

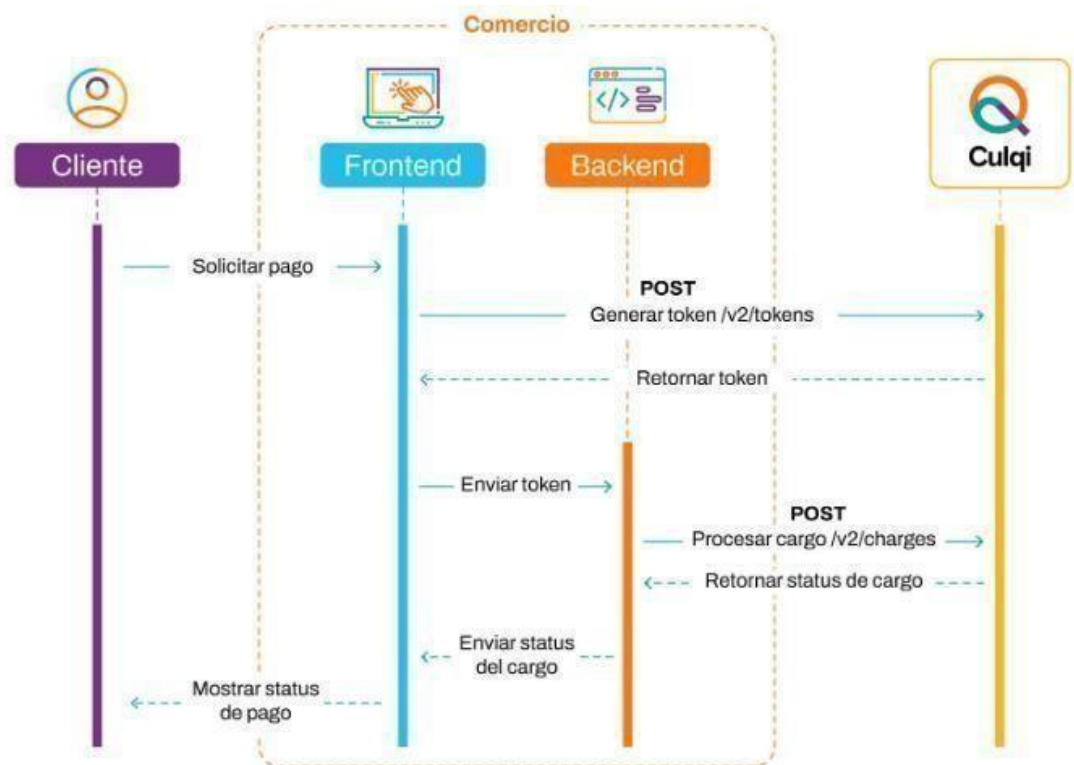
Tokenización de Tarjetas

1. Contexto

Las pasarelas de pagos guardan las tarjetas de crédito en una bóveda encriptada (encriptación en reposo) para evitar que la información sensible se pueda filtrar o que pueda ser interceptada en otro proceso del sistema.

El proceso de tokenización funciona enviando los datos de la tarjeta al tokenizador, este valida y guarda la información en la BD encriptada y devuelve un ID (token) como llave del registro el cual puede ser usado luego en los distintos procesos de culqi.

En el siguiente gráfico puedes ver donde se usa este API de tokenización en el proceso de autorización de una tarjeta de crédito.



Prueba técnica – Backend Gateway Pos

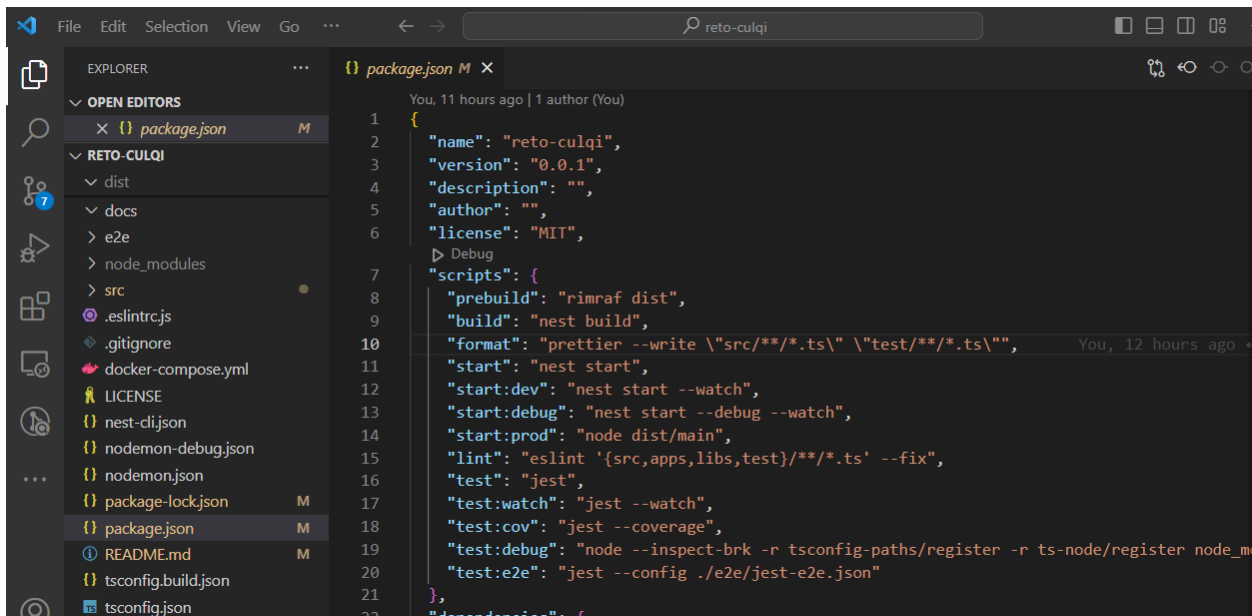
2. Requerimiento técnico

Las tecnologías a utilizar en la prueba son las siguientes:

- **Desarrollo Backend:** TypeScript, nodejs
- **Servicio AWS:** EKS (Deploy de la app) o en Local
- **BD no relacional:** Redis
- **Test:Unitarios:** Jest / jest - koa -mocks

El package.json del proyecto debe tener 2 comandos:

1. Comando para compilar TypeScript y generar el build de la aplicación que debe exponer los métodos que serán utilizados en el servicio de EKS o de forma local.
2. Comando para ejecutar los test de la aplicación en un entorno local.



```
1  {
2    "name": "reto-culqi",
3    "version": "0.0.1",
4    "description": "",
5    "author": "",
6    "license": "MIT",
7    "scripts": {
8      "prebuild": "rimraf dist",
9      "build": "nest build",
10     "format": "prettier --write 'src/**/*.ts' 'test/**/*.ts'",
11     "start": "nest start",
12     "start:dev": "nest start --watch",
13     "start:debug": "nest start --debug --watch",
14     "start:prod": "node dist/main",
15     "lint": "eslint '{src,apps,libs,test}/**/*.ts' --fix",
16     "test": "jest",
17     "test:watch": "jest --watch",
18     "test:cov": "jest --coverage",
19     "test:debug": "node --inspect-brk -r tsconfig-paths/register -r ts-node/register node_m
20     "test:e2e": "jest --config ./e2e/jest-e2e.json"
21   },
22   "dependencies": {
```

Comandos

```
npm run start | npm run start:dev
```





```
npm run build
```

```
npm run test
```

Prueba técnica – Backend Gateway Pos

El proyecto deberá tener un README.MD con la descripción de los pasos a seguir para poder levantar el proyecto en un entorno local y así poder ejecutar los 2 comandos de npm.

MongoDB with Docker (docker-compose.yml)

	mongodb-1	mongo:latest	Running	0.53%	27017:27017	22 hours ago			
	66b0c003a24e								

```
C:\Users\Royer Leandro\Documents\reto-culqi>docker ps -a
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS
66b0c003a24e   mongo:latest   "docker-entrypoint.s..." 23 hours ago   Up 22 hours   0.0.0.0:27017->27017/tcp
```

Reto Culqi Royer Robles - NestJS

Installation

```
npm install
```

Docker

There is a `docker-compose.yml` file for starting Docker.

```
docker-compose up
```

After running the sample, you can stop the Docker container with

```
docker-compose down
```

Run App

Then, run:

```
npm run start
```

Run Tests

Then, run:

Prueba técnica – Backend Gateway Pos

3. Flujos de negocio

I. Creación de un token

Se debe crear un método para simular la tokenización de una tarjeta de crédito/débito, este método recibirá los siguientes parámetros.

Para la creación del Token se tiene que considerar ciertas características:

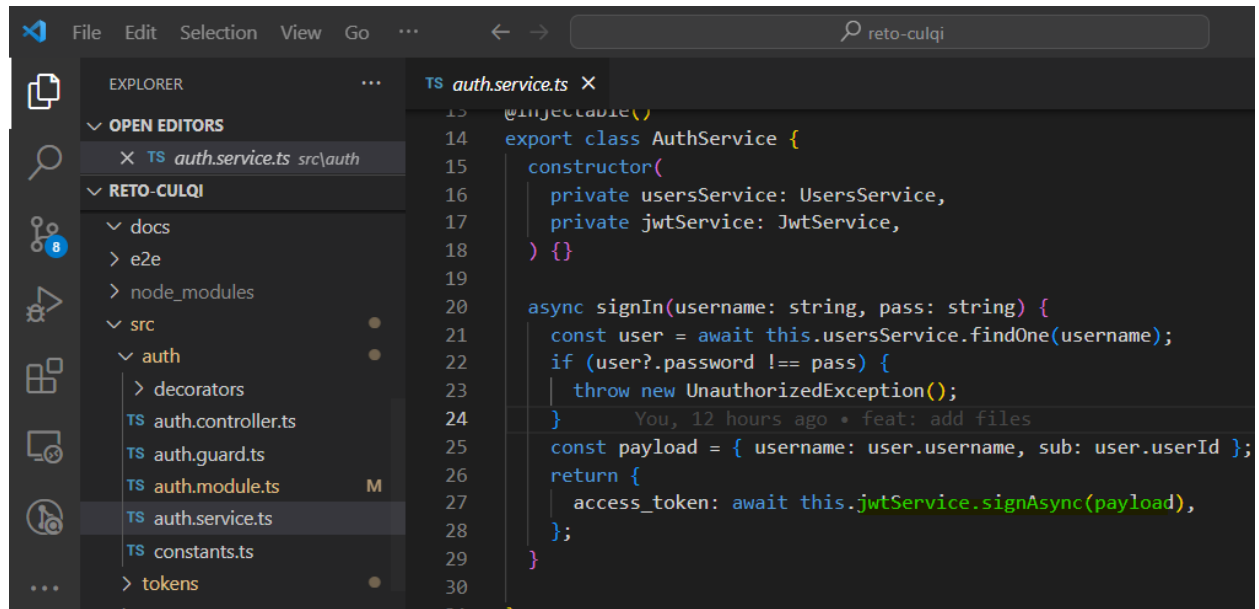
1.- La construcción del token puede ser implementando un Jwt (Json web token) las cuales deben de tener las funciones esenciales como son el (SignIn , Verify).

La función Signin permite registrar el token y darle un tiempo de expiración de 1 min para fines prácticos.

src\auth\auth.module.ts

The screenshot shows the VS Code interface. The Explorer sidebar on the left displays the project structure: 'docs', 'e2e', 'node_modules', 'src', and 'auth'. The 'auth' folder is expanded, showing 'decorators', 'auth.controller.ts', 'auth.guard.ts', and 'auth.module.ts' (which is selected). The Open Editors list shows 'TS auth.module.ts src\... M'. The main editor area displays the source code of 'auth.module.ts', which includes imports for 'UsersModule', 'AuthGuard', 'AuthService', and 'jwtConstants', and the start of the '@Module' decorator with 'imports' and 'register' options.

Prueba técnica – Backend Gateway Pos



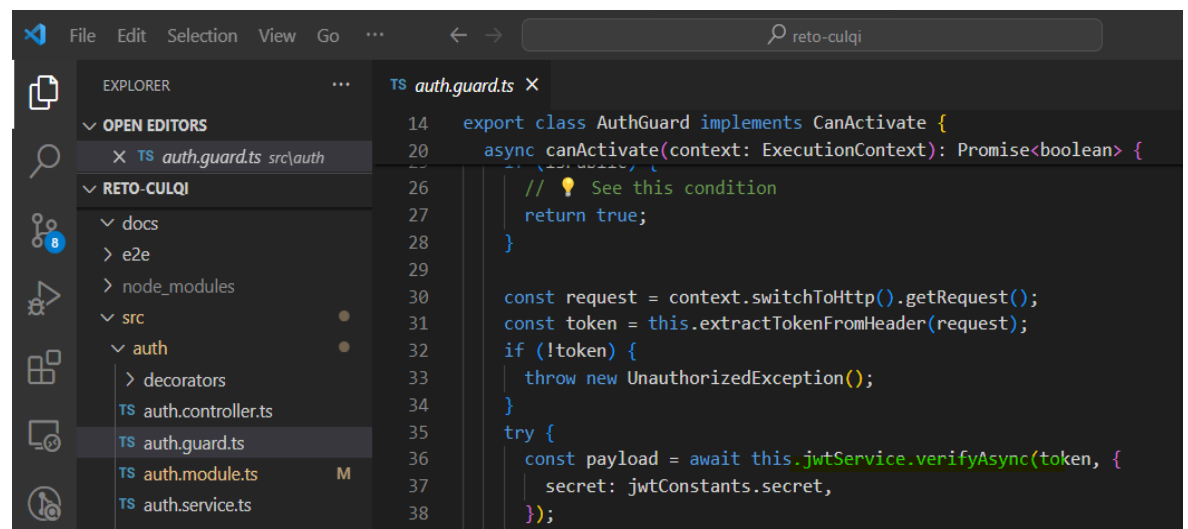
The screenshot shows the Visual Studio Code editor with the file explorer on the left and the code editor on the right. The file explorer shows the project structure with the following folders and files:

- OPEN EDITORS
 - TS auth.service.ts src\auth
- RETO-CULQI
 - docs
 - e2e
 - node_modules
 - src
 - auth
 - decorators
 - auth.controller.ts
 - auth.guard.ts
 - auth.module.ts
 - auth.service.ts
 - constants.ts
 - tokens

The code editor shows the implementation of the AuthService class in auth.service.ts:

```
13 @Injectable()
14 export class AuthService {
15   constructor(
16     private usersService: UsersService,
17     private jwtService: JwtService,
18   ) {}
19
20   async signIn(username: string, pass: string) {
21     const user = await this.usersService.findOne(username);
22     if (user?.password !== pass) {
23       throw new UnauthorizedException();
24     }
25     const payload = { username: user.username, sub: user.userId };
26     return {
27       access_token: await this.jwtService.signAsync(payload),
28     };
29   }
30 }
```

La función verify permite validar y verificar si el token que se pasa en el header del request es correcto o incorrecto.



The screenshot shows the Visual Studio Code editor with the file explorer on the left and the code editor on the right. The file explorer shows the project structure with the following folders and files:

- OPEN EDITORS
 - TS auth.guard.ts src\auth
- RETO-CULQI
 - docs
 - e2e
 - node_modules
 - src
 - auth
 - decorators
 - auth.controller.ts
 - auth.guard.ts
 - auth.module.ts
 - auth.service.ts
 - constants.ts
 - tokens

The code editor shows the implementation of the AuthGuard class in auth.guard.ts:

```
14 export class AuthGuard implements CanActivate {
20   async canActivate(context: ExecutionContext): Promise<boolean> {
21     // See this condition
22     return true;
23   }
24
25   const request = context.switchToHttp().getRequest();
26   const token = this.extractTokenFromHeader(request);
27   if (!token) {
28     throw new UnauthorizedException();
29   }
30   try {
31     const payload = await this.jwtService.verifyAsync(token, {
32       secret: jwtConstants.secret,
33     });
34   }
35 }
```

Prueba técnica – Backend Gateway Pos

src\tokens\tokens.service.ts

```
6 import { JwtService } from '@nestjs/jwt';
7
8 Injectable()
9 export class TokensService {
10   constructor(@InjectModel(Token.name) private readonly tokenModel: Model<Token>,
11     private jwtService: JwtService) {}
12
13   async create(createTokenDto: CreateTokenDto): Promise<Token> {
14     const access_token = await this.jwtService.signAsync(createTokenDto, { expiresIn: '1m' });
15     const record : any = {
```

Ejemplos de las peticiones al Backend

Body del request:

```
POST https://<url-id>.lambda-url.<region>.on.aws/tokens

JSON Bearer Query Header 1 Docs

1 {
2   "email": "gian.corzo@gmail.com",
3   "card_number": "4111111111111111",
4   "cvv": "123",
5   "expiration_year": "2025",
6   "expiration_month": "09"
7 }
```

Prueba técnica – Backend Gateway Pos

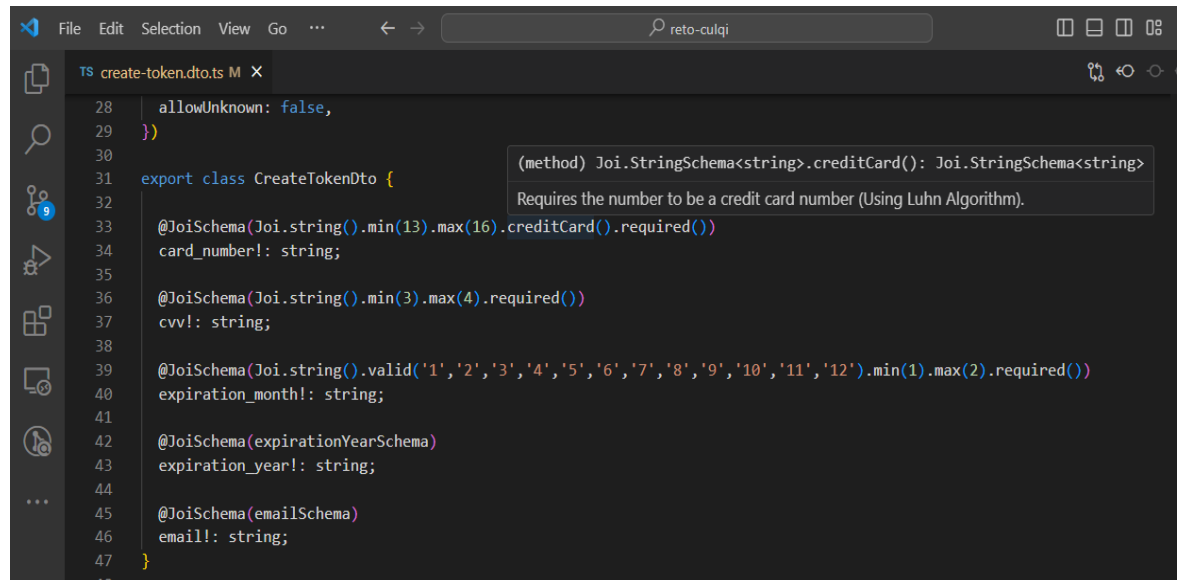
Parameter	Type	Restriction	Description
card_number	number	Length: 13...16	Utilizar el algoritmo de LUHN para garantizar que la tarjeta sea válida.
cvv	number	Length: 3...4	Visa / mastercard: 123 Amex: 4532
expiration_month	string	Length: 1..2	Del 1 a 12
expiration_year	string	Length: 4	Año actual máximo 5 años
email	string	Length: 5..100	Garantizar que sean email válidos utilizando los siguientes dominios "gmail.com", "hotmail.com", "yahoo.es".

src\tokens\dto\create-token.dto.ts

```
File Edit Selection View Go ... reto-culqi

TS create-token.dto.ts M X
You, 35 seconds ago | 1 author (You)
1 import { JoiSchema, JoiSchemaOptions } from 'nestjs-joi';
2 import * as Joi from 'joi';
3
4 const allowedDomains = ['gmail.com', 'hotmail.com', 'yahoo.es'];
5
6 const emailSchema = Joi.string().email().min(5).max(100).custom((value: string, helpers) => {
7   const domain = value.split('@')[1];
8   if (allowedDomains.includes(domain)) {
9     return value;
10  } else {
11    return helpers.error('any.invalid');
12  }
13 }, 'Custom validation').required();
14
15 const currentYear = new Date().getFullYear();
16 const maxYear = currentYear + 5;
17
18 const expirationYearSchema = Joi.string().max(4).custom((value: string, helpers) => {
19   const intValue = parseInt(value);
20   if (!isNaN(intValue) && intValue >= currentYear && intValue <= maxYear) {
21     return value;
22   } else {
23     return helpers.error('any.invalid');
24   }
25 }, 'Custom validation').required();
```

Prueba técnica – Backend Gateway Pos



```
28   allowUnknown: false,
29 })
30
31 export class CreateTokenDto {
32
33   @JoiSchema(Joi.string().min(13).max(16).creditCard().required())
34   card_number!: string;
35
36   @JoiSchema(Joi.string().min(3).max(4).required())
37   cvv!: string;
38
39   @JoiSchema(Joi.string().valid('1','2','3','4','5','6','7','8','9','10','11','12').min(1).max(2).required())
40   expiration_month!: string;
41
42   @JoiSchema(expirationYearSchema)
43   expiration_year!: string;
44
45   @JoiSchema(emailSchema)
46   email!: string;
47 }
```

(method) Joi.StringSchema<string>.creditCard(): Joi.StringSchema<string>
Requires the number to be a credit card number (Using Luhn Algorithm).

Si no logra pasar las validaciones, se debe interrumpir el proceso y arrojar una excepción.

```
{
  "message": "Request validation of body failed, because: \"cvv\" length must be at least 3 characters long",
  "error": "Bad Request",
  "statusCode": 400
}

{
  "message": "Request validation of body failed, because: \"expiration_year\" contains an invalid value",
  "error": "Bad Request",
  "statusCode": 400
}

{
  "message": "Request validation of body failed, because: \"expiration_month\" must be one of [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12]",
  "error": "Bad Request",
  "statusCode": 400
}

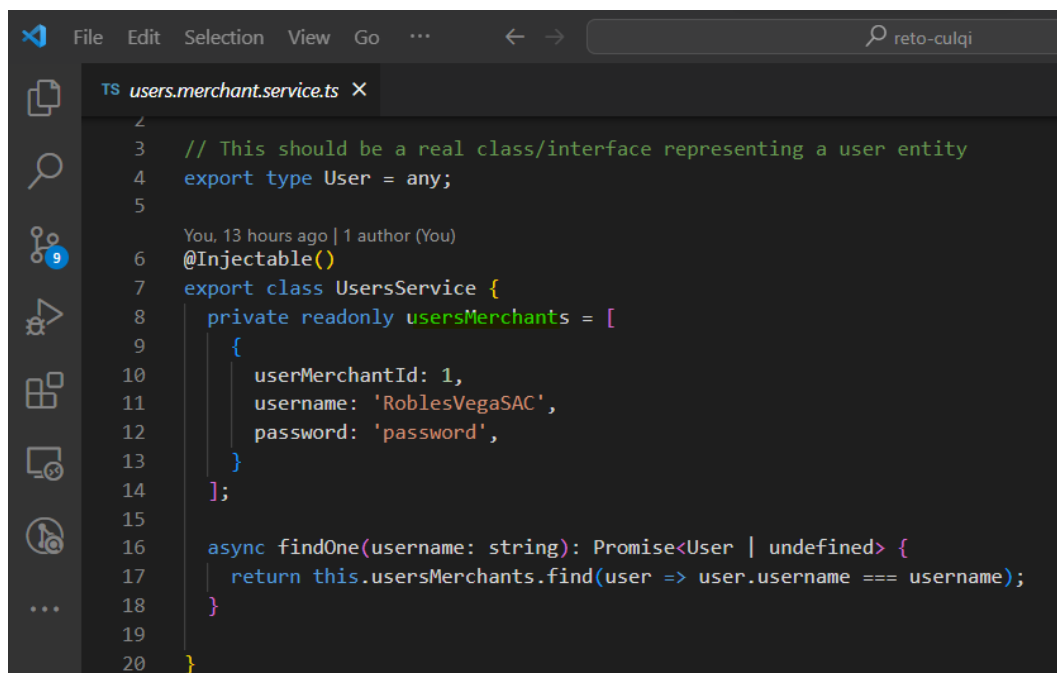
{
  "message": "Request validation of body failed, because: \"email\" contains an invalid value",
  "error": "Bad Request",
  "statusCode": 400
}
```


Prueba técnica – Backend Gateway Pos

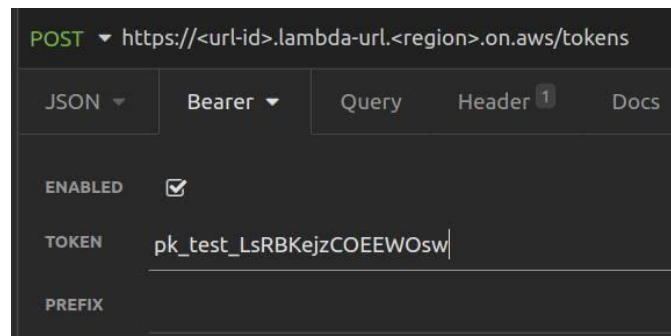
```
}  
... Otros
```

Headers del request:

Las pk son llaves que permiten identificar a los comercios dentro del API, asumir por simplicidad que siempre vamos a usar la misma llave en todos los request pero el sistema debe validar la presencia de la misma y hacer las validaciones necesarias (exista, formato, etc)



```
TS users.merchant.service.ts X  
2  
3 // This should be a real class/interface representing a user entity  
4 export type User = any;  
5  
6 You, 13 hours ago | 1 author (You)  
7 @Injectable()  
8 export class UsersService {  
9     private readonly usersMerchants = [  
10         {  
11             userMerchantId: 1,  
12             username: 'RoblesVegaSAC',  
13             password: 'password',  
14         }  
15     ];  
16  
17     async findOne(username: string): Promise<User | undefined> {  
18         return this.usersMerchants.find(user => user.username === username);  
19     }  
20 }
```

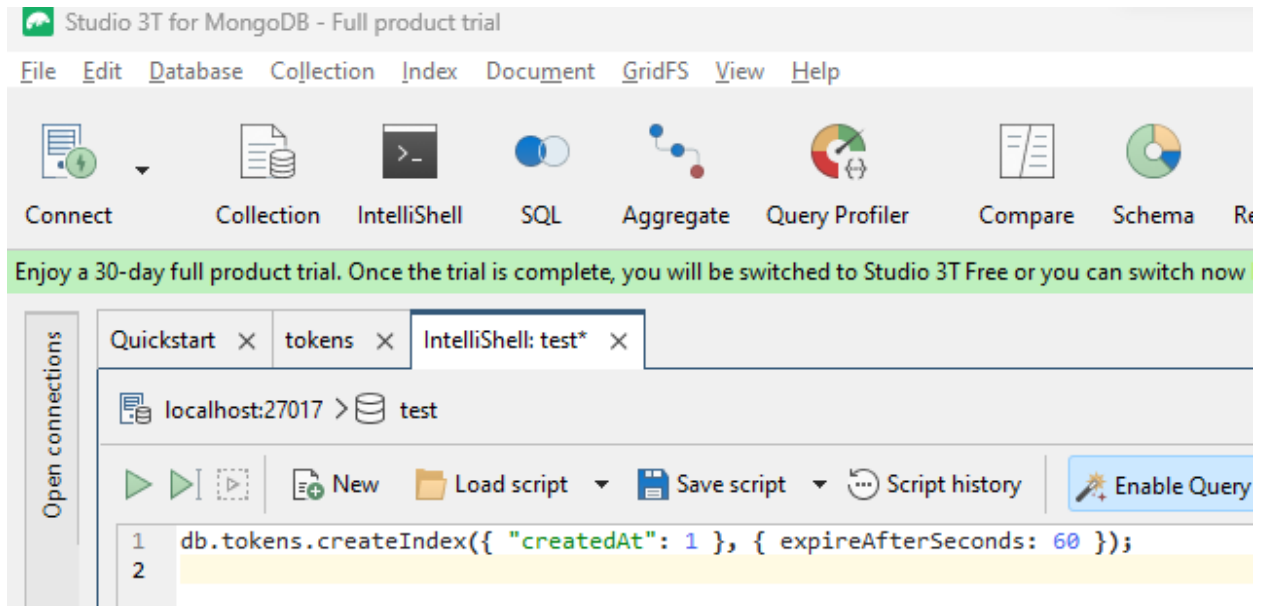


```
POST https://<url-id>.lambda-url.<region>.on.aws/tokens  
JSON Bearer Query Header 1 Docs  
ENABLED ☒  
TOKEN pk_test_LsRBKejzCOEEWOsw  
PREFIX
```

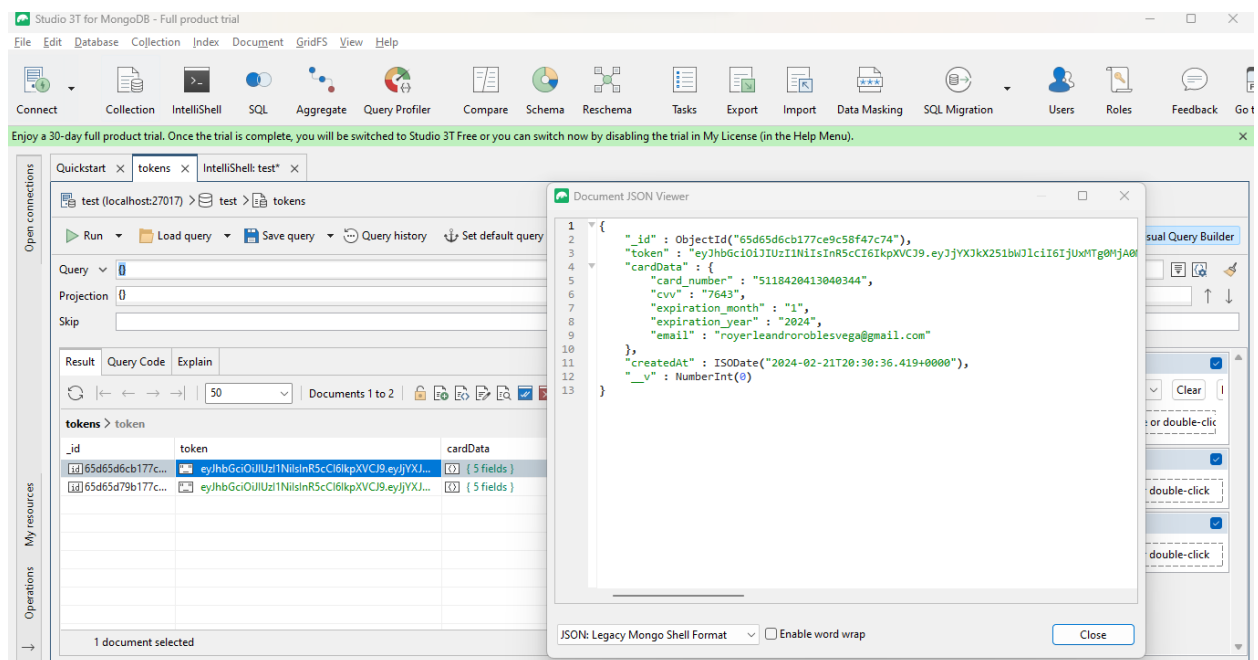
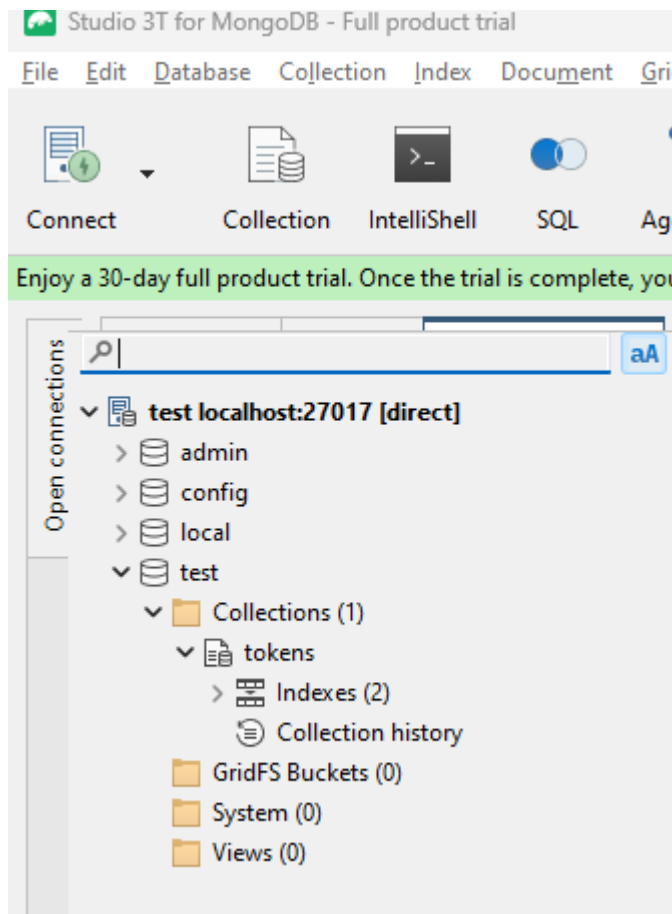
Con los datos recibidos por petición deberá almacenar todos los datos de la petición más el token auto generado en una base de datos no relacional, con una espiración de 1 min (la espiración debe ser automática)

Prueba técnica – Backend Gateway Pos

```
db.tokens.createIndex({ "createdAt": 1 }, { expireAfterSeconds: 60 });
```



Prueba técnica – Backend Gateway Pos



II. Obtener datos de tarjeta

Se debe crear un método para obtener los datos de la tarjeta según los siguientes parámetros.

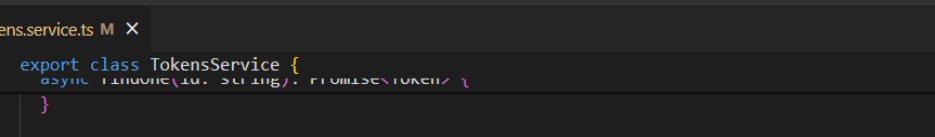
Prueba técnica – Backend Gateway Pos

a. token

Obtener los datos de la tarjeta almacenados en el paso anterior, según el token. Este endpoint sólo debe funcionar para los PK (token) válidos

La respuesta del método debe retornar solo los datos de la tarjeta (sin CVV) en caso el elemento ya no esté presente (porque expiró) debe retornar una respuesta coherente indicando la espiración.

Remove CVV



```
9 export class TokensService {
10     async findOne(token: string): Promise<Token> {
11         // ...
12     }
13
14     async findOneToken(token: string): Promise<any> {
15         const foundToken = await this.tokenModel.findOne({ token: token }).select('-cardData.cvv').exec();
16
17         if (!foundToken) {
18             throw new HttpException('Token expirado, genere un nuevo token', HttpStatus.FORBIDDEN);
19         }
20
21         return foundToken;
22     }
23 }
```

POST

Generar Token User

POST

Create Token

POST

Search By Token

+

▼

No Environment

▼

📄

API Reto Culqi / Token / Search By Token

Save ▼

🔗

💬

POST ▼

http://localhost:3001/tokens/search

Send ▼

Params

Authorization ●

Headers (10)

Body ●

Pre-request Script

Tests

Settings

Cookies

</>

● none

● form-data

● x-www-form-urlencoded

● raw

● binary

● GraphQL

JSON ▼

Beautify

🔍

```

1  {
2    "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJyYXJkX251bWJlcCI6IjUxMTg0MjA0MTMwNDZlLCJpdnYiOiI3NjQzIiwiaXNjaXJhdGlvbi9tY250aCI6IjE1LCJleHBpcmF0aW9uX31lYXkiOiIyMDI0IiwiaWZw1haWwiOiIybj31cmx1YW5kcm9yb2J3sZXN2ZwdhQGdtYWIzLmNvbSIsIm1hdCI6MTcwODU1MDQ1NCwiZXhwIjoxNzA4NTUwNTE0fQ.VEZz0FEQv5yPfxfwKQUOPjr1cUcKKNGRt-MCKySLRZE"
3  }

```

Body

Cookies

Headers (7)

Test Results

🌐

Status: 403 Forbidden

Time: 26 ms

Size: 310 B

📄

Save as example

⋮

Pretty

Raw

Preview

Visualize

JSON ▼

🔗

📄

🔍

```

1  {
2    "statusCode": 403,
3    "message": "Token expirado, genere un nuevo token"
4  }

```

```
{
  "statusCode": 403,
  "message": "Token expirado, genere un nuevo token"
}
```

Prueba técnica – Backend Gateway Pos

4. Consideraciones al enviar la prueba

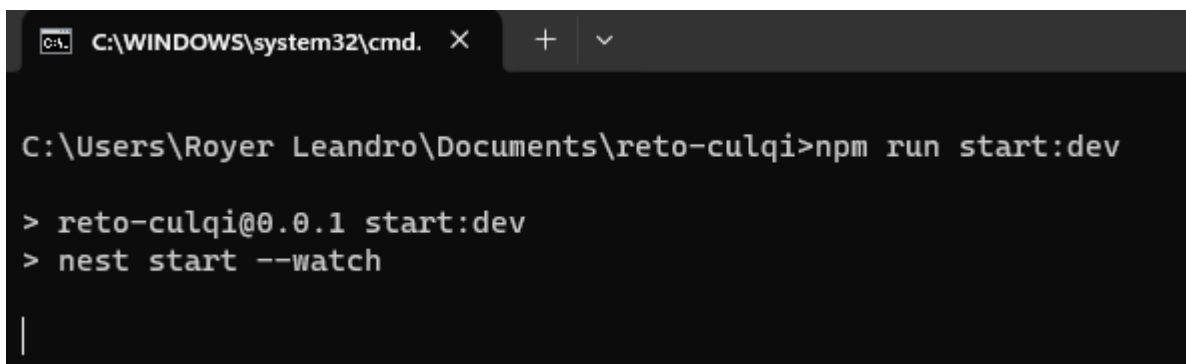
1. Seguir todos las buenas prácticas, el código debe ser legible y bien estructurado. **(Indispensable)**
2. No considerar el desarrollo de un Front End como se muestra en la figura inicial, la interacción sería entre los Request que se pasen por el postman contra el Backend. **(Indispensable)**

Revisar el archivo API Reto Culqi Royer Robles.postman_collection.json

3. Se debe escribir 3 pruebas unitarias como mínimo por cada servicio a desarrollar. Esto debe de realizarse con Jest.
4. Se debe de compartir las rutas del repositorio en donde se subirá el código, asimismo se debe de enviar un word con las respuestas de las pregunta 2 si se llegase a realizar. **(Deseable)**

Run

```
npm run start | npm run start:dev
```



```
C:\WINDOWS\system32\cmd. X + v  
  
C:\Users\Royer Leandro\Documents\reto-culqi>npm run start:dev  
  
> reto-culqi@0.0.1 start:dev  
> nest start --watch  
  
|
```

Prueba técnica – Backend Gateway Pos

```
C:\WINDOWS\system32\cmd. X + v
[14:34:25] Starting compilation in watch mode...
[14:34:51] Found 0 errors. Watching for file changes.

[Nest] 18856 - 21/02/2024, 14:35:37 LOG [NestFactory] Starting Nest application...
[Nest] 18856 - 21/02/2024, 14:35:37 LOG [InstanceLoader] MongooseModule dependencies initialized +98ms
[Nest] 18856 - 21/02/2024, 14:35:37 LOG [InstanceLoader] UsersModule dependencies initialized +1ms
[Nest] 18856 - 21/02/2024, 14:35:37 LOG [InstanceLoader] JwtModule dependencies initialized +1ms
[Nest] 18856 - 21/02/2024, 14:35:37 LOG [InstanceLoader] AppModule dependencies initialized +1ms
[Nest] 18856 - 21/02/2024, 14:35:37 LOG [InstanceLoader] JoiPipeModule dependencies initialized +0ms
[Nest] 18856 - 21/02/2024, 14:35:37 LOG [InstanceLoader] AuthModule dependencies initialized +1ms
[Nest] 18856 - 21/02/2024, 14:35:37 LOG [InstanceLoader] MongooseCoreModule dependencies initialized +41ms
[Nest] 18856 - 21/02/2024, 14:35:37 LOG [InstanceLoader] MongooseModule dependencies initialized +21ms
[Nest] 18856 - 21/02/2024, 14:35:37 LOG [InstanceLoader] TokensModule dependencies initialized +1ms
[Nest] 18856 - 21/02/2024, 14:35:37 LOG [RoutesResolver] AppController {/}: +134ms
[Nest] 18856 - 21/02/2024, 14:35:37 LOG [RoutesResolver] AuthController {/auth/merchant}: +1ms
[Nest] 18856 - 21/02/2024, 14:35:37 LOG [RouterExplorer] Mapped {/auth/merchant/login, POST} route +2ms
[Nest] 18856 - 21/02/2024, 14:35:37 LOG [RouterExplorer] Mapped {/auth/merchant/profile, GET} route +1ms
[Nest] 18856 - 21/02/2024, 14:35:37 LOG [RoutesResolver] TokensController {/tokens}: +0ms
[Nest] 18856 - 21/02/2024, 14:35:37 LOG [RouterExplorer] Mapped {/tokens, POST} route +1ms
[Nest] 18856 - 21/02/2024, 14:35:37 LOG [RouterExplorer] Mapped {/tokens, GET} route +1ms
[Nest] 18856 - 21/02/2024, 14:35:37 LOG [RouterExplorer] Mapped {/tokens/:id, GET} route +1ms
[Nest] 18856 - 21/02/2024, 14:35:37 LOG [RouterExplorer] Mapped {/tokens/search, POST} route +0ms
[Nest] 18856 - 21/02/2024, 14:35:37 LOG [NestApplication] Nest application successfully started +5ms
Application is running on: http://[::1]:3001
```

Build

```
npm run build
```

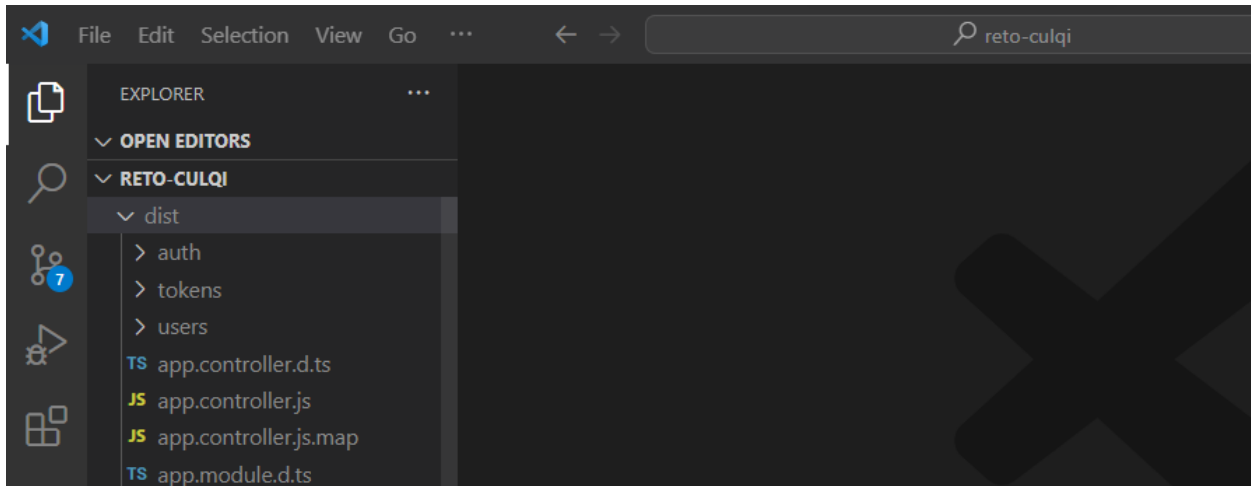
```
C:\Users\Royer Leandro\Documents\reto-culqi>npm run build

> reto-culqi@0.0.1 prebuild
> rimraf dist

> reto-culqi@0.0.1 build
> nest build

C:\Users\Royer Leandro\Documents\reto-culqi>
```

Prueba técnica – Backend Gateway Pos



Tests

```
npm run test
```

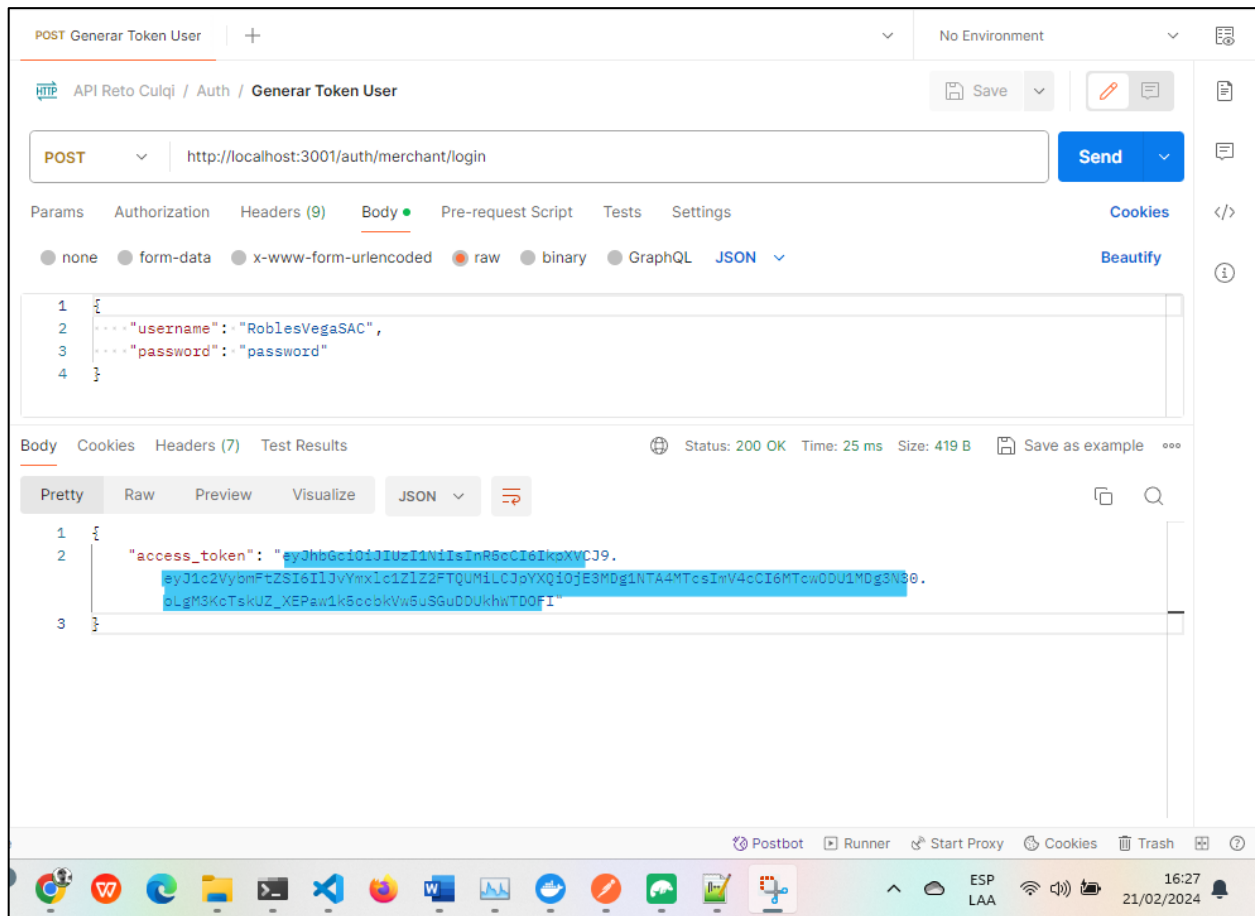
```
Símbolo del sistema x + v

C:\Users\Royer Leandro\Documents\reto-culqi>npm run test

> reto-culqi@0.0.1 test
> jest

PASS src/users/users.merchant.service.spec.ts (41.98 s)
PASS src/app.controller.spec.ts (46.917 s)
PASS src/app.service.spec.ts (47.553 s)
PASS src/tokens/tokens.controller.spec.ts (52.568 s)
PASS src/tokens/tokens.service.spec.ts (54.07 s)

Test Suites: 5 passed, 5 total
Tests:       7 passed, 7 total
Snapshots:   0 total
Time:        59.155 s
Ran all test suites.
```



Prueba técnica – Backend Gateway Pos

2. Crear Token

The screenshot displays the Postman application interface for a POST request named "Create Token". The URL is set to `http://localhost:3001/tokens`. The "Authorization" tab is selected, showing a "Bearer Token" type. A warning message states: "Heads up! These parameters hold sensitive data. To keep this data secure while working in a collaborative environment, we recommend using variables. Learn more about [variables](#)." The token value `eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ...` is visible in the "Token" field. The "Response" section shows a cartoon astronaut and the text "Click Send to get a response". The bottom status bar includes icons for Postbot, Runner, Start Proxy, Cookies, Trash, and system information (ESP LAA, 16:29, 21/02/2024).

POST Create Token + No Environment

API Reto Culqi / Token / **Create Token** Save

POST `http://localhost:3001/tokens` Send

Params **Authorization** Headers (10) Body Pre-request Script Tests Settings Cookies

Type Bearer T...
The authorization header will be automatically generated when you send the request. Learn more about [Bearer Token](#) authorization.

Heads up! These parameters hold sensitive data. To keep this data secure while working in a collaborative environment, we recommend using variables. Learn more about [variables](#).

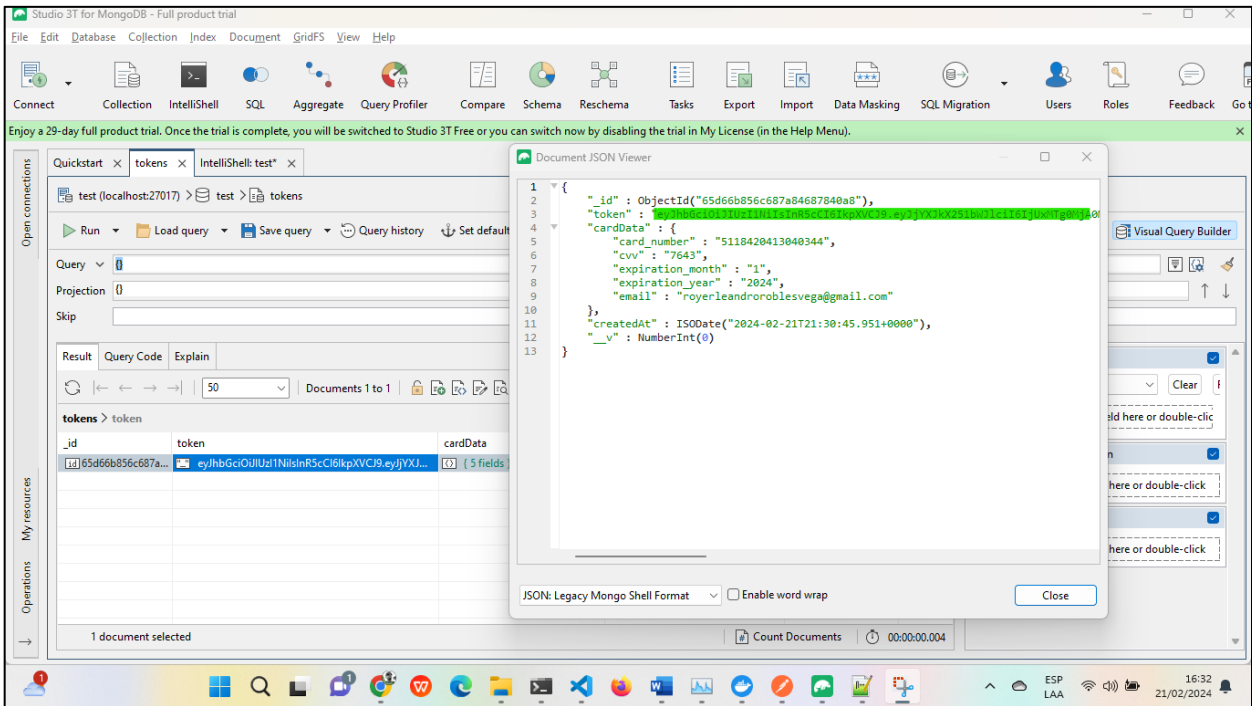
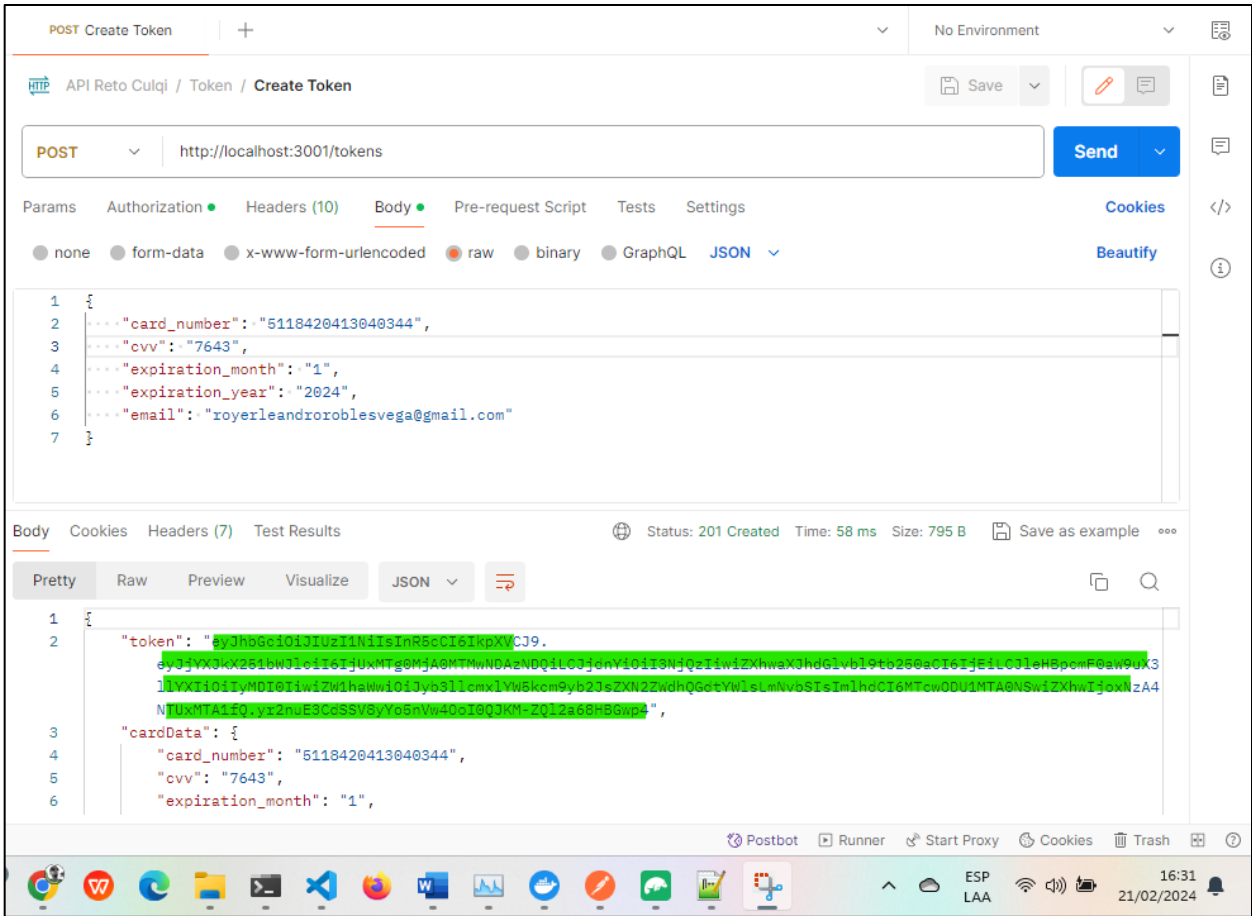
Token `eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ...`

Response
Click Send to get a response

Postbot Runner Start Proxy Cookies Trash 16:29 21/02/2024

Prueba técnica – Backend

Gateway Pos



3. Search By Token

POST Search By Token

+

No Environment

API Reto Culqi / Token / Search By Token

Save

POST

http://localhost:3001/tokens/search

Send

Params

Authorization •

Headers (10)

Body •

Pre-request Script

Tests

Settings

Cookies

</>

Type

Bearer T...

The authorization header will be automatically generated when you send the request. Learn more about [Bearer Token](#) authorization.

Token

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.ey...

Heads up! These parameters hold sensitive data. To keep this data secure while working in a collaborative environment, we recommend using variables. Learn more about [variables](#).

Body

Cookies

Headers (7)

Test Results

Status: 200 OK Time: 19 ms Size: 777 B Save as example ...

Pretty

Raw

Preview

Visualize

JSON

1 {

2 "cardData": {

3 "card_number": "5118420413040344",

4 "expiration_month": "1",

5 "expiration_year": "2024",

6 "email": "royerxleandroroblesvega@gmail.com"

7 },

8 "_id": "65d66cab6c687a84687840aa",

9 "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.

eyJ0eXAiOiJKV1QiLCJhbGciOiJSemVuc2NyaWVsZXNjaXhdGvbl9tb250aCI6IjE1Cj1leHspcmF0aW50ZS1NYXkiOiJlNDI0IiwiaWF0IjoiY3llbm91cyB1cmx1YW5kcm9ybzIsc2lnaHQGeTYwIjwiNmV5SiImIHR5cCI6MTcwODU1MThzOSwiZXhwIjozMTA4NTUxMzk6IGQubSI6IjQ9oyFZoSkbBsuLTwTzr-9Srs1OT0ScM.w",

10 "createdAt": "2024-02-21T21:35:39.676Z",

Postbot

Runner

Start Proxy

Cookies

Trash

16:36

21/02/2024

Prueba técnica – Backend Gateway Pos

