Short Paper: Integrating Web Applications into the HUBzero Gateway Platform

Martin Hunt
HUBzero
Purdue University
West Lafayette, IN 47907
Email: mmh@purdue.edu

Derrick Kearney
HUBzero
Purdue University
West Lafayette, IN 47907
Email: dsk@purdue.edu

Abstract—**The HUBzero® Platform has a long history of supporting interactive simulation tools with X11-based graphical user interfaces through the web browser. In recent years, new HTML and JavaScript based development environments have matured and are being utilized by researchers to help them perform research, develop simulations, clean and explore data, and present results. In this paper, we look at how the HUBzero Platform supports the use and publication of web applications built with Jupyter Notebooks and RStudio's Shiny® and RMarkdown® libraries, and how this infrastructure is generic enough to support other web application platforms.**

I. Introduction

The HUBzero Platform was designed to allow simulations to run in containers with X11 graphics forwarded to the users' browser via the virtual network computing (VNC) protocol. This allowed existing applications with X11-based, desktop style graphical user interfaces to be easily deployed on the servers and run by users around the world.

Over the last four years, advances in web technologies have helped usher in a new wave of interactivity for user interfaces built using HTML, CSS, JavaScript and WebGL. Where previously, users had to push the refresh button in their web browser to check for updates to a web page or database, Asynchronous JavaScript and XML/JSON (AJAX) and HTML5 now allow those updates to be streamed back to the user's browser, improving the interactivity and user experience of applications. Similarly, the HTML5 Canvas and WebGL have brought new life to interactive graphics and data visualization in the browser.

These advances are being wired into products like Jupyter Notebooks [1] and RStudio's Shiny and RMarkdown libraries [2] with the goal of reducing the barriers scientists and researchers face when trying to perform and disseminate their research.

In order for the HUBzero Platform to take advantage of these advances, we first needed to make changes to our architecture that previously only allowed for running X11-based, desktop style applications in a tool container and projecting the screen back to the user's web browser through VNC. In the following sections, we outline how we have supported hundreds of applications through our previous architecture, what challenges were faced adding support for web applications while preserving the strengths of the existing platform, and show examples of published web applications now available on the HUB.

II. Tool containers and X11-based applications on the HUB

When users launch an application on the HUBzero Platform, the application runs in a tool container, a lightweight virtual environment implemented using OpenVZ [3]. On the HUB, tool containers run the Debian GNU/Linux operating system. These OpenVZ based containers help provide isolation between the running instances of applications being served by the HUB, and provide control over network and filesystem access inside of the container. Software can be installed in the container using the operating system's package manager or it can be installed to a shared disk that is mounted within the container.

Each user has a home directory with conventional ownership, access controls, and quota limitations. Applications executed from within the tool container are run with the rights and privileges of the particular user rather than a shared execution account.

Upon initialization, each tool container starts an X Window (X11) Server and the display is project back into the user's web browser through Virtual Network Computing (VNC). Running an X Server inside of the tool container allows the developer to choose from a variety of graphical user interface toolkits with which to build an application. The containers support any GUI toolkit that runs under the X Window System, including popular options like Tcl/Tk, GTK+, Qt, and WxWidgets. To aid scientists and researchers who may not have expertise in user interface development, our team built Rappture, the Rappid APPlication infrastrucTURE [4]. Rappture is a Tcl/Tk and C++ based toolkit that allows developers to quickly assemble and attach a graphical user interface to their science application. On HUBs like nanohub.org and pharmahub.org and mygeohub.org, over 400 applications have been deployed using an X11-based user interface.

III. Adding Support for Web Applications

Rich sets of widgets and design patterns for desktop style graphical user interfaces have dominated our screens since the mid-1970s. Even through the dot-com bubble of the mid to

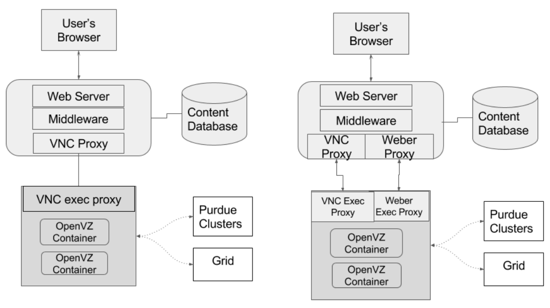
Presented at Gateways 2017, University of Michigan, Ann Arbor, MI, October 23-25, 2017.
https://gateways2017.figshare.com/

late 1990s, websites pushed a paradigm where users submitted forms to interact with the site, and server responses resulted in full page refreshes. It wasn't until the mid 2000s when JavaScript was used to do partial refreshes of webpages, allowing the user to stay at a single web url where the client-server model "application" started to feel and act like a desktop style application.

Today, web applications can take advantage of the HTML5 standard which, among other things, provides support for locally stored data, rich media elements, and official support for the ⟨canvas⟩ element which allows developers to draw graphics with JavaScript. These are all pieces that help to provide web applications with the look, feel, and functionality of desktop style applications.

These advances in web application technology, along with the ubiquity of the web browser as the common user interface platform across devices, have made creating interactive graphical user interfaces with HTML, CSS, and JavaScript a reality and a necessity. With the rising interest from developers to deploy web applications and interest from users to build interactive workflows, our team made adjustments to our software stack to support running web applications on our infrastructure in a way that kept the strengths of the platform, like security and application isolation, in the forefront.

To adapt to the changing face of the graphical user interface, the HUBzero Platform had to first add support in the middleware for HTTPS connections with websockets and then needed to ensure that requests to access tool containers could be properly authenticated.



(a) Supports VNC only (b) Supports VNC and web applications

Fig. 1: To support web applications, two new proxies needed to be added. The Weber Front proxy, runs on the web server and terminates an SSL connection between the user's web browser and the HUB. The Weber Exec proxy directs connections to the tool container where the server side of web applications run.

These challenges were tackled first while developing support for Jupyter Notebooks. The implementation plan involved running the notebook server within the tool container and streaming the HTML, CSS, and JavaScript from the tool container, through the Weber Front and Exec proxies, and out to the user's web browser. By hosting the notebook server inside of a tool container, users were enjoy many of the same benefits they had become accustomed to while running the HUB's desktop style application offerings. Namely, users could access their own HUB disk space, run other applications that had been published on the HUB, and access powerful compute clusters through Submit [5].

Not long after building the Jupyter Notebook proof of concept, the team began trying similar setups with the RStudio's IDE and Shiny, several Node.js® [6] applications, Python's Flask [7] microframework, and Wt [8], a lesser known C++ based toolkit for building web applications. In each case, the robustness of the new HUBzero architecture allowed the application to be deployed within the infrastructure. Let's take a look at a few examples of web applications that have been deployed on the HUB, using this new infrastructure.

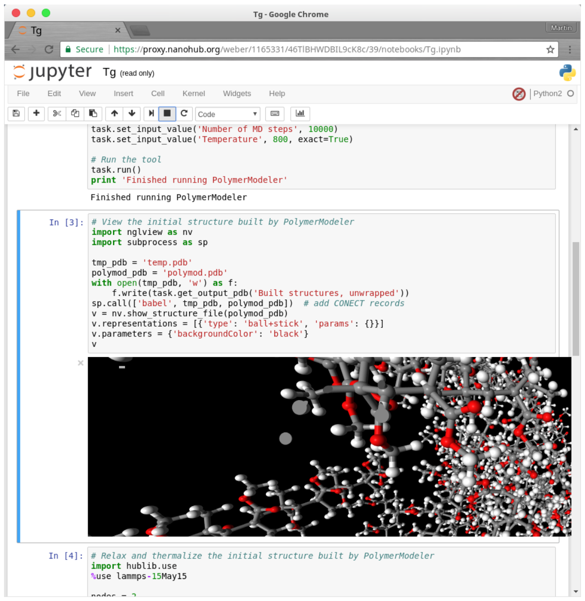
IV. Example web applications running on the HUB

A. *Jupyter*

Over the years, there have been a number of players in the literate computing space. It's a space that blurs the lines between documentation, programming, and the application, by combining narrative, live code, and visualization into one document to tell a story. One of the more recent additions to the space is Project Jupyter with their offering of the Jupyter Notebook, an open-source web application that supports multiple languages and provides an intuitive integrated development environment (IDE) that complements the way many people go about working with data, building models, and running simulations. Jupyter Notebooks were the first web applications to be supported on the HUBzero Platform, and the implementation approach gives it rare capabilities.

To build a Notebook outside of the HUB ecosystem, people commonly download and install the Jupyter Notebook software on their own computer. To share their Notebook, they can post it to GitHub, or one of the other many websites that support the storage of Notebooks, but this means other people need to have the Jupyter Notebook software on their computer in order to view it. Furthermore, if other people wanted to interact with the Notebook and execute the cells with their own data sets, they would need to find and install all of the packages and libraries necessary to run the notebook.

With Jupyter running on the HUB, users can instantly start working in a Notebook, in an environment preloaded with over 450 commonly used packages, spanning the Python, R, and Octave programming languages. In addition to being able work interactively with notebooks, either new ones the user just started or mature ones they uploaded into their HUB drive, users can also publish Notebooks and provide a one click solution for others who want to run their code. Jupyter content can also be published as a tool on the HUB, where the notebook's code cells are hidden and users can only interact with the exposed graphical user interface elements. HUBzero extensions, like HUBLIB, help lower the barriers to using the Jupyter Notebook environment as a tool development framework.


Tg - Google Chrome
Tg
Martin
Secure | https://proxy.nanohub.org/weber/1165331/46TlBHWDBIL9cK8c/39/notebooks/Tg.ipynb
jupyter Tg (read only)
File Edit View Insert Cell Kernel Widgets Help
Python2
Code

```
task.set_input_value('Number of MD steps', 10000)
task.set_input_value('Temperature', 800, exact=True)

# Run the tool
task.run()
print 'Finished running PolymerModeler'
```

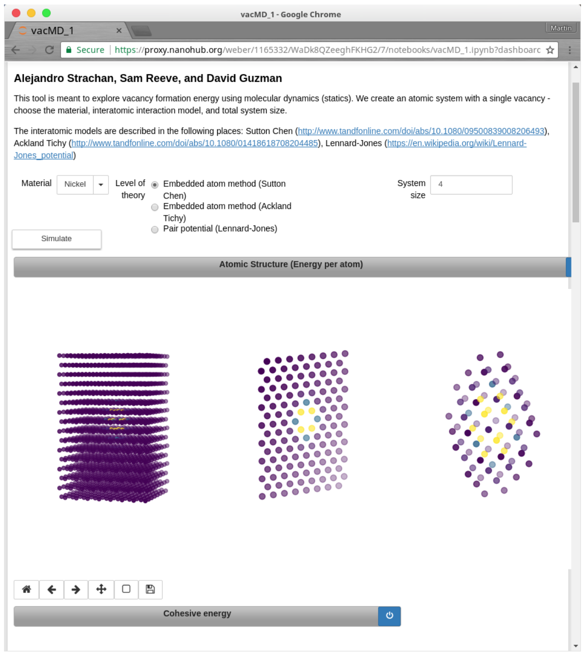
```
Finished running PolymerModeler
```

```
In [3]: # View the initial structure built by PolymerModeler
        import nglview as nv
        import subprocess as sp

        tmp_pdb = 'temp.pdb'
        polymod_pdb = 'polymod.pdb'
        with open(tmp_pdb, 'w') as f:
            f.write(task.get_output_pdb('Built structures, unwrapped'))
        sp.call(['babel', tmp_pdb, polymod_pdb])  # add CONECT records
        v = nv.show_structure_file(polymod_pdb)
        v.representations = [{'type': 'ball+stick', 'params': {}}]
        v.parameters = {'backgroundColor': 'black'}
        v
```

```
In [4]: # Relax and thermalize the initial structure built by PolymerModeler
        import hublib.use
        %use lammps-15May15
```

Fig. 2: Glass Transition Notebook [9]


vacMD_1 - Google Chrome
vacMD_1
Martin
Secure | https://proxy.nanohub.org/weber/1165332/WaDk8QZeeghFKHG2/7/notebooks/vacMD_1.ipynb?dashboarc
Simulate
Atomic Structure (Energy per atom)
Cohesive energy

Alejandro Strachan, Sam Reeve, and David Guzman

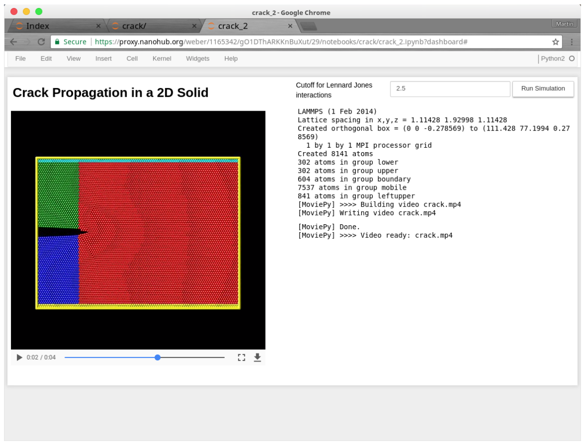
This tool is meant to explore vacancy formation energy using molecular dynamics (statics). We create an atomic system with a single vacancy - choose the material, interatomic interaction model, and total system size.

The interatomic models are described in the following places: Sutton Chen (http://www.tandfonline.com/doi/abs/10.1080/09500839008206493), Ackland Tichy (http://www.tandfonline.com/doi/abs/10.1080/01418618708204485), Lennard-Jones (https://en.wikipedia.org/wiki/Lennard-Jones_potential)

Material: Nickel
Level of theory: [✓ Embedded atom method (Sutton Chen)] Embedded atom method (Ackland Tichy) Pair potential (Lennard-Jones)
System size: 4

Fig. 3: Vacancy Formation Energy with MD [10]

Tg, the application shown in Figure 2, is a Jupyter Notebook published on nanoHUB.org that calculates glass transition temperatures through molecular dynamics calculations. Under the hood, it runs a workflow that incorporates previously published Rappture applications, jobs sent to remote compute clusters, and local calculations. Other types of Jupyter content that can be published on the HUBzero Platform can be seen in Figure 3 and Figure 4. Note the lack of source code and editing tools shown in these applications. With Jupyter based applications, developers have the choice of showing or hiding their source code when the application is published.


crack_2 - Google Chrome
Index
crack/
crack_2
Martin
Secure | https://proxy.nanohub.org/weber/1165342/gO1DThARKKnBuXut/29/notebooks/crack/crack_2.ipynb?dashboard#
File Edit View Insert Cell Kernel Widgets Help
Python2
Run Simulation
0:02 / 0:04

Crack Propagation in a 2D Solid

Cutoff for Lennard Jones interactions: 2.5

```
LAMMPS (1 Feb 2014)
Lattice spacing in x,y,z = 1.11428 1.92998 1.11428
Created orthogonal box = (0 0 -0.278569) to (111.428 77.1994 0.27
8569)
  1 by 1 by 1 MPI processor grid
Created 8141 atoms
302 atoms in group lower
302 atoms in group upper
604 atoms in group boundary
7537 atoms in group mobile
841 atoms in group leftupper
[MoviePy] >>>> Building video crack.mp4
[MoviePy] Writing video crack.mp4

[MoviePy] Done.
[MoviePy] >>>> Video ready: crack.mp4
```

Fig. 4: Creating a New Jupyter Tool

B. *RStudio IDE, Shiny Applications, and RMarkdown Documents*

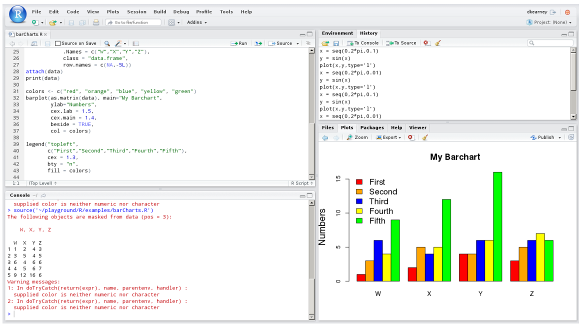
For many R users, the RStudio IDE is the graphical user interface of choice when developing analysis and reports. The desktop version of the application provides access to an R based integrated development environment (IDE) and is available on all major platforms. RStudio Inc., the company behind the RStudio IDE also makes available an Open Source web version of the application that can be deployed on a server to provide access for multiple people to use at a time. While the Open Source version of their IDE lacks the administrative, security, and monitoring tools tools of their commercial version, it does provide many of the features expected by users of the desktop version and can be a good replacement for the desktop version if configured correctly.

Using the web version of the RStudio IDE on the HUB is a particularly good match because the HUB's tool containers already provide the user space and application space isolations that would be required for running an application like this on a web server. Tool containers were developed with isolation and resource management in mind.

On the HUB, when a user starts the RStudio IDE, the request is routed from the user's web browser, through the HUB's webserver, Weber Front and Exec proxies, and into a tool container where an instance of the RStudio Server waits to respond. This same mechanism is used to support applications built using Shiny, RStudio's library for building web applications from within R, and RMarkdown documents. Like the RStudio IDE, Shiny applications and RMarkdown documents require a web server to be running in the tool container to field requests from the user's web browser. By themselves, static HTML documents with CSS and JavaScript can also be served from within a tool container, but there is less of a need for the isolation and security features provided by the tool container.

C. *Node.js, Flask, and Other toolkits*

The changes made to the HUBzero Platform to support web applications are not specific to supporting Jupyter Notebooks



```
            .Names = c("W","X","Y","Z"),
            class = "data.frame",
            row.names = c(NA,-5L))
attach(data)
print(data)

colors <- c("red", "orange", "blue", "yellow", "green")
barplot(as.matrix(data), main="My Barchart",
        ylab="Numbers",
        cex.lab = 1.5,
        cex.main = 1.4,
        beside = TRUE,
        col = colors)

legend("topleft",
       c("First","Second","Third","Fourth","Fifth"),
       cex = 1.3,
       bty = "n",
       fill = colors)
```

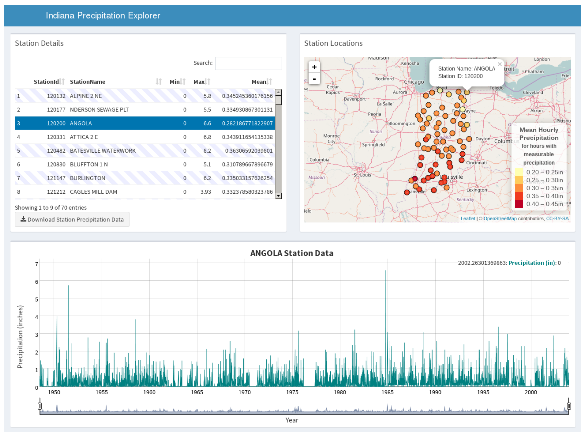
```
  supplied color is neither numeric nor character
> source('~/playground/R/examples/barCharts.R')
The following objects are masked from data (pos = 3):

    W, X, Y, Z

  W  X  Y Z
1 1  2  4 3
2 3  5  4 5
3 6  4  6 6
4 4  5  6 7
5 9 12 16 6
Warning messages:
1: In doTryCatch(return(expr), name, parentenv, handler) :
  supplied color is neither numeric nor character
2: In doTryCatch(return(expr), name, parentenv, handler) :
  supplied color is neither numeric nor character
>
```

```
x = seq(0,2*pi,0.1)
y = sin(x)
plot(x,y,type='l')
x = seq(0,2*pi,0.01)
y = sin(x)
plot(x,y,type='l')
x = seq(0,2*pi,0.1)
y = sin(x)
plot(x,y,type='l')
x = seq(0,2*pi,0.01)
```

Fig. 5: RStudio IDE being served from a HUB tool container. [13]



Indiana Precipitation Explorer

Station Details

Search:

	StationId	StationName	Min	Max	Mean
1	120132	ALPINE 2 NE	0	5.8	0.345245360176156
2	120177	NDERSON SEWAGE PLT	0	5.5	0.334930867301131
3	120200	ANGOLA	0	6.6	0.282186771822907
4	120331	ATTICA 2 E	0	6.8	0.343911654135338
5	120482	BATESVILLE WATERWORK	0	8.2	0.36306592039801
6	120830	BLUFFTON 1 N	0	5.1	0.310789667896679
7	121147	BURLINGTON	0	6.2	0.335033157626254
8	121212	CAGLES MILL DAM	0	3.93	0.332378580323786

Showing 1 to 9 of 70 entries

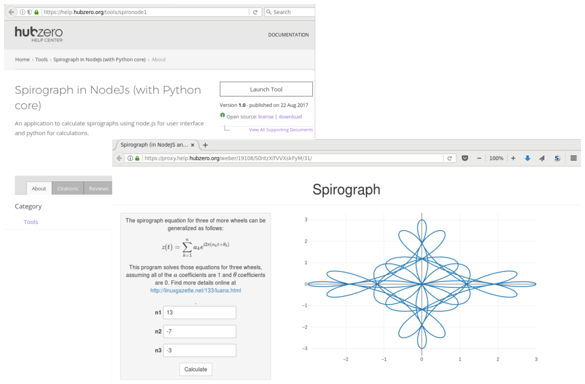
Fig. 6: Indiana Precipitation Explorer is a Shiny web application hosted on myGeoHUB.org. [12]

and RStudio's Shiny and RMarkdown based applications and documents. The pattern of running a web server in the tool container and connecting it to the user's web browser can be extended to many of the web application frameworks available today.

The HUBzero team has built and deployed Node.js and Python Flask examples that show how web applications built using these frameworks can be supported under the same architecture.

Spironode [11] is a Node.js and Python implementation of the Spirograph program and has been deployed as a web application running on the HUBzero Platform. It uses the Express web application framework and EJS templating system to generate the HTML based graphical user interface and handle routing of requests from the user's web browser. The *science* code of calculating the coordinates for the spirograph is performed in Python, The Node.js based user interface calls the separate python program and feeds it inputs from the user. The calculated spirograph coordinates are then sent back to the user where they are rendered by JavaScript running in the browser.

Spiroflask is a similar implementation of the Spirograph program, but instead of being written in Node.js, it uses Python's Flask microframework to generate the web interface,



https://help.hubzero.org/tools/spironode1

Home › Tools › Spirograph in NodeJs (with Python core) › About

Spirograph in NodeJs (with Python core)

An application to calculate spirographs using node.js for user interface and python for calculations.

Version **1.0** - published on 22 Aug 2017

Open source: license | download

https://proxy.help.hubzero.org/weber/19108/S0ntzXlfVVXskFyM/31/

The spirograph equation for three of more wheels can be generalized as follows:

$$z(t) = \sum_{k=1}^{n} a_k e^{i2\pi(n_k t+\theta_k)}$$

This program solves those equations for three wheels, assuming all of the a coefficients are 1 and θ coefficients are 0. Find more details online at http://linuxgazette.net/133/luana.html

n1 13

n2 -7

n3 -3

Fig. 7: Spironode is a web application that uses Node.js to generate the graphical user interface and Python for scientific calculations.

and the core science code is also written in Python.

Another, lesser known library that has been used to build and deploy web applications on the HUB is Wt. Wt serves the C++ community, providing a web application framework and widgets. Applications can be served using FastCGI in coordination with a web server or the application can act as its own web server. The latter is perfect for running in a tool container.

V. Conclusion

The landscape of web applications has changed dramatically over the last few years. With the release of the HTML5 standard and the capabilities that are now available, many developers are choosing to build interactive user interfaces for their applications using HTML, CSS, and JavaScript. The HUBzero Platform can now support these developers in the same way we have supported developers of desktop applications. Furthermore, users of web development environments like Jupyter Notebooks and RStudio have also grown and they are not only interested in creating simulation tools and content, but also in sharing them. With the changes we have made and outlined above, the HUB can now support these new development environments, allowing users to publish not just static HTML web pages, but fully interactive web application stacks, that can be published and shared with users around the world. As new technologies rise, the HUBzero team continues to leverage its unique and flexible platform to serve as a resource where people can develop, publish, and share their online simulations.

References

[1] "Jupyter", http://jupyter.org/
[2] "RStudio - Open Source and enterprise-ready professional software for R", http://shiny.rstudio.com/
[3] "OpenVZ Virtuozzo Containers", https://openvz.org/
[4] McLennan, M.; Kennell, R., *HUBzero: A Platform for Dissemination and Collaboration in Computational Science and Engineering*, Computing in Science and Engineering, **12**(2):48-52 (2010).

[5] M. McLennan, S. Clark, E. Deelman, M. Rynge, K. Vahi, F. McKenna, D. Kearney, C. Song. *HUBzero and Pegasus: integrating scientific workflows into science gateways.* In Concurrency and Computation: Practice and Experience, 2014; 27, pg 328-343, doi: 10.1002/cpe.3257

[6] ”Node.js: a JavaScript runtime built on Chrome’s V8 JavaScript engine”, https://nodejs.org/en/

[7] ”Flask (A Python Microframework)”, http://flask.pocoo.org/

[8] ”Wt: C++ Web Toolkit”, https://www.webtoolkit.eu/wt

[9] Benjamin P Haley; Lorena Alzate-Vargas (2017), ”Glass transition temperature notebook,” https://nanohub.org/resources/tgnb. (DOI: 10.4231/D3N873142).

[10] Sam Reeve (2017), ”Vacancy Formation Energy with MD,” https://nanohub.org/resources/mdvacancy. (DOI: 10.4231/D39S1KM5R).

[11] Derrick Kearney (2017), ”Spirograph in NodeJs (with Python core),” https://help.hubzero.org/resources/spironode1.

[12] Derrick Kearney (2017), ”Indiana Precipitation Explorer,” https://mygeohub.org/resources/incip.

[13] MyGeoHUB.org (2017), ”RStudio,” https://mygeohub.org/resources/rstudio.