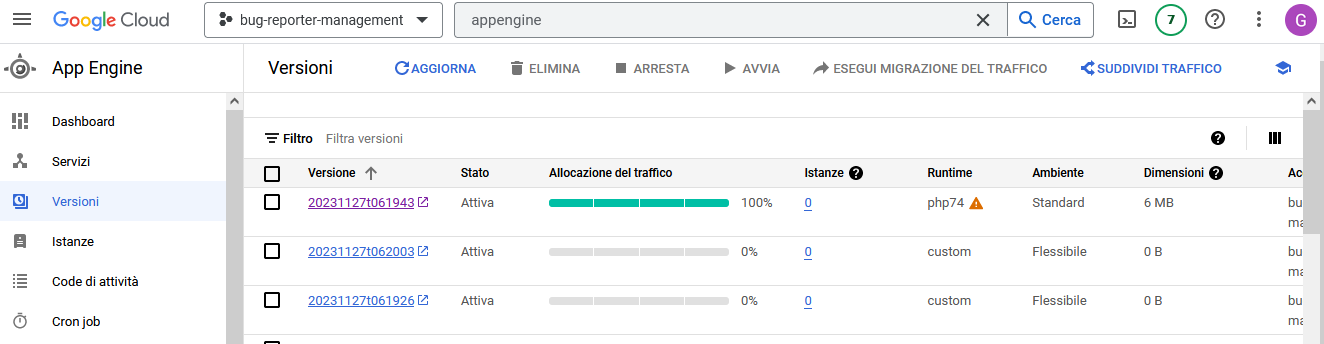
Meanwhile phpmyadmin is reachable to: <http://localhost:8088>

Regarding production environment, when a git push is executed, Google Cloud Build will pull and build the code and deploy it on Google APP Engine

This is the cloud build configuration on Google Cloud Platform

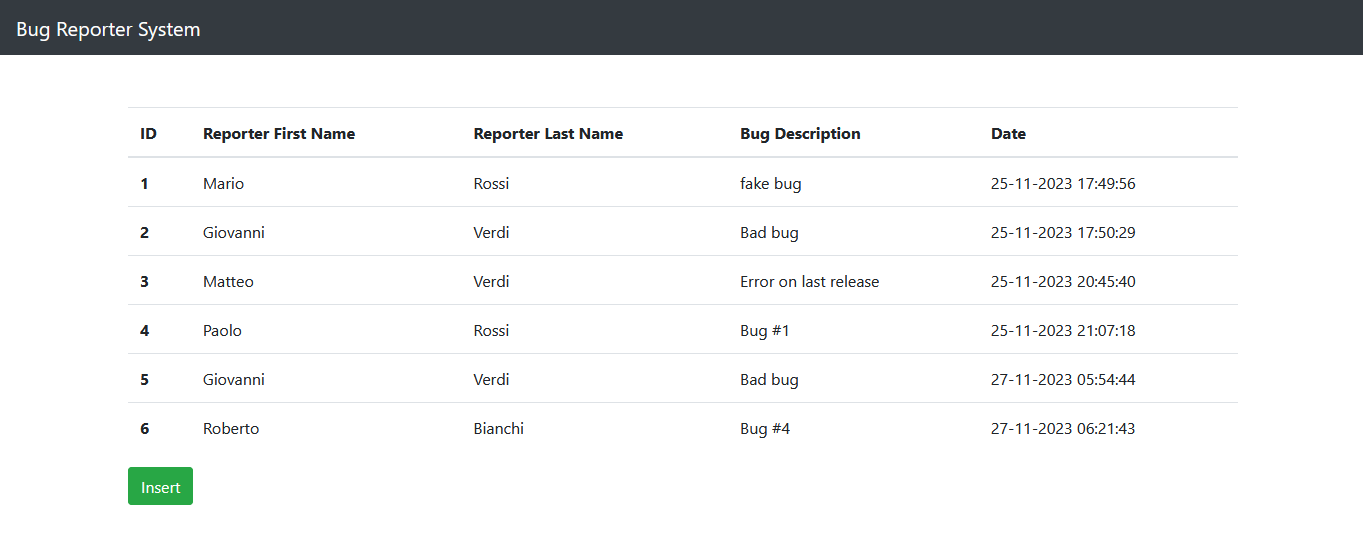


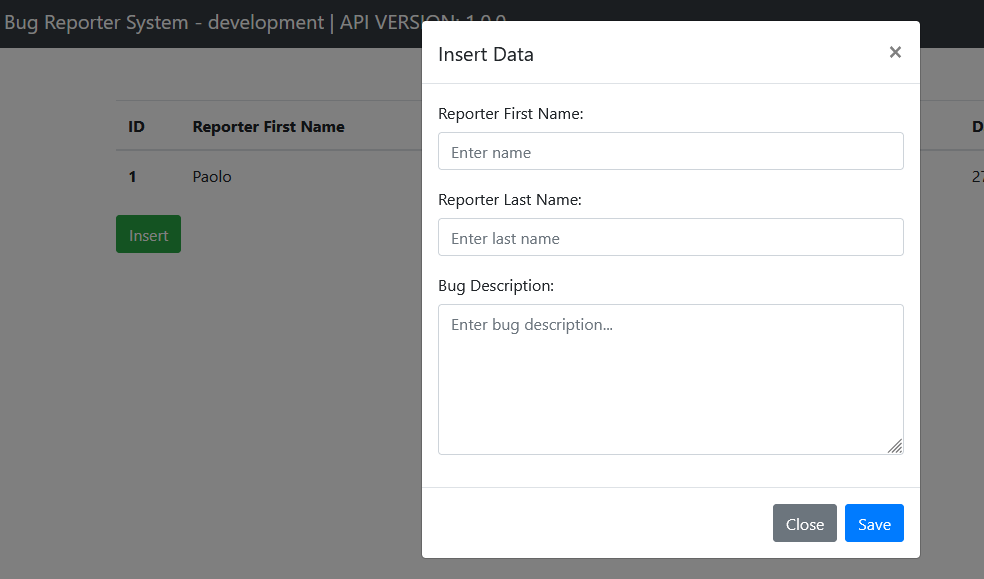




In production, the web application could be reached to the following url:

<https://bug-reporter-management.ew.r.appspot.com/>



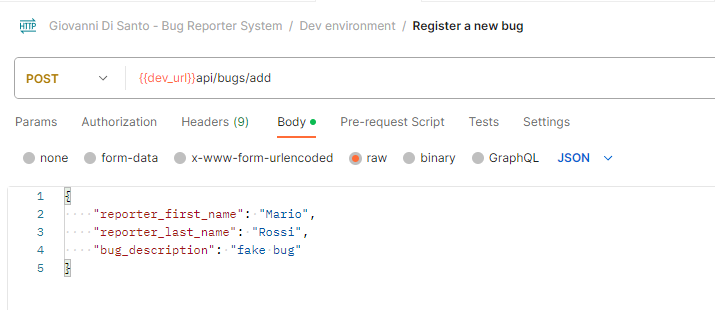


# API specification and testing

The Postman Collection is structured with 2 folders based on environments “Dev” and “Prod”

1. Register a new bug

This api is used in order to register a new bug



Request example with curl client

curl --location 'http://localhost/api/bugs/add' \

--header 'Content-Type: application/json' \

--data '{

"reporter\_first\_name": "Mario",

"reporter\_last\_name": "Rossi",

"bug\_description": "fake bug"

}'

Response

{

    "status": **true**,

    "message": "Bug successfully stored",

    "inserted\_bug": [

        {

            "id": "2",

            "reporter\_first\_name": "Mario",

            "reporter\_last\_name": "Rossi",

            "bug\_description": "fake bug",

            "creation\_date": "2023-11-27 05:53:22"

        }

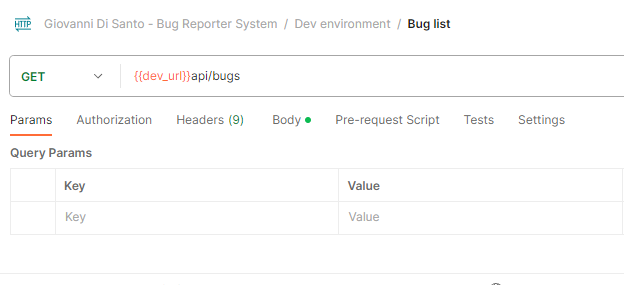
    ],

    "api\_version": "1.0.0"

}

1. List of bugs

This api is used to retrieve all the bugs registered



Request example with curl client

curl --location --request GET 'http://localhost/api/bugs' \

--header 'Content-Type: application/x-www-form-urlencoded' \

--data-urlencode 'query=fake'

Response

{

    "api\_version": "1.0.0",

    "bugs": [

        {

            "id": "1",

            "reporter\_first\_name": "Paolo",

            "reporter\_last\_name": "Rossi",

            "bug\_description": "Bug #1 ",

            "creation\_date": "27-11-2023 05:19:38"

        }

    ]

}

* 1. List of bugs with fulltext filter

This request permits to filter bug list based on filter (optional)

Request example with curl client

curl --location 'https://bug-reporter-management.ew.r.appspot.com/api/bugs?query=Mario'

Response

{

    "bugs": [

        {

            "id": "1",

            "reporter\_first\_name": "Mario",

            "reporter\_last\_name": "Rossi",

            "bug\_description": "fake bug",

            "creation\_date": "25-11-2023 17:49:56"

        }

    ]

}