Helmes

Lõõtsa 6b
Tallinn - 11415
610-6100

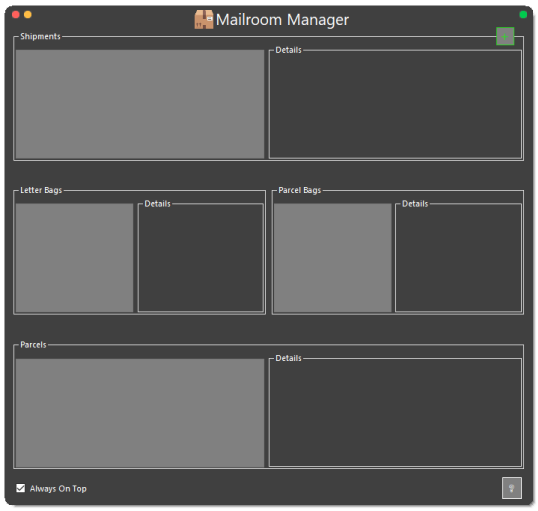
Mailroom Manager

August, 2022

Product Overview

Mailroom Manager is a complete solution for managing parcels, letters, mailbags, and shipments within your mailroom.

Interface



The Mailroom Manager interface is simple and user-friendly. There are four groups of boxes laid out in the flow in which your mailroom works.

Shipments, Bags, & Parcels

The Shipments box contains, on the left, a list of the shipments in your database. When you select one of these shipments, the details will appear on the left in the Details panel. The same goes for each type of mail bag, as well as parcels.

How To Use

When you first run the Mailroom Manager, it will have no items selected. From here, you can add a shipment right away with the green  button. Once you have a shipment within your list, you can select it to see all of its details appear. You can also  Deselect,  Delete, or  Edit it with the Blue, Red, and Yellow buttons that appear, respectively. When you have a shipment selected, you can then add a bag of either type to that shipment. Once again, there will be a

green  button in each of the panels for the types of bags, Letter Bags and Parcel Bags. Once you've got bags in your shipment, you can then see the details when selected, or  Delete the shipment if you wish. When a bag is selected, the same green  button will appear in the Parcels panel. You can then follow the same procedure to add a parcel to the bag in the shipment. Once added, you can also view it's details, or  Edit or  Delete it.

Buttons

Each of the panels will have its own selection of panels. Shipments and Parcels will have all of the buttons available. Bags will have all but edit, as the details from the bags are generated by the parcels they contain. Let's go over each of the buttons and some examples of how they work.

 Deselect

The Deselect button will remove the selection you have made in that panel, as well as any items below it. For example, if you press this button in the Letter Bag when one is selected, it will deselect the Letter Bag, as well as a parcel that is selected. Likewise, if a Shipment is deselected, any selected Bag and Parcel will also be deselected, but if a Parcel is deselected, only the parcel will be deselected.

 Delete

The Delete button will open a dialog window with details regarding the selected item. If you click the Delete button in this window, you'll be prompted once more to ensure you want to delete the item. If you select yes, the item will be deleted and the list will be updated.

 Edit

The Edit button will open a dialog window with details regarding the selected item. You may edit the details of the item as needed, and if you click the Edit button, the item will be updated in the database.

 Add

The Add button will open a dialog which allows you to add an item to the corresponding object, be it a Shipment, Bag, or Parcel. If any details are entered incorrectly, the dialog will let you know what's wrong and how to fix it.

Technical Details & Instructions

How to get the API and App set up and working together.

The Mailroom Manager is a C#.NET solution which contains two projects inside.

The Mailroom API project contains the API functionality which uses the C# .NET Core 6 Web API framework. To run the Web API, ensure you have the latest version of .NET Core installed, as well as Microsoft SQL Server Express (MSSQLe) and the latest version of Visual Studio 2022. To start, open the Mailroom API project in Visual Studio, navigate to the `appsettings.json` file, and edit the `DefaultConnection` string to match your Database's connection string. Next, you will need to build the database using Entity Framework. To do this, open the Package Manager console (*View -> Other Windows -> Package Manager Console*) and navigate into the Mailroom API directory:

```
cd "Mailroom API"
```

Then, you can use the EF commands to migrate and build the database structure:

```
dotnet ef migrations add CreateInitial
```

This will create the migration in the Migrations folder within the project. You can then update the database to build this structure:

```
dotnet ef database update
```

This will update the database to the latest migration. You can also specify a migration if one already exists.

Once the database structure is in place, you can run (without debugging) the Mailroom API. When the API is up and running, take note of its IP address and port, preferably the HTTPS port.

The Mailroom App project contains the front-end Windows application which uses C# .NET Framework 4.8 WinForms. To run the application, ensure you have your API IP address handy in case it needs it on the first launch, then simply run the program, either from Visual Studio, or from the "*Mailroom Manager\Mailroom App\bin\Debug\Mailroom App.exe*" file inside the solution's root directory. If the app is unable to reach the API server, it will throw an error and then prompt you to update the API IP address. When you've entered it in and hit save, restart the application.

When the application is able to contact the API, it will populate the Shipments list if there are any already, and the green  button will appear inside the Shipments box. You can then use the above instructions to use the program, but simply put, add a shipment to the database by clicking the green + button, select the shipment, add bags using their green + buttons, select one, and add parcels to those. Once you've populated your database with some data, you can then try out editing with the  button or deleting with the  button. If you use an external SQL viewer such as Microsoft SQL Server Management Studio or HeidiSQL, you can see these changes made within the database in real-time (or at least when you refresh your database viewer!)

The application will make API calls whenever it sends/needs data. If you select a shipment, the application calls the API to get the bags within that shipment, then populates the lists with those bags. When you select a bag, it calls the API to populate the Parcel list with the parcels in that bag, and so on.

The same goes for adding, editing, and deleting. When the dialog pops up, API is called to get (GET) any info about that item that is needed to populate the dialog window. When you confirm the dialog, the API is called to add (POST), edit (PUT), or delete (DELETE) the given item(s).

With the latest Visual Studio, Swagger UI is a default option within the Web API framework, so you can view the API's routes/schemas, or make calls directly using that if you wish. It follows a standard CRUD format, and the application handles the majority of the complex logic. The API

also has format/error handling to ensure that data sent is in the correct format, in case the front-end is unable to verify the format, though it should in every case.

The code is heavily commented to allow ease of editing, and I've tried to take several different approaches to solving many of the same kinds of tasks, to demonstrate my coding adaptability and knowledge of the language. Though, if given the chance and more time, I would love to unify how the code works, as well as refactor much of it to be more reusable. However, given that this is a test task, I figured it would be smart to show my work at every step, and show several different ways to solve the same problem.

There is also extensive error handling, (most, if I got them all!) errors should be caught and handled with a MessageBox to prevent crashing, and give the user some information about what happened and how to fix it, if possible. If you encounter any errors, however, do let me know! I'd love to see what happened and tinker with this project some more, as it's been a lot of fun and great learning too!

Thank you for the opportunity and I can't wait to see what you think!

With the utmost of thanks and the kindest regards,

-Alex Kotanko

alex.kotanko1@gmail.com