University of Puerto Rico, Mayagüez Campus

Faculty of Engineering

Department of Computer Science and Engineering

Project Disaster Site Resource Locator

Report 1

Fernando Davis Rivera

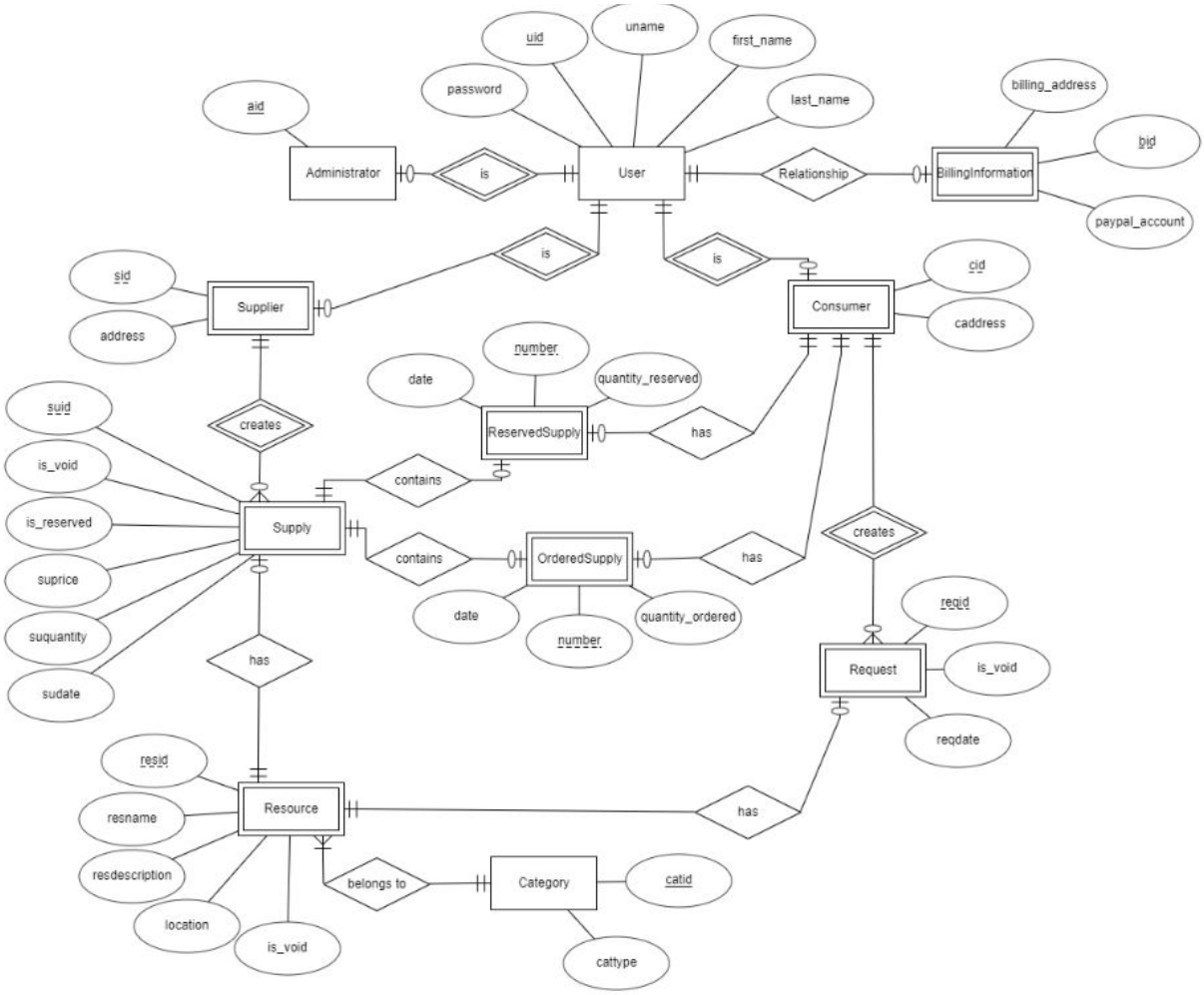
Cristian Rivera

José M. Túa

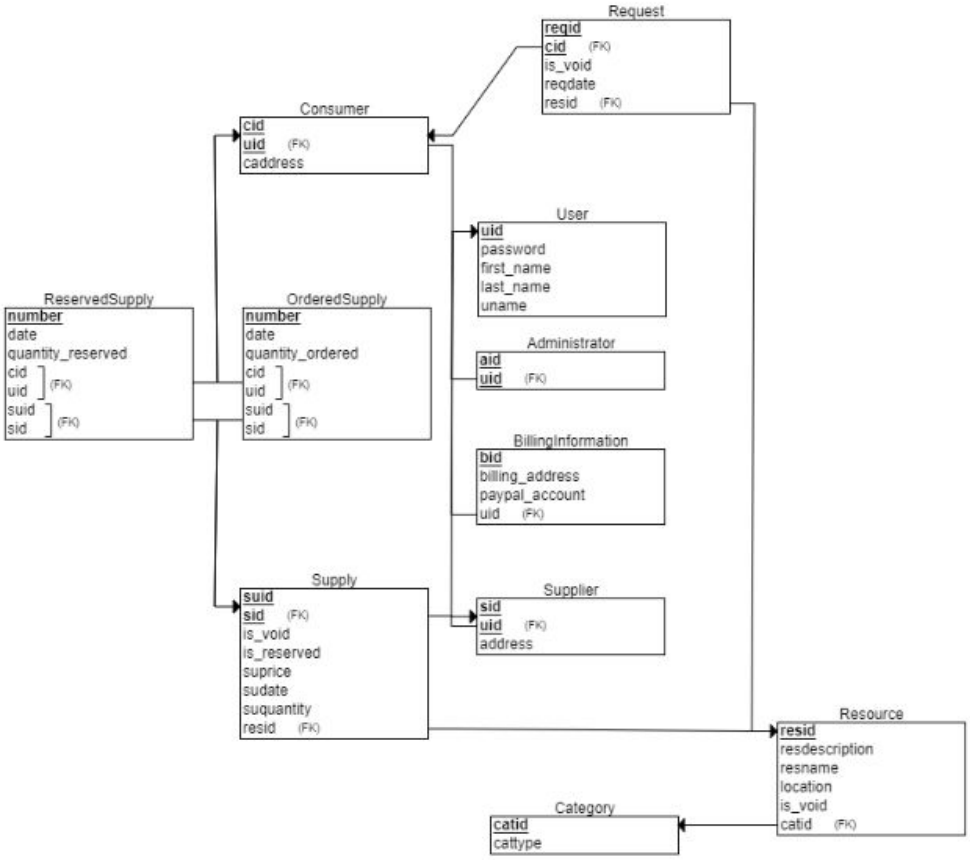
Table of Contents:

I. **Entity Relational Model**
II. **Relational Schema**
III. **Libraries/Tools Used**
IV. **Work Accomplished**
V. **Conclusion**

I. Entity Relational Model



II. Relational Schema



III. Libraries/Tools Used

For the project, we have used PyCharm to be able to implement the first phase. The libraries included are the ones presented in class. These are the following:

- Flask
- Flask Cross Origin Resource Sharing (CORS)
- Postman

We decided to not go through the route of creating a virtual machine to run the code as the professor required in class and as shown in the class recordings. Instead, we simply chose to start a localhost server with either the default port 5432 or port 5000. Depending on what you chose when you installed, Flask will change your port to your specifications, but it will usually be the default port 5432. From this, we can simply choose to sign up or log in. If upon importing the project you receive in each line of the project, it might be the Python Interpreter that needs to be configured. This is easily remedied by going through the following step:

File>Settings>Project:DisasterSite…>Project Interpreter
On the dropdown, you can choose the Python interpreter.

IV. Work Accomplished

For this phase, the database, tables, handers, daos, and paths were all worked accordingly to meet the expected requirements for this phase. As such, there is a REST API implemented to be able to carry out the following processes:

- Create a user (consumer, supplier, administrator) via signup
    - It is managed with the python console
- Verify user account via login
    - Manage user
    - It is managed with the python console
- Create a supply or a request:
    - Includes a resource at the moment of creation
    - Manage supply/request
- Fill out billing information
    - Manage billing information
- Create/Make a reservation/order for a supply
    - Manage reservations and orders.
- Browse supplies, requests, orders, reservations and users by type.

At the start, all the tables and serial values with the database dump were added, followed by the addition of the database backup with instructions on the database management which were presented in the project repo's README on GitHub. After some refactoring, files were added into their appropriate directories and users, consumers, suppliers, and administrators were added, all accessed via the login and signup methods.

The schema.sql file contains all the data creation queries if importing the database is not accessible.

Currently, commented out, are the dao objects which for the most part already work in connection to the databases but were substituted partially with static values for this first phase. All paths return static values, some have additional validations but those are not the final validations that are made with the DAOs and have no input validation or constraints.

The administrator/consumer/supplier handlers do not include a delete method for the main, this is due to constraints with the user handler seeing that they both are attached/share primary keys.