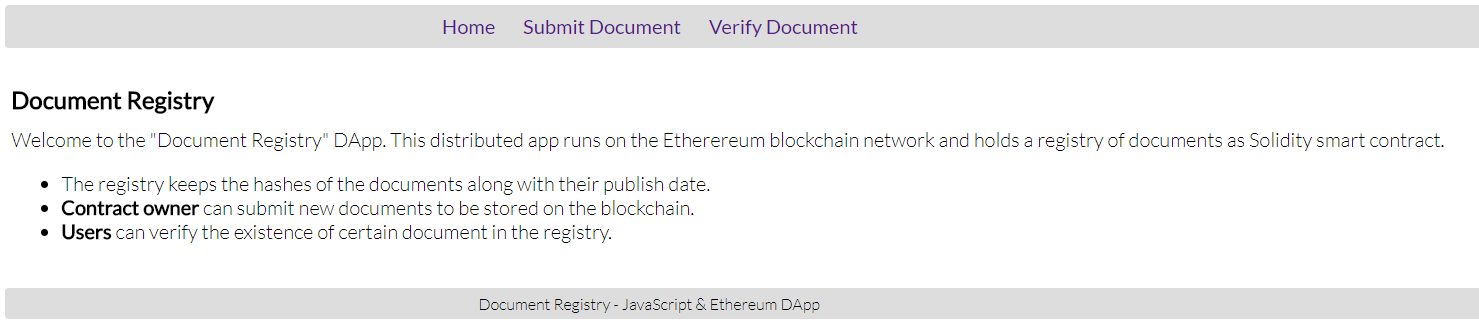
# Exercises: Create Document Registry

DocumentRegistry.sol Smart Contract code using **Remix IDE** and deployed it in a testing Ethereum blockchain network, created by using the **TestRPC** (it is a Node.js package). Connected from a JavaScript app to the smart contract using the **Web3 API**, provided by the **MetaMask** plugin for Chrome. The DApp users were able to check document validity against the blockchain registry (free for anyone). The smart contract owner will be able to upload new valid documents on the blockchain registry through (paid transaction, consuming some gas, allowed for the smart contract owner only).



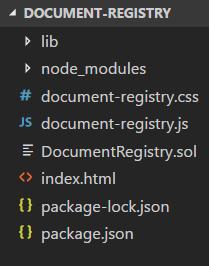
## Setup the project

1. Create the folder **Document-Registry** and run **npm init** command and it should create the **package.json** file then run the following command

|  |
| --- |
| npm install –g lite-server |

(node server, which will serve our web app) and create the following files **document-registry.css**, **document-registry.js** and **index.html**

1. Create folder **lib/** and import the two libraries from the resources given in the site.



1. Get the following css code and paste it to **document-registry.css**

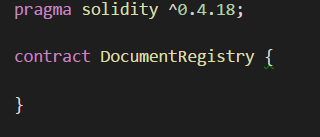
|  |
| --- |
| **document-registry.css** |
| **@import url**(**'https://fonts.googleapis.com/css?family=Lato:300,400'**);  **body** {  **font-family**: **Lato**;  **font-weight**: 300; }  **#menu** {  **font-weight**: 400;  **background**: **#DDD**;  **text-align**: **center**;  **padding**: 5**px**;  **line-height**: 1.5;  **border-radius**: 3**px**;  **overflow**: **auto**; }  **#menu a** {  **text-decoration**: **none**;  **padding**: 5**px** 10**px**;  **border-radius**: 5**px**; }  **#menu a**:**hover** {  **background**: **#BBB**; }  **main** > **section** {  **display**: **none**;  **padding**: 20**px** 5**px**; }  **section#viewHome** {  **display**: **block**; }  **section h1** {  **margin**: 10**px** 0**px**;  **font-size**: 1.2**em**; }  **#infoBox**, **#errorBox**, **#loadingBox** {  **width**: 80%;  **margin**: 10**px auto**;  **color**: **white**;  **text-align**: **center**;  **padding**: 5**px**;  **border-radius**: 3**px**; }  **#infoBox**>**header**, **#errorBox**>**header**, **#loadingBox**>**header** {  **float**: **right**;  **margin-right**: 5**px**; }  **#infoBox**>**header**:**hover**, **#errorBox**>**header**:**hover**, **#loadingBox**>**header**:**hover** {  **transition**: **all** 0.2**s**;  **font-weight**: **bold**;  **cursor**: **pointer**; }  **#loadingBox** {  **background**: **#7CB3E9**; } **#infoBox** {  **background**: **#393**; } **#errorBox** {  **background**: **#F50**; }  **footer** {  **background**: **#DDD**;  **padding**: 5**px** 10**px**;  **font-size**: 0.8**em**;  **text-align**: **center**;  **border-radius**: 3**px**; } |

## Create HTML

|  |
| --- |
| **index.html** |
| <!DOCTYPE **html**>  <**html**>  <**head**>  <**meta charset="utf-8"** />  <**title**>Document Registry DApp</**title**>  <**link rel="stylesheet" type="text/css" href="document-registry.css"** />  <**script src="lib/jquery-3.3.1.min.js"**></**script**>  <**script src="lib/sha256.min.js"**></**script**>  <**script src="document-registry.js"**></**script**> </**head**>  <**body**>  <**header id="menu"**>  <**a href="#" id="linkHome"**>Home</**a**>  <**a href="#" id="linkSubmitDocument"**>Submit Document</**a**>  <**a href="#" id="linkVerifyDocument"**>Verify Document</**a**>  </**header**>   <**main**>  <**section id="loadingBox"**>Loading ...</**section**>   <**section id="infoBox"**><**header**>x</**header**><**p**>Info</**p**></**section**>   <**section id="errorBox"**><**header**>x</**header**><**p**>Error</**p**></**section**>   <**section id="viewHome"**>  <**h1**>Document Registry</**h1**>  Welcome to the "Document Registry" DApp. This distributed app runs on the Etherereum blockchain network and holds a registry of documents as Solidity smart contract.  <**ul**>  <**li**>The registry keeps the hashes of the documents along with their publish date.</**li**>  <**li**><**b**>Contract owner</**b**> can submit new documents to be stored on the blockchain.</**li**>  <**li**><**b**>Users</**b**> can verify the existence of certain document in the registry.</**li**>  </**ul**>  </**section**>   <**section id="viewSubmitDocument"**>  <**h1**>Submit a Document</**h1**>  <**p**>Contract owners can register (upload) new documents to the "Document Registry" smart contract on the Ethereum blockchain decentralized network.</**p**>  <**input type="file" id="documentForUpload"** /><**br**/>  <**input type="button" id="documentUploadButton" value="Submit"** />  </**section**>   <**section id="viewVerifyDocument"**>  <**h1**>Verify a Document</**h1**>  <**p**>Blockchain users can verify documents by checking whether they exist in the "Document Registry" smart contract on the Ethereum blockchain decentralized network.</**p**>  <**input type="file" id="documentToVerify"** /><**br**/>  <**input type="button" id="documentVerifyButton" value="Submit"** />  </**section**>  </**main**>    <**footer**>Document Registry - JavaScript & Ethereum DApp</**footer**> </**body**> </**html**> |

## Create Smart Contract

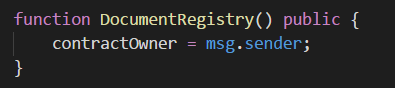
1. Create initial **contract** and define the version:



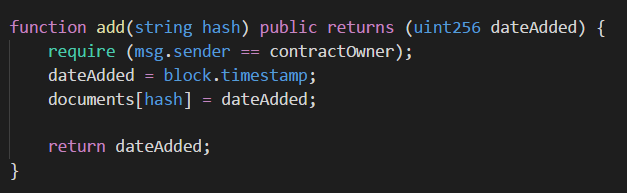
1. Create **mapping** to store the string hash of the document and the date it has been added and address which will store the owner’s address:



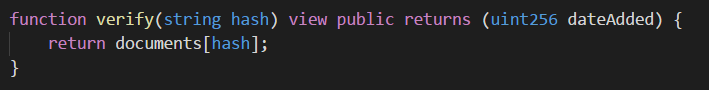
1. Create the **constructor,** which will make the executor owner of the contract:



1. Create an add function, which adds a document by giving its hash and it will store the hash and the current time (**block.timestamp**) in the mapping. The function returns the date:

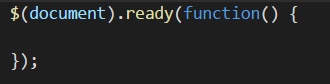


1. Finally, create **verify** function, which will get the hash of the document and will return the timestamp if there is a document in the mapping



## Create JavaScript Front-end

1. Create the jQuery function which will run as soon as the pages DOM become safe to manipulate:



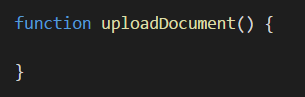
1. Create the constant contract address and contract **ABI**, which will get after deploying our contract after a while:



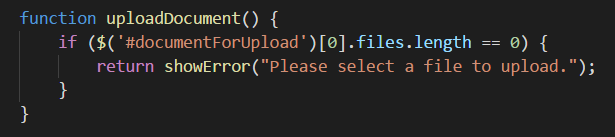
1. Insert the code below for the buttons to the functions and visibility:

|  |
| --- |
| **document-registry.js** |
| $(**'#linkHome'**).click(**function** () {  *showView*(**"viewHome"**) }); $(**'#linkSubmitDocument'**).click(**function** () {  *showView*(**"viewSubmitDocument"**) }); $(**'#linkVerifyDocument'**).click(**function** () {  *showView*(**"viewVerifyDocument"**) }); $(**'#documentUploadButton'**).click(*uploadDocument*); $(**'#documentVerifyButton'**).click(*verifyDocument*);  *// Attach AJAX "loading" event listener* $(**document**).on({  ajaxStart: **function** () {  $(**"#loadingBox"**).show()  },  ajaxStop: **function** () {  $(**"#loadingBox"**).hide()  } });  **function** *showView*(viewName) {  *// Hide all views and show the selected view only* $(**'main > section'**).hide();  $(**'#'** + viewName).show(); }  **function** *showInfo*(message) {  $(**'#infoBox>p'**).html(message);  $(**'#infoBox'**).show();  $(**'#infoBox>header'**).click(**function** () {  $(**'#infoBox'**).hide();  }); }  **function** *showError*(errorMsg) {  $(**'#errorBox>p'**).html(**"Error: "** + errorMsg);  $(**'#errorBox'**).show();  $(**'#errorBox>header'**).click(**function** () {  $(**'#errorBox'**).hide();  }); } |

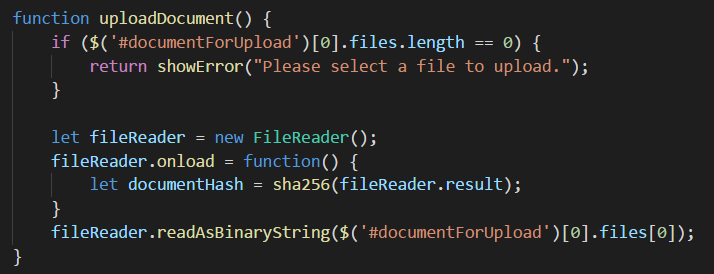
1. Create a function for **uploading** the document:



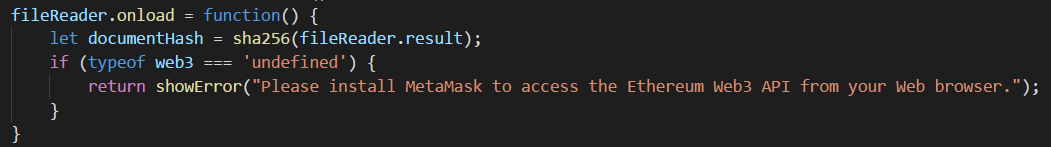
1. Check if there is any file to **upload**, if not, show **error message**:



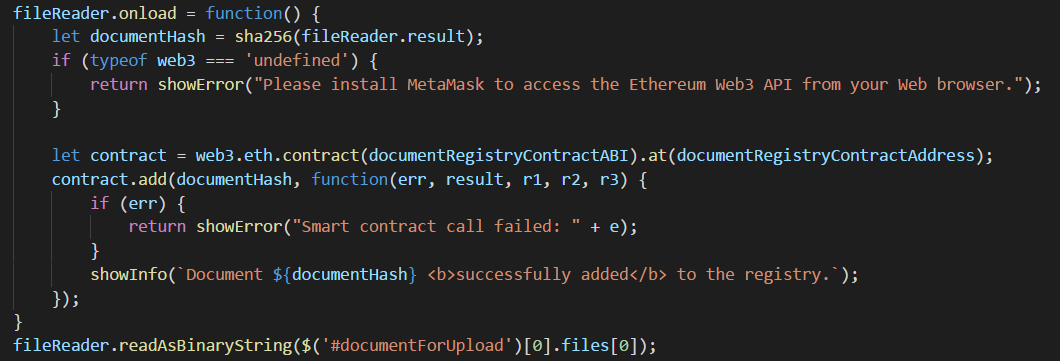
1. Create **file** **reader** to hash the document:



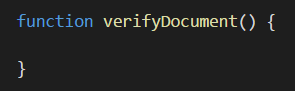
1. **Save** the document hash in variable and check if the **web3** library is defined. You should have **MetaMask** in your browser:



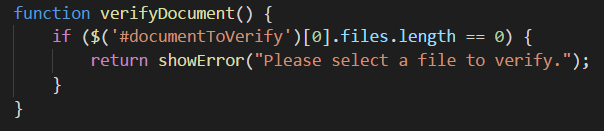
1. **Define** the contract by its **ABI** and address, and **call** the **add** function with the parameters needed:



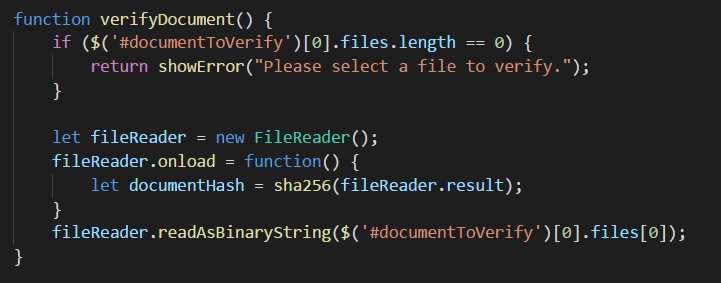
1. Function for **verifying** the document:



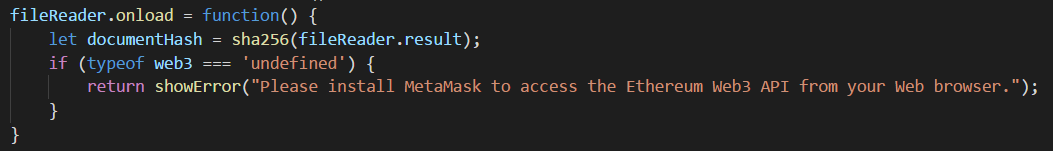
1. **Check** if there is any file to **verify**, if not, show **error message**:



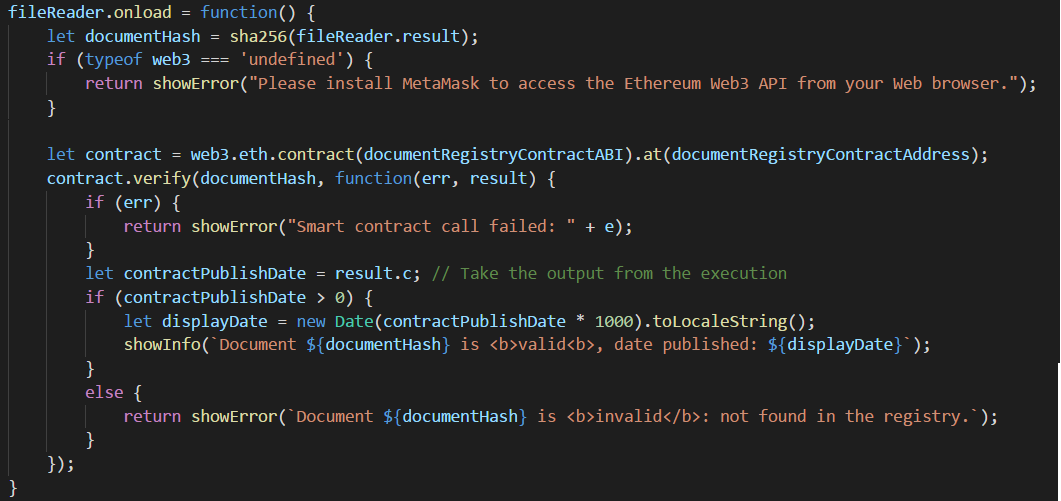
1. **Create** file reader to hash the document:



1. **Save** the document hash in variable and **check** if the **web3** library is defined. You should have **Metamask** at your browser



1. **Define** the contract by its **ABI** and address and **call** the **verify** function with the parameters needed



## Play with the Document Registry

1. Install **testrpc** by the command:

|  |
| --- |
| npm install –g testrpc |

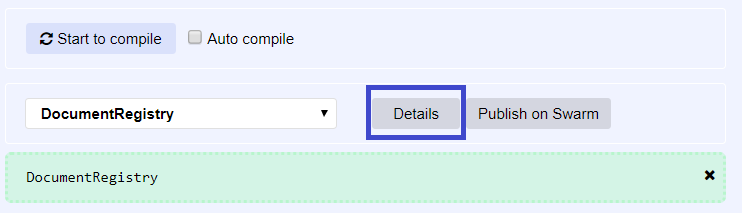
1. Start **testrpc** by typing on the command line:

|  |
| --- |
| testrpc |

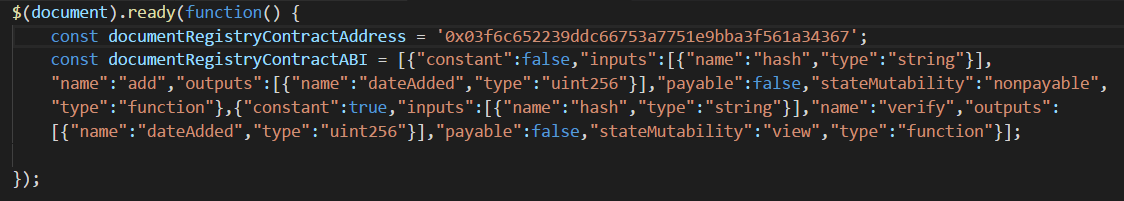
1. Go to **remix.ethereum.org** and paste the smart contract code.
2. Compile and choose the **Web3 Provider** to connect to **testrpc**.
3. **Create** the contract.
4. You can find the contract address after **deploying** the contract from remix here at **run** tab:



1. The ABI is shown after clicking **[Details]**:



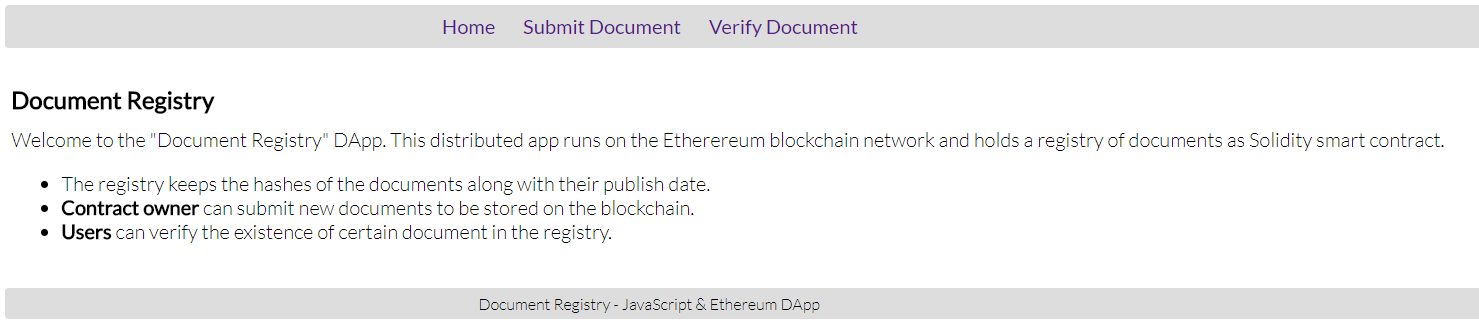
1. Go back to the project and paste the **contract address** and the **ABI** for the variables we previously wrote:



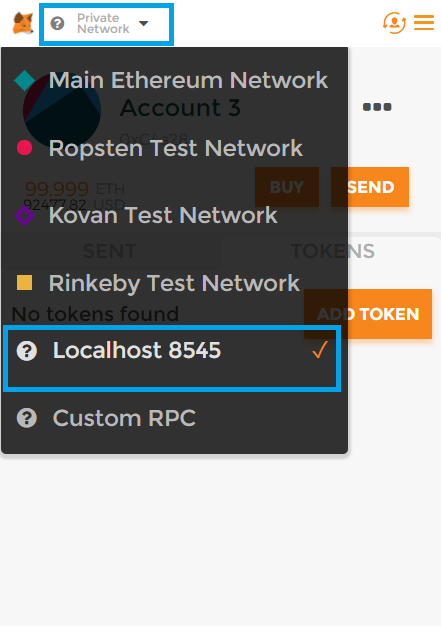
1. Start the Dapp by the command:

|  |
| --- |
| lite-server |

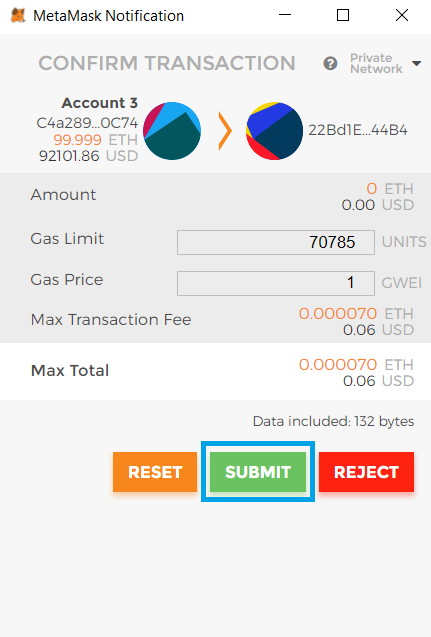
and play with it:



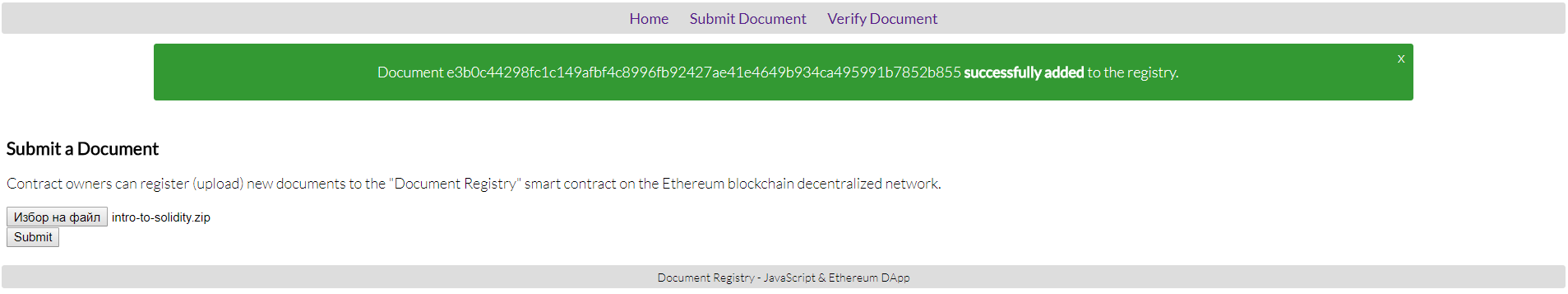
1. Do not forget to log in **MetaMask** and switch to Private Network:



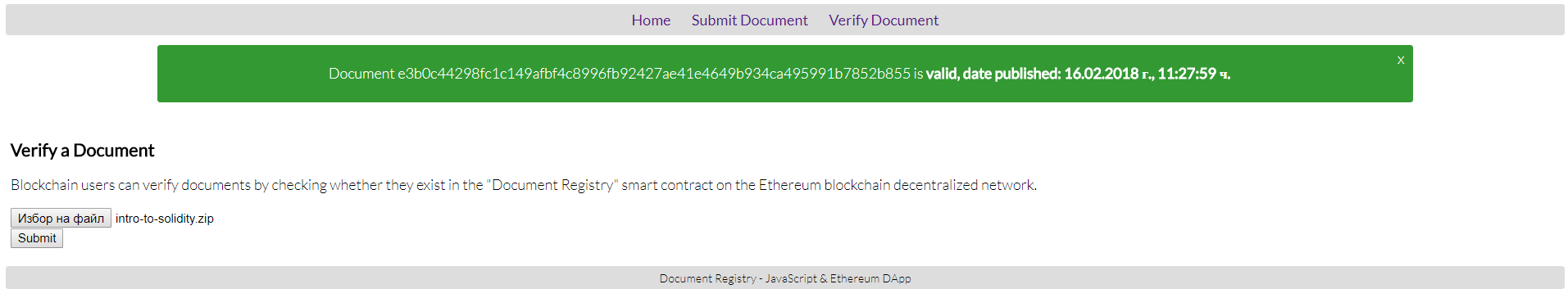
1. Import at least one of the private keys from **testrpc.**
2. Upload any kind of a file in the **[Submit Document]** section. Click **[Submit].** MetaMask will appear, Click **[Submit]**:



1. A message will appear:



1. Test **[Verify Document]** with the same document:



# What to Submit?

Submit as exercise outcome a **zip file** (e.g. your-username-document-registry.zip)withthe source code and smart contract, and at least two screenshots with a successful submission of a document and a successful verification for the same document.