Introduction

GeckoMGV is visual display for pairwise-comparisons capable of working with more than one sequence at the same time. It works through the web and allows the user to navigate dynamically through the comparisons, filtering and consulting information.

In this new versión, due to a backend implementation, it is available the possibility to execute services that process the working data.

We proceed to explain how to register and execute this services.

1.- Backend Administration

To access the administration panel of this application go to the following link: http://pistacho.ac.uma.es/admin
There we can see the administrator panel that contains the models within our database. At first, both users and scripts will be added through this interface.

Administration panel
1.1 Adding users

To add a user we must access the panel ‘Usuarios’ and press on the ‘Añadir usuario’ button.

*Users panel*

*Adding user ‘guest/guest’*
User added
1.2 Adding scripts

A script is composed of:

- An executable (if possible, to be compiled on the server)
- A form defined in Django (Python)
- A database registry

1.2.1 Executables

In this example we will use the program ‘kmersFreq’ that returns a list of kmers given a sequence.

We must add the script to the path ‘/var/www/GeckoMGV/media/scripts/’ of the server.

1.2.2 Form in Django – Python

Our applications requires to have a form for each added script. This form must have a field for each parameter needed by the executable.

To do that we must access the file ‘forms.py’ of our app ‘scripts’ folder located at: ‘GeckoMGV/scripts/forms.py’ and define the fields according to the Django standard that can be reviewed here: https://docs.djangoproject.com/en/1.8/topics/forms/ In our example, the program ‘kmersFreq’ requires of:

- A file field of a sequence.
- An integer field for the ‘K’ size of the kmer.
- A text field ‘Fullout’ that indicates the output format.

In our case we have defined the form ‘kmersForm’ with the previously described fields:

```python
class kmersForm(forms.Form):
    filename = forms.ChoiceField(label="Filename", widget=forms.Select(attrs={"class":"selector"}))
    K = forms.CharField(label="K", max_length=1)
    fullOut = forms.CharField(label="Fullout", max_length=1)
```

An init method will also need to be declared if we want to use file type fields, to exclusively retrieve the logged user’s files.
1.2.3 Database registry

Once we are done with the executable and the form, we can register the service in the database. To achieve that, we must go navigate from the admin panel to ‘Scripts’ > ‘Agregar Nuevo Script’ and fill out the form that contains the following fields:

- **Name**: Service name
- **exeName**: Executables name
- **Path**: Executables path
- **Help**: Help/Information
- **Form**: Name of the defined Django form

For our example, it should look like this:

![Administração de Django](image)

Once this is done, the script can be executed by any registered user.
2. Client execution

To access the canvas display we must go into the following URL: http://pistacho.ac.uma.es
The main page is the canvas display accompanied by its main functionalities.

To make use of the new functions, the user must do the following:

- Login to the system
- Upload the necessary files for the execution of a service.
- Access the service and execute it
2.1 Login to the system

In the navigation bar we can find the ‘Sign in’ button. When clicked a form drops down within the bar that allows us to introduce the login data:

Upon completion, in this case as:

user: guest
pass: guest.

The navigation bar will display that we are logged into system, next to a ‘Logout’ button:

2.2 File uploading

Once logged in the system, we access the ‘File Manager’ tab where we can find our file system. There is also a button with the text ‘Upload file’ that deploys a small menu that allows us to upload files.
We proceed by selecting our desired file to upload, for example ‘sequence.fasta’ and we upload it to our server. After it is done, the file manager tab should look like this:

| Filename     | Size  | Options |
|--------------|-------|---------|
| sequence.fasta.txt | 436 bytes |         |

Once our file is in the server, we can work with it.

### 2.3 Service execution

Now we will executed the previously registered service. To do that we access the ‘Services’ tab to view the list of our services. This list is dynamically constructed every time we access this tab. Therefore, new services will appear right after their registry:

**Services**

- **Calculate Kmers**
  - Calculate Kmers from a sequence given
  - **Start**

After pressing the ‘Start’ button, a new view is generated with a form whose fields have been previously defined when the service was registered.

**Filename:**

```
/var/www/GeckoMGV/media/files/users/guest/sequence.fasta.txt
```

**K:** 2  **fullOut:** 1  **Submit**

After filling out the fields and selecting our file, we can execute the service to obtain the results.
It is necessary to give a new format to the results since these are sent as plain/text even though they might be processed within the program.

Client-side obtained results

Server-side obtained results

Future work:

- Improved point & zoom.
- Allow file saving after executing a script:
  - At the moment it is only possible to copy the results into a local machine.
- Make a list of useful predetermined scripts.
- Work with fragments
  - Add the option to work directly from the contextual menú of the fragments.