Universitat
de les Illes Balears

Title: testIA: Webservice for the scientific comunity to test Image Analysis algorithms

AUTHOR: Milan Sindjelic

Master's Thesis

Master's degree in Computer Engineering
(With a specialty/itinerary in Interactive Technologies)

at the

UNIVERSITAT DE LES ILLES BALEARS

Academic year 2017/2018

Date 28.07.2018

UIB Master's Thesis Supervisor Dr Antoni Jaume-i-Capó

UIB Master's Thesis Co-Supervisor Dr Gabriel Moyà Alcover

testIA: Webservice for the scientific comunity to test Image Analysis algorithms

Milan Sindjelic
Tutors: Antoni Jaume-i-Capó, Gabriel Moyà Alcover
Treball de fi de Màster Universitari Enginyeria Informàtica (MINF)
Universitat de les Illes Balears
07122 Palma de Mallorca
milan.sindjelic1@estudiant.uib.cat

Abstract

In this work we made system which offers execution of any image analysis algorithm to other researchers with their datasets by applying our algorithms on their dataset without sharing the code of experiments. System allows researchers to easily reuse designed experiments and repeat them with the same dataset or with a new dataset. This way we tried to reduce problem of reproducibility in computer science. System is called *testIA* and it consists of two parts: front-end, for researchers and back-end management system for administration. System is developed using Django web framework and RESTful web services.

Keywords: reproducibility, reusability, testing environment, image analysis

1. Introduction

One of the biggest objective in science is critical evaluation of the correctness of scientific results and conclusions of other scientists. This could be carried out if the scientist, who is making critical evaluation, has well defined methodology and if he has documentation of the project which should be evaluated. Documentation needs to cover the description of the process which is executed in the experiment and description of the analysis of gained data. During that process, the scientists try to repeat, replicate and reproduce original results and conclusions. The idea for this work came from my mentors from UGiVIA (Computer Graphics and Vision and Artificial Intelligence Group) Department of Mathematics and Computer Science (DMI) at UIB in order to make a system that allows running and testing different experiments and is available to vast group of researchers. At the end, any obtained result from the experiment is not completely constituted if it is not able to be reproduced independently. There is a lot of disagreement in scientific circles about definitions of these three terms. Here I will use the definitions by the *Association for Computing Machinery* which are proposed based on definitions from the *International Vocabulary of Metrology* [2]:

- **Repeatability** *(Same team, identical experimental setup): The measurement can be obtained with stated precision by the same team using the same measurement procedure, the same measuring system, under the same operating conditions, in the same location on multiple trials. For computational experiments, this means that a researcher can reliably repeat her own computation.* [2]
- **Replicability** *(Dissimilar team, identical experimental setup): We can achieve the measurement with stated precision by a dissimilar team using the same measurement procedure, the same measuring system, under the same operating conditions, in the same or a different location on multiple trials. For computational experiments, this means that an independent group can obtain the same result using the author's own artifacts.* [2]
- **Reproducibility** *(Dissimilar team, Dissimilar experimental setup): We can achieve the measurement with stated precision by a dissimilar team, a dissimilar measuring system, in a dissimilar location on multiple trials. For computational experiments, this means that an independent group can obtain the same result using artifacts which they develop completely independently.* [2]

The biggest problem is how to achieve the reproducibility of algorithms in computer science, since for reproducibility authors need to share an original code of the project. Sharing the code represents the point of dispute among scientists and authors of the works. From one side, code represents the good of its author and if it is shared then that good is gone. On the other hand, if the scientist wants to reprocude some project, he should be able to have an access to the original code. Over the years a new term has emerged among the authors - reusability. This term tries to merge replicability and reproducibility in one. Essentially, it applies the process from the paper to similar question. In case of software engineering that can be applied on the process on a new set of data. We can say that reusability is more important than reproducibility. The main reason for this is that reusability allows getting new results which the author did not expect or even did not think about,

since we apply process to new data. With reproducibility, researchers cannot obtain new results.[3][4] Therefore, the main objective of this project is to offer the possibility to other researchers that they can execute any image analysis algorithm with their dataset without access to the original code. With this, they just need to upload their dataset and after execution, they will get the results. This way we cover one of the main principles of reproducibility in computer science since the researchers will be able to rerun the experiment with new datasets. As we said in the previous paragraph, that possibility, to rerun the experiment with new data, in this case with a new set of images, is very important in today's approaches of computer science. Second objective, which is of great importance, is to collect new images from the researchers. With those images we will be able to get new datasets for future researches and improve already existing experiments. The main objective is mainly intended for the researchers to easily test algorithms with theirs datasets. The second objective is intended for the owner laboratory of the project.
testIA will be used for execution different image analysis algorithms. Image analysis(IA) is extraction of important or meaningful information from the digital image. One example can be detection of different things on the image (text, faces, objects). With *testIA* researchers will have the possibility to run any available IA algorithm with theirs dataset. In IA is used machine learning(ML). Especially, the most used type of ML here is supervised learning. There are some common steps which typically IA does: Image preprocessing, Segmentation, Feature extraction and Classification.
This paper is organized as follows: after the introduction, in next subsection we have view of several similar works. In Section 2 is given a view of the architecture used for developing and why they are chosen. The results and discussion of executing one test on given experiment are presented in Section 3. Finally, the conclusions and the proposals for future studies are given in Section 4.

1.1. Existing work

There is a lot of previous works which are related to image analysis, particularly medical images. Those images are very interesting for researchers since improvements in that field brings a lot of gain for medicine. A specialized case of different types of images are those with human cells. Since I am going to discuss the results by running testIA application on the experiment designed for classification of cells, here I will give some observation of existing systems for analysis images with cells.
Very similar paper is published by group of authors from Carnegie Mellon University and Intel Labs Pittsburgh. During their work they developed a public website which enables to researchers running experiments and checking its status online using their developed user interface. Also, the website enables, for any researcher or researcher group, uploading their own cell images for analysis and comparison. This part represents similarities between my work and this. The differences are reflected in fact of proposed algorithms for image analysis. They presented a few algorithms for cell image analysis where they included image restoration of microscopy's images, cell event detection and cell tracking in a large population[2]. Using developed system, researchers are able to run their experiment on those algorithms.[8]
Another one similar system is *CellProfiler*, developed for analysis of images in order to recognize and quantify different types of cells. CellProfiler is the first free program and it is open source. Primary, it is created for biologist researchers with the aim to obtain important information from microscopic images. For example: numbers of cells or their type, etc. This system enables them to put in process a huge number of images, hundreds and thousands of them. CellProfiller is developed as desktop application and that is the main difference in comparison with this work. Also, on back-end it uses the different algorithms then me here. [9]
There is one similar paper to experiment whose results will be commented in result part. This paper can be watched as one possible experiment which can be run on *testIA*. They tried to find and extract the most relevant features for skin lesion computational diagnosis based on shape properties, color variation and texture analysis using different techniques. The group of features which they used are the same as in our experiment. As we will see the differences are in the classes of extracted objects, they do with skin lesions while our system is more generic, and can do with any classes of objects in IA. One more similarity is reflected through the process which is applied, step by step.[5]

2. Methods

2.1. Django framework

Django is an application web framework for the development of web applications. It is written in Python programming language and is free and open source. The following is an MTV (model-template-view) architectural pattern. One of the great advantages of Django is that it facilitates and accelerates the development of complex web applications that use the database for storing information and retrieving and displaying them on the database-driven website. Those advantages were mainly reason why we decided to use Django to develop web application for research community. Django follows MTV pattern which is based on Model-Viewer-Controller (MVC). MVC is an architectural form that serves to implement user interface. It shares the software in three parts to separate the original representation of information from the way the information is presented to the user. Since MVC is a form, certain architectures that use this form may vary.
First, I will explain how these parts are described in the traditional definition. The central part is a model. It records application behavior in the domains of the same domain as the problem that is independent of the user interface. Also, it access data, logic, and application rules directly. The second part

is a viewer. View is any output result of an application such as a chart, table, or text. It is made possible to create more views of one information. The third part is the controller who receives the input and translates it into an outward-looking model or view. The controller is actually a mediator between the model and the look in both directions. Model-Template-View is a kind of MVC architecture used for web site development. It separates different parts of a web application: display, data access, and website logic. MTV enables independent web application building, enhances system security and simplifies system maintenance. There are three parts: **Model** The model defines data forms and relationships in databases. The Django circle model is a class written in the Python programming language, specifies the variables and methods associated with certain types of data, and has the meaning of the table in the database. Associated variables have the meaning of the column in the table, and the methods define relationships between the variables. The model is closely related to the database and view. From a database, the model retrieves the requested data and props them. The model does not have any knowledge of the existence of the template and functions derived from it. In this way, the database is separated from the remaining two parts of the system; **View** The purpose of the view is to determine which data will be displayed, namely, which data will be retrieved from the database and displayed by the view in the web browser. In the Django environment, a separate view file is created when creating a web page for each application. The view file consists of functions written in the Python programming language. For each page, a special function is written that manages the query execution. In addition to the ability to query the data retrieval model, it has the ability to implement email sending, authentication, input parameter verification, and many more. The view does not know how data is displayed in a web browser. The job of view is to fetch the requested data and forward it to a layer that will show them in the browser. **Template** The template is a MTV architecture layer closely linked to the web browser. It is an HTML page with additional structures that allow display of data passed from the view. The task of the template is the content received from the view to organize and embed into the HTML code that will be displayed in the web browser. The preload file has additional limitations on the impossibility of typing commands in the Python programming language. This prevents mixing of functions of individual layers and provides additional security to the web page. It represents view from MVC pattern.

2.2. RESTful web services

Representational State Transfer (REST) is a model of architecture for distributed hypermedia systems. This model is based on the transfer of the resource state, where the resource can be any meaningful, addressable concept, and the resource view is mainly a document containing the current state of the resource. The largest REST application is the Web itself, characterized by using the HTTP transport protocol and URL addressing mechanism. REST supports all types of media and XML is the most popular method used for the transmission and presentation of structural information. Services that follow REST are called RESTFul services. This type of services is used in this project. REST is not dependent on any protocol, but almost every RESTful service uses HTTP as the basic protocol. These services are much better integrated with HTTP than SOAP services, and as such do not require XML SOAP messages or WSDL definitions. Because it is much simpler, REST has almost completely replaced SOAP and WDSL. This is the first reason why we developed system with RESTFul service, next is that here we do not need Statefull service. From client part we need just data as he is originally so stateless is perfect, without any additional parts between client and server. RESTful services should have the following features and characteristics: no state (Stateless), uniform Interface-URI, explicit use of HTTP methods, transfer XML and / or JSON. All of those features exists in *testIA*. With this type of service, resources (such as static pages, files, database data...) have their own URLs or URIs that identify them. Access to resources is defined by the HTTP protocol, where each call is one action (it creates, reads, modifies or deletes data). The same URL is used for all operations, but the HTTP method that defines the type of operation changes. This is the main feature of REST architectural model is different from other networks. The model is exactly the uniform interface between the components. Uniforms interface completely separates the server and client's duties, thus simplifying the architecture itself. HTTP offers a set of methods that we call verbs: GET, PUT, POST, DELETE. RESTful services are used: with limited bandwidth and resources (feedback can be in any form) and in operations that do not use the state. If an operation needs to be continued then REST is not the right approach and SOAP is probably a better solution. Since in *testIA* we do not have an operation which need to be continued then logic choice was RESTful service. Advantages of RESTful service are next: *simplicity*-clients who call REST services do not have to formulate requests for SOAP specification and do not have to pave the SOAP response in order to extract the result from it; *the flexibility of the format of the returned data*-the format in which the data is returned is not predefined and depends on the service itself. Clients can request data in the format that suits them best, unlike the SOAP format, although it has to be standardized, it must parse. So JavaScript can get data in JSON format that can read easily, and the RSS reader in the RSS-XML format that it can display. *using existing network infrastructure*;*quick mastering technique*. RESTful services are focused on resources and how to provide access to them. When designing a system, the first thing we are paying attention to is the identification of resources and determining how the resources are tied to one another. The principle is similar to the design of the database: identification of entities and connections between them. Once we identify our resources, the next thing we need to do is to present resources to you in our system. REST allows you to use any format for resource

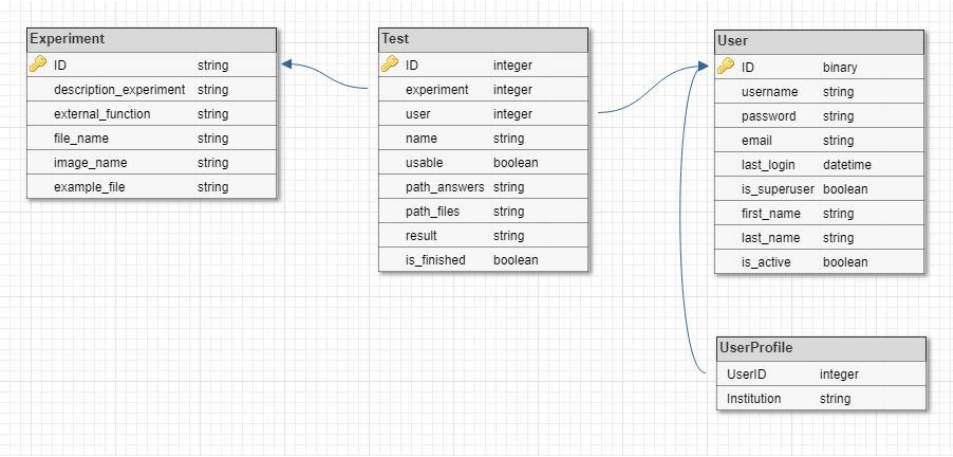

Figure 1: Database schema for *testIA*.

presentation.

2.3. Architecture of application

In this work I developed a web application which consists of two parts. One part is the user interface, where term user correspondents to any researcher from science community who wants to make some test with own dataset on already existing experiment, this part is front-end part of the application. Second part is for administration of the web application and that is back end part of the system, this part is conceived as a management system for experiments. Since both are developed as Django web applications behind both parts is the same architecture (Figure 3). It is architecture based on MVC design pattern which is explained in previous section together with Django framework. The only differences are in numbers of templates used in their design, in front-end part there are more then on back-end. For back-end part regular users do not have access. This architecture is chosen because of its advantages: fast process of development; easy to grow, for future extensions; the model is separated from the user's view. As database we used postgreSQL. We chose SQL type of database before NoSQL, since it more suitable to keep information for this type of web application. Here we do not have dynamic schema of database so SQL was logic choice. Between different DBMS in SQL we chose postgreSQL because it is more suitable for application which is developed in Django framework.

2.3.1. Server equipment

testIA runs on server which is owned by University of Balearic Islands(UIB). It is run on virtual machine. Virtualization allows physical compute, memory, network, and storage resources to be divided between multiple virtual entities. Each virtual device, in this case is *testIA*, is represented within its software and user environments as an actual, standalone entity. Configured properly, virtually isolated resources can provide more secure applications with no visible connectivity between environments. Virtualization also allows new virtual machines to be provisioned and run almost instantly, and then destroyed as soon as they are no longer needed. Configuration of our virtual device is next: the IP address is 130.206.30.141; Names of DNS (Domain name server) testia.uib.es, testia.uib.cat, testia.uib.eu. Those are for three different languages: catalan, spanish and english, respectively. This languages follows the general politics if UIB for officiall languages. For now, is accessible only from UIB network, but the plan is to put web application available to more widely researcher's community and will be available only on English language. The operating system on which server is run is Linux Ubuntu 16.04.3 LTS globally well-known Xenial Xerus. On the server our application has 14GB of space for root directory and 20GB of files important for the application. Since the files for application and for experiment do not take a lot of space, the huge amount of that space will be used to save datasets of users. About how will we treated that saved datasets and information I will say something later in section for security. *testIA* can be achieved by port 80 or 443. Port 80 is with less security and 443 is with SSL (Secure Socket Layer) in order to have a more powerful application in view of security.

2.3.2. Database

As I mentioned before, we used PostgreSQL for the database. In designed schema of *testIA* I have 4 different tables: Experiment, User,Test and UserProfile as is shown on Figure 1. With the last one, UserProfile, is left the possibility to change the schema in the future. If someone wants to add some new attribute in order to make *testIA* more useful or if there are needs for some new data. Every of those tables have their appropriate class in Model part of the architecture (Figure 3). Experiment is table in database which is responsible to save information about experiment which is created by ad-

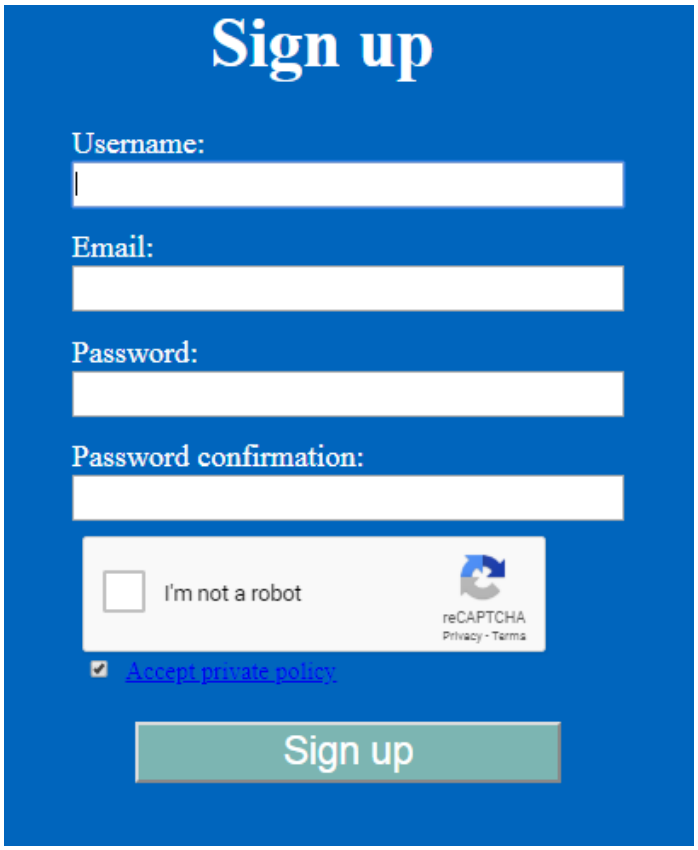

Figure 2: Appearance of Sign Up template

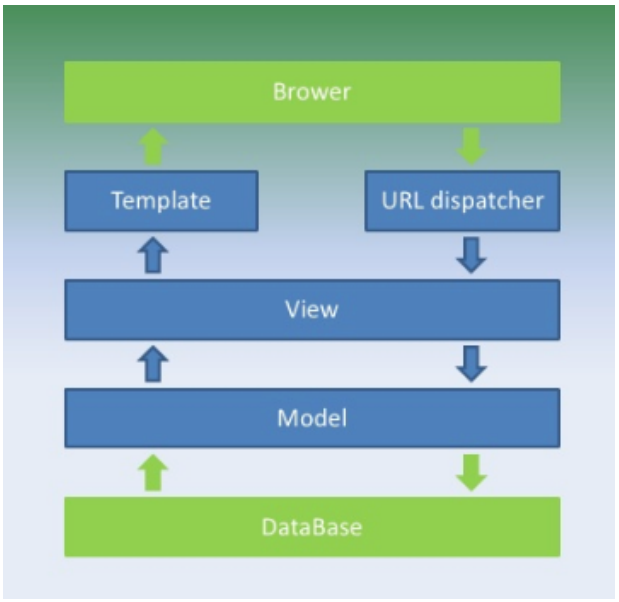

Figure 3: MVC design pattern used in both, back-end and front-end part of testIA

ministrator. For parameters it has ID, which is auto generated like for all other tables; description_experiment - which is here to save explanation of experiment and to give useful information to researcher; external_function - here is the name of main function of experiment, with this is left opportunity for creators of experiment to give different names for different experiments; file_name - keep the name of file in order to import that module when experiment is in running mode; image_name - keep the name of cover image which is used to show to researchers main aim of experiment or some important characteristics; example_file - creators of experiment can added this file for the researchers, inside can be deeper description of experiment and also examples of images used in experiment. Table User keeps information about users of the system. There are username, password, email - in order to contact them when results of experiment are ready, is_active - only users which are activate their account can login, etc. Table Test keeps informations about individual tests run by users. There are attributes necesary to model that: which user is run the test -user, on which experiment was launched test, information about uploaded dataset - path_files, boolean value is_finished which is there to change the state from is running to finished and etc. During development of application we tried also to run *testIA* with SQLite on localhost server with IP address 127.0.0.1 in order to compare performance of two different DBMSs. The results was pretty the same because they both are SQL DBMS and has very similar characteristics. As we can see from Figure 1, there are some relations between those tables in database. Every test has reference to one Experiment, since we should know on which experiment test was launched. Every Experiment can have a larger number of tests, so, this database relation is one to many. Second relation is between User and Test. Every Test has reference to one user which launched it. With this we know which user run, which experiment and later we can give him opportunity to download just results of a test which he was launched. One User can have more tests related to him. Also, this relation is one to many, if we talk from database view. The last relation between User and UsesProfile is one to one relation, one User has reference to just one UserProfile and opposite.

2.3.3. User interface

The user interface is one for researchers. During development of it, I tried to make it so straightforward and for easy to use. Those things were the mainly aims for that part of application. Also, I want to make that interface to be clear, because clarity is one of the key characteristics of good user interface. This means that in every part of application user needs to know what he can do and how. I did not want UI which is confused and frustrating for users. In order to get this characteristics I made several explanation of User interface. As is shown on Figure 5, where is flowchart of front-end part of testIA system, using web application starting with login page. If researcher already have their credential they can put it here and after successfully login they will come to home page. In case that they forgot theirs password I create 'reset password'

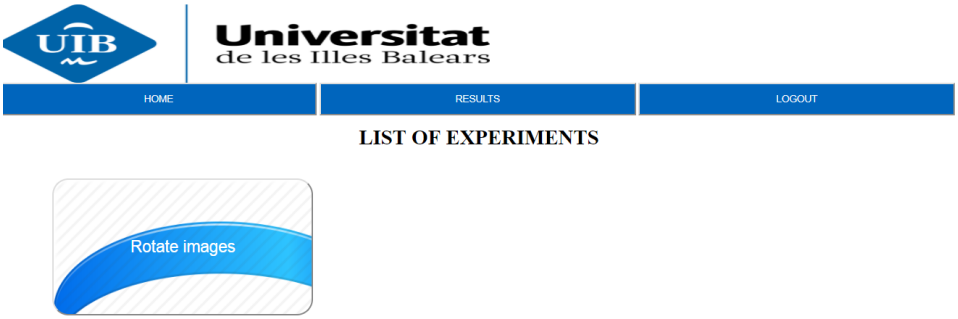

Figure 4: Home part of *testIA*

part where they can reset password and get new one through email provided during registration. For this purpose are created two different templates, login template and home template which is shown on Figure 4. If users are not registered from login page they can go to registration part and fill form with username, password and email, this is the second path from login page. The first is from login to home template. After registration they have to activate account on provided email. For registration is created template with name signup. The appearance of signup template is shown on Figure 2. As we can see there is also the one box for recaptha which represent one type of security in testIA. Also, before registering new users should accept the terms and conditions by click on check box. They can read it on given link next to check box. About security, terms and conditions will be talk for two sections after this. The home page is a place where researchers can choose on which one existing experiment they want to run their tests. On this part is reflected a desire of simplicity of the UI. There are big cards with images of experiments with the name of experiment. Appearance of it is shown on Figure 4. There is one existing experiment which simply rotate the image. That is not confusing for the users which with one click choose desired experiment. Part for the test is also very straightforward. After choosing the desired experiment users come to test page where they have a description of an experiment, and part to upload their dataset. In this point is placed important thing for reproducibility of the experiment. Every user can run test with his own dataset. He starts it by click on button Start. The complete work flow is: First is login, then choosing the experiment, after that uploading the dataset, starting test and on the end users will get notification when results are available in the application. Notification will be send by email on the address provided during registration. In application there is part which is called Results where will be placed all results for current user. For results part is designed template with name results. There are buttons where by click on them user can download the results of every test. This user interface is evaluated using one survey with 10 people. They have tested the application on the same machine. From their opinion, I saw that the simplicity of this UI is achieved. UI is written in HTML(HyperText Markup Language) language. With this User part of application I tried to achieve reusability, but without code sharing. Every researcher can come to testIA and run available algorithm with his dataset. With that he can get some new result of experiment or can get some unexpected conclusion.

2.3.4. Admin interface

Second part of application is developed for administration and they are responsible for making different experiments. It back-end management system for add the experiments. Access to back-end have just super users, the ones which are making experiments. On this way roles in application are separated. Regular researchers are there to make their tests and administrators are there to make new experiments. We need that in order to make our application more usable, since we can have more different experiments and on the end with that more opportunities for researchers. For the administrators there is just one page. Before that they need to login through login page like normal users. The difference is that they after login go to that one admin page which is shown in Figure 6. Through it they can upload information and code for the experiment. Also, here they need to check the libraries for new experiment. With this they have control of available algorithms which can be run on testIA. The list of things which administrator have to provide during creation of experiment is the following: Description of experiment, main function name, name of experiment. python module-code of experiment written in python, cover image. File with examples is optional. Description of experiment is a text which will be shown to users when they click on desired card on home page (Figure 4) and it should describe in several sentences the experiment. Main function name is there for programming reasons. With

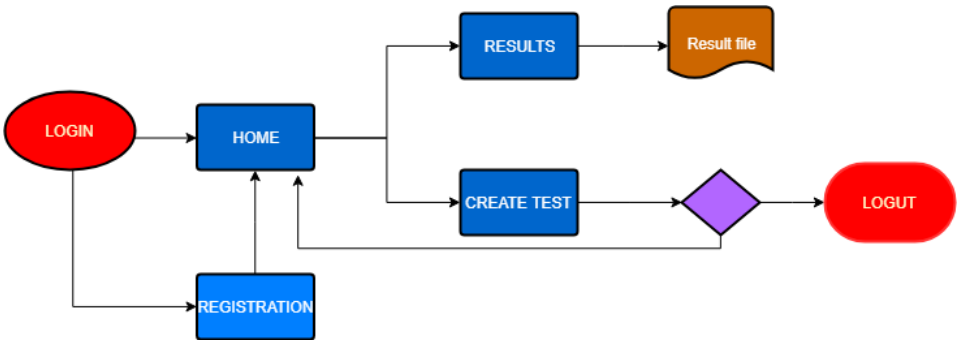

Figure 5: Flowchart for *testIA*

this name we left to author's of experiment freedom to give their name. Name of experiment and cover image appear on the card on home page. Example is zip file where authors of experiment can put whatever they think that is important to researchers to know for example images, detailed description of experiment, etc. After his click to button start experiment becomes available for users. For purpose of this management system is developed one template. The appearance of this part is shown on Figure 6.

2.3.5. Views

In previous subsections are given observations about two of three main parts of the MVT design pattern. The model is processed in subsection Database, and template is processed in last two subsections as a User and admin interface. Now is time to view, the part which connects those two. In testIA views are developed in file views.py, since the convention is to put views in file with that name. As we can see from extension they are written in the Python programming language. Every view is represented with one python function. The input parameter which all views have is web request which contains information from user's side. For example, information about logged user, information about data which are sent from user's browser, etc. Some of the views have additional input parameters which are usually value of ID for database of an experiment, value of ID for one test or user. They returns web response which contains web page which will be rendered on user's side. Some of them with web page returns a list of objects which should be shown to users. Inside every view communicate with the database through model, does some processing of them, and then depending of the request save informations to the database or just put to web response those informations and return it. The featured view is one where test is run. There is used django background task module in order to put the execution of test, which can take a lot of time for a large dataset, in the background. The user immediately get the notification that test is running and when it is finished he will get an email. With background task we have fast response to user and execution of test is separated. Here exists one question: How django knows which views is correspond to a request which is coming? For this there is URL dispatcher. For it, I created a Python module called a URLconf, from URL configuration. This module is pure Python code and is a mapping between URL path expressions to views. With it django knows which function(view) call when some request is on the server.

2.3.6. Security, terms and conditions

During development of *testIA* security was taken into account. There are several responsibility levels of it. First is using SSL, for which we need to get a certificate in order to have HTTPS (Secure HTTP protocol). With this the data which users send from their computer to our servers is safety, since SSL is protocol, which enable secure point-to-point connection between client and server. All data from the client is encrypted and on that way send to server. Next level is that *testIA* provides a login / logout system where just logged users can access to the experiments. On this way we are sure that users who are not logged can not come and run any experiment or download the results. There is also reCAPTCHA system, which is designed to establish that a computer user is human or bots(machine), in order to protect websites from bots. Before registration every user need to check reCAPTCHA field. Whereas reCAPTCHA assists in the digitization of books, *testIA* with using it helps in that big project, too. *testIA* with its Terms and contition satisfies the EU GDPR (European Union General Data Protection Regulation). Users before registration have to accept those terms, since without that registration is not possible. The GDPR aims primarily to give control to citizens and residents over their personal data and to simplify the regulatory environment for international business by unifying the regulation within the EU. The most important parts of this regulative in our case are how we care about data of users and about theirs uploaded datasets. The personal information we will save on the way that satisfy rules of EU. For dataset they can choose will give us permission to use it in next researchers or not. In case when they allow us to use it the future that dataset will be saved on server after test execution.

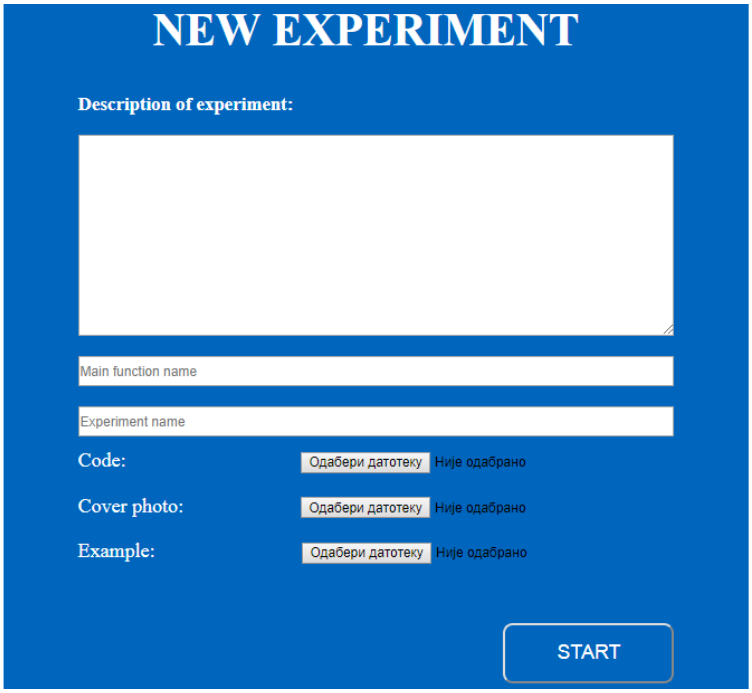

Figure 6: Back-end part of *testIA*

3. Results

We use one example here to demonstrate how *testIA* works. Any researcher can run any available experiment. The user can download the results of execution from the web application.

3.1. Cell classification

First, in order to create and run an experiment, the user needs to have administration's privileges. It is superuser of *testIA* system or administrator. We need to create it through command line. After the administrator's creation, he has access to administration interface. Through it, the administrator can create an experiment. The experiment is developed by Natasa Petrovic from UGiVIA at UIB. This creation is like plugging one module to *testIA*. We upload the code and when the test is in execution, this module will be imported into the system and executed. Developed module classifies the red blood cells using machine learning (ML) algorithm. This experiment is one which covers full image analysis process for given dataset. Inside the module were used 7 different ML algorithms for classification: SVM (Support vector machine), decision tree, extra trees, gradient boosting tree, random forest, KNN (K-Nearest Neighbors) and multilayer perceptron. The result of it is one file where is summarized numbers of three different types of cells (circular, elongated and others) by every algorithm. Experiment is created by uploading one file with code through the admin's interface. The requirement for the new experiments is that they have to include the main function with the parameters which follow some *testIA* rules. Also, along side with the code, user uploads also a cover image of experiment, sets the name of main function and gives the description of experiment. Before the creation of an experiment we should check if there are dependencies to be installed. Since a lot of dependencies are installed beforehand, we did not have to add any. Experiment is created by clicking on button start. After this we obtained one available experiment. For execution of this experiment, we need to have one regular user. We created it through user interface. After the successful registration, test user could see one available experiment where he can put his dataset and run his test. The part for create test dataset is uploaded and after that test started. User first got notification that he will get a notification email when test is finished. After that notification, he can visit the results page and download the results. This way any user can come and execute his test on given experiments with theirs datasets and with this we obtain the reusability, since experiment can be used any number of times for different datasets.

4. Conclusion and future work

In this paper we exposed one possible solution to improve reusability in computer science, since we enabled researchers to reuse already designed experiments for their dataset. Also, in this way we fufilled our secondary objective, which is to use their datasets for the future research. One of possibility for future work is to give access to those datasets for whole research community. Also, one of the possible improvements can be

introduction of parallel programming. This could improve the speed of running the classification algorithms. Parallelism can also improve the image segmentation process.

Apéndice .1. User manual

Apéndice .1.1. Login

When user come to web application first he will see login page.
- In field Username user should put his username, in Password field his password if they are already registered. And after click on button "Login".
- If user is not registered he should click on button "New member.ªnd fill the form for registration.
-If user is registered but he forgot his passsword he should click on link "Forgot password". After that user have to put his email in the field and click button "submit". Than have to go to his email, click on link in email and after that fill the form where will provide new password. After that can log in with new credentials.

Apéndice .1.2. Sign up

- User need to put username, email on which he will get notifications and password. Password has to be at least 8 characters and can not contains some well known words like qwerty, 12345678, etc. After click on button "Sign Up"they need to go to provided email and activate account. Without that they can not login and they are not registered. -

Apéndice .2. Home

-On this page users just need to click on one available card and on this way they choose experiment which what to reuse.

Apéndice .3. Test

-Here user have to fill the form about his test. First should put the name of test. Then, dataset. Here is one option to put his answers if that is necessary for the experiment. And also need to give permission to use his dataset in the future researches. By clicking on the button "Start"test is go to execution and user is redirected to page where web application notify hit that he will get an email when results are ready.

Apéndice .3.1. Results

- To this part of application user can come by click on -esult"button in navigation. From here user can download results of his finished tests. Just need to click on button with the name of the test which want to download.

Apéndice .3.2. Backend part-admin

- Admin need to fill a form in order to add new experiment in system. First should provide description of experiment. Than name of main function in code, name of module, module code and one example file where can put any information regarding to experiment which he creates. By clicking on "Start"button experiment is added to system. If he needs some new dependency, which is not already installed he has to put the name of the dependency in format: name_of_dependency === desired_version_of_dependency in the appropriate field and click add. Bellow that field is list of installed dependencies.

Referencias

[1] Dr. Anirban Mukhopadhyay *Machine Learning in Image Analysis - Theory and Practice - http://www.zib.de/MLIA* (Accessed June 11, 2018)

[2] International Vocabulary of Metrology *Basic and General Concepts and Associated Terms (VIM), 3rd edition, JCGM 200:2012, http://www.bipm.org/en/publications/guides/vim.html*. (Accessed June 11, 2018)

[3] Steven N. Goodman, Daniele Fanelli and John P. A. Ioannidis *What does research reproducibility mean?*. American Association for the Advancement of Science, 2016

[4] Hans E. Plesser *Reproducibility vs. Replicability: A Brief History of a Confused Terminology*. Frontiers in Neuroinformatics 11 (2017): 76. PMC. Web. 11 June 2018.

[5] Roberta B. Oliveiraa, Aledir S. Pereirab and João Manuel R. S. Tavaresa *Computational Diagnosis of Skin Lesions from Dermoscopic Images using a Combination of Features*. Manuscript Draft

[6] Cristian Varela *https://medium.com/@cvarelaruiz/why-i-love-python-and-django-26596ce4d82e* (Accessed June 11, 2018)

[7] Ali Ahadi *http://comtech2.com/web-services-architecture-when-to-use-soap-vs-rest2/* (Accessed June 11, 2018)

[8] Takeo Kanade and Zhaozheng Yin and Ryoma Bise and SeungIl Huh and Sung Eun Eom and Michael Sandbothe and Mei Chen *Cell Image Analysis: Algorithms, System and Applications* IEEE Workshop on Applications of Computer Vision (WACV) 2011

[9] *https://www.ncbi.nlm.nih.gov/pmc/articles/PMC1794559/* (Accessed June 11, 2018)

[10] *https://www.w3.org/TR/soap12/* (Accessed June 11, 2018)