DSSAT-Lite: a web-based application for running crop models and analyzing results from DSSAT-CSM

Jonas de Abreu Resenes[1], **Gerrit Hoogenboom**[2],
Alexandre Lazaretti Zanatta[1], **Willingthon Pavan**[1,3], **Carlos Amaral Holbig**[1]

[1]Graduate Program in Applied Computing (PPGCA)
University of Passo Fundo (UPF)
Campus 1 – 99.052-900 – Passo Fundo – RS – Brazil

[2]University of Florida
Gainesville – FL – United States of America

[3]International Fertilizer Development Center (IFDC)
Muscle Shoals – AL – United States of America

jabreu.ar@gmail.com, gerrit@ufl.edu

zanatta@upf.br, wpavan@ifdc.org, holbig@upf.br

Abstract. *DSSAT is a set of tools that facilitate the creation and management of experiment, soil, and weather files. Its tools were built using programming languages like Visual Basic and Delphi, which are difficult to run remotely or in the cloud. In this work, we present DSSAT-Lite, a web-based application designed to facilitate the simulation of DSSAT-CSM models over HTTP. It provides simulation and visualization tools and integration with highly optimized third-party software for processing DSSAT files. Abstraction layers allow for a standardized methodology for reading multiple DSSAT files into a beautifully formatted data structure. These features allowed DSSAT-Lite to run remote simulations, specifically using mobile devices and web browsers.*

Resumo. *DSSAT é um conjunto de ferramentas que facilita a criação e gerenciamento de arquivos de experimentos, solo e clima. Suas ferramentas foram construídas utilizando linguagens de programação como Visual Basic e Delphi, que se tornam difíceis de serem executadas remotamente ou na nuvem. Neste trabalho, apresentamos o DSSAT-Lite, uma aplicação baseada na web, projetada para facilitar a simulação de modelos DSSAT-CSM sobre HTTP. Ele* *fornece ferramentas de simulação e visualização e integração com software de terceiros altamente otimizado para o processamento de arquivos DSSAT. As camadas de abstração permitem uma metodologia padronizada para ler vários arquivos DSSAT em uma estrutura de dados muito bem formatada. Estas características permitiram ao DSSAT-Lite executar simulações remotas, especificamente usando dispositivos móveis e navegadores web.*

1. Introduction

With each technological advance and with the spread of the use of computer programs, new computational challenges arise. The replacement of the command line interface with the graphical interface facilitated the increasing use of programs by users. However, this transformation caused another problem for software developers: code reuse across different platforms and integration with different programming languages.

In recent years, several innovations in web technology have allowed designers and software engineers to develop responsive web applications [Shahzad 2017]. The web-based approach also reduces the complexity of software distribution, which is a challenge for any development team. However, many tools created before the 2000s use technologies that today are hard to maintain and integrate with modern computational tools.

The tool that drives this work is the Decision Support System for Agrotechnology Transfer (DSSAT), a software package that integrates soil effects, genetic coefficients, climate data, and management options. DSSAT is a collection of databases, crop models, and different applications that are operated by software to help the user select and compare different options and predict the final results [Hoogenboom et al. 2019a].

The DSSAT development was done using different programming languages and with little attention to the structure of the software. It has been redesigned to facilitate the incorporation of new scientific advances, applications, documentation, and maintenance. The principle of this remodeling is a modular structure, called **CSM** (cropping system model), in which the components are separated along scientific discipline lines and structured to allow easy replacement or addition of new modules in the DSSAT-CSM [Jones et al. 2003]. Figure 1, shows that DSSAT-CSM is made up of four different components: a) a database collection, used to organize and store a minimum set of input data required to run crop models; (b) a series of crop models; c) applications to analyze and display the simulated experiments; d) a user interface [Verhagen et al. 2001].

Each DSSAT module has six operational steps, as shown in Figure 1, which are responsible for structuring and organizing a simulation, such as: initialization, crop installation, rate calculations, integration, daily outputs and summary of outputs. The *Main Program* controls when each of these steps is triggered and when each module should perform a task.

The DSSAT functions have been developed primarily to support the use of crop simulation models in decision making [Jones et al. 1998]. The following crop models are currently accessible in the DSSAT environment:

- CERES for cereal models, which include corn, wheat, rice, barley and sorghum..
- CROPGRO de models for legumes such as soybeans and peanuts.
- CROPSIM for root crops such as cassava and potato.

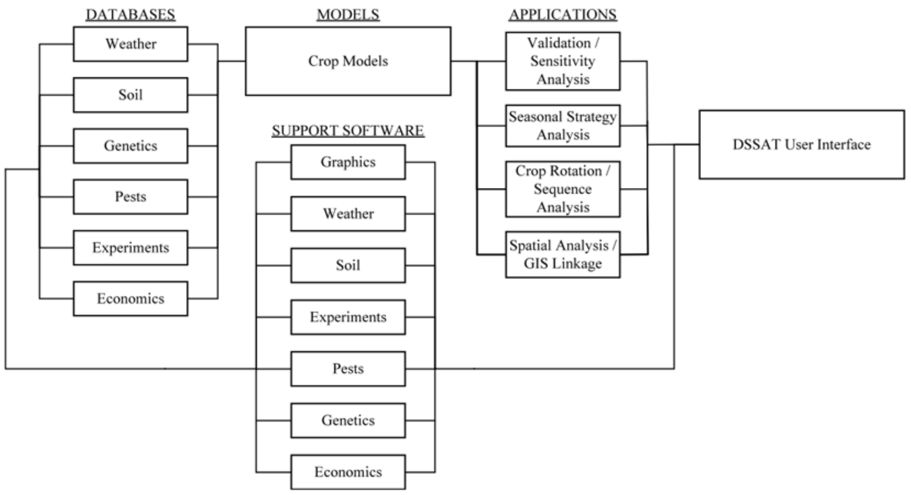


Figure 1. Diagram of database, application, and support software components and their use with crop models for applications in DSSAT [Jones et al. 2003].

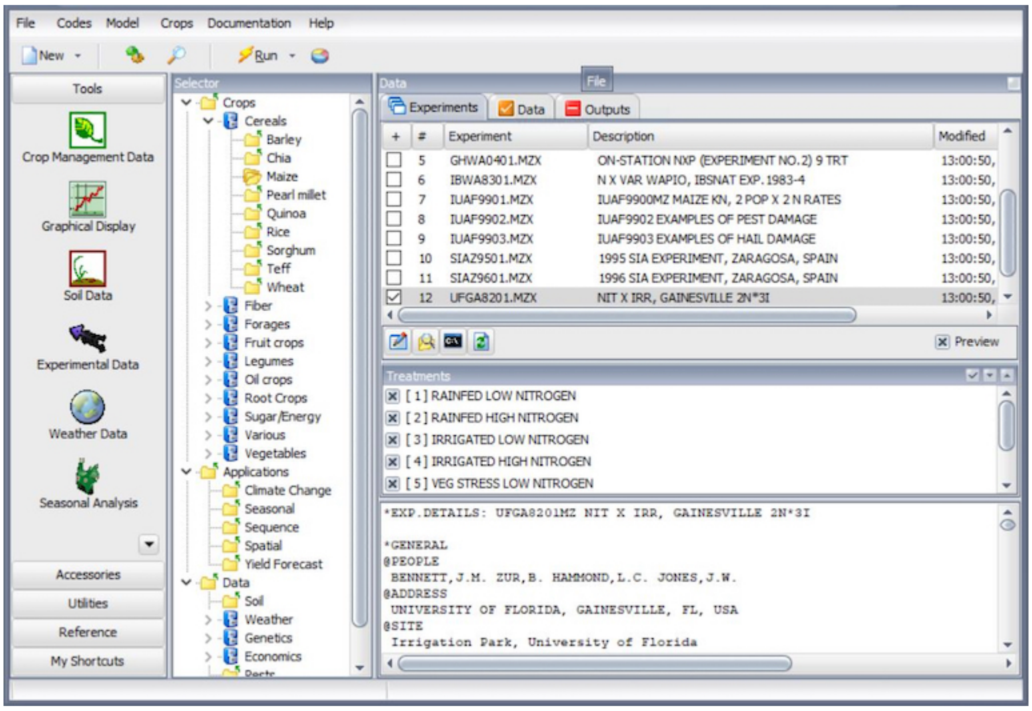
- CROPGRO allows users to modify values in a model culture file without changing any code.

One of DSSAT's strengths is the ability to analyze many management strategies. Several years of simulation are run using climate data generated to simulate the effects of the climate in the future [Jones et al. 2003]. In addition, DSSAT is suitable for long-term studies, which assists in the evaluation of efficient crop management strategies and optimize production [Faria and Bowen 2003].

Among the limitations of the DSSAT, the restriction to homogeneous field-scale analyzes stands out [Jones et al. 2003]. Also, the fact that crop models are written using the FORTRAN programming language makes it difficult to integrate with new components. The DSSAT user interface, Figure 1, is supported on Windows only and uses technologies in the end of life support.

The DSSAT models are executed through the command-line interpreter that requires a particular experience in computer science. Most of the operations in DSSAT are made over a command-line interpreter, which requires a piece of good computational knowledge. Since the DSSAT community is interdisciplinary, not only computer science or others are related, the use of command-line becomes a challenge for most users. Many times, it is a challenge for researchers in the computer science area as well.

There is one alternative for the command-line, which is the DSSAT Shell program (Figure 2). It provides a working environment where various stand-alone tools and applications are seamlessly integrated with the DSSAT crop models. Within the shell, the user can launch applications for creating and modifying data files, running the crop models, and analyzing the results [Hoogenboom et al. 2019b]. However, the current DSSAT Shell is a desktop application that runs on Windows Operation System only, and it was written using Delphi and many tools in Visual Basic language. Like any other desktop application, it requires installing in a local machine, and simulation and data visualization are accessed offline.



+	#	Experiment	Description	Modified
☐	5	GHWA0401.MZX	ON-STATION NXP (EXPERIMENT NO.2) 9 TRT	13:00:50,
☐	6	IBWA8301.MZX	N X VAR WAPIO, IBSNAT EXP.1983-4	13:00:50,
☐	7	IUAF9901.MZX	IUAF9900MZ MAIZE KN, 2 POP X 2 N RATES	13:00:50,
☐	8	IUAF9902.MZX	IUAF9902 EXAMPLES OF PEST DAMAGE	13:00:50,
☐	9	IUAF9903.MZX	IUAF9903 EXAMPLES OF HAIL DAMAGE	13:00:50,
☐	10	SIAZ9501.MZX	1995 SIA EXPERIMENT, ZARAGOSA, SPAIN	13:00:50,
☐	11	SIAZ9601.MZX	1996 SIA EXPERIMENT, ZARAGOSA, SPAIN	13:00:50,
✓	12	UFGA8201.MZX	NIT X IRR, GAINESVILLE 2N*3I	13:00:50,

[1] RAINFED LOW NITROGEN
[2] RAINFED HIGH NITROGEN
[3] IRRIGATED LOW NITROGEN
[4] IRRIGATED HIGH NITROGEN
[5] VEG STRESS LOW NITROGEN

```
*EXP.DETAILS: UFGA8201MZ NIT X IRR, GAINESVILLE 2N*3I

*GENERAL
@PEOPLE
 BENNETT,J.M. ZUR,B. HAMMOND,L.C. JONES,J.W.
@ADDRESS
 UNIVERSITY OF FLORIDA, GAINESVILLE, FL, USA
@SITE
 Irrigation Park, University of Florida
```

Figure 2. Screenshot from the main DSSAT-Shell window, available under DSSAT version 4.7.5 installation package.

This study intent to help the DSSAT community which contains approximately $14,000$ researchers, educators, consultants, extension agents, growers and decision makers in over 179 countries worldwide. This work aims to build a solution that allows the DSSAT community to run and visualize DSSAT crop models over HTTP, to run simulations and visualize results from anywhere in the globe, through a mobile device or via browser using World Wide Web.

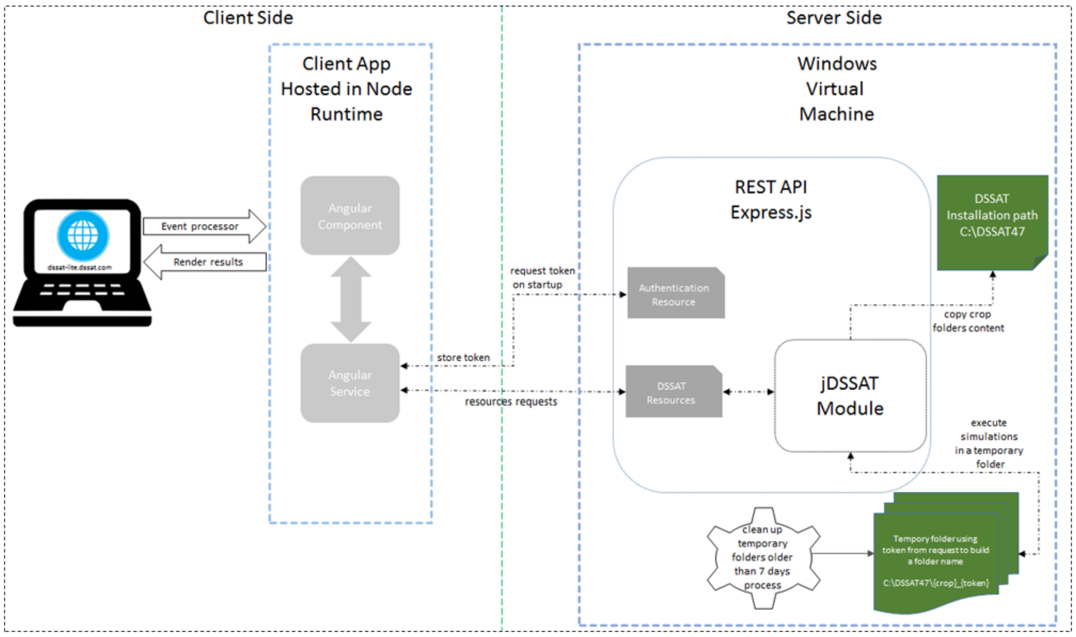
2. Software description

Some functional requirements and functionalities have guided our implementation. The main functionalities from DSSAT Shell that DSSAT-Lite will support are 1) **Responsive**: DSSAT-Lite pages should render well on a variety of devices and window or screen sizes. It also must support mobile devices; 2) **Parallelism**: it must support multiple simulations at once. Simulations should not impact each other; 3) **Performance**: slow sites have a negative impact for the users. DSSAT should be able to run simulations and visualize result quickly; and 4) **User-friendly**: DSSAT-Lite must provide an user-friendly interface to the users; even users that are starting with DSSAT should be able to find the path to run a simulation and to visualize results easily.

3. Technical Implementation

Many frameworks and libraries were used during DSSAT-Lite development. These frameworks are **jDSSAT** [Resenes et al. 2019], **Express.js**, **Angular**, **Bootstrap** and **Jasmine**.

The DSSAT-Lite application design is based on an architecture that follows the Client/Server approach via Representational State Transfer (REST) [Fielding and Taylor 2000]. All the components of the architecture are shown in Figure 3. The client, a web application, handles the DSSAT-Lite user interface with responses to user events such as selecting experiments for a given crop and running DSSAT model simulation. The server-side provides the data for the user's events, which runs on a remote computer. It has a DSSAT-CSM version installed that will run the DSSAT models when requested from the client. The server-side is a NodeJs REST API that is accessible through HTTP from the client. Since the DSSAT folder structure is shared in the remote computer, every user has a temporary folder for experiments and simulation outputs. There is a service installed on the remote computer to clean up the temporary folder every seven days.



Client Side
Server Side
Client App Hosted in Node Runtime
Windows Virtual Machine
dssat-lite.dssat.com
Event processor
Render results
Angular Component
Angular Service
request token on startup
store token
resources requests
REST API Express.js
Authentication Resource
DSSAT Resources
jDSSAT Module
DSSAT Installation path C:\DSSAT47
copy crop folders content
execute simulations in a temporary folder
clean up temporary folders older than 7 days process
Tempory folder using token from request to build a folder name
C:\DSSAT47\{crop}_{token}

Figure 3. The DSSAT-Lite follows client-server architecture.

Once the user accesses the DSSAT-Lite Uniform Resource Locator (URL), a request is made to the Middleware REST API, specifically to the authentication service, in order to generate a token. We keep the token in the browser local storage. This token is used later to create a temporary crop folder to avoid concurrence issues. The simulation from user *A* won't impact a simulation from user *B*. Both users will have different temporary folders.

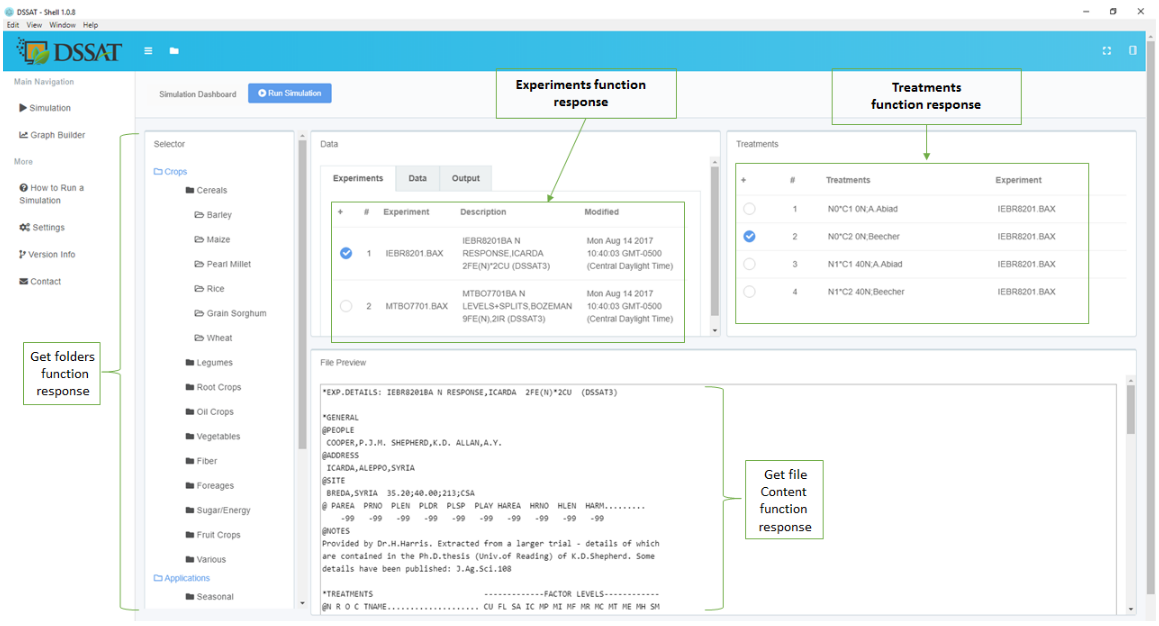
After the token is generated and stored, we can now use the DSSAT resource available in the Middleware API. The DSSAT resource has a different path in the URL. For instance, to retrieve experiments for a given crop, the URL [1] path will contain the crop name. The HTTP method, in this case, will be a GET. The result of this request will be a JSON Array [Crockford 2006], and it will be handle by the client-side before displaying it to the user.

DSSAT-Lite allows any user to run a DSSAT-CSM simulation model remotely. In this case, a temporary folder with the results is created. For example, the user *A* accesses the DSSAT-Lite URL; at this moment, a token is generated by the middleware

[1]https://fqdn/api/dssat/treatments:crop

API (for instance *9f5IEN7BWP*). The user then will select what: the crop, experiments, and treatments to be simulated, in this order, and push *Run Model* button in the DSSAT-Lite page, and a request is made to the Middleware API to run the simulation.

In the DSSAT-Lite interface (Figure 4), there are four menus on the left: Simulation, Graph Builder, Version Info, and Contact. The simulation contains information about the experiments and treatments. Graph Builder is used to plotting graphs and visualize simulation results. The other two menus are version info, including the current release note and contacts used to report issues. At the top of the Simulation page, a dropdown with the available crops to be selected. This means that the users can choose different crops to run a simulation. Next, the button Run Simulation is used to run a simulation in the remote server after selecting Experiments and treatments.



+	#	Experiment	Description	Modified
✓	1	IEBR8201.BAX	IEBR8201BA N RESPONSE,ICARDA 2FE(N)*2CU (DSSAT3)	Mon Aug 14 2017 10:40:03 GMT-0500 (Central Daylight Time)
○	2	MTBO7701.BAX	MTBO7701BA N LEVELS+SPLITS,BOZEMAN 9FE(N),2IR (DSSAT3)	Mon Aug 14 2017 10:40:03 GMT-0500 (Central Daylight Time)

+	#	Treatments	Experiment
○	1	N0*C1 0N;A.Abiad	IEBR8201.BAX
✓	2	N0*C2 0N;Beecher	IEBR8201.BAX
○	3	N1*C1 40N;A.Abiad	IEBR8201.BAX
○	4	N1*C2 40N;Beecher	IEBR8201.BAX

```
*EXP.DETAILS: IEBR8201BA N RESPONSE,ICARDA  2FE(N)*2CU  (DSSAT3)

*GENERAL
@PEOPLE
 COOPER,P.J.M. SHEPHERD,K.D. ALLAN,A.Y.
@ADDRESS
 ICARDA,ALEPPO,SYRIA
@SITE
 BREDA,SYRIA  35.20;40.00;213;CSA
@ PAREA  PRNO  PLEN  PLDR  PLSP  PLAY HAREA  HRNO  HLEN  HARM.........
    -99   -99   -99   -99   -99   -99   -99   -99   -99   -99
@NOTES
Provided by Dr.H.Harris. Extracted from a larger trial - details of which
are contained in the Ph.D.thesis (Univ.of Reading) of K.D.Shepherd. Some
details have been published: J.Ag.Sci.108

*TREATMENTS                        ------------FACTOR LEVELS-----------
@N R O C TNAME.................... CU FL SA IC MP MI MF MR MC MT ME MH SM
```

Figure 4. DSSAT-Lite simulation page. The data card displays all experiments based on the crop selected, Maize, in this case. The treatments displayed on the right card are loaded based on the experiment file the user selected. There is also a File Preview card of the experiment file

4. Illustrative example

DSSAT-Lite has been implemented over HTTP with the integration of **ploty.js** library for graphical results. Furthermore, this solution enables multiple simulations simultaneously using APIs.

Our example, depicted in Figure 4, represents the Maize crop selection by the DSSAT-Lite user. Whenever an event occurs in the Web interface, a request is made to the server where the DSSAT-Lite backend is hosted. Once the server receives the request, it looks at experiments files available from the Maize crop folder. The treatments available for selection are based on the experiment selected. After treatments and experiments are selected, the user can now run a simulation. A graphical representation of running simulation steps is shown in Figure 5.

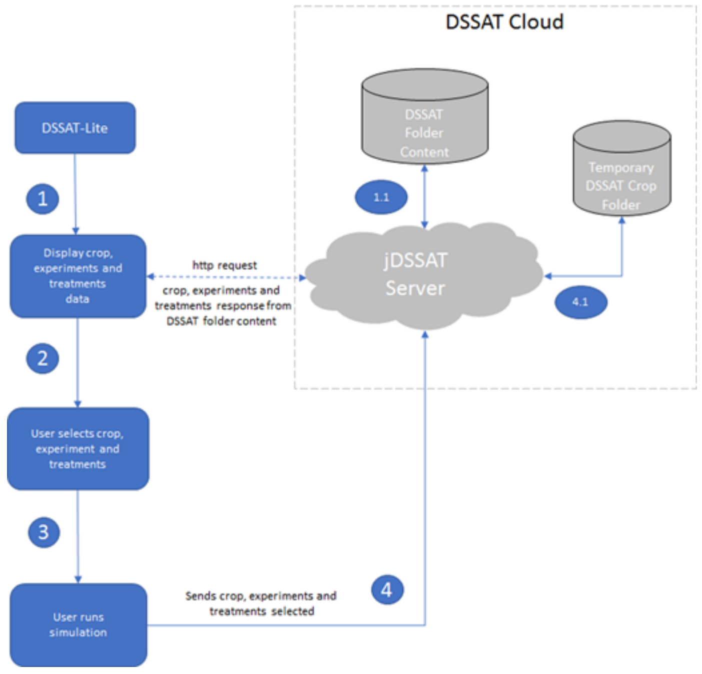

DSSAT Cloud
DSSAT-Lite
DSSAT Folder Content
Temporary DSSAT Crop Folder
1
1.1
Display crop, experiments and treatments data
http request
crop, experiments and treatments response from DSSAT folder content
jDSSAT Server
4.1
2
User selects crop, experiment and treatments
3
User runs simulation
Sends crop, experiments and treatments selected
4

Figure 5. Flow diagram showing the steps to run a simulation. Step 1 is for the DSSAT-Lite to load the crops, experiments ad treatments using jDSSAT server API (step 1.1). Step 2 represents DSSAT-Lite displaying the data. Step 3 represents experiment and treatments selection by the user. Finally, the user runs a simulation. The jDSSAT server processes the simulation request and creates a temporary folder to save the simulation results and retrieve them later (steps 4 and 4.1).

After a simulation is completed, the user can visualize the results. Crop simulation models are process-based and provide very detailed outputs of the simulated crop and soil processes [Yang et al. 2014]. However, analyzing the outputs is a challenge, and the existing tools such Easy Grapher [Yang et al. 2014] do not run through the web. The DSSAT-Lite is responsive, and it automatically resizes, hides, shrinks, or enlarges its content to make its user interface fit in all devices (desktops, tablets, and phones). Figure 7, shows how the DSSAT-Lite layout fits in a mobile device. The DSSAT-Lite provides a page to visualize simulation results, shown in Figure 6.

5. Final Considerations

The philosophy of abstracting complexity from software execution should enable the evolution of DSSAT as software. DSSAT-Lite combines APIs and third-party tools to allow the DSSAT community to run experiments remotely. In the current DSSAT-Lite version, the users meet limited usage features compared with the current DSSAT-Shell. However, running applications through the web opens up the possibility to run multiple simulations at once. The APIs used by DSSAT-Lite is versatile and extensible. These APIs can be used to build other solutions, such as the R package, to run simulations using R language, leveraging its power of data visualization.

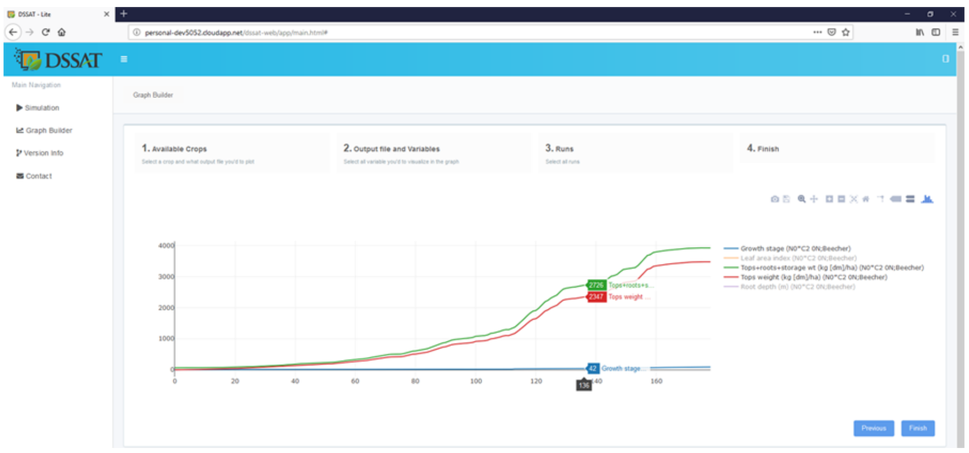
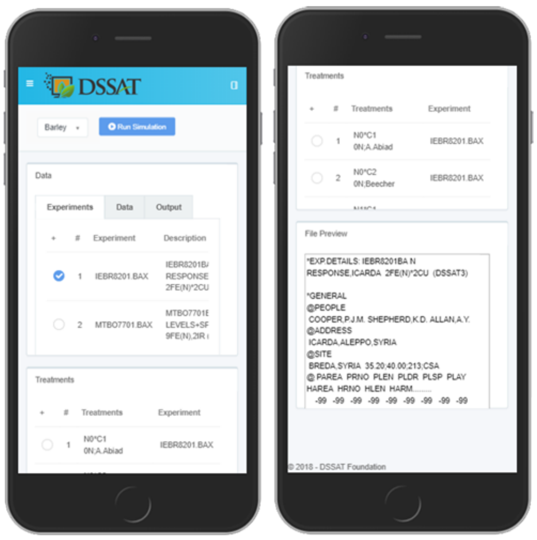

DSSAT
Main Navigation
Simulation
Graph Builder
Version info
Contact
Graph Builder
1. Available Crops
2. Output file and Variables
3. Runs
4. Finish
Growth stage (N0*C2 0N;Beecher)
Leaf area index (N0*C2 0N;Beecher)
Tops+roots+storage wt (kg [dm]/ha) (N0*C2 0N;Beecher)
Tops weight (kg [dm]/ha) (N0*C2 0N;Beecher)
Root depth (m) (N0*C2 0N;Beecher)
Previous
Finish

Figure 6. Graph builder page. It displays the values of the variables from a simulation. It has options to download the graph, zoom in and out and show/hide variables.



+	#	Experiment	Description
✓	1	IEBR8201.BAX	IEBR8201BA RESPONSE 2FE(N)*2CU
	2	MTBO7701.BAX	MTBO7701E LEVELS+SF 9FE(N),2IR

Treatments

+	#	Treatments	Experiment
	1	N0*C1 0N;A.Abiad	IEBR8201.BAX
	2	N0*C2 0N;Beecher	IEBR8201.BAX

File Preview

```
*EXP.DETAILS: IEBR8201BA N
RESPONSE,ICARDA 2FE(N)*2CU (DSSAT3)

*GENERAL
@PEOPLE
 COOPER,P.J.M. SHEPHERD,K.D. ALLAN,A.Y.
@ADDRESS
 ICARDA,ALEPPO,SYRIA
@SITE
 BREDA,SYRIA 35.20;40.00;213;CSA
@ PAREA PRNO PLEN PLDR PLSP PLAY
HAREA HRNO HLEN HARM.........
   -99 -99 -99 -99 -99 -99 -99 -99 -99
```

© 2018 - DSSAT Foundation

Figure 7. DSSAT-Lite running in a mobile device.

References

Crockford, D. (2006). The application/json media type for javascript object notation (json). *RFC*, 4627:1–10. `https://doi.org/10.17487/RFC4627`.

Faria, R. T. d. and Bowen, W. T. (2003). Evaluation of dssat soil-water balance module under cropped and bare soil conditions. *Brazilian Archives of Biology and Technology*, 46(4):489–498.

Fielding, R. T. and Taylor, R. N. (2000). *Architectural styles and the design of network-based software architectures*, volume 7. University of California, Irvine Doctoral dissertation.

Hoogenboom, G., Porter, C., Boote, K., Shelia, V., Wilkens, P., Singh, U., White, J., Asseng, S., Lizaso, J., Moreno, L., Pavan, W., Ogoshi, R., Hunt, L., Tsuji, G., , and Jones, J. (2019a). The dssat crop modeling ecosystem. In Boote, K., editor, *Advances in Crop*

Modeling for a Sustainable Agriculture, pages 173–216. Burleigh Dodds Science Publishing. http://dx.doi.org/10.19103/AS.2019.0061.10.

Hoogenboom, G., Porter, C., Boote, K., Shelia, V., Wilkens, P., Singh, U., White, J., Asseng, S., Lizaso, J., Moreno, L., Pavan, W., Ogoshi, R., Hunt, L., Tsuji, G., and Jones, J. (2019b). The dssat crop modeling ecosystem. In Boote, K., editor, *Advances in Crop Modeling for a Sustainable Agriculture*, pages 173–216. Burleigh Dodds Science Publishing. http://dx.doi.org/10.19103/AS.2019.0061.10.

Jones, J., Hoogenboom, G., Porter, C., Boote, K., Batchelor, W., Hunt, L., Wilkens, P., Singh, U., Gijsman, A., and Ritchie, J. (2003). The DSSAT cropping system model. *European Journal of Agronomy*, 18(3):235–265. https://doi.org/10.1016/S1161-0301(02)00107-7.

Jones, J. W., Tsuji, G. Y., Hoogenboom, G., Hunt, L. A., Thornton, P. K., Wilkens, P. W., Imamura, D. T., Bowen, W. T., and Singh, U. (1998). Decision support system for agrotechnology transfer: DSSAT v3. In Tsuji, G. Y., Hoogenboom, G., and Thornton, P. K., editors, *Understanding Options for Agricultural Production*, pages 157–177. Springer Netherlands, Dordrecht. https://doi.org/10.1007/978-94-017-3624-4_8.

Resenes, J. A., Pavan, W., Hölbig, C. A., Fernandes, J. M. C., Shelia, V., Porter, C., and Hoogenboom, G. (2019). jDSSAT: A javascript module for dssat-csm integration. *SoftwareX*, 10:100271. https://doi.org/10.1016/j.softx.2019.100271.

Shahzad, F. (2017). Modern and responsive mobile-enabled web applications. *Procedia Computer Science*, 110:410 – 415. 14th International Conference on Mobile Systems and Pervasive Computing (MobiSPC 2017) / 12th International Conference on Future Networks and Communications (FNC 2017) / Affiliated Workshops. https://doi.org/10.1016/j.procs.2017.06.105.

Verhagen, A., Conijn, J., and Schapendonk, A. (2001). Quickscan of simulation models. Technical report, Plant Research International BV. Scientific report, Nota 130. Available at: https://edepot.wur.nl/26948.

Yang, J., Drury, C., Yang, J., Li, Z., and Hoogenboom, G. (2014). Easygrapher: software for data visualization and statistical evaluation of dssat cropping system model and the canb model. *International Journal of Computer Theory and Engineering*, 6(3):210. https://doi.org/10.7763/IJCTE.2014.V6.864.