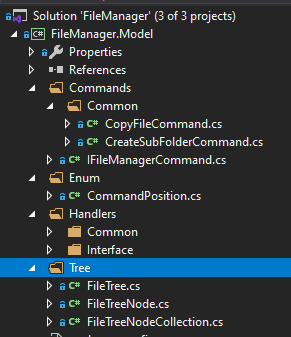
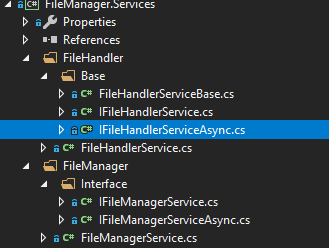
**The solution consists of 3 key projects:**

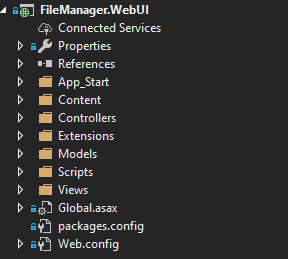
**1. Model** - is a set of basic interfaces and classes that are used in the application. There should be a description of the main types that will be used. Contains object models for interacting with services, handlers, and the file tree.



**2. Services** - a library that stores a description of all application services that contain this or that business logic, as well as the level of the data access layer



**3. WebUI**. - A web application, which is a site with a file manager interface. It Allows you to manage files and folders on the server through a web interface.



**Key components:**

**1. FileManagerService**

  
It is a type that implements a file processing interface. Contains methods that allow you to perform certain operations with the file. Interaction with the service is carried out through the implementation of dependencies, the level of interaction through the interface.

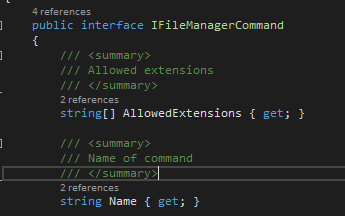
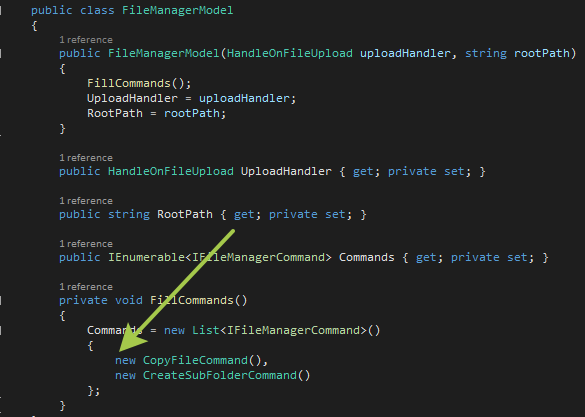
**2**. **FileHandlerService**.  
  


Allows you to perform the necessary operations when performing certain events: such as "Download" or "Upload". The level of interaction with the service through the interface

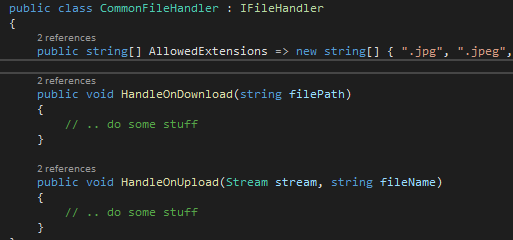
3. **IFileManagerCommand**  
  
  
  
The interface is a description of the command that will be displayed in the context menu or tool bar of the file manager. The command settings for each file type are manually configured in the "**AllowedExtensions**" property, where all file extensions for which this command will be displayed should be listed, as well as some other descriptions.

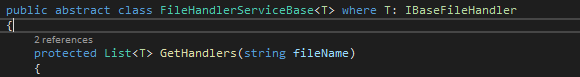
The design of the application is very flexible and scalable. It is implemented with all the key requirements of the technical specifications and the principles of **SOLID**.

**In Detail:**

1. In order for a developer to add a new command to the file manager, all that is needed is to add a new implementation of the "**IFileManagerCommand**" interface and add a command instance when loading the model into the view.  
  
  
  


The visibility of the command is also configured in the "**AllowedExtensions**" property as indicated earlier. And the web application, using JavaScript, will automatically hide or show the command for the selected file, if its extension is specified in the command properties

2. In order to add your event handler for files with a specific extension, you just need to add the implementation of the "**IFileHandler**" interface. The program will automatically apply a handler for these files.  
  
(Example):  


3. In order to write your own mechanism for calling the event handler for files, you just need to add the implementation of the abstract class "**FileHandlerServiceBase**" and inherit it in the service for processing events.  
  
  
  
  
**How the database will be created, and the application will be improved for permanent storage:**

**1.** To interact with the database, you we can use **Entity Framework**. An implementation of the **DBContext** class is required, where the connection string from the web configuration file will be transmitted. The context must be hidden and encapsulated in a service that will interact with the database. Then, the service must be registered in the **dependency injection** container in order to interact with it through the interface.

**2**. Next, we need to add support for migrations, so that with small efforts you can make certain changes to the database structure. This can be implemented using various libraries (such as a **fluent migrator** for example). Next, the first basic migration will be necessary, which will create all the necessary tables and the relationships between them. All relations and table structures should be described in the same way in the "**Entity**" classes, and in the **Profiles** with which the **Entity Framework** will interact.

**3**. The database structure for tree-based storage of information about files and folders should certainly contain a relation to the parent element, a unique identifier, and the relative path of the file (folder) on the server.