UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola Tècnica Superior d'Enginyeria de Telecomunicació de Barcelona

telecos BCN

Work-flux optimization for the use of GRASP algorithm in quasi-real-time

Master Thesis
submitted to the Faculty of the
Escola Tècnica d'Enginyeria de Telecomunicació de Barcelona
Universitat Politècnica de Catalunya
by

Mireia López Barrón

In partial fulfillment
of the requirements for the master in
(Write the name of your Master) **ENGINEERING**

Advisor: name of the advisor
Barcelona, Date XXXXX

telecos BCN

Contents

List of Figures

Listings

List of Tables

Revision history and approval record

Revision	Date	Purpose
0	10/12/2025	Document creation
1	dd/mm/yyyy	Document revision

DOCUMENT DISTRIBUTION LIST

Name	e-mail
[Student name]	
[Project Supervisor 1]	
[Project Supervisor 2]	

Written by:		Reviewed and approved by:	
Date	dd/mm/yyyy	Date	dd/mm/yyyy
Name	Xxxxxxx yyyyyyy	Name	Zzzzzzz Wwwwwww
Position	Project Author	Position	Project Supervisor

Abstract

Every copy of the thesis must have an abstract. An abstract must provide a concise summary of the thesis. In style, the abstract should be a miniature version of the thesis: short introduction, a summary of the results, conclusions or main arguments presented in the thesis. The abstract may not exceed 150 words for a Degree's thesis.

1 Introduction

Recently, the need to analyze and monitor atmospheric aerosols has grown significantly, as their negative impact has grown significantly in different sectors: human health, climatology and air transport. GRASP (Generalized Retrieval of Atmosphere and Surface Properties), which was introduced by Dubovik, is the first unified algorithm used for obtaining atmospheric properties from different remote sensing observations including satellite, ground-based and airborne measurements, both passive and active, of atmospheric radiation and their combinations[25].

The main objective of this work is to develop a software that automates the full GRASP execution process, which includes:

1. Download photometer data from the Aeronet web site
2. Download Lidar (Light Detection and Ranging) data from the ACTRIS-EARLINET web site
3. Combine the data downloaded from both web sites
4. Run GRASP
5. Plot the obtained results

1.1 Gantt Diagram

The time organization of the project is shown in the Gantt diagram of Figure 1 representing a time lapse of one semester. It is important to note that even there are different groups of tasks for implementation and testing, tests have been executed during the implementation phase to ensure the correct functionality of the different steps.

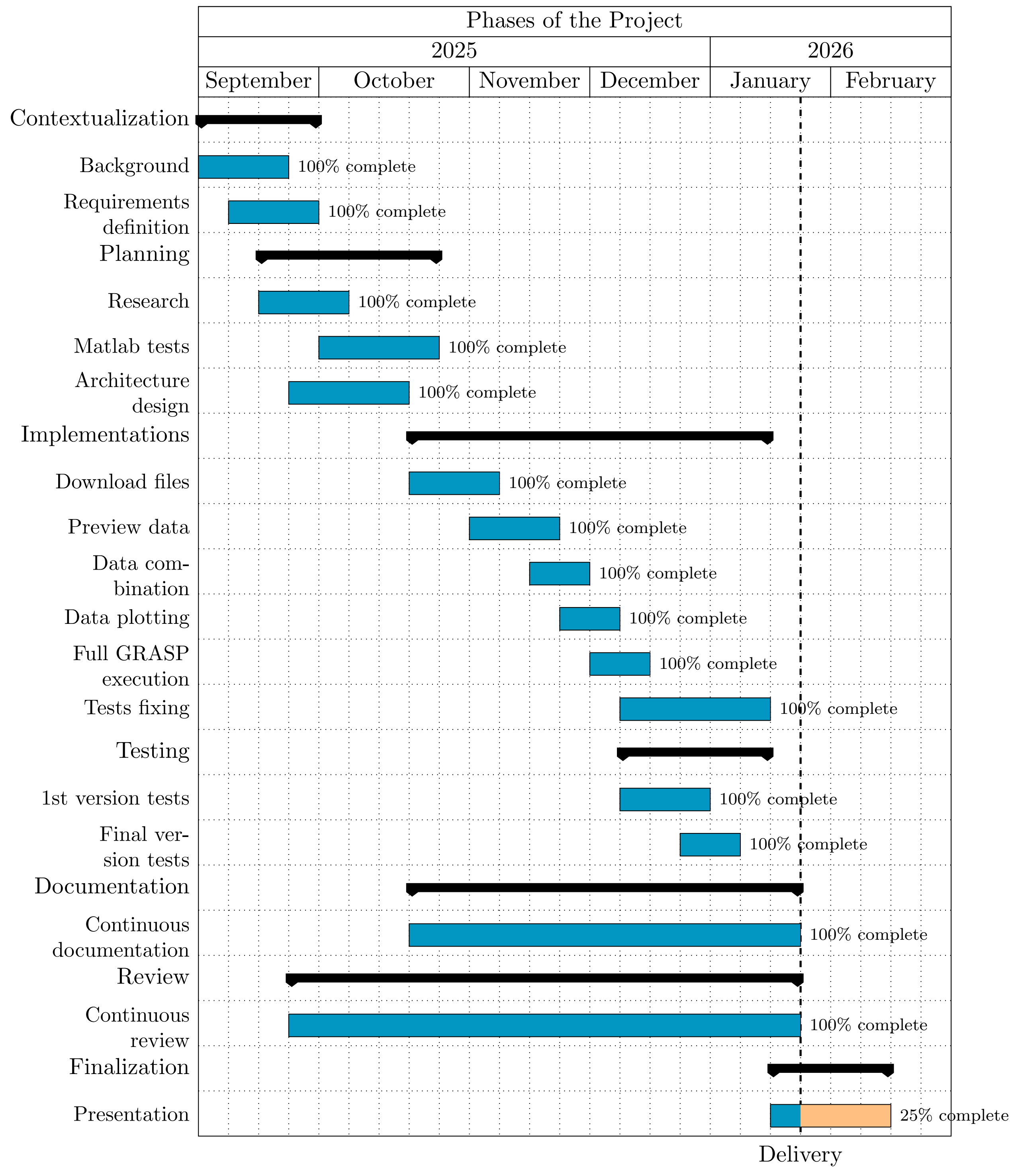


Figure 1: Gantt diagram of the project

1.2 Deviation from the initial plan

2 State of the art

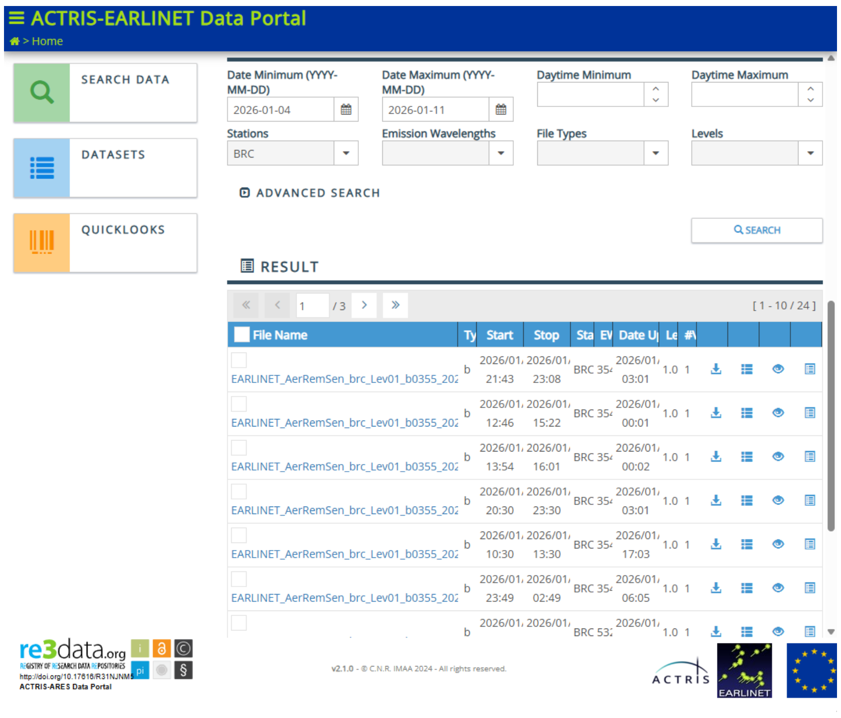
Universitat Politècnica de Catalunya (UPC) research department for Remote Sensing has been using the GRASP algorithm to process and obtain information from data measured by a lidar and a photometer, in order to study the presence of aerosols in the atmosphere. In order to do that, different steps need to be executed manually, without a graphical interface or centralized application, which increases the risk of errors, reduces reproducibility, and demands significant user intervention.

Other students have previously developed Matlab scripts in order to automate the preprocessing stage of the workflow used by the department. However, this solution is not optimal as it requires the user to have a basic knowledge of Matlab and all the different web sites and files used.

Every time someone wants to use the GRASP algorithm, it need to follow the steps described in the different points of this section.

2.1 Download data

First website to download data is Actris-Earlinet. This interface is used to download the data measured by the Lidar, and can be easily downloaded by configuring different parameters like start and finish date, and selecting the desired station, as it can be seen in Figure 2. The different files that can be downloaded will appear listed under the configuration parameters list.



Date Minimum (YYYY-MM-DD)	2026-01-04
Date Maximum (YYYY-MM-DD)	2026-01-11
Daytime Minimum	
Daytime Maximum	
Stations	BRC
Emission Wavelengths	
File Types	
Levels	

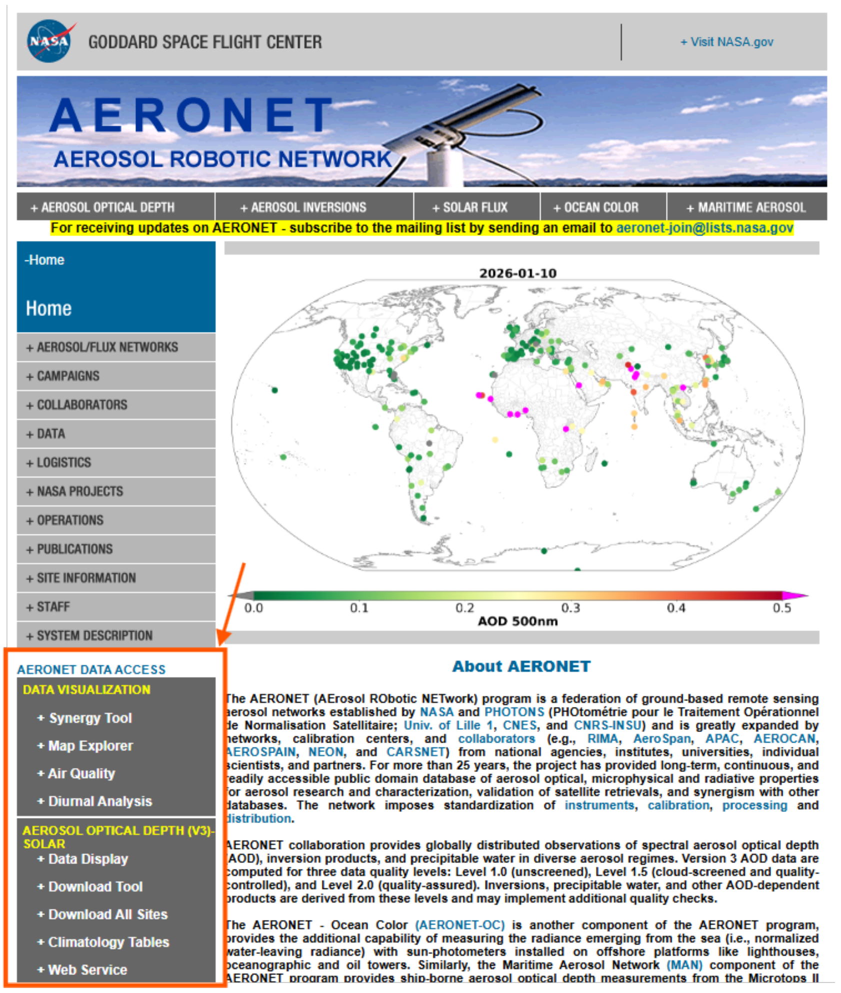
1 / 3 [1 - 10 / 24]

File Name	Ty	Start	Stop	Sta	EV	Date U	Le	#\				
EARLINET_AerRemSen_brc_Lev01_b0355_202	b	2026/01, 21:43	2026/01, 23:08	BRC 354		2026/01, 03:01	1.0	1	⬇	☰	👁	▤
EARLINET_AerRemSen_brc_Lev01_b0355_202	b	2026/01, 12:46	2026/01, 15:22	BRC 354		2026/01, 00:01	1.0	1	⬇	☰	👁	▤
EARLINET_AerRemSen_brc_Lev01_b0355_202	b	2026/01, 13:54	2026/01, 16:01	BRC 354		2026/01, 00:02	1.0	1	⬇	☰	👁	▤
EARLINET_AerRemSen_brc_Lev01_b0355_202	b	2026/01, 20:30	2026/01, 23:30	BRC 354		2026/01, 03:01	1.0	1	⬇	☰	👁	▤
EARLINET_AerRemSen_brc_Lev01_b0355_202	b	2026/01, 10:30	2026/01, 13:30	BRC 354		2026/01, 17:03	1.0	1	⬇	☰	👁	▤
EARLINET_AerRemSen_brc_Lev01_b0355_202	b	2026/01, 23:49	2026/01, 02:49	BRC 354		2026/01, 06:05	1.0	1	⬇	☰	👁	▤
	b	2026/01,	2026/01,	BRC 532		2026/01,	1.0	1	⬇	☰	👁	▤

v2.1.0 - © C.N.R. IMAA 2024 - All rights reserved.

Figure 2: Earlinet website for downloading data

Aeronet website offers more information and options to download data, so it needs to be done in different steps. First, the user needs to locate the Aeronet Data Access page, as it can be seen in Figure 3, and select a data option.



About AERONET

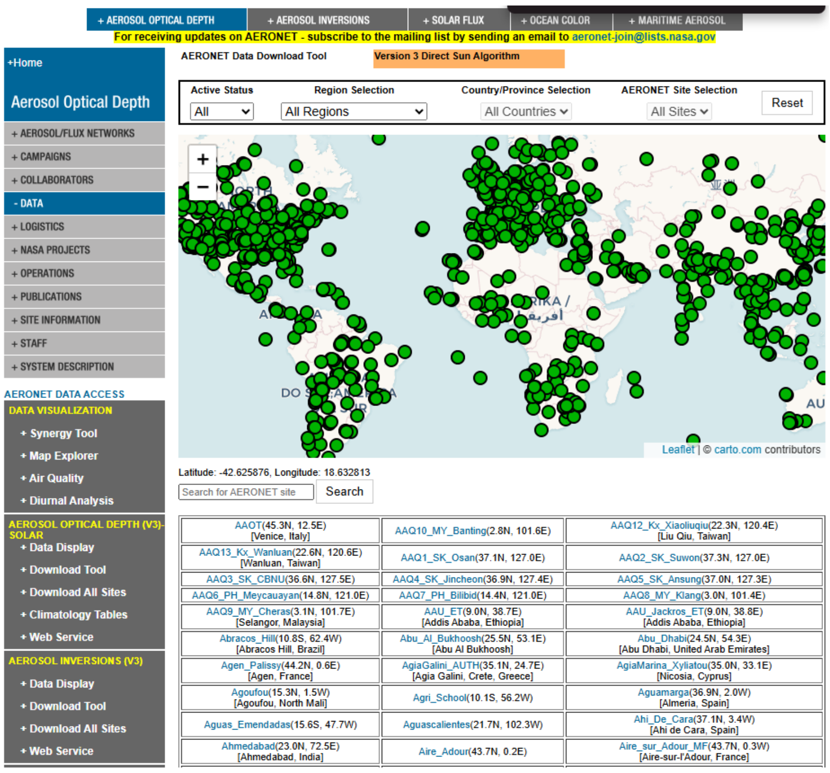
The AERONET (AErosol RObotic NETwork) program is a federation of ground-based remote sensing aerosol networks established by NASA and PHOTONS (PHOtométrie pour le Traitement Opérationnel de Normalisation Satellitaire; Univ. of Lille 1, CNES, and CNRS-INSU) and is greatly expanded by networks, calibration centers, and collaborators (e.g., RIMA, AeroSpan, APAC, AEROCAN, AEROSPAIN, NEON, and CARSNET) from national agencies, institutes, universities, individual scientists, and partners. For more than 25 years, the project has provided long-term, continuous, and readily accessible public domain database of aerosol optical, microphysical and radiative properties for aerosol research and characterization, validation of satellite retrievals, and synergism with other databases. The network imposes standardization of instruments, calibration, processing and distribution.

AERONET collaboration provides globally distributed observations of spectral aerosol optical depth (AOD), inversion products, and precipitable water in diverse aerosol regimes. Version 3 AOD data are computed for three data quality levels: Level 1.0 (unscreened), Level 1.5 (cloud-screened and quality-controlled), and Level 2.0 (quality-assured). Inversions, precipitable water, and other AOD-dependent products are derived from these levels and may implement additional quality checks.

The AERONET - Ocean Color (AERONET-OC) is another component of the AERONET program, provides the additional capability of measuring the radiance emerging from the sea (i.e., normalized water-leaving radiance) with sun-photometers installed on offshore platforms like lighthouses, oceanographic and oil towers. Similarly, the Maritime Aerosol Network (MAN) component of the AERONET program provides ship-borne aerosol optical depth measurements from the Microtops II

Figure 3: Aeronet website for downloading data

For each type of data, the user needs to select the desired site from the shown list, as it can be seen in Figure 4.



For receiving updates on AERONET - subscribe to the mailing list by sending an email to aeronet-join@lists.nasa.gov

AERONET Data Download Tool — Version 3 Direct Sun Algorithm

Active Status: All — Region Selection: All Regions — Country/Province Selection: All Countries — AERONET Site Selection: All Sites

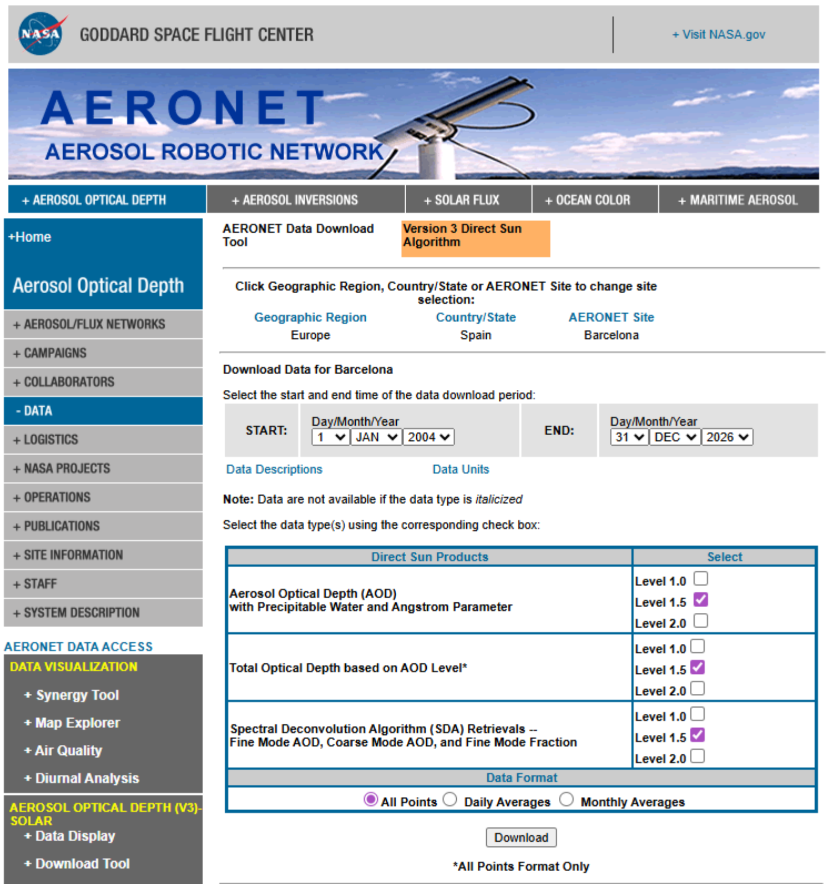
Latitude: -42.625876, Longitude: 18.632813

Search for AERONET site

AAOT(45.3N, 12.5E) [Venice, Italy]	AAQ10_MY_Banting(2.8N, 101.6E)	AAQ12_Kx_Xiaoliuqiu(22.3N, 120.4E) [Liu Qiu, Taiwan]
AAQ13_Kx_Wanluan(22.6N, 120.6E) [Wanluan, Taiwan]	AAQ1_SK_Osan(37.1N, 127.0E)	AAQ2_SK_Suwon(37.3N, 127.0E)
AAQ3_SK_CBNU(36.6N, 127.5E)	AAQ4_SK_Jincheon(36.9N, 127.4E)	AAQ5_SK_Ansung(37.0N, 127.3E)
AAQ6_PH_Meycauayan(14.8N, 121.0E)	AAQ7_PH_Bilibid(14.4N, 121.0E)	AAQ8_MY_Klang(3.0N, 101.4E)
AAQ9_MY_Cheras(3.1N, 101.7E) [Selangor, Malaysia]	AAU_ET(9.0N, 38.7E) [Addis Ababa, Ethiopia]	AAU_Jackros_ET(9.0N, 38.8E) [Addis Ababa, Ethiopia]
Abracos_Hill(10.8S, 62.4W) [Abracos Hill, Brazil]	Abu_Al_Bukhoosh(25.5N, 53.1E) [Abu Al Bukhoosh]	Abu_Dhabi(24.5N, 54.3E) [Abu Dhabi, United Arab Emirates]
Agen_Palissy(44.2N, 0.6E) [Agen, France]	AgiaGalini_AUTH(35.1N, 24.7E) [Agia Galini, Crete, Greece]	AgiaMarina_Xyliatou(35.0N, 33.1E) [Nicosia, Cyprus]
Agoufou(15.3N, 1.5W) [Agoufou, North Mali]	Agri_School(10.1S, 56.2W)	Aguamarga(36.9N, 2.0W) [Almeria, Spain]
Aguas_Emendadas(15.6S, 47.7W)	Aguascalientes(21.7N, 102.3W)	Ahi_De_Cara(37.1N, 3.4W) [Ahi de Cara, Spain]
Ahmedabad(23.0N, 72.5E) [Ahmedabad, India]	Aire_Adour(43.7N, 0.2E)	Aire_sur_Adour_MF(43.7N, 0.3W) [Aire-sur-l'Adour, France]

Figure 4: Aeronet site selection

Once the site is selected, the user needs to choose the different data optinons to be downloaded for both, AOD and Inversions, as it can be seen in Figure 5 and Figure 6, respectively.



AERONET Data Download Tool

Version 3 Direct Sun Algorithm

Click Geographic Region, Country/State or AERONET Site to change site selection:

Geographic Region	Country/State	AERONET Site
Europe	Spain	Barcelona

Download Data for Barcelona

Select the start and end time of the data download period:

START: Day/Month/Year 1 JAN 2004   END: Day/Month/Year 31 DEC 2026

Data Descriptions   Data Units

Note: Data are not available if the data type is *italicized*

Select the data type(s) using the corresponding check box:

Direct Sun Products	Select
Aerosol Optical Depth (AOD) with Precipitable Water and Angstrom Parameter	Level 1.0 ☐ Level 1.5 ☑ Level 2.0 ☐
Total Optical Depth based on AOD Level*	Level 1.0 ☐ Level 1.5 ☑ Level 2.0 ☐
Spectral Deconvolution Algorithm (SDA) Retrievals -- Fine Mode AOD, Coarse Mode AOD, and Fine Mode Fraction	Level 1.0 ☐ Level 1.5 ☑ Level 2.0 ☐

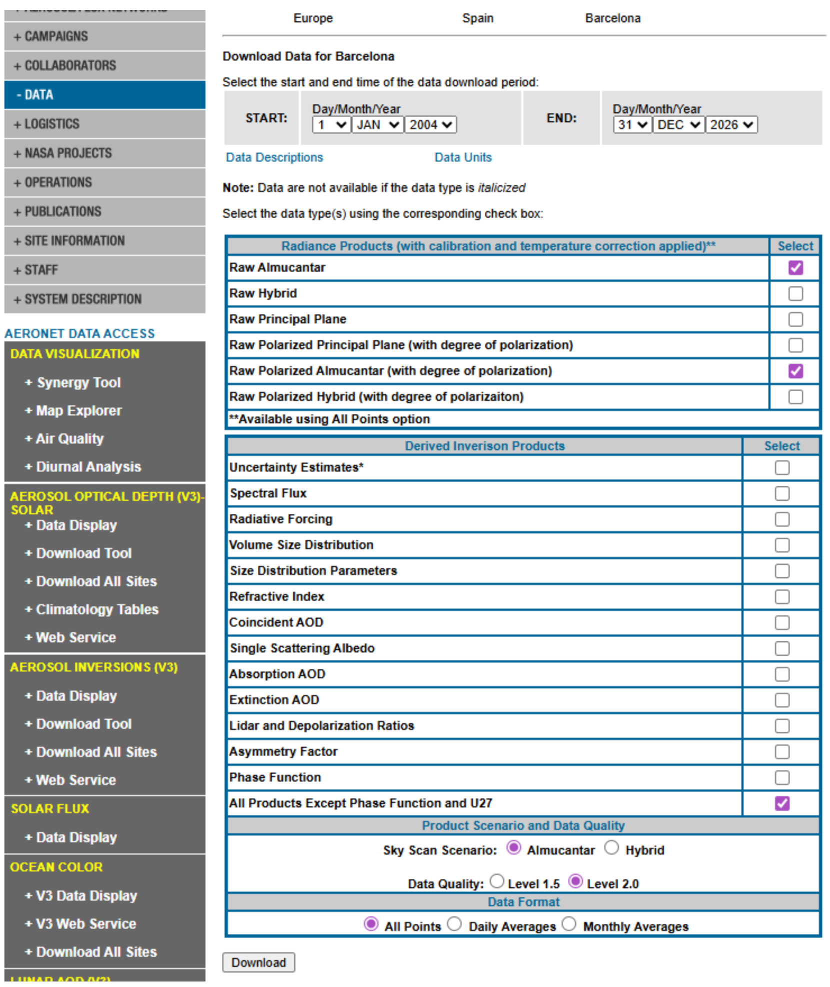
Data Format

[✓ All Points] Daily Averages Monthly Averages

Download

*All Points Format Only

Figure 5: Aerosol Optical Depth download UI



Download Data for Barcelona

Select the start and end time of the data download period:

START: Day/Month/Year 1 JAN 2004 END: Day/Month/Year 31 DEC 2026

Note: Data are not available if the data type is *italicized*

Select the data type(s) using the corresponding check box:

Radiance Products (with calibration and temperature correction applied)**	Select
Raw Almucantar	✓
Raw Hybrid	
Raw Principal Plane	
Raw Polarized Principal Plane (with degree of polarization)	
Raw Polarized Almucantar (with degree of polarization)	✓
Raw Polarized Hybrid (with degree of polarizaiton)	
**Available using All Points option	

Derived Inversion Products	Select
Uncertainty Estimates*	
Spectral Flux	
Radiative Forcing	
Volume Size Distribution	
Size Distribution Parameters	
Refractive Index	
Coincident AOD	
Single Scattering Albedo	
Absorption AOD	
Extinction AOD	
Lidar and Depolarization Ratios	
Asymmetry Factor	
Phase Function	
All Products Except Phase Function and U27	✓

Product Scenario and Data Quality

Sky Scan Scenario: [✓ Almucantar] Hybrid

Data Quality: Level 1.5 [✓ Level 2.0]

Data Format

[✓ All Points] Daily Averages Monthly Averages

Figure 6: Aerosol Inversions download UI

Once all data has been downloaded, it needs to be located in a file accessible for the Matlab scripts.

2.2 Select configuration and execute Matlab scripts

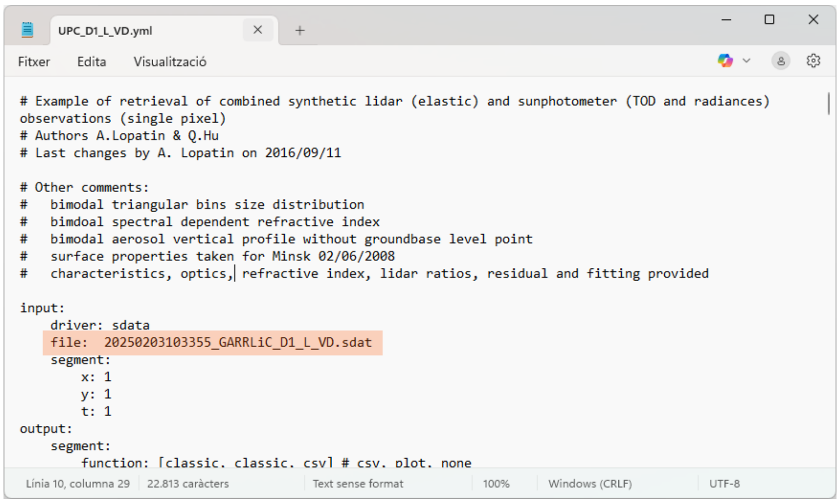
It exists different configurations, depending ont the available data downloaded:

- D1L: Only photometer and Lidar data are available
- D1P_L: It is also available polarization data in photometer measurements
- D1L_VD: It is also available depolarization data in Lidar measurements
- D1P_L_VD: All data is available

Knowing the available files and the selected dates, in order to run the Matlab Scripts succesfully, user should choose manually the configuration and execute the corresponding script, since it exists one script per each configuration possibility.

2.3 Modify manually configuration file for GRASP algorithm

After executing the Matlab scripts, a data file is created named ¡measureID¿_GARRLIC_¡configuration¿.s which will be used as input for the GRASP algorithm. But it is also needed a configuration file, that indicates different parameters to the algorithm, being one of them the name of the data file to be used. This file is names UPC_¡configuration¿.yml.



```
# Example of retrieval of combined synthetic lidar (elastic) and sunphotometer (TOD and radiances)
observations (single pixel)
# Authors A.Lopatin & Q.Hu
# Last changes by A. Lopatin on 2016/09/11

# Other comments:
#   bimodal triangular bins size distribution
#   bimdoal spectral dependent refractive index
#   bimodal aerosol vertical profile without groundbase level point
#   surface properties taken for Minsk 02/06/2008
#   characteristics, optics, refractive index, lidar ratios, residual and fitting provided

input:
    driver: sdata
    file:  20250203103355_GARRLiC_D1_L_VD.sdat
    segment:
        x: 1
        y: 1
        t: 1
output:
    segment:
        function: [classic, classic, csv] # csv, plot, none
```

Figure 7: Example of GRASP configuration file

In Figure 7, it can be seen the example of a configuration file when the configuration chosen is D1L_VD and the MeasureID is 20250203103355, corresponding to the 3rd of February at 10:33:55.

2.4 Execute GRASP algorithm in CALCULA

When all required files have been created and correctly configured, the GRASP algorithm can be executed in CALCULA. The user should upload the files to the virtual enviroment and execute the following command lines in the command prompt:

```
cd <directory to files>
/opt/bin/grasp-1.1.5/grasp UPC<config>.yml
```

Once the algorithm has been executed correctly, files named UPC_¡configuration¿_screen.txt and UPC_¡configuration¿_out.txt will be created in same directory.

2.5 Execute Matlab script for postprocessing and results analysis

After the execution, the user needs to download the output files from CALCULA and execute the Matlab post processing script in order to obtain the results and the corresponding figures to analyze.

2.6 Need of optimization

After analyzing the process, it can be seen that it is not optimal as it requires the user to access different websites in order to download the needed files, analyze the available data in order to select the correct configuration and execute scripts manually, without a graphical interface or centralized application, which increases the risk of errors, reduces reproducibility, and demands significant user intervention.

Working with different Operative Systems (mostly Windows and CALCULA) also increases the time needed to execute the process.

The main objective of this project is to optimize all of the steps of this sequence so it is reduced the time consumption and the error introduction possibilities by the user.

3 Methodology

In order to create a new application , it is necessary to follow the following steps, to create a strong foundation.

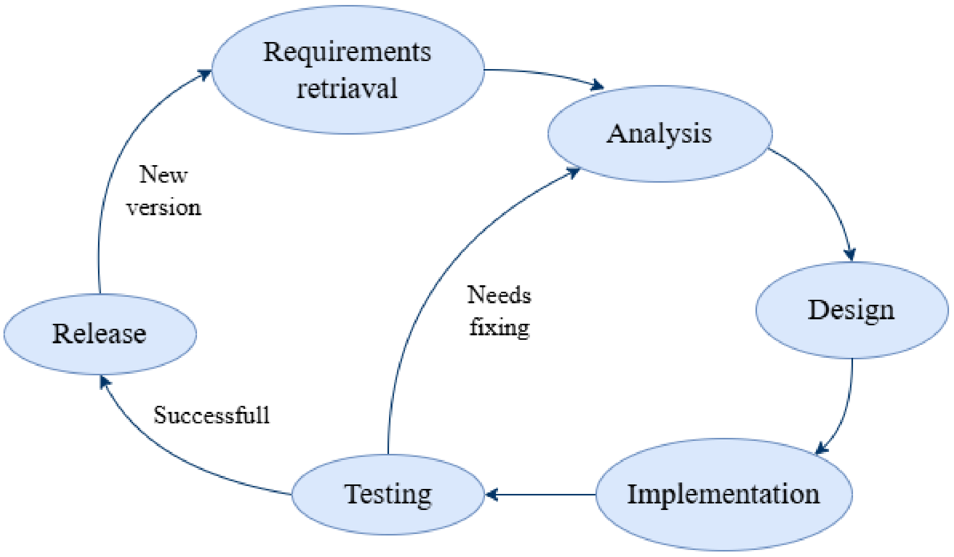


Figure 8: Project development cycle

This diagram shows the ideal development process for a project. Since the application developed is the first version, requierements retrieval should only happen once, and the project should finish with the Release phase, once all tests have been passed correctly.

For the development of the application, it has been decided to use an hybrid structure, using Object Oriented Programming (OOP) with C# and MATLAB scripting.

The main advantages of using C# with OOP are that it allows the code to be reusable, organized and easy to maintain [7]. In this project, it has been used for the development of the Graphical User Interface and the controllers for both, web services and MATLAB script management.

On the other hand, MATLAB provides a specialized environment for numerical calculations, simulations, and complex data analysis[8]. Working directly with matrixes and arrays makes the data processing and visualitzation easier and more efficient.

4 List of requirements

This first step is crucial for the correct result. It is needed to define a clear and well defined list of requirements. This defines the main scope of the project, the different futures to be implemented and prevents from building something that is not desired. This project, need to be able to fulfill the following requirements:

1. Application can be executed in CALCULA operating system (Linux)
2. Download measurements from the internet
   (a) Download measurements from a defined date range
   (b) Download measurements from a defined location
   (c) Download measurements from the Actris-Earlinet Data Portal [5]
      - Download ELPP products
      - Download Optical products
   (d) Download measurements from the Aeronet web [6]
      - Download Raw Almucantar Sky Scan Radiance measurements
      - Download Raw Hybrid Sky Scan Radiance measurements
      - Download Raw Principal Plane Sky Scan Radiance measurements
      - Download Raw Polarized Principal Plane Sky Scan Radiance and Degree of Polarization measurements
      - Download Raw Polarized Almucantar Sky Scan Radiance and Degree of Polarization measurements
      - Download Raw Polarized Hybrid Sky Scan Radiance and Degree of Polarization measurements
3. Filter Earlinet measurement files depending on the the data type
   (a) 002, 008
   (b) 007
4. Execute pre-execution:
   (a) Read data of all downloaded files that are used as inputs for grasp algorithm
   (b) Preview data in order to choose a correct value for minimum and maximum heights
   (c) Obtain available configurations for the selected measure ID depending on available data
   (d) Choose a configuration
   (e) Generate a .sdat file that stores all data for selected measure ID and heights

(f) Create .yml configuration file containing corresponding .sdat file name and choosen configuration

5. Execute the GRASP algorithm with corresponding configuration .yml file and .sdat file

6. Give the user the possibility to plot the different results after executing the algorithm

   - Detect all available measure IDs folders in output directory
   - Generate figure data files .mat from GRASP output data in selected measure ID output directory
   - Plot data from .mat files

7. Application can work with project/workspaces

   (a) User can create, open and import a project

   (b) Downloaded data from web services are stored in the project folder

   (c) Results of the GRASP algorithm are stored in the project folder

8. Application can have different configurations

   - Repository directories can be modified for both, earlinet and aeronet downloaded files
   - Ouput directory can be modified
   - GRASP installation directory can be modified: this path can be different for each user

5 Research, plan and adapt

This step helps optimize both time and resources by preventing unnecessary work. Since this project builds upon a previous one, it is not required to develop new code for certain application requirements, but to adapt it to the new needs. Furthermore, the availability of APIs for downloading measurements from the ACTRIS–EARLINET Data Portal [5] and AERONET web application [6] provides a convenient and efficient solution that can significantly reduce development effort.

5.1 Web services development

5.1.1 ACTRIS–EARLINET service

The ACTRIS-EARLINET Data Portal provides a REST API for downloading measurements [9]. This web, shows all the different requests that are supported by the API, gives examples of how to use them and the expected response.

Having all this information available is really helpful, since it allows to localize the requests that are needed for the project requirements.

After testing and comparing the different options and the obtained responses, it becomes clear that the best option is to use the "/products/downloads" endpoint, which allows to download the measurements in a compressed format, filtering the data by type, date and station name.

Even if other configuration parameters are available (measurementID, wavelength and opticaltype), it is not necessary to use them for the project requirements.

Once the behaviour is understood, a class is created to handle the requests and responses of the API.

```
public class EarlinetService{
    string baseUrl = "https://api.actris-ares.eu/api/services/restapi/";
    public virtual System.Threading.Tasks.Task<bool>
        DownloadProductByDateRangeAsync(string type,string FromDate,
        string ToDate,string outputFolder){
         return DownloadProductByDateRangeAsync(type,FromDate,ToDate,
            outputFolder,System.Threading.CancellationToken.None);
    }

    public async Task<bool> DownloadProductByDateRangeAsync(string type,
        string FromDate,string ToDate,string outputFilePath,System.
        Threading.CancellationToken cancellationToken = default){
         try{
             string url = $"{baseUrl}products/downloads?kind={type}&
                fromDate={FromDate}&toDate={ToDate}&stations=brc";
             using (HttpClient client = new HttpClient()){
                 HttpResponseMessage response = await client.GetAsync(url
                    );
                 response.EnsureSuccessStatusCode();
                 using (var fs = new FileStream(outputFilePath,FileMode.
                    Create,FileAccess.Write,FileShare.None)){
```

```
                    await response.Content.CopyToAsync(fs);
                }
                System.IO.FileInfo f = new FileInfo(outputFilePath);
                if (f.Length <= 1048)
                    return false;
                return true;
            }
        }
        catch (Exception ex){
            return false;
        }
    }
}
```

For the creation of this class it has been taken into account the fact that the DownlaodProductByDateRangeAsync method can be used in different points of the program with different configurations.It has also been implemented in such a way that adding new functionalities is easy using the same design pattern.

5.1.2 AERONET service

The AERONET web does not provide a REST API portlet, but it includes web servicie documentation that helps the implementation of a costume service.

The amount of files to be downloaded in this website is larger thant the one in Earlinet, and its documentation can be found in three different sections of the web: For the creation of this class it has been taken into account the fact that the DownlaodProductByDateRangeAsync method can be used in different points of the program with different configurations.It has also been implemented in such a way that adding new functionalities is easy using the same design pattern. For clarity and organization, it has been created

Table 1: AERONET Data Products and Extensions

Category	**Product Description**	**Extension**
Optical Depth [10]	Aerosol Optical Depth (AOD): (descripcion de los datos)	.lev15
	Spectral Deconvolution Algorithm (SDA): (descripcion de los datos)	.ONEILL_lev15
Aerosol Inversions [12]	Inversion products: (descripcion de los datos)	.all
Raw Products Optical Depth [11]	Raw Almucantar: (descripcion de los datos)	.alm
	Raw Polarized Almucantar: (descripcion de los datos)	.alp

the DataType Enum, since the different data types need different Url content in order to be downloaded.

```
public class AeronetService{
    private string baseUrl = "https://aeronet.gsfc.nasa.gov/cgi-bin/";
    public async Task DownloadDataAsync(string destinationFile,string url){
        try{
            using (HttpClient client = new HttpClient()){
                var response = await client.GetAsync(url);
                response.EnsureSuccessStatusCode();

                var content = await response.Content.ReadAsByteArrayAsync();
                await File.WriteAllBytesAsync(destinationFile,content);
            }
        }
        catch (Exception ex){
            Console.WriteLine($"Error downloading data: {ex.Message}");
        }
    }

    public string BuildUrl(DataType _dataType,DateTime startDate,DateTime endDate,string productType = "",string site = "",string product = "",string AVG = "",bool isEnabled = false){
        switch (_dataType){
            case DataType.AerosolInversions:
                if (isEnabled)
                    return BuildUrlAerosolInversions(startDate,endDate,productType,site,product,AVG);
                else
                    return BuildUrlAerosolInversions(startDate,endDate);

            case DataType.OpticalDepth:
                if (isEnabled)
                    return BuildUrlOpticalDepth(startDate,endDate,productType,site,AVG);
                else
                    return BuildUrlOpticalDepth(startDate,endDate,productType);

            case DataType.RawProductsOpticalDepth:
                if (isEnabled)
                    return BuildUrlRawProductsOpticalDepth(startDate,endDate,productType,site,AVG);
                else
                    return BuildUrlRawProductsOpticalDepth(startDate,endDate,productType);
        }
        return "";
    }

    {...} //BuildUrl methods can be found in GitHub repository
}
```

In this application, this class is only used to download the data files listed above, but other types could be downloaded by calling only the DownloadDataAsync and BuildUrl methods in a line.

5.2 Matlab controller development

As it was mentioned in the introduction, this project aims to automatize an execution process of a set of scripts in MATLAB.

The previously developed code has been adapted to the new hybrid system in order to be executed with C#. In order to do that, different actions have been taken.

5.2.1 Create script configuration files and output files

Since the project is hybrid, it is necessary to create a configuration file for the script that will be executed in C#. This file will contain the parameters needed to execute the script. In the previous approach, this files were not needed, since GUI introduced parameters were being stores in the MATLAB workspace. The use of these files, allow the communication between the two different languages. Configuration files allow the different scripts to obtain the information and parameters that the user introduced, while the output files allow the application to know the corresponding results and notify the user about it.

5.2.2 Automatize MATLAB repository for needed data files

Previously, it was necessary to download manually all the data files needed, then, when Matlab Scripts where executed, a dialog window would appear asking for the location of each file. For different data, a different amount of files where needed, having a maximum of five files to select. In order to automatize this process, a script has been created that downloads the data files from the AERONET website and stores them in a local repository.

In order to be able to automatize the process of downloading and giving the application the capacity of work with different workspaces, the repository location has been modified and automatized inside the application. The corresponding file is saved as configuration parameter in both, application and script.

5.2.3 Adapt old scripts to new file formats

5.2.4 Execute MATLAB scripts from C#

In order to execute the MATLAB scripts from C#, it is needed to create a method that uses the ProcessStartInfo class, that can be imported as a library, to execute the MATLAB script. The following example shows a method called RunMatlabScript that takes a string parameter containing the path to the script to be executed. The method uses the Process class to execute the script and it redirects the standard output and standard error to the console.

```
public static void RunMatlabScript (string scriptPath){
    try{
        ProcessStartInfo startInfo = new ProcessStartInfo{
            FileName = "matlab",
            Arguments = $"-batch␣\"run('{scriptPath}')\"",
            RedirectStandardOutput = true,
            RedirectStandardError = true,
            UseShellExecute = false,
            CreateNoWindow = true
        };
        using var process = new Process { StartInfo = startInfo };
        process.OutputDataReceived += (s,e) =>{
            if (e.Data != null)
                Messenger.Default.Send<string>("WriteMatlabOutput",e.Data);
        };
        process.ErrorDataReceived += (s,e) =>{
            if (e.Data != null)
                Messenger.Default.Send<string>("WriteMatlabErrors",e.Data);
        };
        process.Start();
        process.BeginOutputReadLine();
        process.BeginErrorReadLine();
        process.WaitForExit();
        if (process.ExitCode == 0){ //OK
            Logger.Log($"Script␣was␣executed␣correctly,executing␣post␣execution␣actions␣.␣.␣.");
            script.PostExecutionActions();
        }
        else{
            Logger.Log($"Error␣occured␣during␣execution,executing␣post␣execution␣actions␣.␣.␣.");
            script.PostExecutionActions(false);
        }
    }
    catch (Exception ex){
        Logger.Log($"Error␣occured␣during␣execution,executing␣post␣execution␣actions␣.␣.␣.");
        script.PostExecutionActions(false);
    }
}
```

6 Architecture

6.1 MVVM

Design pattern MVVM (Model-View-ViewModel) is a software architecture patter designed with the goal of having a clear separation between the user interface (frontend) and the business logic (backend) [13]. There are different freameworks that use this standard, such as Angular, .NET WPF...

This application needs to be able to execute in CALCULA operating system (Linux), so it is necessary to use a framework that is compatible with this system. Originaly, it was going to be developed using .NET WPF, but it was decided to use Avalonia instead, as it is a cross-platform framework that can be executed in different operating systems. The differences between both frameworks are very low, mainly being the name of the files (.axaml instead of .xaml) and the way to define the user interface.

This architecture is divided in three main components:

- Model: It contains the data and most of the app logic. It structures the information (classes and identities), how to retrieve it and how to manage it (services, data bases and controllers).
- View: It defines all the elements in the user interface and what the user will see and will interact with. The only responsability of the code defines in this sections is to define the visual structure of the app and to create input elements for the user to interact with.
- ViewModel: It is the bridge between the Model and the View. It is the one that controls the flow of data between the Model and the View.

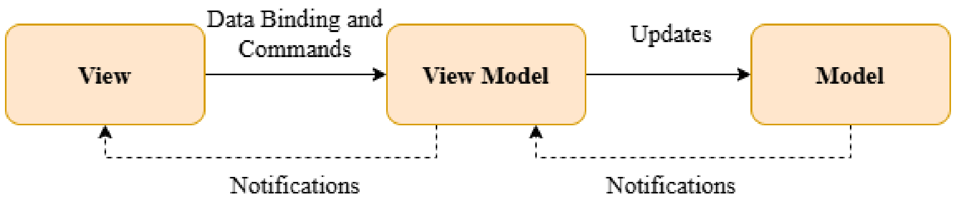


Figure 9: MVVM design pattern

The most important advantages of using this desing pattern are:

- Changes only affect the view or the model, not both
- Separated testing for views and models
- User interface can be modified without affecting the model
- Teams with different developers can work on the same project simultaneously without affecting each other

6.1.1 Observable Object

In order be able to update the user interface automatically when the data changes, it is used Data Binding, a feature of the MVVM pattern. It is implemeted using ObservableObject as base class.

```
public class User : ObservableObject
{
    private string name;

    public string Name
    {
        get => name;
        set => SetProperty(ref name, value);
    }
}
```

In the example, it can be seen how a variable is updated, when it is detected that the new value is different from the old one, SetProperty is called, which implements INotifyPropertyChanged interface, which is the one in charge to notify the corresponding view.

In order that the view is updated, it is necessary to call the variable with the same name and to indicate that it is Bindable in the corresponding .axaml file, as it can be seen in the following example:

```
<TextBlock Text="{Binding␣Name,␣Mode=TwoWay,␣UpdateSourceTrigger=
    PropertyChanged}" />
```

In this case, mode option indicate that variable can be updated from the view to the model and viceversa, and update source trigger option indicate that the update will be done when the property changes. Other options can be used, but these are the ones that are most common and that have been used in this project.

6.1.2 Relay Command

Relay command is used to implement commands in the MVVM pattern. It is implemeted using ICommand as base class. It can be used in different ways, but in this project it has been implemented using the following pattern for all View Models:

```
public class ViewModel : ObservableObject
{
  public ICommand Cmd => new RelayCommand(Execute, CanExecute);
  private void Execute(object _)
  {
      ExecutionActions();
  }

  private bool CanExecute(object _)
  {
      return true;
  }
}
```

In order to execute the command, it is necessary to link an interface element of the .axaml file to the Execute method

```
<Button Content="Execute" Command="{Binding␣Cmd}" />
```

6.2 SOLID principles

SOLID principles are a set of five principles that are used to design software with the obejective of making it easy to maintain and modify, making a project more maintainable ands scalable [16]. It lowers the dependency between classes, minimizing the impact of changes and making the code more reusable, mantainable, flexible and stable.

They also support growth in development, making it easier to add new features without the need of modifying already implemented code.

These principles are:

- Single Responsibility Principle (SRP)
- Open/Closed Principle (OCP)
- Liskov Substitution Principle (LSP)
- Interface Segregation Principle (ISP)
- Dependency Inversion Principle (DIP)

6.2.1 Single Responsibility Principle

States that *a class should have only one reason to change*, meaning that it should have only one responsibility, job or purpose within the softwar project.

In order to apply this principle, it is necessary to divide the code into different classes, each one with a specific responsibility. It is important to note that this principle is not always possible to apply, but it is a good practice to try to do so.

6.2.2 Open/Closed Principle

States that *an entity should be open for extension, but closed for modification*, meaninig that it should be easy to extend the functionality of the entity without modifying its existing code.

In order to apply this principle it can be used abstraction by implementing Interfaces and Abstract classes and by using different design patterns like Strategy or Decorator.

6.2.3 Liskov Substitution Principle

Introduced in 1987 by Barbara Liskov, it states that *derived or child classes must be substitutable for their base or parent classes*, ensuring that any class can be used in place of its parent class without any loss of functionality.

In order to apply this principle, it is important to ensure that subclasses can be used anywhere where the base class is used without changing the expected behaviour, avoiding overriding methods and making sure to not thow NotImplementedException. This can be done by using Interfaces and Abstract classes.

6.2.4 Interface Segregation Principle

This principle applies directly to interfaces, and it states that *do not force any client to implement an interface which is irrelevant to them*. The main objective of this principle is to not implement too big interfaces and to implement instead smaller and more specific interfaces.
If this principle is used in a project, the corresponding software architecture should have many interfaces implementations with specific purposes.

6.2.5 Dependency Inversion Principle

States that *high-level modules should not depend on low-level modules, but both should depend on abstractions*, ensuring that the dependencies are inverted and that the high-level modules are not dependent on the low-level modules.
In order to apply this principle, it is suggested to use abstractions and interfaces to define the dependencies between modules.

6.3 Project architecture

Applying the MVVM design pattern and the SOLID principles, the project architecutre can be summed up as follows:

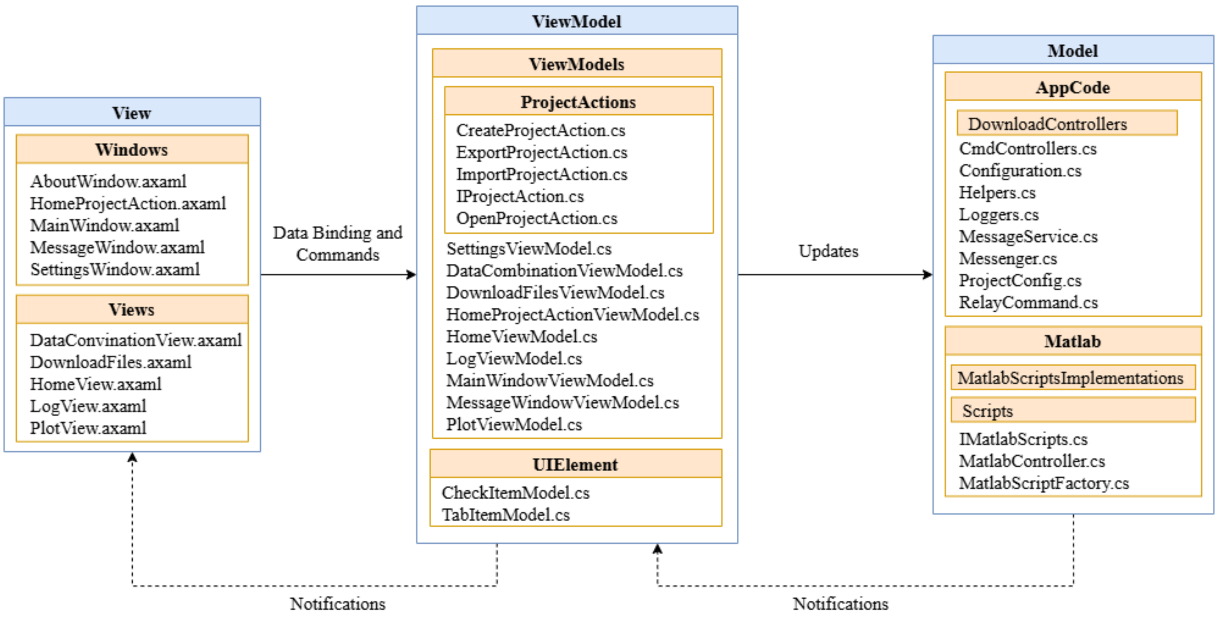


Figure 10: MVVM design pattern applied to the project architecure

In Figure 10 it can be seen how the Views are divided into different folders. The Windows folder contains all the files containing the definitions of the different windows and dialogs of the application, while the Views folder contains all the files containing the definitions of different UserControls. UserControls are used as Content in the MainWindow, in this case, as Contents of the different tab items.

The ViewModel section contains all the files containing the definitions of the different ViewModels of the application. Most of them are located in the ViewModels folder, but it can also be appreciated that there is a folder named UIElement, where the models for CheckItems and TabItems are located. With this classes, adding these items to Views and ViewModels is easier to mantain and update.

Finally, the Model component is composed by all the internal classes of the application. It has been divided by AppCode, where is located all the implementations related to app Logic, and by Matlab, where all the MATLAB scripts are located aswell as the necessary implementations for the app to execute them.

7 Graphic User Interface (GUI) design

In order to give to the user only the essential information, it was chosen to divide the different main actions that the application should perform into different views. In order to accomplish that, different UI implementations can be used. In this case, it was decided to implement a TabMenu, with three different views for each action: Download data files, Combine Data for GRASP algorithm execution and Plot GRASP results.

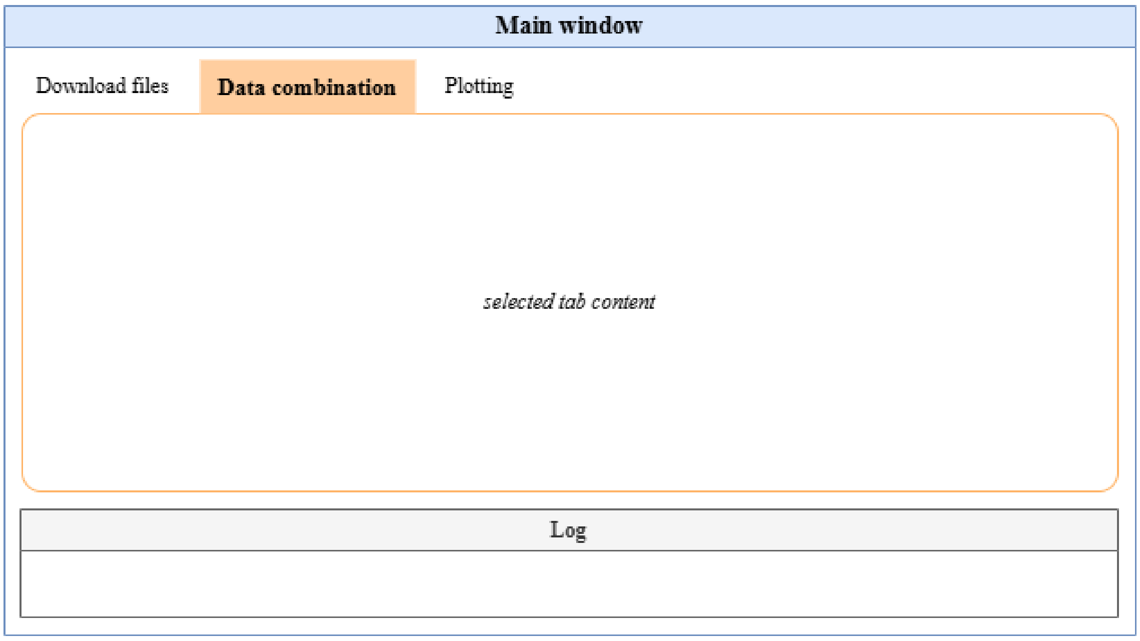


Figure 11: TabMenu design

Other necessary Windows where for Configuration, About and Message Windows. The conentet of these windows should be as simple as possible, since they are complementary windows and they do not contain a lot of information. For the Configurations window, it is only necessary to have one input element in order to introduce the value for each configuration parameter. Design can be modified having into account the necessity of the current application.

When the application is first initialized, it is needed a Home View, with different buttons, where the user can choose which type of action it wants to perform: Create, Open or Import project.

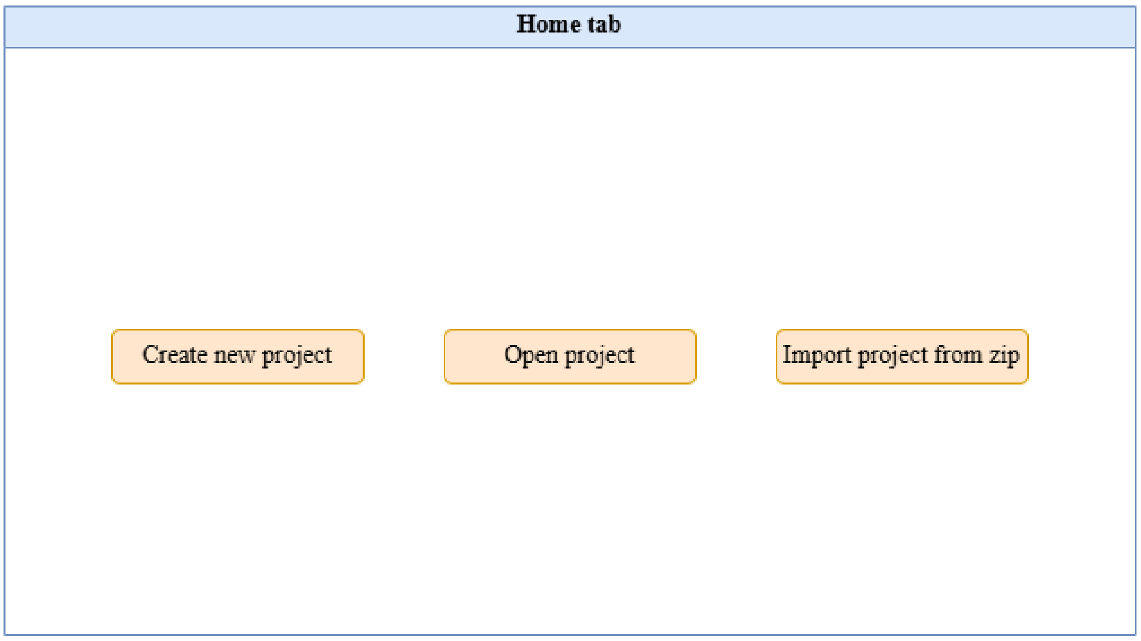


Figure 12: Home design

7.1 Download data files

In the requirements list, it was specified that the user should be able to download the data files within a selected date range and a selected station. After that, user should be able to see a list of the downloaded files from each website.In order to implement this functionality, the first desing uses two calendar selectors in order to choose "From date" and "To date". A text selector is used in order to select the station and a Download Button in order start the execution.To finish, a list of the downloaded files is shown in a text block. All these features can be seen in the Figure 13.

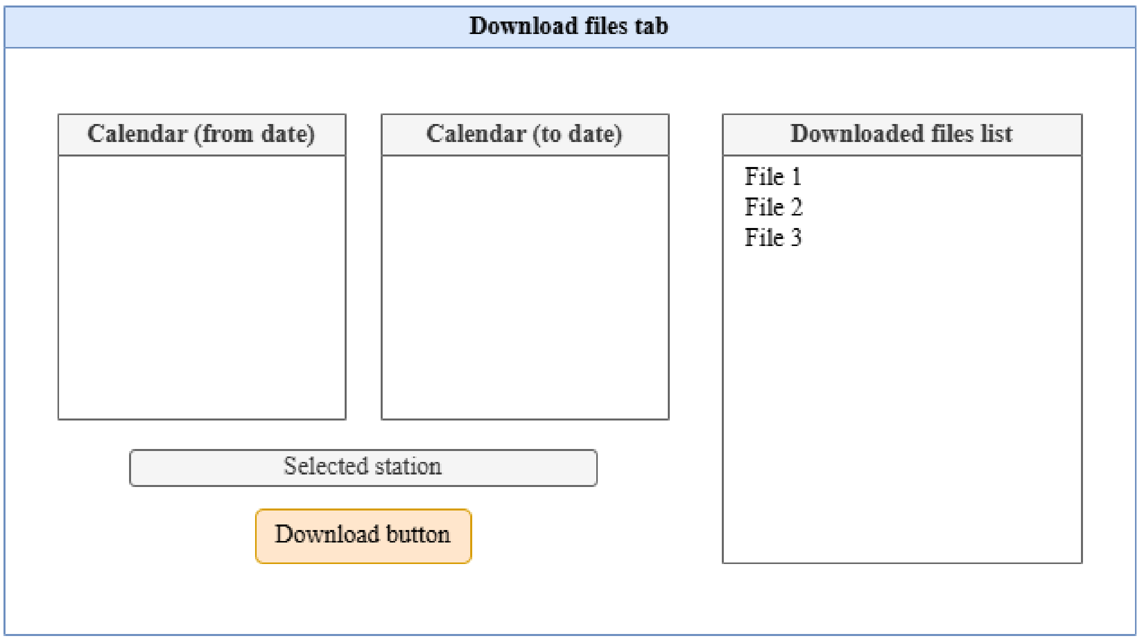


Figure 13: Download tab design

7.2 Data combination

For the data combination step, the user needs to select with wich of the available measure IDs it is going to be executed the GRASP algorithm. It is necessary also two options in order to select the minimum and maximum height for the data combination.

Once the data is previewed, it is necessary to notify the user the different configurations that are available in order that the user chooses one of them depending on the available data for selected date and time.

Two buttons are used in order to start the different scripts execution: one for the data preview and the other one for generating the combination .sdat and .yml files and executing the GRASP algorithm execution.

All these features can be seen in Figure 14. In addition, a text section should be added in order to see the Ouput and Errors obtained from executing the different Matlab scripts used for pre-processing the data. Thanks to that element, the user will be able to see the process during the execution and the errors that may occur during it.

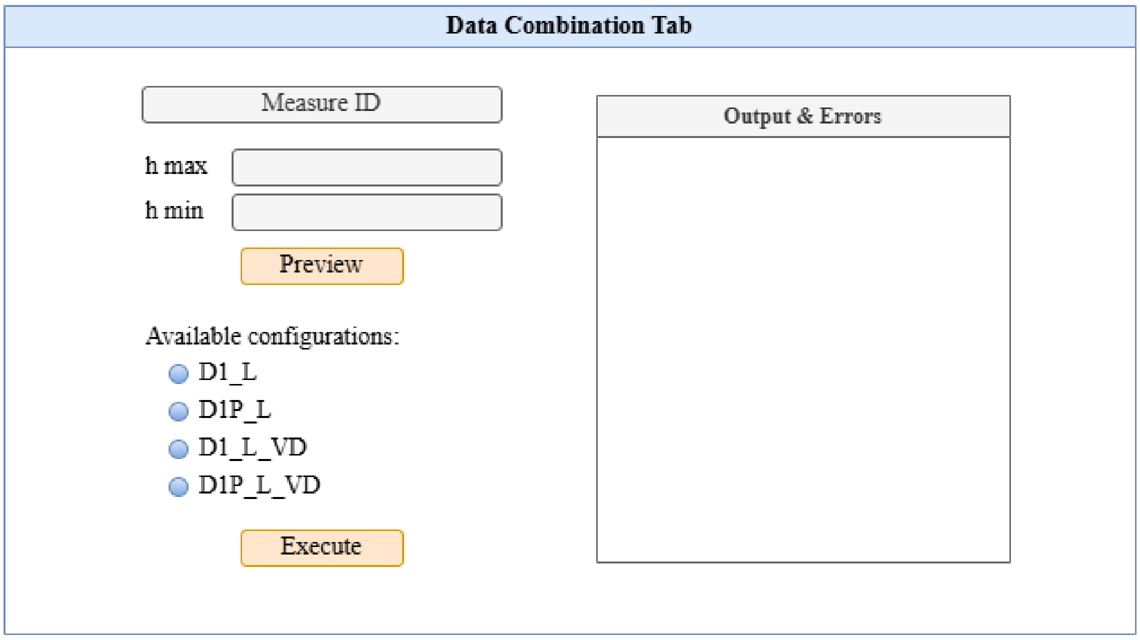


Figure 14: Data combination tab design

7.3 Plot GRASP results

For the last tab, the plotting tab, it is necessary to understand how the project folders are organized in order to know all the necessary elements that are going to be needed in order that the user can introduce all the necessary information.As it can be appreciated in Figure 15, the project folder is organized in different subfolders, each one with a specific purpose. Data folder will contain all the downloaded data and Output data will contain the results of both, the data combination script and the GRASP algorithm execution.

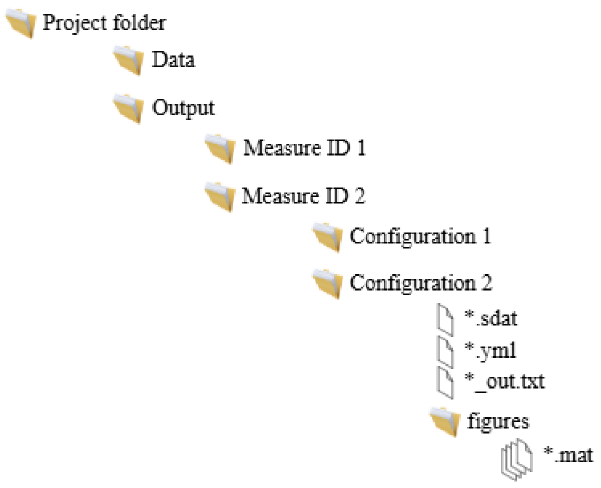


Figure 15: Project folders organization

GRASP execution output is saved in a file named UPC_{config}_out.txt. This file will be the one used in order to create and save all the different files for each figure. Therefore, it is necessary first to specify the measure ID, then the measurement file and to finish, the figure name that is desired to be seen.

This actions can be implemented with the different buttons and text inputs shown in Figure 16. Additionally, since this tab will also be executing Matlab scripts, it will be helpfull for the user to see the progress and possible errors that can appear.

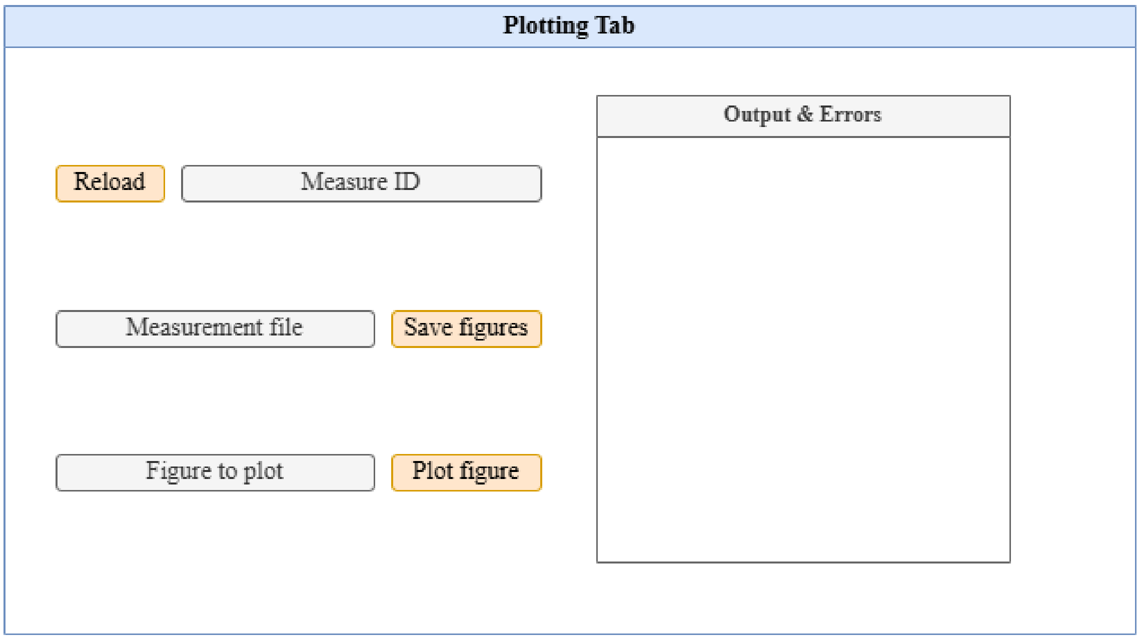


Figure 16: Plot tab design

8 Implementation

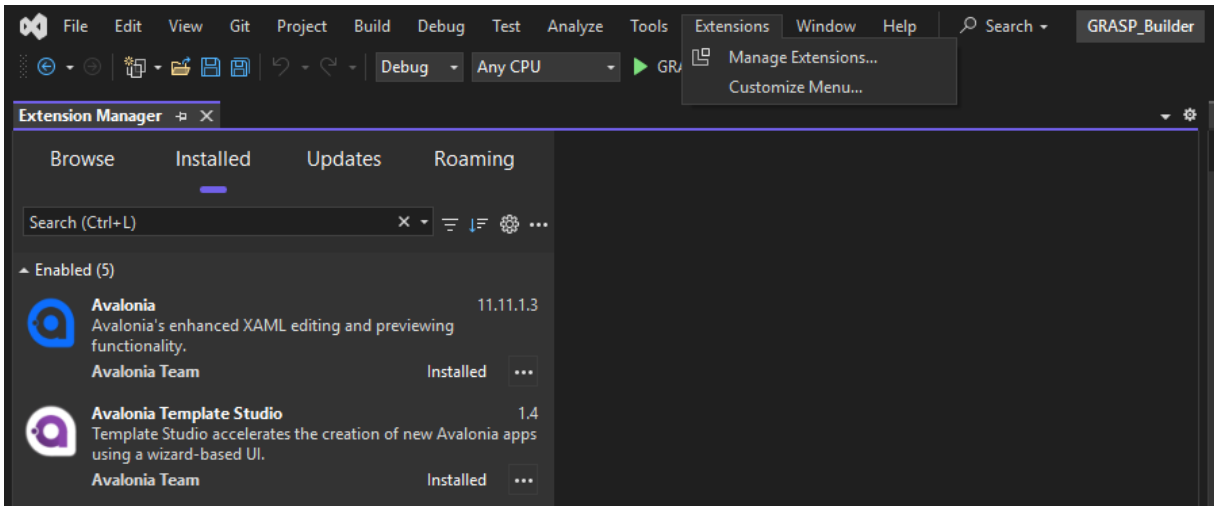
The code implmeneted for this project can be found in the GitHub repository[22]. Some implementations have been explained previously in other sections of this document like the Matlab Controller or the Aeronet Download Service, but this section aims to explain the purpose of the different classes and the different design patterns that have been used in order to follow the SOLID principles as much as possible.

8.1 Set up development enviroment

In order to create this project, it is necessary to meet several requirements to have a correct develpment enviroment. First of all, it is necessary to choose a correct IDE. There are different options, including JetBrains Rider and Visual Studio Code, but for this project, it has been decided to use Visual Studio Community [17].

The application will be run using .NET, which is a free and open-source application platform supported by Microsoft used for developing different type of applications using C# [19] and which is usally already installed in the system[18].

In order to use Avalonia, it is necessary to install the corresponding extension for Visual Studio. It can be done by accessing to Extensions¿ManageExtensions:



Avalonia 11.11.1.3
Avalonia's enhanced XAML editing and previewing functionality.
Avalonia Team Installed

Avalonia Template Studio 1.4
Template Studio accelerates the creation of new Avalonia apps using a wizard-based UI.
Avalonia Team Installed

Figure 17: Extensions installed in Visual Studio

Since the app will be executing Matlab scripts, it is also necessary to have Matlab installed. In this case it has been used Matlab R2024a[20].

In order to run the GRASP algorithm it is necessary to acces to CALCULA, where it is already installed and running in Ubuntu that will only be used for testing.

8.2 Views and ViewModels

In order to not overload the content of the different files for the User Interface implementation (Views and Windows) and management (View Models) it has been created a different View class for each Tab and for each Window, that can be resumed in the following list:

- About Window: Window that shows general information about the app and containg button ”Help” which opens User Manual
- Data Combination View: Tab where the user can select Measure ID to combine data, showing available configurations for selected date and time and executing GRASP.
- Data Download View: Tab where the user can select dates and station and downloading corresponding data form Earlinet and Aeronet websites.
- Home View: Shows the following action options: to create, open or import a project.
- Home Project Action Window: Window that appears when the user clicks on a project action option in Home View and in charge of performing it.
- Main Window: Window that contains all the tabs and shows them when necessary. It can show the Home View when no project is loaded or a combination of the other tabs with an additional Log and a Menu in order to manage the app Settings or the current project.
- Message Window: Window that shows a message to the user that can be an Error, a Warning or just informatiom.
- Log View: Tab that shows the Log content of the app.
- Plot View: Tab where the user can select and create the desired figures after executing GRASP algorithm.
- Settings View: Window where the user is able to modify the app configuration parameters.

All this classes for both Views and ViewModels are implemented using the MVVM pattern using ObservableObject and RelayCommand classes from the .NET community toolkit. In addition, all ViewModels classes have been structured by regions in order to homogeneously group the different properties, commands and methods.

In order to stablish communication between different classes to notify modifications, it has been used the Messenger class from the .NET community toolkit. It consists on defining a Send Action, that can include some parameter or variable and a Receive Action, that will be executed when the Send Action is called, executing the linked method.

In the following example it can be seen how the DataCombinationViewModel class has been implemented using the Messenger class to receive modifications in the buttons enabled state:

```
public DataCombinationViewModel()
{
    Messenger.Default.Register<bool>("UpdateButtonsEnabled",
        UpdateButtonsEnabled);
}

private void UpdateButtonsEnabled(bool status)
```

```
{
    AreButtonsEnabled = status;
    if (!status)
    {
        isEnabled_D1_L = false;
        isEnabled_D1P_L = false;
        isEnabled_D1_L_VD = false;
        isEnabled_D1P_L_VD = false;
    }
}
```

In the corresponding controller, in order to send the modification in order to enable the buttons, for exemple, it is necessary to execute the following line:

```
Messenger.Default.Send<bool>("UpdateButtonsEnabled", true);
```

Using Messenger allows to notify modifications in the App between classes that exist simultaneously, in this case between the different Tabs and between ViewModels and Controllers. In the case of creating new Windows like the Settings one, it is not necessary, since the needed information can be passed as parameters to the Constructor when it is initialized.

8.3 Controllers

In the backend of the app, different classes and controllers have been implemented following the Single Responsability Principle (SRP). Each one has its own responsability and purpose:

- Download Controllers: Different controllers have been developed for each different web site. They both implement the IDownloadController Interface. Each controller is responsible of the different actions that need to be executed when downloading the respective data by managing their respective web services.
- CmdController: This controller is responsible of the different actions needed to execute commands, in this case, necessary to execute the GRASP algorithm via the command line. Since this application have been designed to work with CALCULA, command line for executing GRASP only is executed with Ubuntu sintaxis.
- AppConfig: This class manages the application configuration, obtaining and saving the different configuration parameters from the configuration file (config.json), located in the execution directory.
- Helpers: Different helper classes have been developed to perform different actions. FormatHelpers are used for changing the format of variables, for exemple: Converting a dictionary to a string variable. FileHelpers are used for managing the files and directories renaming, copying, etc.
- Logger: This class is responsible of managing the logging actions.
- MessagesController: This controller is responsible of managing the different messages that can be shown to the user: Errors, warnings and information messages.

- ProjectConfig: This class manages the project configuration, obtaining and saving the different configuration parameters from the project file (project.grasp), located in the project directory.
- MatlabScriptController: This controller is responsible of managing the different actions needed to execute any MatlabScript.

8.4 Polymorphism through interfaces

With the objective of applying the Open/Closed Principle (OCP) and the Interface Segregation Principle (ISP), different interfaces have been implemented in order to be used in different points of the project. Using polymorphism, enables the possibility of using objects of a derived class as objects of a base class[23].

Polymorphism can be implemented with a parent or base class, which defines the common properties and implements virtual methods. Derived classes can use the already implemented methods of the parent class, or override them.

Another possible implementation of polymorphism is through the use of abstract classes or interfaces, which is the one used in this project. Interfaces are contracts that define a set of methods that a class must implement. This enables the possibiltiy to work with different classes, where each class implements the interface in a different way but all classes and actions are used uniformly by calling the interface methods.

8.4.1 Factory pattern

Factory pattern is a creational design pattern that provides an interface for creating objects of a parent class, but deciding which subclass to instantiate[24]. This allows to resume all the different conditions in one single method, instead of using multiple if-else conditions during the development of the code.

An exemple of this pattern can be seen in the following example, showing the implementation of the DownloadControllerFactory Create method:

```
public static IDownloadController Create(DownloadType type, string
    _repositoryDirectory, string _workingDirectory)
{
    switch (type)
    {
        case DownloadType.Earlinet:
            return new EarlinetDownloadController(_repositoryDirectory,
                _workingDirectory);
        case DownloadType.Aeronet:
            return new AeronetDownloadController(_repositoryDirectory,
                _workingDirectory);
        default:
            throw new NotSupportedException($"Download type {type.
                ToString()} is not supported");
    }
}
```

8.4.2 Project Actions

The first usage of this coding strategy can be found in IProjectAction interface. This interface is used when initializing a project when the application is opened, where the user can choose between the different actions that can be executed: Create, Open or Import from .zip file. Export action has also been implemented, but it is also accessible through the File Menu, once a project is loaded.

```
public interface IProjectAction
{
    public string Title { get; set; }
    public string ProjectName { get; set; }
    public string DirectoryPath { get; set; }
    public bool IsProjectNameVisible { get; set; }
    public bool IsDirectoryPathVisible { get; set; }
    public Task<bool> Execute();
    public Task Browse();
}
```

This interface defines five different properties, that are used for initializing the View and ViewModel before showing the new window, and two methods, that correspond to the different actions that can be executed in this window: Browse a directory and Execute.

With this implementation, IProjectAction object is used in HomeProjectActionViewModel. Only in the constructor the type of action needs to be specified. In the rest of the class, all the Bindings and Commands are executed using this object equally for all the types. A simple example is the implementation of Execute command.

```
public ICommand BrowseCmd => new RelayCommand(async _ => await
   BrowseExecute(), CanBrowse);

private bool CanBrowse(object _)
{
    return true;
}

private async Task BrowseExecute()
{
    _projectAction.DirectoryPath = DirectoryPath;
    _projectAction.ProjectName = ProjectName;

    await _projectAction.Browse();

    DirectoryPath = _projectAction.DirectoryPath;
    ProjectName = _projectAction.ProjectName;
}
```

In this Command, it can be appreciated the simplicity of the code and the lack of if-else conditions and knowledge of the type of action that is being executed.

8.4.3 Download Controllers

Another class that can take advantage of the use of polymorphism is the DownloadController, which needs a simpler interface.

```
public interface IDownloadController
{
    public string RepositoryDirectory { get; set; }
    public string WorkingDirectory { get; set; }
    public Task Download(DateTime FromDate, DateTime ToDate, string
        station);
}
```

With this implementation, the only method that needs to be called is Download, but in each implementation, this method execute different actions.

For the moment, only two classes are implemented using this interface, but in the future, if it was desired to add any other data source, it would be possible to implement a new class that implements this interface and used it in the ViewModel as it is done now with the IDownloadController object.

```
IDownloadController downloadController = DownloadControllerFactory.
    Create(DownloadType.Earlinet, _earlinetRepositoryDirectory,
    _workingDirectory);
await downloadController.Download(FromDate, ToDate,"BRC");
```

8.4.4 Matlab Scripts Implementations

The last interface that has been implemented is IMatlabScript. Running Matlab scripts using C# is not a complicated task, since the use of the library ProcessStartInfo agilizes all the process, needed only the name of the script to be executed. But for this project it was requiered to execute more actions.

```
public interface IMatlabScript
{
    public string Name { get; }
    public Dictionary<string, object> vars { get; set; }
    public void PreExecutionActions();
    public void PostExecutionActions(bool resultOK = true);
}
```

It was needed not only to run a script but to have communication between the Matlab scripts and the application. In order to solve this problem, it was decided to create different files:

- Configuration files: Parameters are saved from C# application and read in Matlab before running the corresponding script.
- Ouput files: Results are saved from Matlab and read in C# application after running the corresponding script, in order to actualize the UI or execute further actions.

This is very useful, since it allows to have a clear separation between the different tasks and to make the code more maintainable. If it is desired to add a new script, it is only necessary to create a new class using this same interface and call the method for running a Matlab script as in the following exemple:

```
MatlabController.RunMatlabScript(ScriptType.Preview, dict,"_DC");
```

This is possible because this interface is only used for running Matlab Scripts. The MatlabController class is responsible of executing all the sequence for running a script, which consists on:

1. Execute pre-execution actions
2. Run the script (start matlab process)
3. Receive output from Matlab
4. Receive results from Matlab
5. Execute post-execution actions

Each Script implmentation will be the responsible of defining the different actions to be executed in each step, having into account if the script finished with error or was executed successfully.

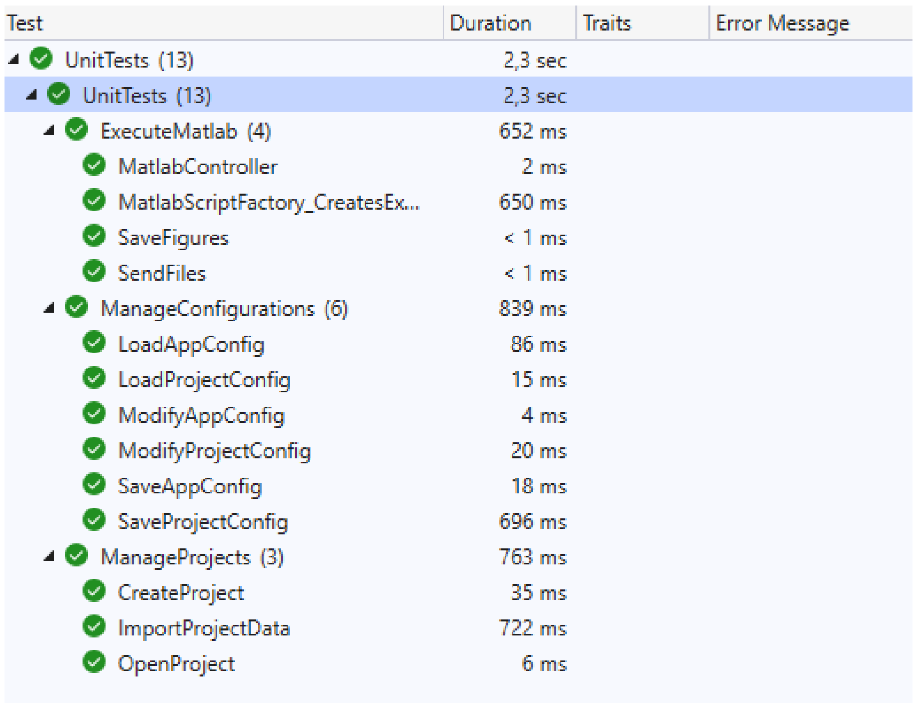
9 Testing

During the development of the application, different tests have been performed to ensure that the application works as expected:

1. First contact:
   In order to know how the initial implementation was working and optimize the process it was needed to know how the different steps where being executed and the expected results
   - Download files
   - Execute Matlab scripts with old data
   - Execute GRASP with old data
2. Test downloaded data: After automatizing the download process, it was needed to test the different matlab scripts with the new data. During this tests, it was detected that some data was being downloaded with a new file format. This was a problem, since the old scripts were not compatible with this new format and they had to be modified.
3. Test GRASP with generated data: After modifying the scripts, it was needed to test the obtained data with the GRASP algorithm. It was detected that the data was not being processed correctly since the configuration file did not mathc the correct values. This was an issue with file versions that was solved by updating the configuration file to the correct version.
4. Test full UI execution: After defining the application as configurable and workspace independent, it was needed to test the full execution of the application. During this round of tests, it was checked that the project folders were working as expected, improving the project management process.
5. Test full UI execution in CALCULA: Since CALCULA works with LINUX different aspects of the application had to be tested. Some changes were needed for the folders and files management, since some routing was only working in Windows. During this tests the full execution of the application was performed successfully, since it was the only test where the GRASP algorithm could be executed.

Unit testing has also been performed to ensure that the application is working as expected. Unit testing consists in testing the separated components of the application, in this case the Model classes, to ensure that they are working as expected[26].

Implementing this type of testing in a project allows to detect and fix some issues that were not detected during the integration tests and reduces the time consumption of some testing process.



Test	Duration	Traits	Error Message
✓ UnitTests (13)	2,3 sec		
✓ UnitTests (13)	2,3 sec		
✓ ExecuteMatlab (4)	652 ms		
✓ MatlabController	2 ms		
✓ MatlabScriptFactory_CreatesEx...	650 ms		
✓ SaveFigures	< 1 ms		
✓ SendFiles	< 1 ms		
✓ ManageConfigurations (6)	839 ms		
✓ LoadAppConfig	86 ms		
✓ LoadProjectConfig	15 ms		
✓ ModifyAppConfig	4 ms		
✓ ModifyProjectConfig	20 ms		
✓ SaveAppConfig	18 ms		
✓ SaveProjectConfig	696 ms		
✓ ManageProjects (3)	763 ms		
✓ CreateProject	35 ms		
✓ ImportProjectData	722 ms		
✓ OpenProject	6 ms		

Figure 18: Implemented Unit Tests

In 18 it can be seen the implemented unit tests for this project, where different classes have been tested: application configuration, project configuration and Matlab script management. The main advantages of this type of testing are that they can be perform efficiently, in this case they have been executed within seconds, and can be performed during development. This, helps the developer to check that the new code is still working as expected.

10 Results

In this section it is described the released version of the application and the final execution sequence, even a more detailed explanation can be found in the User Manual, in the Annex.

10.1 Final User Interface

During the development of the application, the user interface has been modified to be more user-friendly and to add more features that were not contemplated during the design phase.

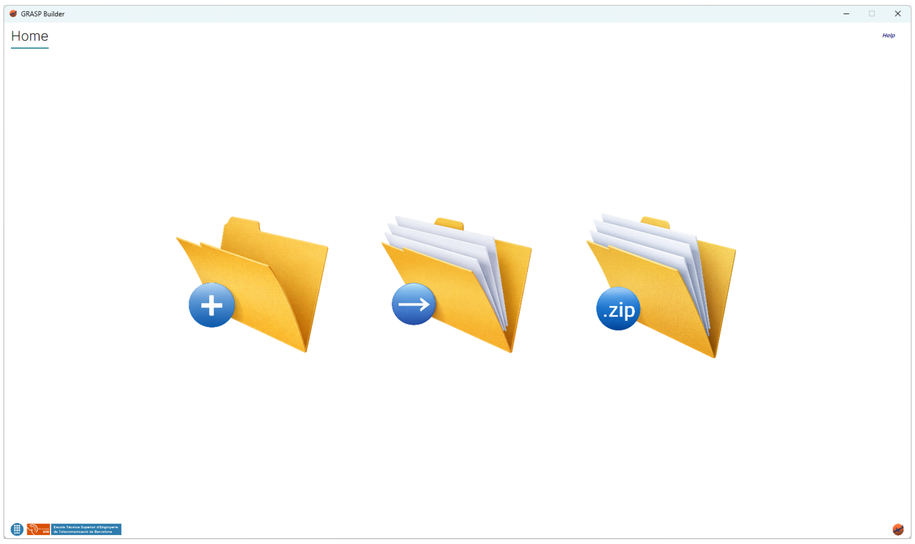
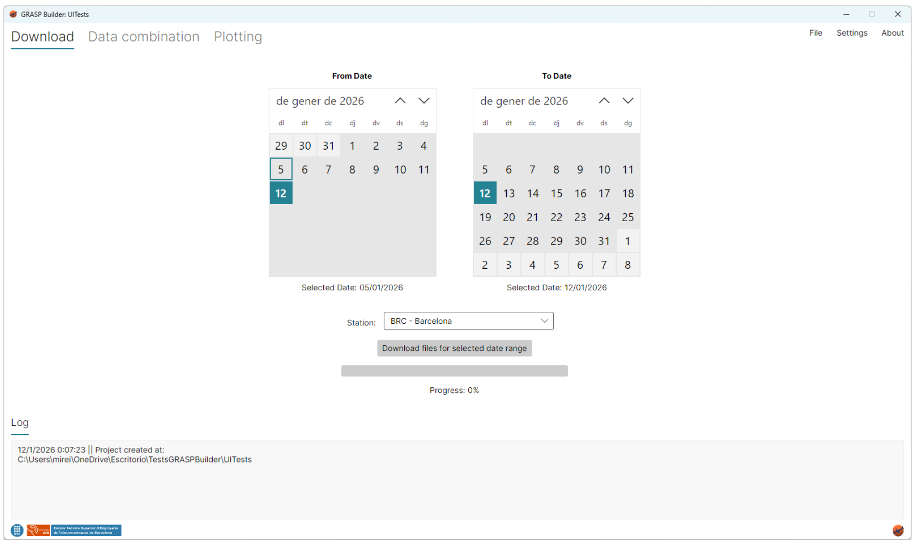


Figure 19: Home view

The Home view, which is shown when the application is initialized, allows the user to select the different options available in the application: Create, Open or Import project. This can be seen in Figure 19. It has also been added the help button, which opens the User Manual in .pdf format.

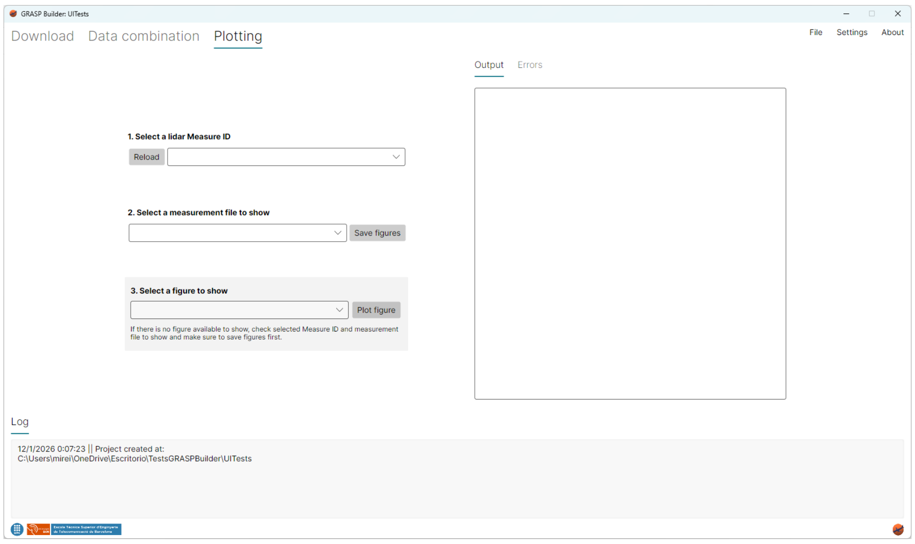


Selected Date: 05/01/2026
Selected Date: 12/01/2026
Station: BRC - Barcelona
Download files for selected date range

```
12/1/2026 0:07:23 || Project created at:
C:\Users\mirei\OneDrive\Escritorio\TestsGRASPBuilder\UITests
```

Figure 20: Download view

The Download View mantains mostly the initial design. Since the process of downloading files can depend on the user's internet connection, it has been added a progress bar to show the progress of the download.



1. Select a lidar Measure ID

2. Select a measurement file to show

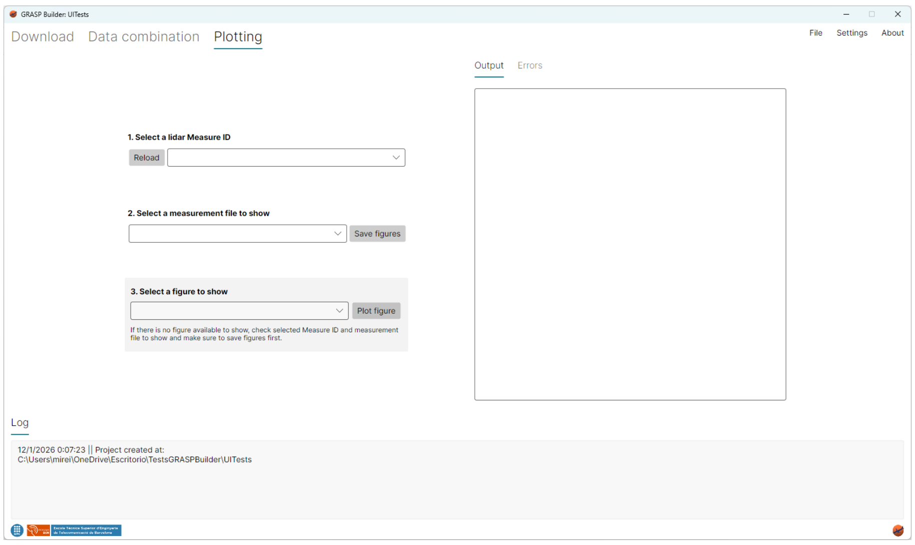
3. Select a figure to show

If there is no figure available to show, check selected Measure ID and measurement file to show and make sure to save figures first.

```
12/1/2026 0:07:23 || Project created at:
C:\Users\mirei\OneDrive\Escritorio\TestsGRASPBuilder\UITests
```

Figure 21: Data Combination view

Data Combination has not been modified from the initial design, since it allows the user to select the different files to be combined and the different parameters to be used for the combination. This can be seen in Figure 27.



1. Select a lidar Measure ID

2. Select a measurement file to show

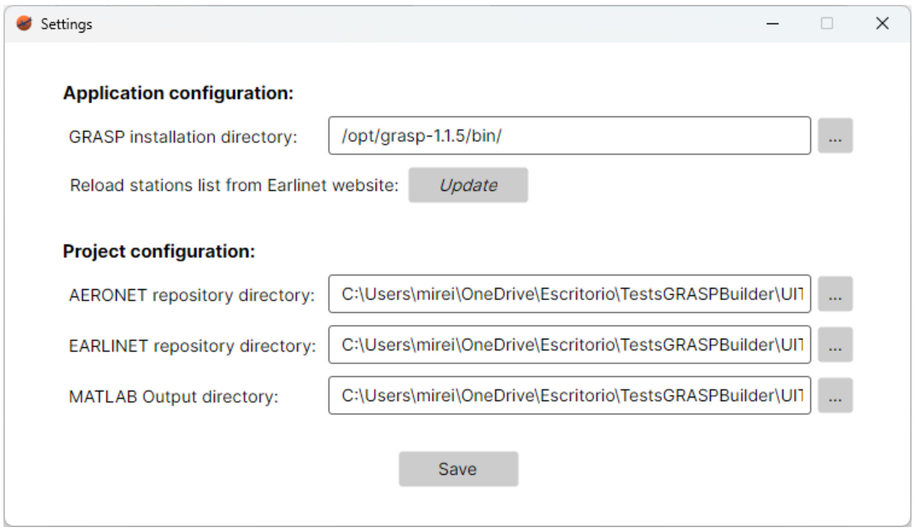
3. Select a figure to show

If there is no figure available to show, check selected Measure ID and measurement file to show and make sure to save figures first.

```
12/1/2026 0:07:23 || Project created at:
C:\Users\mirei\OneDrive\Escritorio\TestsGRASPBuilder\UITests
```

Figure 22: Plotting view

Plotting view has not been modified from the initial design neither, since it allows the user to select the different files to be plotted and the different parameters to be used for the plotting, shown in Figure 28.



Application configuration:

GRASP installation directory: /opt/grasp-1.1.5/bin/

Reload stations list from Earlinet website:

Project configuration:

AERONET repository directory: C:\Users\mirei\OneDrive\Escritorio\TestsGRASPBuilder\UIT

EARLINET repository directory: C:\Users\mirei\OneDrive\Escritorio\TestsGRASPBuilder\UIT

MATLAB Output directory: C:\Users\mirei\OneDrive\Escritorio\TestsGRASPBuilder\UIT

Figure 23: Settings window

Settings window has been created in order to allow the user to configure both the application and project settings, shown in Figure 23. This type of configuration has been clearly separated, in order to let the user know which modifications will affect only the project and which ones will affect all executions of the application, independently of the project loaded.

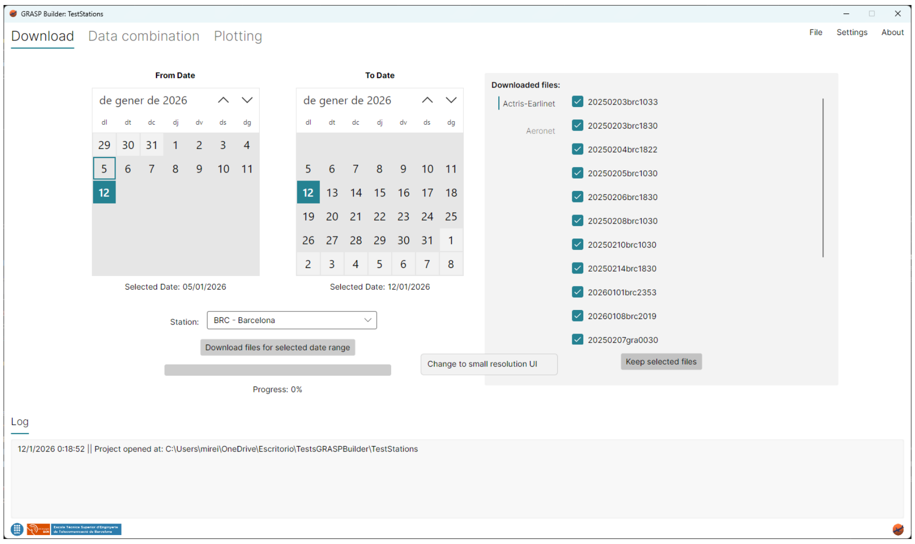
10.2 Work-flux

The final work-flux for working with GRASP algorithm can be resumed in the following steps, all included in the developed application and executed in CALCULA environment:

1. Download files: Select date range and station and download the corresponding files. Available files will be shown in same View
2. Combine data: Select the desired MeasureID, choosing the data and time range to be used. Heights can be configured and available configurations will be automatically shown.
3. Execute GRASP: Send data will create the input files automatically for GRASP algorithm and will execute it.
4. Plot results: Available results will be automatically detected. If possible, list of figures will be created and saved in Output directory. The user will be able to choose which figure to show.

10.3 Low resolution User Interface

Using Avalonia provokes the inability to resize the application window, since it does not mantain the correct format for the moment. In some tests, it has been detected that all actions can not be executed in a low resolution system. Then, it has been created the option of a low resolution UI, which can be selected by right clicking the application window and selecting the option "Change to Low resolution UI", shown in Figure 24.



Selected Date: 05/01/2026

Selected Date: 12/01/2026

Station: BRC - Barcelona

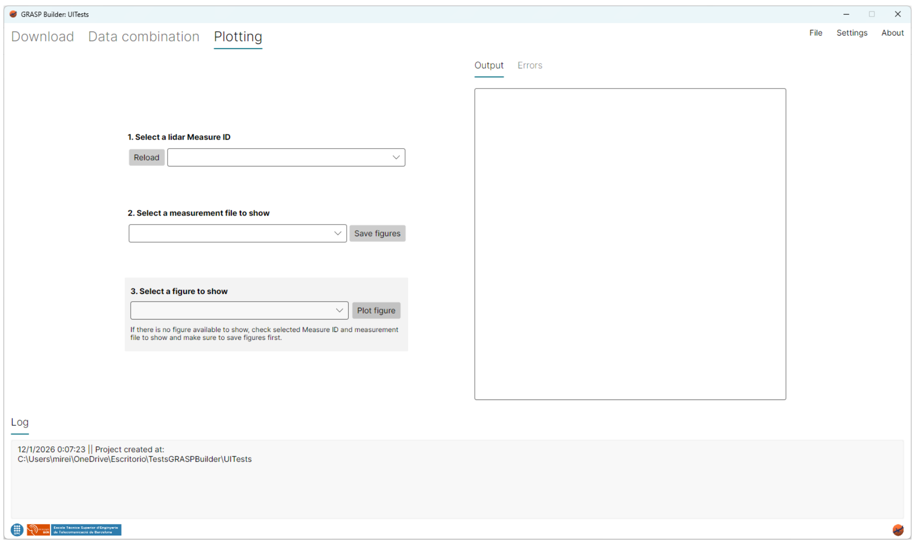
Progress: 0%

- ✓ 20250203brc1033
- ✓ 20250203brc1830
- ✓ 20250204brc1822
- ✓ 20250205brc1030
- ✓ 20250206brc1830
- ✓ 20250208brc1030
- ✓ 20250210brc1030
- ✓ 20250214brc1830
- ✓ 20260101brc2353
- ✓ 20260108brc2019
- ✓ 20250207gra0030

```
12/1/2026 0:18:52 || Project opened at: C:\Users\mirei\OneDrive\Escritorio\TestsGRASPBuilder\TestStations
```

Figure 24: Low resolution UI

This option allows the user to execute the application in a low resolution system, having a slightly modified UI, which can be seen in the following figures: Figure 24

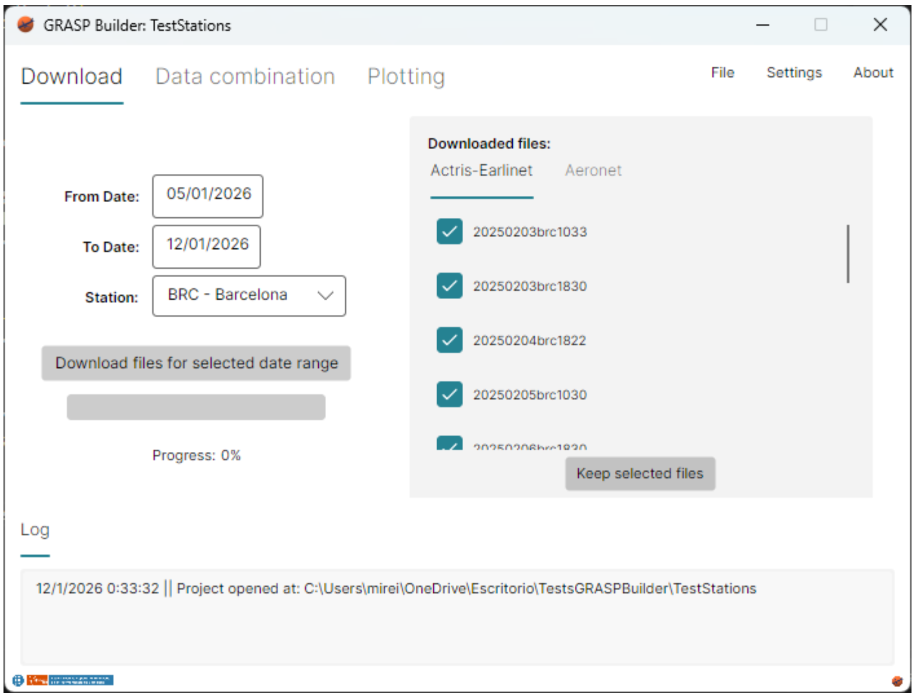


If there is no figure available to show, check selected Measure ID and measurement file to show and make sure to save figures first.

```
12/1/2026 0:07:23 || Project created at:
C:\Users\mirei\OneDrive\Escritorio\TestsGRASPBuilder\UITests
```

Figure 25: Plotting view

Plotting view has not been modified from the initial design neither, since it allows the user to select the different files to be plotted and the different parameters to be used for the plotting, shown in Figure 28.



From Date: 05/01/2026

To Date: 12/01/2026

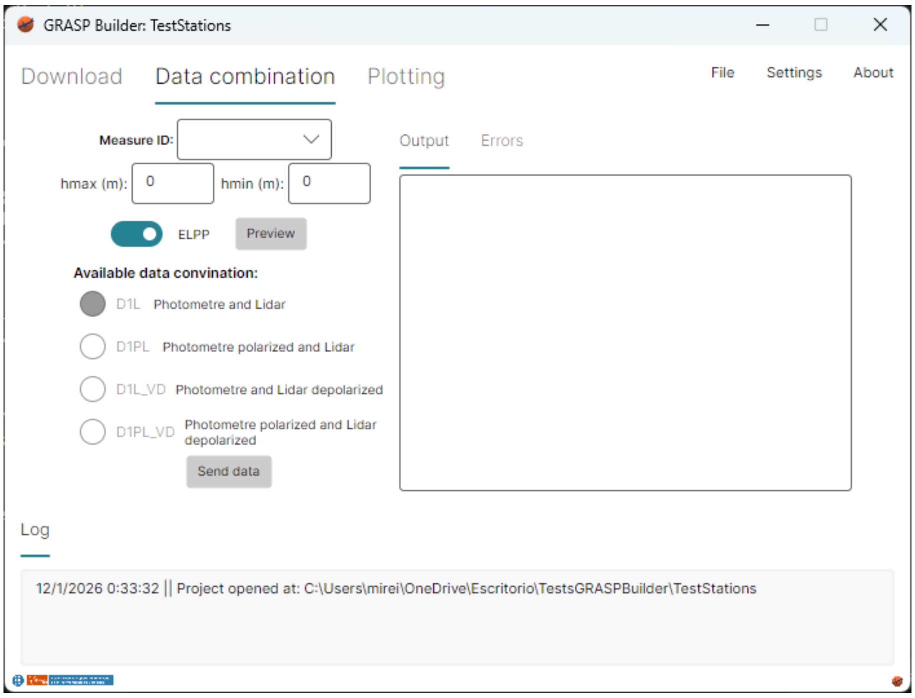
Station: BRC - Barcelona

Progress: 0%

- ✓ 20250203brc1033
- ✓ 20250203brc1830
- ✓ 20250204brc1822
- ✓ 20250205brc1030
- ✓ 20250206brc1830

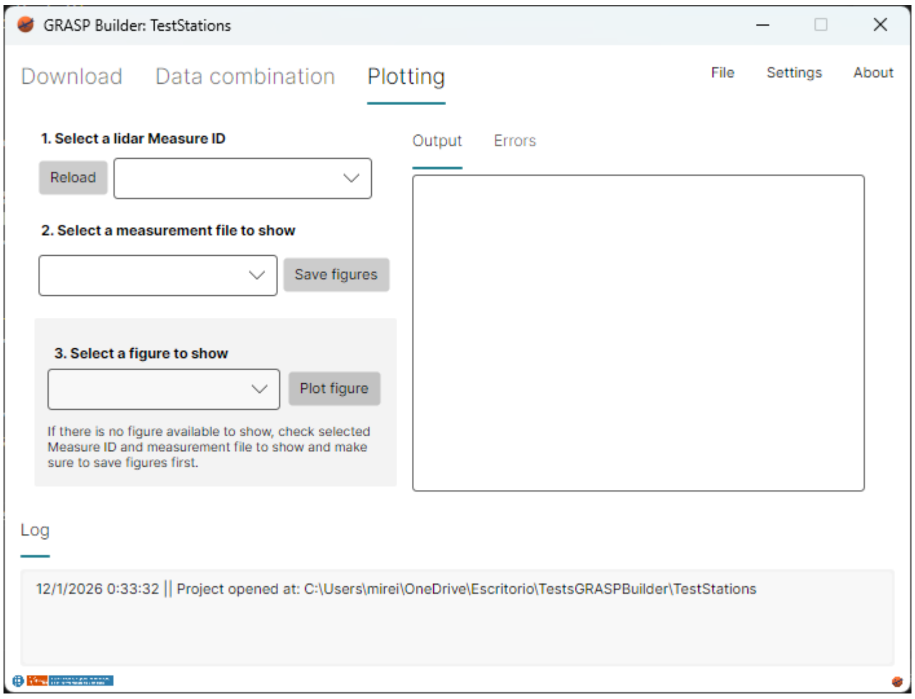
```
12/1/2026 0:33:32 || Project opened at: C:\Users\mirei\OneDrive\Escritorio\TestsGRASPBuilder\TestStations
```

Figure 26: Download view for low resolution UI


GRASP Builder: TestStations
Download
Data combination
Plotting
File
Settings
About
Measure ID:
hmax (m): 0
hmin (m): 0
ELPP
Preview
Available data convination:
D1L Photometre and Lidar
D1PL Photometre polarized and Lidar
D1L_VD Photometre and Lidar depolarized
D1PL_VD Photometre polarized and Lidar depolarized
Send data
Output
Errors
Log

```
12/1/2026 0:33:32 || Project opened at: C:\Users\mirei\OneDrive\Escritorio\TestsGRASPBuilder\TestStations
```

Figure 27: Data Combination view for low resolution UI


GRASP Builder: TestStations
Download
Data combination
Plotting
File
Settings
About
1. Select a lidar Measure ID
Reload
2. Select a measurement file to show
Save figures
3. Select a figure to show
Plot figure
If there is no figure available to show, check selected Measure ID and measurement file to show and make sure to save figures first.
Output
Errors
Log

```
12/1/2026 0:33:32 || Project opened at: C:\Users\mirei\OneDrive\Escritorio\TestsGRASPBuilder\TestStations
```

Figure 28: Plotting view for low resolution UI

This option allows the user to execute the application in a low resolution system, since this User Interface is smaller than the lowest resolution option.

11 Conclusions

References

[1] Wolfgang Skala. Drawing gantt charts in latex with tikz.

[2] Albert Einstein. Zur Elektrodynamik bewegter Körper. (German) [On the electrodynamics of moving bodies]. *Annalen der Physik*, 322(10):891–921, 1905.

[3] Michel Goossens, Frank Mittelbach, and Alexander Samarin. *The LaTeX Companion*. Addison-Wesley, Reading, Massachusetts, 1993.

[4] Donald Knuth. Knuth: Computers and typesetting.

[5] C.N.R. IMAA. Actris-earlinet data portal. Web Application available at `https://data.earlinet.org/earlinet/desktop.zul`, 2024.

[6] NASA. Aerosol robotic network (aeronet). Web Application available at `https://data.earlinet.org/earlinet/desktop.zul`, 2025.

[7] Miriam Martínez Canelo. ¿qué es la programación orientada a objetos? Web Application available at `https://profile.es/blog/que-es-la-programacion-orientada-a-objetos/`, 2025.

[8] MathWorks. Matlab. Web Application available at `https://es.mathworks.com/products/matlab.html`, 2025.

[9] C.N.R. IMAA. Actris-earlinet rest api. Web Application available at `https://data.earlinet.org/api/swagger-ui/`, 2025.

[10] NASA. Aerosol optical depth. Web Application available at `https://aeronet.gsfc.nasa.gov/print_web_data_help_v3_new.html`, 2025.

[11] NASA. Raw products aerosol optical depth. Web Application available at `https://aeronet.gsfc.nasa.gov/print_web_data_help_v3_raw_sky_new.html`, 2025.

[12] NASA. Aerosol inversions. Web Application available at `https://aeronet.gsfc.nasa.gov/print_web_data_help_v3_inv_new.html`, 2025.

[13] Microsoft. Mvvm. Web Application available at `https://learn.microsoft.com/en-us/dotnet/architecture/maui/mvvm`, 2025.

[14] Microsoft. Observableobject. Web Application available at `https://learn.microsoft.com/en-us/dotnet/communitytoolkit/mvvm/observableobject`, 2025.

[15] Microsoft. Relaycommand. Web Application available at `https://learn.microsoft.com/en-us/dotnet/communitytoolkit/mvvm/relaycommand`, 2025.

[16] GeeksforGeeks. Solid principles. Web Application available at `https://www.geeksforgeeks.org/system-design/solid-principle-in-programming-understand-with-real-life-examples/`, 2025.

[17] Microsoft. Download visual studio community. Web Application available at `https://visualstudio.microsoft.com/es/vs/community/`, 2025.

[18] Microsoft. Download .net. Web Application available at https://dotnet.microsoft.com/en-us/download/dotnet, 2025.

[19] Microsoft. What is .net? Web Application available at https://dotnet.microsoft.com/en-us/learn/dotnet/what-is-dotnet, 2025.

[20] MathWorks. Download matlab. Web Application available at https://es.mathworks.com/help/install/ug/install-products-with-internet-connection.html, 2025.

[21] CALCULA. Calcula. Web Application available at https://ondemand.tsc.upc.edu/pun/sys/dashboard, 2025.

[22] GitHub. Github. Web Application available at https://github.com/mireialopez0910/TFM/tree/main, 2025.

[23] Microsoft Learn. Polymorphism. Web Application available at https://learn.microsoft.com/en-us/dotnet/csharp/fundamentals/object-oriented/polymorphism, 2025.

[24] Microsoft Learn. Factory pattern. Web Application available at https://www.geeksforgeeks.org/system-design/factory-method-for-designing-pattern/, 2025.

[25] GRASP Open. Grasp. Web Application available at https://www.grasp-open.com/doc/ch01.php#idm73, 2025.

[26] AWS. Unit testing. Web Application available at https://aws.amazon.com/es/what-is/unit-testing/, 2025.

Appendices

Appendices may be included in your thesis but it is not a requirement.