

20.08.20

# **Leda**

## **Electronic Part Tracker**

1.0.2

By  
Mustafa C. Ülker

Pievision 2020

The main purpose of the project is easily tracking the count of parts and displaying important features. The main page is designed for this purpose.

Parts can be defined spec by spec, which allows users to create detailed, clear parts and features of this part. While defining the new part feature, the user can choose the visibility of this feature. Only visible selected items will be shown on the Main page. This application feature is designed for a noncrowded, clear feature track.

Also, on the List Part page, the user can look at both visible and invisible features.

Export data application feature, export List Part page as .xlsx file to the desired path.

### **Used Libraries & Environments**

- ❖ IntelliJ IDEA 2020.2
- ❖ Java JDK 14.0.2
- ❖ JavaFX 14.0.2
- ❖ SQLite JDBC 3.30.1
- ❖ Apache POI 4.1
- ❖ Install4j
- ❖ SQLite Expert

### **UML diagrams**

#### **Use case Diagram**

The user manages the electronic part stock via GUI.

GUI and Database log their events. The user or the developer can analyze the log file.

Database managed by UI Controllers, indirectly by the user.

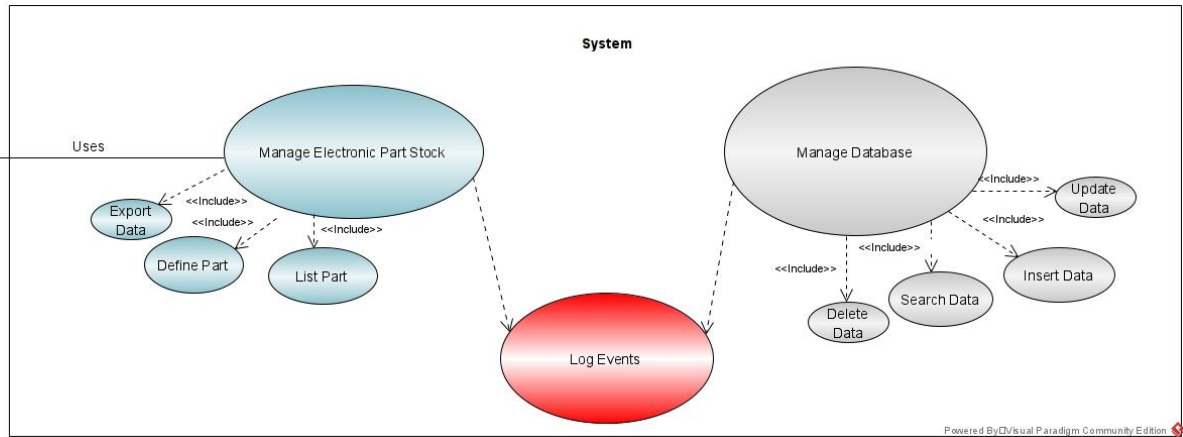


Figure 1. Use Case diagram for Project Leda

## Sequence diagram

Sequence diagram for “Add Row” request by the user via its button.

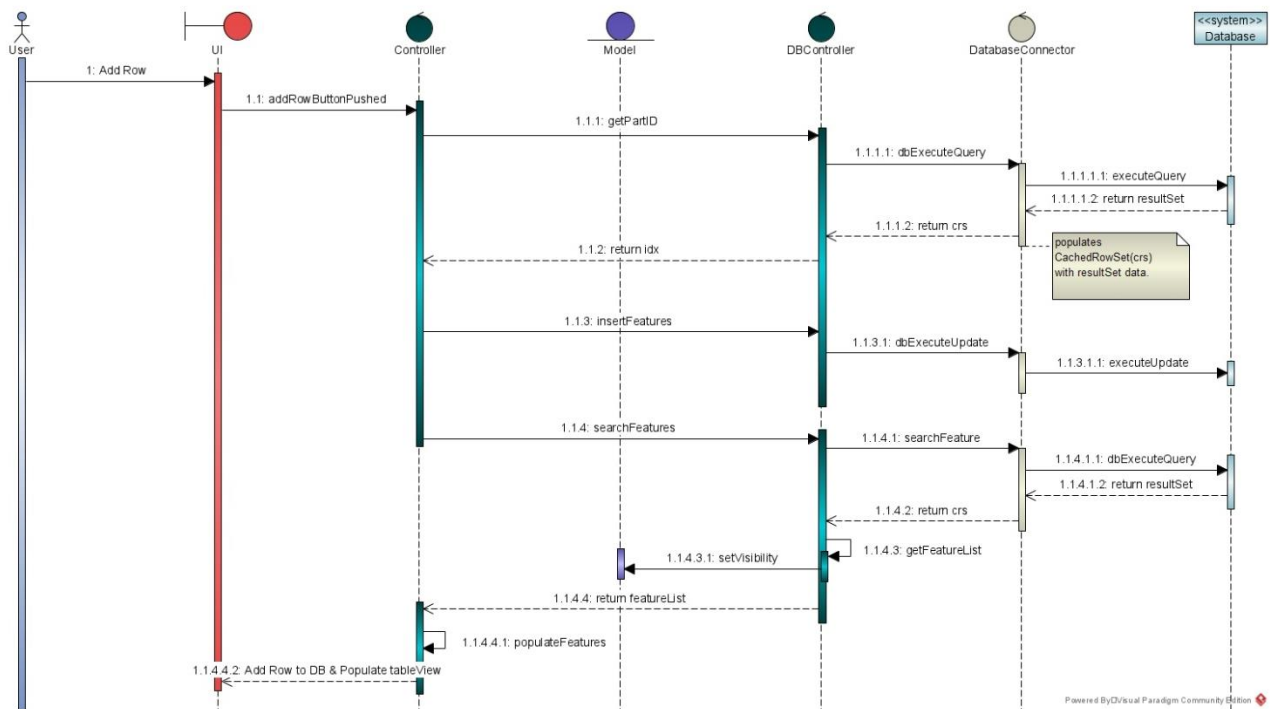


Figure 2. Sequence diagram for "Add Row"

## Class diagrams

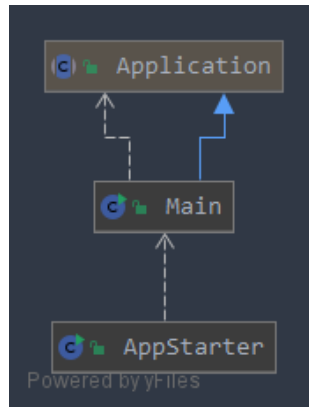


Figure 3. AppStarter Class Diagram

AppStarter class calls Main class. This class is used for application deployment. If this class is not created, the following problem is encountered when trying to run the “.jar” file. “Error: Java FX Packager: Can't build artifact - fx:deploy is not available in this JDK”

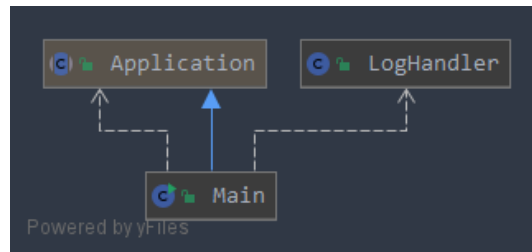


Figure 4. Main Class Diagram

Main class loads Main Page, calls LogHandler class, and adjusts Splash screen logo.

```
new FileHandler( pattern: "Leda_Log.log");
```

Figure 5. FileHandler Path String5

LogHandler class creates a log file in the application execution path. All user & application actions are kept in this file.

Two levels were used in logging. Info and Warning.

I used info messages for when the actions are reacted as it is supposed to be. Warning messages for unusual events or errors.

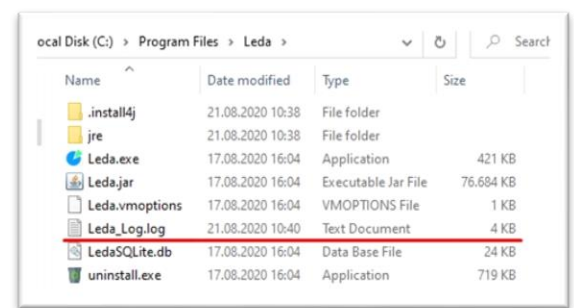


Figure 6. Log File in Installation Path

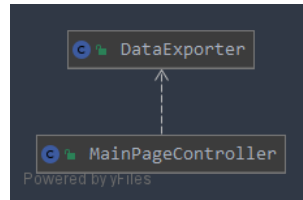


Figure 7. DataExporter Class Diagram

MainPageController calls DataExporter to export the database to .xlsx file when exporter called.

```
File file = fil_chooser.showSaveDialog(new Stage());
```

Figure 8. FileChooser String

The export path is chosen by the user via FileChooser.

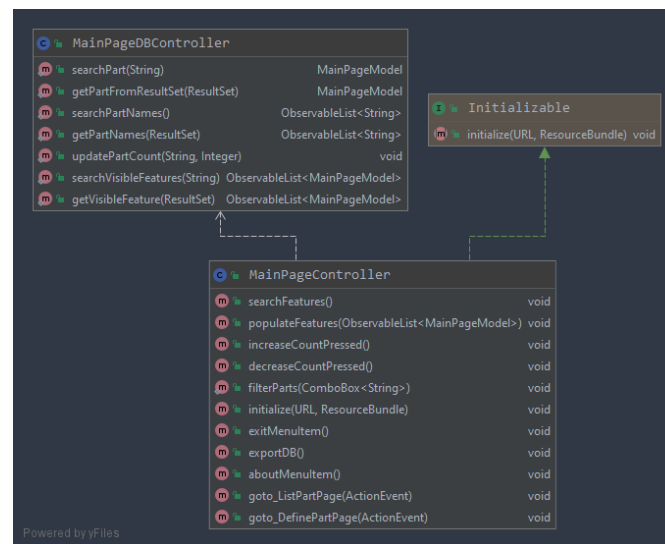


Figure 9. MainPageController connected to MainPageDBController Class Diagram

MainPageController controls Main Page, this class contains onAction functions of MainPage view. MainPageDBController executes DB queries of this page.

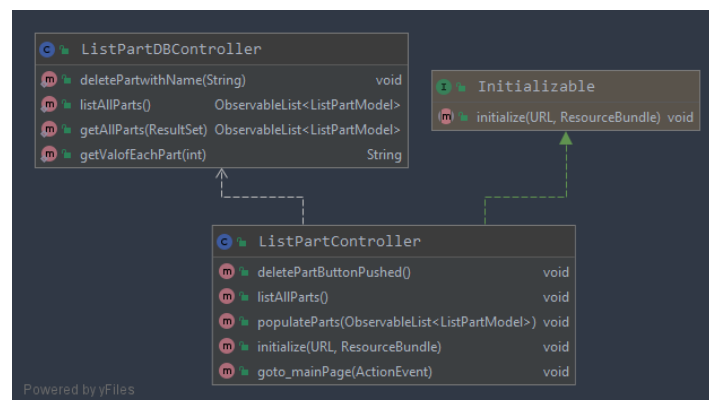


Figure 10. ListPartController connected to ListPartDBController Class Diagram

ListPartController controls ListPart Page, this class contains onAction functions of ListPart view. ListPartDBController executes DB queries of this page.

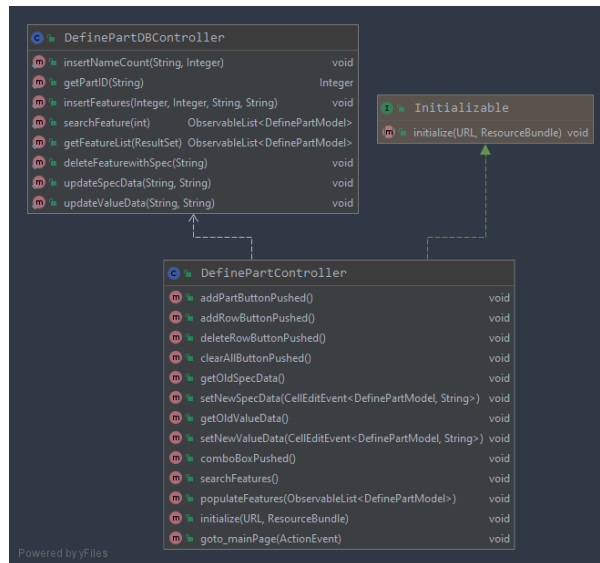


Figure 11. DefinePartController connected to DefinePartDBController Class Diagram

DefinePartController controls DefinePart Page, this class contains onAction functions of DefinePart view. DefinePartDBController executes DB queries of this page.

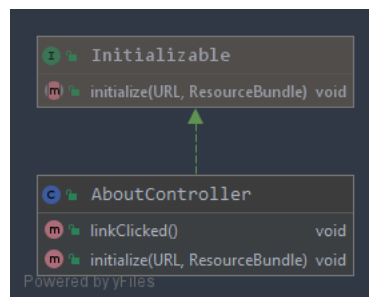


Figure 12. AboutPage Class Diagram

AboutController controls About Page contains onAction functions of About view.

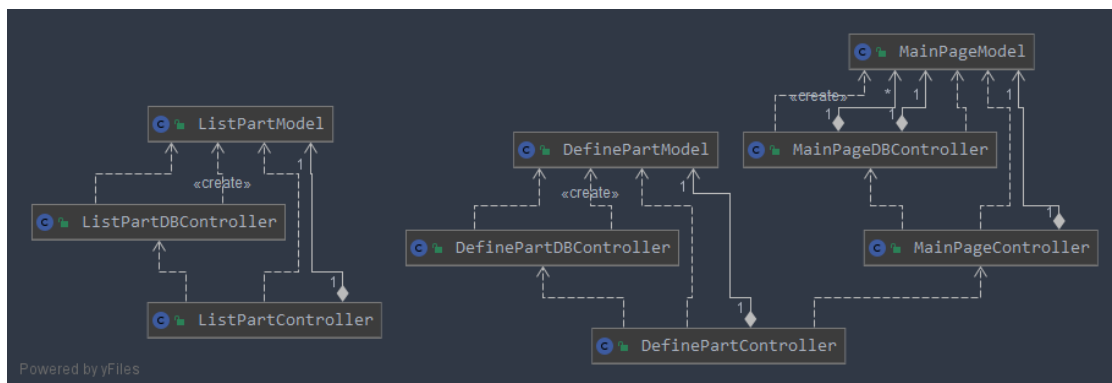


Figure 13. Controller, DBController, Model diagrams of three pages

All page classes with their Model classes. DefinePartController class calls filterParts() method from MainPageController class.

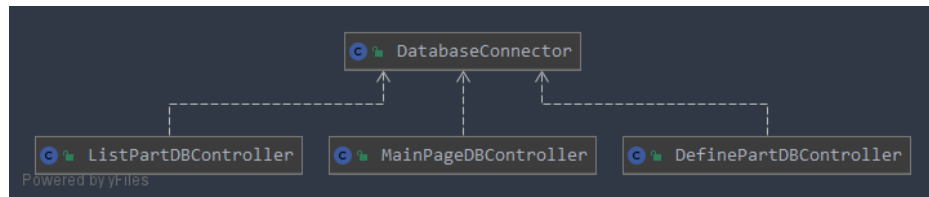


Figure 14. DatabaseConnector and DBControllers Diagrams

All DBController classes connect to the DatabaseConnector class. Which connects to DB and executes DB queries in the database.

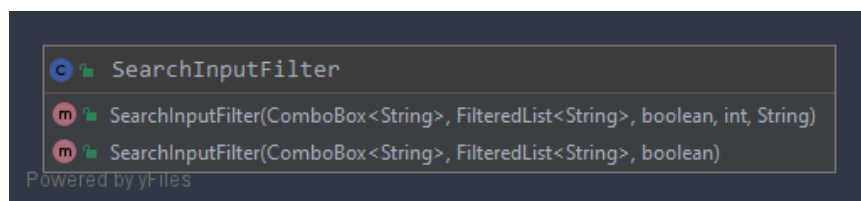


Figure 15. SearchInputFilter Class Diagram

SearchInputFilter filters comboBox elements based on user input.

### 3.3 Views

#### Main Page

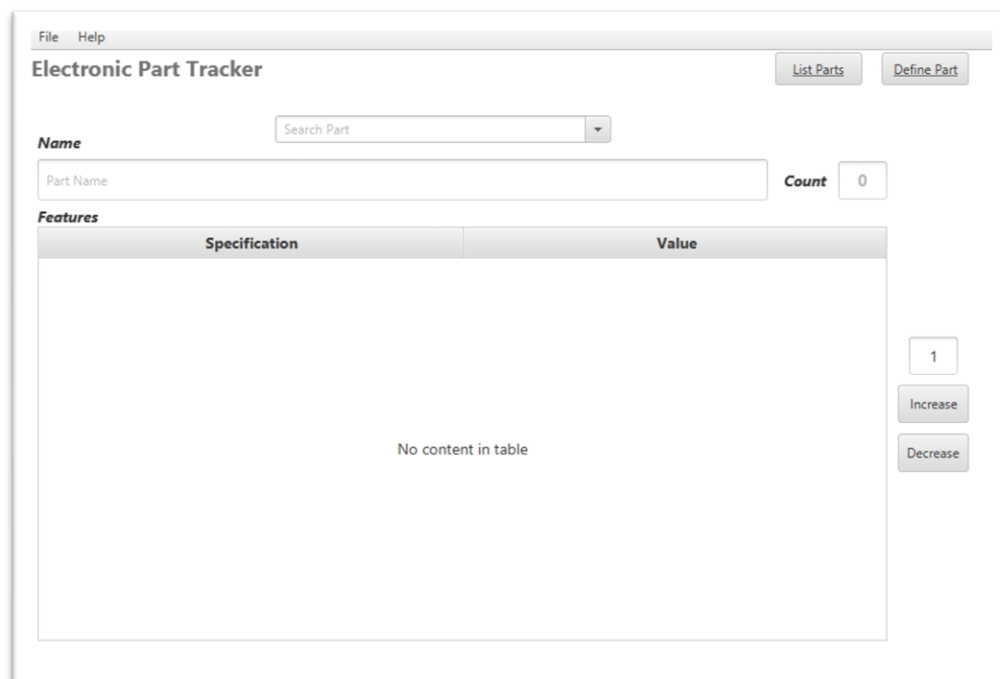


Figure 16. MainPage View

Main page designed for quick and easy display of selected part and increase/decrease selected part's count.

The desired part can be searched from ComboBox. This area allows the user to filter parts via input.

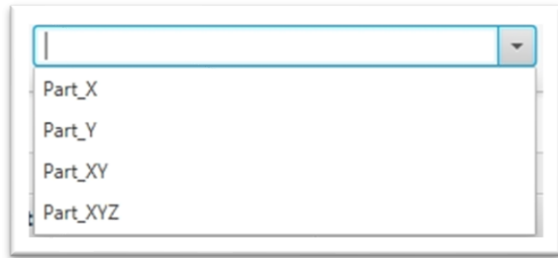
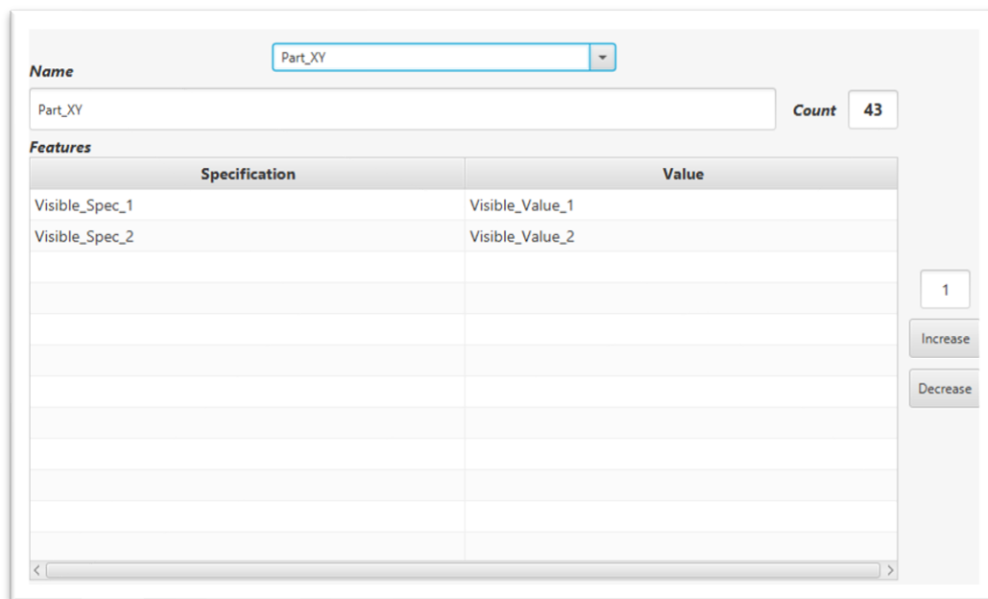


Figure 17. Combobox with all parts



Figure 18. Combobox with filtered parts



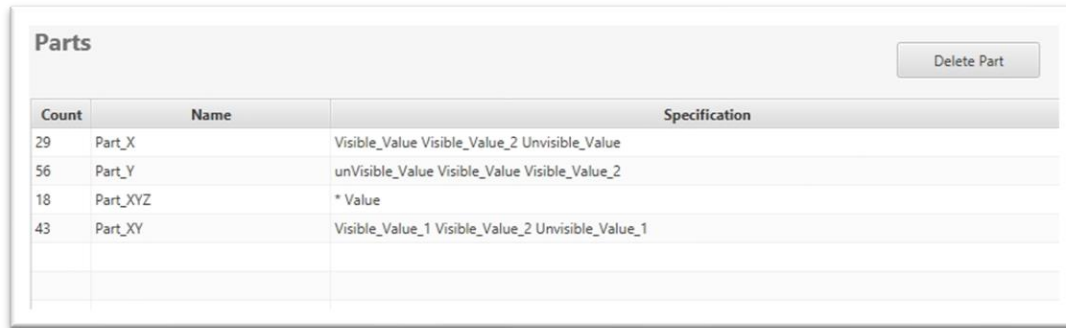
The part's only "visible" specifications and these values shown in the tableView.

Figure 19. Part\_XY's only visible features shown in the tableView

This feature prevents the user from a crowded screen by displaying only important features.

## List Part Page





Parts		
Count	Name	Specification
29	Part_X	Visible_Value Visible_Value_2 Unvisible_Value
56	Part_Y	unVisible_Value Visible_Value Visible_Value_2
18	Part_XYZ	* Value
43	Part_XY	Visible_Value_1 Visible_Value_2 Unvisible_Value_1

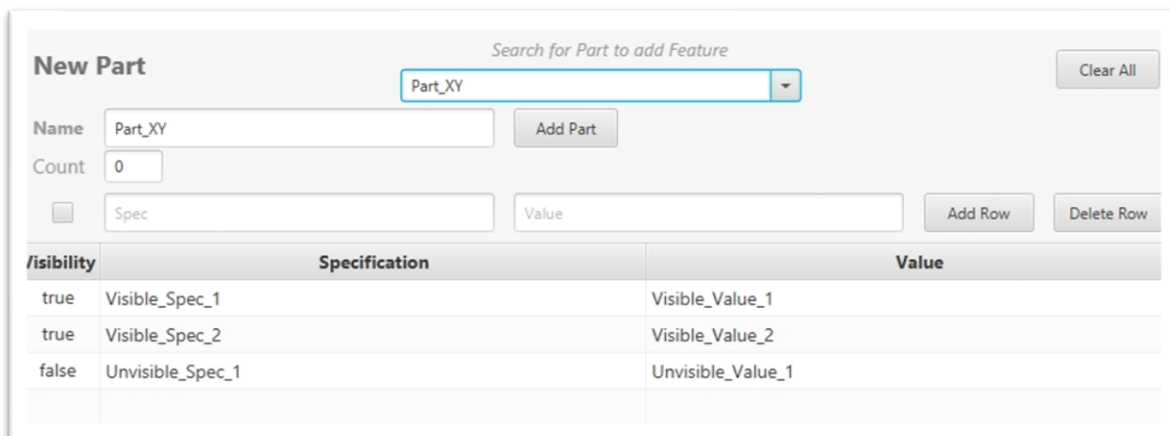
Figure 20. ListPart page view

All parts in the database shown in the ListPart page with their visible and invisible specifications together.

The defined part can be deleted from this page by selecting the desired part on the table.

The design that I planned to do on this page would create a separate tableView Column for each specification, and the part corresponding to this specification column would write values in the cell if there is that specification, if there is no specification, the cell would be left blank. But after 3-4 days of effort, I could not set up the system successfully. At this stage of the project, the design was renewed in order not to exceed the allowed time and I switched to the model of writing all values side by side.

## Define Part Page



**New Part**

Search for Part to add Feature

Part\_XY

Clear All

Name: Part\_XY Add Part

Count: 0

☐ Spec: Value: Add Row Delete Row

Visibility	Specification	Value
true	Visible_Spec_1	Visible_Value_1
true	Visible_Spec_2	Visible_Value_2
false	Unvisible_Spec_1	Unvisible_Value_1

Figure 21. DefinePart page view

- ❖ Parts can be defined and modified on the DefinePart page.
- ❖ Users can add a new part with its specification and value or edit already defined part by selecting a part from comboBox. This search box also has an input filter.
- ❖ Specifications and values can be edited by double-clicking the desired specification or value on the table.

- ❖ Delete Row button deletes selected specification and value.
- ❖ Clear All button sets all page elements to default.

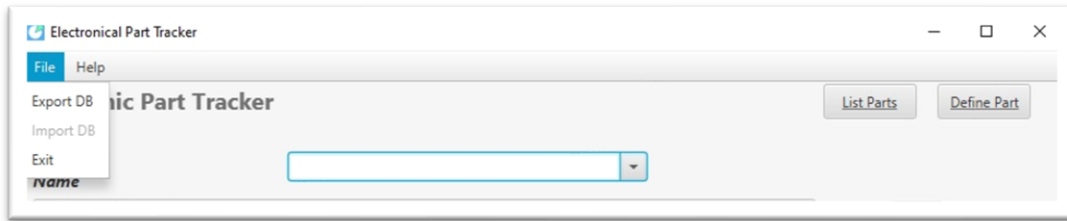


Figure 22. MainPage Menubar & File Menu content

All parts in the database can be exported by selecting a path with FileChooser.

### 3.4 Database

Leda uses SQLite as a database management system. “LedaSQLite.db” file is embedded into the installer and will be placed in the install directory.

“.db” file must be in the same directory as .exe because of connStr searches for .db in application execution location.

```
String connStr = "jdbc:sqlite:LedaSQLite.db";
```

Figure 23. .db file path string

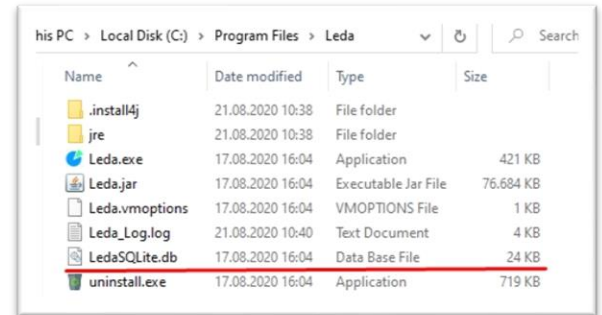


Figure 24. .db file in installation file

DB has two tables, tbl\_part & tbl\_relation:

- ❖ **tbl\_part** has 3 columns: “part\_id”, “name”, “count”
- ❖ **tbl\_relation** has 4 columns: “part\_id”, “visibility”, “spec”, “value”

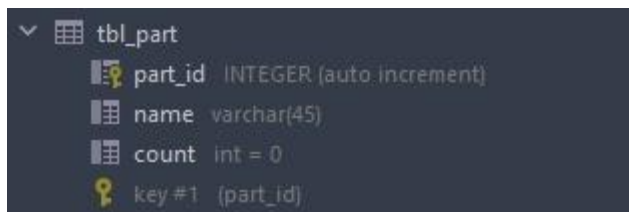


Figure 25. DB tbl\_part table

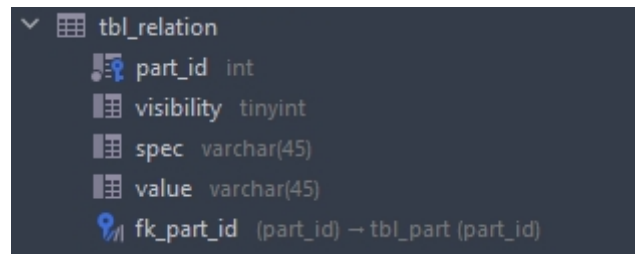


Figure 26. DB tbl\_relation table

tbl\_relation uses “part\_id” from tbl\_part as a foreign key. Additionally, part\_id on tbl\_relation has ON DELETE CASCADE, because when a part is deleted, its features must also be deleted from the database.

## Database ER Diagram

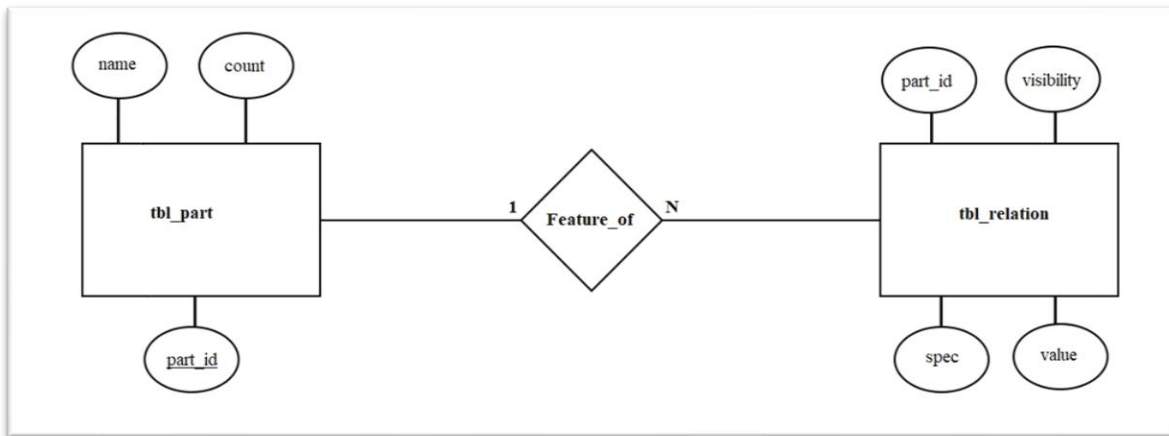
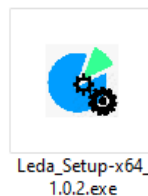


Figure 27. DB ER diagram

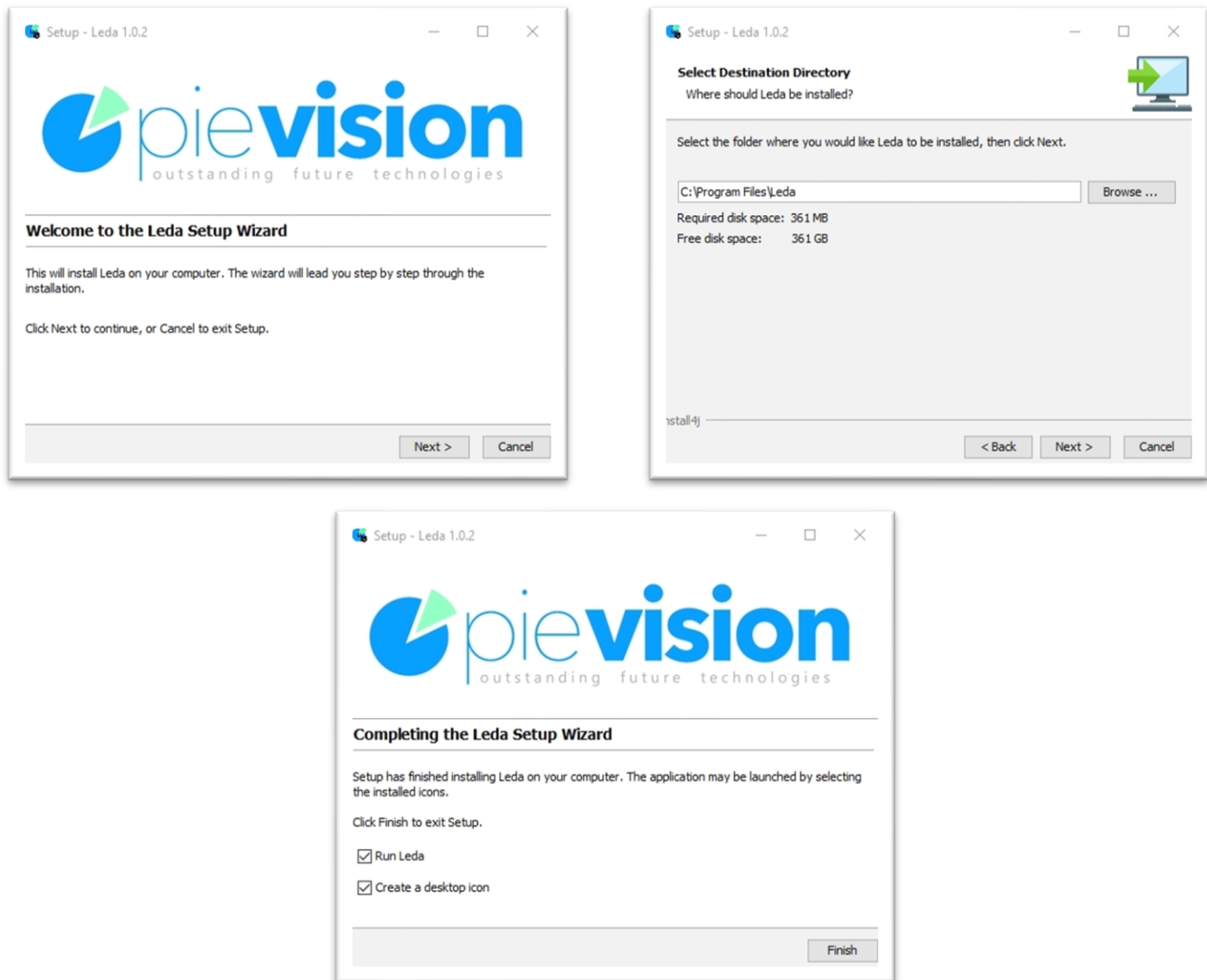
## Setup & Installation

Install4j is used for building the installer. Open-source project license requested from the developer.

```
License Key for Project Leda: L-M8-LEDA#XXXXXXXXXXXXXXXXXXXXX
(Should be requested from the company when necessary)
```



## Installer interfaces.



## SQL Scripts

### ❖ tbl\_part

```
CREATE TABLE tbl_part (  
    part_id INTEGER PRIMARY KEY AUTOINCREMENT,  
    name VARCHAR (45),  
    count INT DEFAULT 0  
);
```

### ❖ tbl\_relation

```
CREATE TABLE tbl_relation (  
    part_id INT NOT NULL CONSTRAINT fk_part_id  
        REFERENCES tbl_part  
        ON DELETE CASCADE,  
    visibility TINYINT,  
    spec VARCHAR (45),  
    value VARCHAR (45)  
);
```

### Ideas to Enhance the Application

- ❖ In the ListPart page, instead of keeping the specification column as a single column and writing all the values side by side, creating a separate column for each specification and writing the value to corresponding specification and name cells.
- ❖ Multi-User: By keeping the database on the server, access and modification of multiple users can be provided
- ❖ Create a category for parts, and the parts can be separated by category.