



DLYTIME

Base de datos
Kevin Aroca
Sebastián Rodríguez
Gerson Corredor
Samuel Prieto

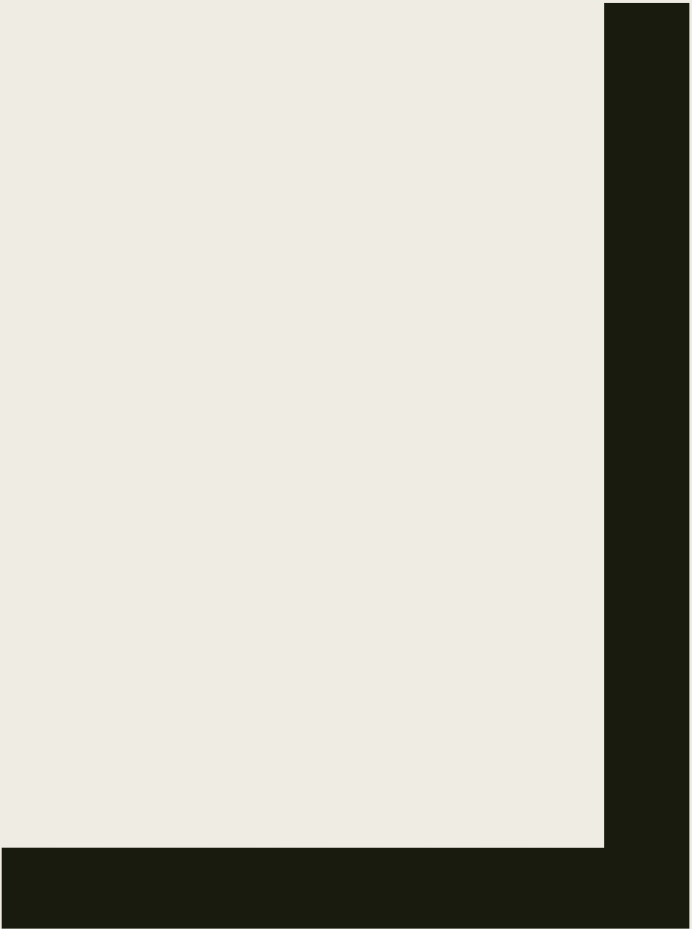


Tabla de contenido

API Interna	API Externa	

¿CUÁL ES EL PROBLEMA?

tusLentes Shop es un negocio ubicado en la localidad de Kennedy, especializado en brindar servicios de asesoría y venta de artículos para la vista. A través de las técnicas de levantamiento de información, se identificó un problema: agendan sus citas por redes sociales y medios físicos (cuadernos y libretas). Sin embargo, estos procesos no son eficientes en el agendamiento. En WhatsApp, se pueden borrar los mensajes, y las anotaciones en un cuaderno se pueden extraviar o incluso dañar, lo que resulta en la pérdida de información. En una ocasión, un empleado se confundió al llamar al cliente para avisarle sobre la cita. Más tarde, dos clientes llegaron a la misma hora, generando confusión entre ellos. Uno de los clientes se quejó por la mala administración de las citas, lo que llevó a cancelar la cita con ellos.

A thick black L-shaped frame is positioned on the left and bottom edges of the slide, framing the content.

OBJETIVO GENERAL

Desarrollar un sistema de información orientado al
agendamiento de citas con el fin de mejorar la gestión de
clientes en el negocio tusLentes shop.

API INTERNA



Envío de Correos electrónicos

¿Cuál es su objetivo?

- Enviar información por medio de un correo electrónico estableciendo una parte del proceso de recuperación de contraseña.

```
/* Prueba de Api de enviar correos */  
const transporter = nodemailer.createTransport({  
  service: "gmail",  
  auth: {  
    user: "dlytime987@gmail.com", /* Correo */  
    pass: "cvqn rxuh gkif quww", /* Contraseña de Aplicacion */  
  },  
});
```

Requisitos para el uso

- Se requiere una cuenta de correo electrónico para realizar los envíos y configurarla.

¿Qué método se utiliza?

- El método a utilizar es el POST, debido a que se utiliza un correo ingresado por el usuario.

```
await Axios.post("http://:3001/Enviar-correo", {  
  to: forgotEmail,  
  subject: "Restablecimiento de Contraseña",  
});
```



CLIENTE

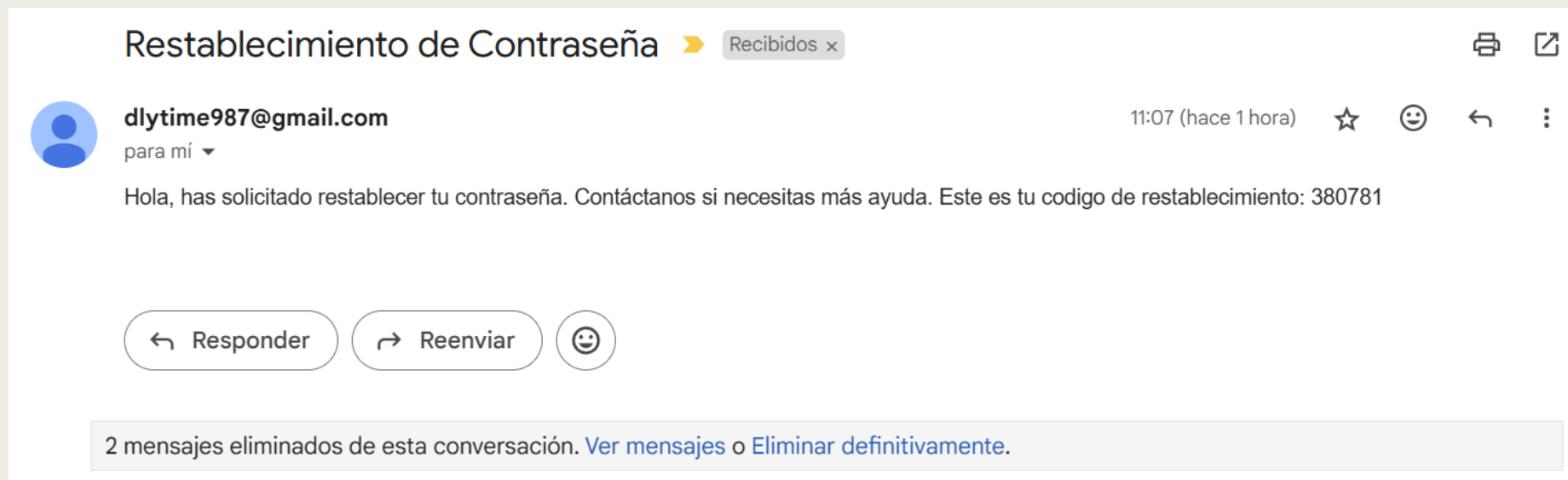
SERVER



```
const mailOptions = {  
  from: "dlytime987@gmail.com",  
  to, // Dirección proporcionada por el usuario  
  subject, // Asunto del correo  
  text: `Hola, has solicitado restablecer tu contraseña. Contáctanos si necesitas más ayuda. Este es tu código de restablecimiento: ${code}`,  
};  
  
transporter.sendMail(mailOptions, (error, info) => {  
  if (error) {  
    console.error(error);  
    console.log(mailOptions);  
    return res.status(500).send("Error enviando el correo");  
  }  
  res  
    .status(200)  
    .send("Correo enviado con éxito, Ingresar código temporal enviado.");  
});
```

Consumo

- La API se implementará en los apartados de recuperación de contraseña siendo una parte fundamental el envío del código temporal.



Pruebas con thunder

POST ▼ http://localhost:3001/Enviar-correo Send

Query Headers ² Auth Body ¹ Tests Pre Run

JSON XML Text Form Form-encode GraphQL Binary

JSON Content Format

```
1 {  
2   "to": "sebitasrodriguez286@gmail.com",  
3   "subject": "Restablecimiento de Contraseña"  
4 }
```

Status: **200 OK** Size: **61 Bytes** Time: **2.07 s**

Response Headers ⁷ Cookies Results Docs

1 Correo enviado con éxito, Ingresar código temporal enviado.

POST ▼ http://localhost:3001/Enviar-correo Send

Query Headers ² Auth Body ¹ Tests Pre Run

JSON XML Text Form Form-encode GraphQL Binary

JSON Content Format

```
1 {  
2   "to": "sebitasrodriguez286@gmail.",  
3   "subject": "Restablecimiento de Contraseña"  
4 }
```

Status: **500 Internal Server Error** Size: **24 Bytes** Time: **573 ms**

Response Headers ⁷ Cookies Results Docs

1 Error enviando el correo

Gestión de empleados

¿Cuál es su objetivo?

- Gestionar los datos del personal, implementando funciones para la creación de nuevos registros, consulta detallada de información y modificación de datos existente.

¿Qué método se utiliza?

- Debido a que su función principal es realizar creación, consultas y modificaciones, se establece solicitudes GET, POST, Y PATCH

Consulta de empelados

```
/* Crud Empleados */
/* Consulta los Usuarios que son empleados */
app.get("/crud_empleados_referencia", (req, res) => {
  db.query("select * from persona where idRol = 2", (err, result) => {
    if (err) {
      console.log(err);
    } else {
      res.send(result);
    }
  });
});
```

```
/* Agregar Nuevos empleados */
app.post("/Crud_empleado_Registrar", async (req, res) => {
  const numeroDocumento = req.body.numeroDocumento;
  const idRol = req.body.idRol;
  const idTipoIdentificacion = req.body.idTipoIdentificacion;
  const Nombres = req.body.nombre;
  const Apellidos = req.body.apellido;
  const idGenero = req.body.idGenero;
  const correo = req.body.correo;
  const clave = req.body.clave;
  const telefono = req.body.telefono;
  const estado = req.body.estadoPersona;

  try {
    const hashedPassword = await bcryptjs.hash(clave, 10);
    // Guardar el usuario en la base de datos
    db.query(
      "INSERT INTO persona (numeroDocumento, idRol, idTipoIdentificacion, Nombres, Apellidos, idGenero, correo, telefono, hashedPassword, estado, "
      [
        numeroDocumento,
        idRol,
        idTipoIdentificacion,
        Nombres,
        Apellidos,
        idGenero,
        correo,
        telefono,
        hashedPassword,
        estado,
      ], // Usamos hashedPassword en lugar de clave
      (err, result) => {
        if (err) {
          return res.status(500).send("Error al registrar el empleado");
        } else {
          res.send(result);
        }
      }
    );
  }
});
```

REGISTRO DE EMPLEADOS

```

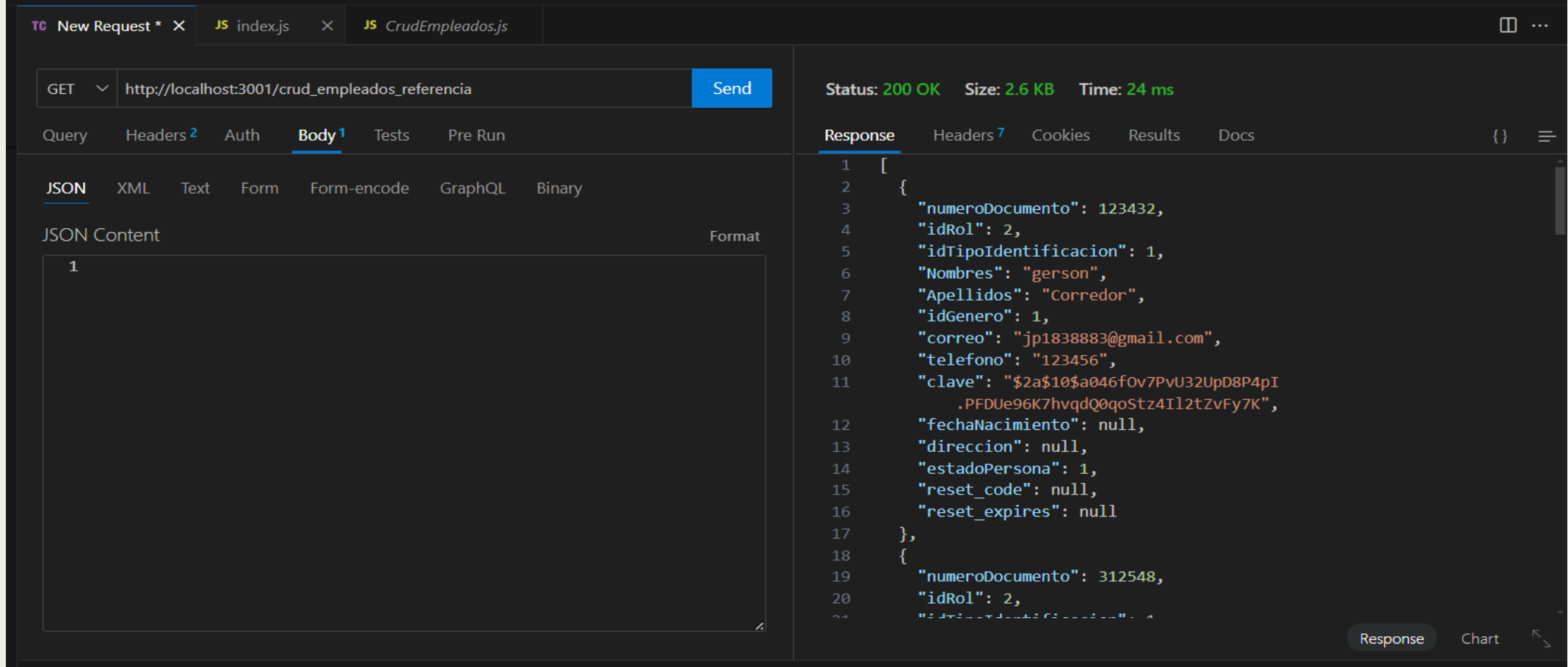
/* Actualizar Empleado */
app.patch("/ActualizarEmpleado", async (req, res) => {
  const numeroDocumento = req.body.numeroDocumento;
  const idTipoIdentificacion = req.body.idTipoIdentificacion;
  const nombres = req.body.nombre;
  const apellidos = req.body.apellido;
  const idGenero = req.body.idGenero;
  const correo = req.body.correo;
  const telefono = req.body.telefono;
  const estado = req.body.estadoPersona;

  console.log("funciona");

  db.query(
    "UPDATE persona SET idTipoIdentificacion = ?, Nombres = ?, Apellidos = ?, idGenero = ?, correo = ?, telefono = ?, estadoPersona = ?"
    [
      idTipoIdentificacion,
      nombres,
      apellidos,
      idGenero,
      correo,
      telefono,
      estado,
      numeroDocumento,
    ],
    (err, result) => {
      if (err) {
        console.error(err);
        res.status(500).send({ error: "Error al actualizar el empleado" });
      } else {
        res.send(result);
      }
    }
  )
})

```

ACTUALIZAR O MODIFICAR EMPLEADOS



PRUEBAS CON THUNDER

Método GET

POST

http://localhost:3001/Crud_empleado_Registrar

Send

Query

Headers2

Auth

Body1

Tests

Pre Run

JSON

XML

Text

Form

Form-encode

GraphQL

Binary

JSON Content

Format

```
1 {
2   "numeroDocumento": 7665544,
3   "idRol": 2,
4   "idTipoIdentificacion": 1,
5   "nombre": "gerson",
6   "apellido": "Corredor",
7   "idGenero": 1,
8   "correo": "jp1838883@gmail.com",
9   "clave": "hola",
10  "telefono": 123456,
11  "estadoPersona": true
12 }
```

Status: 200 OK

Size: 127 Bytes

Time: 85 ms

Response

Headers7

Cookies

Results

Docs

```
1 {
2   "fieldCount": 0,
3   "affectedRows": 1,
4   "insertId": 0,
5   "serverStatus": 2,
6   "warningCount": 0,
7   "message": "",
8   "protocol41": true,
9   "changedRows": 0
10 }
```

MÉTODO POST

PATCH

▼

http://localhost:3001/ActualizarEmpleado

Send

Query

Headers 2

Auth

Body 1

Tests

Pre Run

JSON

XML

Text

Form

Form-encode

GraphQL

Binary

JSON Content

Format

```
1 {
2   "numeroDocumento": 123432,
3   "idRol": 2,
4   "idTipoIdentificacion": 1,
5   "nombre": "Fabian",
6   "apellido": "Sanabria",
7   "idGenero": 1,
8   "correo": "sebitasrodriguez286@gmail.com",
9   "clave": "hola",
10  "telefono": 123456,
11  "estadoPersona": true
12 }
```

Status: 200 OK

Size: 168 Bytes

Time: 18 ms

Response

Headers 7

Cookies

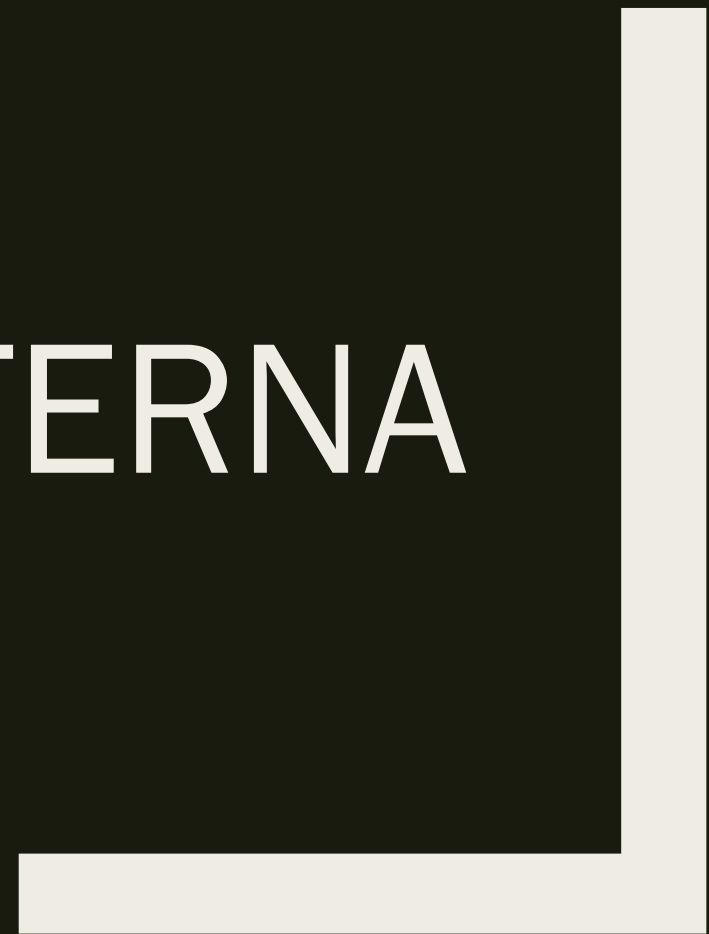
Results

Docs

```
1 {
2   "fieldCount": 0,
3   "affectedRows": 1,
4   "insertId": 0,
5   "serverStatus": 2,
6   "warningCount": 0,
7   "message": "(Rows matched: 1  Changed: 1  Warnings: 0)",
8   "protocol41": true,
9   "changedRows": 1
10 }
```

MÉTODO PATCH

API EXTERNA



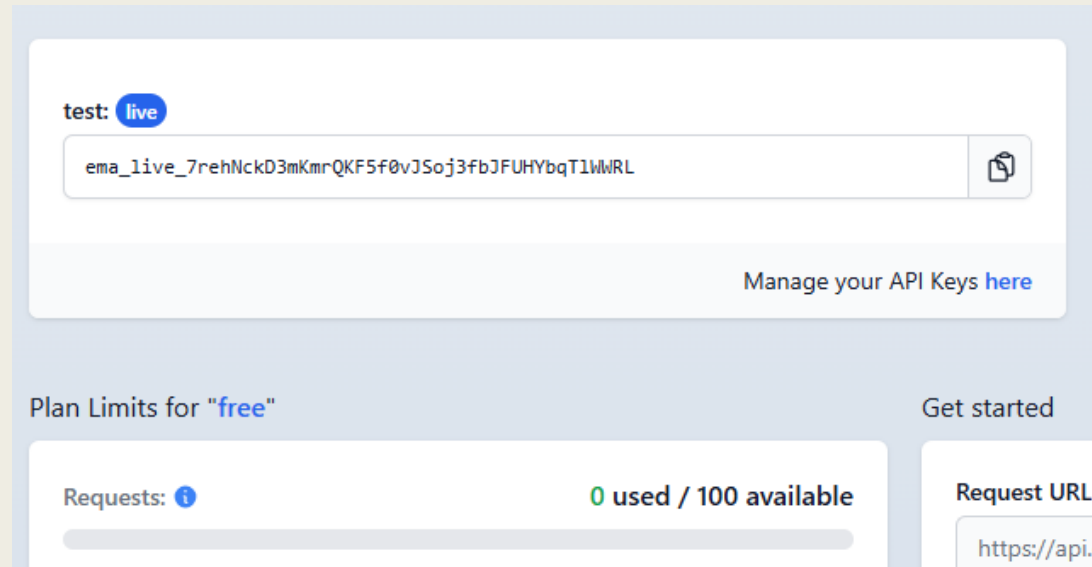
Emailvalidation

¿Cuál es su objetivo?

- Recuperar información sobre cualquier dirección de correo electrónico, estableciendo una forma de validar si un correo electrónico sea existente.

Requisitos para el uso

- Para iniciar se solicita un registro de API key, siendo la forma para empezar el consumo de la API externa.



¿Qué método se utiliza?

- Al comentario de realizar el consumo, se está realizando una consulta estableciendo que el método a utilizar es el GET, aunque la API brinda otra opción para autenticar las solicitudes por medio del encabezado HTTP.

```
39  /* metodo get */
40  const client = new Emailvalidation('ema_live_6WmdRIZwQrF3ji7fd4X89YctsfGTBvGUa9L9JqsX');
41
42  client.info(correo, { catch_all: 0 })
43  .then(response => {
44    console.log(response)
45    console.log(response.smtp_check)
```

```
main.936e7a3a9c25407...25.hot-update.js:70
{email: 'jhngilramos@gmail.com', user: 'jhngilramos', tag: '', domain:
▼ 'gmail.com', smtp_check: true, ...} ⓘ
  catch_all: null
  did_you_mean: ""
  disposable: false
  domain: "gmail.com"
  email: "jhngilramos@gmail.com"
  format_valid: true
  free: true
  mx_found: true
  reason: "valid_mailbox"
  role: false
  score: 0.64
  smtp_check: true
  state: "deliverable"
  tag: ""
  user: "jhngilramos"
▶ [[Prototype]]: Object
```

- En el momento de hacer el consumo, la API retornará datos en formato JSON ya con validación establecida.

Consumo

- La API se implementará en los apartados de registrar algún usuario, estableciendo validación en cada inserción a la base de datos evitando correos falsos.

localhost:3000 dice
Empleado Registrado

Aceptar

¿Ya tienes una cuenta?

Iniciar Sesión

Registrarse

gerson

corredor

jhgilramos@gmail.com

Hombre

C.C

34234124412

...

...

Registrarse

El correo es válido.

Pruebas con thunder

The screenshot displays the Thunder client interface with a request and response view. The request is a GET call to `https://api.emailvalidation.io/v1/status?apikey=ema_live_6WmdRIZwQrF3ji7fd4X89V`. The response is a 200 OK status with 134 bytes of data in 1.45 seconds. The response body is a JSON object containing account information and quotas.

Request:

- Method: GET
- URL: `https://api.emailvalidation.io/v1/status?apikey=ema_live_6WmdRIZwQrF3ji7fd4X89V`
- Body: Empty (JSON Content)

Response:

```
1 {
2   "account_id": 377289137159540736,
3   "quotas": {
4     "month": {
5       "total": 100,
6       "used": 49,
7       "remaining": 51
8     },
9     "grace": {
10      "total": 0,
11      "used": 0,
12      "remaining": 0
13    }
14  }
15 }
```

GET

https://api.emailvalidation.io/v1/info?apikey=ema_live_6WmdRIZwQrF3ji7fd4X89Yct

Send

Query

Headers 3

Auth

Body 1

Tests

Pre Run

JSON

XML

Text

Form

Form-encode

GraphQL

Binary

JSON Content

Format

1 {

2 "email": "jp1838883@gmail.com"

3 }

Status: 200 OK Size: 273 Bytes Time: 5.05 s

Response

Headers 24

Cookies

Results

Docs

1 {

2 "email": "jp1838883@gmail.com",

3 "user": "jp1838883",

4 "tag": "",

5 "domain": "gmail.com",

6 "smtp_check": true,

7 "mx_found": true,

8 "did_you_mean": "",

9 "role": false,

10 "disposable": false,

11 "score": 0.64,

12 "state": "deliverable",

13 "reason": "valid_mailbox",

14 "free": true,

15 "format_valid": true,

16 "catch_all": null

17 }

Response

Chart

GET https://api.emailvalidation.io/v1/info?apikey=ema_live_6WmdRIZwQrF3ji7fd4X89Yct

Send

Query

Headers 3

Auth

Body 1

Tests

Pre Run

JSON

XML

Text

Form

Form-encode

GraphQL

Binary

JSON Content

Format

```
1 {
2   "email": "sebas21313112@gmail.com"
3 }
```

Status: 200 OK Size: 286 Bytes Time: 783 ms

Response

Headers 24

Cookies

Results

Docs

{}

```
1 {
2   "email": "sebas21313112@gmail.com",
3   "user": "sebas21313112",
4   "tag": "",
5   "domain": "gmail.com",
6   "smtp_check": false,
7   "mx_found": true,
8   "did_you_mean": "",
9   "role": false,
10  "disposable": false,
11  "score": 0.48,
12  "state": "undeliverable",
13  "reason": "invalid_mailbox",
14  "free": true,
15  "format_valid": true,
16  "catch_all": null
17 }
```

Response

Chart