

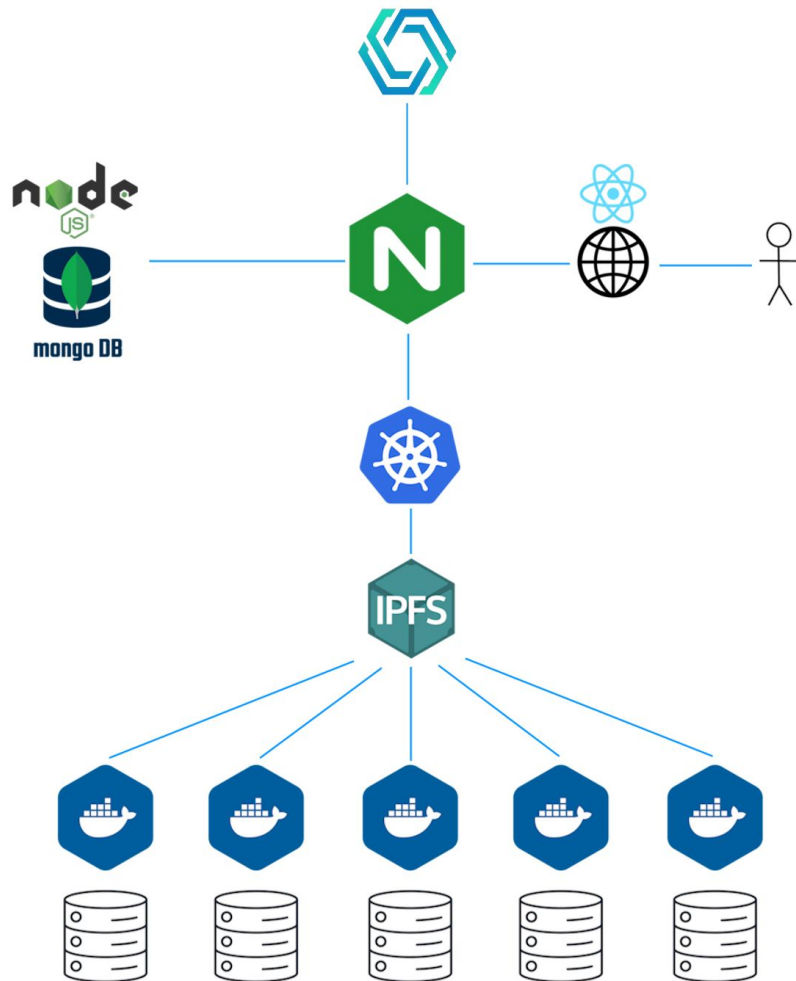
**UPC Cloud**

Javier Armaza Bravo  
Sergi Colomer Segalés  
Sofía López Olivares  
Pablo Rosas Roda

# Índice

- Propuesta inicial
- Paquetes trabajo
  - Nginx
  - FrontEnd
  - Backend
  - IPFS
  - Kubernetes
- Tareas restantes
- Carta Gantt

# Propuesta inicial



# Paquetes trabajo

1. IPFS



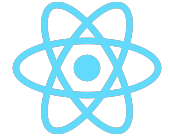
2. Nginx



3. BackEnd



4. FrontEnd



5. Kubernetes



6. Blockchain



# Nginx

```
# Default server configuration
#
server {
    listen 8080 default_server;
    #listen [::]:80 default_server;

    # SSL configuration
    #
    # listen 443 ssl default_server;
    # listen [::]:443 ssl default_server;
    #
    # Note: You should disable gzip for SSL traffic.
    # See: https://bugs.debian.org/773332
    #
    # Read up on ssl_ciphers to ensure a secure configuration.
    # See: https://bugs.debian.org/765782
    #
    # Self signed certs generated by the ssl-cert package
    # Don't use them in a production server!
    #
    # include snippets/snakeoil.conf;

    root /var/www/upcccloud/html;

    # Add index.php to the list if you are using PHP
    index index.html index.htm index.nginx-debian.html;

    server_name _;

    location / {
        # First attempt to serve request as file, then
        # as directory, then fall back to displaying a 404.
        #try_files $uri $uri/ =404;
        try_files $uri /index.html;
    }
}
```

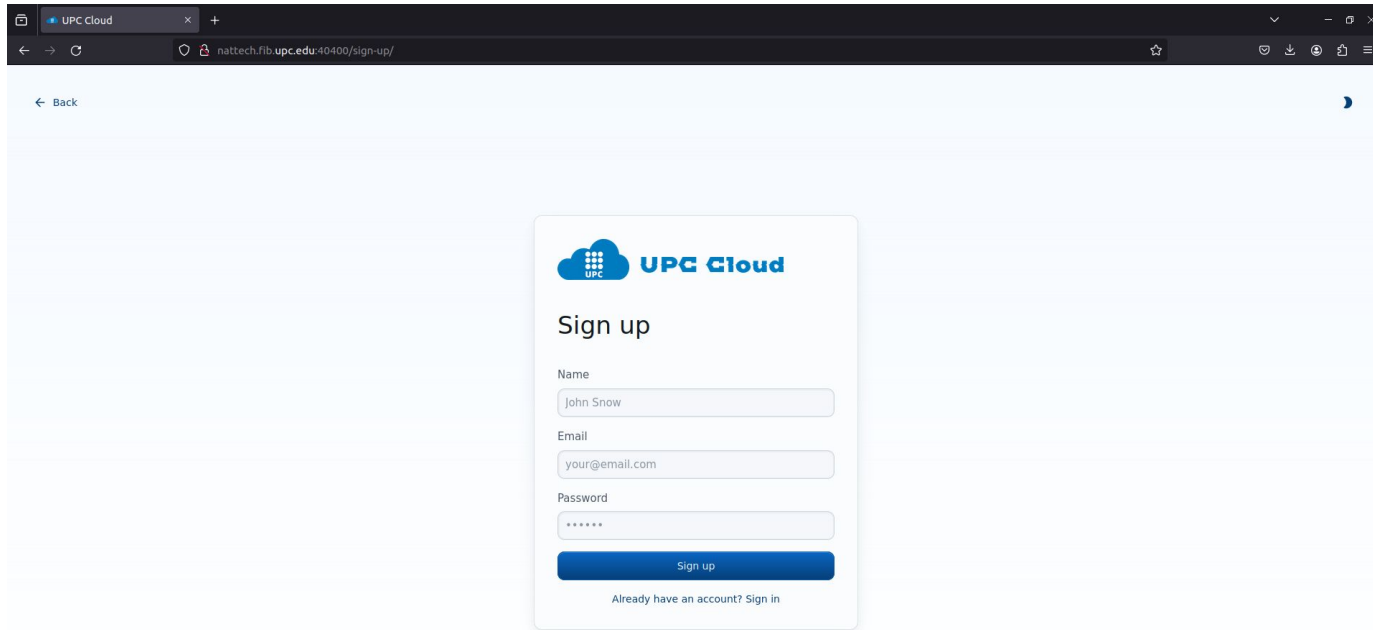
```
alumne@warmachine:~$ ls /var/www/upcccloud/html/
asset-manifest.json favicon.ico index.html manifest.json robots.txt static upcccloud_logo.png
alumne@warmachine:~$ _
```

# Nginx



Welcome to UPC Cloud!

# FrontEnd




The screenshot shows a web browser window with a single tab titled "UPC Cloud". The address bar displays the URL "nattech.fib.upc.edu:40400/sign-up/". The page content features a "Sign up" form centered on a light blue background. The form includes the "UPC Cloud" logo, a title "Sign up", and three input fields for "Name", "Email", and "Password". The "Name" field contains "John Snow", the "Email" field contains "your@email.com", and the "Password" field is masked with six asterisks. A blue "Sign up" button is positioned below the fields, and a link "Already have an account? Sign in" is located at the bottom of the form.

UPC Cloud

nattech.fib.upc.edu:40400/sign-up/

← Back

 **UPC Cloud**

## Sign up

Name

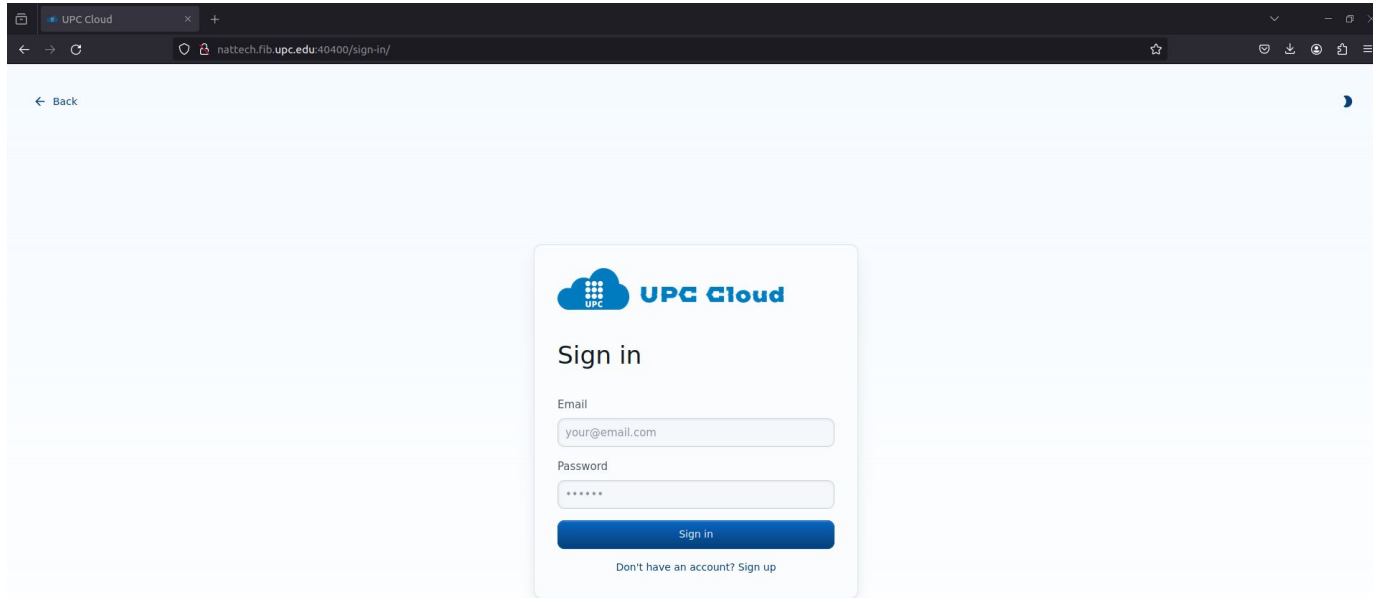
Email

Password

[Sign up](#)

[Already have an account? Sign in](#)

# FrontEnd

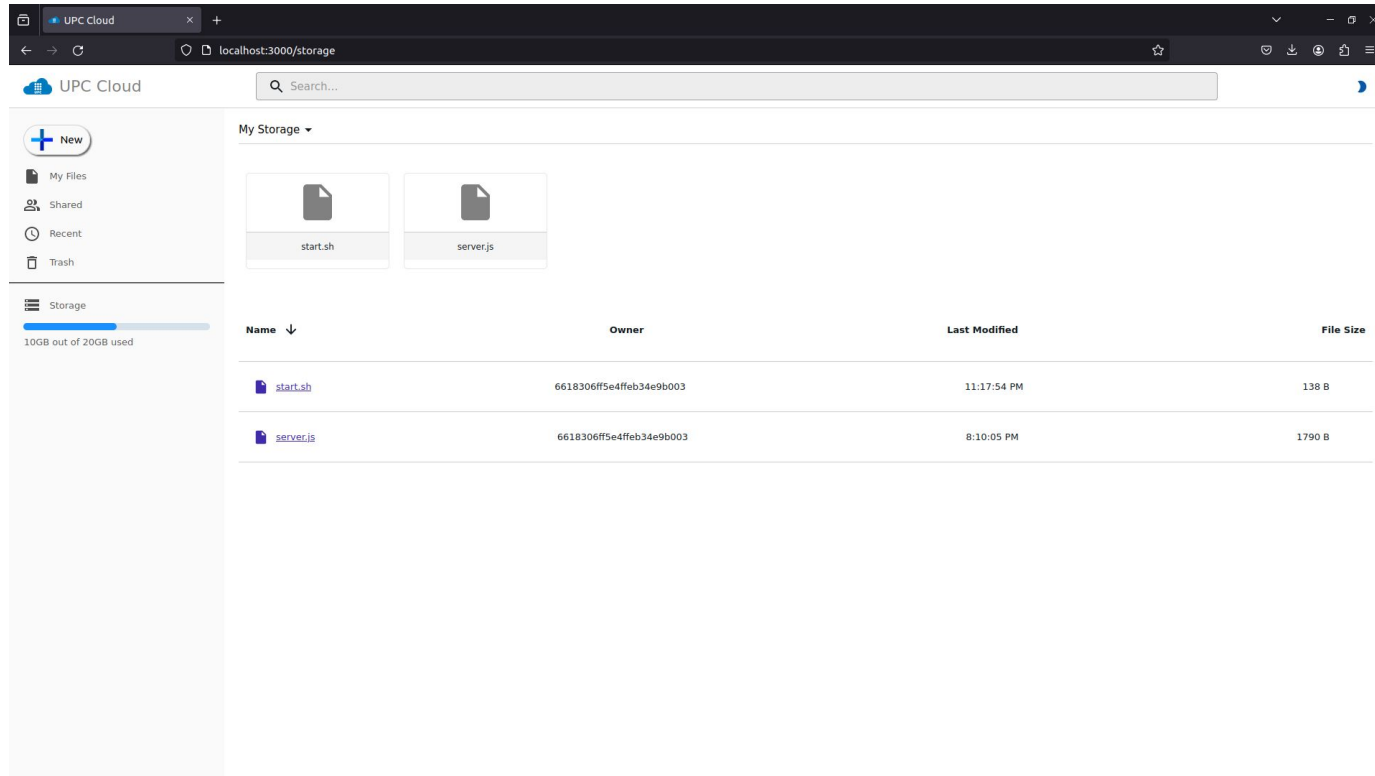


The screenshot shows a web browser window with the address bar displaying "nattech.fib.upc.edu:40400/sign-in/". The page features a "Sign in" form with the following elements:

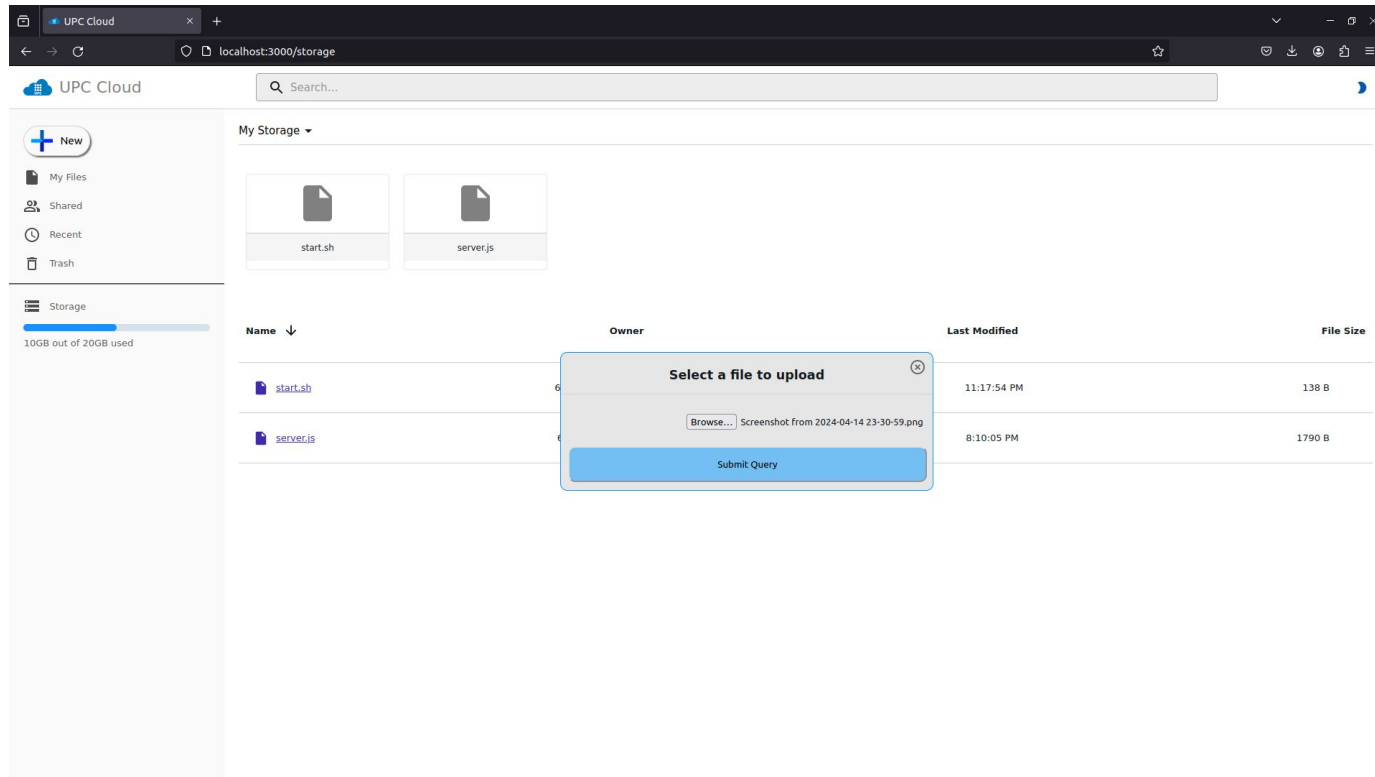
- UPC Cloud logo:** A blue cloud icon with a grid pattern and the text "UPC Cloud".
- Sign in title:** The text "Sign in" in a bold, black font.
- Email field:** A text input field with the placeholder "your@email.com".
- Password field:** A text input field with masked characters "\*\*\*\*\*".
- Sign in button:** A blue button with the text "Sign in".
- Link:** A link below the button that reads "Don't have an account? Sign up".



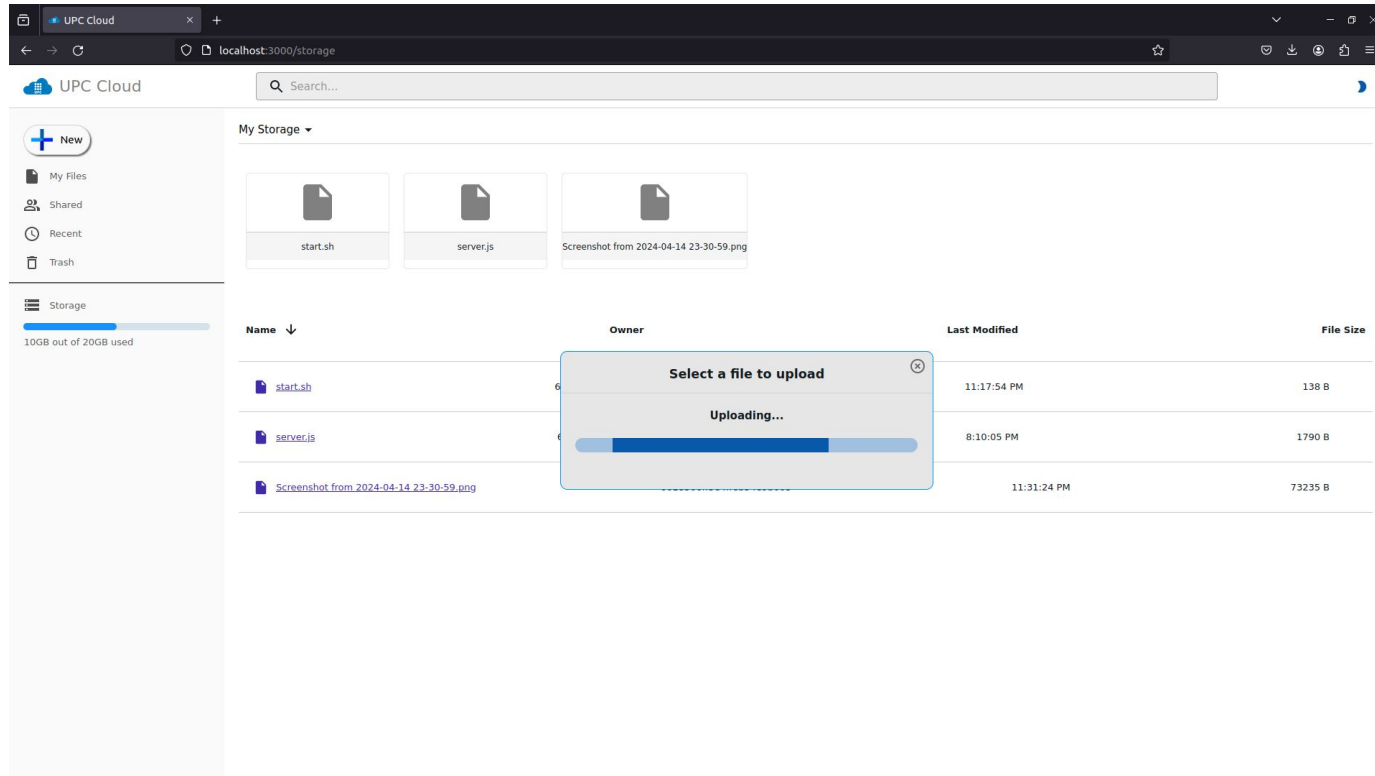
# FrontEnd



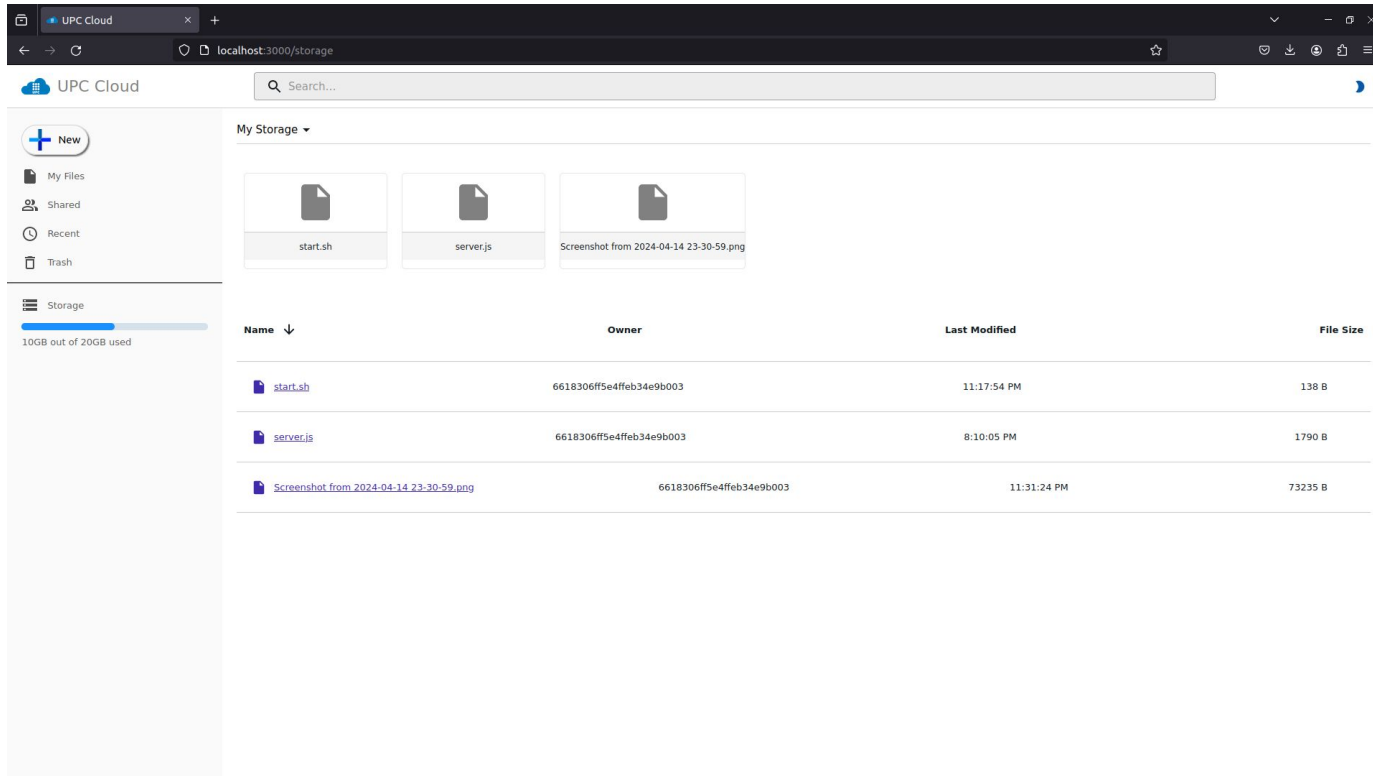
# FrontEnd



# FrontEnd

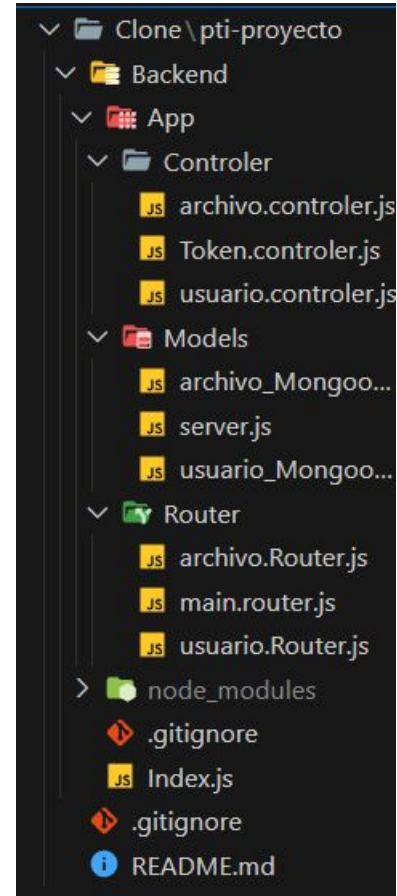


# FrontEnd



# Backend

En desarrollo, Base de datos desplegada



# Backend

```
JS Index.js X
Clone > pti-proyecto > Backend > JS Index.js > ...
1  const mongoose = require('mongoose');
2  const cors = require('cors');
3  const express = require('express');
4  const db = require('./App/Models/server.js');
5  const app = express();
6  const port = 3000;
7  app.use(cors());
8  app.use(express.json());
9  app.use(express.urlencoded({ extended: true }));
10 db.mongoose.connect('mongodb://localhost:27017/Proyecto', {
11   useNewUrlParser: true,
12   useUnifiedTopology: true
13 }).then(() => {
14   console.log('Conexión a la base de datos exitosa');
15 }).catch((error) => {
16   console.log('Error al conectar a la base de datos');
17   console.log(error);
18 });
19 app.get('/', (req, res) => {
20   res.send('Hello World!');
21 });
22 require('./App/Router/main.router')(app);
23 app.listen(port, () => {
24   console.log(`Example app listening at http://localhost:${port}`);
25 });
```

# Backend

- crearusuario
- obtenerusuario
- eliminarusuario
- modificarusuario

```
const db = require('../Models/server.js');
const usuario = require('../Models/usuario_Mongoose.js');
const mongoose = require('mongoose');
const token = require('../Controler/Token.controler.js');

const crearusuario = async (req, res) => {
  const { nombre, clave, correo } = req.body;
  try {
    const usuarioExistente = await usuario.usuario_MongooseModel.findOne({ correo });
    if (usuarioExistente) {
      return res.status(400).send('El usuario ya existe');
    }
    const nuevoUsuario = new usuario.usuario_MongooseModel({
      _id: new mongoose.Types.ObjectId(),
      nombre,
      clave,
      correo
    });
    await nuevoUsuario.save();
    res.status(201).send('Usuario registrado correctamente');
  } catch (error) {
    console.error('Error al registrar usuario:', error);
    res.status(500).send('Error interno del servidor: ' + error.message);
  }
};
```

# Backend

JS usuario\_Mongoose.js X

Clone > pti-proyecto > Backend > App > Models > JS usuario\_Mongoose.js > ...

```
1  const mongoose = require("mongoose");
2
3  const Schema = mongoose.Schema;
4  const model = mongoose.model;
5
6  const usuario_Mongoose = new Schema({
7    "_id": mongoose.ObjectId,
8    "nombre": String,
9    "clave": String,
10   "correo": String,
11 }, { collection: "usuario" });
12
13 const usuario_MongooseModel = model("usuario_Mongoose", usuario_Mongoose);
14
15 module.exports = { usuario_Mongoose, usuario_MongooseModel };
```

JS usuario.Router.js X

Clone > pti-proyecto > Backend > App > Router > JS usuario.Router.js > ...

```
1  const router = require('express').Router();
2  const { crearusuario, obtenerusuario, eliminarusuario, modificarusuario, login } = require('../Controler/usuario.controler');
3
4  router.get('/', (req, res) => {
5    res.send('Ruta de usuario');
6  });
7
8  router.post('/crearusuario', crearusuario);
9  router.get('/obtenerusuario', obtenerusuario);
10 router.delete('/eliminarusuario', eliminarusuario);
11 router.put('/modificarusuario', modificarusuario);
12 router.post('/login', login);
13
14 module.exports = router;
```



# Backend

```
Token.controller.js X
Clone > pti-proyecto > Backend > App > Controller > JS Token.controller.js > ...
1  const jwt = require('jsonwebtoken');
2  const key = 'clave';
3  const Usuario = require('../Models/usuario_Mongoose');
4
5  async function crearToken(usuario) {
6    const clave = usuario.clave;
7    const correo = usuario.correo;
8
9    try {
10     const usuarioExistente = await Usuario.usuario_MongooseModel.findOne({ correo });
11     if (!usuarioExistente) {
12       throw new Error('El usuario no existe');
13     }
14     const token = jwt.sign({ correo, clave }, key, { expiresIn: '1h' });
15     return token;
16   } catch (error) {
17     console.error('Error al crear token:', error);
18     throw new Error('Error interno del servidor: ' + error.message);
19   }
20 }
21
```

# Backend

- crearusuario
- obtenerusuario
- eliminarusuario
- modificarusuario
- **login**

```
const login = async (req, res) => {
  try {
    const { correo, clave } = req.body;
    if (correo && clave) {
      const usuarioExistente = await usuario.usuario_MongooseModel.findOne({ correo, clave });
      if (usuarioExistente) {
        const tokenres = await token.crearToken(usuarioExistente);
        console.log("Conectado");
        return res.send(tokenres);
      } else {
        return res.status(404).send('Usuario no encontrado');
      }
    } else {
      return res.status(400).send('Correo o clave no proporcionados');
    }
  } catch (error) {
    console.error('Error al iniciar sesión:', error);
    res.status(500).send('Error interno del servidor: ' + error.message);
  }
};

module.exports = { crearusuario, obtenerusuario, eliminarusuario, modificarusuario, login };
```

# Backend

- creararchivo
- obtenerarchivo
- getall
- eliminararchivo
- modificararchivo

```
const db = require('../Models/server.js');
const archivo = require('../Models/archivo_Mongoose.js');
const mongoose = require('mongoose');
const usuario = require('../Models/usuario_Mongoose.js');
const Token = require('../Controler/Token.controller.js');

const creararchivo = async (req, res) => {
  const { nombreArchivo, ruta, fechaSubida, tamano, hora, correo } = req.body;
  try {
    const usuarioExistente = await usuario.usuario_MongooseModel.findOne({ correo });
    if (!usuarioExistente) {
      return res.status(404).send('El propietario especificado no existe');
    }
    const nuevoArchivo = new archivo.archivo_MongooseModel({
      _id: new mongoose.Types.ObjectId(),
      nombreArchivo,
      ruta,
      fechaSubida,
      tamano,
      hora,
      propietario: usuarioExistente._id,
    });

    await nuevoArchivo.save();

    res.status(201).send('Archivo creado correctamente');
  } catch (error) {
    console.error('Error al crear archivo:', error);
    res.status(500).send('Error interno del servidor: ' + error.message);
  }
};
```

# Backend

JS archivo.Router.js X

Clone > pti-proyecto > Backend > App > Router > JS archivo.Router.js > ...

```
1 const router = require('express').Router();
2 const { creararchivo, obtenerarchivo, modificararchivo, eliminararchivo, getall } = require('../Controler/archivo.controler');
3 router.get('/', (req, res) => {
4   res.send('Ruta de archivo');
5 });
6 router.post('/creararchivo', creararchivo);
7 router.get('/obtenerarchivo', obtenerarchivo);
8 router.delete('/eliminararchivo', eliminararchivo);
9 router.put('/modificararchivo', modificararchivo);
10 router.post('/getall', getall);
11
12 module.exports = router;
```

JS archivo\_Mongoose.js X

Clone > pti-proyecto > Backend > App > Models > JS archivo\_Mongoose.js > ...

```
1 const mongoose = require("mongoose");
2
3 const Schema = mongoose.Schema;
4 const model = mongoose.model;
5
6 const archivo_Mongoose = new Schema({
7   "_id": mongoose.ObjectId,
8   "nombreArchivo": String,
9   "ruta": String,
10  "fechaSubida": String,
11  "tamano": String,
12  "hora": String,
13  propietario: {
14    type: mongoose.Schema.Types.ObjectId,
15    ref: 'Usuario',
16  },
17 }, { collection: "archivo" })
18
19 const archivo_MongooseModel = model("archivo_Mongoose", archivo_Mongoose);
20
21 module.exports = { archivo_Mongoose, archivo_MongooseModel };
```

# Backend

```
async function validartoken(token, correo) {
  try {
    const usuarioExistente = await Usuario.usuario_MongooseModel.findOne({ correo });
    if (!usuarioExistente) {
      throw new Error('El usuario no existe');
    }
    const tokenValido = jwt.verify(token, key);
    if (tokenValido.correo == correo) {
      return tokenValido;
    }
    else {
      throw new Error('El token no corresponde al usuario');
    }
  } catch (error) {
    console.error('Error al validar token:', error);
    throw new Error('Error interno del servidor: ' + error.message);
  }
}

module.exports = { crearToken, validartoken };
```

# Backend

The screenshot shows a VS Code editor with a REST client request and response, and a terminal window.

**REST Client Request:**

- Method: POST
- URL: `http://localhost:3000/usuario/login`
- Body (JSON):

```
{  "clave": "A123",  "correo": "usuario312@gmail.com"}
```

**REST Client Response:**

- Status: 200 OK
- Size: 191 Bytes
- Time: 47 ms
- Response body:

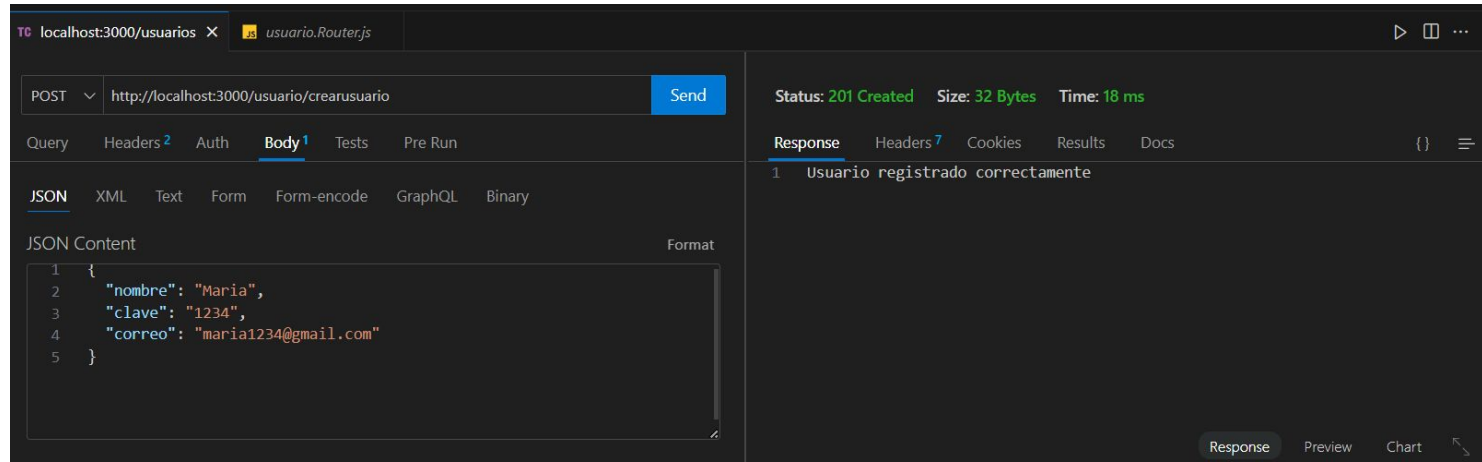
```
1 eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJjb3JyZW8iOiJ1c3VhcmVhZyQgdWlslmNvbSIsImN5YXZlIjoieTEyMyIsIm1ldhcI6MTcxMzEyNjI3OCwiZXhwIjoxNzEzMTI5ODc4fQ.nfaIK4iEM3188BKn12nDgodGtzVKaeGfKlKqdh9CGPU
```

**Terminal Window:**

```
PS C:\Users\SOFI\Desktop\PTI2\Clone\pti-proyecto\Backend>
PS C:\Users\SOFI\Desktop\PTI2\Clone\pti-proyecto\Backend>
PS C:\Users\SOFI\Desktop\PTI2\Clone\pti-proyecto\Backend>
PS C:\Users\SOFI\Desktop\PTI2\Clone\pti-proyecto\Backend>
PS C:\Users\SOFI\Desktop\PTI2\Clone\pti-proyecto\Backend>
PS C:\Users\SOFI\Desktop\PTI2\Clone\pti-proyecto\Backend>
PS C:\Users\SOFI\Desktop\PTI2\Clone\pti-proyecto\Backend>
PS C:\Users\SOFI\Desktop\PTI2\Clone\pti-proyecto\Backend>
PS C:\Users\SOFI\Desktop\PTI2\Clone\pti-proyecto\Backend>
PS C:\Users\SOFI\Desktop\PTI2\Clone\pti-proyecto\Backend>
PS C:\Users\SOFI\Desktop\PTI2\Clone\pti-proyecto\Backend>
(node:3608) [MONGODB DRIVER] Warning: useUrlParser is a deprecated option: useUrlParser has no effect since Node.js Driver version 4.0.0 and will be removed in the next major version
(Use `node --trace-warnings ...` to show where the warning was created)
(node:3608) [MONGODB DRIVER] Warning: useUnifiedTopology is a deprecated option: useUnifiedTopology has no effect since Node.js Driver version 4.0.0 and will be removed in the next major version
Example app listening at http://localhost:3000
Conexión a la base de datos exitosa
Conectado
[]
```



# Backend



localhost:3000/usuarios X usuario.Router.js

POST http://localhost:3000/usuario/crearusuario Send

Query Headers 2 Auth Body 1 Tests Pre Run

JSON XML Text Form Form-encode GraphQL Binary

JSON Content Format

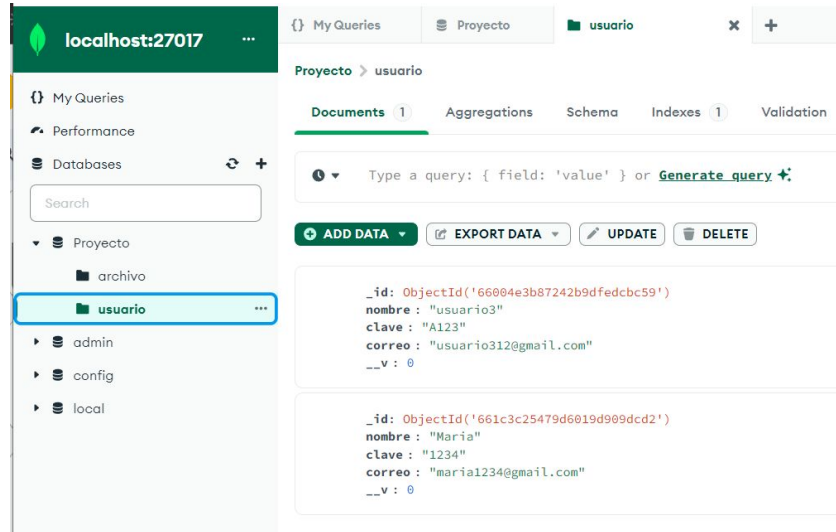
```
1 {
2   "nombre": "Maria",
3   "clave": "1234",
4   "correo": "maria1234@gmail.com"
5 }
```

Status: 201 Created Size: 32 Bytes Time: 18 ms

Response Headers 7 Cookies Results Docs

1 Usuario registrado correctamente

Response Preview Chart



localhost:27017 ...

My Queries Performance Databases Proyecto Search

Proyecto > usuario

Documents 1 Aggregations Schema Indexes 1 Validation

Type a query: { field: 'value' } or Generate query

ADD DATA EXPORT DATA UPDATE DELETE

```
_id: ObjectId('66084e3b87242b9dfedcb59')
nombre: "usuario3"
clave: "A123"
correo: "usuario312@gmail.com"
__v: 0
```

```
_id: ObjectId('661c3c25479d6019d909dcd2')
nombre: "Maria"
clave: "1234"
correo: "maria1234@gmail.com"
__v: 0
```

# Backend

```
_id: ObjectId('661c3d05479d6019d909dcd5')
nombreArchivo: "prueba1"
ruta: "s/s/s"
fechaSubida: "01/04/24"
tamano: "200"
hora: "23:59"
propietario: ObjectId('661c3c25479d6019d909dcd2')
__v: 0
```

```
_id: ObjectId('661c3d32479d6019d909dcd8')
nombreArchivo: "prueba2"
ruta: "s/s/p"
fechaSubida: "02/04/24"
tamano: "200"
hora: "23:59"
propietario: ObjectId('661c3c25479d6019d909dcd2')
__v: 0
```

The screenshot displays a REST client interface with two panels. The top panel shows a POST request to `http://localhost:3000/archivo/creararchivo` with a JSON body. The response is a 201 status code, indicating the file was created successfully, with a response time of 10 ms. The bottom panel shows a similar POST request with a slightly different JSON body. The response is also a 201 status code, with a response time of 8 ms. Both panels show the JSON content of the request and response, and the response body is `1 Archivo creado correctamente`.

**Top Panel:**

- Method: POST, URL: `http://localhost:3000/archivo/creararchivo`
- Body (JSON):

```
{
  "nombreArchivo": "prueba1",
  "ruta": "s/s/s",
  "fechaSubida": "01/04/24",
  "tamano": "200",
  "hora": "23:59",
  "correo": "maria1234@gmail.com"
}
```
- Status: 201 Created, Size: 28 Bytes, Time: 10 ms
- Response: 1 Archivo creado correctamente

**Bottom Panel:**

- Method: POST, URL: `http://localhost:3000/archivo/creararchivo`
- Body (JSON):

```
{
  "nombreArchivo": "prueba2",
  "ruta": "s/s/p",
  "fechaSubida": "02/04/24",
  "tamano": "200",
  "hora": "23:59",
  "correo": "maria1234@gmail.com"
}
```
- Status: 201 Created, Size: 28 Bytes, Time: 8 ms
- Response: 1 Archivo creado correctamente



# Backend

The screenshot shows a web client interface with a dark theme. At the top, there are tabs for 'localhost:3000/usuarios' and 'archivo.Router.js'. The main area is divided into two panels. The left panel shows a POST request to 'http://localhost:3000/archivo/getall' with a 'Send' button. Below the URL bar, there are tabs for 'Query', 'Headers', 'Auth', 'Body', 'Tests', and 'Pre Run'. The 'Body' tab is selected, showing a JSON object: 

```
{  "correo": "maria1234@gmail.com"}
```

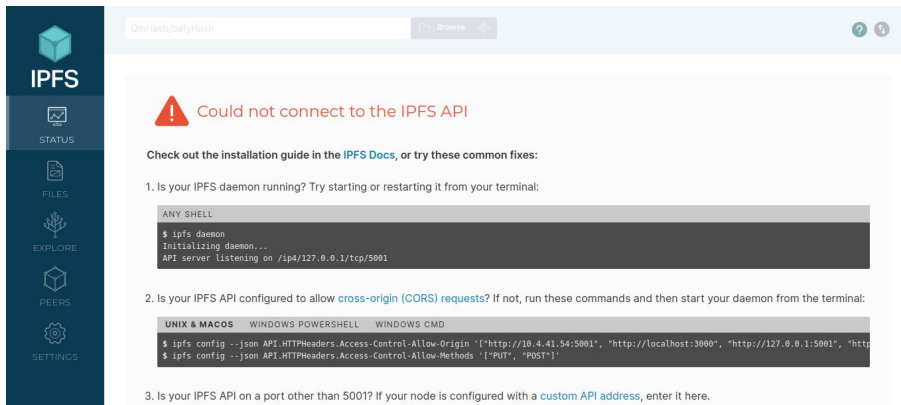
. The right panel shows the response status '200 OK', size '361 Bytes', and time '17 ms'. Below this, there are tabs for 'Response', 'Headers', 'Cookies', 'Results', and 'Docs'. The 'Response' tab is selected, showing a JSON array of two objects: 

```
[  {    "_id": "661c3d05479d6019d909dcd5",    "nombreArchivo": "prueba1",    "ruta": "s/s/s",    "fechaSubida": "01/04/24",    "tamano": "200",    "hora": "23:59",    "propietario": "661c3c25479d6019d909dcd2",    "__v": 0  },  {    "_id": "661c3d32479d6019d909dcd8",    "nombreArchivo": "prueba2",    "ruta": "s/s/p",    "fechaSubida": "02/04/24",    "tamano": "200",    "hora": "23:59",    "propietario": "661c3c25479d6019d909dcd2",    "__v": 0  }]
```

# IPFS

## Implementado Nodo base IPFS

## Implementando Conjuntos Nodos



The screenshot shows the IPFS Desktop application interface. On the left is a dark blue sidebar with icons and labels: IPFS, STATUS, FILES, EXPLORE, PEERS, and SETTINGS. The main area has a light blue header with a search bar containing 'QmRasdybafyKash' and a 'Browse' button. Below the header, a red warning icon is followed by the text 'Could not connect to the IPFS API'. Underneath, it says 'Check out the installation guide in the [IPFS Docs](#), or try these common fixes:'. There are three numbered steps: 1. 'Is your IPFS daemon running? Try starting or restarting it from your terminal:' followed by a terminal snippet showing 'ipfs daemon' being run successfully. 2. 'Is your IPFS API configured to allow cross-origin (CORS) requests? If not, run these commands and then start your daemon from the terminal:' followed by terminal snippets for 'UNIX & MACOS' and 'WINDOWS POWERSHELL / WINDOWS CMD' showing 'ipfs config' commands to set CORS headers. 3. 'Is your IPFS API on a port other than 5001? If your node is configured with a [custom API address](#), enter it here.'

# Kubernetes

IPFS

Configurado en máquinas

Falta unir los workers al clúster

# Tareas restantes

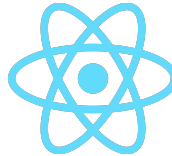
- Implementación de IPFS en Backend



- Orquestación del sistema de IPFS



- Acabar FrontEnd

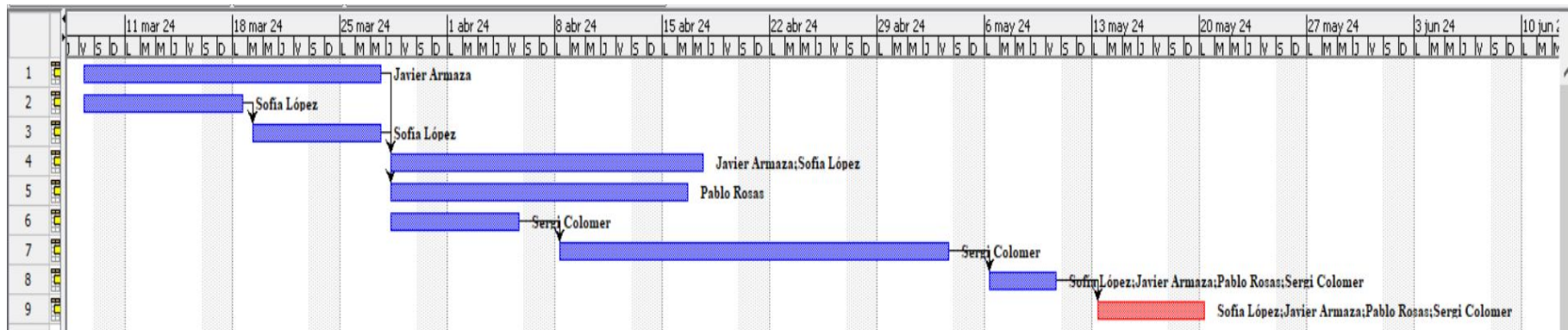


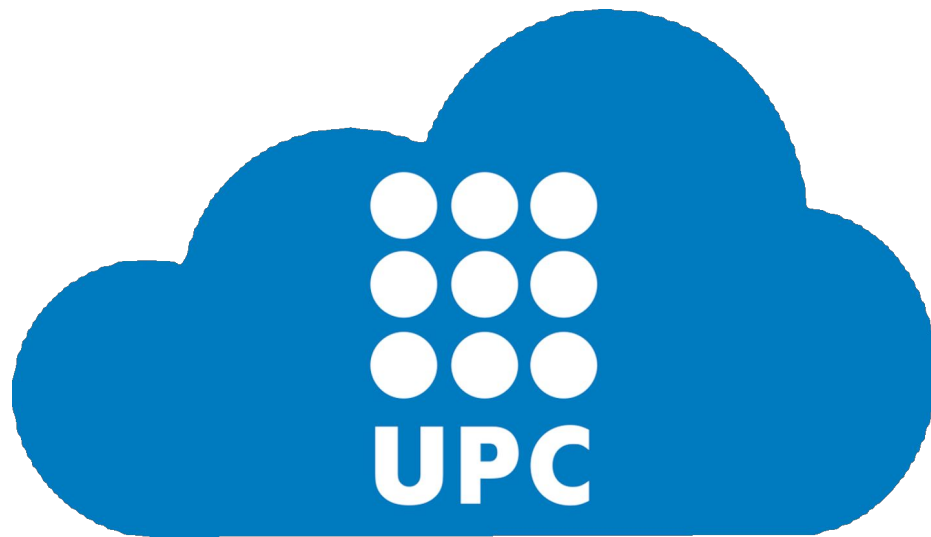
- Creación de blockchain



# Carta Gantt

		Nombre	Duración	Inicio	Terminado	Predecesores	Nombres del Recurso
1		Servidor Nginx	14 days	08-03-24 8:00	27-03-24 17:00		Javier Armaza
2		Base de datos	7 days	08-03-24 8:00	18-03-24 17:00		Sofía López
3		Back End	7 days	19-03-24 8:00	27-03-24 17:00	2	Sofía López
4		frond	15 days	28-03-24 8:00	17-04-24 17:00	3	Javier Armaza;Sofía López
5		IPFS	14 days	28-03-24 8:00	16-04-24 17:00	1	Pablo Rosas
6		Docker	7 days	28-03-24 8:00	05-04-24 17:00		Sergi Colomer
7		Kubernetes	20 days	08-04-24 8:00	03-05-24 17:00	6	Sergi Colomer
8		Conexion	5 days	06-05-24 8:00	10-05-24 17:00	7	Sofía López;Javier Armaza;...
9		Blockchain	5 days	13-05-24 9:00	20-05-24 9:00	8	Sofía López;Javier Armaza;...





**UPC Cloud**

Javier Armaza Bravo  
Sergi Colomer Segalés  
Sofía López Olivares  
Pablo Rosas Roda