The Golden House Restaurant

Integrator Project I

Jean Carlo Alvarez Lopez

Systems engineering department, Universidad ICESI.

Juan Reyes

April 13th, 2021

Functional Requirements

The program should be able to:

RF1. Manage all the products offered by the restaurant.

RF1.1. Create a new product given its name, type, set of ingredients, and its different sizes and prices.

RF1.2. Update an existent product.

RF1.3. Delete an existent product.

RF1.4. Disable an existent product.

RF2. Delete any model's object (product, type of product, order, etc.), only if it isn't being referenced by any other object.

RF3. Disable any model's object (product, type of product, order, etc.), making it impossible to be referenced by any other object in the future.

RF4. Enable any model's object (product, type of product, order, etc.) that is currently disabled, allowing it to be referenced again by any other object.

RF5. Manage type of products.

RF5.1. Create a new type of product given its name.

RF5.2. Update an existent type of product.

RF5.3. Delete an existent type of product.

RF5.4. Disable an existent type of product.

RF6. Manage ingredients.

RF6.1. Create a new ingredient given its name.

RF6.2. Update an existent ingredient.

RF6.3. Delete an existent ingredient.

RF6.4. Disable an existent ingredient.

RF7. Manage customers.

RF7.1. Create a new customer given its names, last names, id number (optional), address, phone number, and some specific comments about that customer.

RF7.2. Update an existing customer.

RF7.3. Delete an existing customer.

RF7.4. Disable an existing customer.

RF8. Manage employees.

RF8.1. Create a new employee given its names, last names and id number.

RF8.2. Update an existing employee.

RF8.3. Delete an existing employee.

RF8.4. Disable an existing employee.

RF9. Manage system users.

RF9.1. Create a new system user given a certain employee (since all system users are employees), a username and a password.

RF9.2. Update an existing system user.

RF9.3 Delete an existing system user.

RF9.4. Disable an existing system user.

RF10. Keep the customer list ordered alphabetically in a descending way by last name and name.

RF10.1. Insert neatly a new customer into the list.

RF11. Manage orders.

RF11.1. Create a new order given a list of products, the quantity for each product, a status, the customer that requested it, the employee that will deliver it and the comments that it may have. It should also have an auto-generated code and the exact date (day and hour) when it was requested.

RF11.2. Update an existing order.

RF11.2.1. Update the order’s status from ‘REQUESTED’ to ‘IN PROCESS’ to “SENT’ to “DELIVERED”, but not backwards. The status can be also updated to “CANCELED” at any given moment.

RF11.3. Delete an existing order.

RF11.4. Disable an existing order.

RF12. Display a list of each kind of object (product, order, users, etc.) with columns displaying their main attributes.

RF12.1. Modify any of these objects after clicking its row **twice**.

RF13. Save the user that created an object, and the last user that modified it.

RF14. Serialize all the information through its objects.

RF14.1. Save the information each time an object is registered (created, imported) or modified (disabled, deleted, updated).

RF15. Export a csv file containing the orders, with a row per order and each order with the name, address and phone of the customer that requested it, the name of the employee that delivers it, the order's status, and all the order's information. It should also have three columns for each product (name, quantity, unit value).

RF15.1. Export the csv file within a date and hour requested by the user (the initial hour should be the '00:00' and final hour the '23:59' of the actual day by defect).

RF15.2. Export the csv file with its information in ascending order by date and hour.

RF15.3. Ask the user which separator wants to be used in the file before exporting the information (this will be ';' by defect).

RF16. Export a csv file containing the employees, with the number of orders delivered by them and the sum of these orders value. It should also have a row at the end with the sum of all the numeric values.

RF16.1. Export the csv file within a date and hour requested by the user (the initial hour should be the ‘00:00’ and final hour the ‘23:59’ of the actual day by defect).

RF16.2. Ask the user which separator wants to be used in the file before exporting the information (this will be ‘;’ by defect).

RF17. Export a csv file containing the products, with the number of times it was requested and the total amount of money paid for them. It should also have a row at the end with the sum of all the numeric values.

RF17.1. Export the csv file within a date and hour requested by the user (the initial hour should be the ‘00:00’ and final hour the ‘23:59’ of the actual day by defect).

RF17.2. Ask the user which separator wants to be used in the file before exporting the information (this will be ‘;’ by defect).

RF18. Display all the products in ascending order by price.

RF19. Display all the ingredients in descending order by name (alphabetically).

RF20. Search efficiently a given customer by its id (changed from “name”, since many customers may have the same name).

RF20.1. Display how much time this search took.

RF21. Import a csv file with information of customers.

RF22. Import a csv file with information of products.

RF23. Import a csv file with information of orders.