D4.6 AGRICORE interface

agricore

Deliverable Number	D4.6
Lead Beneficiary	AAT
Authors	AAT, IDE
Work package	WP4
Delivery Date	M39
Dissemination Level	Public

www.agricore-project.eu

The Agricore project has received funding from the European Union's Horizon 2020 research and innovation programme under the Grant Agreement No. 816078

Document Information

Project title	Agent-based support tool for the development of agriculture policies
Project acronym	AGRICORE
Project call	H2020-RUR-04-2018-2019
Grant number	816078
Project duration	1.09.2019-31.8.2023 (48 months)

Version History

Version	Description	Organisation	Date
0.1	ToC definition	AAT	25-ene-2023
0.2	Content inclusion (first draft)	AAT	30-ene-2023
0.3	Revision and comments	IDE	09-feb-2023
0.4	Second draft	AAT/IDE	16-feb-2023
1.0	Final version (exportation and formatting)	IDE	27-feb-2023

Disclaimer

Executive Summary

AGRICORE is a research project funded by the European Commission under the RUR-04-2018 call, part of the H2020 programme, which proposes an innovative way to apply agent-based modelling to improve the capacity of policymakers to evaluate the impact of agricultural-related measurements under and outside the framework of the Common Agricultural Policy (CAP).

Within the AGRICORE suite of tools, its main graphical user interface (GUI) can be found. Developed as a cross-platform desktop application, this tool enables users to configure and launch simulations on agricultural policies using synthetic populations and the Agent-based model (ABM). In such simulations, the analysis of the effect of the different policy measures is carried out through the computation of a series of key performance indicators (KPIs). Once finished the simulations, these KPIs will be used to assist in the analysis and visualisation of the results, representing them graphically in interactive charts and maps that will help users and policymakers to draw conclusions and support decision-making processes.

This document presents an analysis of different tools for the visualisation of data in charts and maps with the aim of providing the platform with the most suitable tools based on the objectives, requirements and characteristics of the AGRICORE project. The document begins with an introductory chapter that analyses the need to explore these tools and why they are necessary for the project. Then, it continues with a chapter explaining and comparing, with advantages and disadvantages, each of the tools selected and analysed. Finally, it explains the tool chosen for data visualisation in AGRICORE.

A third chapter explains how the chosen solution will be used in the platform, and finally, a concluding chapter summarises the conclusions drawn and explores the next steps to be taken.

Abbreviations

Abbreviation	Full name
ABM	Agent-Based Model
AoI	Attributes of interest
ARDIT	Agricultural Research Data Index Tool
CAP	Common Agricultural Policy
DWH	Data Warehouse
FR	Functional Requirement
GUI	Graphical user interface
KPI	Key Performance Indicator
LDAP	Lightweight Directory Access Protocol
LP	Linear programming
MILP	Mixed-integer linear programming
MINLP	Mixed-integer non-linear programming
MIQP	Mixed-integer quadratic programming
NFR	Non-Functional Requirement
NLP	Non-linear programming
PD	Policy description
SP	Synthetic population
SPG	Synthetic population generation
SQL	Structured Query Language
TEO	Techno-economic orientation
WP	Work Package

List of Figures

Table of Contents

1 Introduction

As explained in previous deliverables, Agricore Project is a European project focused on innovation and data gathering in the European Agricultural sector with the overall aim of improving and evaluating its performance. In particular, this document is focused on the microservice highlighted in red in Figure 1, called the AGRICORE interface.

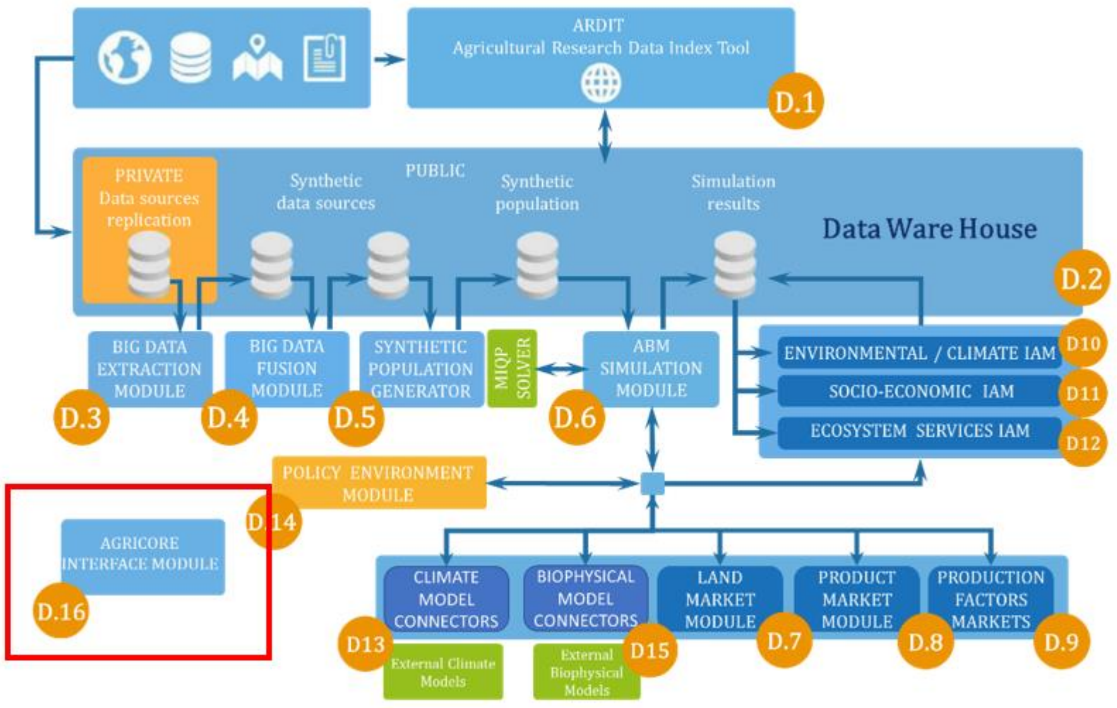


Figure 1. AGRICORE Modular Architecture.

As indicated in deliverable D6.1 - AGRICORE architecture and interfaces, the main objective of the AGRICORE Interface is to centralise user interactions with the overall project application suite and to provide an intuitive and user-friendly interface based on the latest web technologies. To achieve this, the AGRICORE interface establishes a connection to the necessary modules already developed in the previous deliverables that provide the data and services needed to perform the user inputs. Especially, the purpose of this document is to show in detail the tools used, architecture and methodology of use, which have been employed during the development phase of the AGRICORE Interface application.

2 Requirements of the AGRICORE suite

In this section, the requirements of the application will be listed. These requirements have been obtained through the analysis and study of the main objectives of the project described in the deliverable D4.1 - AGRICORE requirements and project management platform. These requirements are shown below, classified into functional and non-functional requirements.

2.1 Functional requirements

- **D16.FR.001. Centralise the interaction of the users with the AGRICORE suite**. The AGRICORE interface module should centralise all the interactions of the users with the AGRICORE suite.
- **D16.FR.001-1-1. Login.** Previously registered users with a valid account should be able to access the Suite by introducing their username and password. Once logged, they should be directed to the AGRICORE Suite Lobby view.
- **D16.FR.001-1-2. User Registration.** Users must be able to register and request an account for the AGRICORE Suite. To that end, they should provide a name, country, valid email address, and affiliation (company, institution). The registrant user might also agree to receive emails with news about the platform or other data sources.
- **D16.FR.001-1-3. Recover account credentials.** Users must be able to recover their credentials (normally the password) if they lose them.
- **D16.FR.001-2. Simulation Setup.** Users must be able to configure the simulation to be run. The user can start a new simulation setup, complete it, and launch it, or save an incomplete setup to resume it at a later time. To that end, it should be possible:
    - To save an incomplete setup file.
    - To save a complete setup file.
    - To load an incomplete setup file for finishing it.
    - To load a complete setup file to proceed directly with the simulation execution.
    - To launch a simulation once all their elements have been defined.
- **AG.D16.FR.001-2-1. Simulation Setup Page.** Users should be shown a screen (lobby) with as many buttons as components need to be configured in order to allow the simulation. These buttons should have an empty background (or another alternative colour) as long as the component to which they correspond has not been defined; once the user has configured the component, the background of the button should change to filled (or another alternative colour). If the user tries to close the SPG without downloading the setup file, a message should be prompted, warning about the possible loss of work. The 'Launch Simulation' button will be disabled as long as the simulation setup has not been completed (all its components have been defined).
- **D16.FR.001-2-2. Synthetic Population Selection.** General users (basic and advanced) should have two options (buttons/tabs/another alternative):
    - (a) One to select an existing population stored in a repository. In principle, this would be the user's private repository, stored in the space reserved for him/her in the DWH. However, it is proposed as a mere possibility the existence of a "public" repository, with a series of populations provided by their authors (public administrations, academic institutions, JRC or others) provided that the requirements of anonymisation, data protection and use of the datasets allow it.

Some filters (Geographical Scope, TEO, Economic Size, etc.) should be offered to discriminate among available populations, both in the private or public SP repositories.

  - b) Another option is to load a synthetic population directly from a continent file available on a physical medium. In this case, they should be provided with a pop-up window to the file browser to select the location of the file for uploading. For advanced users, a third option should also be provided that allows them, via a button, to open a new window with the Synthetic Population Generator interface.

- **D16.FR.001-2-3. Selection of Policies to be included in the simulation.** This window should have two subsections/tabs, depending on whether you want to perform a simulation in a positive (P) or normative (N) configuration.
  - In the case of positive configuration, the screen is divided into three sections:
    - P1: A 'General Catalogue' of Policy Descriptions (PDs), from where these descriptions, previously made public by their authors, can be imported to the user's Personal catalogue.
    - P2: A 'Personal Catalogue' of Policy Descriptions, containing both PDs imported from the General catalogue and PDs built by the user itself. Once in the Personal catalogue, the user can:
      - Add the policy described by a certain PD to the simulation setup,
      - Rate (evaluate) a PD imported from the General catalogue,
      - Delete a PD from the Personal catalogue.
    - P3: A 'Selected policies' section containing the policies to be actually incorporated into the simulation setup. Once a PD has been selected, the user can edit the main parameters of the Policy by directly editing the PD file.

  Adequate filters should be provided in all sections to discriminate among PDs. The options for the policy configuration case are yet to be defined.

- **AG.D16.FR.001-2-4. Selection of Simulation Period.** This is a very simple view/tab in principle, which should allow the user to simply define the number of years/agricultural seasons over which the simulation should run. It should be noted that the starting year must coincide with the year of the data on the basis of which the synthetic population has been constructed. This data could be extracted from the SP file and displayed in this view as well in order to have complete information on the temporality of the simulation.
- **D16.FR.001-2-5. Selection of Solver and Biophysical Model(s) for the Simulation.** This section, as well as the previous one, is a simple screen where the users can choose the solver and the biophysical model to be used in the simulation.
  - The solver is the mathematical software (external to AGRICORE) that is invoked during the simulation to solve the optimisation problem (constrained objective function maximisation) that controls the actions of each individual agent. The combined selection of the policies to be simulated and of the synthetic population makes the resulting optimisation problem of a specific type (LP, NLP, MILP, MIQP, MINLP, etc.). Once the system detects the type of resulting optimisation problem, it should provide the user with a list of commercial solvers capable of solving that particular type of problem. At this point, the user must select one of them and provide the AGRICORE simulation engine with a path (on his own computer or in a network location) to the executable file of the selected solver. If once a solver has been selected, changes are made to the selected SP or policies combo making

the resultant type of optimisation problem incompatible with the selected solver, and the system will display an error message.

- Biophysical models are simulation software elements (external to AGRICORE) whose function within the simulation environment is to allow each agent to predict the effect that control actions (agro-management decisions) (may) have on the biophysical state of their land and livestock. The selection of biophysical models is, therefore, directly dependent on the selected synthetic population, as these model(s) must be able to simulate the types of farming (crops and or livestock) that are relevant for the impact assessment to be carried out (those TFs present in the different synthetic agents).

- **D16.FR.001-2-6. Selection of default KPIs to be computed.** This view/tab should have, for all users, a selector that allows choosing one or several from a list of pre-existing KPIs. These selected KPIs will be computed automatically during the simulation and stored by default in the results file so that they can be directly displayed in the data display window(s) as soon as the simulation is finished. For advanced users, a selector should also be provided to take them to another view/tab from where they can define, using a controlled and predefined language and format, their own KPIs. Once created and selected, they will also be computed by default during the simulation.
- **D16.FR.001-3. My Simulations Module.** Users must be able to launch a simulation based on a simulation setup file. This file can be preloaded directly from the output of the simulation setup module, or it can be loaded manually by the user into the simulation module using a previously generated configuration file.

  This screen must have two sections:

  - S.1. Ongoing simulations must appear under the 'Simulations in progress' section, with their corresponding progress bar and percentage, and buttons to watch (live results), pause or cancel the simulation.
  - S.2. Previously completed simulations must appear listed under the 'Finished simulations' section, along with buttons to visualise the results (opening Visualisation Module), download the result data into a plaintext/rich text file, or delete the result data.
- **D16.FR.001-4. Visualisation Module.** Users must be able to visualise the results from a finished simulation. These results can be automatically immediately displayed after the end of a simulation, or uploaded by the user using a previously obtained results file. It should be possible to download each visualisation individually as an image file or all the visualisation together as a PDF report. Advanced users must have the possibility of opening a Jupyter IDE to create their own customised visualisations by directly processing the raw result data.
- **D16.FR.001-5. Synthetic Population Generator Module.** Advanced users should be able to generate their own Synthetic Populations using the interface and linking with ARDIT and with the Data Extraction, Data Fusion, Bayesian Network Generation and SPG Modules.
- **D16.FR.001-5-1. Synthetic Population Generator Configurator.** Advanced users landing on this window of the synthetic population generator should be able to configure at least the following parameters of the synthetic population to be generated:
  - Geographic Scope of the farms to be synthesised.
  - Technical-economic orientation(s) of the farms to be synthesised.
  - Economic Dimension Range of the farms to be synthesised.
  - Baseline year of the simulation, which will define the data to be used in the SPG process.

  - (Desirable) other parameters (e.g. gender of owners, etc.).
  - It should also be possible to configure how the remaining "non-interest" farms are modelled. That is, if a super-agent is created to group them all together, or by TEOs, etc.
  - Once all the parameters have been selected, the user should be able to launch a search for available data sources. This search is powered through ARDIT, and the result is a list of potential datasets to be used for each one of the Attributes of Interest of the agents.
- **D16.FR.001-5-2. Selection of DataSource for SPG.** The result of the (multi)query to ARDIT should be a view/tab listing all the Attributes of Interest (AoI) of each agent (according to the WP3 ABM definition). When selecting each attribute, the possible data source(s) available to extract the information to assign values to that attribute should be displayed. Ideally, there should be a mechanism (colours, ticks, others) to visualise for which attributes a data source has already been selected and for which ones the selection is still to be made. Once at least one data source has been selected for each AoI, a selector/button should be enabled to trigger the generation of the SPG itself.
- **D16.FR.001-5-3. Synthetic Population Generator (Down)Loader.** Once the synthetic population generation process is completed, this view/tab should allow the advanced user:
  - Download the generated population to his/her computer as a file with a format and extension yet to be defined.
  - Load the generated population in the Simulation Setup module and be redirected to the simulation setup lobby.
- **D16.FR.001-6. AGRICORE Suite Main Page.** Once logged, users must be able to directly navigate to any of the interface modules (simulation setup, synthetic population generator, simulation run, visualisation of results and (optionally) commuting to/from ARDIT). The user should also be able to access his/her Catalogue of Policies and their past executed simulations (under My Simulations).

2.2 Non-functional requirements

Non-functional requirements (NFR) define how the system should be performed through constraints or quality objectives that it must satisfy from the non-technical point of view. The non-functional requirement for the AGRICORE interface is the following point:

- **D16.NFR.001. Developed as a cross-platform desktop application web technologies.** The Agricore interface module will be implemented as a cross-platform desktop application using web technologies. The application should be compatible with the Chrome web browser.

3 AGRICORE interface architecture

This part of the document will detail the structure of the application, following the principles established in the AGRICORE project. The proposed solution for the AGRICORE interface is a desktop application that runs locally on a computer device, unlike web applications, such as the ARDIT tool, which are not accessible from any browser and require installation on the device where it will be used.

Some of the advantages of desktop applications are:

- **Efficiency:** as they are installed directly on the machine, these applications can manage the resources of the machine itself in a more appropriate way.
- **Privacy:** all the data processed by the machines are processed locally, without the need for interaction with third parties, which makes the information more secure.
- **Greater level of personalisation:** they can be configured according to preferences, depending on the demands of the users who decide to use them.

Taking into account the knowledge and experience acquired by the team in the development of the ARDIT tool. The proposed solution for the AGRICORE interface is a layer-based structure, which is shown below.

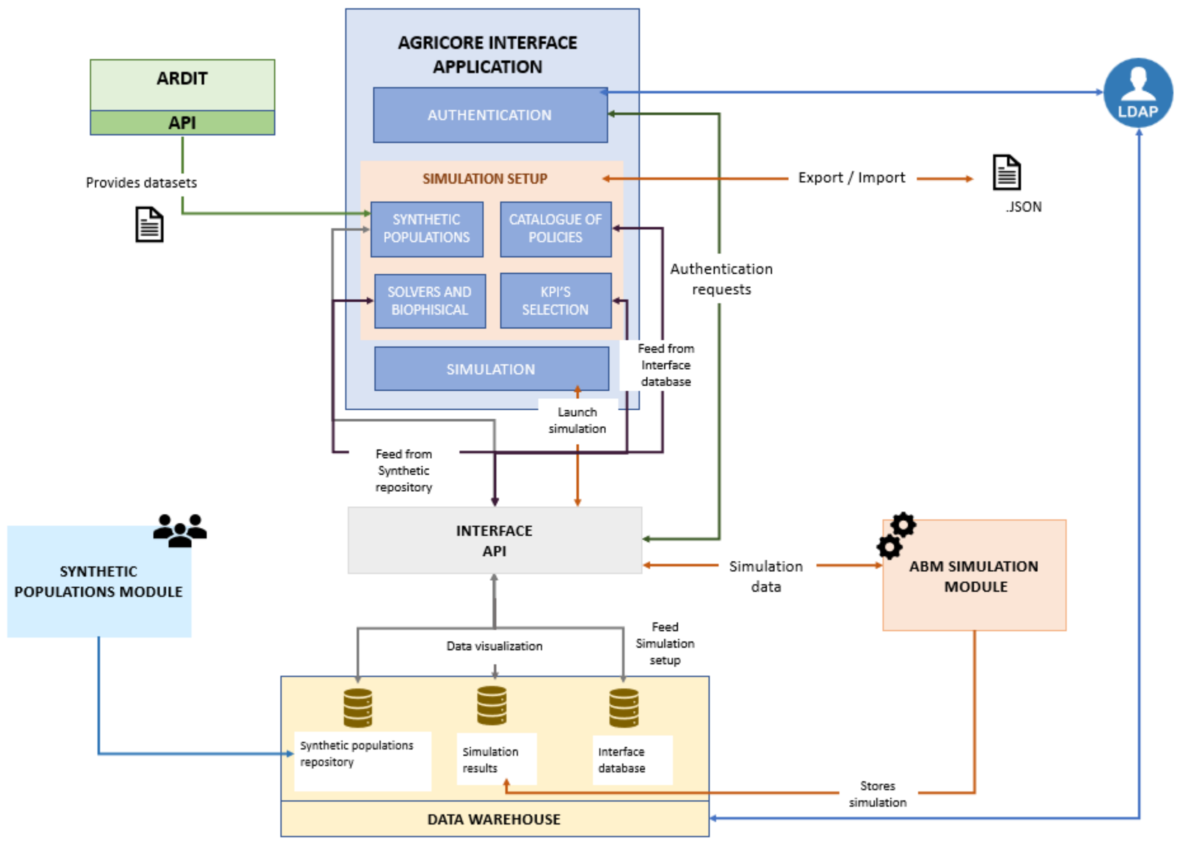


Figure 2. Layer-based structure for the AGRICORE interface.

In the diagram, the following parts may be distinguished:

- **The visual layer or FrontEnd** is the one destined to give the style to the application. In the diagram, it is represented as the **AGRICORE INTERFACE APPLICATION**, where users will be able to perform authentication operations and configure simulations in a simple way.

- **The logical or Backend layer**, which is formed by the INTERFACE API, manages all the communications and operations of the application.
- **DataWareHouse**, the storage unit where operations requiring Big Data processes will be carried out.

As can be seen in the diagram, these layers are complemented by a series of resources, which are as follows:

- LDAP tool, to manage users and credentials for both the application and the DataWarehouse, which will share users.
- PostgreSQL database, where the application data will be stored.
- Synthetic population module and ABM simulation module, microservices with which the AGRICORE interface communicates. The objective of the SPG module is to create the synthetic population required to carry out a simulation, which will be carried out by the microservice Synthetic population module. On the other hand, the microservice ABM simulation module will have the objective to execute and save the results of these simulations. These microservices will store the information in the Data WareHouse.

To communicate between the different layers of the application, the following structures will be created.

- REST API will be established, through which the different layers and resources of the application will communicate. This API will use the standard for the creation of tokens or keys that allow the authentication and authorisation of the user (identity and privileges held) or JSON Web Token (JWT) to guarantee security.
- Synchronizer will be a service that will be implemented in the backend, by means of which we will be able to synch the data that is stored in the Data Warehouse and store it in the database of the interface application (see Figure 3). This process will be carried out in the synthetic populations, with the objective of having this information available in the AGRICORE interface application.

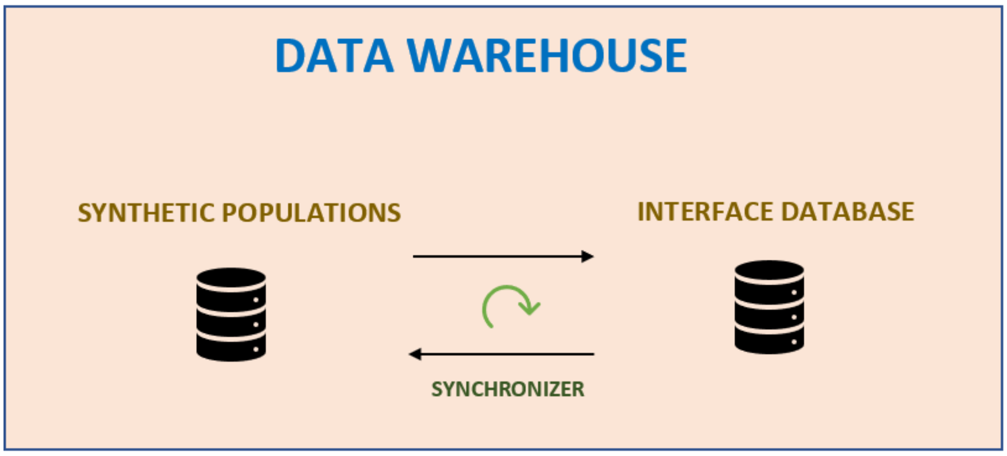


Figure 3. Diagram about how Synchronizer works.

Finally, after defining the structure, the set of technologies involved with the help of this diagram is briefly named. In Figure 4, the general concept of the application architecture and the technology chosen to produce each layer are shown.

- Visual layer, developed in Electron[1] and Angular[2], both of them are JavaScript frameworks which simplify the development of applications. Also, to enhance their functionality, these frameworks implement libraries, such as E-Charts, which makes the creation of charts an easy task.
- The logical layer will be developed with Spring Boot[3], and it will manage the entire application, doing operations such as managing and communicating with the microservices, the DataWarehouse and the front layer, as well as storing and modifying the data stored.
- DataWarehouse has been implemented using Hadoop technology, in which the results of the simulations, the generated synthetic population and the AGRICORE interface application database will be stored. In addition to these, it will be used to carry out the Big Data operations of the project.

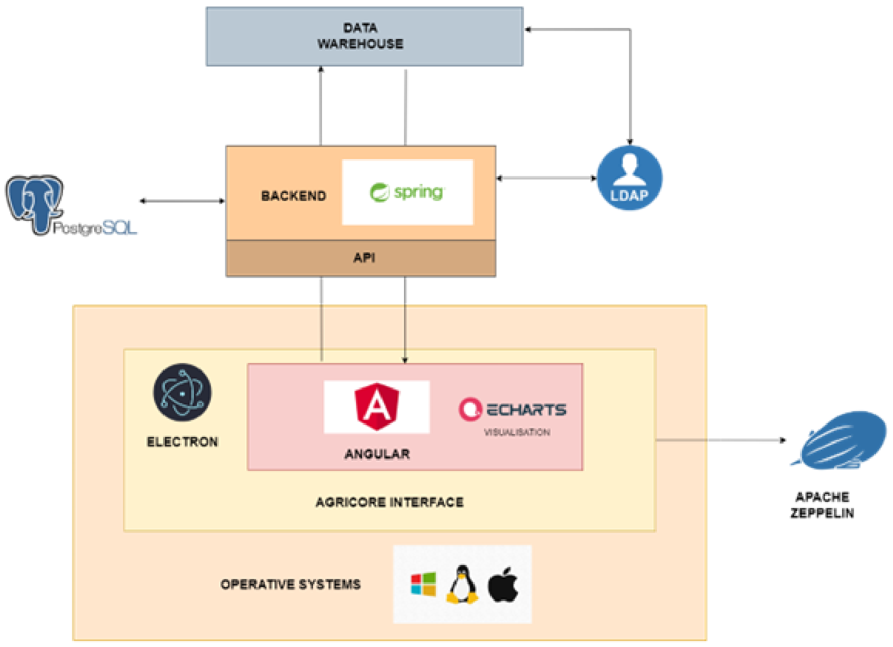


Figure 4. Technologies involved in architecture development.

3.1 Data model

In this section of the document, the structure of the database will be detailed. The database structure contains several tables that contain the data required for the AGRICORE interface. In Figure 5, we can see the structure of the data model and the tables that compose it.

[1] https://www.electron.build
[2] https://auth0.com/blog/creating-beautiful-apps-with-angular-material/
[3] https://spring.io/

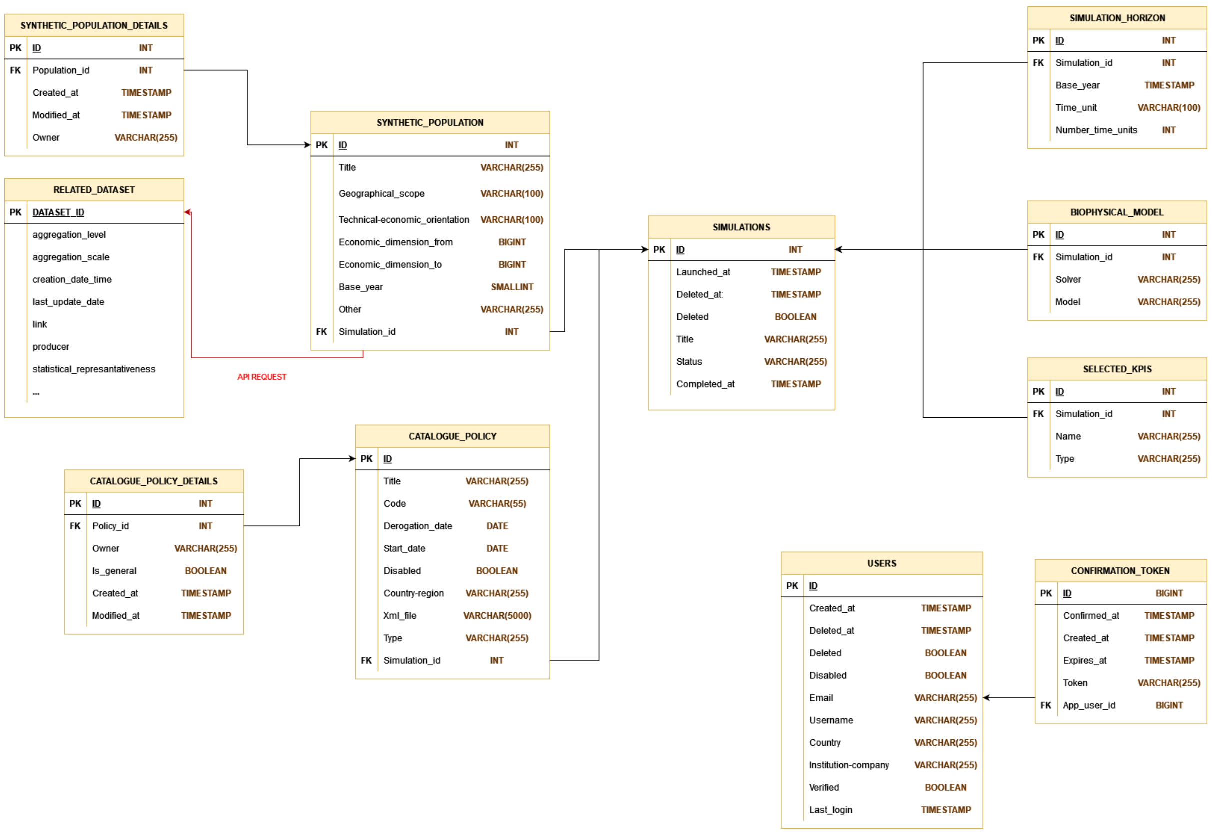


Figure 5. Structure of the data model and tables that compose it.

The contents of each of the tables in the diagram are briefly described below.

- **Simulations**: This table stores the properties of the simulations launched from the interface. Since simulations are the main objective of the application, they are related to the rest of the tables of the application.
- **Simulation horizon, Byophisical Model and Selected KPIs**: These tables store properties related to the simulations.
- **Synthetic population**: It stores the properties of the synthetic populations. It is related to the current simulation being created.
- **Synthetic population details**: This table stores data to be used for information purposes and that are not required by the synthetic population module. It is related to the parent synthetic population.
- **Catalogue policy**: Stores the properties of the policy catalogues. It is related to the current simulation being created.
- **Catalogue policy details**: Table that stores the informative data. It is related to the parent policy catalogue.
- **User**: It contains information related to the users of the application. With the information stored in this table, it is possible to know if the user is validated or banned, when was his last login, etc.
- **Confirmation token**: This table stores the different tokens that are generated in the application. This table has a relationship with the User table, which allows us to relate the tokens with the users of the application. Currently, there are two types of tokens:

  - Registration confirmation token generated during registration.
  - Password recovery token generated when we want to recover the password.

Finally, this data model will be implemented in the Postgres database of the AGRICORE interface hosted in the Data Warehouse.

3.2 Graphical User Interface

This section shows how our application, which will support the AGRICORE interface, will look like. At this point, it is worth remembering that the main objective of the AGRICORE interface is the centralisation of user interactions with the AGRICORE suite and to provide a user-friendly and intuitive interface based on the latest web technologies.

As explained in the development process, a procedure has been followed for the development of the application interface. The design has gone through the following phases:

- Study of tools similar to the ARGRICORE interface.
- Study of the design approach.
- Creation of the colour palette and establishment of the workflow.
- Study of the tools for the mockups of the application.
- Design of mockups.
- Development of the interface.

The process discussed above is documented in detail and can be consulted in D4.3 - Validated design for the AGRICORE interface.

Next, images of the already-developed interface application are exposed. The main goal in the development process was to develop the AGRICORE interface in the most faithful way compared to the mockups created in the design phase, taking into account that the images may vary as new functionalities and modifications are implemented in the current application.

- **Authentication screen,** this screen contains 2 fields, email and password. Once filled in with the correct information, it takes us to the main screen. Moreover, there is a checkbox with the “Remember me” option to remember the user each time we launch the application. In case of not being able to remember the user password, we can create a new account or recover the lost password.

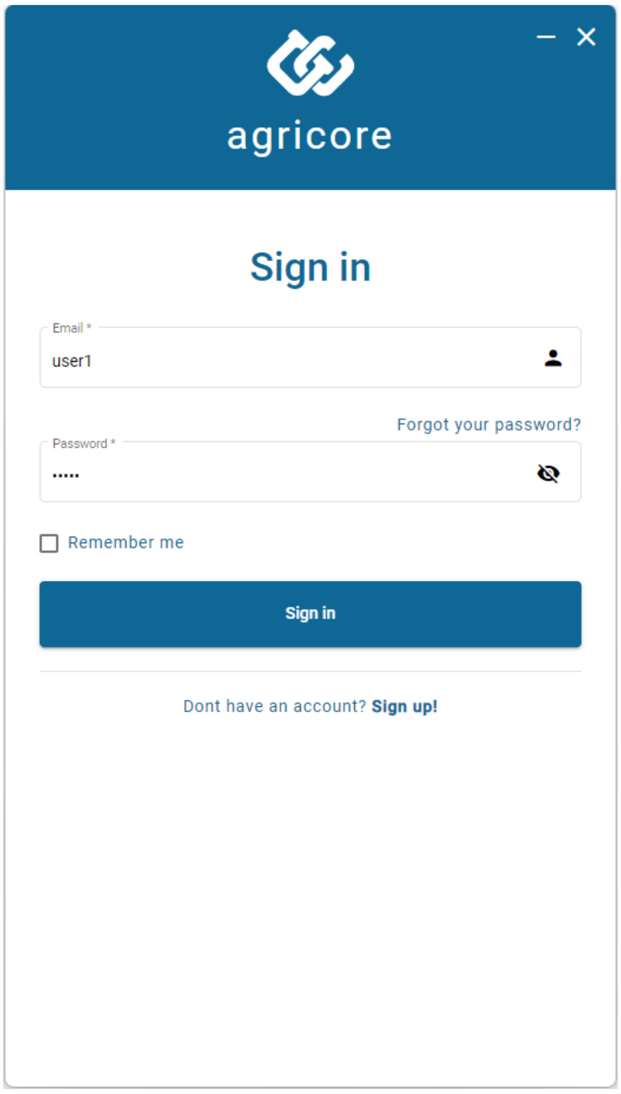
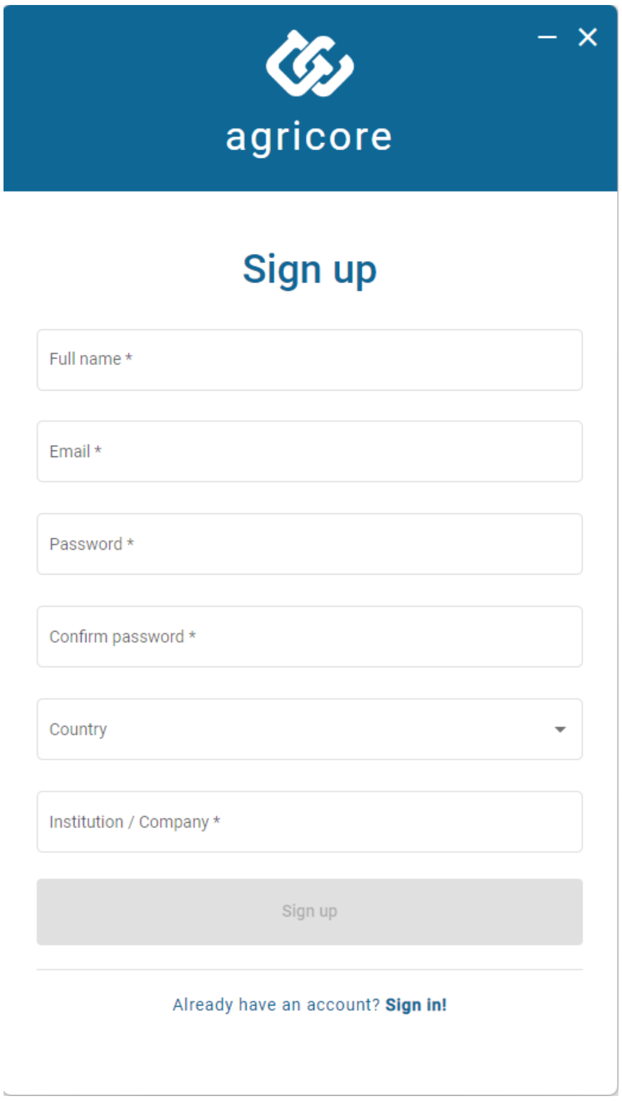


Figure 6. Authentication screen.

- **Registration screen**, on this screen, the standard registration fields of an application are shown. There are also 2 fields, “Country” and “Institution / Company”, to register this user data. Once the registration request is made, a confirmation email is sent to the user with a token that must be confirmed to register.



Sign up

Full name *

Email *

Password *

Confirm password *

Country

Institution / Company *

Already have an account? **Sign in!**

Figure 7. Registration screen.

- **Password recovery**, this screen only has an email field, the one linked to the account to be recovered, which will send a password recovery token to the user's email.

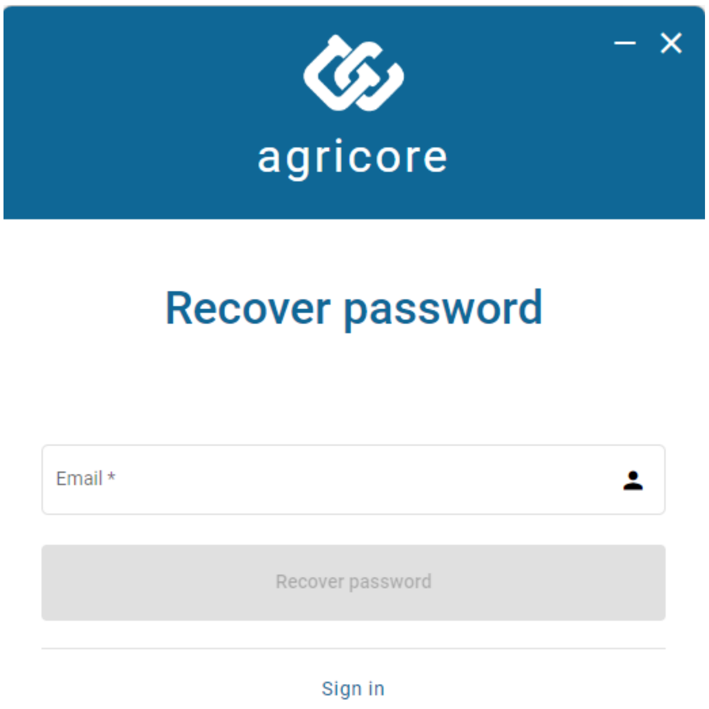


Figure 8. Password recovery screen.

- **Application top bar** provides a search bar, user information, and desktop application controls.

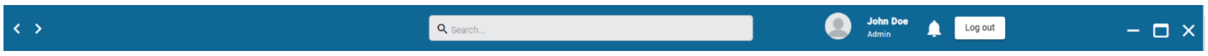


Figure 9. Application top bar.

- **Sidebar** with access to all sections of the AGRICORE interface.

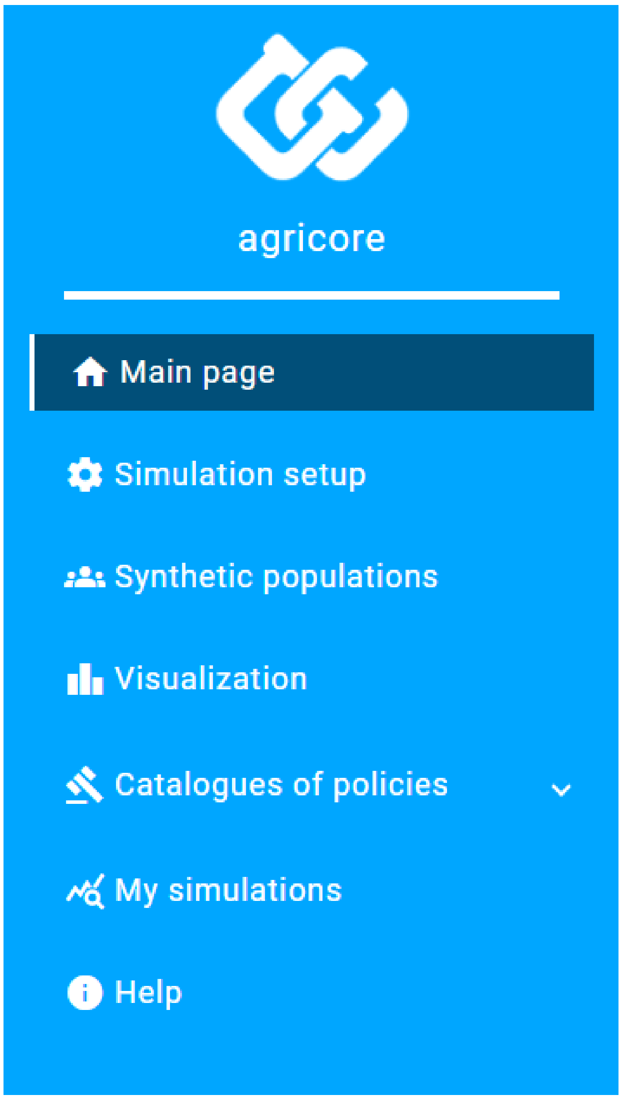
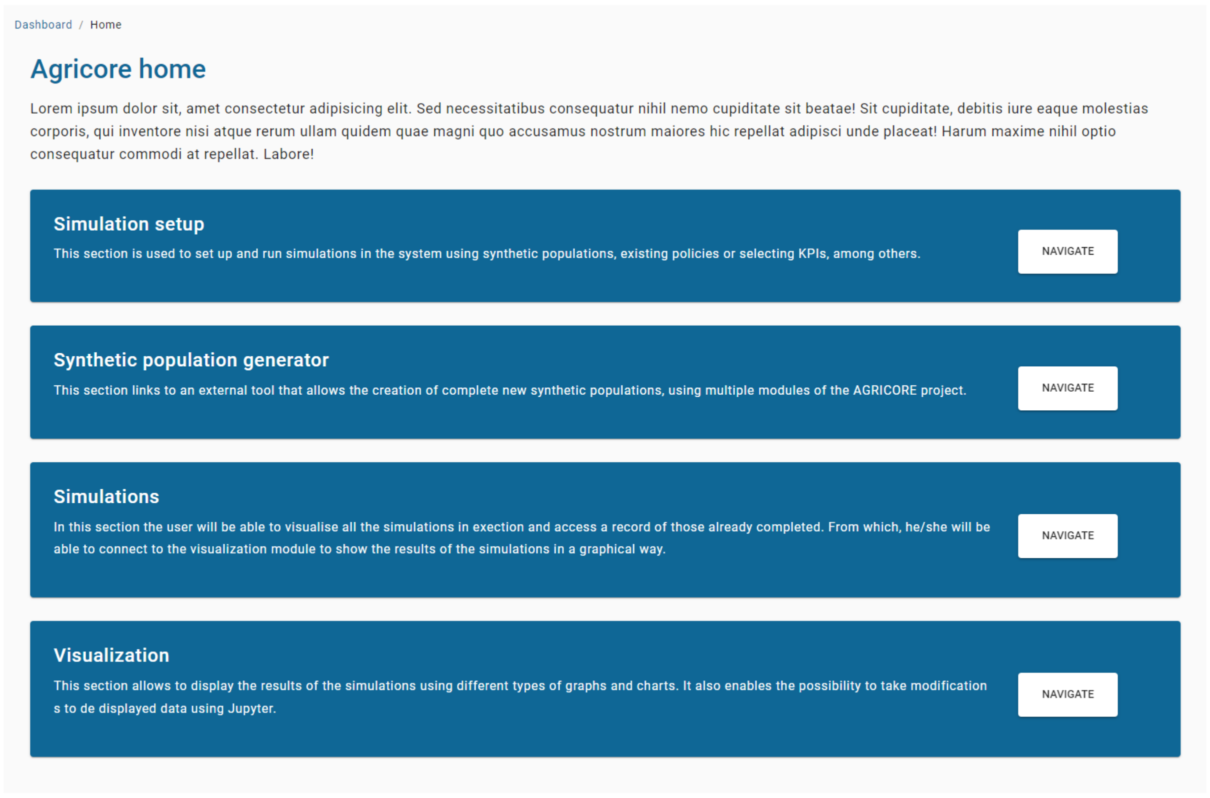


Figure 10. Sidebar.

- **Main screen** of the application. Here, there is a summary of the sections of the application with a direct access button to navigate to the different sections; in each section of the application, there is a brief explanation of the functionalities of the application screen.



Agricore home

Lorem ipsum dolor sit, amet consectetur adipisicing elit. Sed necessitatibus consequatur nihil nemo cupiditate sit beatae! Sit cupiditate, debitis iure eaque molestias corporis, qui inventore nisi atque rerum ullam quidem quae magni quo accusamus nostrum maiores hic repellat adipisci unde placeat! Harum maxime nihil optio consequatur commodi at repellat. Labore!

Simulation setup

This section is used to set up and run simulations in the system using synthetic populations, existing policies or selecting KPIs, among others.

Synthetic population generator

This section links to an external tool that allows the creation of complete new synthetic populations, using multiple modules of the AGRICORE project.

Simulations

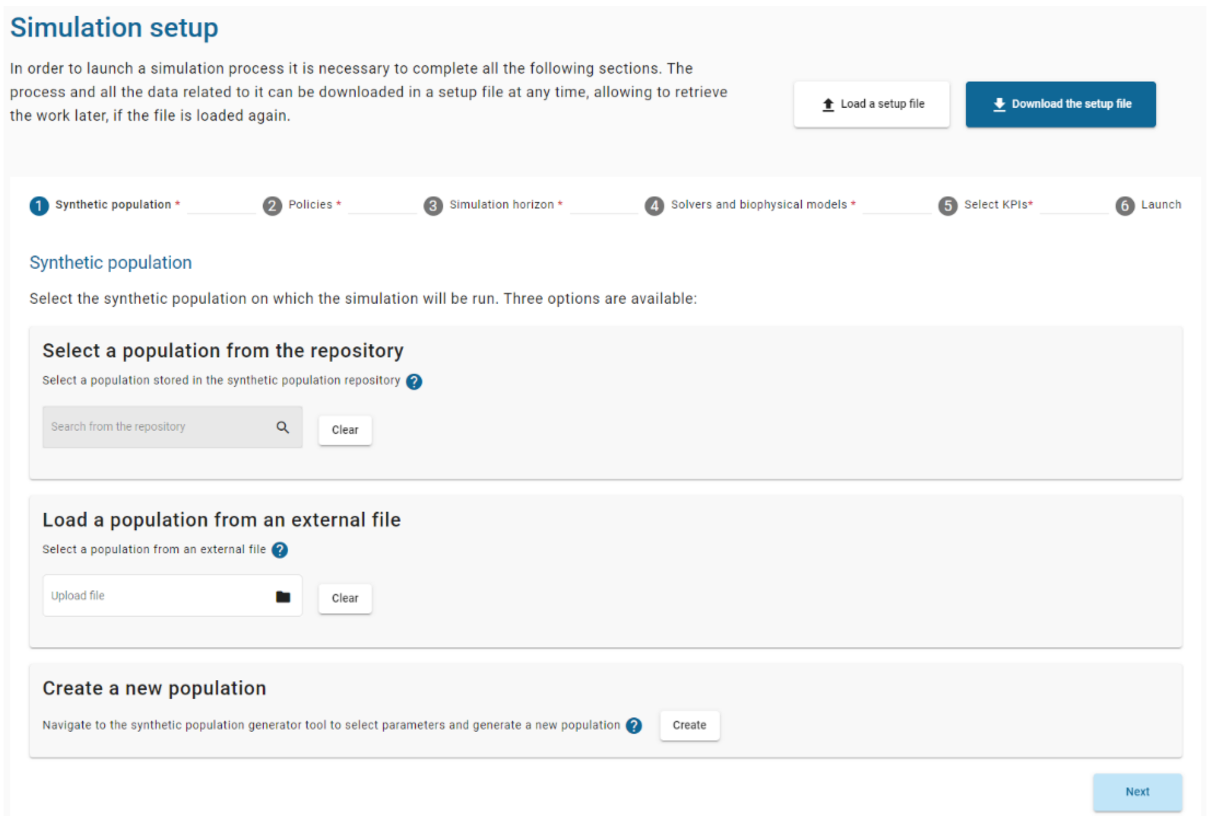
In this section the user will be able to visualise all the simulations in exection and access a record of those already completed. From which, he/she will be able to connect to the visualization module to show the results of the simulations in a graphical way.

Visualization

This section allows to display the results of the simulations using different types of graphs and charts. It also enables the possibility to take modification s to de displayed data using Jupyter.

Figure 11. Main screen of the application.

- Application screen that contains the **simulation setup**, with setup file upload and download buttons, the process has 6 steps to be filled out by the user that are stored. Once the 6th and last step is reached, the simulation will be launched if all the steps are filled properly.



Simulation setup

In order to launch a simulation process it is necessary to complete all the following sections. The process and all the data related to it can be downloaded in a setup file at any time, allowing to retrieve the work later, if the file is loaded again.

Synthetic population

Select the synthetic population on which the simulation will be run. Three options are available:

Select a population from the repository

Select a population stored in the synthetic population repository

Load a population from an external file

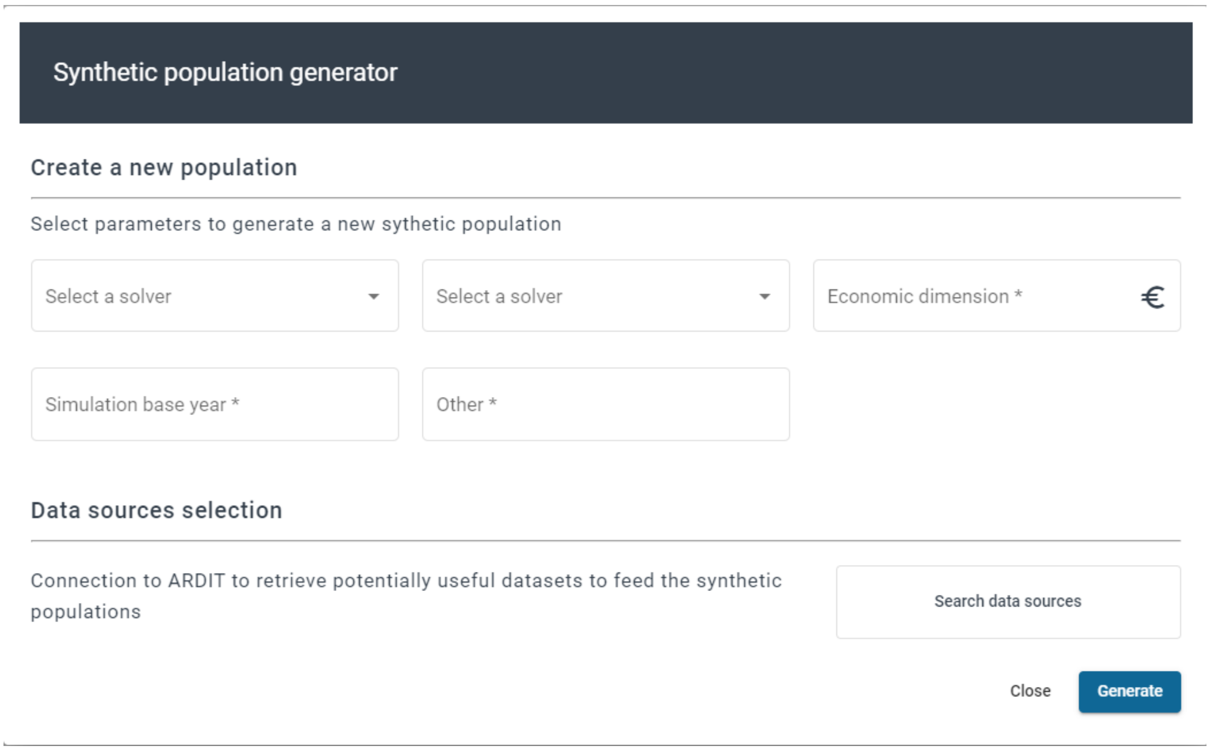
Select a population from an external file

Create a new population

Navigate to the synthetic population generator tool to select parameters and generate a new population

Figure 12. Simulation setup screen.

- Pop-up of the **synthetic population generator**, the example of an application pop-up for the creation of a new synthetic population with the fields to be chosen by the user and the option to make a connection to the ARDIT API and supply the synthetic population with information from the extracted datasets.


Synthetic population generator

Create a new population

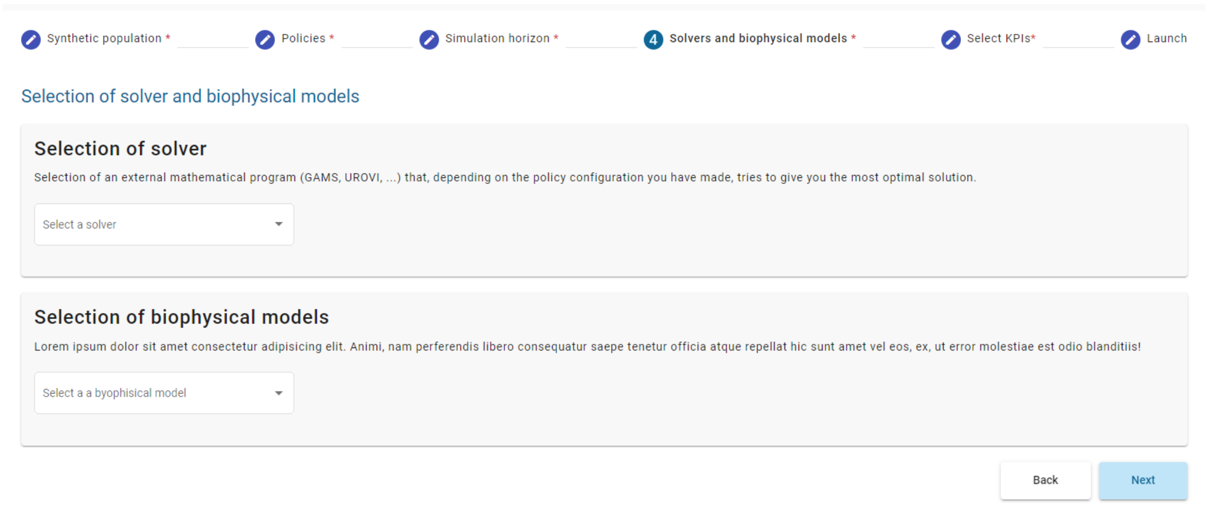
Select parameters to generate a new sythetic population

Data sources selection

Connection to ARDIT to retrieve potentially useful datasets to feed the synthetic populations

Figure 13. Pop-up of the synthetic population generator.

- Example of **simulation setup screen, selection of solvers and biophysical models.**



Selection of solver and biophysical models

Selection of solver

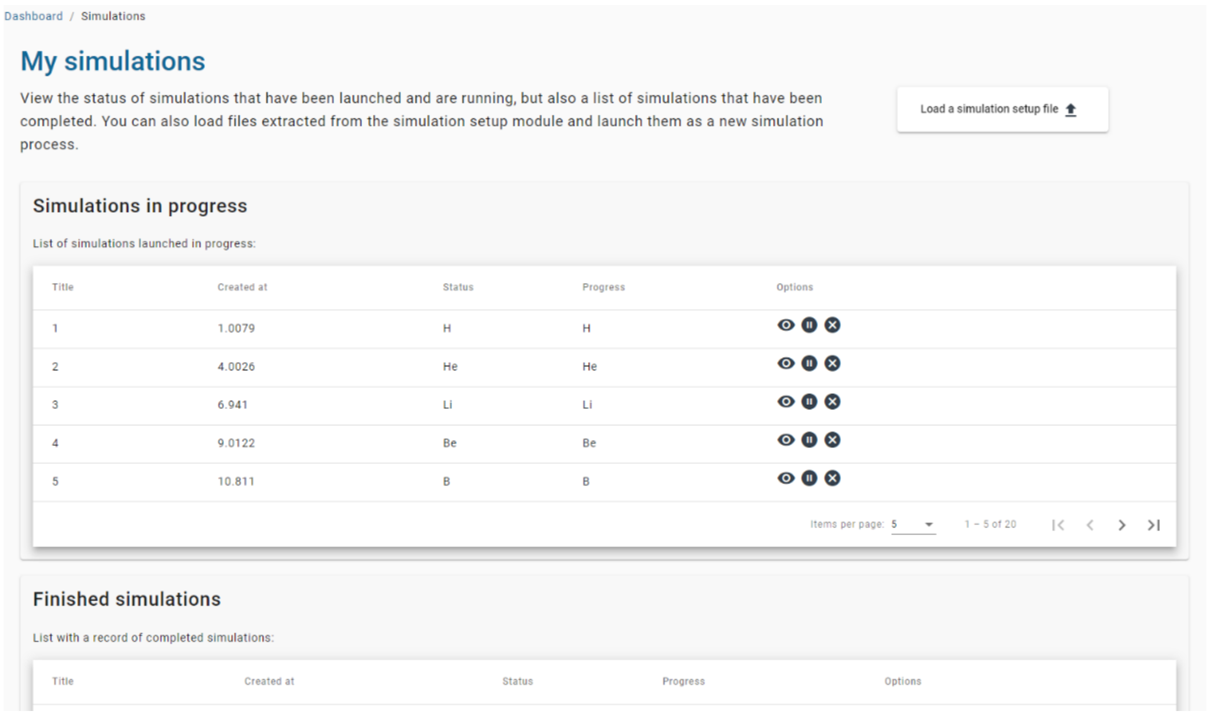
Selection of an external mathematical program (GAMS, UROVI, ...) that, depending on the policy configuration you have made, tries to give you the most optimal solution.

Selection of biophysical models

Lorem ipsum dolor sit amet consectetur adipisicing elit. Animi, nam perferendis libero consequatur saepe tenetur officia atque repellat hic sunt amet vel eos, ex, ut error molestiae est odio blanditiis!

Figure 14. Example of simulation setup screen, selection of solvers and biophysical models.

- **My simulations** screen, where we can visualise or control the launched simulations, pausing or deleting the already launched simulations. In addition, already created simulation setups can be uploaded to be relaunched.



Dashboard / Simulations

My simulations

View the status of simulations that have been launched and are running, but also a list of simulations that have been completed. You can also load files extracted from the simulation setup module and launch them as a new simulation process.

Load a simulation setup file

Simulations in progress

List of simulations launched in progress:

Title	Created at	Status	Progress	Options
1	1.0079	H	H	
2	4.0026	He	He	
3	6.941	Li	Li	
4	9.0122	Be	Be	
5	10.811	B	B	

Items per page: 5 1 – 5 of 20

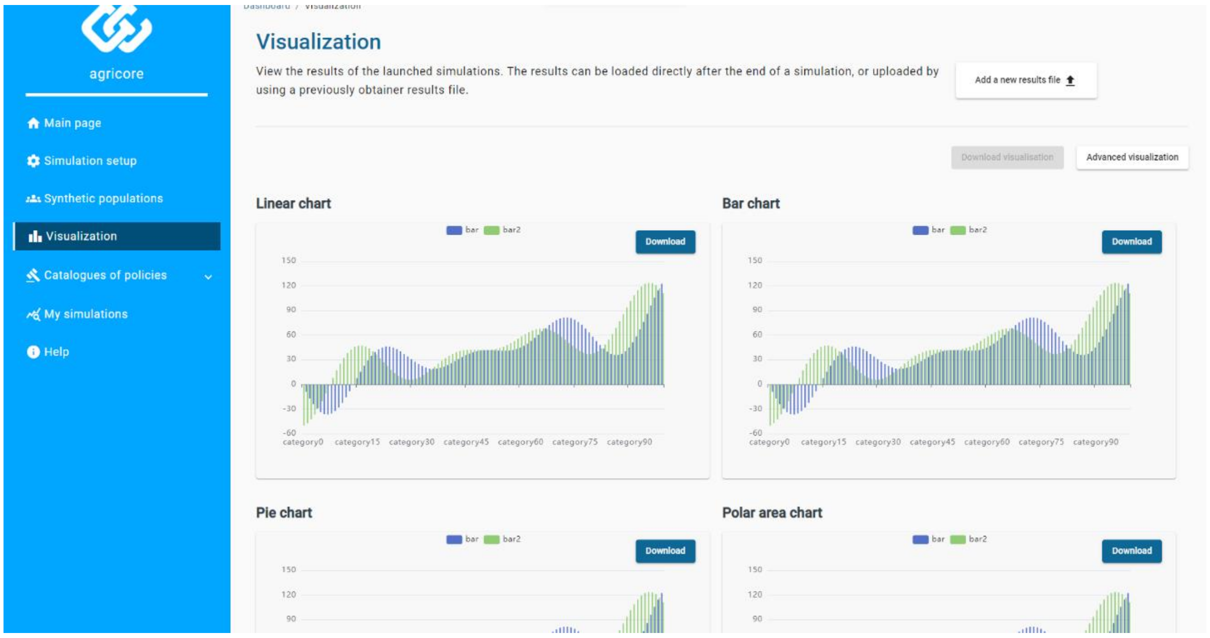
Finished simulations

List with a record of completed simulations:

Title	Created at	Status	Progress	Options

Figure 15. My simulations screen.

- **Visualisation** screen with several charts of the launched simulations. It consists of 4 charts, a linear chart, a bar chart, a pie chart, and a polar area chart. The screen contains a button to download the different charts, another button to upload a simulation, and an advanced graphics button to launch the Apache Zeppelin[4] application.



Visualization

View the results of the launched simulations. The results can be loaded directly after the end of a simulation, or uploaded by using a previously obtainer results file.

Figure 16. Visualisation screen.

[4] https://www.cloudera.com/products/open-source/apache-hadoop/apache-zeppelin.html

4 AGRICORE suite integrations

During the integration process, the integration with three modules of the application interface has been carried out. In this part of the document, it is explained how the different integrations have been carried out, as well as de difficulties encountered while doing this process in the AGRICORE Interface as described in deliverable D6.1 - AGRICORE Architecture and interfaces.

4.1 Data WareHouse

The objective of establishing the connection between the interface and the Data WareHouse is to be able to consult the information that resides in it. After studying the different possibilities of connecting with Data WareHouse, it has been decided to implement this connection through the Hive layer for the following reasons:

- Similarity with SQL in syntactic terms.
- Ease of integration with the Backend, via JDBC client.

Once the technology to interact with the Data WareHouse has been selected, it will be integrated into our environment. Specifically, this integration has been carried out in the Backend part of the project, since we had already installed the JDBC client. Finally, to establish the connection, it was only necessary to indicate to the JDBC client, already installed, the connection string, together with the credentials. During this process, the biggest obstacle encountered was the documentation available explaining how to connect to the HIVE technology via the JDBC client, which made this integration process not an easy task.

4.2 ARDIT

The AGRICORE interface connects with ARDIT in order to consult the datasets that are currently available and then assign them to a synthetic population. This integration has been carried out via API. Taking advantage of the fact that the ARDIT application provides a REST API. After deciding on the form of integration, it is sufficient to define the integration process, which is based on making requests to the endpoints supplied by ARDIT on the datasets from the FrontEnd layer of the application.

4.3 ABM simulation module

The objective of integrating with the ABM simulation module is based on being able to run simulations from the AGRICORE interface. Once again, it has been decided to carry out this integration via API REST, where the ABM simulation module provides an API to which the AGRICORE interface connects and via an endpoint, the AGRICORE interface provides all the information necessary to carry out a simulation. In the last two integrations, we did not have many problems since the teams had experience developing REST APIs, and the communication between the stakeholders was fluent and agile.

5 Development process

The objective of this section is to perform a review of the stages that the tool has undergone through its development. Throughout the development period, the tool has been continuously evaluated by the various project partners for analysis and monitoring, thus helping its possible improvement or detecting its usability errors.

1. **Creation and study of the requirements:** This is the first phase of the project, in which the functional and non-functional requirements of the application were defined.
2. **Prototyping of the application**: This phase of the project encompasses the complete process of the design and usability principles of the application. The application has been designed based on the standards of a multiplatform desktop application and good practices focused on user experience. Before starting the prototyping process, a preliminary study was carried out on the design tools that meet the requirements imposed by the EU and, analysing the advantages and disadvantages, “MarvelApp” was chosen mainly because of its collaborative functionality. Once the tool has been selected, we start by defining a palette of blue and greyish colours for the application, as well as a diagram of the workflow of the end users, including permission restrictions. It ends with a design validation plan that consists of two phases. The first phase is an internal validation by the project partners, who establish their level of satisfaction through a form. The second phase is the same process as the first one but with potential users, agents and experts external to the project.
3. **Creation of the environment and the base project:** Once the architecture has been defined, the environment in which the application will communicate is created. This environment is made up of:
    a. LDAP, which will be in charge of managing the application’s users.
    b. PostgreSQL, persistence unit in which the application data will be stored.

All this environment with which the application interacts has been dockerised with the aim of facilitating the installation of this environment in any type of device. Once the entire environment has been built, we begin to create the base versions of the interface and the business logic.

1. **For the business logic or Backend**, a project has been created in Spring Framework that provides an API, to which the interface will be connected. During the creation of the base project, the connections to the environment have been established, which include the connection to LDAP, PostgreSQL, along with the Data WareHouse connection.
2. **For the graphical interface or FrontEnd**, an angular project based on Electron has been created with the aim of generating a cross-platform desktop application.
3. **Application user management:** After creating the base projects, the first step consists of creating the application’s user management module. The user management module is responsible for providing access to the application, as well as managing access to the different resources of the application, depending on the user’s role.
4. **Application development:** During the course of this phase, the development of the application has been carried out.
    a. On the user interface side, the screens proposed in the mockups for the AGRICORE interface application, which are defined in deliverable D4.3 - Validated design for the AGRICORE interface, have been developed.
    b. On the backend, all the necessary logic has been generated to fulfil the application requirements defined in the requirements section of this document.

   c. Once the development of all the functionality has been completed, the different microservices involved in the AGRICORE interface have been integrated. This integration will be detailed in the integrations section of this document.

5. **Testing phase**: A testing plan is proposed and performed, including unit, integrated, functional and user tests.
6. **Preparation for use**: After concluding the developments, the last step would be to generate the executable with the application developed in Electron so that any user could install it on their personal computer.
7. **Development of the deliverable**: Once all the development is finished, we proceed to generate the deliverable detailing how the project has been developed.

As part of the methodology to follow in the development, meetings have been held with the project leaders to monitor and review progress and clarify doubts focused on specifying aspects of integration with the rest of the modules of the Agricore Suite. In this way, the project partners are involved in an Agile methodology, meaning that they are involved in a constant collaboration in a project that can be continuously improved upon throughout its life cycle.

6 Programming languages, tools and libraries

6.1 Backend

As mentioned in the architecture section of this deliverable, based on the knowledge acquired during the development of the ARDIT tool, it has been decided to implement all the business logic using the same technologies that were used to create the Backend of the ARDIT tool.

- **LDAP**: As defined in D1.9 - Agricultural Research Data Index Tool (ARDIT) ANNEX II: Frameworks programming languages, tools and libraries in the section on LDAP, it is a tool that stores information in a tree structure. Its function, as in ARDIT, is to manage the users of the application. This tool is in charge of storing the information related to the user in a secure way and isolated from the rest of the information. In order to carry out this user management, the user's name, associated password and roles are stored, which allows the user's access to the different resources of the application to be regulated according to their role.
- **Spring Framework:** This is a Java-based framework that allows the development of applications. For more information, please consult D1.9 - Agricultural Research Data Index Tool (ARDIT) ANNEX II: Frameworks programming languages, tools and libraries in the Spring Boot section. Spring Framework will be used to develop all the business logic of the AGRICORE interface, in which queries, creations, modifications and deletions of the different application data will be managed.
- **Endpoints:** The AGRICORE interface tool will provide an API with a set of endpoints that can be queried. These endpoints will be documented using the Swagger guidelines, as defined for ARDIT in document D1.9 - Agricultural Research Data Index Tool (ARDIT) ANNEX II: Frameworks programming languages, tools and libraries in the Endpoints section.
- **Docker[5]:** This is the technology that has been selected to manage the different deployments of the AGRICORE interface application, following the guidelines that were established for the ARDIT and are defined in the document D1.9 - Agricultural Research Data Index Tool (ARDIT) ANNEX II: Frameworks programming languages, tools and libraries in the Docker section.
- **PostgreSQL[6]:** As the main data persistence unit for the application, we have a PostgreSQL database. This technology was defined in section D1.9 - Agricultural Research Data Index Tool (ARDIT) ANNEX II: Frameworks programming languages, tools and libraries in the PostgreSQL section. This tool will be used to store all the data of the AGRICORE interface application.

6.2 Frontend

The AGRICORE interface has been designed as a cross-platform desktop application according to the requirement **AG.D16.NFR.001**. Figure 4 shows the implementation architecture used to make this possible. After the approval of the mockups made to define and adjust the design of the graphic interface, its implementation has been carried out through Angular CLI (command-line interface), which allows the development of the graphical interface and from which the use of different libraries is made to establish connections to API services, integrate different modules and include multiple application assets.

[5] https://www.docker.com/
[6] https://www.postgresql.org/

The libraries used to provide the functionalities required by the AGRICORE interface are listed below:

- **Angular CLI (Component Library)**: This library is used to facilitate the design of the interface thanks to its extensive collection of components, which are highly used in the whole application.
- **Ngx echarts**[7]: Apache echarts in Angular library mode, allows rendering the different graphics that are displayed in the "Visualisation" section of the application.

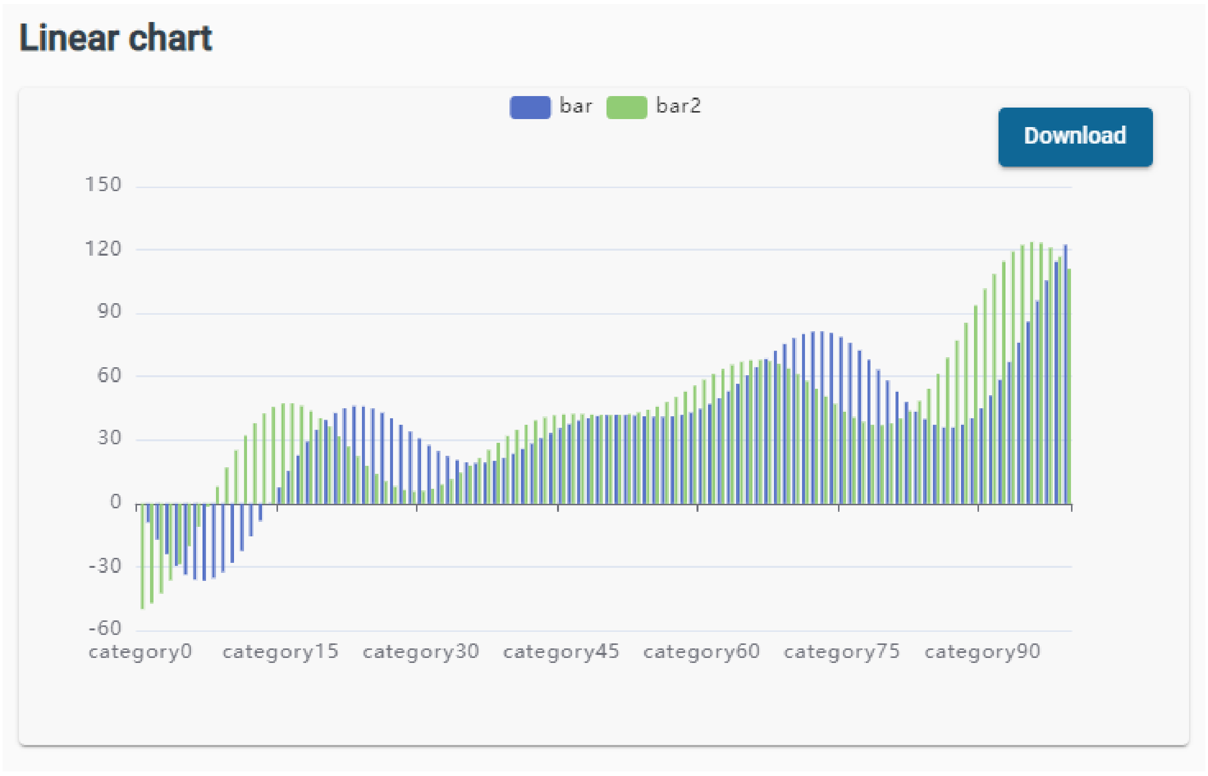


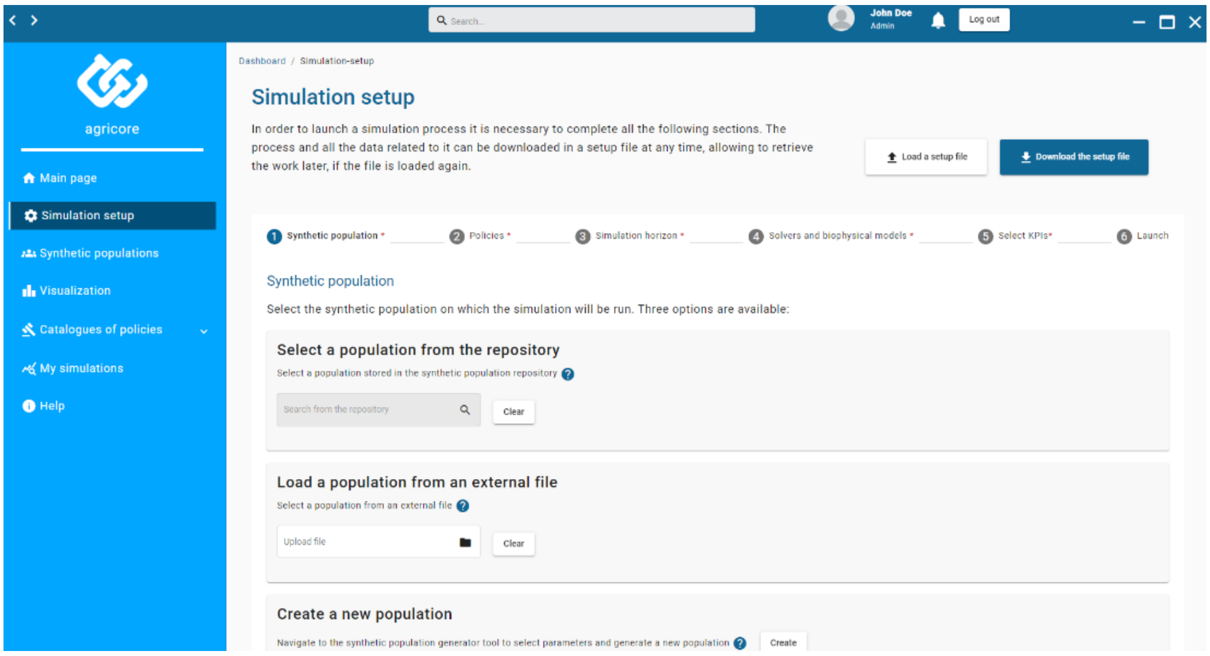
Figure 17. Example of bar chart with Ngx echarts.

- **Apache Zeppelin**: It is used to display graphics with advanced visualisation options from the "Advanced Visualization" button, which is accessed in the "Visualisation" section. This component allows you to enable functionalities to modify the display of the graph through the configuration of the different parameters. It is important to note that advanced technical knowledge is required to modify the visualisation, as it is an analytical tool.
- **Xng breadcrumb**[8]: Component used to implement simple breadcrumbs (menus) at the beginning of each section and facilitate navigation through the interface.

Figure 18. Example of breadcrumb with Xng breadcrumb.

[7] https://www.freakyjolly.com/angular-e-charts-using-ngx-echarts-tutorial/
[8] https://github.com/udayvunnam/xng-breadcrumb

Once the libraries were installed, the different sections of the application were developed based on (references to the graphical user interface), making all the connections to obtain the data to be displayed in the interface, mainly this data is obtained from the Data Warehouse (DWH link) and its different databases stored in it, as well as access to it to perform CRUD operations. From the interface, calls were also made to the microservices of the external modules, such as simulations or the creation of synthetic populations.



Simulation setup

In order to launch a simulation process it is necessary to complete all the following sections. The process and all the data related to it can be downloaded in a setup file at any time, allowing to retrieve the work later, if the file is loaded again.

Synthetic population

Select the synthetic population on which the simulation will be run. Three options are available:

Select a population from the repository

Select a population stored in the synthetic population repository

Load a population from an external file

Select a population from an external file

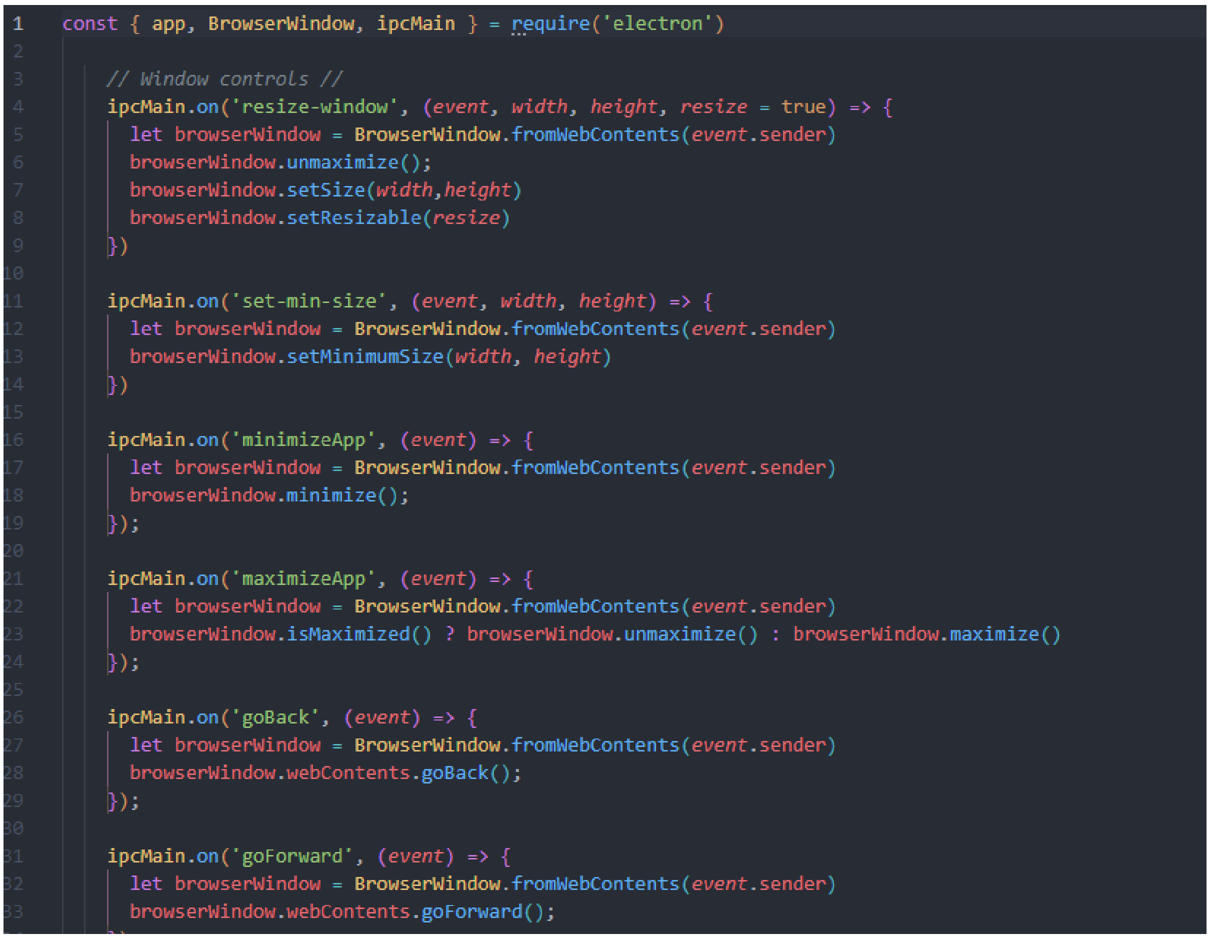
Create a new population

Navigate to the synthetic population generator tool to select parameters and generate a new population

Figure 19. Definition of the synthetic population.

To accomplish the requirement **AG.D16.NFR.001**, it is necessary to install an additional layer to run the previous development on different operating systems, compatible with browsers based on chromium technology and open source, for this Electron is installed, with the advantage that we can continue to develop with Javascript technology.

With this, the application is able to run on operating systems. Previously, it has been created in angular, the standard controls that offer desktop applications. These controls can be executed thanks to the connection of angular services with the native ‘ipcMain’ of Electron. To highlight the app.js file that contains the logic that can revise in Figure 20.



```javascript
const { app, BrowserWindow, ipcMain } = require('electron')

    // Window controls //
    ipcMain.on('resize-window', (event, width, height, resize = true) => {
      let browserWindow = BrowserWindow.fromWebContents(event.sender)
      browserWindow.unmaximize();
      browserWindow.setSize(width,height)
      browserWindow.setResizable(resize)
    })

    ipcMain.on('set-min-size', (event, width, height) => {
      let browserWindow = BrowserWindow.fromWebContents(event.sender)
      browserWindow.setMinimumSize(width, height)
    })

    ipcMain.on('minimizeApp', (event) => {
      let browserWindow = BrowserWindow.fromWebContents(event.sender)
      browserWindow.minimize();
    });

    ipcMain.on('maximizeApp', (event) => {
      let browserWindow = BrowserWindow.fromWebContents(event.sender)
      browserWindow.isMaximized() ? browserWindow.unmaximize() : browserWindow.maximize()
    });

    ipcMain.on('goBack', (event) => {
      let browserWindow = BrowserWindow.fromWebContents(event.sender)
      browserWindow.webContents.goBack();
    });

    ipcMain.on('goForward', (event) => {
      let browserWindow = BrowserWindow.fromWebContents(event.sender)
      browserWindow.webContents.goForward();
```

Figure 20. Javascript file with the standard controls that offer desktop applications.

This file contains the main Electron thread along with the application controls and various functions, such as window resizing. Finally, the electron builder library is added, which allows packaging the whole application in installers for different architectures and operating systems, as well as allowing portable packaging of the application. When it comes to use, the AGRICORE interface is a standard application with few but complete sections that cover the needs in an effective and friendly way for the final users. The application is governed by a system of users with registration, login and password recovery, which have certain permissions to be able to act or not on the sections of the application.

7 Conclusions

The deliverable' D4.6 - AGRICORE interface' provides a detailed guide to the application, the understanding of how it has been built, its architecture and the methodology that has been followed to reach the end of the application development where the many other deliverables converge to present the end user with an interface. The launch of the tool allows the user to create through the simulation process to configure each section and its configuration parameters. Once all the necessary configuration has been filled in, the simulation can be launched. In the case of not having finished the configuration of the simulation, a file can be saved that can later be loaded to continue filling in.

In this configuration process, it is also possible to create, edit and choose aspects such as synthetic populations or policy catalogues thanks to the sections of these that are found within the application. Once the simulation is launched in the 'my simulations' section, the state in which it is shown, having pause or delete controls. When the simulation is finished, the results are depicted in various graphs in the 'visualisation' section, or advanced graphs of this, thanks to Apache Zeppelin, are available, as well as being able to load previously completed simulations.

Although most of the developments in reference to this deliverable are complete and have passed the established test plans, the application is open to future developments integrating new functionalities or improvements of the current ones or fixes of future bugs that may be found.

For preparing this report, the following deliverables have been taken into consideration:

Deliverable Number	Deliverable Title	Lead beneficiary	Type	Dissemination Level	Due date
D6.1	D6.1 AGRICORE architecture and interfaces	IDE	Report	Public	M18
D2.4	D2.4 Synthetic population generation module	AAT	Other	Public	M39
D4.3	D4.3 Validated design for the AGRICORE interface	AAT	Report	Public	M21
D4.7	D4.7 Report on testing and validation activities of the AGRICORE interface	AAT	Report	Public	M39
D1.9	D1.9 European data sources index module	AAT	Other	Public	M31