|  |
| --- |
| IT Storage management System  Technical Documentation |
| Viktoria Bors-Pajuste |





*ProOktatas*

*C# Programozo kepzes – Basic Modul*

*2023.April – 2023.November*

## Table of Contents

1. Introduction

1.1 Purpose of the Application

1.2 Target Audience

1.3 Database and Technology Stack requirements

2. Database Structure

2.1 Database table creation

2.2 Tables and their relationships

2.3 MySQL Connection

3. User interface

3.1 Main Window (Form1)

3.2 Storage Window

3.3 Computer Part Window

3.3.1 Choosing Computer Type Window

3.4 Computer Part Detail Window

3.5 Computer Part Pieces In Storage Window

3.6 Database Pre-Request Window

3.7 Computer Part Control

4. Classes

4.1 Computer part classes

4.2 Static Class

4.3 Design Static Class

4.4 MySql Datahandler Class

4.5 Function Result Class

5. Error handling

6. User functionality

6.1 Filtering and searching

6.2 Export to CSV and Import from CSV

6.3 Logging

7. Conclusion

7.1 Summary of functionality

7.2 Future Considerations

## Introduction

#### Purpose of the application

The application’s purpose is to help small business specified to selling and importing computer parts to track their storage balance. It is a user-friendly, easy to use desktop application.

#### Target audience

This technical documentation is intended for developers who may need to understand, maintain, or extend the functionality of the IT Storage Management System.

#### Database and technology stack requirements

The application uses relational database (namely MySql database) for data storage and written in C# with .Net framework.

## Database Structure

#### 2.1. Database table creation

The application handles the creation of the database and its tables automatically upon the first launch. This process occurs when the application is opened for the first time on a computer where the "storagemanagmentdb" database does not exist.

#### 2.2. Tables and their relations

There are 4 basic storage tables:

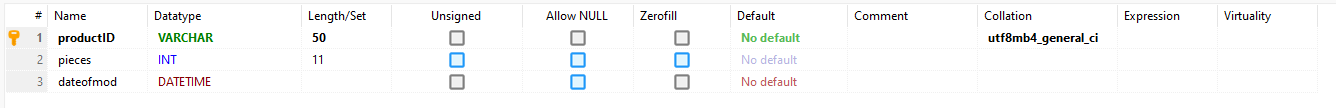
A. Main Storage,

B. Secondary Storage,

C. Export Storage,

D. Extra Storage

All four table consists of 3 field:



* ProrductID: Serves as a Primary Key.
* Pieces: Int and show how many part we have in the given storage.
* Dateofmod: Date of modification.

Additionally, there are detail and value tables. The detail table describes the properties of each computer part, while the value table contains the actual values associated with a given computer part.

The value table consists of the following fields:

A screenshot of a computer

Description automatically generated

* ID: Autogenerated at the time of insertion and it serves as a Primary Key.
* DescriptionID: Links to the ID fields in the description table, specifying the type of computer part in the corresponding line.
* ProductID: A programmatically generated field used to identify the specific product, subsequently inserted into the database.
* The basic information of the computer part does not allow for the insertion of null values, while the rest of them does.

The Description Table contains the following fields:

A screenshot of a computer screen

Description automatically generated

* ID: Autogenerated at the time of insertion and it serves as a Primary Key.
* PartType: A brief description indicating the type of computer part in the respective line.
* All fields in both tables are of VARCHAR type. The basic information of the computer part does not allow for the insertion of null values, while the remaining fields can be nullable, accommodating null values during insertion.

The relation of the value and description table is the following:

A white rectangular object with black text

Description automatically generated

#### 

#### 2.3 MySql connection

Database connection is only open when database action needed, for example inserting a new record. After that the connection is closed.

## User Interface

All windows, excluding Form1 (the main window), are located within the "Forms" folder in the project directory.

#### 3.1 Main window

The main window serves as the starting point for the application. Users can select a storage and view the type of computer part stored within it. Additionally, users can explore all types of computer parts, allowing them to create new parts or modify existing ones. Importing a computer part into the database is also possible with a button by selecting a CSV file. The main window employs picture boxes for the storages, flow layout panels, and Part Control Cards to facilitate the selection of computer parts.

#### 3.2 storage window

The storage window serves the purpose of displaying the types of computers stored in a specific storage. Accessible by clicking on one of the picture boxes representing the storage on the main form. The Storage Window lists computer parts in a listbox.

Storage Window Features:

* Search Section: Includes a textbox and two numeric up-down components for searching by product ID, type/name, and a specified price range. A button allows users to clear search options.
* Add/Delete Operations: Users can add or delete parts directly from the storage.
* Storage Value: A button provides the option to view the total value of the items stored in the selected storage.

These features enhance the user's ability to manage and interact with the contents of each storage efficiently.

#### 3.3 computer part window

The Computer Part window is accessible from the main window by clicking either the "Create New Part" or "Modify Given Part" buttons. Different components are utilized based on the type of computer part selected.

##### 3.3.1 Choosing Computer Type Window

When adding a new part to the database, a selection form appears where the user can specify the type of new computer part they want to create. The form incorporates picture boxes for easy navigation.

#### 3.4 computer part detail window

This form is activated when the user double-clicks on a part within the list on the Storage Window. In this window, users can access detailed information about the selected part. The user has the option to export this information to a CSV file for future use.

#### 3.5 computer part pieces in storage window

This window is accessed when the user wants to add or delete a part from the selected storage in the Storage Window.

* Add Operation

Available Computer Parts List: When adding a part, a list displays all computer parts available in the entire database, not just the current storage. The list initially selects the same computer part as chosen from the listbox in the Storage Window.

* Delete Operation

Selected Computer Part: When deleting a part, the user cannot choose any other computer part from the list.

Additional information displays the current quantity of the selected part in the given storage and the expected quantity after the add or delete operation..

#### 3.6 database pre-request window

The very first, initial interaction users have with the application is through the Database Initialization Window. This window prompts users to create the necessary database and its tables. Users are presented with two options: accepting the prompt results in the creation of the database and tables, after which the main application window appears, allowing the application to run. Conversely, if the user declines, the application will be closed.

#### 3.7 computer part control

This is not a separate window; rather, it is a Control Card embedded within the Main Window's flow layout panel. Within this card-like form, users can view all the computer parts stored in the database. The control card includes a checkbox that aids users in selecting the specific part they wish to modify on the Main Window.

## Classes

#### 4.1 Computer part classes

##### 4.2.1 Part Abstract Class

This serves as the base/abstract class for all computer parts, featuring the following properties:

* Id: Equals the product ID in the value table.
* Manufacturer: String.
* Type: String, denoting the product name.
* Price: Int., in HUF.
* Warranty: Int, in months

Additionally, there is an abstract function named BruttoPrice.

##### 4.2.2 Processor/CPU Class

The price property of the Processor class is calculated/generated and cannot be manually assigned.

This class encompasses the following attributes:

* Manufacturer: AMD or Intel.
* Package: Enum from the Packaging (LGA1200, LGA1700, AM4, or AM5). Depending on the manufacturer, only the respective packaging type is acceptable (LG for Intel, AM for AMD).
* ClockSpeed: Int, in Mhz.
* L3Size: Int, L3 cache size, in MB.
* Cores: Int, number of cores.

Prices are generated using the given equation:

**Price = (clock speed / 100) \* l3 size \* cores;**

4.2.3 Motherboard Class

This class contains:

* Manufacturer: ASRock, ASUS, MSI, Gigabyte.
* ProcessorSocket: Enum for the type of processor socket.
* Chipset: Enum for the type of chipset, depending on the processor socket.
* Illuminated: Bool.

4.2.4 RAM/Memory Class

Includes:

* Manufacturer: APACER, Crucial, Corsair, Hynix, Kingstone, Gskill.
* Price: Generated according to the equation:

**Price = (clock speed / 10) \* size \* 2;**

* Size: Int.
* Gen: Enum for generation (DDR4, DDR5).
* Clock speed: Depends on the generation (3000-4400mhz for DDR4, 5200-8000mhz for DDR5), given in Mhz.
* Timing: Int, in CL.

4.2.5 Graphic Card/GPU Class

Contains:

* Manufacturer: AMD, Intel, NVIDIA.
* Ram: Enum for RAM type (GDDR6, GDDR6X).
* Size: Int, in MB.
* Core clock speed: Int, in Mhz.
* Power consumption: Int, in watt.
* Raytracing: Bool.

4.2.6 Power Supply Unit/ PSU Class

Contains:

* Power output: Int, in watt.
* Quality: Enum for quality (Bronze, Silver, Gold).

#### 4.2 Static class

The static class encompasses enums, parameters, and functions crucial for widespread application use.

* ModificationType Enum (add, delete): Utilized when updating computer part quantities in a specific storage.
* PreProductID Enum (Proc, MotherB, Mem, Video, PSU): Used for generating unique product IDs after adding new parts to the value table.

It features three essential functions: Logging, ExportPartToCSV, and ImportPartToDatabase.

#### 4.3 design static class

The Design Static Class was designed to facilitate hover effects on picture boxes and labels.

#### 4.4 mysql datahandler class

This class manages basic database connections, checks for the program's first run, and creates necessary data tables if they don't exist. It includes functions for creating, modifying, and reading computer parts. Additionally, there are storage table-related functions (insert, delete, update) and helper functions like PartExistInValueTable, which checks if a product ID already exists, and ProductId\_Update, which updates/generates product IDs during new insertions.

#### 4.5 function result class

The Function Result Class is a helper class providing detailed information about function outcomes. It includes:

* FunctionResultType Enum (ok, error, fatal): Indicates success, error, or fatal outcome.
* Result: Boolean result.
* Message: Customized message.

Example of usage:

Importing a computer part to database form a CSV file, where first, program checks that the new computer part already exists in our database based on its product id.

public static FunctionResult PartExitsInValueTable(string id)

{

FunctionResult result = new FunctionResult();

int piece;

try

{

connection.Open();

using (MySqlCommand cmd = new MySqlCommand("SELECT Count(\*) FROM `valuetable` WHERE productId=@id", connection))

{

cmd.Parameters.AddWithValue("@id", id);

piece = Convert.ToInt32(cmd.ExecuteScalar());

result.Result = piece > 0 ? true : false; **// result is true if there is already a part with the given productid and false if there is not.**

result.Message = result.Result ? "There is already a part with this product ID." : "There has not saved part with this product ID."; **// depend on the result different message can be shown.**

result.Fresult = FunctionResultType.ok**;// successfully connected to database**

}

}

catch (Exception ex) **// fatal error during connection or reading of database**

{

result.Message = ex.Message;

result.Result = false;

result.Fresult = FunctionResultType.fatal;

}finally { connection.Close(); }

return result;

}

## Error Handling

Error handling mechanisms are implemented to ensure robustness and graceful degradation in case of unexpected issues.

The application employs try-catch blocks consistently whenever a database handler or function is invoked. Additionally, the Function Result Class has been integrated to manage errors, allowing for the provision of more comprehensive details in case of an issue.

## User Functionality

#### 6.1 filtering and searching in a given storage

In each storage, users can filter computer parts based on part type and product ID. Additionally, there's an option to filter by price range. Real-time filtering is applied for type and product ID, while the price range filter utilizes a dedicated filter button. Users can easily delete filter options using the provided delete filter option button.

#### 6.2 export computer part to csv and import from csv to datbase

Exporting a specific computer part to a CSV file takes place within the Part Detail Form. Each CSV file corresponds to a single computer part, and users have the flexibility to choose the file path and name.

Importing a CSV file to the database is initiated from the button on the Main Window. Prior to blindly inserting data from the CSV file into the database, a check is performed to verify whether the specified computer part already exists in our value table, based on its product ID.

#### 6.3 logging

Logging occurs exclusively when the quantity of a specific computer part changes within a designated storage. The log captures details such as the storage name, the computer part, the extent of the changes (quantity), the nature of the changes (addition or deletion), and the timestamp indicating when the changes occurred.

## Conclusion

#### 7.1 Summary

The IT Storage Management System is a beginner application for managing IT components in warehouses. It provides a user-friendly interface, data storage with MySQL, and various features for component manipulation.

#### 7.2 future considerations

Future improvements may include additional features, enhanced user interfaces, and optimizations for scalability.