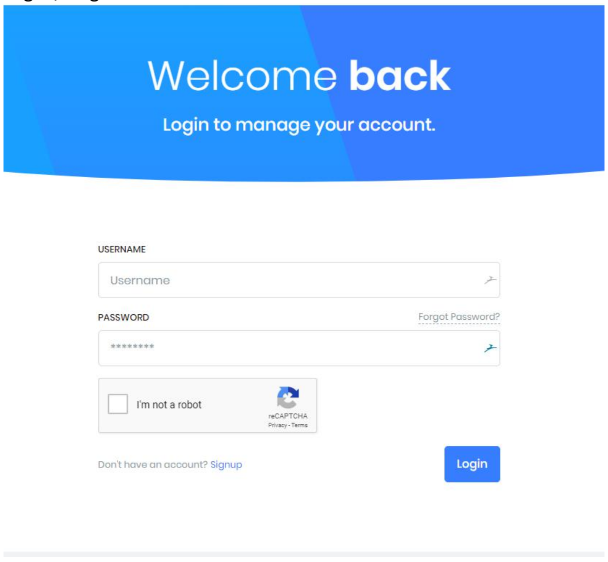
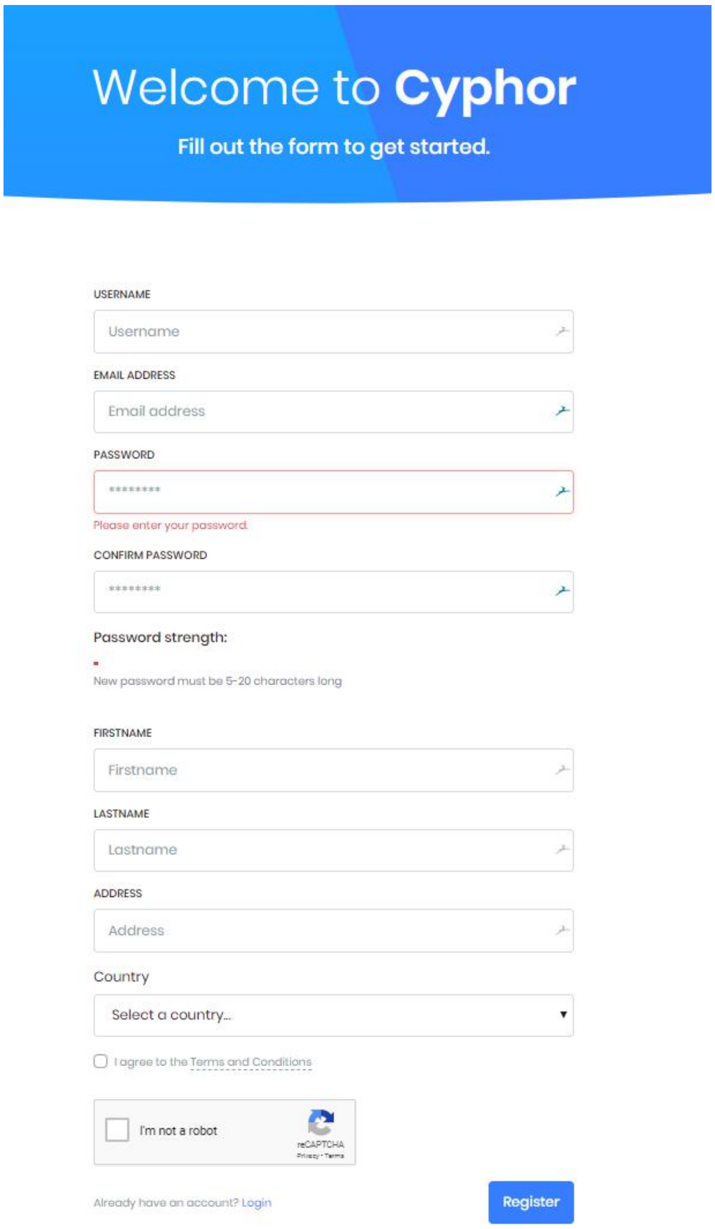
Cyphor Technologies Website.

Design:

The design is what we will use to attract the customer. We will use light blue as the main color scheme.
As of now, we have worked with an existing HTML template, we plan on keeping it that way unless the developer has a better alternative. Proof of concepts of design we like for the project:
https://sb-admin-pro-angular.startbootstrap.com/dashboard
https://preview.themeforest.net/item/endless-angular-admin-template/full_screen_preview/23884779?_ga=2.26330365.1527452838.1579562919-2143150482.1561189069

Below we have listed all the pages we require and their functionality, the design, however, should be applied differently using our new bootstrap template.
Login / Register:





Welcome to **Cyphor**

Fill out the form to get started.

USERNAME

Username

EMAIL ADDRESS

Email address

PASSWORD

****

Please enter your password.

CONFIRM PASSWORD

****

Password strength:

New password must be 5-20 characters long

FIRSTNAME

Firstname

LASTNAME

Lastname

ADDRESS

Address

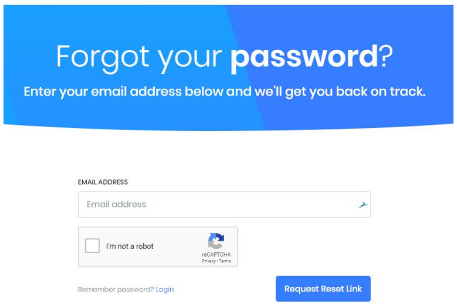
Country

Select a country...

☐ I agree to the Terms and Conditions

Already have an account? Login

Forgot password:



Forgot your **password**?

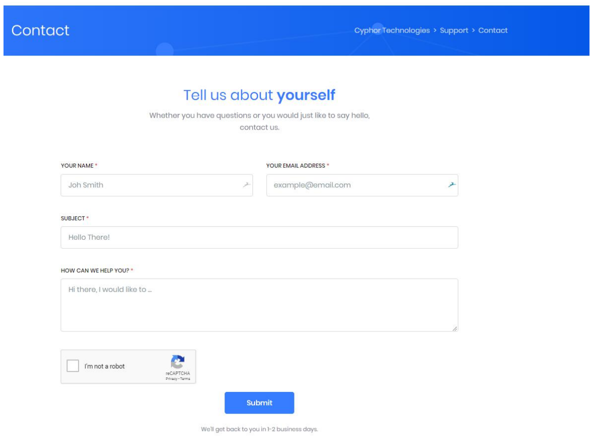
Enter your email address below and we'll get you back on track.

EMAIL ADDRESS

Email address

Remember password? Login

Contact us page:



Contact

Cyphor Technologies > Support > Contact

Tell us about **yourself**

Whether you have questions or you would just like to say hello, contact us.

YOUR NAME *

Joh Smith

YOUR EMAIL ADDRESS *

example@email.com

SUBJECT *

Hello There!

HOW CAN WE HELP YOU? *

Hi there, I would like to ...

I'm not a robot

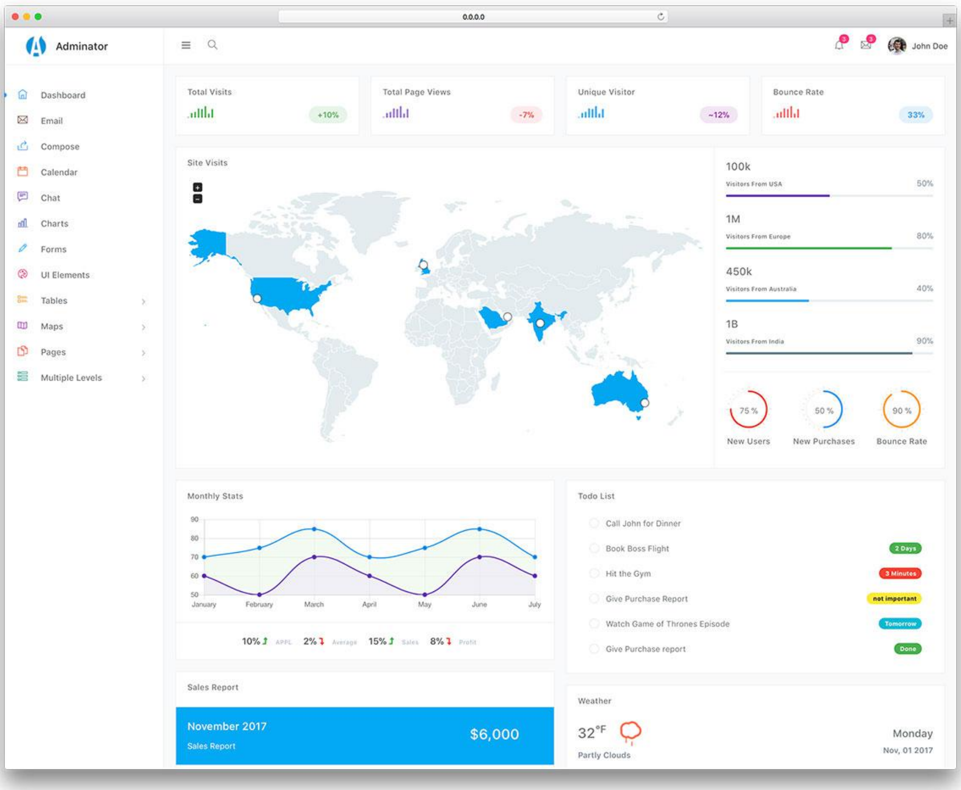
reCAPTCHA

Privacy - Terms

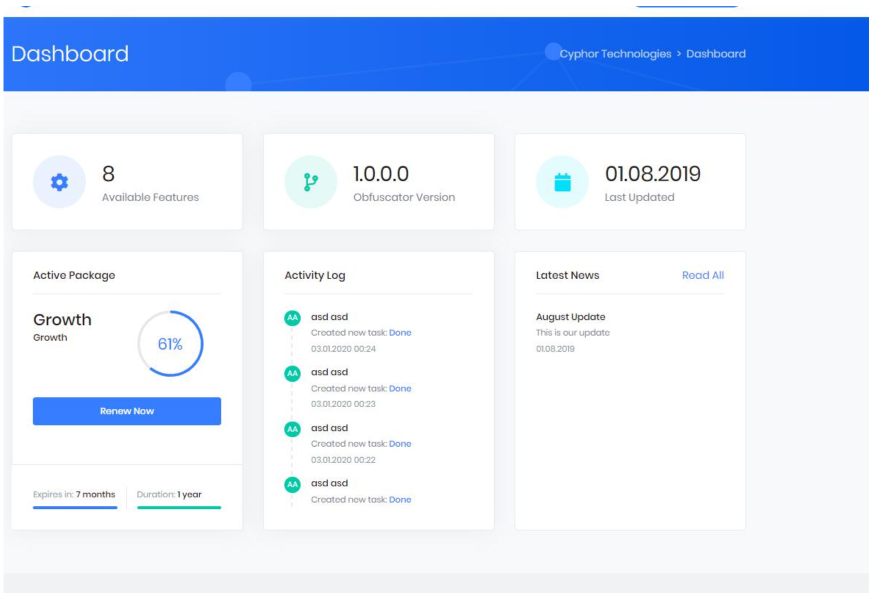
Submit

We'll get back to you in 1-2 business days.

Dashboard: (Proof of concept).
Our dashboard would instead display the following information:
- Obfuscator version.
- Tickets waiting reply.
- Total obfuscation methods.
- Active package/features (if any).
- Change-log of the service.
- Resume of usage history.



The dashboard we currently use displays more or less the same information we want to show with the new design:



Dashboard

Cyphor Technologies > Dashboard

8
Available Features

1.0.0.0
Obfuscator Version

01.08.2019
Last Updated

Active Package

Growth
Growth

61%

Renew Now

Expires in: 7 months | Duration: 1 year

Activity Log

AA asd asd
Created new task: Done
03.01.2020 00:24

AA asd asd
Created new task: Done
03.01.2020 00:23

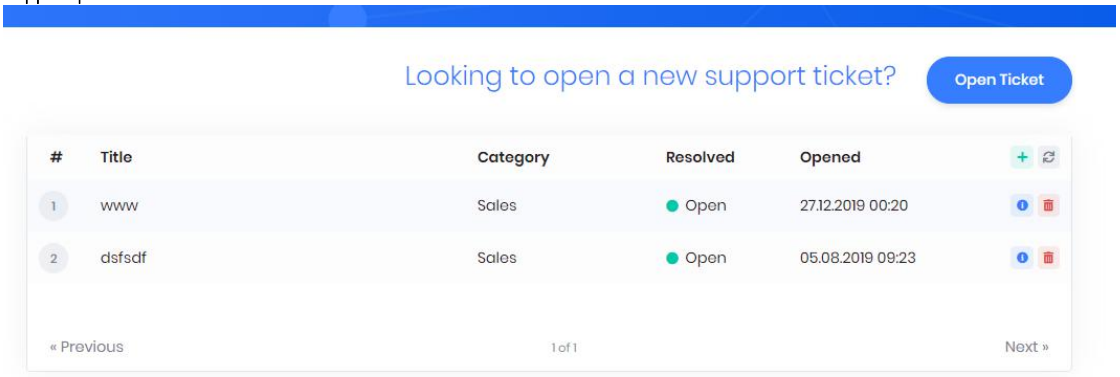
AA asd asd
Created new task: Done
03.01.2020 00:22

AA asd asd
Created new task: Done

Latest News | Read All

August Update
This is our update
01.08.2019

Support panel:

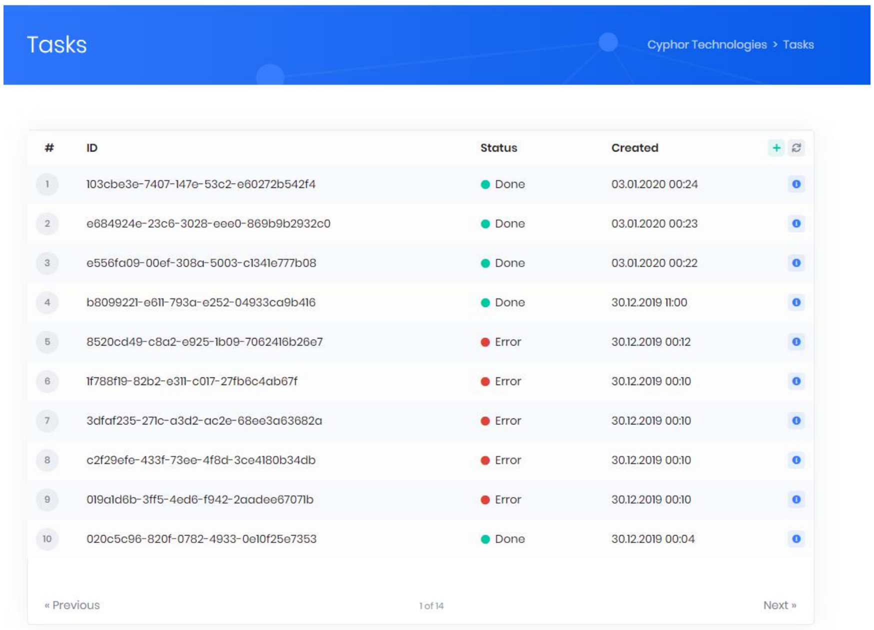


Looking to open a new support ticket? Open Ticket

#	Title	Category	Resolved	Opened
1	www	Sales	● Open	27.12.2019 00:20
2	dsfsdf	Sales	● Open	05.08.2019 09:23

« Previous 1 of 1 Next »

Task history:



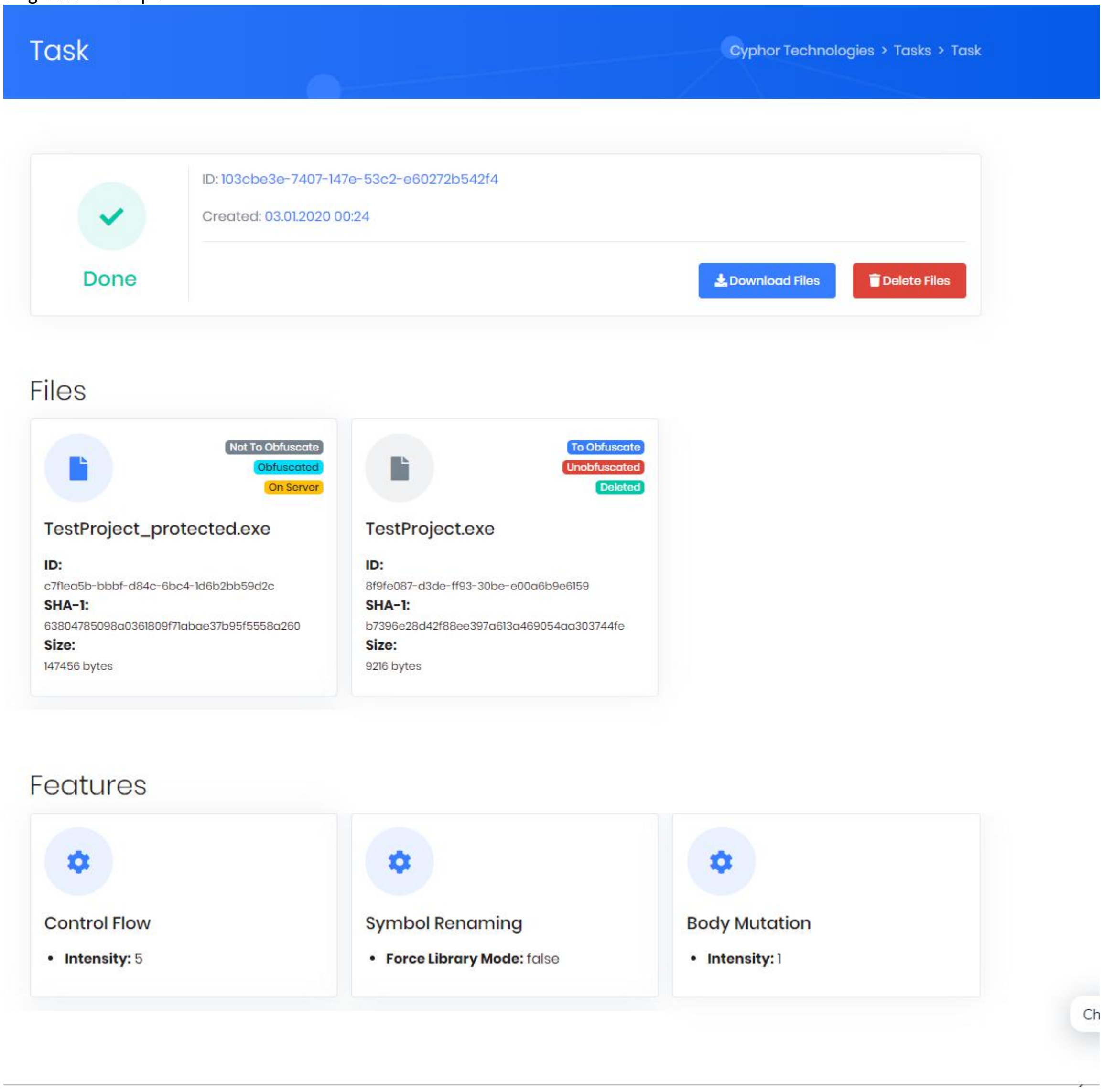
Tasks

Cyphor Technologies > Tasks

#	ID	Status	Created
1	103cbe3e-7407-147e-53c2-e60272b542f4	🟢 Done	03.01.2020 00:24
2	e684924e-23c6-3028-eee0-869b9b2932c0	🟢 Done	03.01.2020 00:23
3	e556fa09-00ef-308a-5003-c1341e777b08	🟢 Done	03.01.2020 00:22
4	b8099221-e611-793a-e252-04933ca9b416	🟢 Done	30.12.2019 11:00
5	8520cd49-c8a2-e925-1b09-7062416b26e7	🔴 Error	30.12.2019 00:12
6	1f788f19-82b2-e311-c017-27fb6c4ab67f	🔴 Error	30.12.2019 00:10
7	3dfaf235-271c-a3d2-ac2e-68ee3a63682a	🔴 Error	30.12.2019 00:10
8	c2f29efe-433f-73ee-4f8d-3ce4180b34db	🔴 Error	30.12.2019 00:10
9	019a1d6b-3ff5-4ed6-f942-2aadee67071b	🔴 Error	30.12.2019 00:10
10	020c5c96-820f-0782-4933-0e10f25e7353	🟢 Done	30.12.2019 00:04

« Previous   1 of 14   Next »

Single task example:



Task

Cyphor Technologies > Tasks > Task

Done

ID: 103cbe3e-7407-147e-53c2-e60272b542f4

Created: 03.01.2020 00:24

Files

TestProject_protected.exe

ID:
c7f1ea5b-bbbf-d84c-6bc4-1d6b2bb59d2c
SHA-1:
63804785098a0361809f71abae37b95f5558a260
Size:
147456 bytes

TestProject.exe

ID:
8f9fe087-d3de-ff93-30be-e00a6b9e6159
SHA-1:
b7396e28d42f88ee397a613a469054aa303744fe
Size:
9216 bytes

Features

Control Flow

- **Intensity:** 5

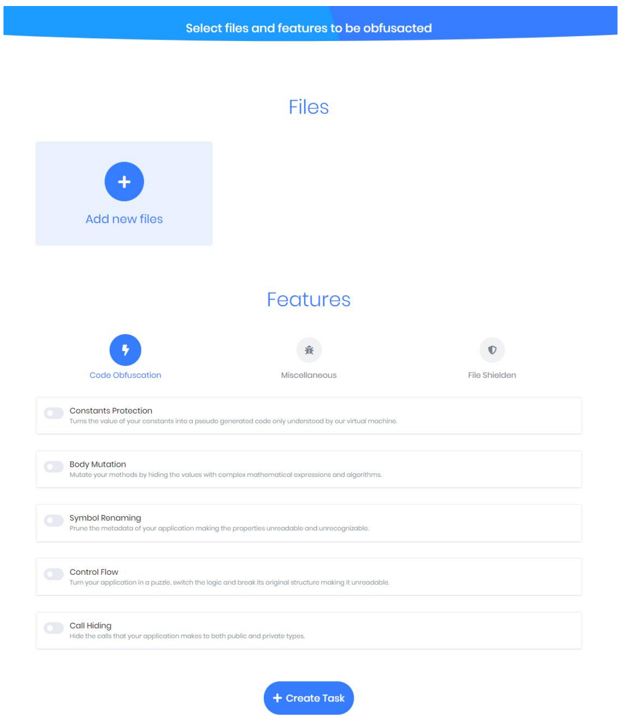
Symbol Renaming

- **Force Library Mode:** false

Body Mutation

- **Intensity:** 1

Obfuscation Panel:


Select files and features to be obfusacted

Files

Add new files

Features

Code Obfuscation | Miscellaneous | File Shielden

Constants Protection
Turns the value of your constants into a pseudo generated code only understood by our virtual machine.

Body Mutation
Mutate your methods by hiding the values with complex mathematical expressions and algorithms.

Symbol Renaming
Prune the metadata of your application making the properties unreadable and unrecognizable.

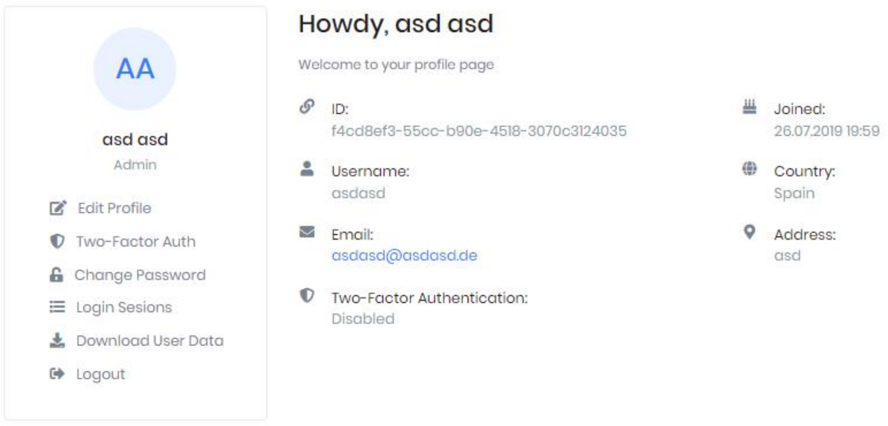
Control Flow
Turn your application in a puzzle, switch the logic and break its original structure making it unreadable.

Call Hiding
Hide the calls that your application makes to both public and private types.

+ Create Task

The photo showed is our current implementation, since the initial design has base errors, we are required to modify it. This will be explained carefully on a later stage of the PDF.

User Panel:


AA

asd asd
Admin

Edit Profile
Two-Factor Auth
Change Password
Login Sesions
Download User Data
Logout

Howdy, asd asd

Welcome to your profile page

ID:
f4cd8ef3-55cc-b90e-4518-3070c3124035

Username:
asdasd

Email:
asdasd@asdasd.de

Two-Factor Authentication:
Disabled

Joined:
26.07.2019 19:59

Country:
Spain

Address:
asd

Download user panel part will be discarded in this new website.

Initial features:

- Login / Register.
- Forgot password.
- News.
- Support.
- Shop.
- Dashboard.

- User profile.
- Obfuscation.
- Admin Panel.
- User Profile.
- Historical Usage.

Login and register:

For auth, we will use Argon2 as the hashing algorithm.
Upon register, we will ask the user to fill the following fields:

If Individual (freelancer, etc):
- Username.
- Email address.
- Password.
- First Name.
- Last Name.
- Address.
- Country.

If Business:
- Username.
- Email address.
- Password.
- First Name.
- Last Name.
- Business Name
- Busines Number
- VAT ID (optional)
- Address.
- Country.

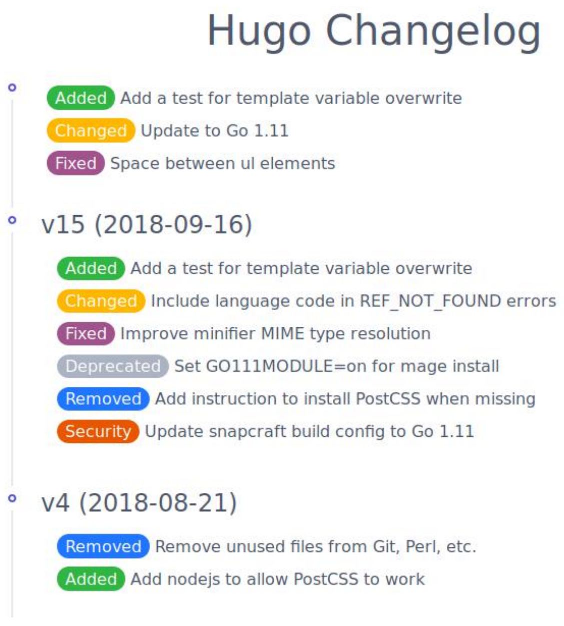
At the same time, before being able to register, the user must accept our TOS and pass the captcha. (GOOGLE RECAPTCHA V3).
Finally, to log in, the user must pass our captcha and fill with proper credentials, if the user has no 2FA, we will redirect them to dashboard, otherwise to 2FA page.

FAQ:
Does the user need to verify their email upon register? No.
Does the password need to meet specific security standards? Yes, a minimum of 6 chars length.
What is allowed as a username? Usernames must be between 4 and 20 characters. [a-Z]\[0-9].
Are we sending an email after register? Yes, note that we have HTML templates for emails. We're using mailgun.org for sending emails.

<u>How are sessions supposed to work:</u>
Upon login, we will make sure that the user does not logout randomly (defining random as any reason external to cookies expiring or the user explicitly clicking logout).
Our current website suffers from that issue; if an account is logged in multiple devices, it will disconnect in the machines that were initially connected.
We want to keep sessions open even if somebody logs in the account from another location.
Having this in mind, we will want to display in the login history, the active sessions that the account has linked.

Changelogs:

Changelogs are small posts that will be displayed in the dashboard of the website, they only support plain text, but we would like to be able to inject icons to determine the nature of the update (added, removed, fixed ...)
Proof of concept of what we need:



Hugo Changelog

Added Add a test for template variable overwrite
Changed Update to Go 1.11
Fixed Space between ul elements

v15 (2018-09-16)

Added Add a test for template variable overwrite
Changed Include language code in REF_NOT_FOUND errors
Fixed Improve minifier MIME type resolution
Deprecated Set GO111MODULE=on for mage install
Removed Add instruction to install PostCSS when missing
Security Update snapcraft build config to Go 1.11

v4 (2018-08-21)

Removed Remove unused files from Git, Perl, etc.
Added Add nodejs to allow PostCSS to work

Shall the developer know that there is no need to over-engineer this feature as we could allow basic HTML format in these changelogs, and we would inject those icons manually.

Support:

If the account type is business, we will offer the customer the option to make the ticket "critical".
When a ticket gets marked as critical, we will show an icon in the support panel that shows that the ticket is critical. At the same time, an email will be sent to all the accounts that have an admin role.

Besides that field, we will ask the user the following:

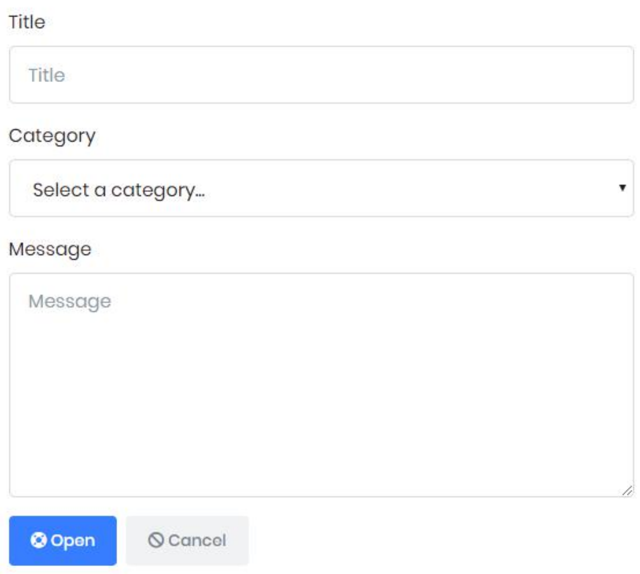
- Title of the ticket.
- Body of the ticket.
- Category of the ticket
- Status.
- **Note to dev:** It's planned that in the future, we assign tickets to individual staff members.
- **Note to dev:** We will notify the STAFF members, whenever ANY ticket receives an update, we will achieve such by implementing Desktop notifications

What is Status?
Status is the progress made towards the resolution of the ticket when resolving the arising problem, the supporter has the option to set the status value, which goes from 0 to 100.
When Status reaches 100, the ticket will automatically close.

Note: It is EXTREMELY important that all the values attached are interpreted as text, we have seen our old system being exploited by XSS in the past.

Proof of concept (lacks of status feature and critical):

Client side:


Title

Title

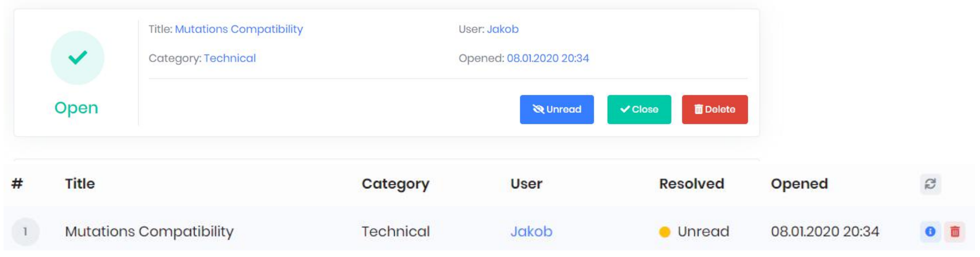
Category

Select a category...

Message

Message

Supporter side:


Title: Mutations Compatibility | User: Jakob

Category: Technical | Opened: 08.01.2020 20:34

#	Title	Category	User	Resolved	Opened
1	Mutations Compatibility	Technical	Jakob	● Unread	08.01.2020 20:34

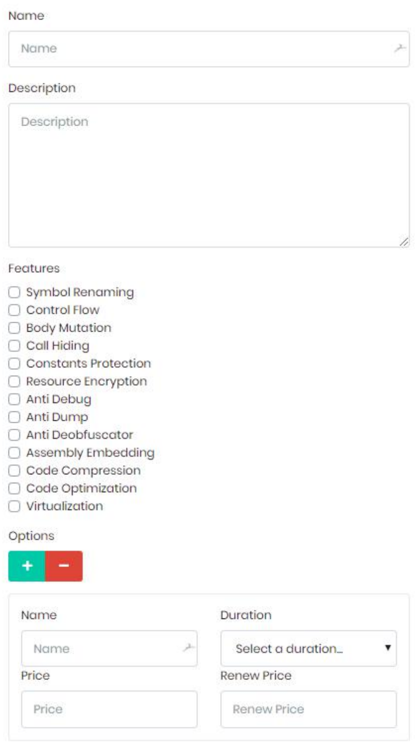
Note: It is important that, whenever an update to a ticket occurs, the admins get notified with a Desktop notification or email. Ideally we want to use desktop notifications, however, if this implies bigger complications on the development, emails can be a valid option.

Shop:

The whole shop follows a subscription-based service. We will be providing two different products:

- The plans, which include a set of features at a fixed price.
- The features, which are just individual settings used when using our service.

Each plan and feature has two prices: **The initial payment and the renewal payment,** the initial payment is only paid once, and the renewal is paid yearly after the initial payment was made.
Proof of concept of how a plan is built:


Name

Name

Description

Description

Features

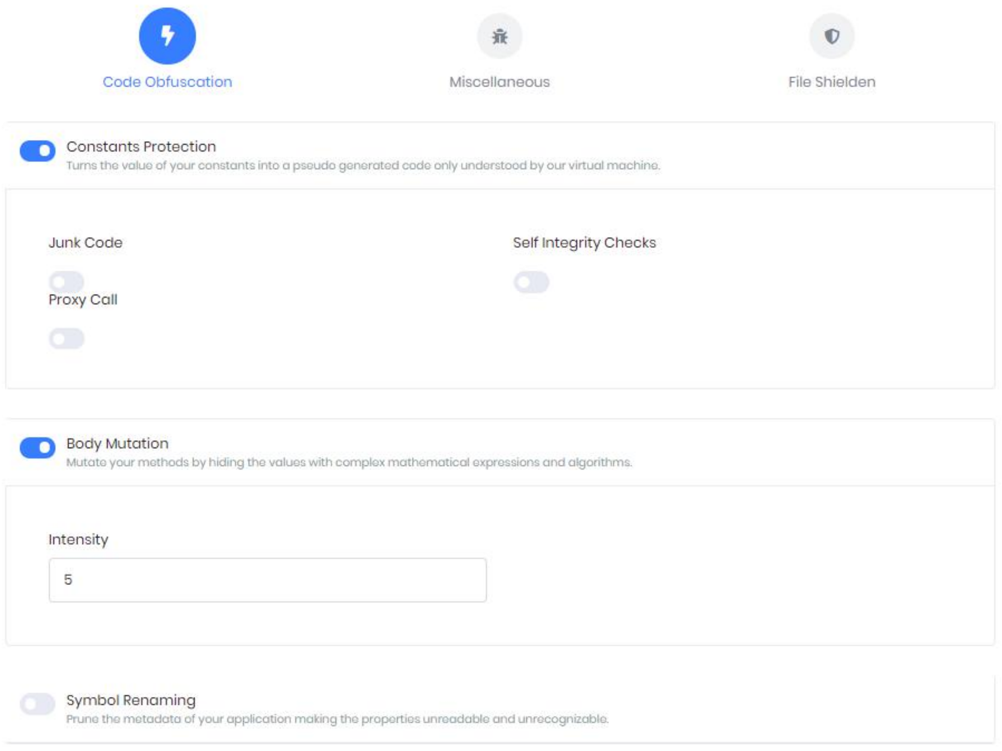
- [ ] Symbol Renaming
- [ ] Control Flow
- [ ] Body Mutation
- [ ] Call Hiding
- [ ] Constants Protection
- [ ] Resource Encryption
- [ ] Anti Debug
- [ ] Anti Dump
- [ ] Anti Deobfuscator
- [ ] Assembly Embedding
- [ ] Code Compression
- [ ] Code Optimization
- [ ] Virtualization

Options

+ −

Name	Duration
Name	Select a duration...
Price	**Renew Price**
Price	Renew Price

We will describe what does a feature represent in a later stage of the pdf.



Constants Protection
Turns the value of your constants into a pseudo generated code only understood by our virtual machine.

Junk Code

Self Integrity Checks

Proxy Call

Body Mutation
Mutate your methods by hiding the values with complex mathematical expressions and algorithms.

Intensity

5

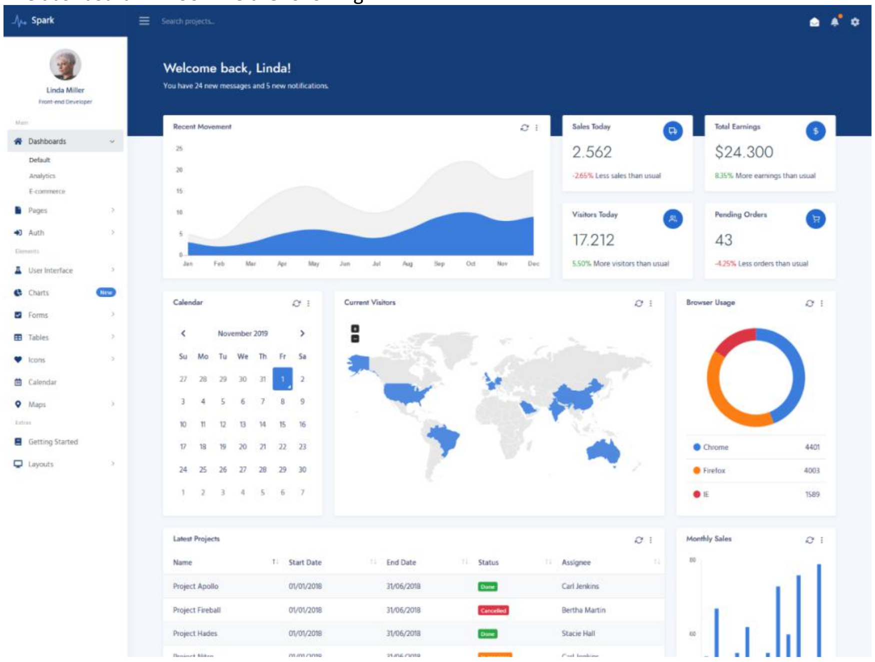
Symbol Renaming
Prune the metadata of your application making the properties unreadable and unrecognizable.

We will initially have two gateways in our services: Stripe and Paypal, it is planned that in a future update we add CoinGate as well.

Finally, it is important that shortly after the purchase (only if successful), we redirect the user to a “success” page, this page will only have the purpose for google and other SEO companies to measure successful leads. (the users that after visit, get to this page will be considered as clients and thus makes a successful lead).

Dashboard:

The dashboard will look like the following:



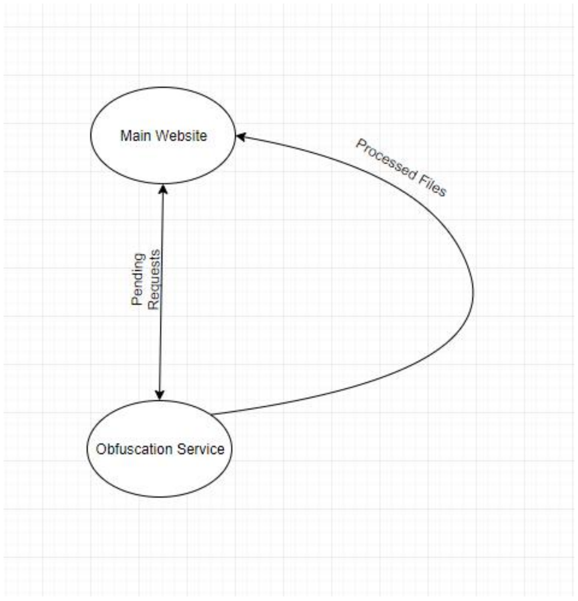
Name	Start Date	End Date	Status	Assignee
Project Apollo	01/01/2018	31/06/2018	Done	Carl Jenkins
Project Fireball	01/01/2018	31/06/2018	Cancelled	Bertha Martin
Project Hades	01/01/2018	31/06/2018	Done	Stacie Hall

The dashboard will display the following information:
Brief summary of obfuscation history.
Overall statistics about the obfuscator (such as version, total obfuscated files …)
Latest obfuscation tasks.
Latest News about the obfuscator.
The Dashboard was previously described properly in the design section, the developer shall refer to that section to get a further list of the details we need to display in this apge.

Obfuscation:

Clarification: The developer does not need to create the obfuscation service; we have that handled already. The developer will handle the part of the website, which does the following.
1) After files are transferred to our web servers via the Obfuscation Panel and task is created, we will allow our obfuscation service to retrieve these files and process them (this will be done via an API).
2) When the obfuscation service is done processing the files, it will bring us back the new files via a new API request.
These new files are what we will deliver to the customer.

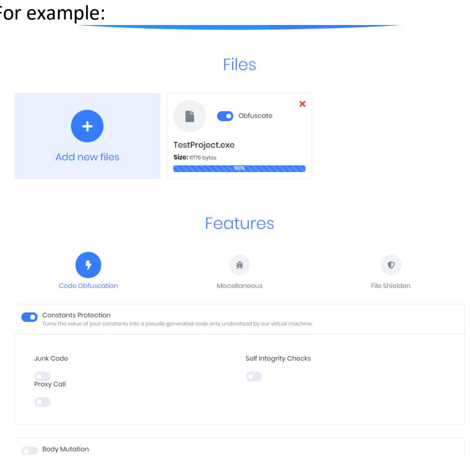
Idea visualized:



The Main Website API:
Our API will only have three purposes for now.
1) Deliver pending tasks to the Obfuscation Service.

2) Profile Checker.
3) Update task results.
We will now scrutinize each feature.

<u>Deliver pending tasks to the Obfuscation Service.</u>
After the user clicks the "Obfuscate" button in our panel, we will create a new task in our database, which will have linked all the files the user updated for the task plus the settings that were picked on the panel.
For example:



Files

Add new files

Obfuscate
TestProject.exe
Size: 11776 bytes
100%

Features

Code Obfuscation
Miscellaneous
File Shielden

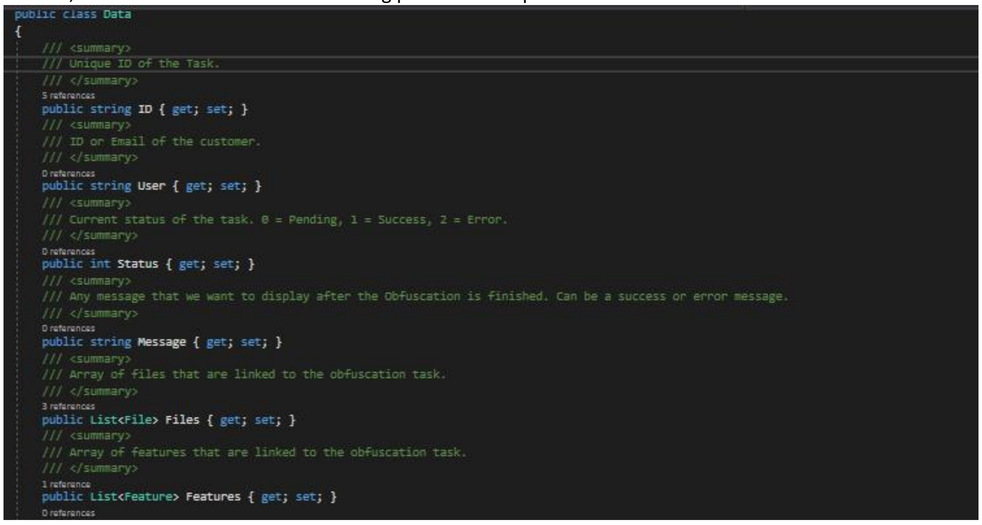
Constants Protection
Turns the value of your constants into a pseudo generated code only understood by our virtual machine.

Junk Code
Proxy Call
Self Integrity Checks

Body Mutation

If the user clicked obfuscate, we would link to the new task the file “TestProject” and the feature(s) Constants Protection.

<u>FAQ:</u>
How many file(s) will the user upload? It depends on the customer; it goes from 1 to 50.
How do we upload the files to the server? Because the size of the data can be significant, we require to use chunked file uploads. We must get this done in the best way possible as otherwise, we will have a lot of problems.
Please be aware of the possible security risks not implementing this feature properly can lead to, we don't want people to write unknown files into our disks.
This Github resource achieves what we need for uploading the files: https://github.com/khanzadimahdi/UploadManager
What is the exact structure of the Task? When the Obfuscation service queries this feature of the API, we will return an **ARRAY** of tasks, each task will look like the following proof of concept:



```
public class Data
{
    /// <summary>
    /// Unique ID of the Task.
    /// </summary>
    5 references
    public string ID { get; set; }
    /// <summary>
    /// ID or Email of the customer.
    /// </summary>
    0 references
    public string User { get; set; }
    /// <summary>
    /// Current status of the task. 0 = Pending, 1 = Success, 2 = Error.
    /// </summary>
    0 references
    public int Status { get; set; }
    /// <summary>
    /// Any message that we want to display after the Obfuscation is finished. Can be a success or error message.
    /// </summary>
    0 references
    public string Message { get; set; }
    /// <summary>
    /// Array of files that are linked to the obfuscation task.
    /// </summary>
    3 references
    public List<File> Files { get; set; }
    /// <summary>
    /// Array of features that are linked to the obfuscation task.
    /// </summary>
    1 reference
    public List<Feature> Features { get; set; }
    0 references
```

What format to use? We will be using JSON.

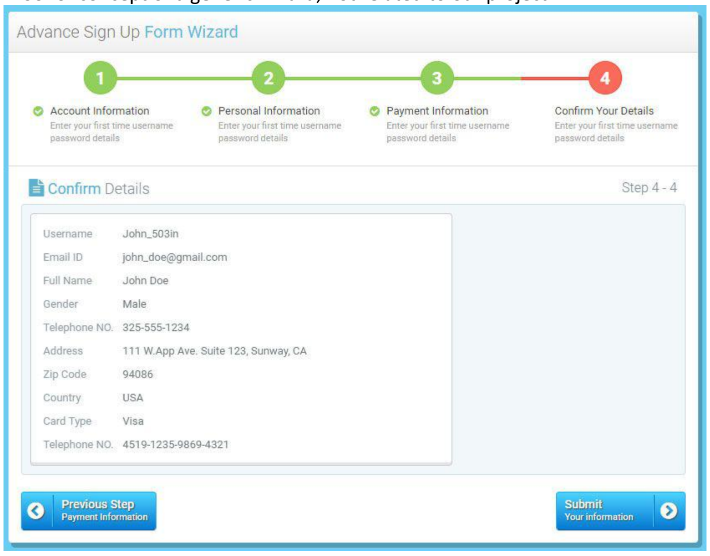
<u>Profile Checker.</u>
The following feature is straightforward, since the users can use our SDK for adding special obfuscation in special parts of their code (**This is something handled outside of the website**).
This feature allows the user to select features in our SDK and not on the website, allowing them to use features they don’t have access to. To counter this “bypass” we require of an API feature that, from an USER_ID or EMAIL, returns the available features for that account.

What format to use? We will be using JSON.

<u>Update task results</u>.
After a task is fully processed by our obfuscation service, we will have to update the website task. This will be achieved using the previous ID that identified our Task.

The HTTP request that we will send to update the task has the following fields:
(String Content) Success: Will be an string: True/False, this will set the task status to Error or Success.
(String Content) Message: A message generated by the obfuscation service, must be displayed on the user panel. It's present whether if the obfuscation was successful or not.
(Byte Array Content) Files: A set of files that were generated after the obfuscation service finished processing the files. To simplify the work on this stage, we will ZIP all the files so that if the user wants to retrieve the files, he/she will download a zip file and not each file one by one.

The Interface:
The interface must be as intuitive as possible. We thought of using wizard to achieve the most user-friendliness.
Proof of concept of a generic wizard, not related to our project:



Advance Sign Up Form Wizard

1 Account Information — Enter your first time username password details
2 Personal Information — Enter your first time username password details
3 Payment Information — Enter your first time username password details
4 Confirm Your Details — Enter your first time username password details

Confirm Details — Step 4 - 4

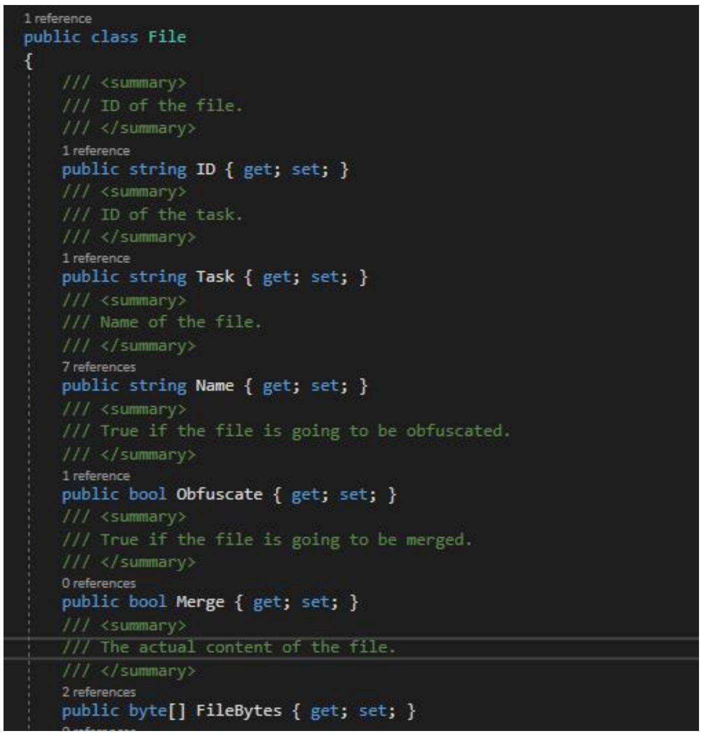
Username	John_503in
Email ID	john_doe@gmail.com
Full Name	John Doe
Gender	Male
Telephone NO.	325-555-1234
Address	111 W.App Ave. Suite 123, Sunway, CA
Zip Code	94086
Country	USA
Card Type	Visa
Telephone NO.	4519-1235-9869-4321

We will have the following stages (understanding as stage each "layer" of the wizard form).
1) Files to obfuscate, the user will upload those files he wants to protect. **Remember**, a single task can have more than 1 file to protect.
2) Dependencies, the user will upload the dependencies of the main application, these are files that are required to run the main app and are not obfuscated. These files are still forwarded to the obfuscation service.
The dependencies will have the following options: ***Merge and Cache.***
3) **Obfuscation Settings**, we will display the different obfuscation categories we have and the file we're setting the features for.
4) **Obfuscation**, the final stage in which the only thing left is to press obfuscate.

Shortly after obfuscation is pressed, we will redirect the user to an **status page** in which we will be querying the API for information regarding the task. This will be achieved by using ajax any other technology that allows us to update the page in real time.

What is Merge?
Merge is just a Boolean field that will be linked with the File when transferred to the Obfuscation service.
How do we deal
What is Cache?
Caching is a feature made so that the customer does not have to upload the same dependencies all over again. Ideally we will create profiles, and inside those profiles we will store the files that were marked as "Caching". The user can create or remove profiles at any time.
We should discuss with the developer how this feature can be implemented as we thought of different ways that could be simpler.

How should we transfer each file?
Following the structure of the Task we described before, we had an array of files linked with the task.
Each **File instance** stores the file itself plus the settings linked to it, visualized we get the following structure:



```
1 reference
public class File
{
    /// <summary>
    /// ID of the file.
    /// </summary>
    1 reference
    public string ID { get; set; }
    /// <summary>
    /// ID of the task.
    /// </summary>
    1 reference
    public string Task { get; set; }
    /// <summary>
    /// Name of the file.
    /// </summary>
    7 references
    public string Name { get; set; }
    /// <summary>
    /// True if the file is going to be obfuscated.
    /// </summary>
    1 reference
    public bool Obfuscate { get; set; }
    /// <summary>
    /// True if the file is going to be merged.
    /// </summary>
    0 references
    public bool Merge { get; set; }
    /// <summary>
    /// The actual content of the file.
    /// </summary>
    2 references
    public byte[] FileBytes { get; set; }
```

The Admin Panel:

The admin panel will have the following features:

- Add or Remove features.
- Add or Remove plans.
- Modify Features.
- Modify Plans.
- Add/Remove/Modify Obfuscation Categories.
- Interact with Tickets (Reply, Close, Set Status).
- View Accounts (All the information we store about the user, logins, register IP, active plans).
- Edit Accounts (Send Reset Email, Reset Password, Ban, modify owned features/plan(s)).
- Create News.
- View Orders.
- Download Invoices.
- Create Invoices.

Notes:

- It is essential that no user with Admin panel can give staff role to other accounts.
- Invoices are manual, meaning that them will follow our standard model with the difference of having a custom price/features that we will set. Once paid, we will assign the features/plan(s) manually.

The User Panel:

The user profile is the zone of the website in which the account holder can modify the data regarding his account.
Thus includes:

- Login(s) to the account, along with active sessions.
- Purchases made on the website.
- Enable/Disable 2FA.
- Task History.
- Reset Email.
- Reset password.
- Information regarding the account (date of register, Country, First Last name, email associated with the account).