**INSTITUTO POLITÉCNICO NACIONAL**

**ESCUELA SUPERIOR DE CÓMPUTO**



**Final Project:**

**“Digital Portraits”**

**Learning Unit:** Cryptography

|  |  |
| --- | --- |
| **Team members:**  Victor Nolasco Cid  Mayra Sofía Hernández Oseguera  López Ayala Eric Alejandro | **Professor:** Sandra Díaz Santiago |

**Deadline:**

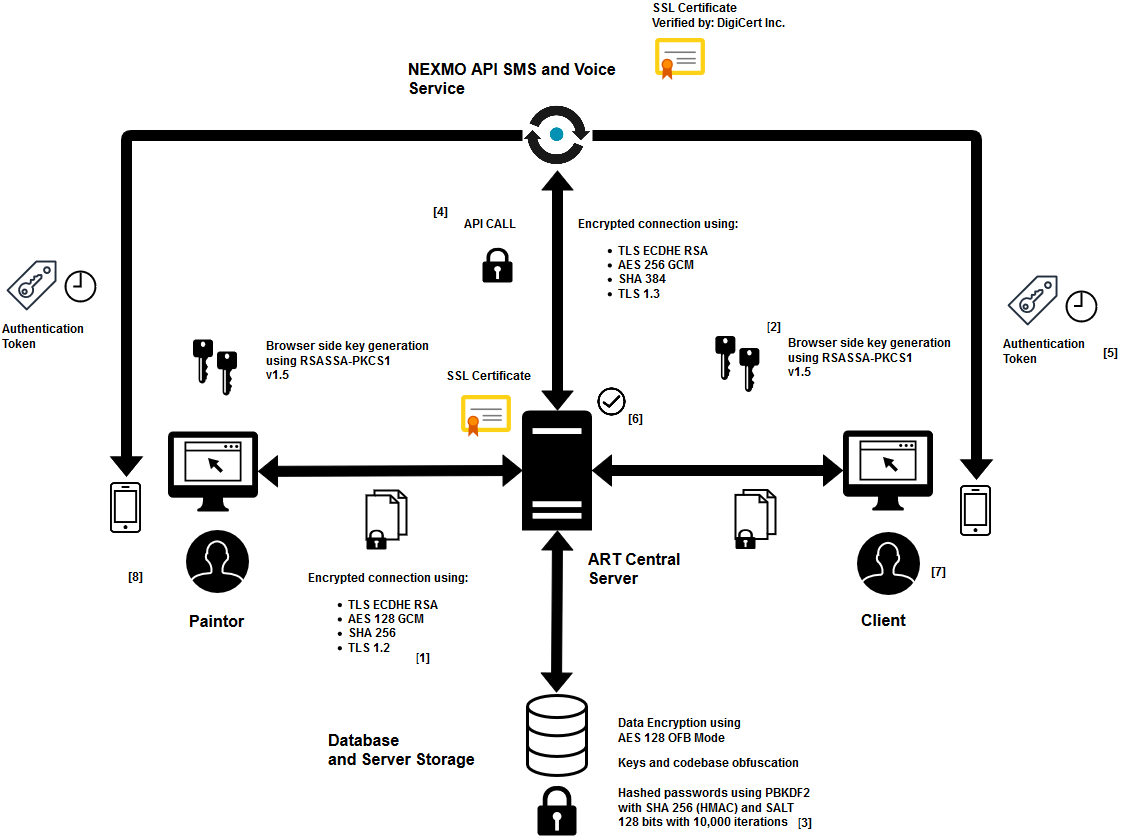
June 3rd 2019

**Problematic**

The problem that we had to resolve was the following: A painter comes up with a new business idea, he wants to offer custom paintings from photos. Both the photos and paintings will be transmitted in digital form via Internet. One concern that he has is discretion towards his customers, since potentially embarrassing photos, e.g. nude photos, might be sent to him. Hence, the photo data should not be accessible for third parties during transmission.

The painter needs multiple weeks for the creation of a painting, and hence he wants to assure that he cannot be fooled by someone who sends in a photo assuming a false name. He also wants to be assured that the painting will definitely be accepted by the customer and that she/he cannot deny the order. Choose the necessary security services for the transmission of the digitized photos from the customers to the painter.

**Solution**

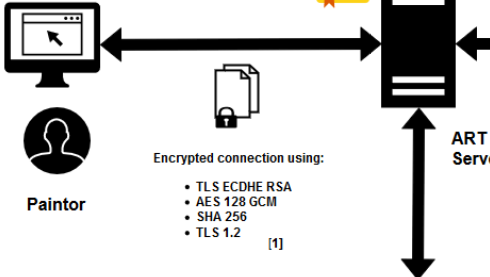


*Diagram of the main components of the system.*

**Description**

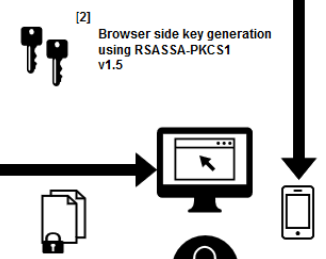
**[1] Client - Server connection**

The connection between the client and the server is secure due to the use of a digital certificate, which provides an HTTPS connection through the use of SSL and TLS. All the messages that are sent from the client to the server and vice versa are encrypted using AES 128 in GCM mode, also the integrity of each message is verified using SHA 256. Using the certificate we guarantee the integrity, confidentiality and authentication of the server, of each of the messages transmitted in the communication channel.



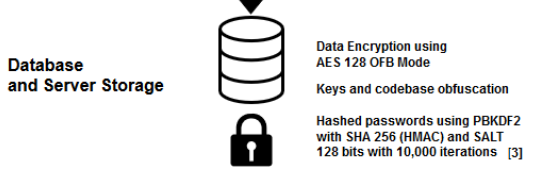
**[2] Browser side key generation**

It is required that the generation of the key pair (public and private) of each user is on the client's side, for this it is necessary to make use of the Crypto API cryptographic library, which allows us to make use of several cryptographic primitives from the browser of the client, using Javascript code. The user has access to the generation of its keys once they have logged into the system, and can request the update of them whenever required, the system will request the private key to the user each time he wishes to make a new order, in this way we can validate that the order was made by the user and can not reject the order once requested (The user is responsible for the management and administration of their keys).



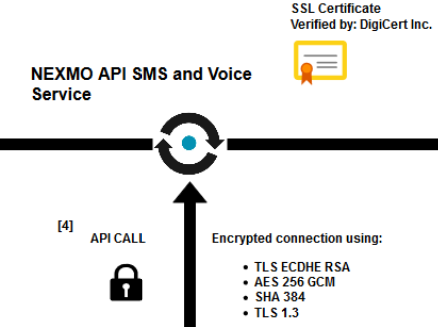
**[3] Database and server storage**

As any other web based application, we needed a database to store information related with our business. The database that we used is MySQL that runs locally within our server. There the sensitive data of our user is secured and stored, additionally we designated a space in the server's storage to store the images and it's keys (encrypted using AES 128 in OFB mode). Also the passwords are stored in the database using PBKDF2 algorithm with HMAC-SHA256 and SALT 128 bits with 10,000 iterations, according to NIST recommendations.



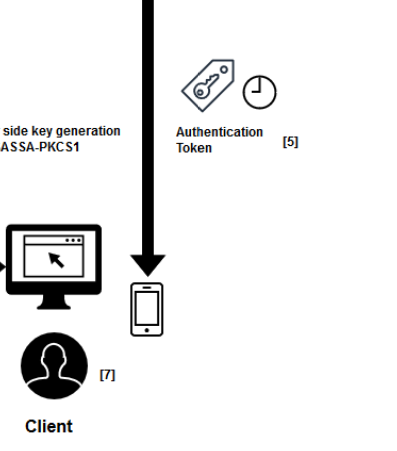
**[4] Nexmo API SMS and Call service**

One of the features that our system has is the two factor authentication at the moment a user sign up, to provide this service we used NEXMO, which is an API that allow us to generate, manage, distribute and remove authentication tokens. When a new user sign up and the registration form has been filled correctly, our server makes a call to the API in order to generate a new token for that user, once the API has generated the code it sends it via SMS to the user (In case the user doesn't insert the authentication code within the first 2 minutes after SMS arrival, then the API makes a phone call to that user to repeat the code). If the user has used that code or 5 minutes had already passed after SMS arrival, then the token expires. The only data that is send to the API is the phone number of the user and NEXMO API credentials of the server, we receive a verification id linked to the token to make the user validation. NEXMO service uses a SSL certificate for the data transmission, so the communication channel through which our data is sent to the API is secure .



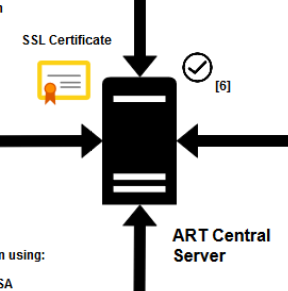
**[5] Token authentication**

At the moment a user sends its registration form, an authentication code will be send to him|her via SMS, once it has received the authentication code, it has 5 minutes to use it and complete the registration, otherwise the token will expire and the user will have to proceed to do all the registration process once again. Once the user has used it's code it will be redirected to the login so it can access to the application.



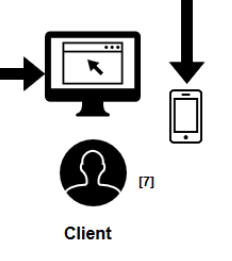
**[6] ART central server**

Our web application uses a central server to manage the connection with users, manage requests and manage the services we offer. Within our server, the validation of the orders is performed, the access of the users to their respective views; Another function is to store the information generated by the application as well as make the respective calls to the NEXMO API.



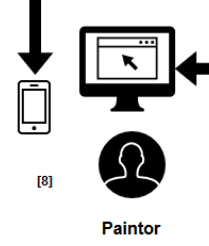
**[7] Client**

The client is one of the actors that interact with our system, whose functions consist in request new orders, visualize deliveries and take the administration of its keys. It is also him|her responsibility to register within the system and complete the verification process of two factors.



**[8] Painter**

The other actor that interacts with our system is the painter, whose functions consist in visualizing the orders requested by the clients, downloading the original images sent, making deliveries of portraits and carrying out the administration of its keys. It is also him|her responsibility to register within the system and complete the verification process of two factors.



**Cryptographic services**

The cryptographic services that we cover are listed below:

**Integrity.**

One of the requirements established by the painter is that the data of the order should be correct and complete throughout the whole process (from the moment a portrait is requested, to its deliver) in order to avoid misunderstandings or wrong requests that lead to further problems. This cryptographic service is covered in multiple stages of our system; every order is signed with the client’s private key, as well the deliver it’s signed with the painter private key. Once it has arrived to the server, a verification process is been made so we can validate the order. In this process we verify the image sent to assure its integrity. Also the communication channel implements a digital certificate, that implements SSL and TLS to provide integrity, by calculating a message digest using SHA256, in order to validate the data integrity of the messages transmitted.

**Confidentiality.**

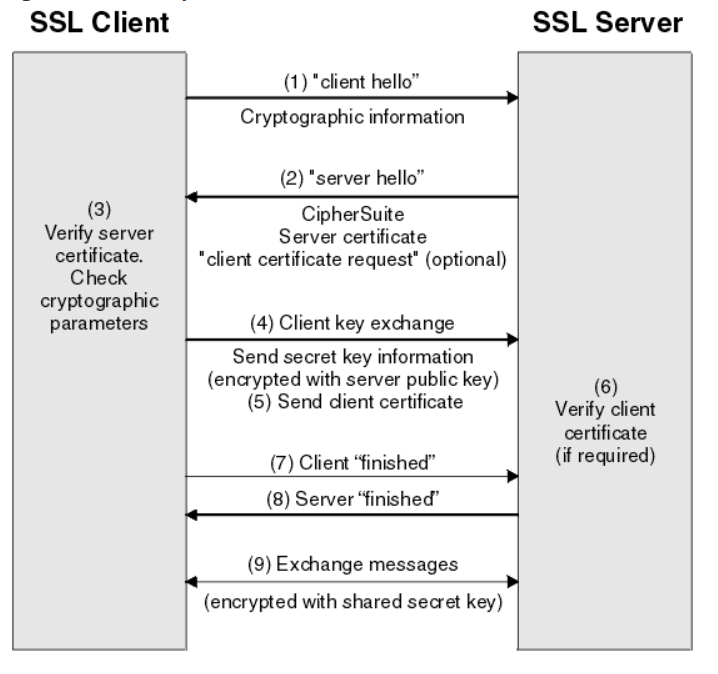
One of the problems with transmitting digital images vía internet is that those images might be embarrassing and we don’t want everyone to see it, so confidentiality service is required in the sensitive data transmission. This cryptographic service is covered in multiple stages of our system; when the client or the painter send information to the server, all the data request is encrypted before being transmitted. The implemented digital certificate uses SSL and TLS to use a combination of symmetric and asymmetric encryption to ensure message privacy. During the SSL or TLS handshake, the SSL or TLS client and server agree an encryption algorithm and a shared secret key to be used for one session only. All messages transmitted between the SSL or TLS client and server are encrypted using that algorithm and key, ensuring that the message remains private even if it is intercepted. SSL supports a wide range of cryptographic algorithms. Because SSL and TLS use asymmetric encryption when transporting the shared secret key, there is no key distribution problem.

When the data arrives to the server, the image is decipher, so we need to apply once again another encryption process in order to store the image in our server storage. Given the case that our server has been hacked, the attacker will not be able to see the images, also all the passwords of our users are hashed in order to protect their accounts.

**Authentication.**

Every order that the painter receives can not be sent by someone assuming a false name, so authentication of the users is required in order to assure that all the orders that are received aren’t from fake users. This cryptographic service is covered multiple times in our system; when a user sign up, we implement a two factor authentication as a method to confirm a user’s claimed identity, by utilizing an authentication token that the user will receive in its phone, this way we prevent spam at creating new users, and we have more certainty about the identity of the person. Once registered the user must login using its username and password, this parameters are validated and authentified with the data stored in our database, if successful a new session is created and the user will have access to the system (only to the views that it is allowed).

For server authentication, the client uses the server's public key to encrypt the data that is used to compute the secret key. The server can generate the secret key only if it can decrypt that data with the correct private key. The exchange of finished messages that are encrypted with the secret key (steps 7 and 8) confirms that authentication is complete. If any of the authentication steps fail, the handshake fails and the session terminates. The exchange of digital certificates during the SSL or TLS handshake is part of the authentication process.



**Non repudiation.**

Finally we need to ensure that once the order has been confirmed, the delivery can not be denied by the user, in order to achieve this, non repudiation service is required. In our system this process is made from the client side, once the client has its public and private key pair, the system requires that every order must be signed with the client’s private key so every order is signed and then is verified by the server. If the request is successful then the order is stored and make it available for the painter. This way we can assure that effectively the client cannot deny the order because he has already sign the order and that order can be verified at any moment with its public key (which a copy is stored by us in the server), the user is responsible of the use of his private key.

**Cryptographic primitives**

**AES 128 OFB**

Advanced Encryption Standard (AES) is a symmetric block cipher standardized by National Institute of Standards and Technology.

AES and in general block ciphers provides **privacy** primitive.

We used the following configuration of AES for the images encryption on the server.

* Data block size: 128 bits
* Key size: 128 bits
* Number of rounds: 10 rounds
* Operation mode: Output Feedback Mode (OFB)

Output FeedBack, defined in NIST SP 800-38A. It is an operation mode that leads to a stream cipher. Each byte of plaintext is XOR-ed with a byte taken from a keystream: the result is the ciphertext. The keystream is obtained by recursively encrypting the Initialization Vector.

**AES 128 GCM**

Our SSL certificate use AES, we set the following configuration:

* Data block size: 128 bits
* Key size: 128 bits
* Number of rounds: 10 rounds
* Operation mode: Galois / Counter Mode (GCM)

Galois / Counter Mode, defined in NIST SP 800-38D. GCM provides assurance of the authenticity of the confidential data using a universal hash function that is defined over a NIST Special Publication 800-38D Binary Galois field.

**RSASSA-PKCSv1\_5**

PKCS#1 v1.5 is an old but still solid Digital Signature scheme based on **RSA**. It is more formally called RSASSA-PKCS1-v1\_5 in Section 8.2 of RFC8017.

Digital Signature provides **integrity**, **authentication** and **non repudiation primitives**.

* The hash function that we implement with this scheme is SHA256.

**SHA 256**

SHA-256 belongs to the SHA-2 family of cryptographic hashes. This hash function produces a 256 bit digest of a message.

Hash functions provides of **integrity** primitive, and we used it on the RSASSA-PKCS1-v1\_5 schema for Digital Signature.

**HMAC-SHA256**

HMAC (Hash-based Message Authentication Code) is a MAC defined in RFC2104 and FIPS-198 and constructed using a cryptographic hash algorithm.

It is usually named HMAC-X, where X is the hash algorithm; for instance HMAC-SHA256.

On our SSL certificate we configured a HMAC-SHA256.

**SALT**

The salt parameter is used to derive the key to be used for encryption. Whenever you pass a password to Django authentication module to encrypt something, you might also specify a salt to increase the input entropy of the whole process.

We set SALT with 128 bits and 10,000 iterations, according to NIST recommendations.

**PBKDF2**

PBKDF2 (Password-Based Key Derivation Function 2) is a key derivation function with a sliding computational cost, used to reduce vulnerabilities to brute force attacks.

PBKDF2 applies a pseudorandom function, such as hash-based message authentication code (HMAC), to the input password or passphrase along with a salt value and repeats the process many times to produce a derived key, which can then be used as a cryptographic key in subsequent operations. The added computational work makes password cracking much more difficult, and is known as key stretching.

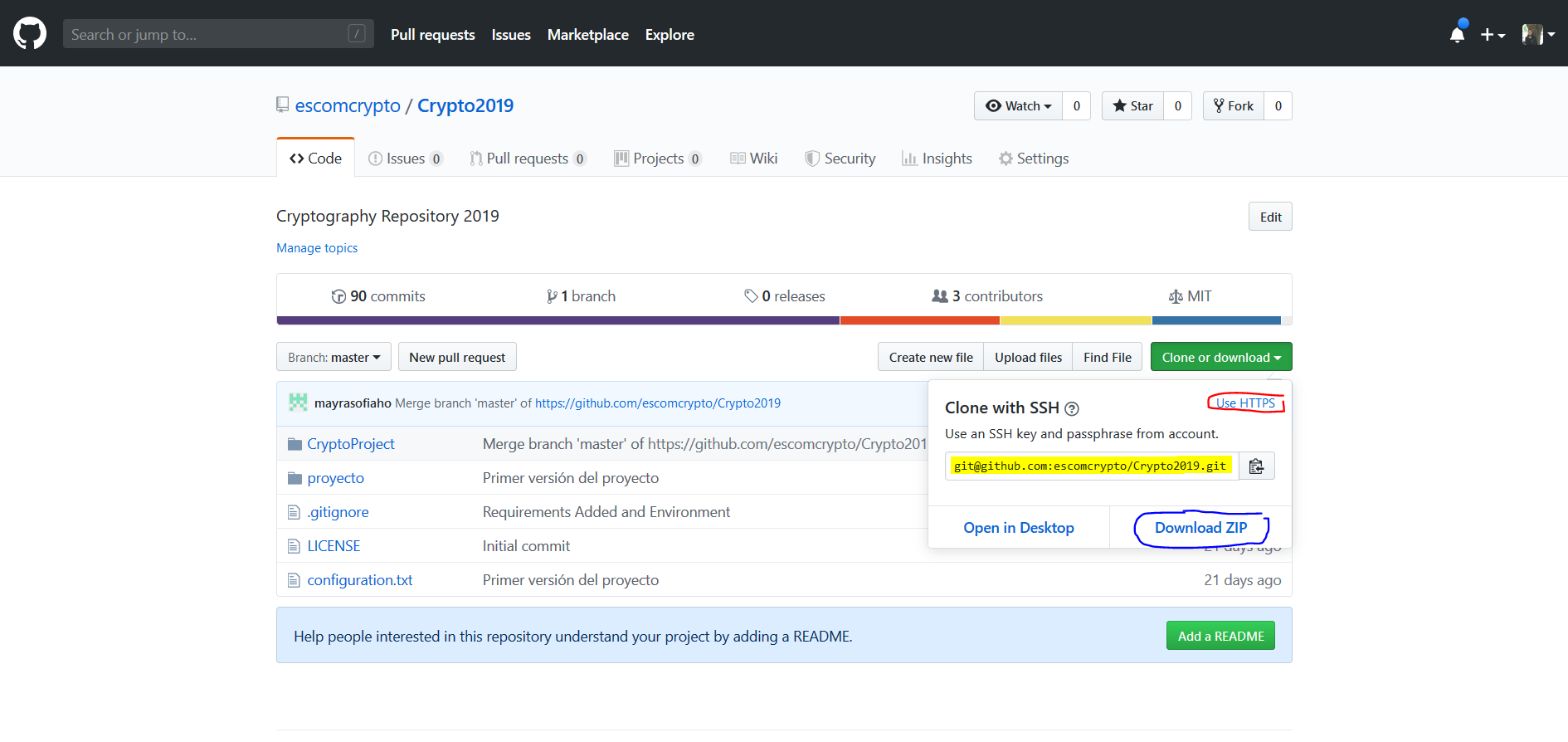
This function allow us to offer both **confidentiality and authentication** of a user.

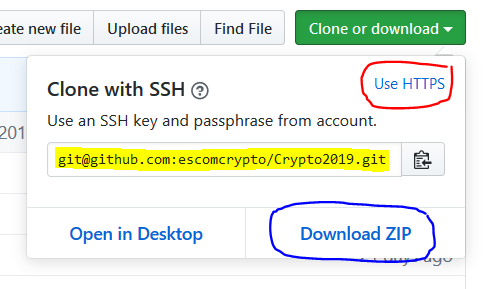
**Installation Guide**

This is a web based application that uses django framework, along with other python libraries, therefore Python programming language should be installed in the computer. We use Python version 3.7.3 which requires 64-bit processors; Python is multiplatform, and runs on MAC OS X, Windows and Linux. The installation process between each operative system may differ, so please check out the one you require at:

<https://www.python.org/downloads/release/python-373/>

Along with python, the project must be downloaded from our Github repository, to download it you have two options:





**Option 1:** Download the ZIP compressed project to your local disk, and then unzip it.

**Option 2:** Clone the repository with git using HTTPS or SSH, by executing one of the following commands on your git terminal:

*$ git clone* [*https://github.com/escomcrypto/Crypto2019.git*](https://github.com/escomcrypto/Crypto2019.git)**(HTTPS)**

*$ git clone* [*git@github.com*](mailto:git@github.com)*:escomcrypto/Crypto2019.git* **(SSH)**

Once python has been installed in the computer, we need to create a python virtual environment, a virtual environment helps us to keep dependencies required by different projects separate by creating isolated python virtual environments for them. The following commands create a new python environment, and make it active (Is important that before executing the following steps it must had already been added Python source to the environment variables of your operative system).

**Step 1:** Open your command shell and go to the root directory of the project.

(Optional in case you don’t have by default): Install the virtualenv package using pip command: *> pip install virtualenv*

**Step 2:** Create a new virtual environment for the project using the following command:

*> virtualenv “venv\_name”*

**Step 3:** Activate the virtual environment. You can activate the python environment by running the following command:

**Windows:** > *“venv\_name”\Scripts\activate* **Linux/MAC:** source *$ “venv\_name”/bin/activate*

**Step 4:** Install all the dependencies of the project. All the dependencies of the project are listed in the requirements.txt file and you can install them all at once by running the following command:

*(“venv\_name”) > pip install -r requirements.txt*

**Step 5:** Once the dependencies have been installed, we need to configure the database and the project migrations. The database that we used for this project is *(MySQL > 3.6)* but you can use any of the databases supported by django. To install MySQL on your computer please check the following link: <https://dev.mysql.com/downloads/installer/>.

After the installation we have to open mysql console and create the database required for the project by running the following command:

*mysql > create database crytoproject;*

**Step 6:** Once the database has been created and MySQL service running according to the configuration file *db.cnf,* we need to do the proper project migrations. To migrate the project, we type the following commands in the console a then we sequentially run them.

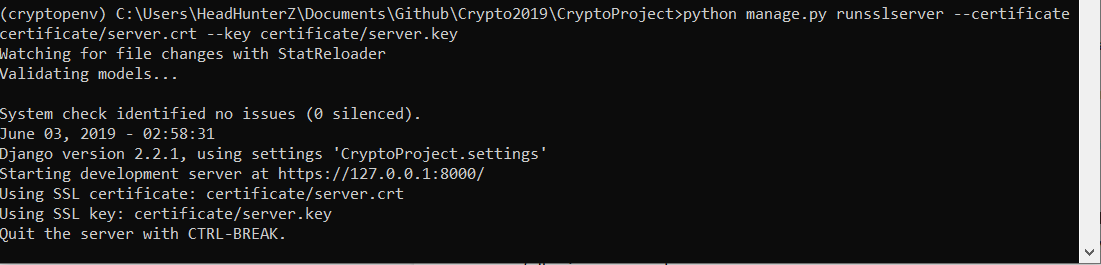
*(“venv\_name”) > python manage.py makemigrations*

*(“venv\_name”) > python manage.py migrate*

**Step 7:** If your virtual environment is active, the dependencies installed and the database with its migrations settled, the only thing left is running the server. To run the server we type the following command in the console and then run it:

> *python manage.py runsslserver --certificate certificate/server.crt --key certificate/server.key*

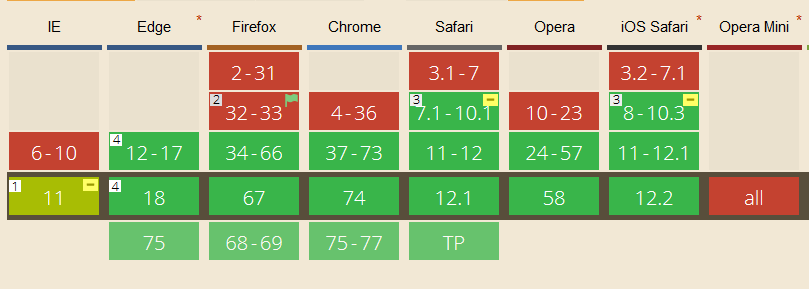
Then on your console the following image should appear:

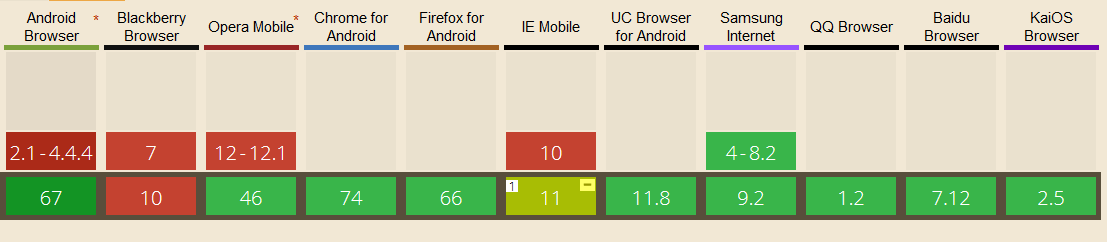


This means that all the previous steps have been successfully completed and now we are ready to use the system. (In case port 8000 has already been occupied by another application, then you may change the port in the settings.py file) .

**Browser Requirements**

Even though Web Crypto API is a standard defined by the W3C, some old versions of browsers might not support it. Please check the following table so you can guarantee the compatibility with yours:



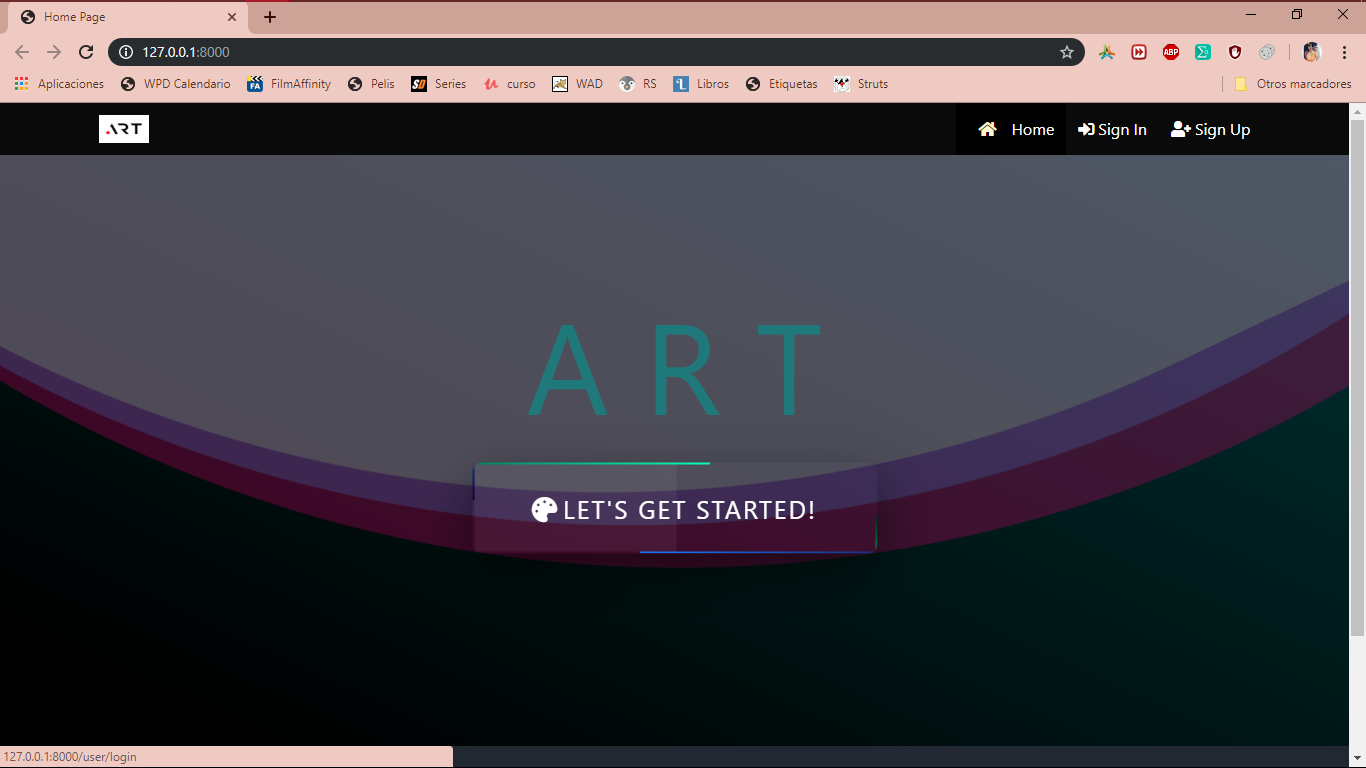


Nomenclature:

* Green: Fully supported
* Yellow: Partially supported
* Red: Not supported

**User Guide**

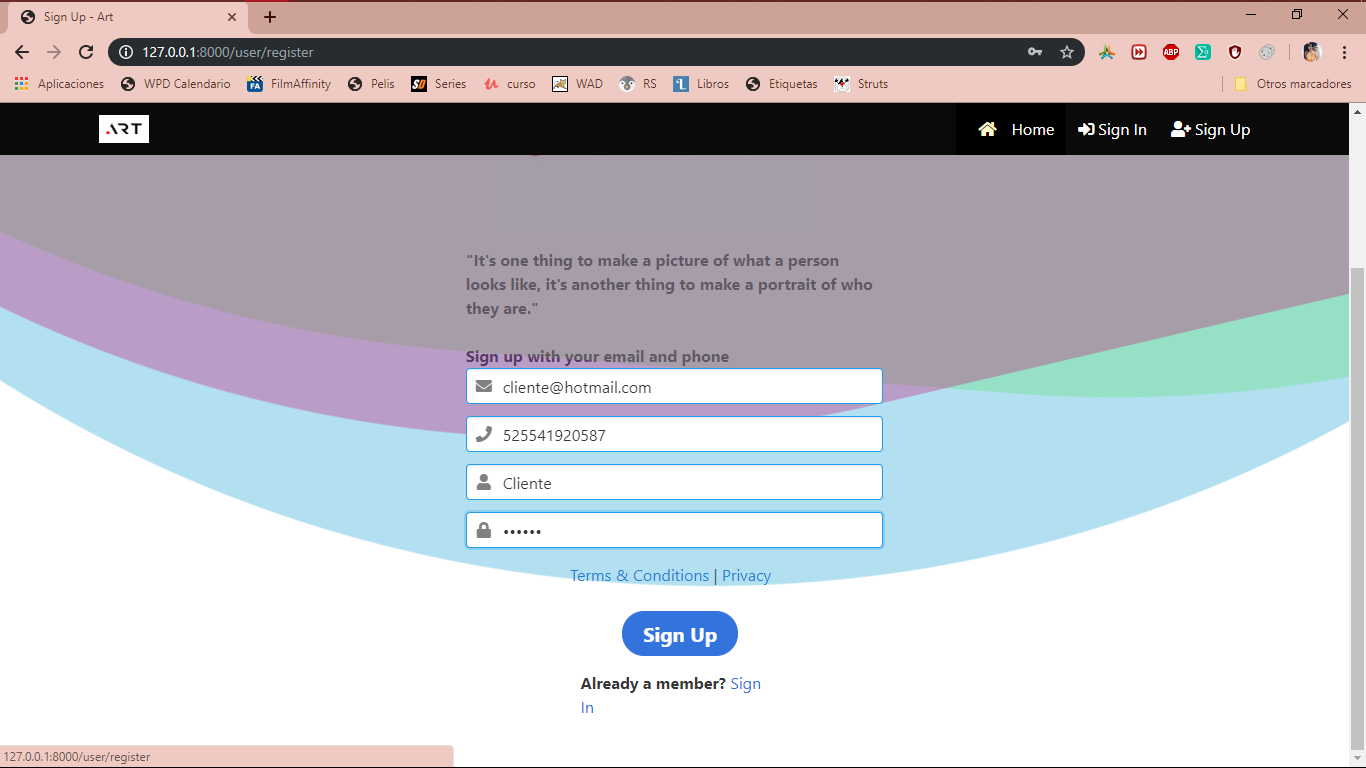
This is the main page of the system, it will appear once followed the steps of the installation guide.

****

*Image 1. Main page.*

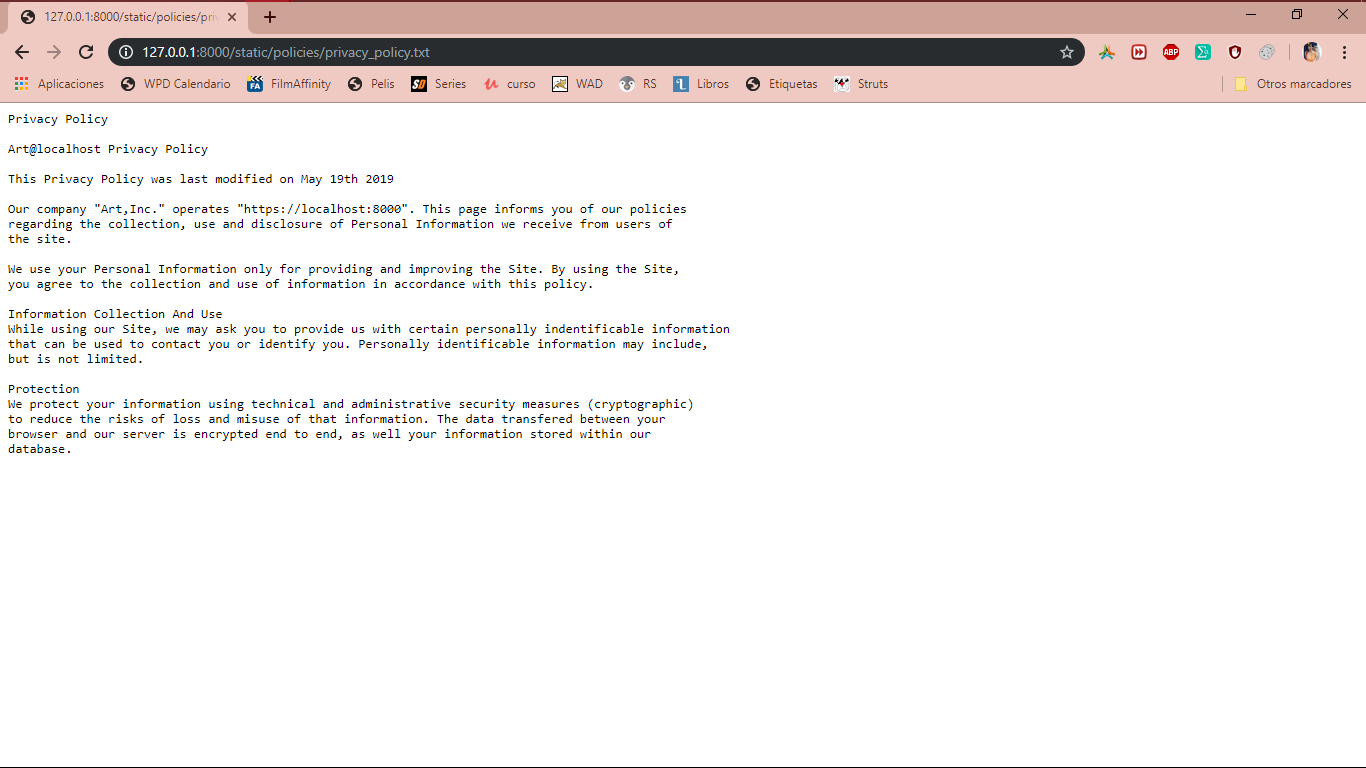
**If you are a client:**

If you are a new customer, the next step is to register from the menu's ‘Sign up’ option at the top of the *Image 1. Main Page*. On this page you must enter your data, email, telephone, adding the name, name, which will be your username and password. After filling all the fields click on the *Sing up* button.

****

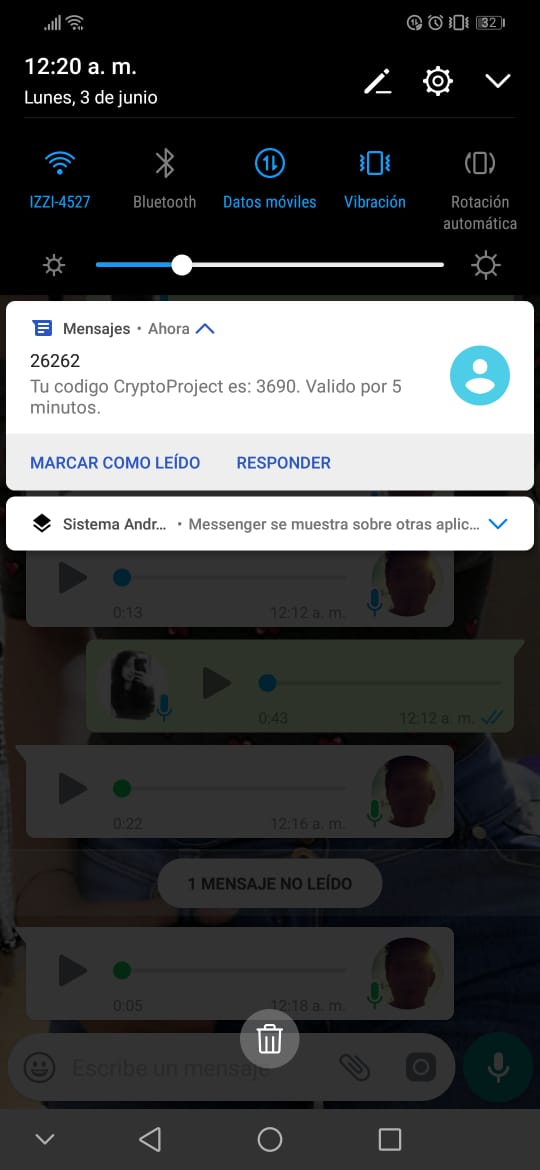
*Image 1.1. Registration page.*

If you click on the privacity link of the *Image 1.1. Registration page*, you can see our privacy policy.



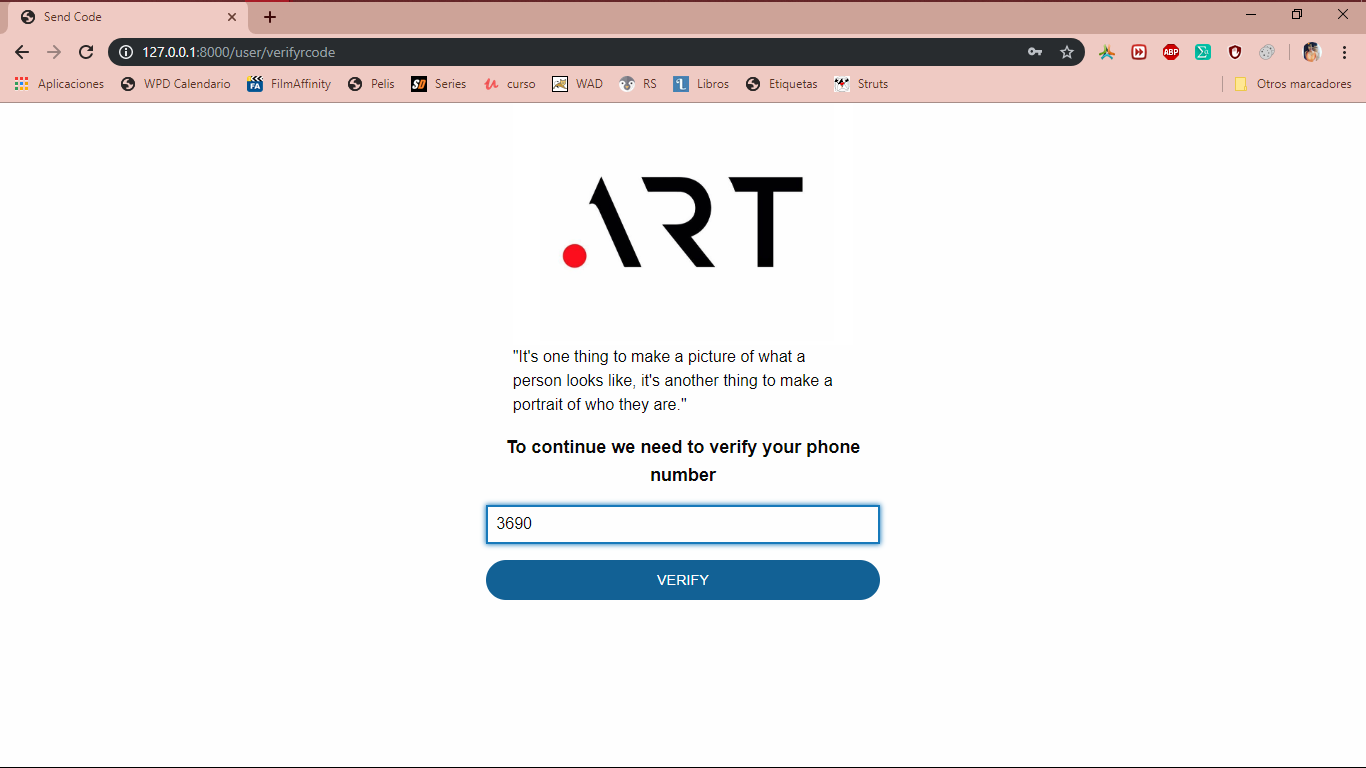
*Image 1.2. Privacy Policy.*

After you have registered, a text message will sent on the phone that you provided in the *Image 1.1. Registration page*, similar to the following, in which your verification code will appear. The message will be valid for the next five minutes, if you do not enter it during the next two minutes in *Image 1.4. Verification Code Page.*, you will have a call indicating your verification code, which will now have an expiration of three minutes.



*Image 1.3 Verification Code Message.*

This is the page where you must enter the verification code that was provided, either via message or via telephone. And then click on *Verify* button.



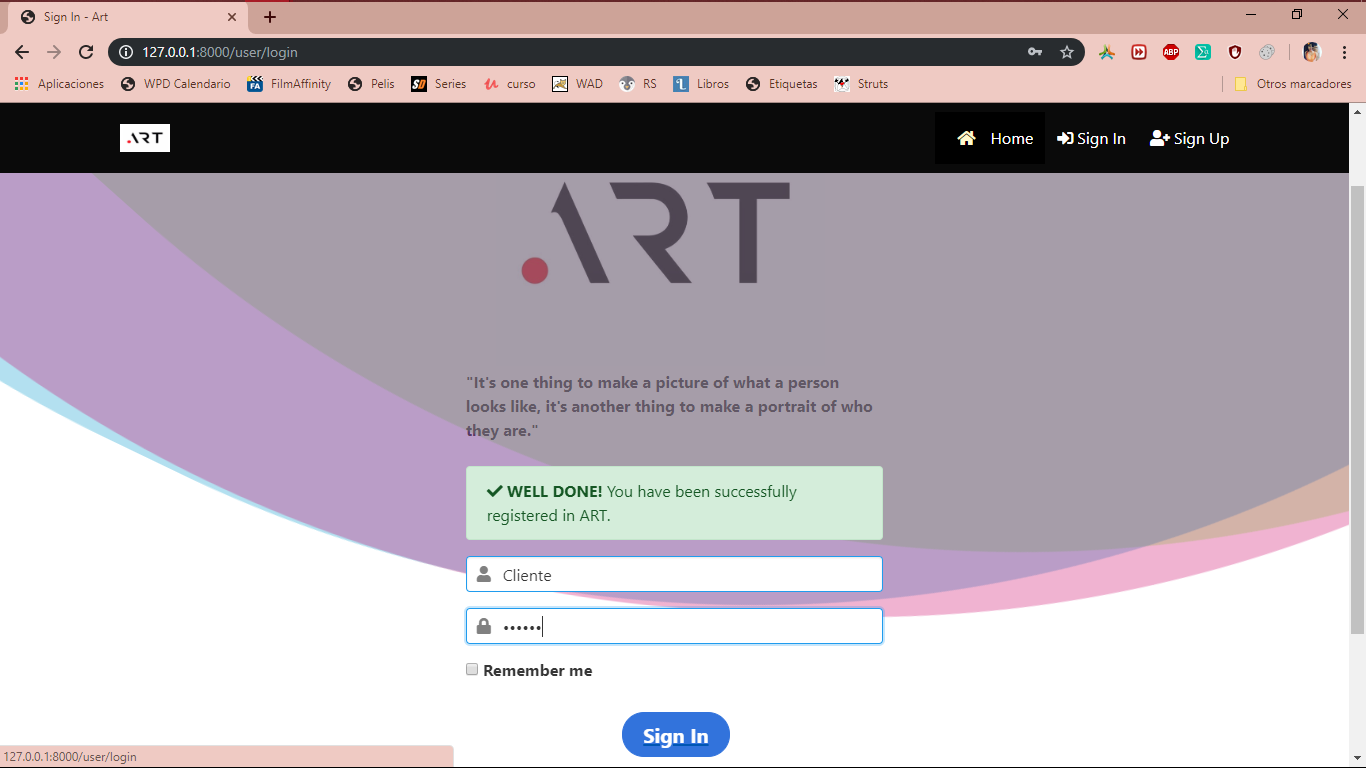
*Image 1.4. Verification Code Page.*

After the above, this page will appear in which, you will be able to enter the system.

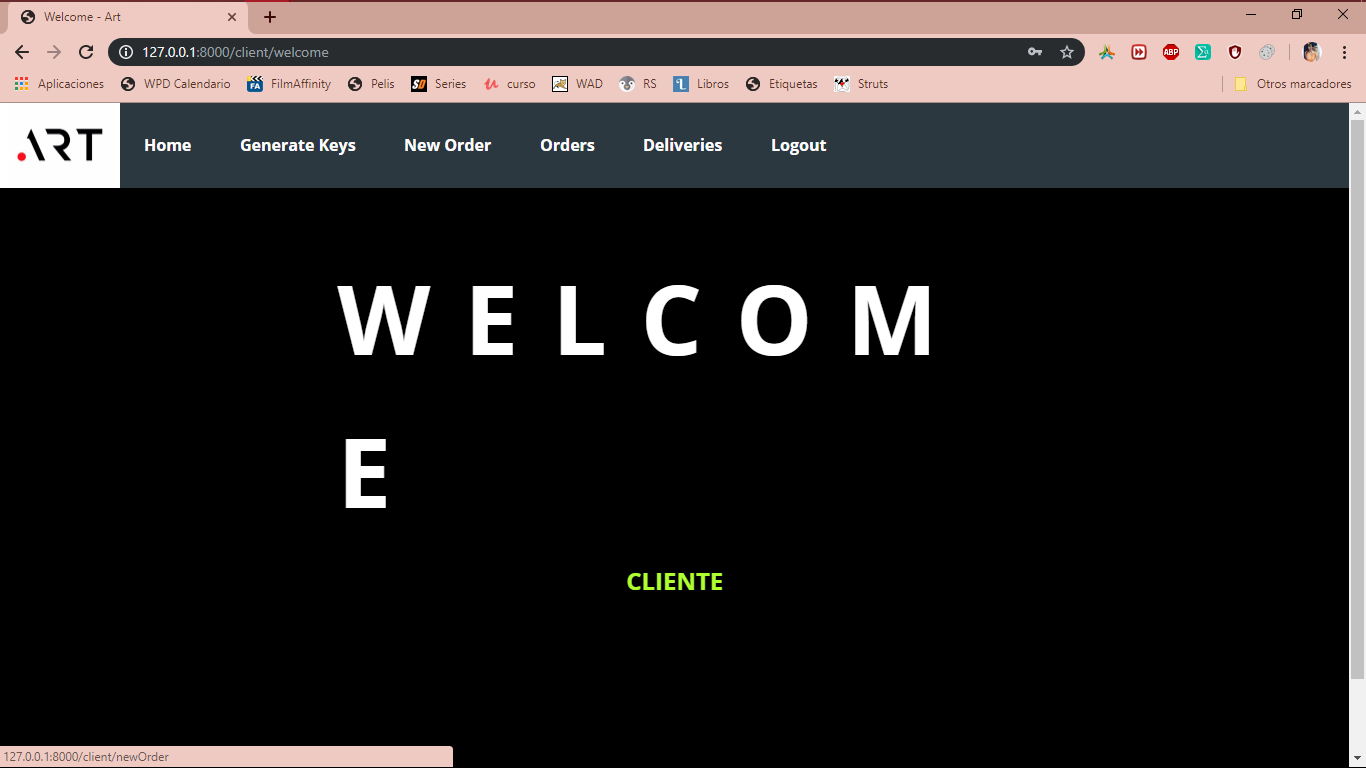
If you are already a registered customer, you can start from here. To this page can be reached from the *Sign in* button of the *Image 1. Main page.* menu, or from that same page the *Let's get started!* button. And from the *Image 1.1. Registration page.* in the *Sign in* link.

If you are not already registered and you are on this page, please click on the *Sign up* link and follow the steps above to register..

On this page you must enter the username and password with which you registered previously and then click on the *Sing in* button

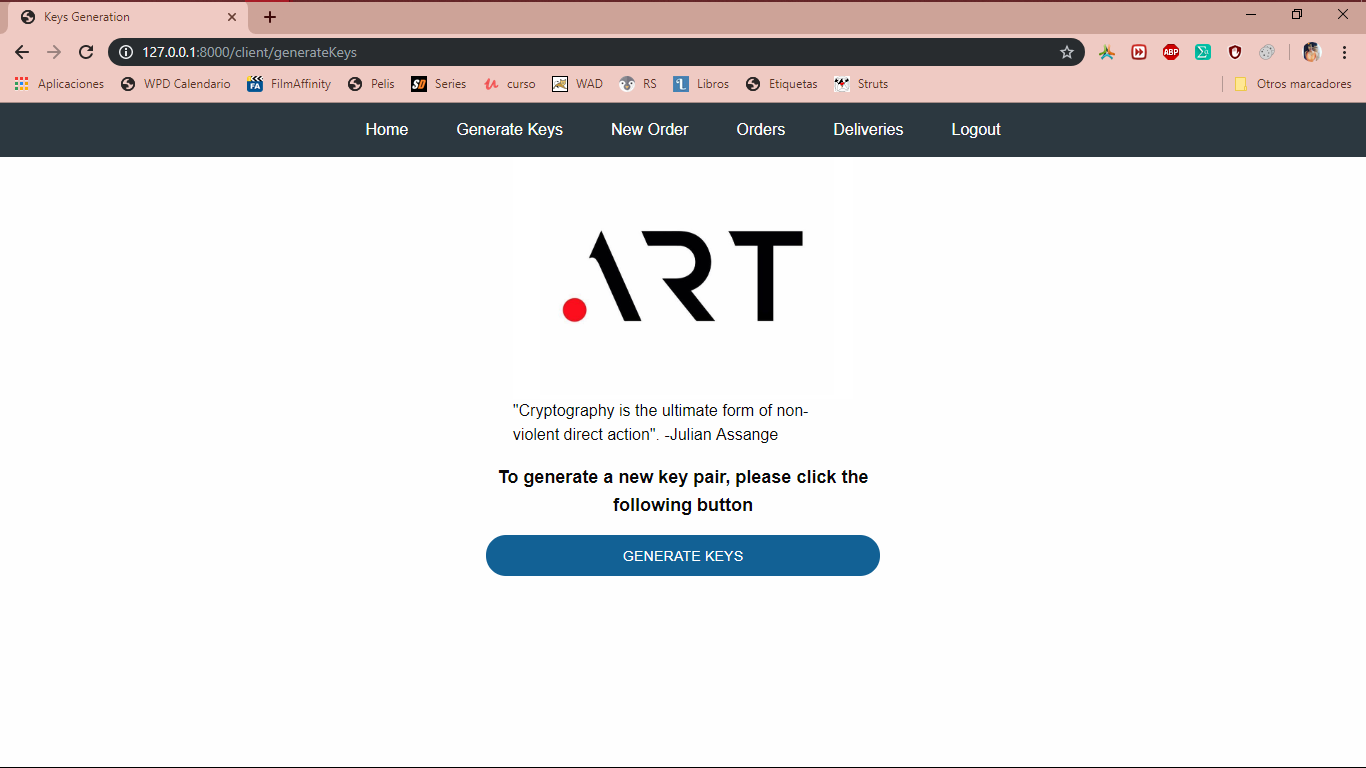


If you have already entered the system, this is the welcome page for customers, from it you can generate the keys that identify you as a customer, make an order to get a portrait, see the orders you have made and also see the deliveries what you have obtained.



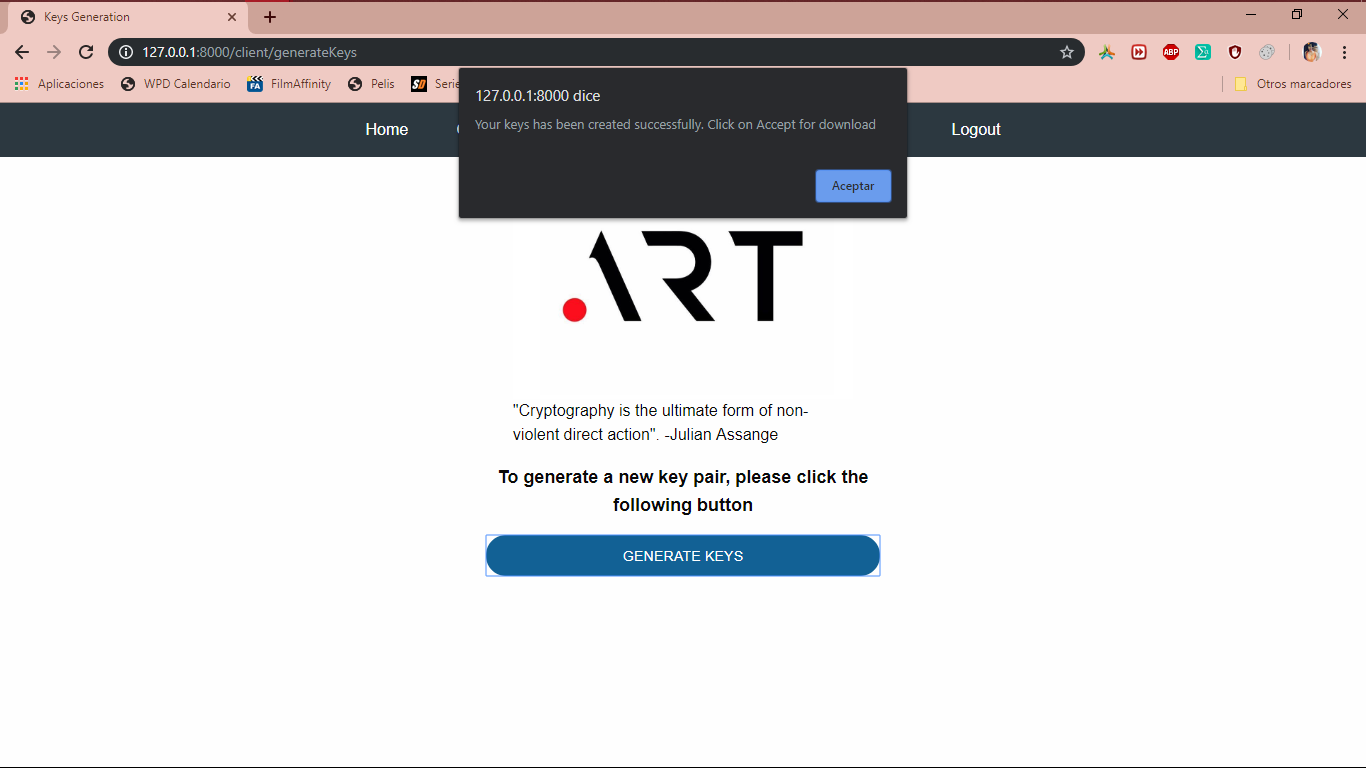
*Imagen 2. Client Welcome.*

The best thing is that after entering the system for the first time, go to the *Generate Keys* section of the menu, in order to generate the keys with which the images sent will be signed.



*Imagen 2.1. Generate Keys.*

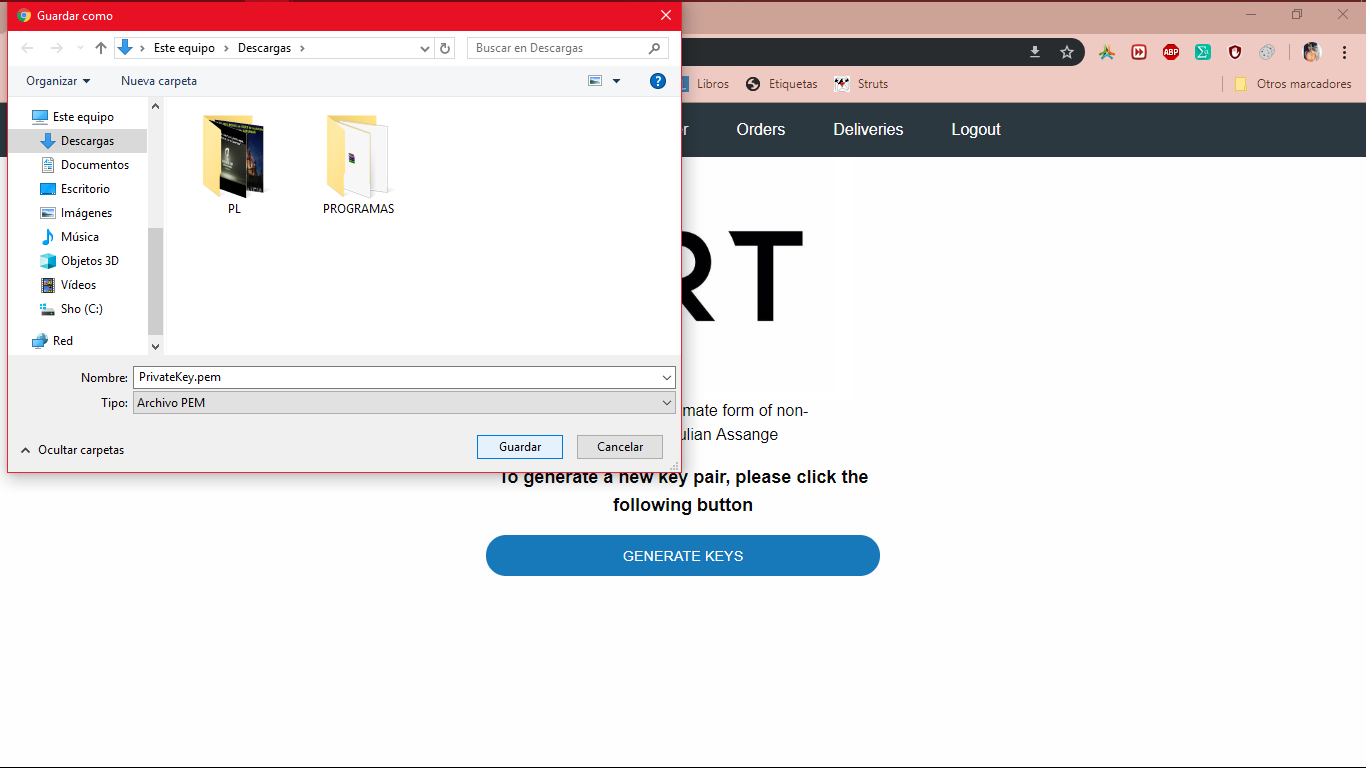
After clicking on the button *Generate Keys*, the following alert appears. Click *Aceptar.*



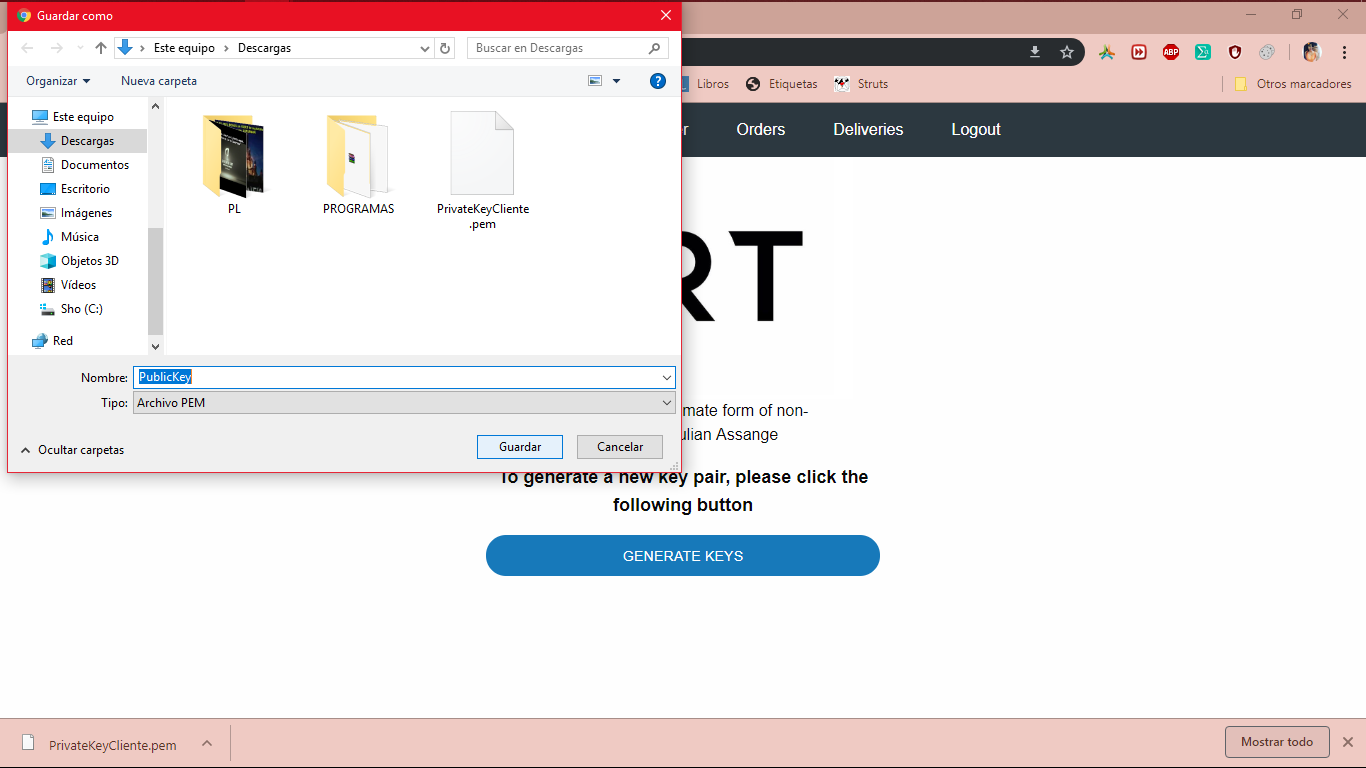
*Imagen 2.3. Generate Keys Alert.*

Then automatically download your keys, the public and private. At the same time the public key will be stored on the server, so it is important that you do not lose your downloaded keys. The private key will be stored inside a cookie.

In case of generating new keys, the orders that have been requested may no longer be seen.

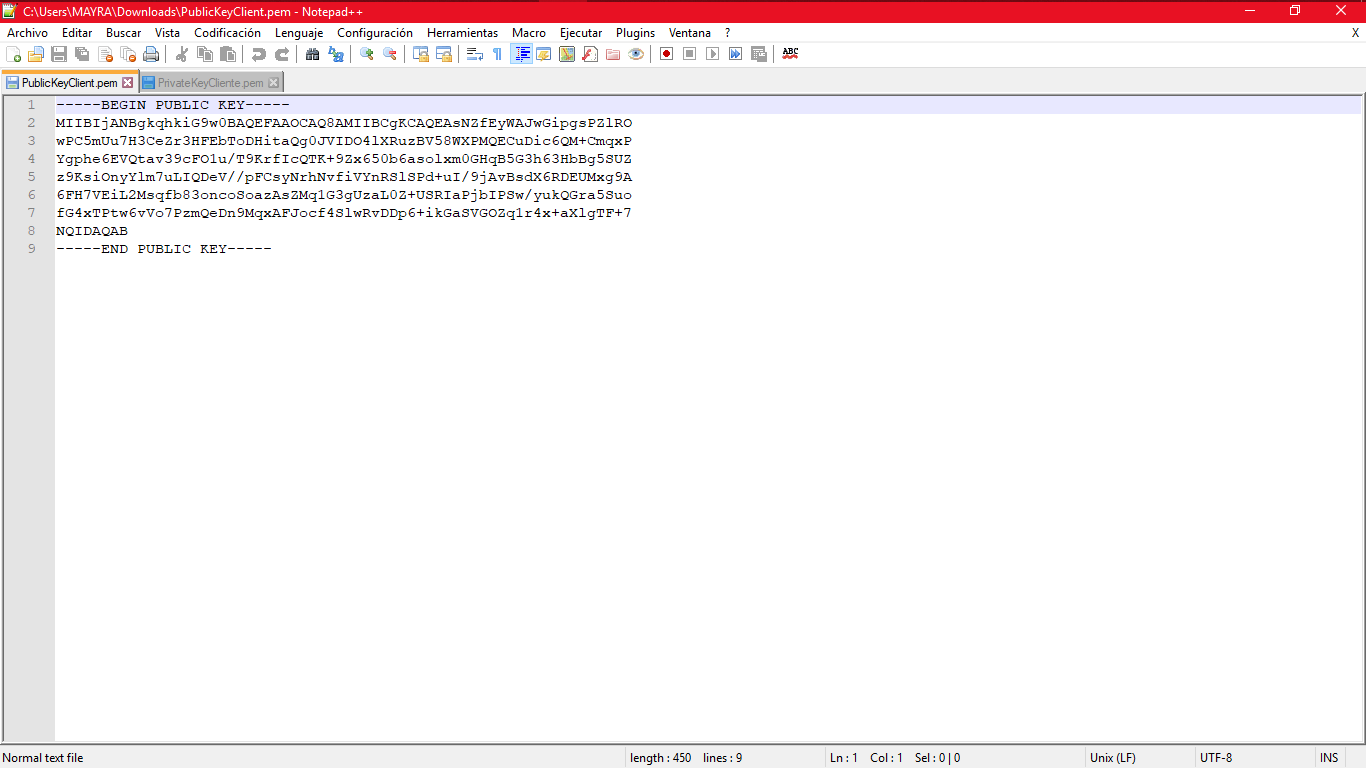


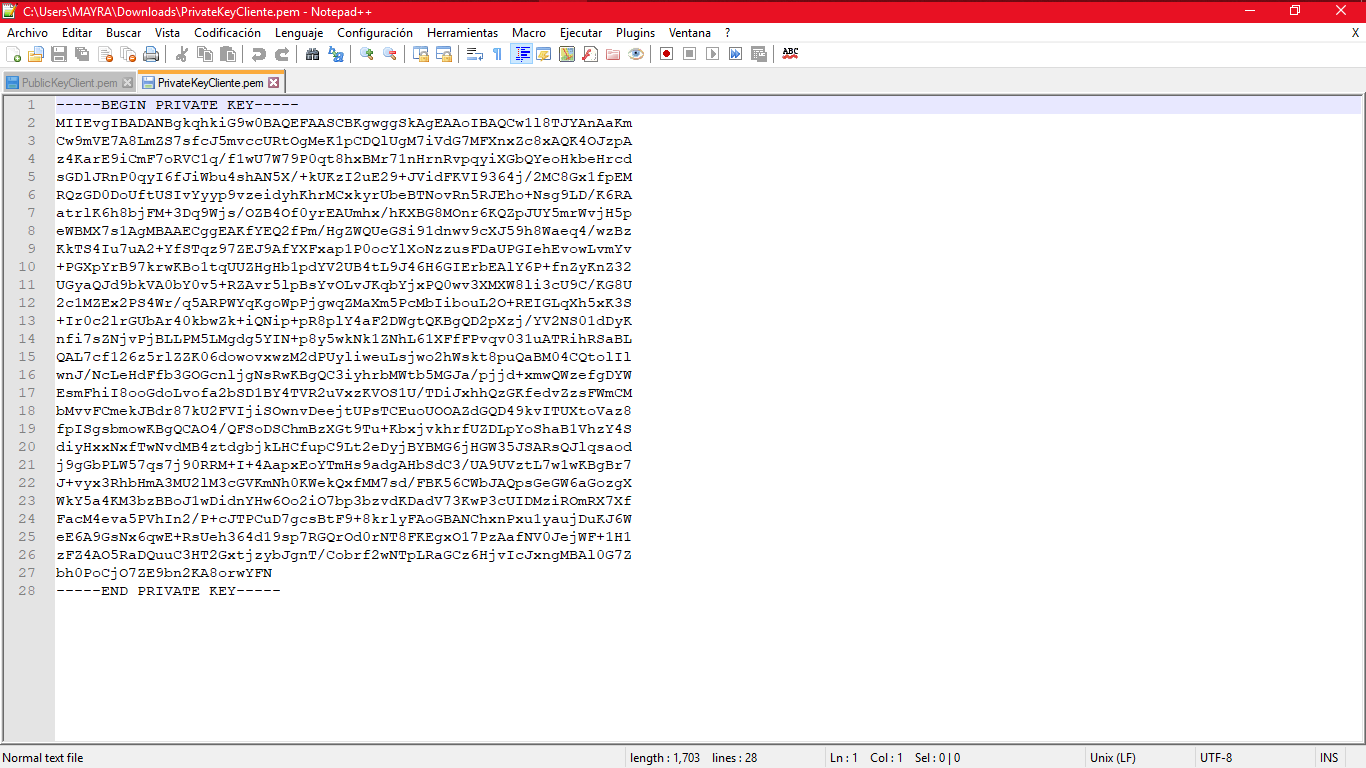
*Imagen 2.4. Download Private Key.*



*Image 2.5. Download Public Key.*

This is how you should see the keys you downloaded.





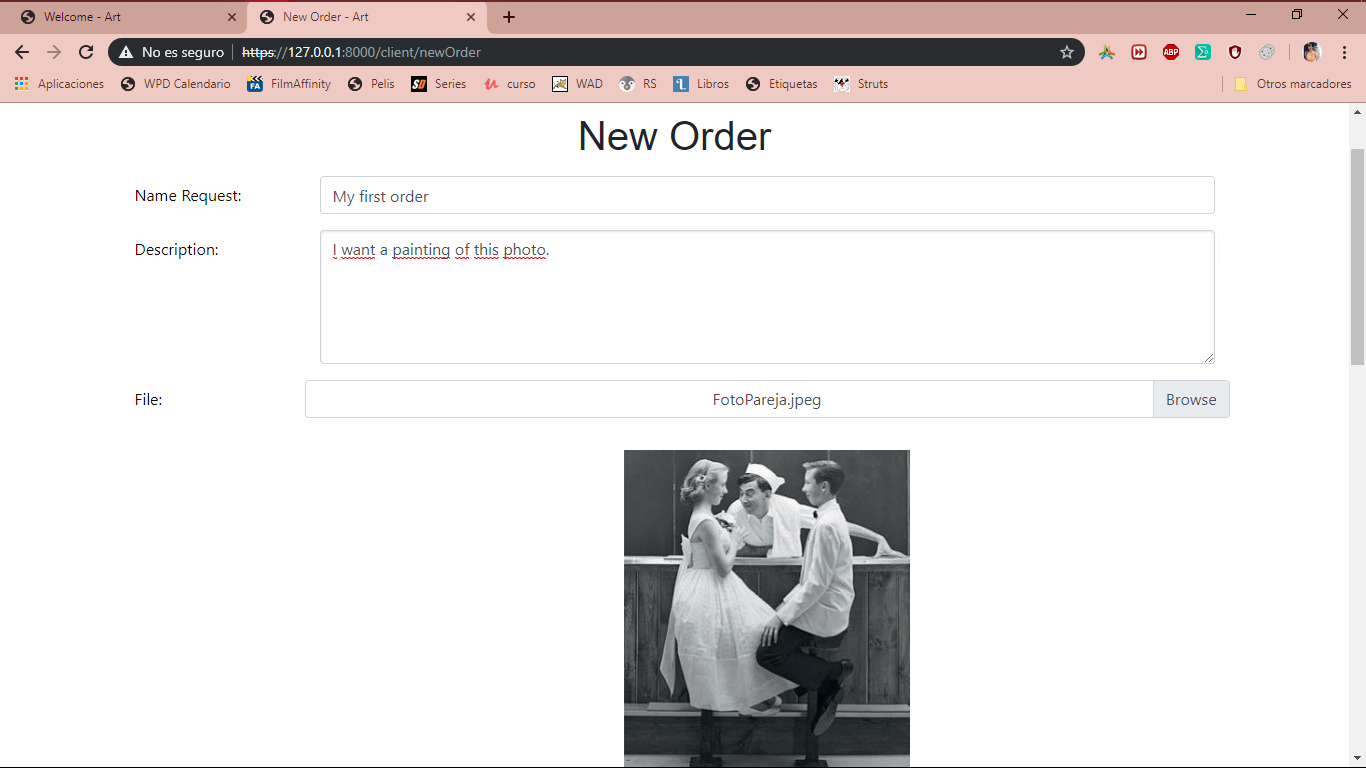
*Image 2.6. Public and private keys.*

After generating the keys, you can now make an order, click on the option *New order* of the menu, in this form you will have to give some information about the order such as the name, a description and the image you want to be modified.

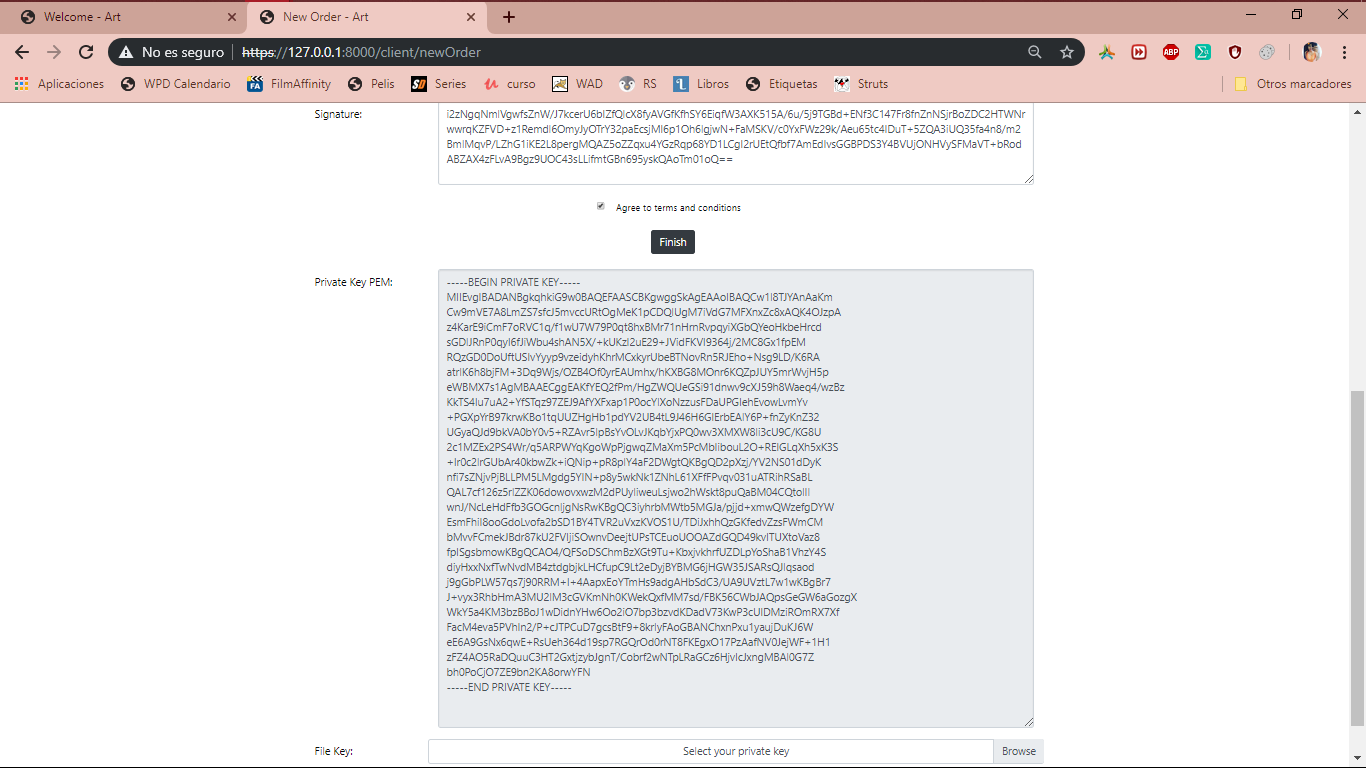
The private key with which it will be signed will already be loaded in the cookie so it will not be necessary to upload the file, but in case that for any reason there is no such cookie, you can upload the .pem file that was downloaded with your private key.

After the key and the image appear in the form, it will be signed, and the *Signature* field will be generated automatically.

To finish, the terms and conditions must be accepted to enable the Finish button.



*Image 2.7.1. New Order Form.*

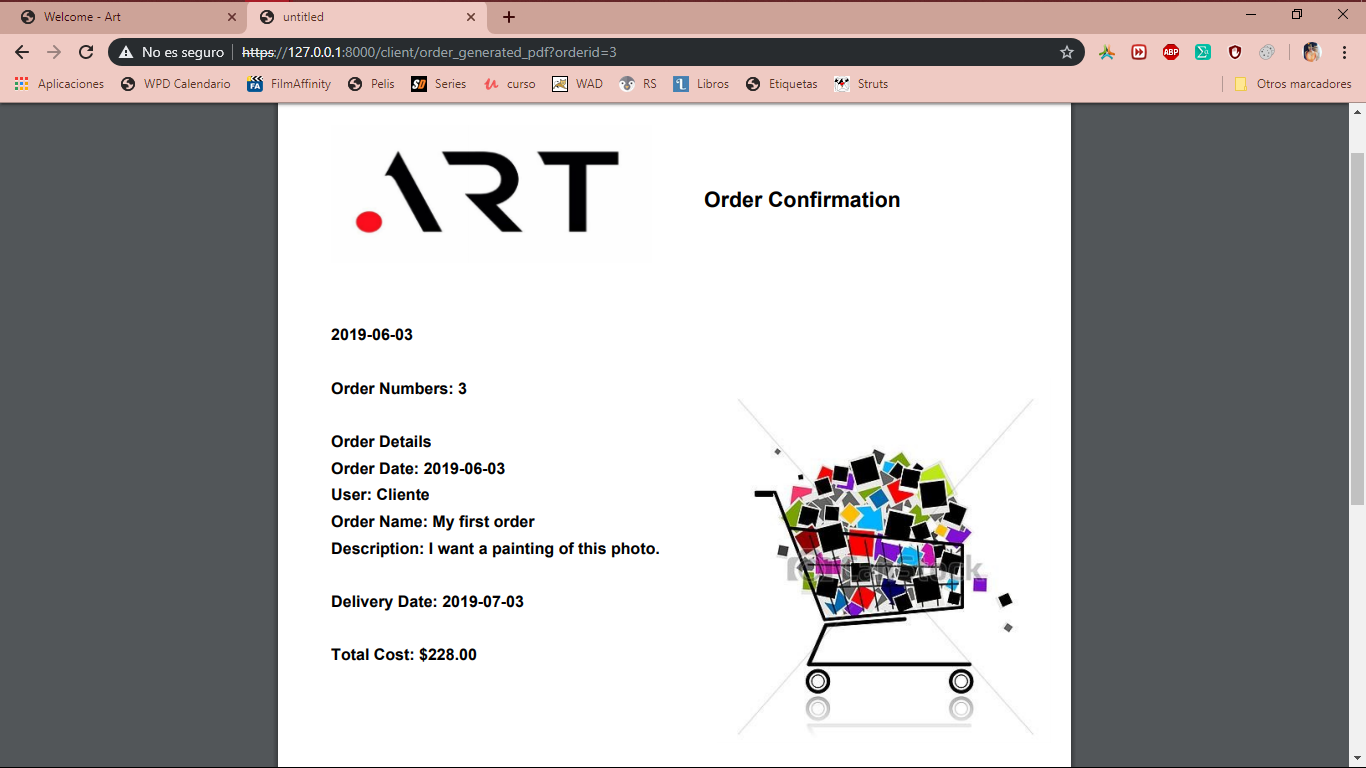


*Image 2.7.2. New Order Form.*

The order has already been created and will appear in the *Orders* section, with the date on which each order was made and the date of the delivery. On this page you can see a PDF with the confirmation of the order.

*mage 2.8. Orders of the client.*

Here you can see the PDF of the confirmation of the order, to see it you must click on the PDF icon belonging to each order that appears in *Image 2.8. Orders of the client.*

**

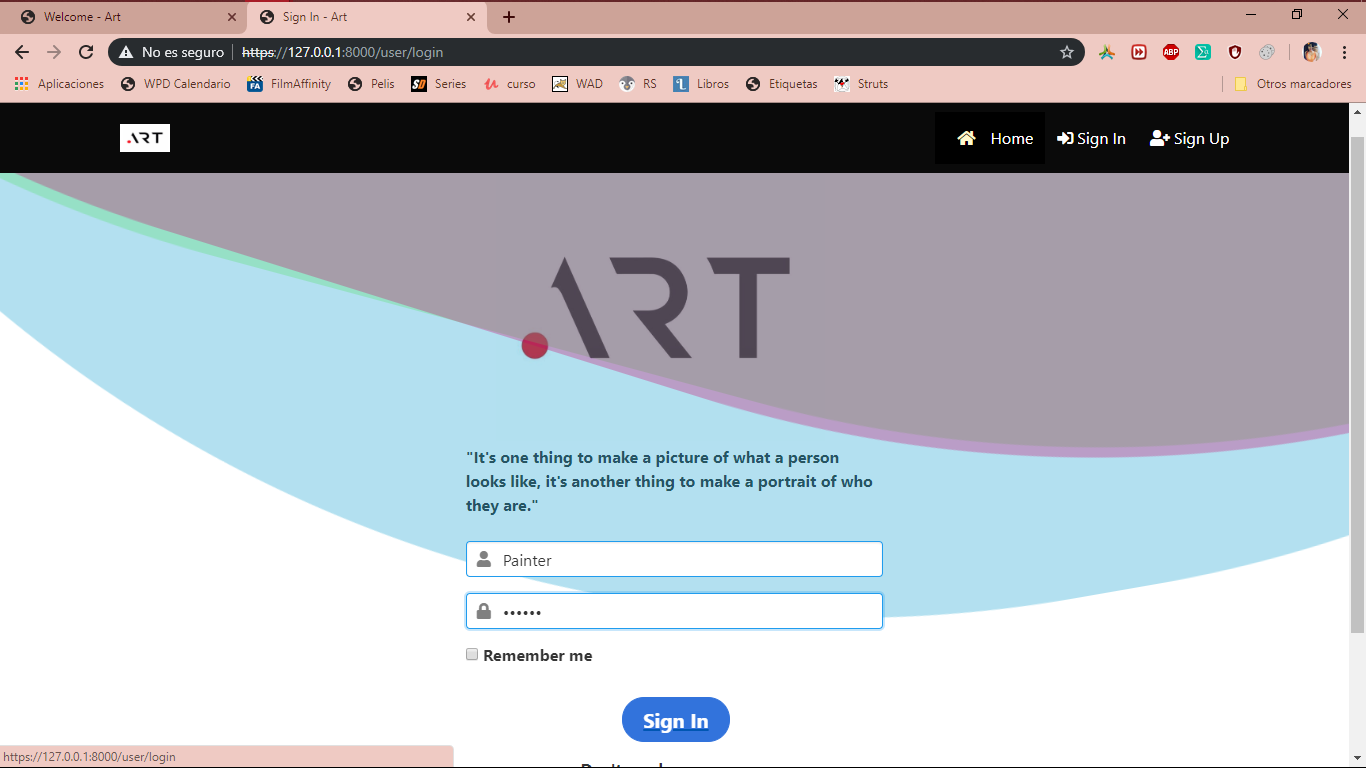
**If you are the painter:**

Now we will start session with the role of painter.

If you are already a registered painter, you can start from here. To this page can be reached from the *Sign in* button of the *Image 1. Main page.* menu, or from that same page the *Let's get started!* button. And from the *Image 1.1. Registration page.* in the *Sign in* link.

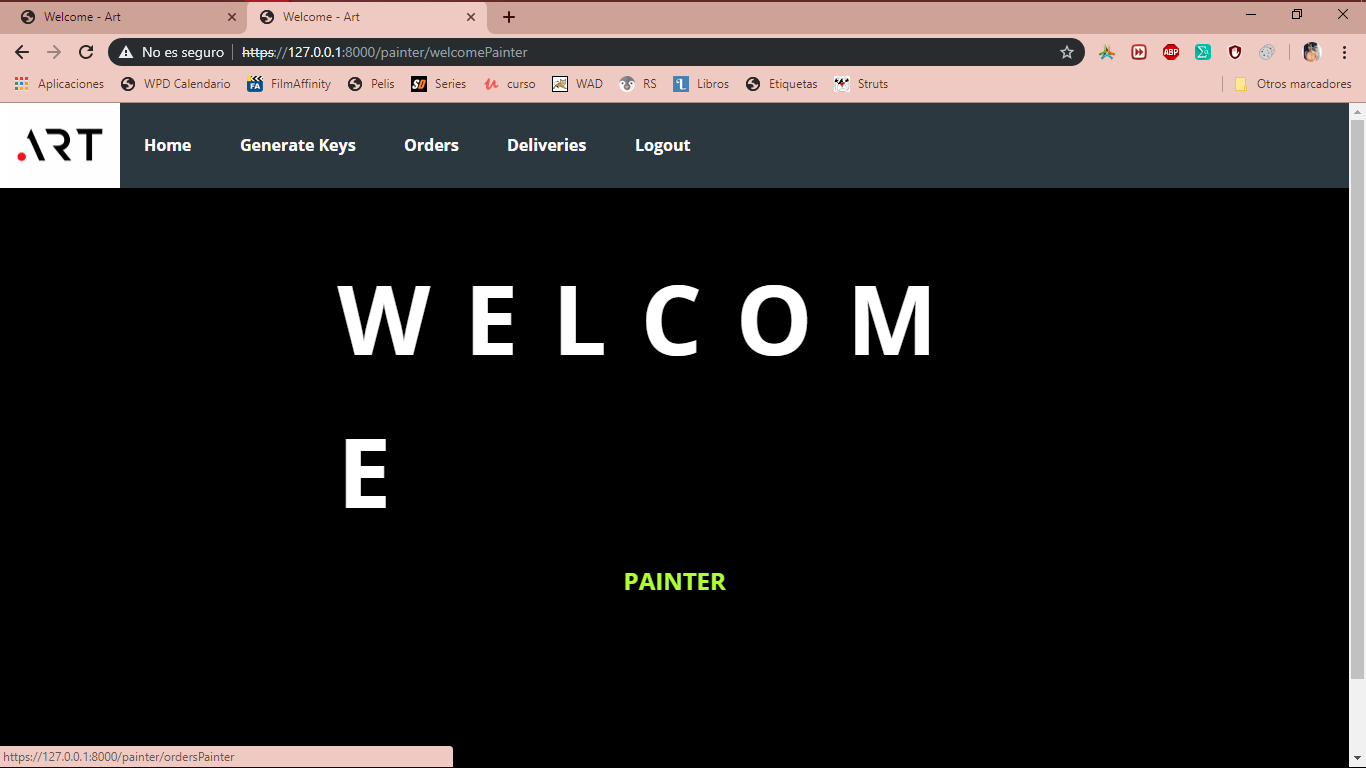
If you are not already registered and you are on this page, please click on the *Sign up* link and follow the steps above to register..

On this page you must enter the username and password with which you registered previously and then click on the *Sing in* button



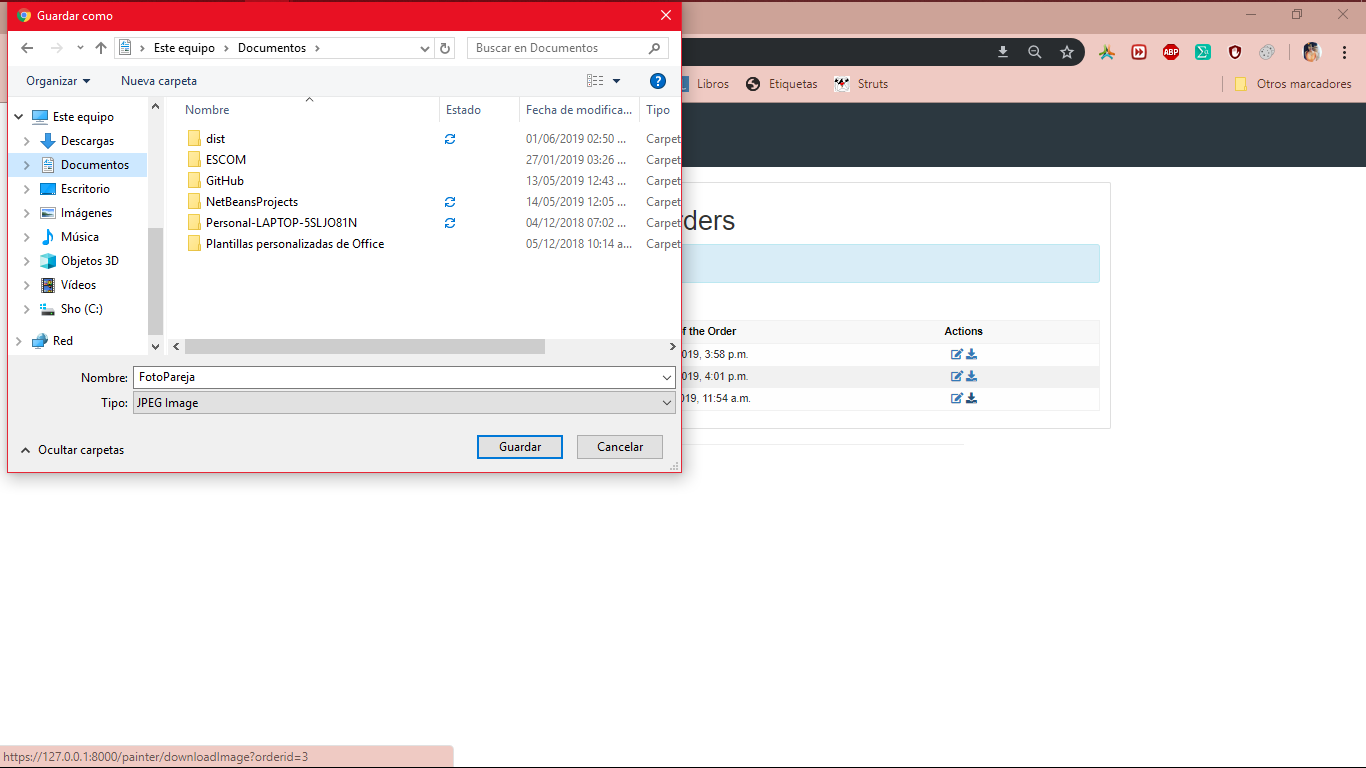
*Image 3.1. Log in Painter.*

If you have already entered the system, this is the welcome page for painter, from it you can generate the keys that identify you as a painter, view your customers orders, make portraits delivers and visualize those delivers.



*Image 3.2. Welcome Painter.*

In this image is shown how a the painter can download the client original photo from the view orders module, so it can get started to work.



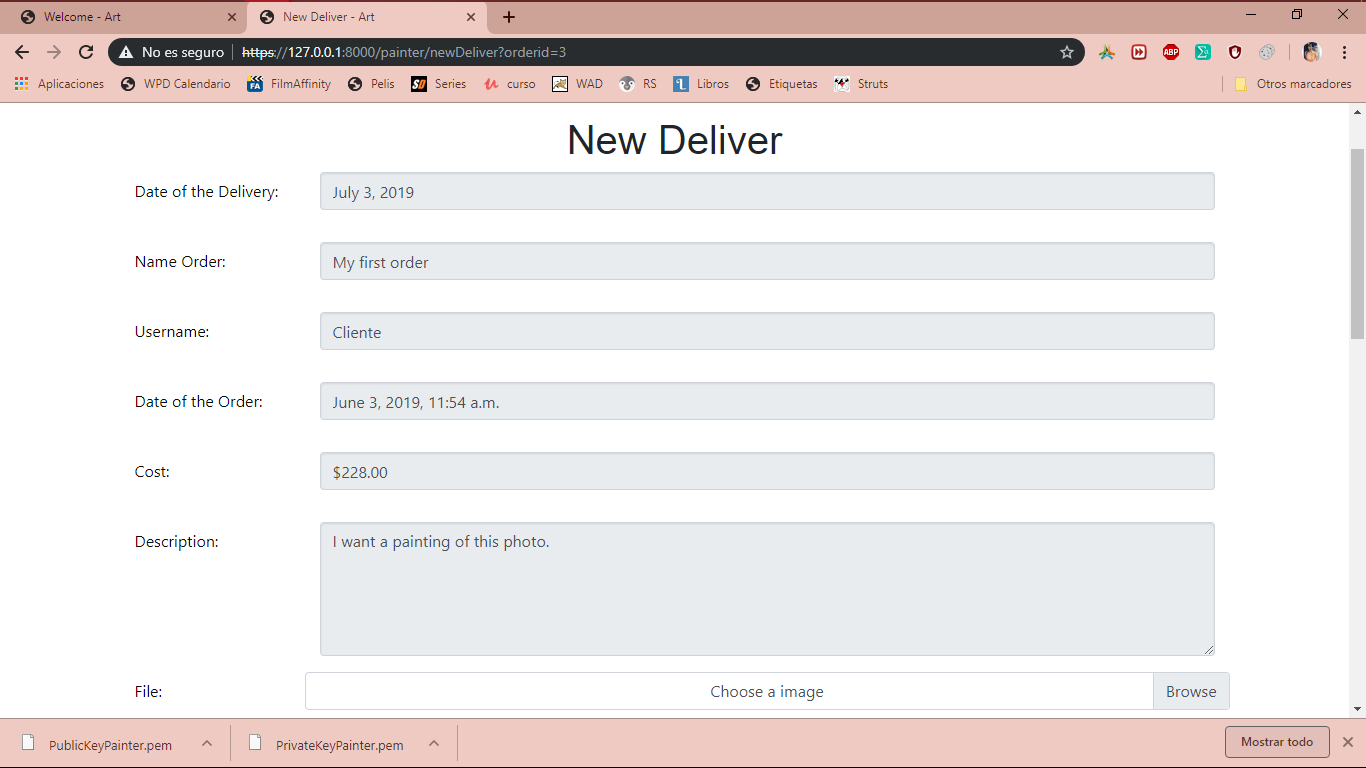
*Image 3.3. Download client photo.*

In this page you can see the information of the order and upload the image of deliver.

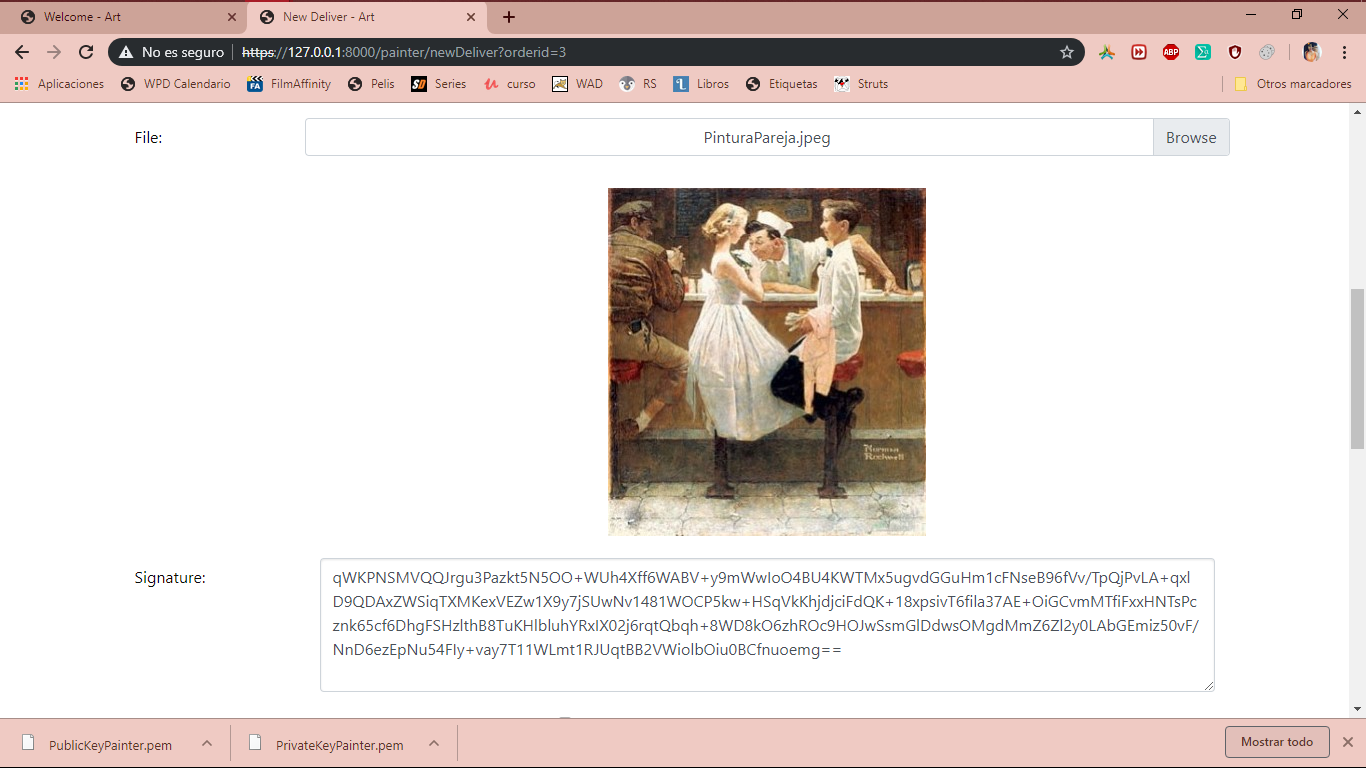
The private key with which it will be signed will already be loaded in the cookie so it will not be necessary to upload the file, but in case that for any reason there is no such cookie, you can upload the .pem file that was downloaded with your private key.

After the key and the image appear in the form, it will be signed, and the *Signature* field will be generated automatically.

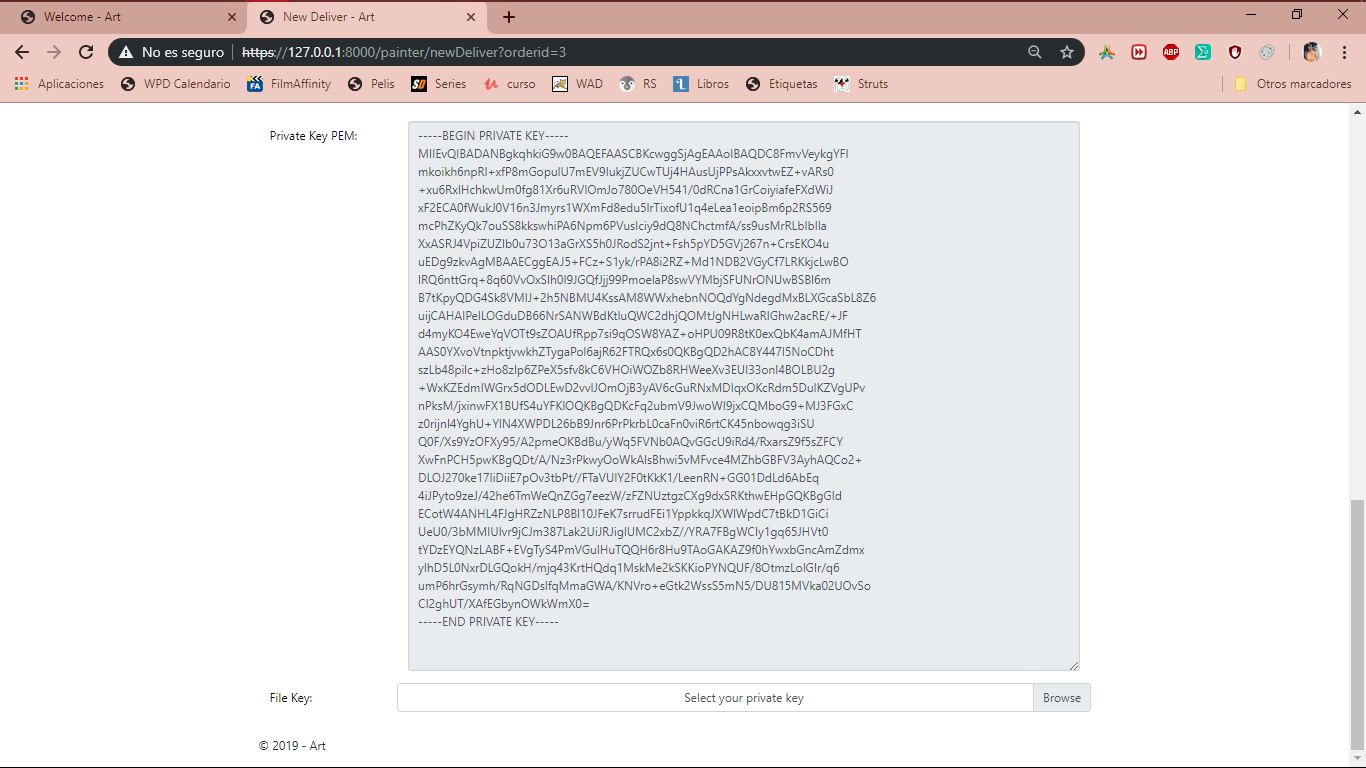
To finish, the terms and conditions must be accepted to enable the Finish button.



*Image 3.4.1. Order details.*

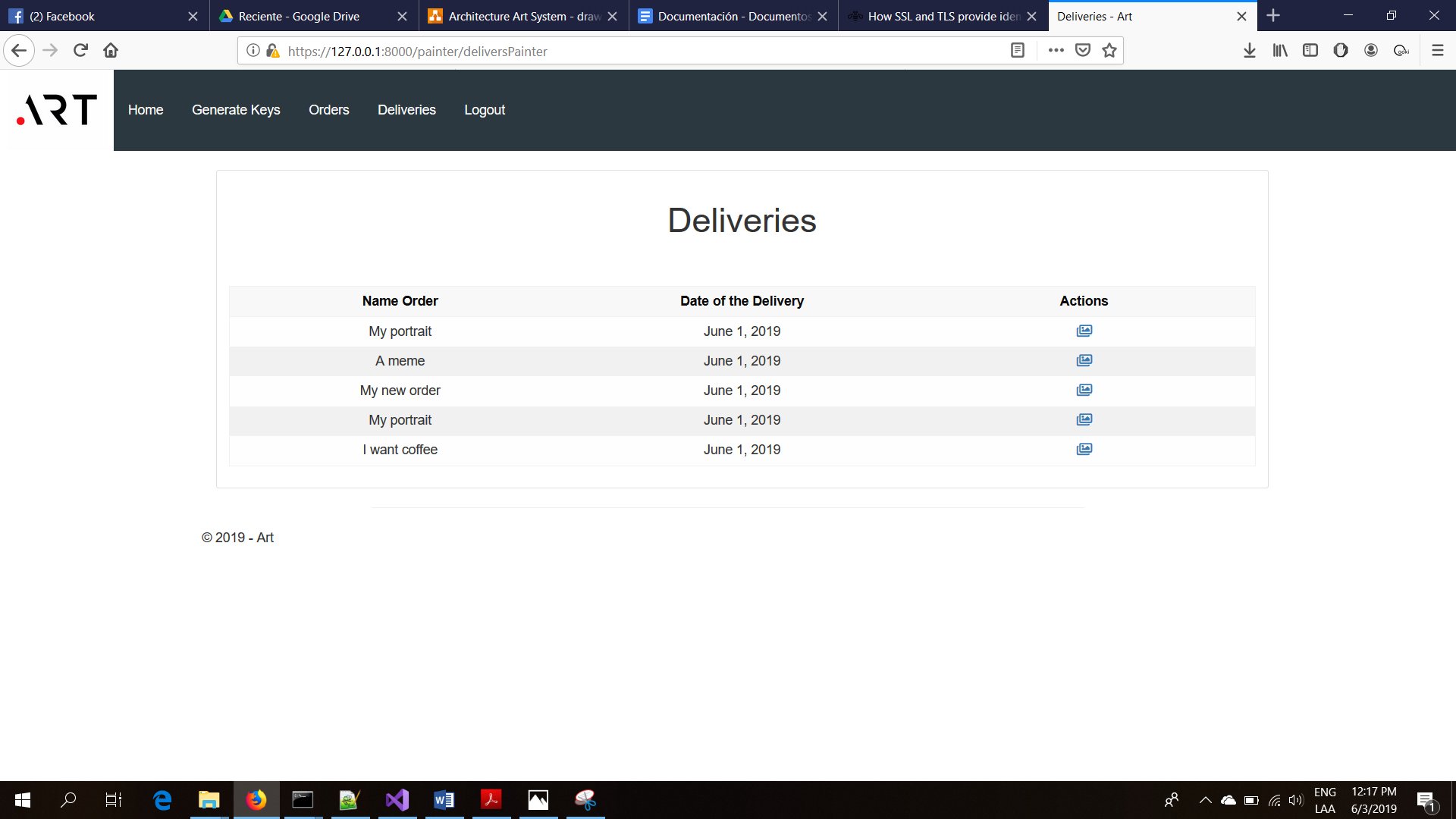


*Image 3.4.2. Order details. Signature.*



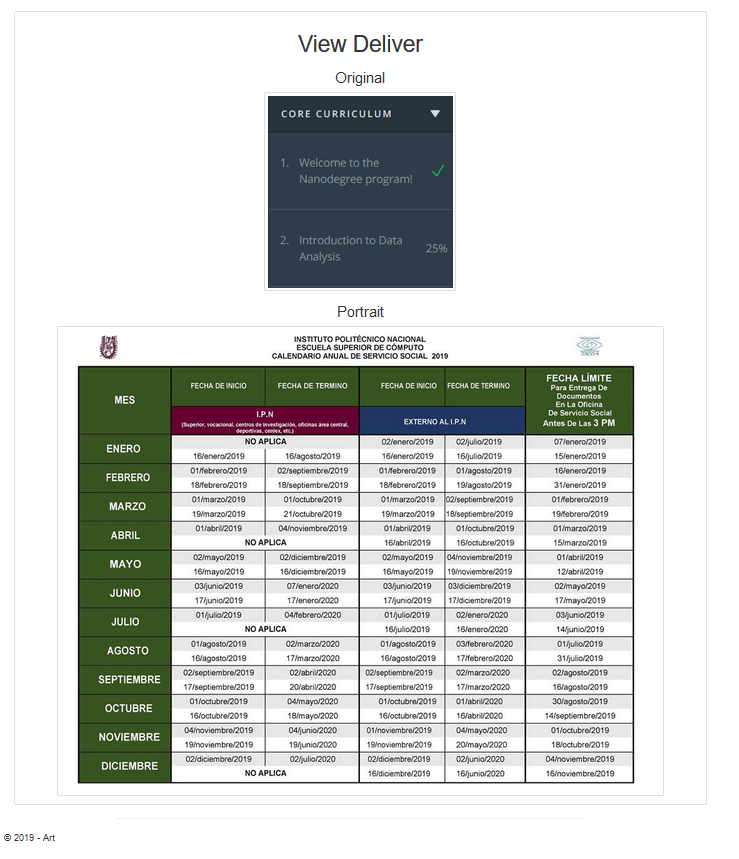
*Image 3.4.2. Order details.*

In this page the painter can view all the deliveries that he/she had made, the action available is visualize the deliver that has been made.



*Image 3.5. Deliveries of the Painter.*

Here we can see the comparison of the original image an image delivered by the painter.



*Image 3.6. View Delivers.*