Technical Development Manual

badgeGo
Quality Lab Pack

2023

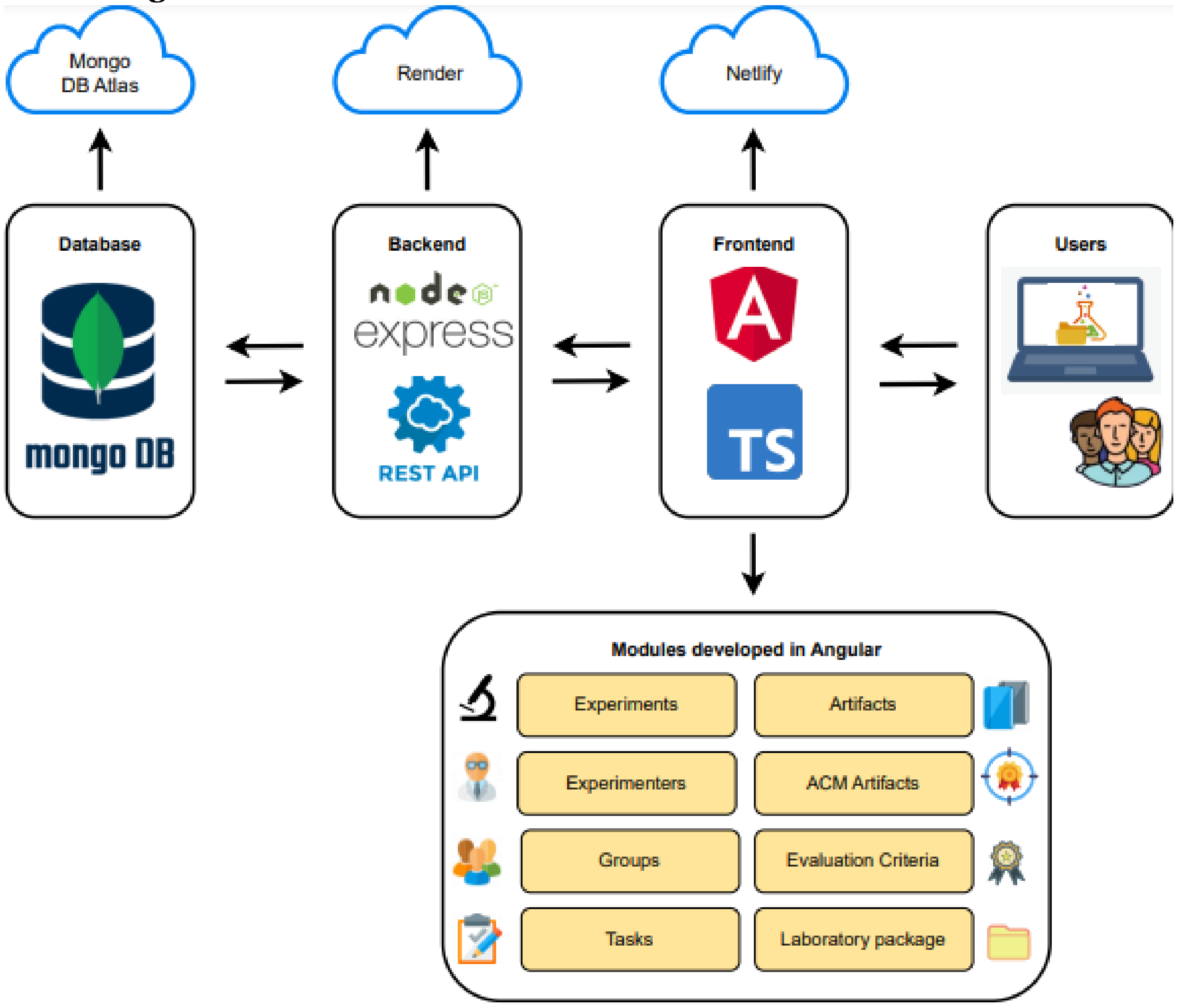
Introduction

This manual contains the technical documentation about the platform known as **badgeGo** within this manual you will find information about the architecture used, tools used during the development process of the platform. It is necessary to mention that this manual will take into account aspects of the current state of the platform in order to know the current operation of the application.

Index

Contents

Technological Architecture



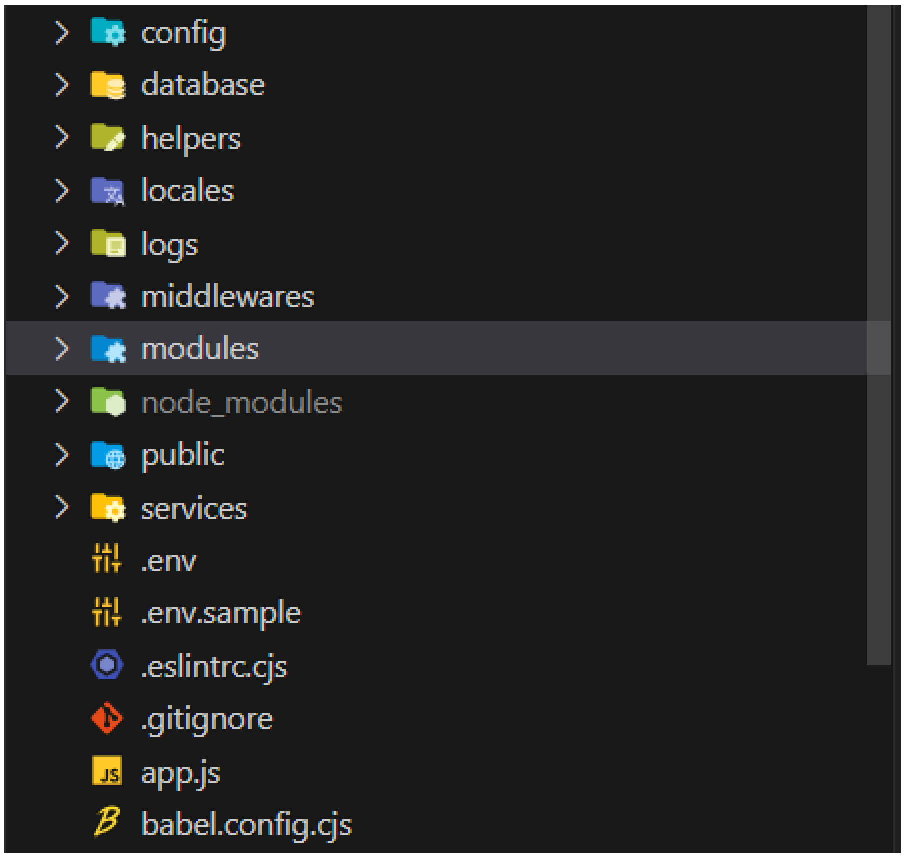
In the following illustration you can see the diagram of the technological architecture of the badgeGo platform. It is worth mentioning that this architecture has two main components: the backend and the frontend.

The backend component was developed using Node JS and Express JS which is a very popular JavaScript framework for the backend. It is necessary to mention that for the database we use MongoDB and Mongoose which is an ORM to connect Express JS with MongoDB.

The frontend component was developed with Angular version 11 and the TypeScript programming language was also used.

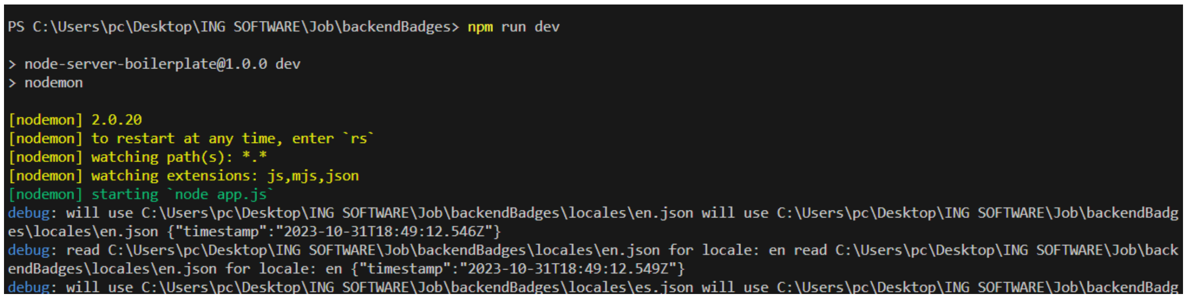
Backend Components

The developed backend has the following folder structure:

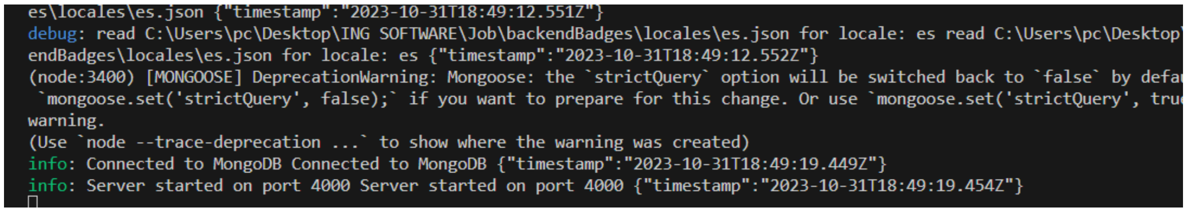


```
> config
> database
> helpers
> locales
> logs
> middlewares
> modules
> node_modules
> public
> services
  .env
  .env.sample
  .eslintrc.cjs
  .gitignore
  app.js
  babel.config.cjs
```

To run this project, it is necessary to use the command **npm run dev** as you can see in the following pictures.



```
PS C:\Users\pc\Desktop\ING SOFTWARE\Job\backendBadges> npm run dev

> node-server-boilerplate@1.0.0 dev
> nodemon

[nodemon] 2.0.20
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node app.js`
debug: will use C:\Users\pc\Desktop\ING SOFTWARE\Job\backendBadges\locales\en.json will use C:\Users\pc\Desktop\ING SOFTWARE\Job\backendBadg
es\locales\en.json {"timestamp":"2023-10-31T18:49:12.546Z"}
debug: read C:\Users\pc\Desktop\ING SOFTWARE\Job\backendBadges\locales\en.json for locale: en read C:\Users\pc\Desktop\ING SOFTWARE\Job\back
endBadges\locales\en.json for locale: en {"timestamp":"2023-10-31T18:49:12.549Z"}
debug: will use C:\Users\pc\Desktop\ING SOFTWARE\Job\backendBadges\locales\es.json will use C:\Users\pc\Desktop\ING SOFTWARE\Job\backendBadg
```



```
es\locales\es.json {"timestamp":"2023-10-31T18:49:12.551Z"}
debug: read C:\Users\pc\Desktop\ING SOFTWARE\Job\backendBadges\locales\es.json for locale: es read C:\Users\pc\Desktop
endBadges\locales\es.json for locale: es {"timestamp":"2023-10-31T18:49:12.552Z"}
(node:3400) [MONGOOSE] DeprecationWarning: Mongoose: the `strictQuery` option will be switched back to `false` by defa
 `mongoose.set('strictQuery', false);` if you want to prepare for this change. Or use `mongoose.set('strictQuery', tru
warning.
(Use `node --trace-deprecation ...` to show where the warning was created)
info: Connected to MongoDB Connected to MongoDB {"timestamp":"2023-10-31T18:49:19.449Z"}
info: Server started on port 4000 Server started on port 4000 {"timestamp":"2023-10-31T18:49:19.454Z"}
```

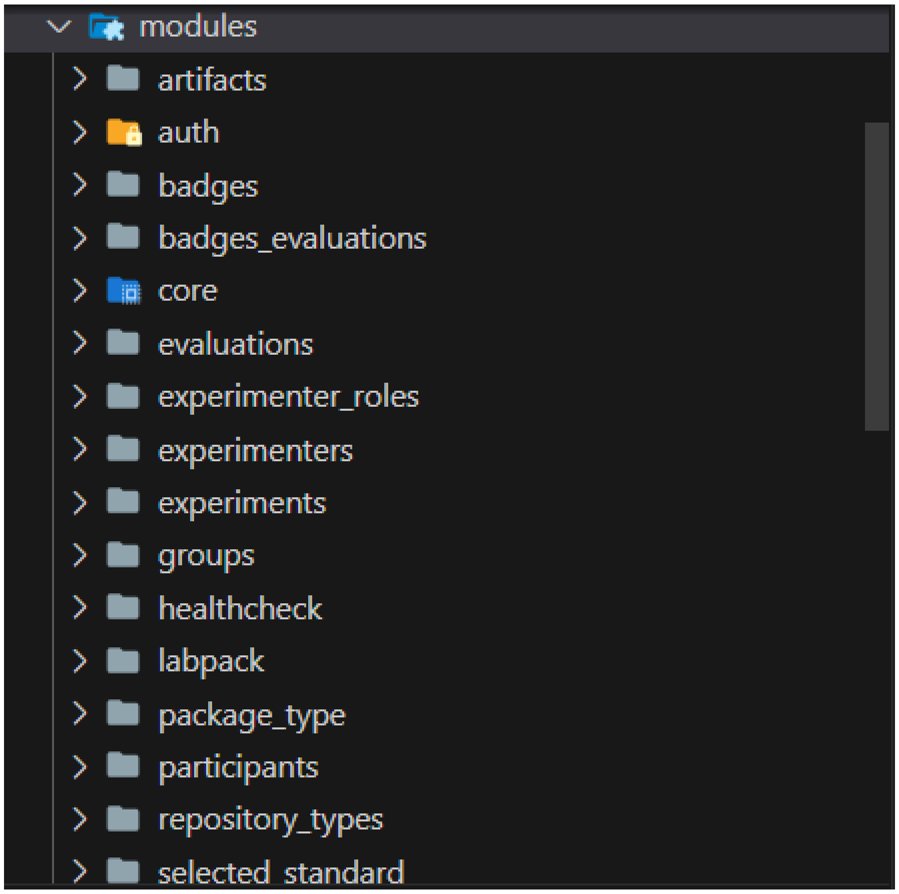
Inside the **config** folder are all the files related to the creation of the server, it is worth mentioning that in the moongose.js file is where the connection to the mongo DB database is made, likewise the file known as vars.js allow to assign values to the environment variables such as the port number or the url to the database.

The folder called **database** contains the seeds for the collections of the mongo database, seeds were created to quickly enter records in the database and to be able to perform tests to run the created seeds the **npm run seed** command should be used if necessary, however in the current state of the platform it is not necessary to use seeds for the moment.

The **locales** folder contains .json files that contain keywords to display messages to users depending on their language.

The **middlewares** folder contains middlewares for user authentication, error handling, language handling and the schemas used for mongoDB database collections.

The **modules** folder contains each one of the developed modules such as the experiments module, experimenter's module, tasks module, etc, the following is a screenshot of the contents of the modules folder.



```
modules
  artifacts
  auth
  badges
  badges_evaluations
  core
  evaluations
  experimenter_roles
  experimenters
  experiments
  groups
  healthcheck
  labpack
  package_type
  participants
  repository_types
  selected_standard
```

The core folder inside the modules folder allows you to establish a general structure for all modules in order to reuse code. Within this folder there are the files **model.js, query_model.js and request.js**

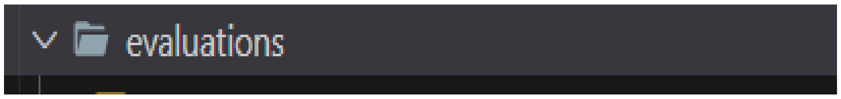
The **model.js** file allows the generic creation of each of the queries to be used, this file contains operations to search, delete, update documents in the collections.

The **query_model.js** file contains methods to perform queries generically with each of the database collections.

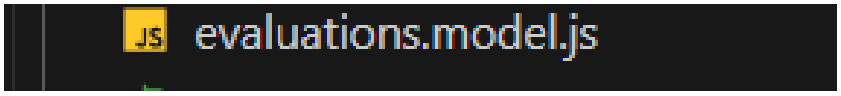
The **request.js** file is a file containing methods to perform post, put, delete and get requests.

Creating a new collection in the mongo database is easy, just follow the instructions below.

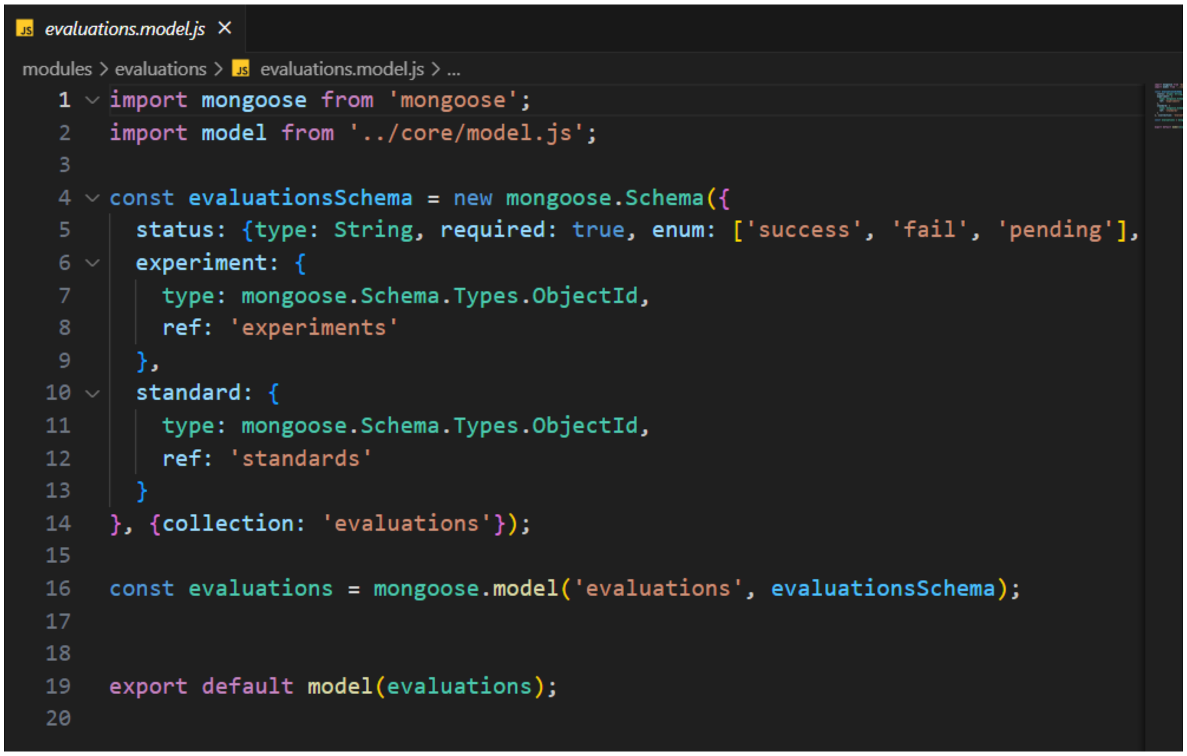
First inside the folder modules create a folder with the name.



Then you must create a file with the name of the collection followed by model.js as in the illustration.

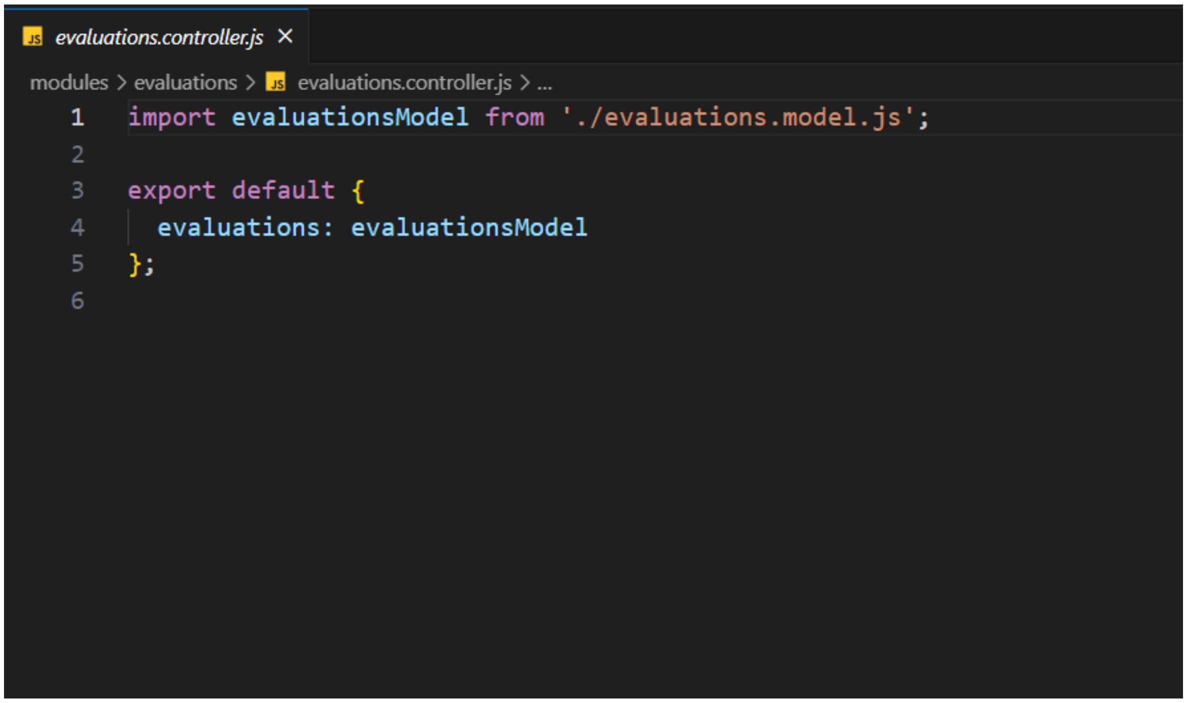


Within this file you must specify the model of the collection, i.e. detail the structure and each of the fields of the collection, as shown in the following illustration.



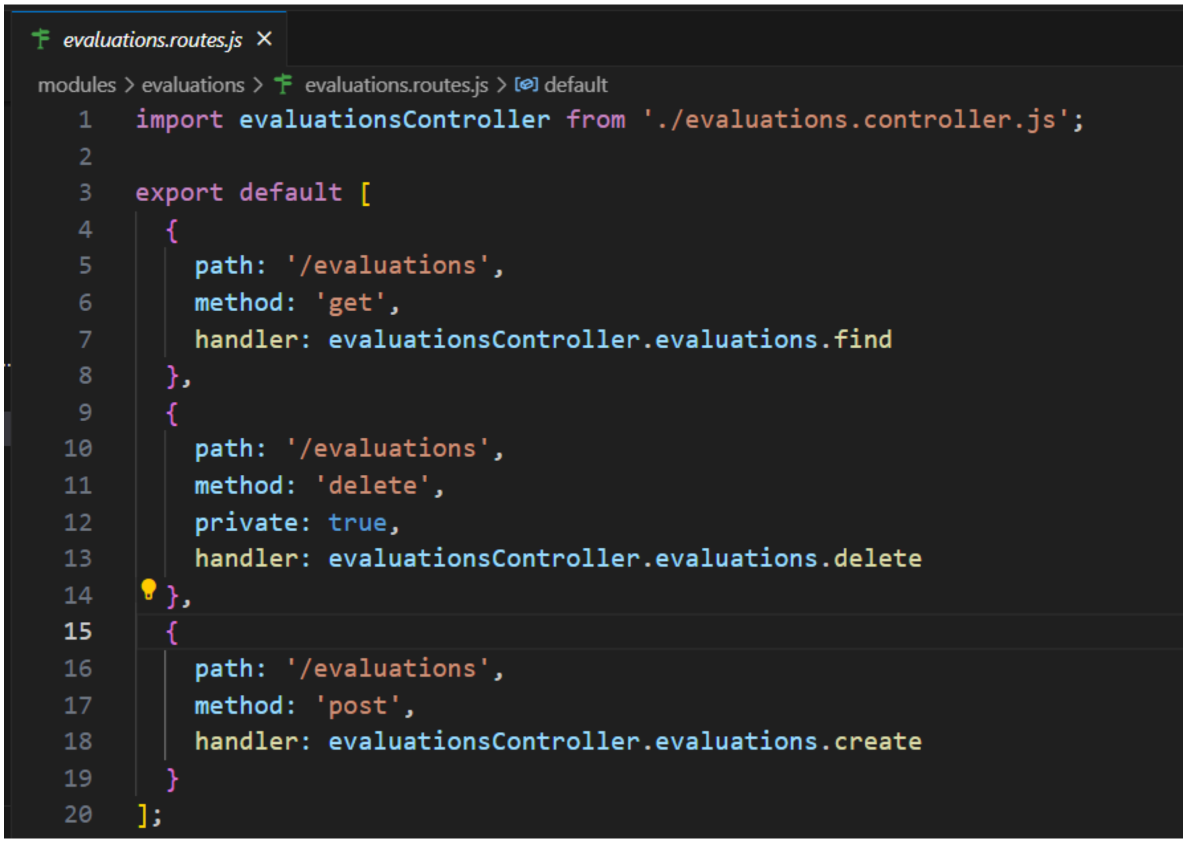
```
import mongoose from 'mongoose';
import model from '../core/model.js';

const evaluationsSchema = new mongoose.Schema({
  status: {type: String, required: true, enum: ['success', 'fail', 'pending'],
  experiment: {
    type: mongoose.Schema.Types.ObjectId,
    ref: 'experiments'
  },
  standard: {
    type: mongoose.Schema.Types.ObjectId,
    ref: 'standards'
  }
}, {collection: 'evaluations'});

const evaluations = mongoose.model('evaluations', evaluationsSchema);


export default model(evaluations);
```

Then it is necessary to create a file with the extension **.controller.js** and it is necessary to import the model that was created.


evaluations.controller.js
modules > evaluations > evaluations.controller.js > ...

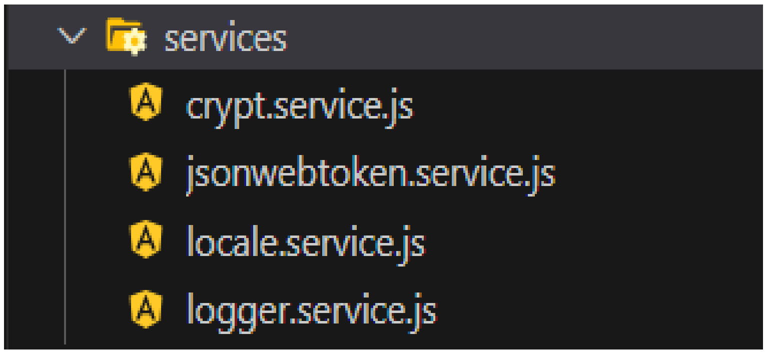
```javascript
import evaluationsModel from './evaluations.model.js';

export default {
  evaluations: evaluationsModel
};
```

Finally you must specify each of the routes to be used with the respective method in the case of a post request a create method will be used, in the case of a get request a find method will be used, in the case of a put request an update method will be used and for a delete request a delete method will be used, it is necessary to mention that the private field allows to establish if a route is going to be private or public and in the path field is where you can give a name to the route.



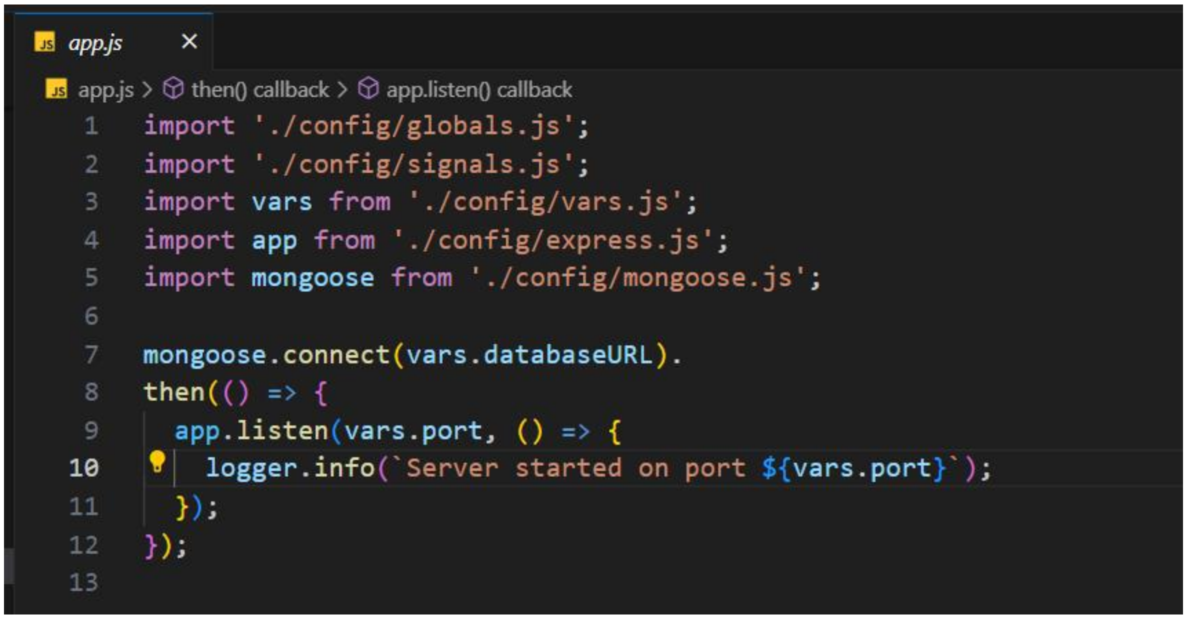
```
import evaluationsController from './evaluations.controller.js';

export default [
  {
    path: '/evaluations',
    method: 'get',
    handler: evaluationsController.evaluations.find
  },
  {
    path: '/evaluations',
    method: 'delete',
    private: true,
    handler: evaluationsController.evaluations.delete
  },
  {
    path: '/evaluations',
    method: 'post',
    handler: evaluationsController.evaluations.create
  }
];
```

The **services** folder contains files for password encryption and decryption, as well as files for jwt and language management.



```
services
    crypt.service.js
    jsonwebtoken.service.js
    locale.service.js
    logger.service.js
```

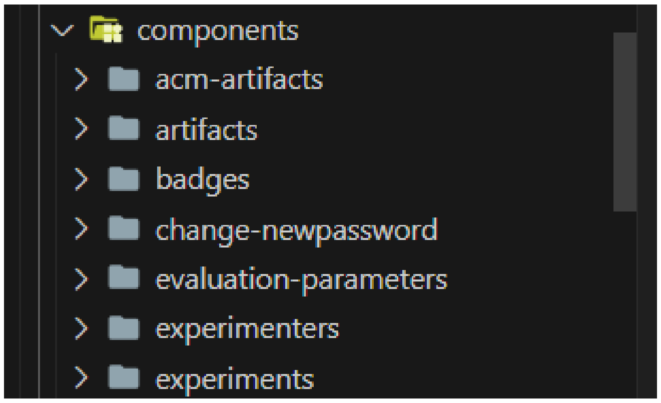
The app.js file is the main file that allows running the application created in the backend.



```
import './config/globals.js';
import './config/signals.js';
import vars from './config/vars.js';
import app from './config/express.js';
import mongoose from './config/mongoose.js';

mongoose.connect(vars.databaseURL).
then(() => {
  app.listen(vars.port, () => {
    logger.info(`Server started on port ${vars.port}`);
  });
});
```

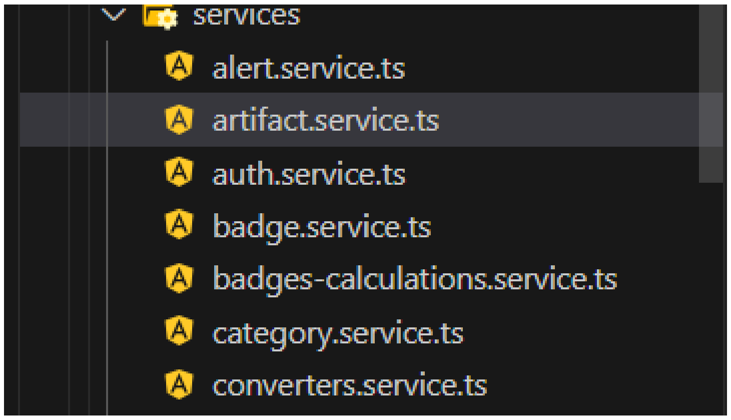
Frontend Components
To run the frontend part of the application, use the following command **ng serve.**

In the frontend there are a total of 8 components for each of the modules which are experimenter's module, experiments module, tasks module, groups module, artifacts module, etc, as shown in the following illustration.



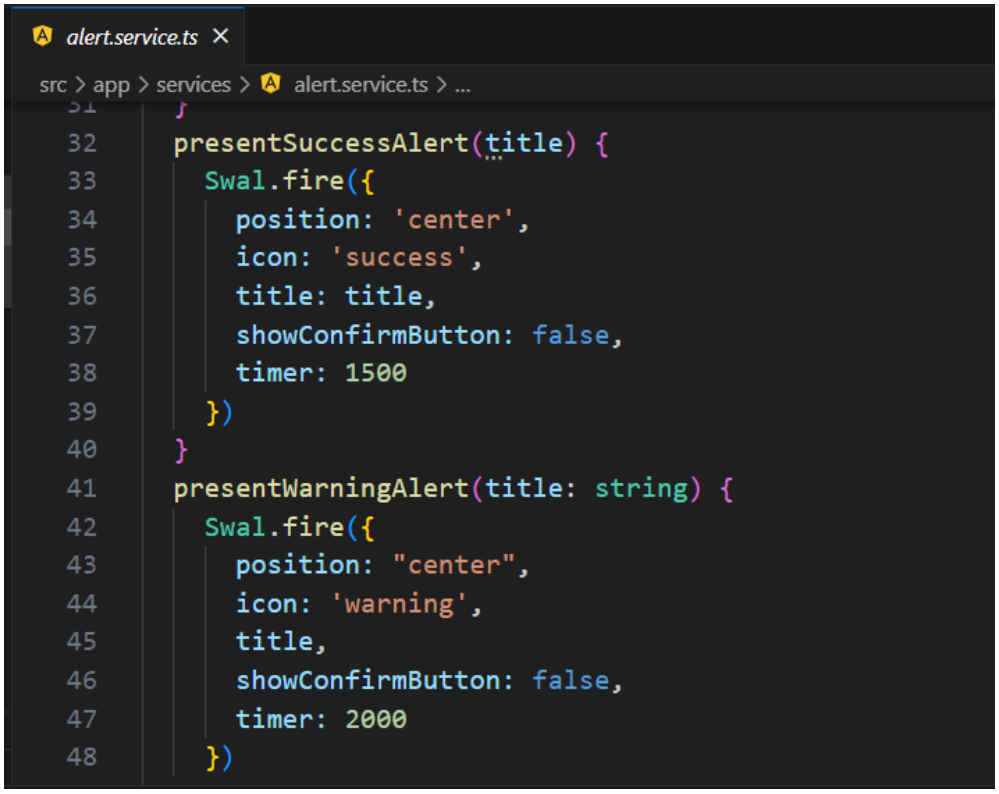
```
components
  acm-artifacts
  artifacts
  badges
  change-newpassword
  evaluation-parameters
  experimenters
  experiments
```

In the services folder were created each of the services to consume the API created in the backend.



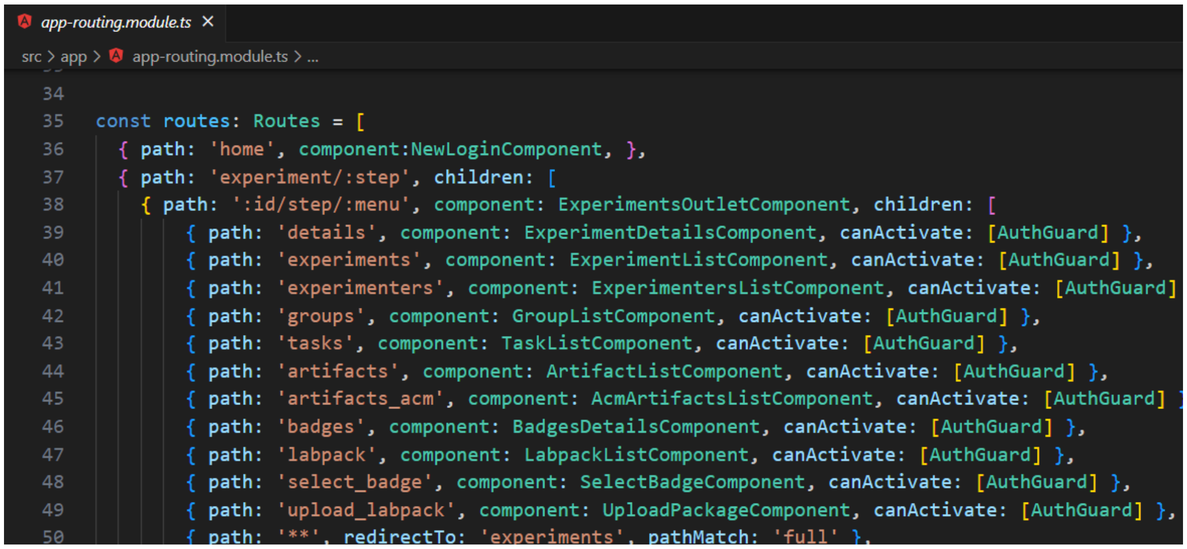
```
alert.service.ts
artifact.service.ts
auth.service.ts
badge.service.ts
badges-calculations.service.ts
category.service.ts
converters.service.ts
```

To display confirmation, information, warning and error messages, the sweet alert library was used. In the following file you can see how the library was implemented.



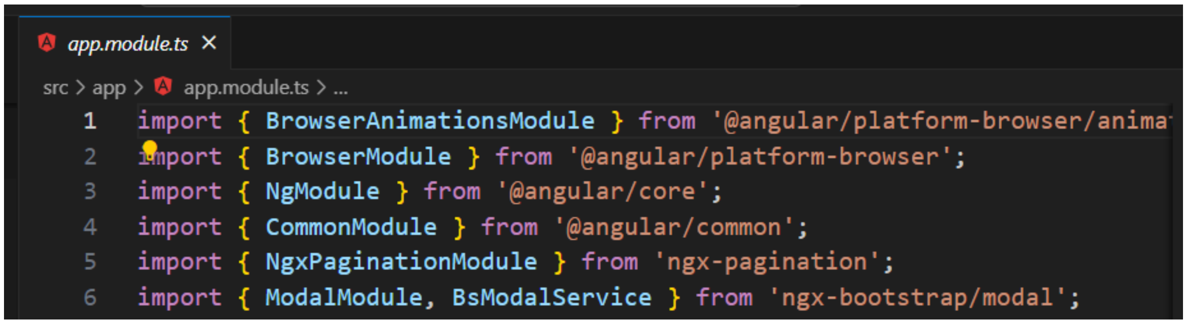
```
  }
  presentSuccessAlert(title) {
    Swal.fire({
      position: 'center',
      icon: 'success',
      title: title,
      showConfirmButton: false,
      timer: 1500
    })
  }
  presentWarningAlert(title: string) {
    Swal.fire({
      position: "center",
      icon: 'warning',
      title,
      showConfirmButton: false,
      timer: 2000
    })
```

In angular routes file is where you can add new routes or edit existing routes in the application is also possible to protect the routes with the use of guards. In the following image you can see an example of how routes are handled in Angular.



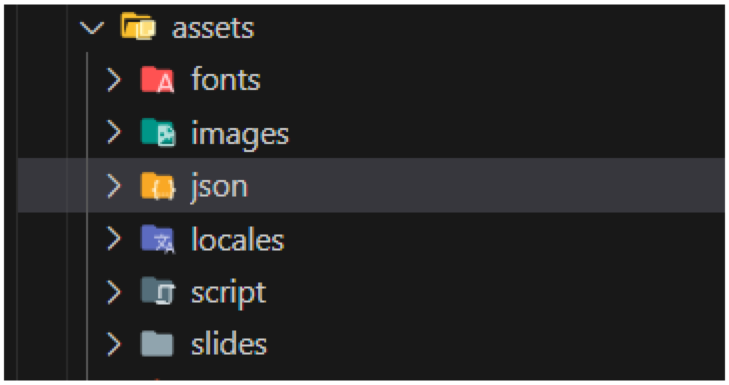
```
const routes: Routes = [
  { path: 'home', component:NewLoginComponent, },
  { path: 'experiment/:step', children: [
    { path: ':id/step/:menu', component: ExperimentsOutletComponent, children: [
        { path: 'details', component: ExperimentDetailsComponent, canActivate: [AuthGuard] },
        { path: 'experiments', component: ExperimentListComponent, canActivate: [AuthGuard] },
        { path: 'experimenters', component: ExperimentersListComponent, canActivate: [AuthGuard]
        { path: 'groups', component: GroupListComponent, canActivate: [AuthGuard] },
        { path: 'tasks', component: TaskListComponent, canActivate: [AuthGuard] },
        { path: 'artifacts', component: ArtifactListComponent, canActivate: [AuthGuard] },
        { path: 'artifacts_acm', component: AcmArtifactsListComponent, canActivate: [AuthGuard] }
        { path: 'badges', component: BadgesDetailsComponent, canActivate: [AuthGuard] },
        { path: 'labpack', component: LabpackListComponent, canActivate: [AuthGuard] },
        { path: 'select_badge', component: SelectBadgeComponent, canActivate: [AuthGuard] },
        { path: 'upload_labpack', component: UploadPackageComponent, canActivate: [AuthGuard] },
        { path: '**', redirectTo: 'experiments', pathMatch: 'full' },
```

In the **app.module.ts** file is where you can see each of the packages or dependencies that will be used for the project developed in Angular.

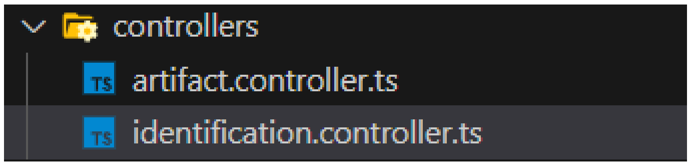


```
import { BrowserAnimationsModule } from '@angular/platform-browser/anima
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { CommonModule } from '@angular/common';
import { NgxPaginationModule } from 'ngx-pagination';
import { ModalModule, BsModalService } from 'ngx-bootstrap/modal';
```

Inside the assets folder there is a directory called locals where the **.json** files for the English and Spanish language management are configured.



In the controllers folder are the files to manage the upload and storage of artifacts in Firebase this is done through the **artifact.controller** file and **identification.controller** allows to validate if a DNI is valid.


controllers
artifact.controller.ts
identification.controller.ts

The jspdf library was used to generate the pdf files, if you want more information, you can use the following link.

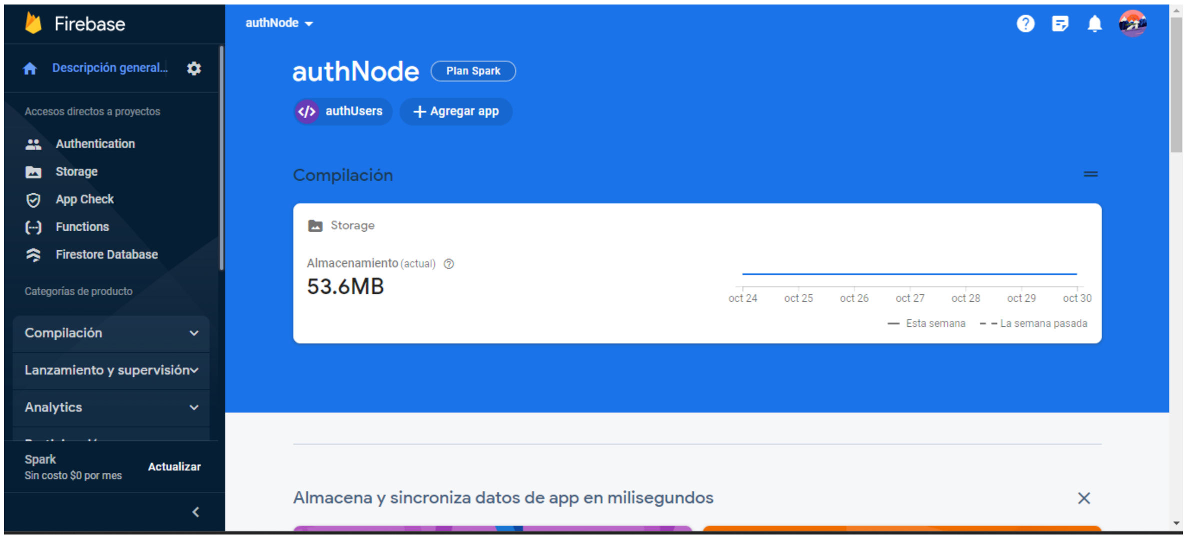
https://artskydj.github.io/jsPDF/docs/jsPDF.html

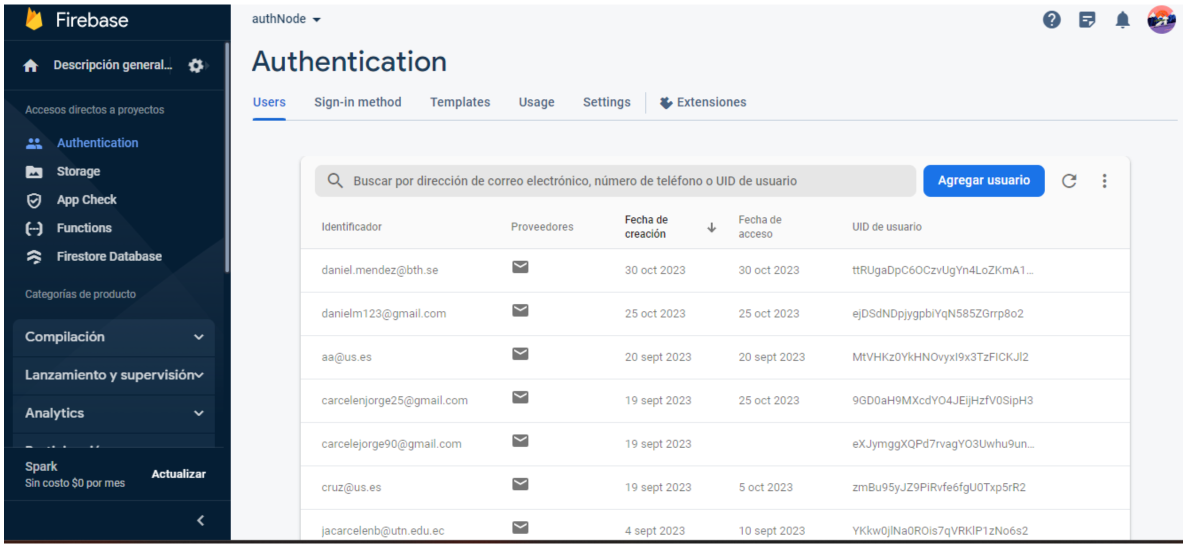
Firebase Configuration

The firebase platform was mainly used for its user authentication and file storage services.

User Authentication

In the following illustrations you can see the screenshots of the firebase user management console. This service was integrated to the frontend developed in angular through a service key and with the use of angular fire.

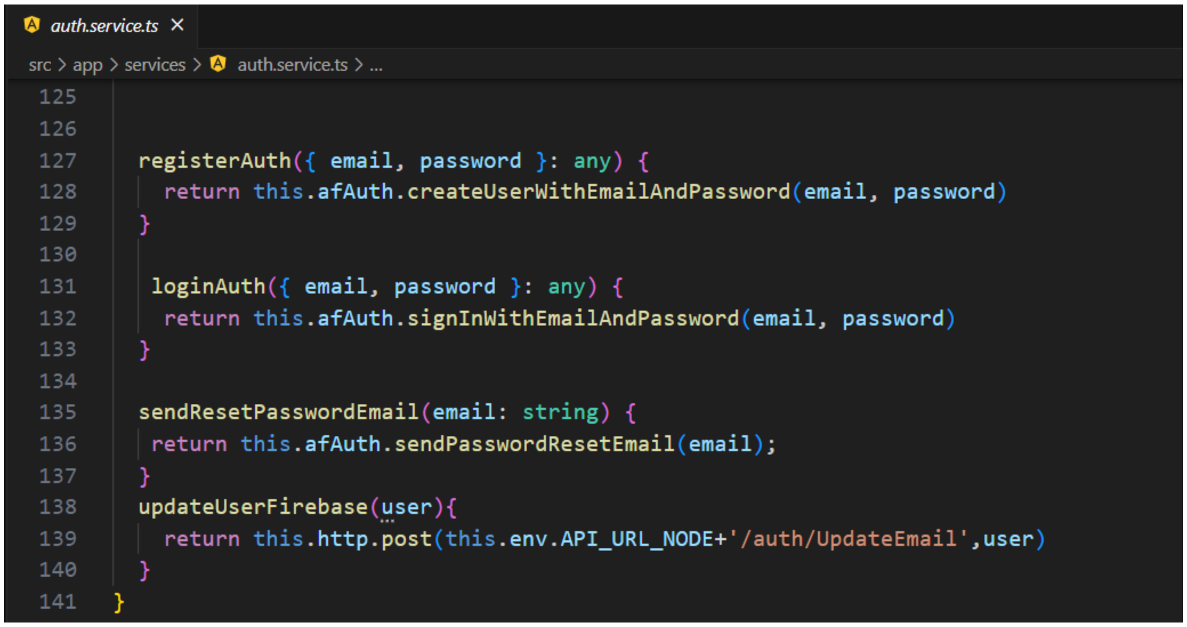




Authentication

Identificador	Proveedores	Fecha de creación	Fecha de acceso	UID de usuario
daniel.mendez@bth.se	✉	30 oct 2023	30 oct 2023	ttRUgaDpC6OCzvUgYn4LoZKmA1...
danielm123@gmail.com	✉	25 oct 2023	25 oct 2023	ejDSdNDpjygpbiYqN585ZGrrp8o2
aa@us.es	✉	20 sept 2023	20 sept 2023	MtVHKz0YkHNOvyxI9x3TzFICKJl2
carcelenjorge25@gmail.com	✉	19 sept 2023	25 oct 2023	9GD0aH9MXcdYO4JEijHzfV0SipH3
carcelejorge90@gmail.com	✉	19 sept 2023		eXJymggXQPd7rvagYO3Uwhu9un...
cruz@us.es	✉	19 sept 2023	5 oct 2023	zmBu95yJZ9PiRvfe6fgU0Txp5rR2
jacarcelenb@utn.edu.ec	✉	4 sept 2023	10 sept 2023	YKkw0jlNa0ROis7qVRKlP1zNo6s2

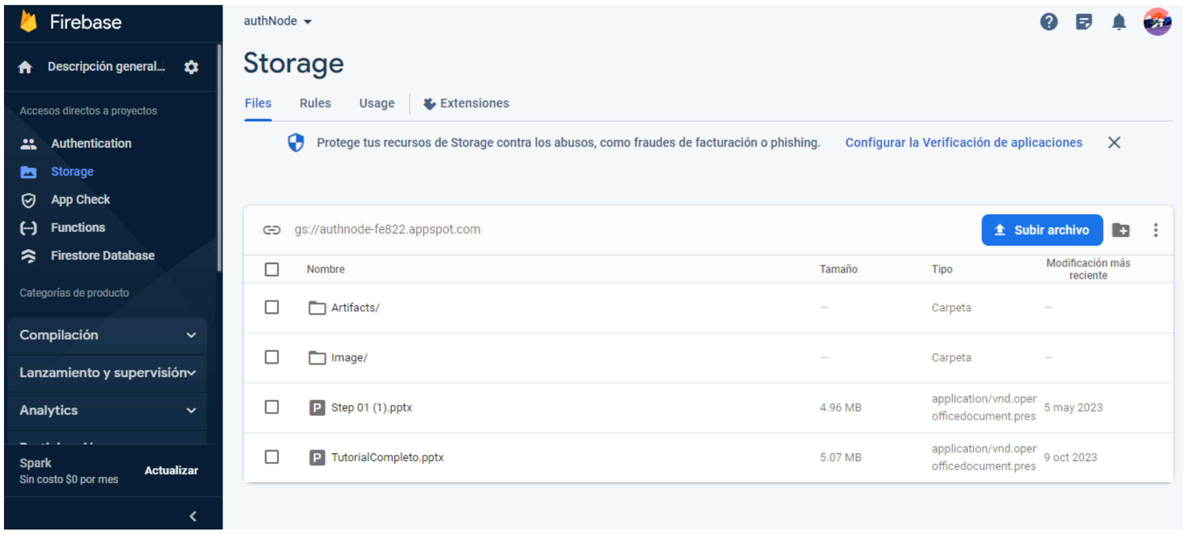
In this section you can see the code that was used to integrate the firebase user authentication service with Angular.



```


  registerAuth({ email, password }: any) {
    return this.afAuth.createUserWithEmailAndPassword(email, password)
  }

   loginAuth({ email, password }: any) {
    return this.afAuth.signInWithEmailAndPassword(email, password)
  }

  sendResetPasswordEmail(email: string) {
   return this.afAuth.sendPasswordResetEmail(email);
  }
  updateUserFirebase(user){
    return this.http.post(this.env.API_URL_NODE+'/auth/UpdateEmail',user)
  }
}
```

File Storage

For file storage, the firebase storage service was used and later integrated with Angular to upload and download files.


Firebase
Descripción general...
Accesos directos a proyectos
Authentication
Storage
App Check
Functions
Firestore Database
Categorías de producto
Compilación
Lanzamiento y supervisión
Analytics
Spark
Sin costo $0 por mes
Actualizar

authNode

Storage

Files Rules Usage Extensiones

Protege tus recursos de Storage contra los abusos, como fraudes de facturación o phishing. Configurar la Verificación de aplicaciones

gs://authnode-fe822.appspot.com

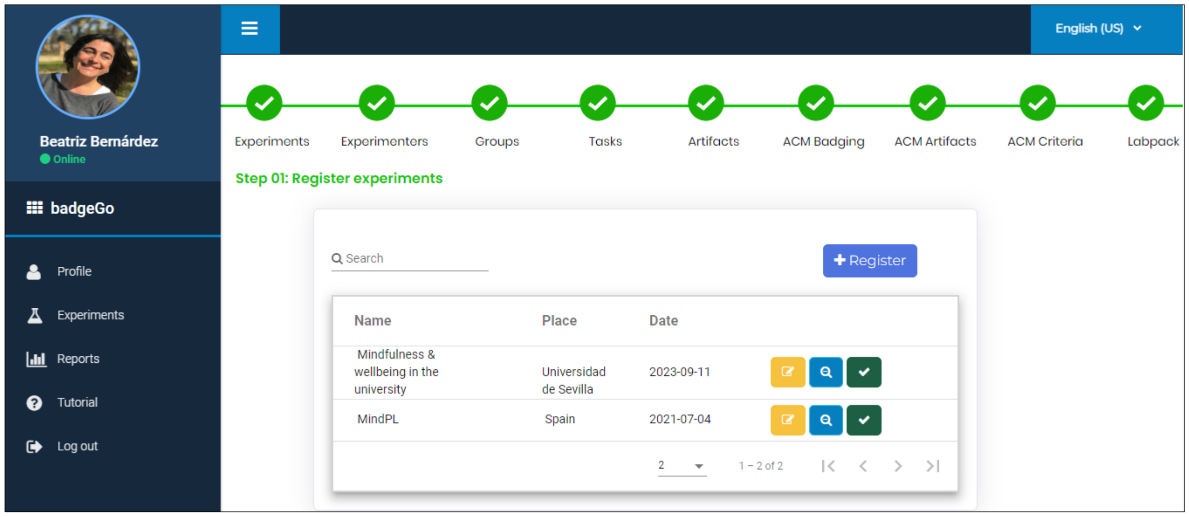
Subir archivo

Nombre	Tamaño	Tipo	Modificación más reciente
Artifacts/	—	Carpeta	—
Image/	—	Carpeta	—
Step 01 (1).pptx	4.96 MB	application/vnd.oper officedocument.pres	5 may 2023
TutorialCompleto.pptx	5.07 MB	application/vnd.oper officedocument.pres	9 oct 2023

Developed modules on Angular

Experiment module

The experiments module allows you to record, update and view experiment information.


Beatriz Bernárdez
Online
badgeGo
Profile
Experiments
Reports
Tutorial
Log out

English (US)

Experiments — Experimenters — Groups — Tasks — Artifacts — ACM Badging — ACM Artifacts — ACM Criteria — Labpack

Step 01: Register experiments

Search

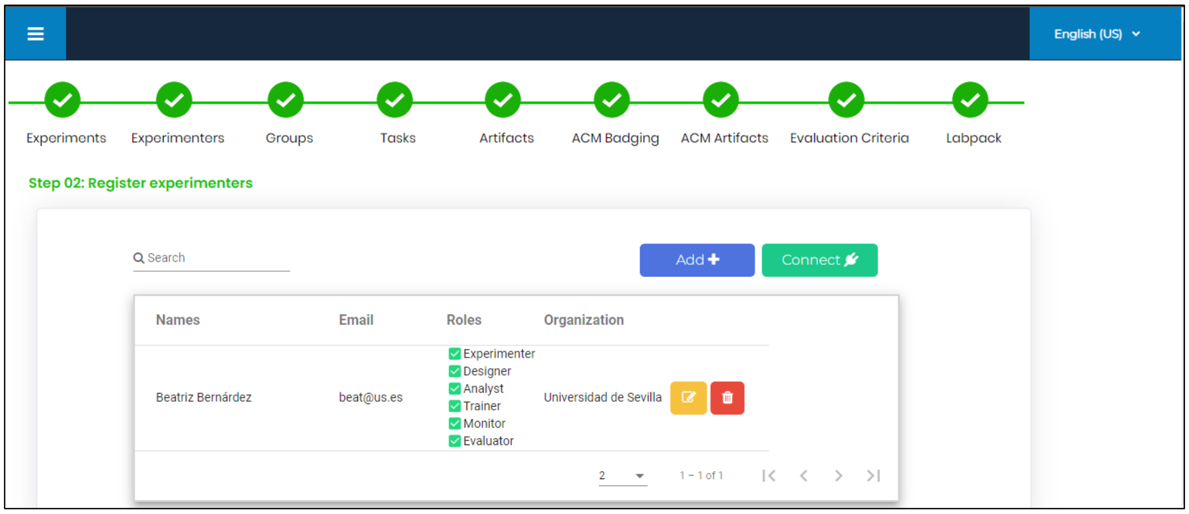
Register

Name	Place	Date
Mindfulness & wellbeing in the university	Universidad de Sevilla	2023-09-11
MindPL	Spain	2021-07-04

2 1 – 2 of 2

Experimenters' module

The experimenter module allows you to register, update, connect and remove experimenters from an experiment.



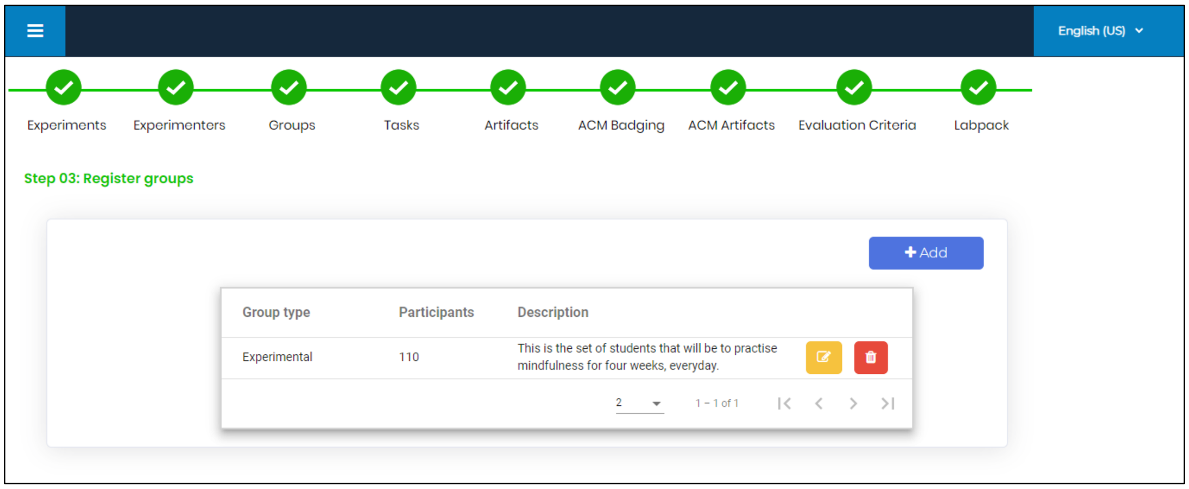
Step 02: Register experimenters

Names	Email	Roles	Organization
Beatriz Bernárdez	beat@us.es	✓ Experimenter ✓ Designer ✓ Analyst ✓ Trainer ✓ Monitor ✓ Evaluator	Universidad de Sevilla

2 1 – 1 of 1

Group's module

The groups module allows you to register, update and delete groups.



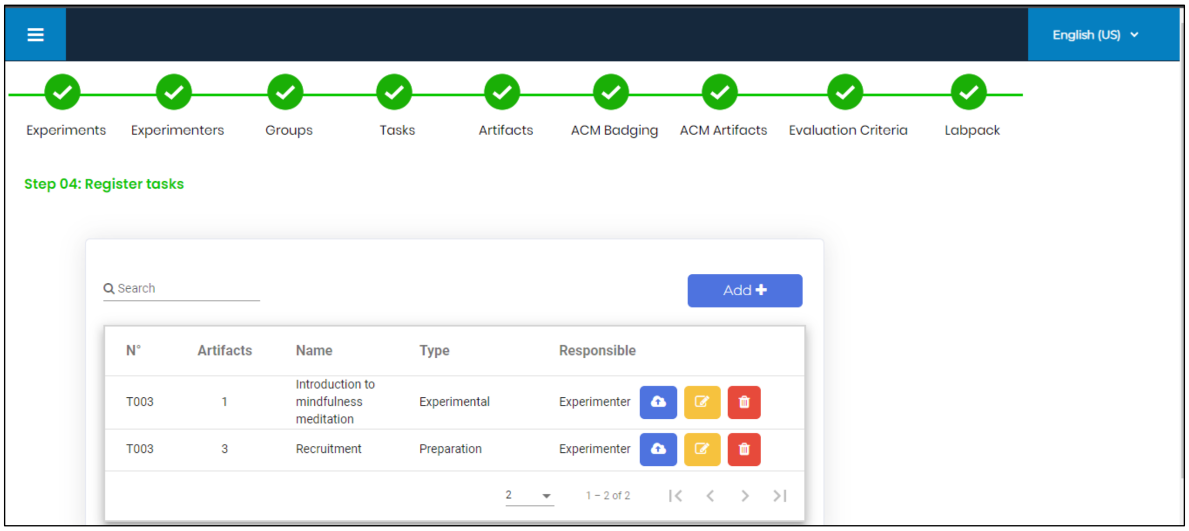
Step 03: Register groups

Group type	Participants	Description
Experimental	110	This is the set of students that will be to practise mindfulness for four weeks, everyday.

2 1 – 1 of 1

Task's module

The task module allows you to register, update, delete tasks and upload artifacts for tasks.

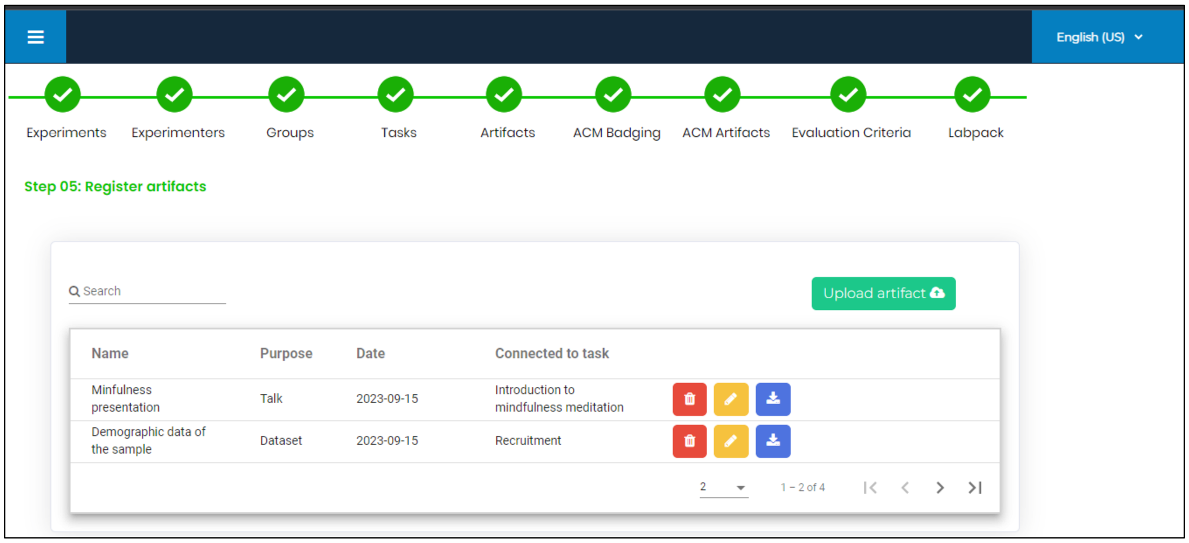


Step 04: Register tasks

N°	Artifacts	Name	Type	Responsible
T003	1	Introduction to mindfulness meditation	Experimental	Experimenter
T003	3	Recruitment	Preparation	Experimenter

Artifact's module

The artifacts module allows you to register, update and delete artifacts.

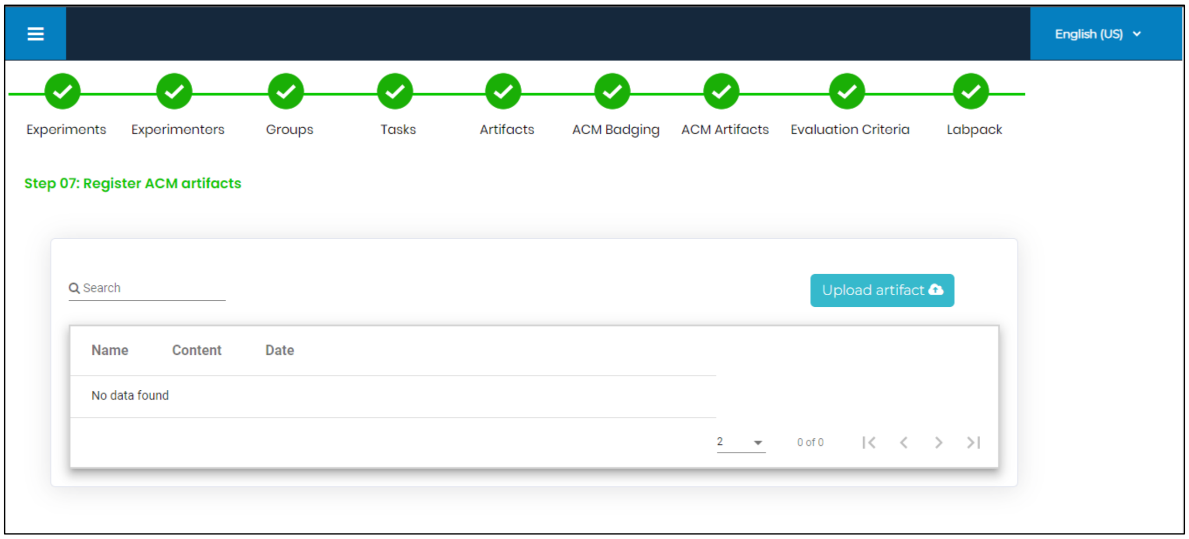


Step 05: Register artifacts

Name	Purpose	Date	Connected to task
Minfulness presentation	Talk	2023-09-15	Introduction to mindfulness meditation
Demographic data of the sample	Dataset	2023-09-15	Recruitment

ACM Artifacts module

The ACM artifacts module allows to register, update, delete artifacts from the ACM.



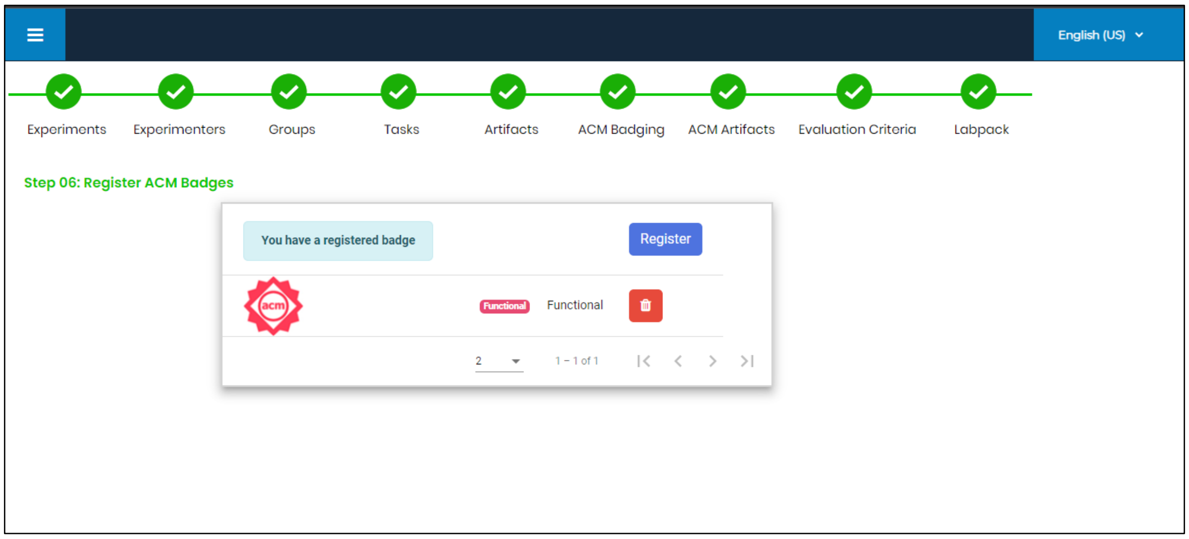
Step 07: Register ACM artifacts

Name	Content	Date
No data found		

2 0 of 0

ACM Badging module

The ACM Badging Module allows you to register the badges you wish to obtain for your experiment.



Step 06: Register ACM Badges

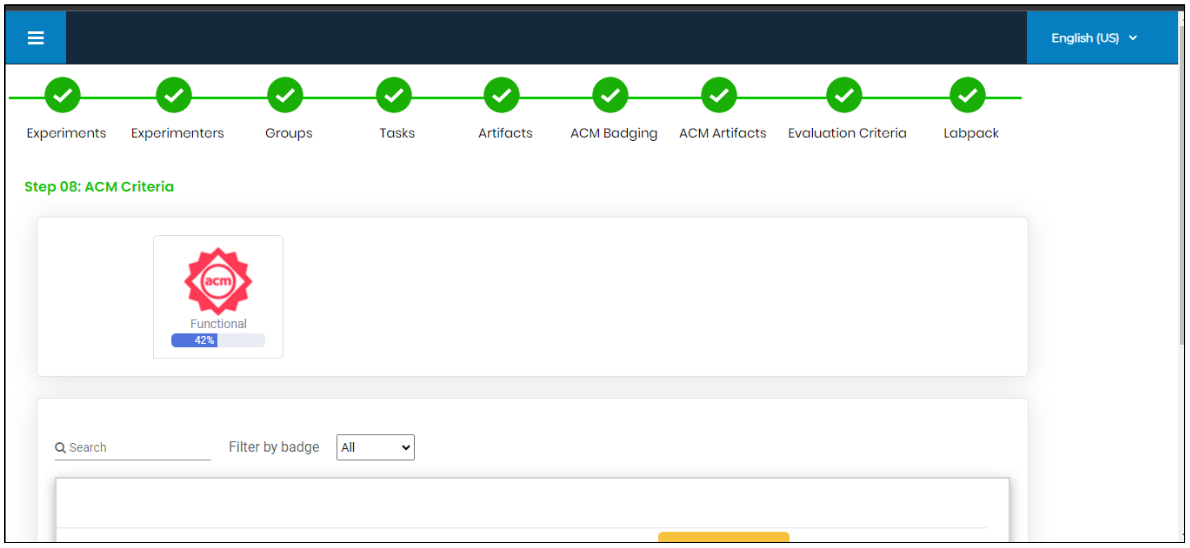
You have a registered badge

Functional

2 1 – 1 of 1

Evaluation Criteria

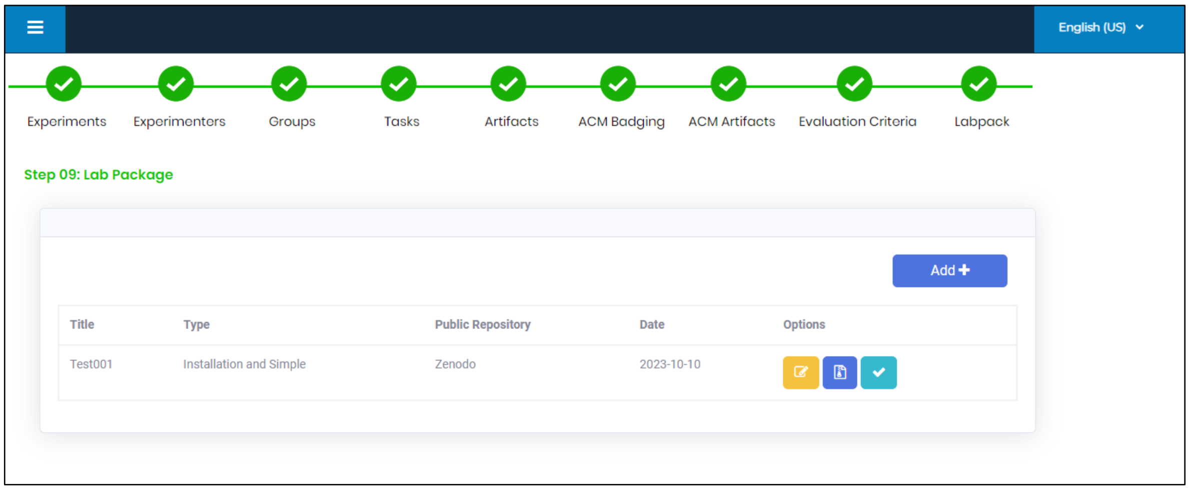
This module allows users to complete the parameters to obtain the badge they want.



Step 08: ACM Criteria

Labpack module

The lab package module allows you to register, update and generate the lab package.

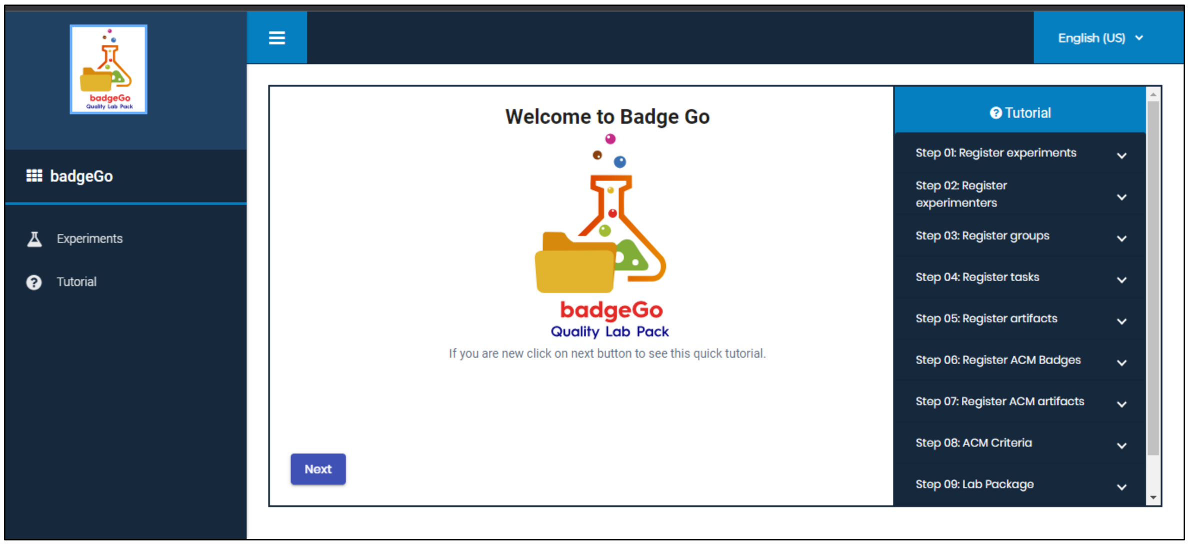


Step 09: Lab Package

Title	Type	Public Repository	Date	Options
Test001	Installation and Simple	Zenodo	2023-10-10	

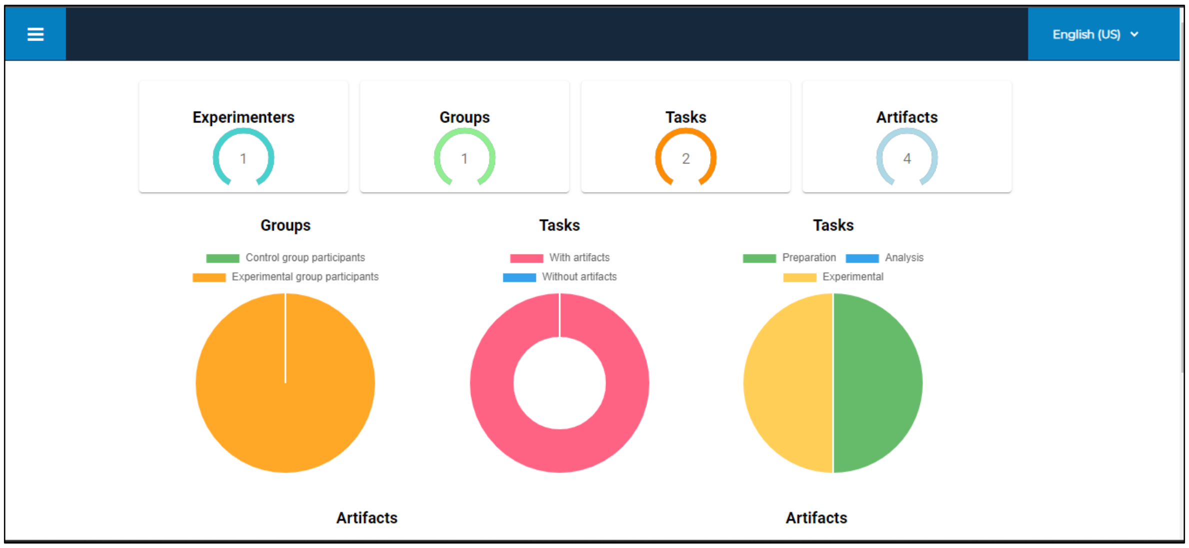
Tutorial Section

This section contains help for users.


badgeGo
Experiments
Tutorial
English (US)
Welcome to Badge Go
badgeGo
Quality Lab Pack
If you are new click on next button to see this quick tutorial.
Next
Tutorial
Step 01: Register experiments
Step 02: Register experimenters
Step 03: Register groups
Step 04: Register tasks
Step 05: Register artifacts
Step 06: Register ACM Badges
Step 07: Register ACM artifacts
Step 08: ACM Criteria
Step 09: Lab Package

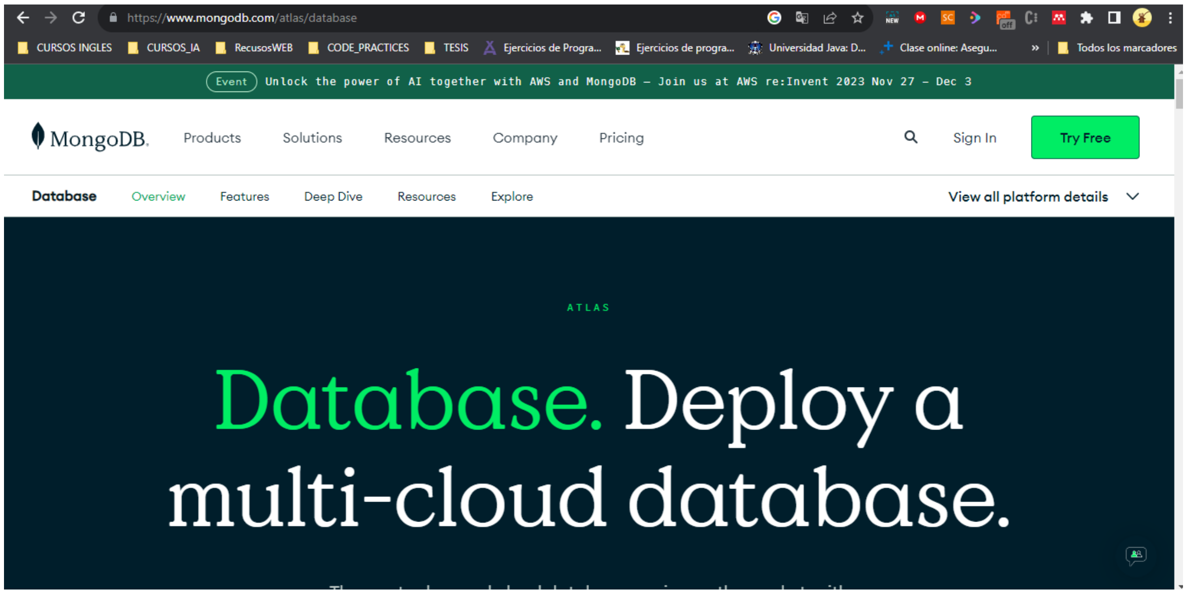
Reports Section

This section shows reports on the status of the experiment.

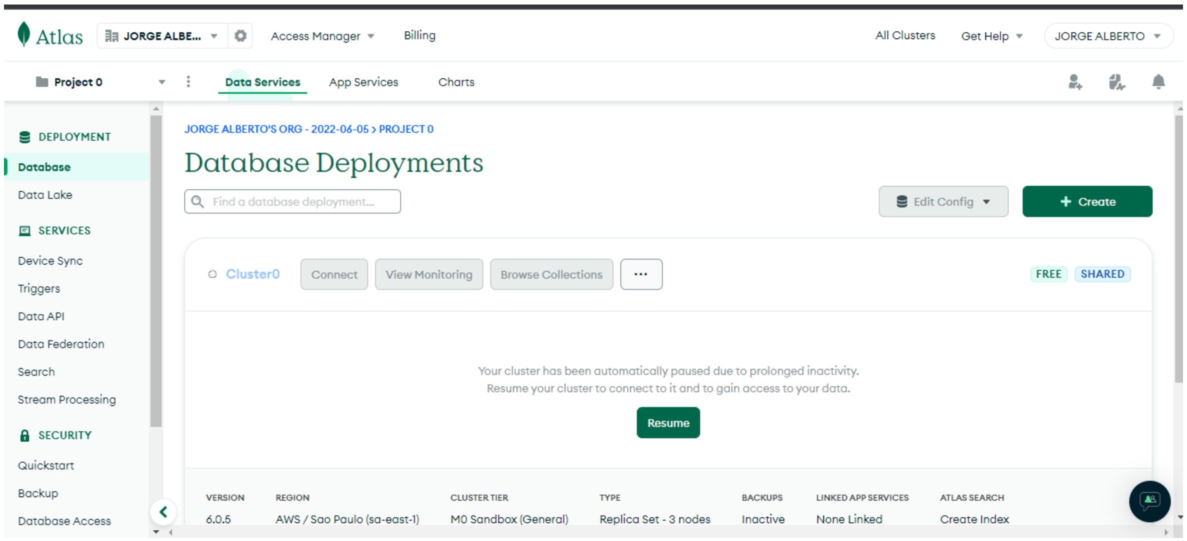


Deploy database on MongoDB Atlas

1. Go to MongoDB Atlas and Sign up.


https://www.mongodb.com/atlas/database
Event Unlock the power of AI together with AWS and MongoDB – Join us at AWS re:Invent 2023 Nov 27 – Dec 3
MongoDB. Products Solutions Resources Company Pricing Sign In Try Free
Database Overview Features Deep Dive Resources Explore View all platform details
ATLAS
Database. Deploy a multi-cloud database.

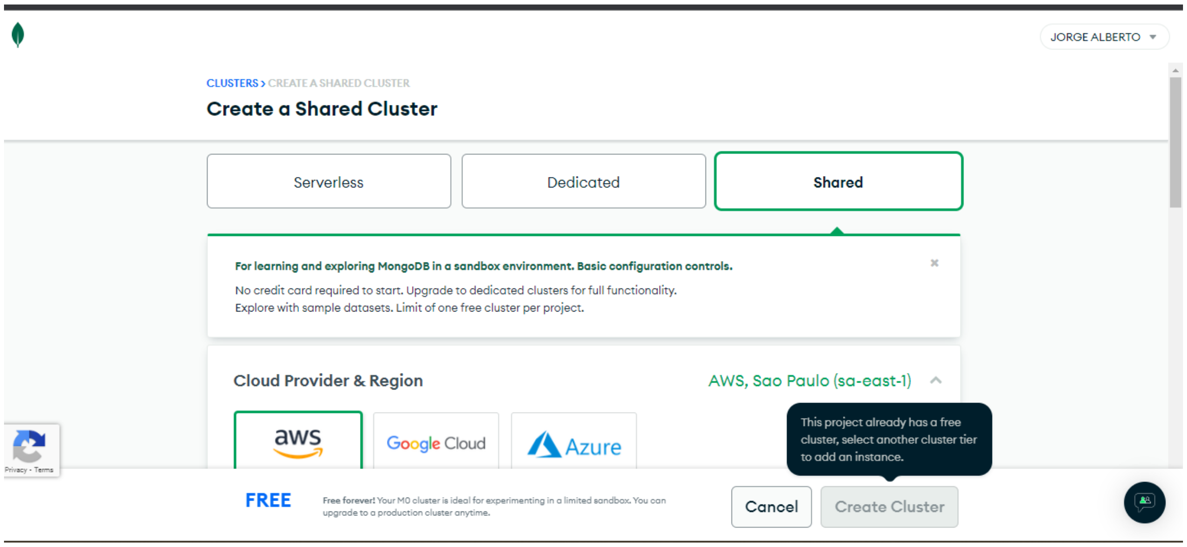
2. Once you are on the main screen click on the create button.


Atlas JORGE ALBE... Access Manager Billing All Clusters Get Help JORGE ALBERTO
Project 0 Data Services App Services Charts
DEPLOYMENT Database Data Lake
SERVICES Device Sync Triggers Data API Data Federation Search Stream Processing
SECURITY Quickstart Backup Database Access
JORGE ALBERTO'S ORG - 2022-06-05 > PROJECT 0
Database Deployments
Find a database deployment...
Edit Config
+ Create
Cluster0 Connect View Monitoring Browse Collections ... FREE SHARED
Resume

Your cluster has been automatically paused due to prolonged inactivity.
Resume your cluster to connect to it and to gain access to your data.

VERSION	REGION	CLUSTER TIER	TYPE	BACKUPS	LINKED APP SERVICES	ATLAS SEARCH
6.0.5	AWS / Sao Paulo (sa-east-1)	M0 Sandbox (General)	Replica Set - 3 nodes	Inactive	None Linked	Create Index

3. Then select the shared option and enter each of the fields and click on create cluster.


CLUSTERS > CREATE A SHARED CLUSTER

Create a Shared Cluster

For learning and exploring MongoDB in a sandbox environment. Basic configuration controls.

No credit card required to start. Upgrade to dedicated clusters for full functionality. Explore with sample datasets. Limit of one free cluster per project.

Cloud Provider & Region: AWS, Sao Paulo (sa-east-1)

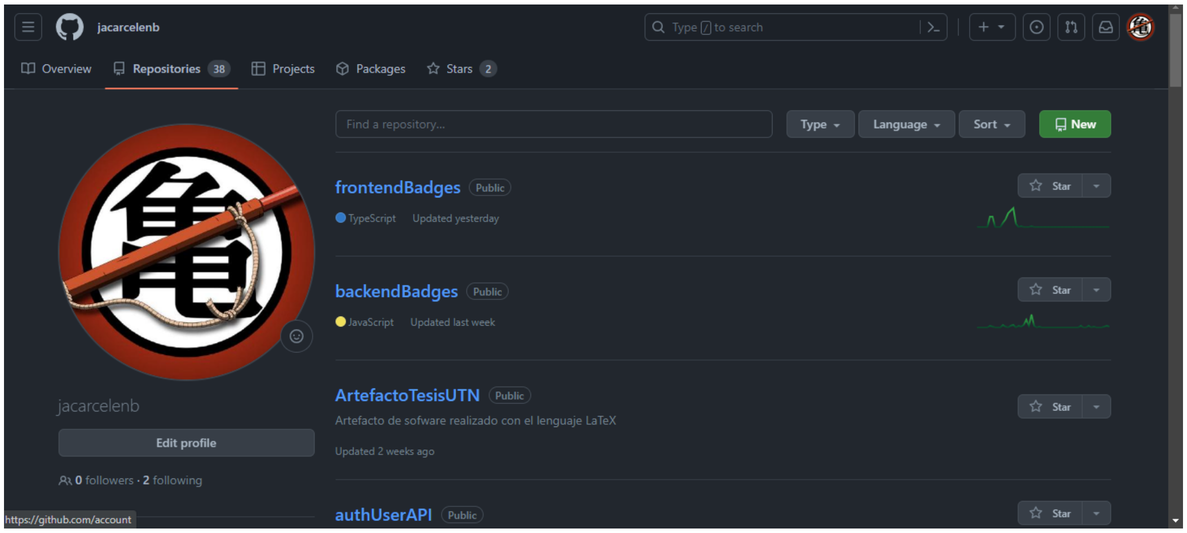
This project already has a free cluster, select another cluster tier to add an instance.

FREE — Free forever! Your M0 cluster is ideal for experimenting in a limited sandbox. You can upgrade to a production cluster anytime.

Deploy backend project on Render

To deploy the backend in Render you must follow the instructions below:

1. Check if you have access to the GitHub repository as shown in the following image.



jacarcelenb

0 followers · 2 following

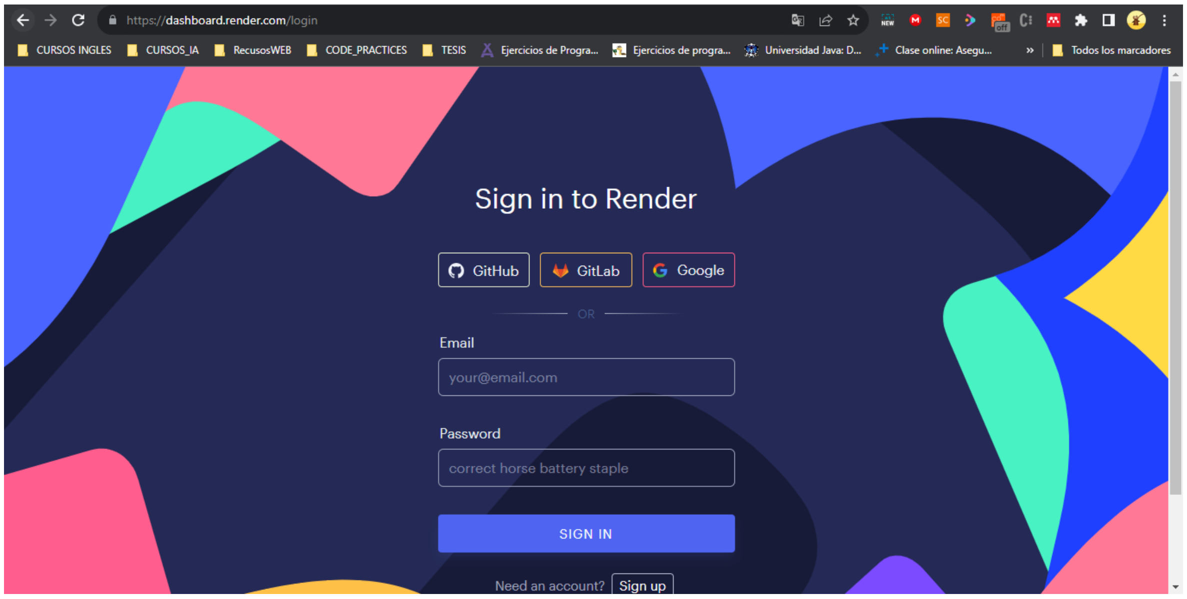
frontendBadges (Public) — TypeScript, Updated yesterday

backendBadges (Public) — JavaScript, Updated last week

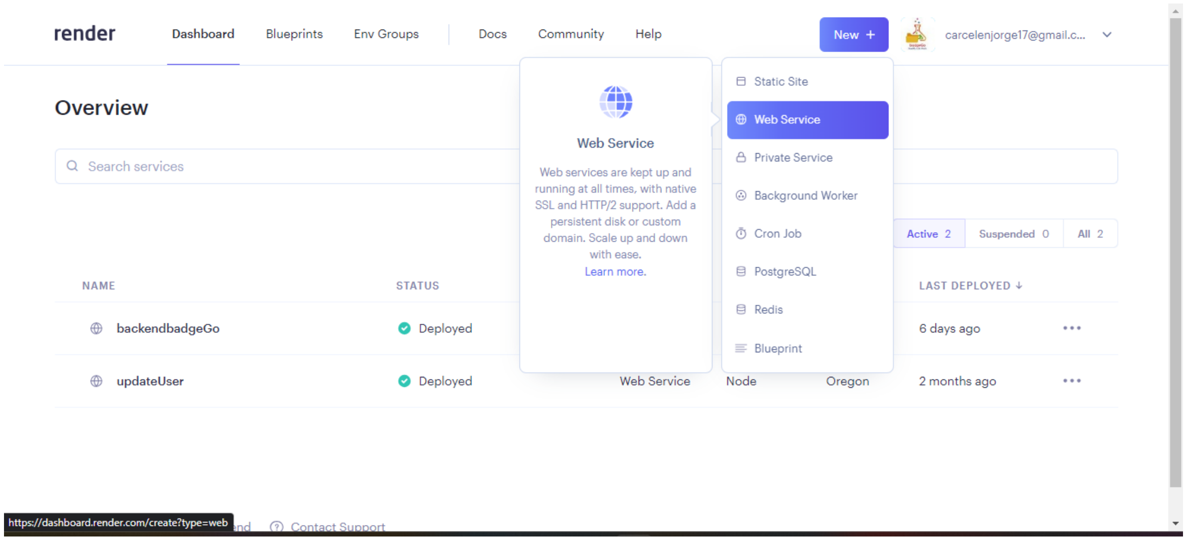
ArtefactoTesisUTN (Public) — Artefacto de sofware realizado con el lenguaje LaTeX — Updated 2 weeks ago

authUserAPI (Public)

2. Go to this page https://dashboard.render.com/login and sign up.



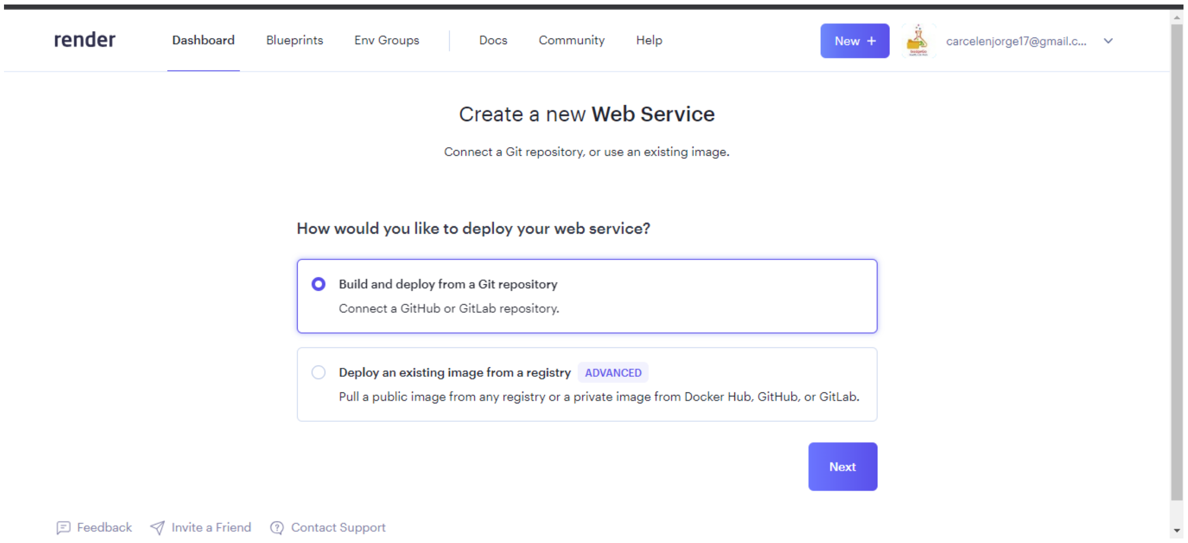
3. Click the new button and choose web services option.



Web services are kept up and running at all times, with native SSL and HTTP/2 support. Add a persistent disk or custom domain. Scale up and down with ease.

NAME	STATUS				LAST DEPLOYED ↓
backendbadgeGo	Deployed				6 days ago
updateUser	Deployed	Web Service	Node	Oregon	2 months ago

4. Choose the first option to use the repository on GitHub and click on Next.



render Dashboard Blueprints Env Groups Docs Community Help New + carcelenjorge17@gmail.c...

Create a new **Web Service**

Connect a Git repository, or use an existing image.

How would you like to deploy your web service?

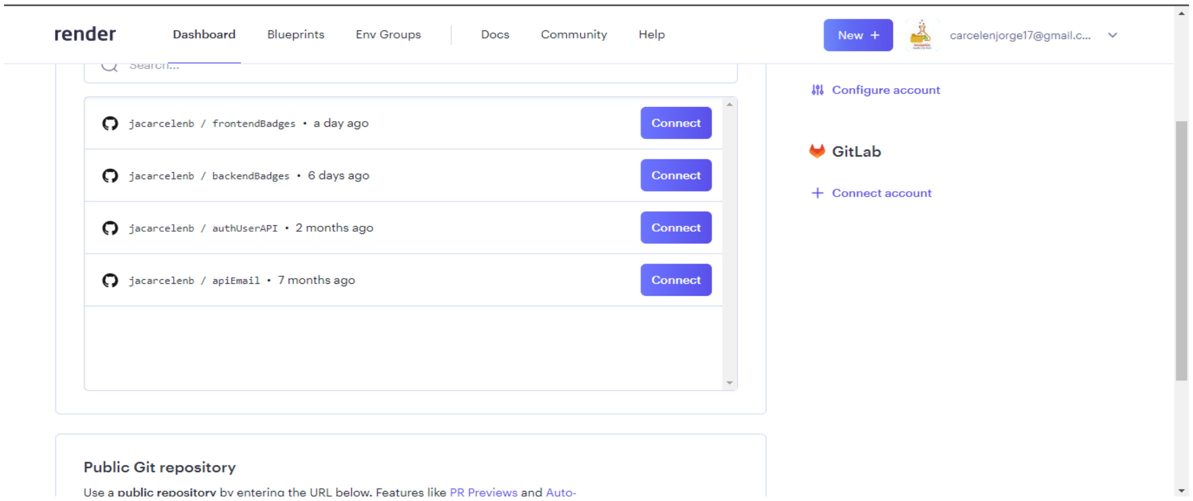
● Build and deploy from a Git repository
Connect a GitHub or GitLab repository.

○ Deploy an existing image from a registry ADVANCED
Pull a public image from any registry or a private image from Docker Hub, GitHub, or GitLab.

Next

Feedback Invite a Friend Contact Support

5. Select the repository



render Dashboard Blueprints Env Groups Docs Community Help New + carcelenjorge17@gmail.c...

jacarcelenb / frontendBadges • a day ago	Connect
jacarcelenb / backendBadges • 6 days ago	Connect
jacarcelenb / authUserAPI • 2 months ago	Connect
jacarcelenb / apiEmail • 7 months ago	Connect

Configure account

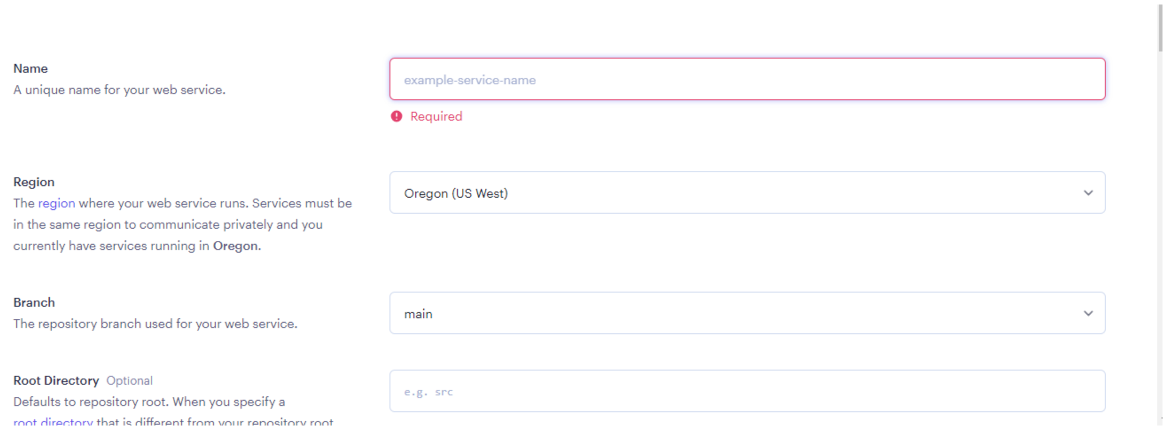
GitLab

+ Connect account

Public Git repository

Use a public repository by entering the URL below. Features like PR Previews and Auto-

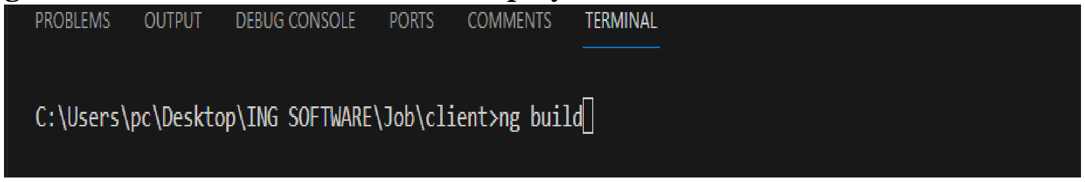
6. Complete the following fields and click on Create Web Service.



Name A unique name for your web service.	example-service-name Required
Region The region where your web service runs. Services must be in the same region to communicate privately and you currently have services running in **Oregon**.	Oregon (US West)
Branch The repository branch used for your web service.	main
Root Directory Optional Defaults to repository root. When you specify a root directory that is different from your repository root	e.g. src

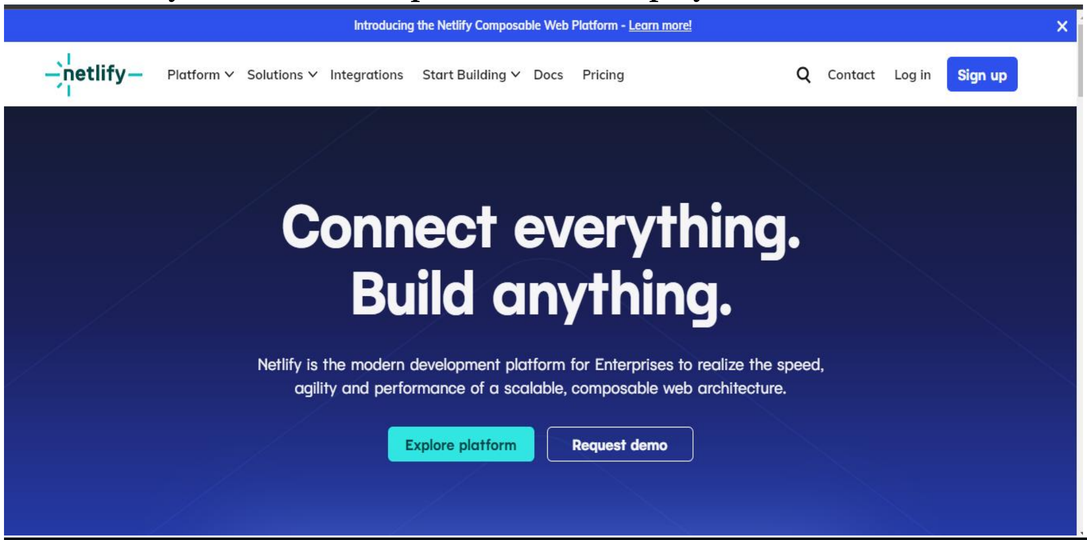
Deploy frontend project on Netlify

1. Before going to Netlify you must run the following command **ng build** to get the dist folder for the fronted deployment

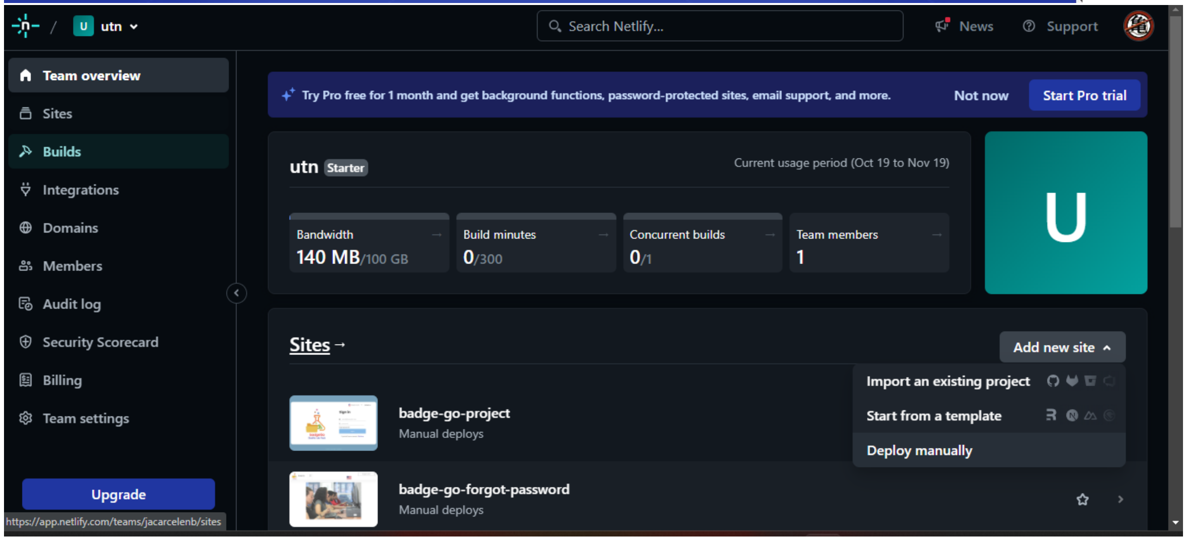


```
C:\Users\pc\Desktop\ING SOFTWARE\Job\client>ng build
```

2. Go to Netlify and select the option Manual Deployment.



Netlify is the modern development platform for Enterprises to realize the speed, agility and performance of a scalable, composable web architecture.



Try Pro free for 1 month and get background functions, password-protected sites, email support, and more.

utn Starter — Current usage period (Oct 19 to Nov 19)

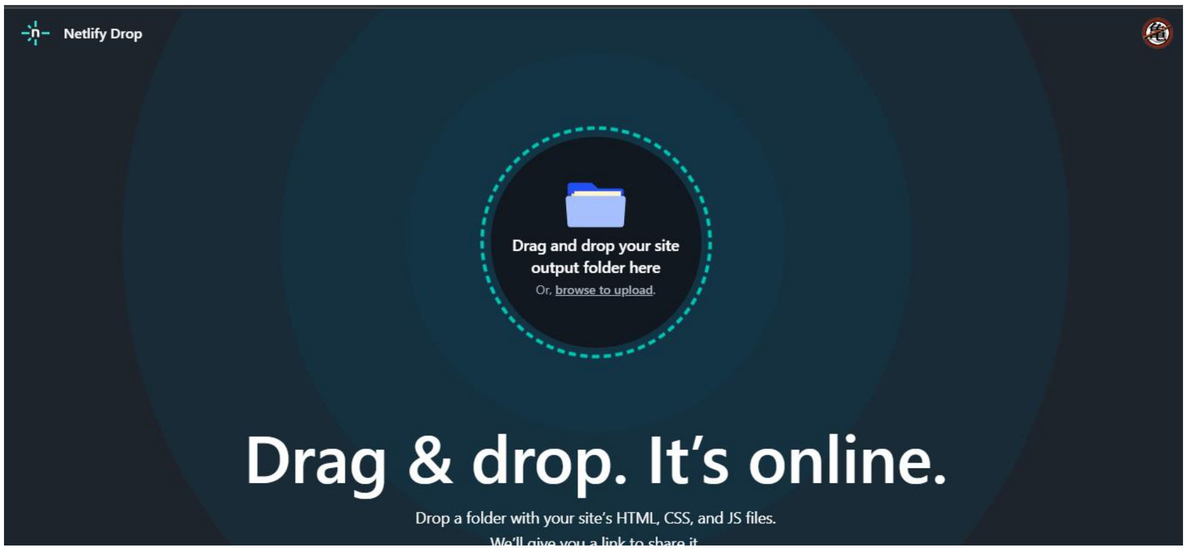
Bandwidth	Build minutes	Concurrent builds	Team members
140 MB/100 GB	0/300	0/1	1

badge-go-project
Manual deploys

badge-go-forgot-password
Manual deploys

https://app.netlify.com/teams/jacarcelenb/sites

3. Load the dist folder and wait until the application is displayed.



Drag & drop. It’s online.

Drop a folder with your site’s HTML, CSS, and JS files.

We’ll give you a link to share it

Recommendations

- It is not necessary to use platforms such as Render or Netlify, if necessary, you can opt for other platforms.

- The version used to work with Angular is version 12, currently this version is no longer maintained, so it is recommended to continue working with this version and avoid migrating to another version because it could seriously affect the application.

- Avoid using libraries or packages that are not necessary because it may affect the performance of the application.