

GNU GPL

GNU GPL



LINKEDIN



USV StudDocs – facilitating interaction with your University

An awesome software tool for interacting with the dean office for getting student's certificate!

[Explore the docs »](#)

[View Demo](#) · [Report Bug](#) · [Request Feature](#)

► [Table of Contents](#)

About The Project

This is a piece of software which is aimed to facilitate the interaction with the dean office in USV. It uses USV Google OAuth2 authentication which makes it even easier to use.



Universitatea
„Ștefan cel Mare”
din Suceava



AUTORIZAȚI-VA CU USV GOOGLE

By signing up or signing in you're agree with our [terms and conditions](#)

Admin user could do the managing for the dictionaries and basic settings:

USV Stud Docs



Setari de baza



Lista studentilor



Programele de studii



Facultatile



Personalul facultatilor

Setari administrative

Anul universitar curent

Aici puteti selecta **inceputul** anul universitar curent.

2022.

lun., 3 oct.

< octombrie 2022 >

D	L	M	M	J	V	S
						1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	31					

Motivele predefinite pentru cereri

Vor fi afisate studentilor. Totodata studentii vor avea posibilitatea de a introduce manual motivul, daca respectivul nu va fi gasit in aceasta lista.

Introduceti motivul ca text dupa care apasati Enter si veti putea introduce urmatorul.

Motivele predefinite pentru cereri

Motivul1 este aici

Motivul3 pentru studenti3

Motivul4 pentru studenti

Autorizare cont pentru expediere e-mail

Cu acest buton se testa expedierea email-urilor. Vetii primi un email pe contul indicat.

[TEST EMAIL](#)

USV Stud Docs

- Setari de baza
- Lista studentilor
- Programele de studii
- Facultatile
- Personalul facultatilor

Facultatea de inginerie electrica si
stiinta calculatoarelor

IMPORT

Stiinta si ingineria calculatoarelor

Anul 1

Anul 2

Importul studentilor

De aici puteti descarca template-uri CSV pentru completarea lor cu datele studentilor pentru importul ulterior.



Versiunea cu numele, initiala tatalui si prenumele in coloane diferite - DESCARCA

Versiunea cu numele, initiala tatalui si prenumele concatenate impreuna intr-o coloana - DESCARCA



In cazul in care numele, initiala tatalui si prenumele sunt concatenate intr-un singur sir, este important ca initiala tatalui sa fie urmata de un punct ".", iar in cazul in care prenumele tatalui este dublu - initialele sa nu contina spatii (de exemplu, Alin Petru trebuie sa fie ca "A.P").

In cazul in care numele, initiala tatalui si prenumele se afla intr-o singura coloana, va rugam sa selectati optiunea de mai jos



Numele, initiala tatalui si prenumele sunt intr-o singura coloana



CSV/XLSX fisierul cu studentii

INCARCA CSV/XLSX

USU

Universitatea
Științelor
și Tehnologiei

ADRYANBARBE@GMAIL.COM

USV Stud Docs

Setari de baza

Lista studentilor

Programele de studii

Facultatile

Personalul facultatilor

Facultatea de inginerie electrica si
stiinta calculatoarelor

IMPORT

Stiinta si ingineria calculatoarelor

Anul 1

Anul 2

	#	Numele, Prenumele	Email	Actiuni
<input type="checkbox"/>	1		an.barbe@student.usv.ro	
<input type="checkbox"/>	2		u.stroe1@usv.ro	

Editarea

Nume

Barbe

Prenume

Adrian

Initiala tatalui

M.

Email

adrian.barbe@student.usv.ro

☒ Masculin

☐ Feminin

REFUZA

SALVEAZA

+

USV

Universitatea
Ștefan cel Mare
de Suceava

ADRYANBARBE@GMAIL.COM

USV Stud Docs

Setari de baza

Lista studentilor

Programele de studii

Facultatile

Personalul facultatilor

Numele	Prenumele	Initiala tatalui	Username	Email	Actiuni
Dospinescu	Laura		adrian.barbe@intercode.io	adrian.barbe@intercode.io	<div></div> <div></div>
Milici	Dan				<div></div> <div></div>
Curelaru	Elena				<div></div> <div></div>

Rânduri pe pagină: 10 1-3 din 3


Student would use the same auth screen and inside the app will see the list of already made requests and the button for creating a new one.



Numele, IT, Prenumele: Barbe M. Adrian
Facultatea: FIESC
Domeniu de studii: Masterat
Program de studii: SIC
An studii: 1
Statut financiar: Fără taxă

CREAZA NOUA CERERE PENTRU ADEVERINTA

Data trimiterii	Numarul de inregistrare	Motivul	Starea	Secretarul responsabil
23.06.2023 03:51	125/1/FIESC/22.06.2023	Motivul1 este aici	Seminat	Dospinescu Laura
08.06.2023 16:48	124/1/FIESC/08.06.2023	Motivul1 este aici	Aprobat	Dospinescu Laura
Rânduri pe pagină: 10 1-2 din 2 < >				



ADRIAN.BARBE@STUDENT.USV.RO

Numele, IT, Prenumele:

Barbe M. Adrian

Facultatea:

FIESC

Domeniu de studii:

Masterat


Program de studii:

SIC

Cererea unei noi adeverinte

Rugam sa selectati motivul cererii din lista predefinita, iar in cazul cand nu va fi gasita una corespunzatoare se va indica motivul manual.

Sa se ia in considerare ca nu puteti trimite alta cerere de adeverinta pana cand cea precedenta nu va fi aprobata sau respinsa.



Odata depusa cererea aici, ea poate fi aprobata sau respinsa de secretarul corespunzator programului de studii. Aceasta va fi vizualizat aici si de asemenea veti primi un e-mail. Daca cererea este respinsa, veti primi si informatii cu privire la motivul respingerii.

Odata aprobata, adeverinta va fi semnata si veti primi o confirmare ca adeverinta poate fi ridicata. In caz ca confirmarea nu a fost primita dupa un timp adecvat dupa aprobare, va trebui sa mergeti in persoana la secretariat sa ridicati adeverinta.

Selectati din lista predefinita sau indicati manual motivul cererii

REFUZA

TRIMITE CEREREA

Dean office secretary will receive the certificate request and will follow up it according to the procedure.



ADRIAN.BARBE@INTERCODE.IO

NOI

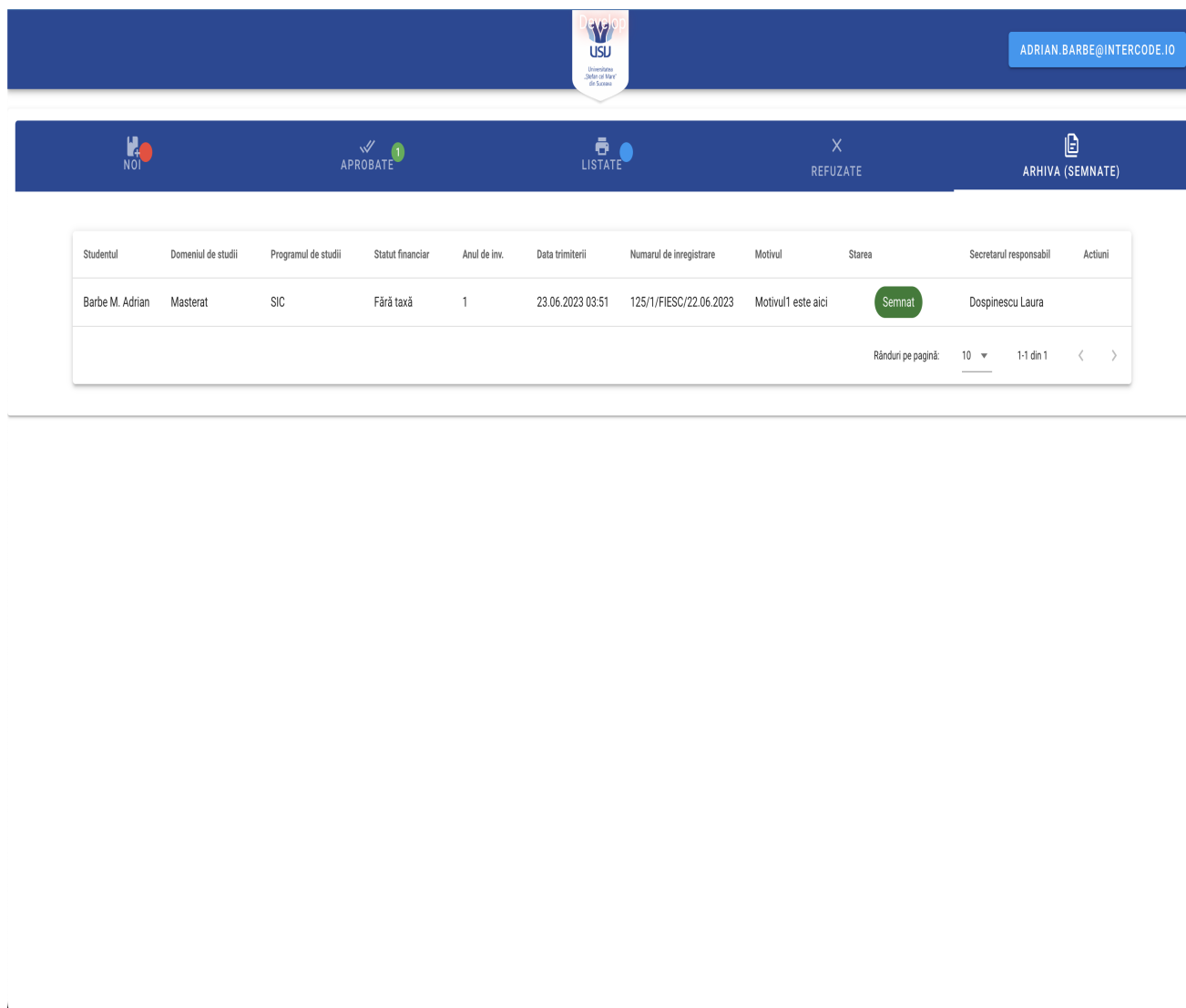
APROBATE 1

LISTATE

REFUZATE

ARHIVA (SEMNATE)

Studentul	Domeniul de studii	Programul de studii	Statut financiar	Anul de inv.	Data trimiterii	Numarul de inregistrare	Motivul	Starea	Secretarul responsabil	Actiuni
Barbe M. Adrian	Masterat	SIC	Fără taxă	1	08.06.2023 16:48	124/1/FIESC/08.06.2023	Motivul1 este aici	Aprobat	Dospinescu Laura	
Rânduri pe pagină: 10 1-1 din 1										



[\(back to top\)](#)

Built With

- 
- 
- 

[\(back to top\)](#)

Project design

This project is build as a Client-API application with a client developed using the VueJs 3 front-end framework and API developed by using .NET 6 C#. We used Entity Framework as main ORM and the chosen approach was Code First. This choice is dictated by the usability and maintainability of this approach. It creates uniform namings across the DB, there's no need to write down SQL code and maintain SQL code base.

The project is a solution that consists of 5 .NET projects that corresponds to the architectural layers.

For this project we have chosen 3-tier / Layer architecture. It comprises of 3-tiers, Presentation Layer, Business Logic Layer, and Data Access Layer. Each Layer will have its own namespace, assembly and project (classes and folders) in the solution.

We considered this architecture because there are some benefits of N-tier Architecture:

- Reuse of the code
- Improve of the Maintainability
- Looser Coupling

In our project we have next sub-projects:

- USVStudDocs.DAL - Data access layer, where the entity configurations, migrations and EF Context is saved.
- USVStudDocs.Entities – a class project where the Entities are grouped. They are suffixed by "*Entity"
- USVStudDocs.Models - a class project where the Data Transfer Objects (DTO) are located
- USVStudDocs.BLL - Business logic layer, a class project where the services and other business logic (authentication, 3rd party API integration, validation) is stored.
- USVStudDocs.Web – API project, presentation layer. A set of Controllers with exposed API endpoints, where the routing, authentication guards are implemented.
- USVStudDocs.UI – UI project, VueJs. Is what the user could see.

Implementation

Some notable libraries (NuGets) used in our application are:

- FluentValidation – for DTO validation
- RestSharp - for sending API requests during the validation of OAuth2 code
- Serilog and Serilog.Sinks.Console and Serilog.Sinks.Telegram – for collecting and sending logs to console, and warnings and up – to Telegram.
- JWT - for generating user-facing JWT

Business layer

Business layer contains logic for Authentication, Custom Exceptions, Extension methods, Helpers, Mappers (between Entities and DTOs and vice-versa) and the most important – Services.

Validators are created based on FluentValidation.

Among services, there are CRUD Services for administration, for creating the certificate requests, the services for use by dean office secretary. There is also AwsMinioClient service that is used for uploading files to MinIO

Some notable places in BLL are:

- use of IHttpContextAccessor in BLL for getting the current user.

```
var subValue = _httpContextAccessor.HttpContext.User.FindFirst(JwtRegisteredClaims.Sub)?.Value;

bool parseRes = int.TryParse(subValue, out var userId);

if (!parseRes)
{
    Log.ForContext<AuthorizationService>().Error("Cannot parse user id");

    throw new ValidationException("User id is not an integer");
}

return userId;
}
```

- JWT generation

```
var jwtSecretKey = Environment.GetEnvironmentVariable("JwtSecretKey") ?? _jwtSettings.SecretKey;

var tokenBuilder = new JwtBuilder()
    .WithAlgorithm(new HMACSHA256Algorithm())
    .WithSecret(jwtSecretKey)
    .AddClaim(JwtRegisteredClaimNames.Exp,
        DateTimeOffset.UtcNow.AddMinutes(_jwtSettings.ExpirationMinutes))
    .AddClaim(JwtRegisteredClaimNames.Iss, _jwtSettings.Issuer)
    .AddClaim(JwtRegisteredClaimNames.Aud, _jwtSettings.Audience)
    .AddClaim(JwtRegisteredClaimNames.Sub, userEntity.Id)
    .AddClaim(JwtRegisteredClaimNames.UniqueName, userEntity.Username)
    .AddClaim(JwtRegisteredClaimNames.Email, userEntity.Email)
    .AddClaim(JwtRegisteredClaimNames.Iat, DateTimeOffset.UtcNow.AddMinutes(0));

tokenBuilder.AddClaim("role", roles);

return tokenBuilder.Encode();
```

- Splitter of the Surname, Patronymic initials and name

```
var studentSurnameNamePatronymic = res.Result.SurnameNamePatronymic.Trim()
    .Replace("`", "").Replace("'", "")
    .Replace("''", "").Replace(" ", " ")
    .Split(" ");
```

```

List<string> surname = new List<string>();
List<string> name = new List<string>();
List<string> patronymic = new List<string>();

foreach (var studentString in studentSurnameNamePatronymic)
{
    if (!studentString.Contains('.') && patronymic.Count == 0)
    {
        surname.Add(studentString);
    }
    else if (studentString.Contains('.') && patronymic.Count == 0)
    {
        patronymic.Add(studentString);
    }
    else if (!studentString.Contains('.') && patronymic.Count == 0)
    {
        name.Add(studentString);
    }
}

```

- Authorizing e-mail service to use Gmail account

```

var clientSecrets = new ClientSecrets
{
    ClientId = clientId,
    ClientSecret = clientSecret
};

var token = new TokenResponse
{
    AccessToken = accessToken,
    RefreshToken = refreshToken
};

var credential = CreateUserCredential(clientSecrets, token, oAuthEmail)

// Check if the access token has expired
if (credential.Token.IsExpired(credential.Flow.Clock))
{
    // Refresh the access token
    if (credential.RefreshTokenAsync(CancellationToken.None).Result)
    {
        accessToken = credential.Token.AccessToken;
        oAuthEmailAccessToken.Value = accessToken;
        _context.Settings.Update(oAuthEmailAccessToken);

        refreshToken = credential.Token.RefreshToken;
        oAuthEmailRefreshToken.Value = refreshToken;
        _context.Settings.Update(oAuthEmailRefreshToken);
    }
}

```

```

        _context.SaveChanges();
    }
    else
    {
        throw new Exception("Failed to refresh access token.");
    }
}

var service = new GmailService(new BaseClientService.Initializer
{
    HttpClientInitializer = credential
});

```

Deployment

For the project deployment we used Docker containers. In `Docker_files` you'll find configurations for the API and UI.

One of the simplest ways to run the app on the server is running by the means of Docker Swarm, using `docker-composer.yml`. As a server we used Ubuntu Server 22.04, and `nginx` as a reverse-proxy server. The following instruction will be for Ubuntu 22.04.

1. Please install Docker engine.

```

sudo apt-get update
sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin

```

- you may not like to use "sudo" every time you execute Docker commands.

If you want to omit this, then execute the following:

```

sudo groupadd docker
sudo gpasswd -a $USER docker

```

2. Initialize the Swarm:

```

sudo docker swarm init

```

Make a note of the command output, as it will contain the token needed to add other nodes to the Swarm.

3. Create a directory on your home directory or another which you prefer (e.g. `/home/usv-stud-docs`)

4. Copy the file `docker-compose.yml` to this directory on the server.

5. Update the ENVs in the `docker-compose.yml`.

- `POSTGRES_PASSWORD` - the desired PostresDB instance password.
- `ASPNETCORE_ENVIRONMENT` - to Development or Production
- `DbConnectionString` - according to this sample: `Host=<host ip>;Port=15432;Database=usv-stud-docs;Username=usv-user;Password=<password>` indicated in `POSTGRES_PASSWORD`
- `GoogleClientId` - which you obtained by visiting <https://console.cloud.google.com/apis/credentials> and creating oAuth2 credentials (after you created the project)
- `GoogleClientSecret` - the secret from the previous place
- `GoogleRedirectUri` - should correspond to the one indicated in Google Console, and should be according to this sample: `https://<host_name>/auth/redirect`
- `GoogleEmailRedirectUri` - this is used for authorizing for sending emails, should correspond to the one indicated in Google Console, and should be according to this sample: `https://<host_name>/auth/redirect`
- `JwtSecretKey` - a random string 16-32 chars length

6. Create new directory called `db`, assign the ownership to the user that will run the app (e.g. `usvdocs`) and assign 777 access rights.

```
mkdir db
sudo chown usvdocs db
sudo chmod 777 db
```

6. In real-world scenario you'd need to push the Docker image to your own private DockerHub, where the Docker images are stored. In our case, you could use the already stored public images, that are indicated in the `docker-compose.yml` (this needs no changes). This would require to execute `docker login -u {username}` command.

7. You need to start your services by executing next command:

```
docker stack deploy --compose-file docker-compose.yml usv-docs --with-registry
```

8. You may want to check the services status:

```
docker service ls
```

9. After the services were created, please connect to the DB (postgresql), don't forget to use the corresponding port from the docker-compose.yml and execute the file `USVStudDocs.DAL/MigrationSQL/migration.sql`.
10. After this your app instance will be populated with the data structure and the initial data.
11. You may want to set-up the load balancer, to serve the L7 network, e.g. nginx. You could find a sample Nginx set-up in `Docker_files/nginx_lb/usvstud.conf` which should be placed in `/etc/nginx/sites-available` and then a symbolic link (`ln -s`) should be created to `/etc/nginx/sites-enabled`.
12. You may also want to install some LetsEncrypt SSL certificates in case no other options are available:

```
sudo snap install core; sudo snap refresh core
sudo snap install --classic certbot
sudo ln -s /snap/bin/certbot /usr/bin/certbot
sudo certbot --nginx
```

Getting Started

This is a description of how you should run up this project locally. To get a local copy up and running follow these simple example steps.

Prerequisites

First of all, make sure you have installed NodeJs major version 16. You could download it from the [official site](#) or by using [nvm](#). I highly recommend you to use Yarn as a package manager tool. Please install it by next command:

```
npm install --global yarn
```

Here you could see the Swagger API documentation for the back-end end-points: [View API documentaion](#)

USVStudDocs.Web 1.0 OAS3

/swagger/v1/swagger.json

CertificateSecretary ^

GET	/api/certificateSecretary/getPrint/{id}	▼
GET	/api/certificateSecretary/{status}	▼
GET	/api/certificateSecretary/count	▼
PUT	/api/certificateSecretary/confirm/{id}	▼
PUT	/api/certificateSecretary/confirmPrint/{id}	▼
PUT	/api/certificateSecretary/confirmSignature/{id}	▼
PUT	/api/certificateSecretary/reject/{id}	▼

CertificateStudent ^

GET	/api/certificateStudent/me	▼
GET	/api/certificateStudent	▼
POST	/api/certificateStudent	▼

Faculty ^

GET	/api/Faculty	▼
POST	/api/Faculty	▼

Installation

There are two projects in the Git repository – UI and API. You'll find the UI project into RemoteFinder.UI directory.

1. `git clone https://github.com/adrianbarbe/RemoteFinder.git`

2. Install the NPM packages by executing

```
yarn install
```

3. Run the project for development

```
yarn serve
```

To run the API project, you'll need to create a .env file in RemoteFinder.Web with the following keys:

1. `ASPNETCORE_ENVIRONMENT` which could be `Development` or `Production`

2. `DbConnectionString` the connection string to your PostgreSQL database engine installation.
3. `GoogleClientId` is the client id for the Google OAuth2 Credentials. For creating new credentials please access [Google Developers Console](#)
4. `GoogleClientSecret` client secret generated in Google Developers Console
5. `GoogleRedirectUri` redirect URI which you indicated in the Google OAuth2 application settings
6. `GoogleEmailRedirectUri` redirect URI for email sending, which you indicated in the Google OAuth2 application settings
7. `JwtSettings:SecretKey` and `JwtSecretKey` – a hash string for JWT tokens.

After creating the `.env` file you could start your project locally by running `dotnet run` command or by using the configuration profile for JetBrains Rider which is stored in the project repository.

Deployment

In the project root you'll find a directory `Docker_files` that contains Docker definitions for projects and a `docker-compose.yml` file. You could use it for deploying the project into a Docker Swarm cluster. You could create it by installing Docker on the server and initialize a Swarm by `doker swarm init` command. Then, by placing the `docker-compose.yml` file in the desired directory, execute next command to initialize the Docker Swarm services:

```
docker stack deploy --compose-file docker-compose.yml usv-docs --with-registry
```

Don't forget to update correspondingly the environment variables in the `docker-compose.yml` file for `be` service.


[\(back to top\)](#)

Usage

Here you could see the usage examples for the student.

[View the Demo Here >>>>](#)

To add a new request for a certificate to your dean office just click the main button above the list of the request and you'll see this Add New Certificate request modal:



ADRIAN.BARBE@STUDENT.USV.RO

Numele, IT, Prenumele: Barbe M. Adrian

Facultatea: FIESC


Domeniu de studii: Masterat

Program de studii: SIC

Cererea unei noi adeverinte

Rugam sa selectati motivul cererii din lista predefinita, iar in cazul cand nu va fi gasita una corespunzatoare se va indica motivul manual.

Sa se ia in considerare ca nu puteti trimite alta cerere de adeverinta pana cand cea precedenta nu va fi aprobata sau respinsa.



Odata depusa cererea aici, ea poate fi aprobata sau respinsa de secretarul corespunzator programului de studii. Aceasta va fi vizualizat aici si de asemenea veti primi un e-mail. Daca cererea este respinsa, veti primi si informatii cu privire la motivul respingerii.


Odata aprobata, adeverinta va fi semnata si veti primi o confirmare ca adeverinta poate fi ridiacata. In caz ca confirmarea nu a fost primita dupa un timp adecvat dupa aprobare, va trebui sa mergeti in persoana la secretariat sa ridicati adeverinta.

Selectati din lista predefinita sau indicati manual motivul cererii

REFUZA

TRIMITE CEREEA

By default student will see the list of already made the requests. Stuent is able to add new request only then the previous one was solved:



ADRIAN.BARBE@STUDENT.USV.RO

Numele, IT, Prenumele:

Barbe M. Adrian

Facultatea:

FIESC

Domeniu de studii:

Masterat

Program de studii:

SIC

An studii:

1

Statut financiar:

Fără taxă

CREAZA NOUA CERERE PENTRU ADEVERINTA

Data trimiterii	Numarul de inregistrare	Motivul	Starea	Secretarul responsabil
23.06.2023 03:51	125/1/FIESC/22.06.2023	Motivul1 este aici	Seminat	Dospinescu Laura
08.06.2023 16:48	124/1/FIESC/08.06.2023	Motivul1 este aici	Aprobat	Dospinescu Laura

Rânduri pe pagină: 10 1-2 din 2

[\(back to top\)](#)

Roadmap

- ☒ Add Basic functionality
- ☒ Add Multiple Faculties
- ☒ Add possibility to check the status of the certificate request
- ☒ Receive email when the status is changed
- ☒ Administration of the data from the UI
- ☐ Make mobile-ready web-version
- ☐ Multi-language Support

See the [open issues](#) for a full list of proposed features (and known issues).

[\(back to top\)](#)

Contributing

Contributions are what make the open source community such an amazing place to learn, inspire, and create. Any contributions you make are **greatly appreciated**.

If you have a suggestion that would make this better, please fork the repo and create a pull request. You can also simply open an issue with the tag "enhancement". Don't forget to give the project a star! Thanks again!

1. Fork the Project
2. Create your Feature Branch (`git checkout -b feature/AmazingFeature`)
3. Commit your Changes (`git commit -m 'Add some AmazingFeature'`)
4. Push to the Branch (`git push origin feature/AmazingFeature`)
5. Open a Pull Request

[\(back to top\)](#)

License

Distributed under the GNU GPL License. See `LICENSE.txt` for more information.

[\(back to top\)](#)

Contact

Adrian Barbe - [@adryanbarbe](#) - adryanbarbe@gmail.com

Project Link: <https://github.com/adrianbarbe/usv-stud-docs>

[\(back to top\)](#)

Acknowledgments

- [Choose an Open Source License](#)
- [Img Shields](#)
- [Vuetify](#)

[\(back to top\)](#)