Industrial Informatics

Semester project

Retailer - Warehouse stock management application

Tester: Diana Racila
Developers: Felix-Marius Mada
Marius Dragomir
Team leader: Darius-Daniel Vikk

Table of Contents:

Introduction

- *The problem*

Stock management for an industrial parts distributor is essential in a successful business. Restocking and checking the availability of products is a routine and it should be as efficient as possible, but human error, limited work hours and several other factors can create delays, errors and inconsistencies, leading to unsatisfied customers.

- *The objectives*

Having a direct connection to a database that is continuously updated with the products and their quantities can ensure the retailer can easily restock from their warehouse at any given time and also monitor the availability of the products.

- *The specifications*

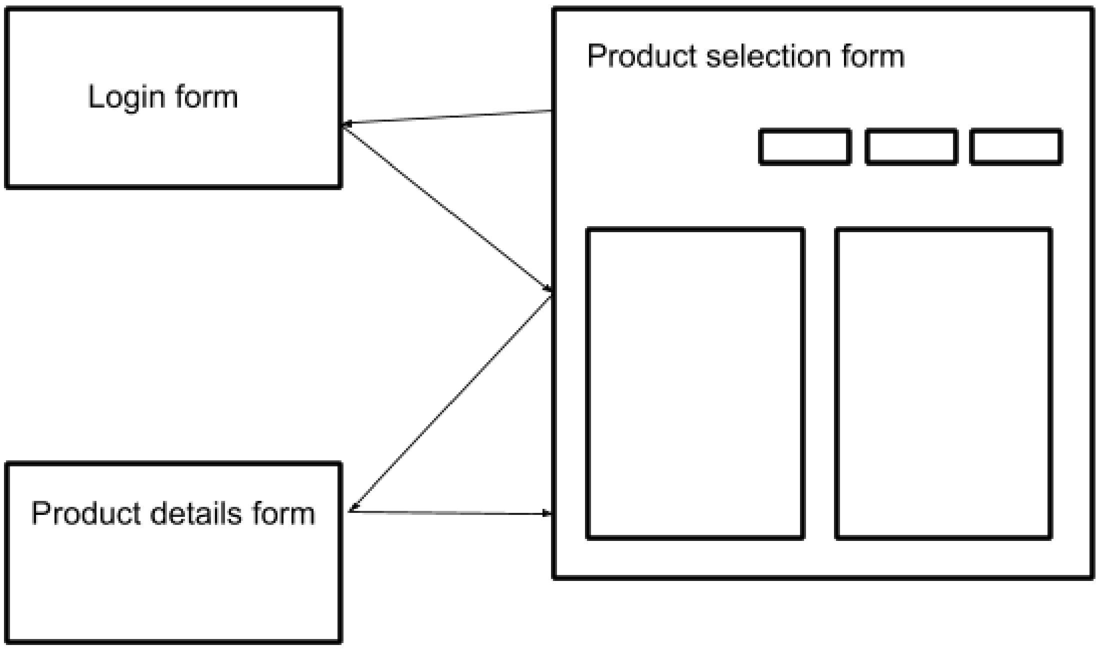
The application will consist of a Windows Forms Application with multiple forms that is going to use a web service in order to provide information from a database.

Architecture

- *Windows Form Application*

The front end of the application consists of multiple forms, the ‘Log in’, where the manager or other users would log into the application using their given username and password. Once the user successfully logs in, he/she would select one of the product categories from the left list, at which point the right list would be populated with the given products. At this point the user will be able to view all the details about the selected product in a new form by pressing the ‘View’ button.

The manager would have administrator access, enabling the functionality of the ‘Add’, ‘Delete’, ‘Edit’ buttons. These buttons would use the web service to modify the database.



Flow of the application’s forms

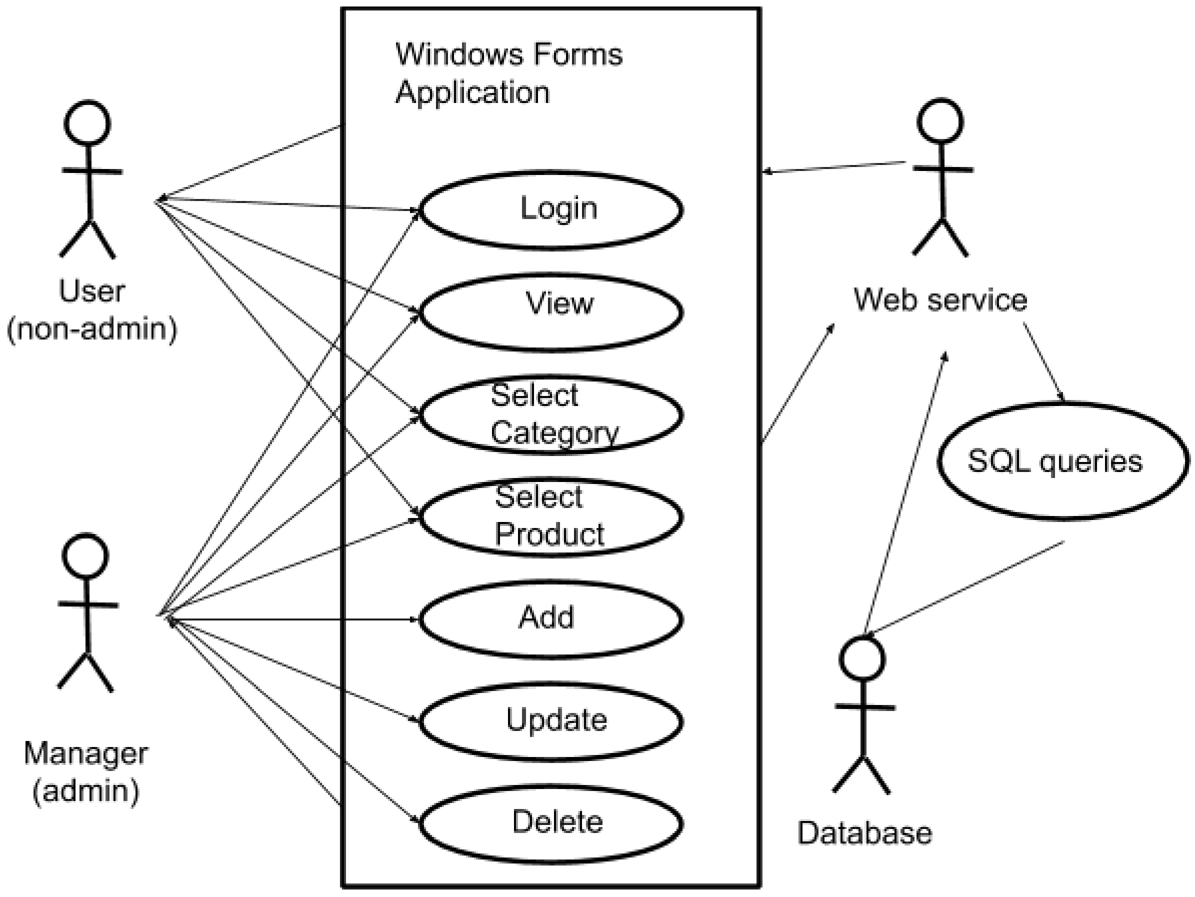
- *Web Service*

The web service is going to handle the interrogations and transactions with the database using multiple methods. The first method used, ‘check_login’, will verify the user’s credentials and granting the access to the application.

After the user has logged in, the ‘get_category_list’ method will return the data to populate the first listbox, presenting the user with the categories of products available in the retailer’s warehouse. Whenever the user will select one of the categories, ‘get_products_from_category’ method will return the products from the selected category.

Pressing the ‘View’ button, ‘get_product_details’ will present the user with the product’s details within the form.

The ‘Add’, ‘Update’ and ‘Delete’ will have their corresponding methods for interacting with the database

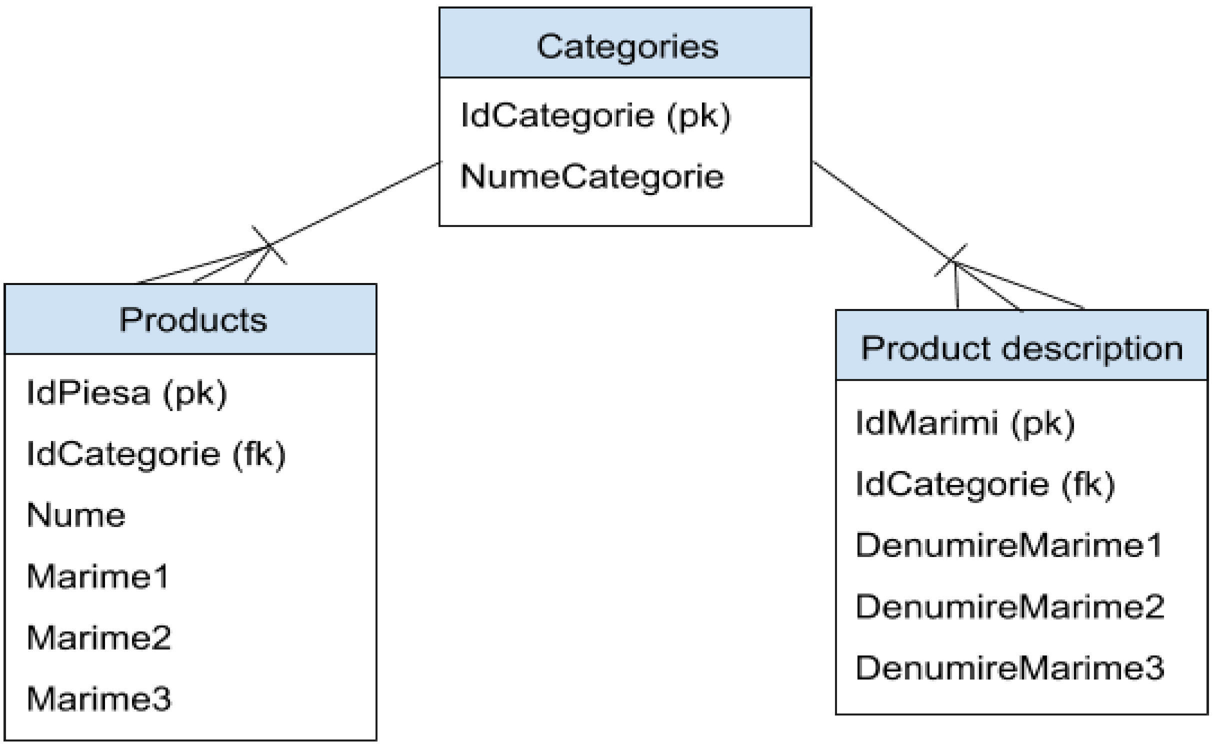


.

Use Case Diagram of the application

- *Database*

The database the application uses consists of three tables containing: the main categories of products, the products within the categories and the product details. Each table has a primary key, the main table being the ‘Categories’ and the other two, ‘Products’ and ‘Product details’ using a foreign key based on the category ‘Id’.

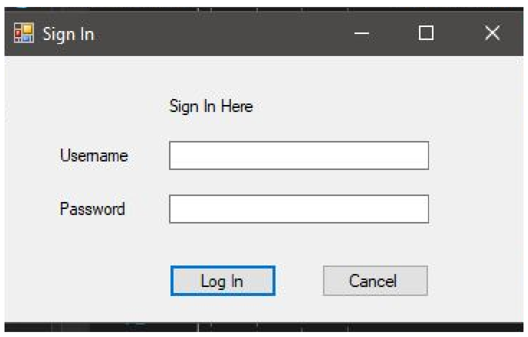


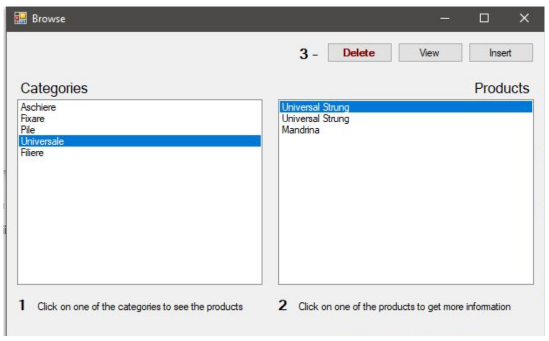
Database diagram

Implementation

- *Windows Form Application*

The from within the application are the following:

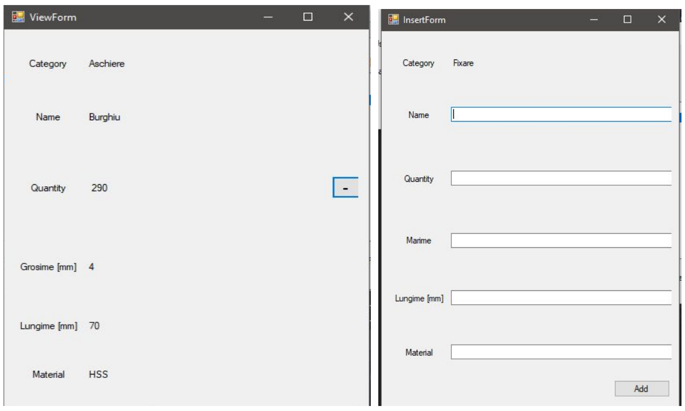




1 Click on one of the categories to see the products

2 Click on one of the products to get more information

The login form and the product brower form



Category	Aschiere
Name	Burghiu
Quantity	290
Grosime [mm]	4
Lungime [mm]	70
Material	HSS

Category	Fixare
Name	
Quantity	
Marime	
Lungime [mm]	
Material	

The view and add forms

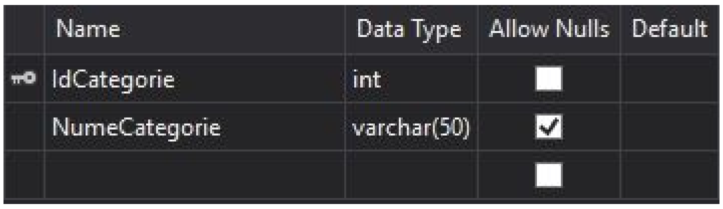
- *Web Service*

The following methods were used in the implementation of the web service:

```
        [WebMethod(Description = "Returns a dataset containing all database data")]
    DataSet get_all_db()

    [WebMethod(Description = "Returns a dataset containing category id and name")]
    public DataSet get_category_list()

         [WebMethod(Description = "Returns a dataset containing products from a category (based
on category id)")]
   public DataSet get_products_from_category(int categoryId)
    [WebMethod(Description = "Returns a dataset containing the details of a product (based on
product id)")]
    public DataSet get_product_details(int productId)

    [WebMethod(Description = "Returns a dataset containing the names of the properties of a given
category (based on category id)")]
    public DataSet get_category_property_names(int categoryId)

    [WebMethod(Description = "Adds a product to the database")]
    public void add_product(int categoryId, String name, string v1, string v2, string v3, int quantity)

    [WebMethod(Description = "Change product details")]
    public void update_product_details(int productId, int categoryId, String name, string v1, string v2,
string v3)

    [WebMethod(Description = "Change product quantity")]
    public void update_product_quantity(int productId, int changedVal)

    [WebMethod(Description = "Delete a product from database")]
    public void delete_product(int productId)

    [WebMethod(Description = "Returns 0 if login is unsuccessful, 1 if normal login is successful, 2 if
login is successful and has admin rights")]
    public int check_login(string userName, string passVal)
```

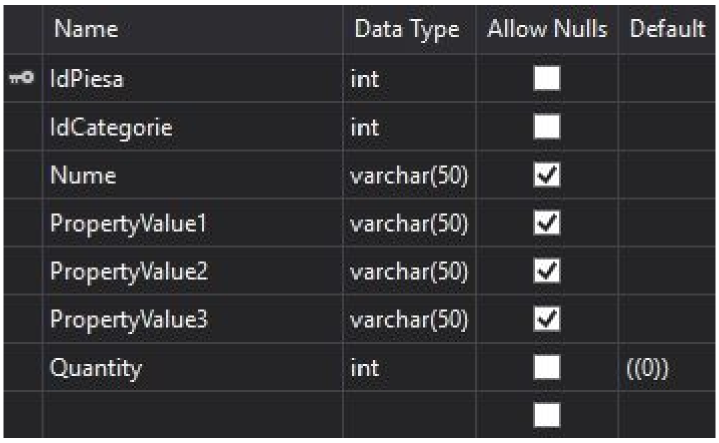
- *Database*

Three tables were created within the database with the following table definitions:



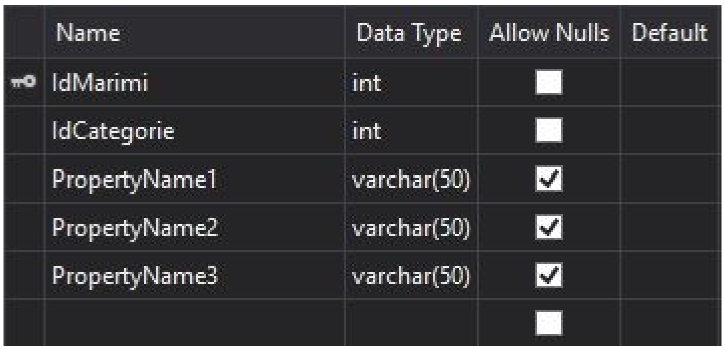
	Name	Data Type	Allow Nulls	Default
🔑	IdCategorie	int	☐	
	NumeCategorie	varchar(50)	✓	
			☐	

Category table definition



	Name	Data Type	Allow Nulls	Default
🔑	IdPiesa	int	☐	
	IdCategorie	int	☐	
	Nume	varchar(50)	✓	
	PropertyValue1	varchar(50)	✓	
	PropertyValue2	varchar(50)	✓	
	PropertyValue3	varchar(50)	✓	
	Quantity	int	☐	((0))
			☐	

Product table definition



	Name	Data Type	Allow Nulls	Default
🔑	IdMarimi	int	☐	
	IdCategorie	int	☐	
	PropertyName1	varchar(50)	✓	
	PropertyName2	varchar(50)	✓	
	PropertyName3	varchar(50)	✓	
			☐	

Product properties table definition

Testing

In such application smooth functionality and proper security is mandatory. Thorough testing has been performed to ensure that no critical flaws were left unsolved.

Regarding the front end, the user can’t log in if credentials are not valid or left empty. The ‘Add’, ‘Delete’ and ‘View’ buttons are not enabled until the user selects one of the product categories.

The user cannot press any buttons until at least a category is selected. (fixed bug)

On the web service side the possibility of SQL injection has been eliminated from the ‘get_products_from_category’ and ‘update_product_details’ methods.

Clicking somewhere else in the list box besides on an item the ‘Delete’, ‘Insert’ and ‘View’ buttons remain enabled. (fixed bug)

Pressing any of the buttons when nothing is selected will activate the first item in the category list. (fixed bug)

- Known bugs:

The view form doesn’t clear previous properties and names of products (fixed) and pressing the ‘-’ button is unreliable, resulting in exceptions in the web service.

Annex

The full source code can be accessed via github using the following link:

https://github.com/MariusDgr/ProiectII