**Descripción conceptual del proceso de registro en Dory**

**Objetivo:** Conocer cuáles fueron los métodos implementados para registrar usuarios en la aplicación web Dory

El proceso de registro de la plataforma Dory se lleva a cabo cuando los datos que el usuario ingresados en el formulario de registro o los que están en su cuenta de Google quedan almacenados en la tabla *usuarios* de la base de datos ***process.env.DB\_NAME*** y su cuenta queda verificada.

El nombre ***process.env.DB\_NAME*** hace referencia a una variable de entorno que almacena el nombre de la base de datos y se puede encontrar en el archivo ***/.env*** en el código de proyecto en un ambiente de desarrollo y en la configuración de la aplicación en Heroku en un ambiente de pruebas y de producción. Para más información sobre variables de entorno diríjase a las recetas “Receta - Explicación configuración de las variables de entorno para almacenar la configuración de la aplicación en el ambiente de desarrollo” y “Receta - Explicación configuración de las variables de entorno para ambiente de prueba o producción en Heroku”.

La cuenta queda verificada cuando en el campo ***estaVerificado*** de la tabla ***usuarios*** (Imagen 1. Tabla ususarios de la base de datos de la plataforma Dory.) de la base de datos ***process.env.DB\_NAME*** se almacena el entero 1.

En esta receta se presentan los dos métodos de registros implementados en la plataforma Dory:

* En el primer método se recibe la información del usuario a través de un formulario y se envía una solicitud al endpoint /api/usuario/create **API-REST** de la plataforma Dory para crear un nuevo registro.
* En el segundo método el usuario se autentica con su cuenta de Google. Se extrae la información básica de la misma y se envía una solicitud al endpoint /api/usuario/create **API-REST** de la plataforma Dory para crear un nuevo registro.

Los siguientes diagramas (imagen 1, imagen 2) muestran los procesos completos de registro usando formulario y autenticación con cuenta de Google. Los rectángulos azules en estos diagramas representan piezas de software que hacen parte de la plataforma Dory y los cuadros grises representan sistemas externos.

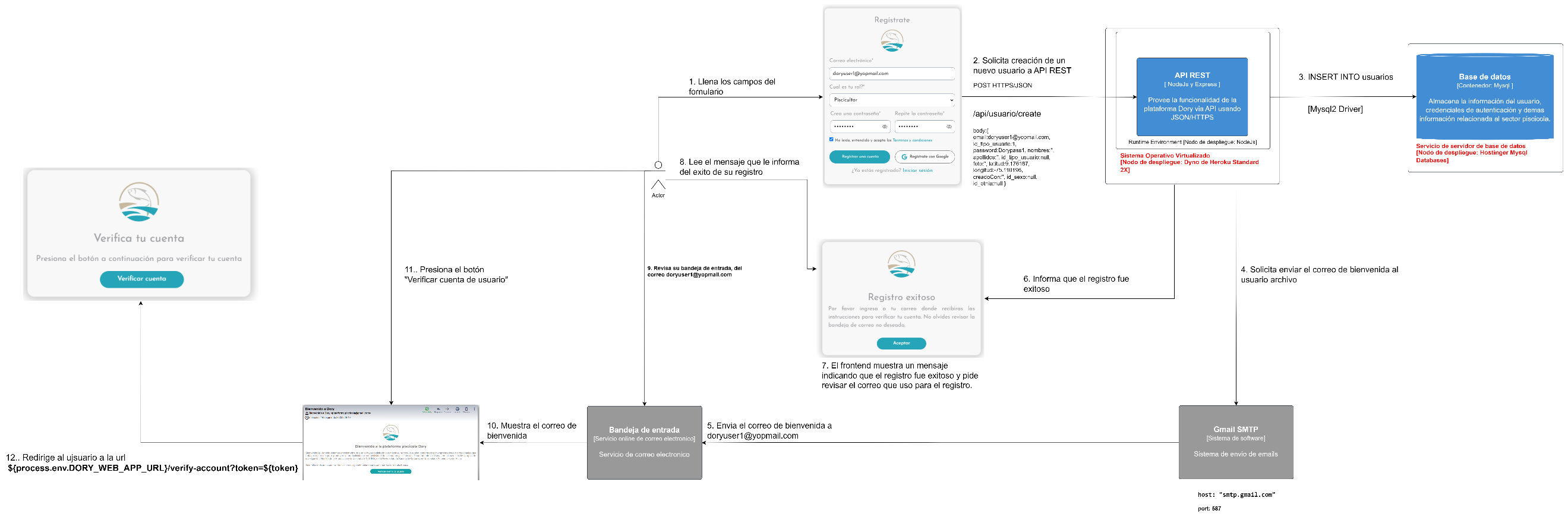


Imagen 1. Proceso completo de registro por formulario en la plataforma Dory

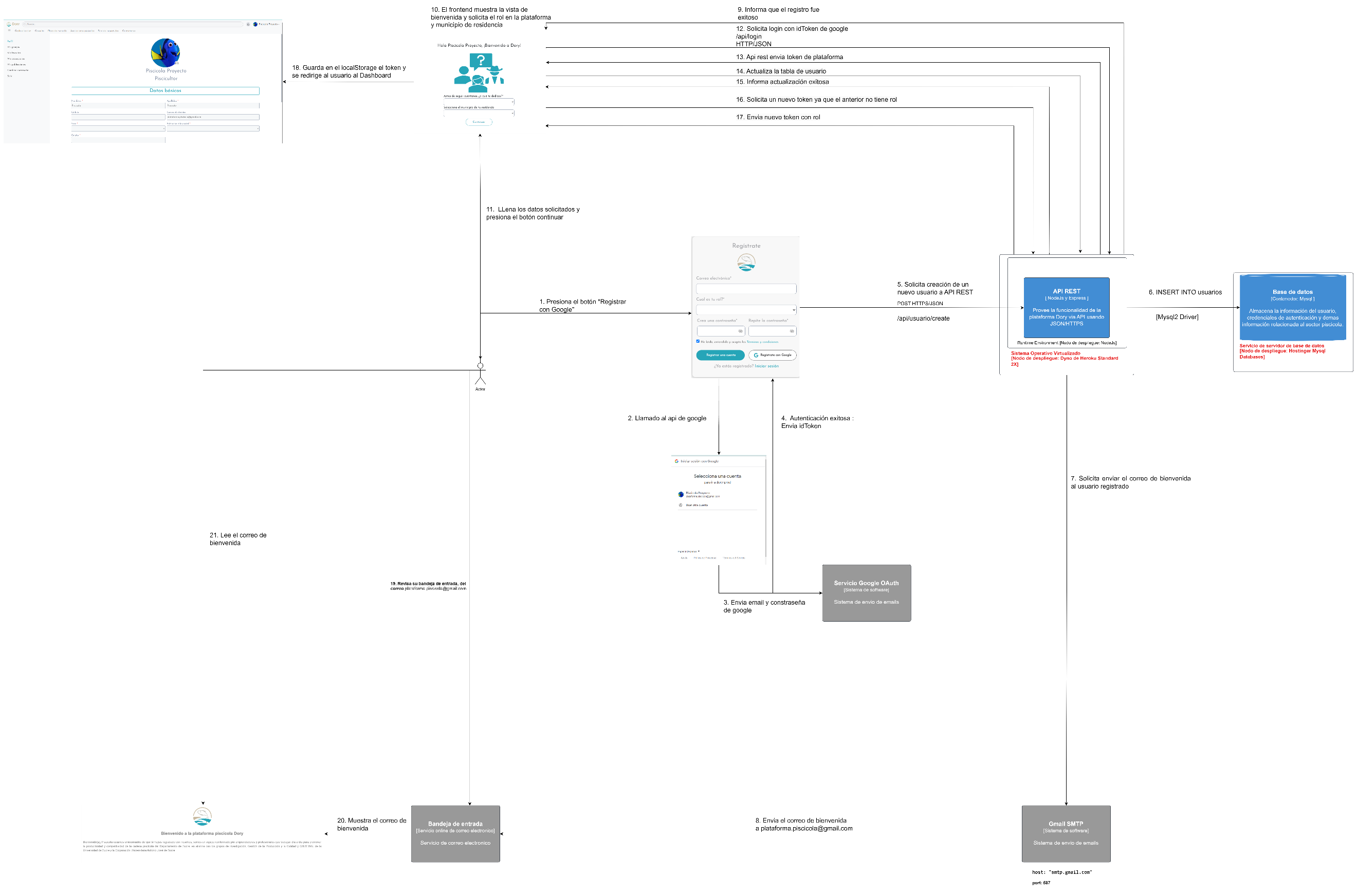


Imagen 2. Proceso de registro completo usando una cuenta de Google.

**Recursos necesarios:**

* Cliente Http
* Base de datos ***process.env.DB\_NAME*** en línea
* API-REST de Dory en línea
* Dory web app en línea
* Cuenta de Google(se usará para obtener el ID de Cliente OAuth 2.0)
* Ver la receta “Token JWT” para conocer sobre tokens JWT.

**Pasos:**

**0. Conocer la base de datos tabla usuarios**

La siguiente imagen muestra la tabla (tabla *usuarios*) donde se almacenarán los datos del registro. Para este proceso solo se necesitaran los campos *email, id\_tipo\_usuario, password*. Los demás datos se pueden llenar cuando el usuario se encuentre dentro de la plataforma.

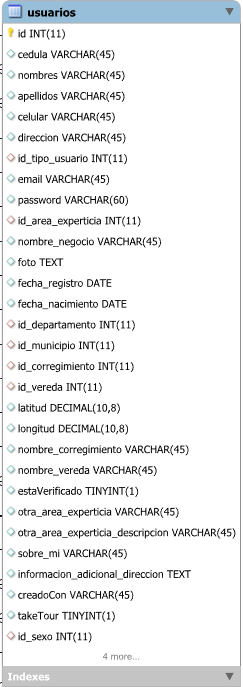


Imagen Tabla usuarios de la base de datos de la plataforma Dory.

**1. Conocer el método de registro usando formularios**

En la aplicación web Dory, se implementó un método de registro utilizando formularios estándar. Los pasos para el registro mediante formularios son los siguientes:

a. El usuario accede a la página de registro en la aplicación.



Imagen 4. Pagina home de la aplicación web Dory.

b. Se muestra un formulario que solicita al usuario ingresar su correo electrónico, rol en la plataforma y contraseña.

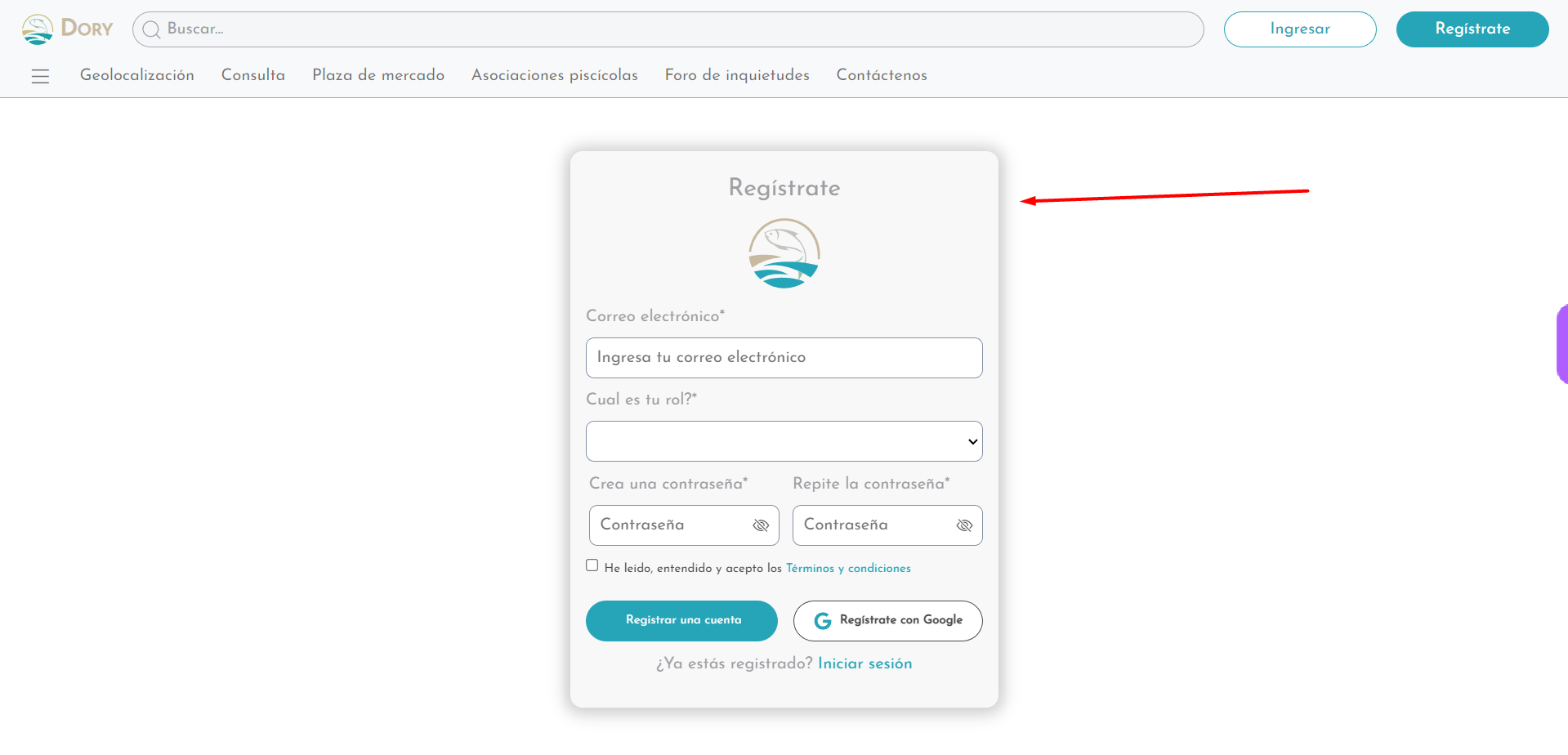


Imagen 5. Formulario de registro de la aplicación web Dory.

c. El usuario completa los campos del formulario con la información solicitada



Imagen 6. Datos del formulario ingresados por el usuario.

d. El componente RegistroComponent verifica los datos ingresados en el formulario y envía una solicitud al API REST para crear un nuevo usuario y este a su vez envía los datos a la base de datos. A continuación se muestra el proceso que ocurre después de enviar la solicitud de registro.

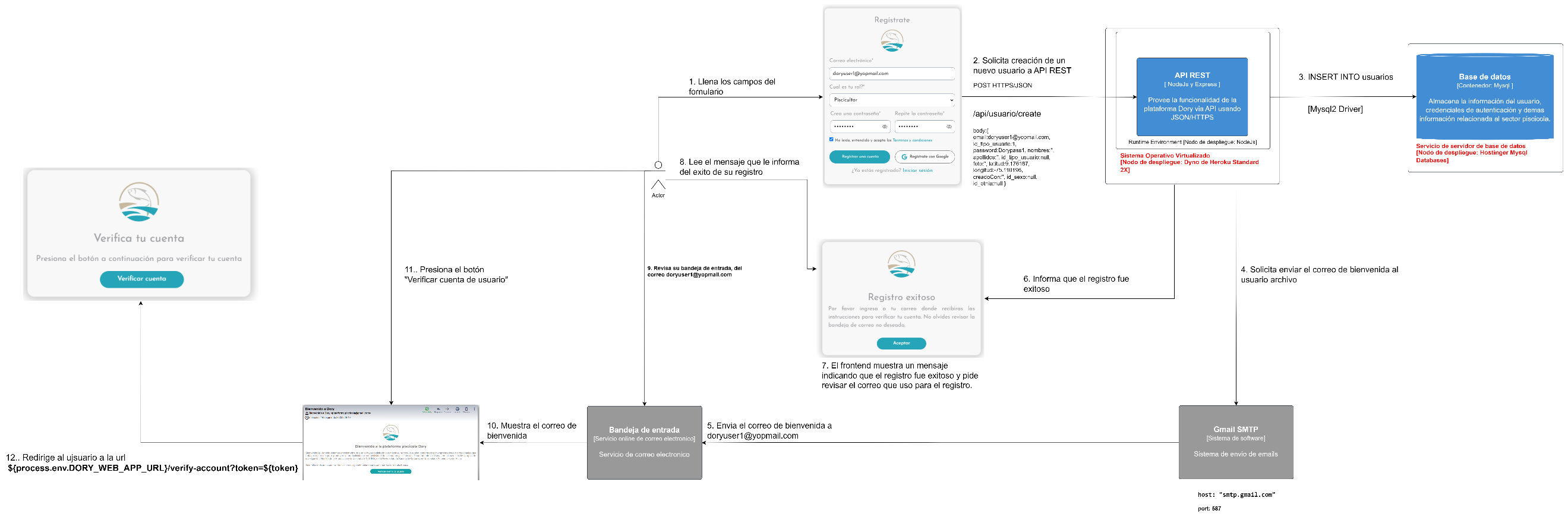


Imagen Diagrama registro por formulario de un usuario en la plataforma Dory.

A continuación, se explicará paso a paso la imagen 8:

Primero veamos el paso 2. Solicita creación de un nuevo usuario al API-REST

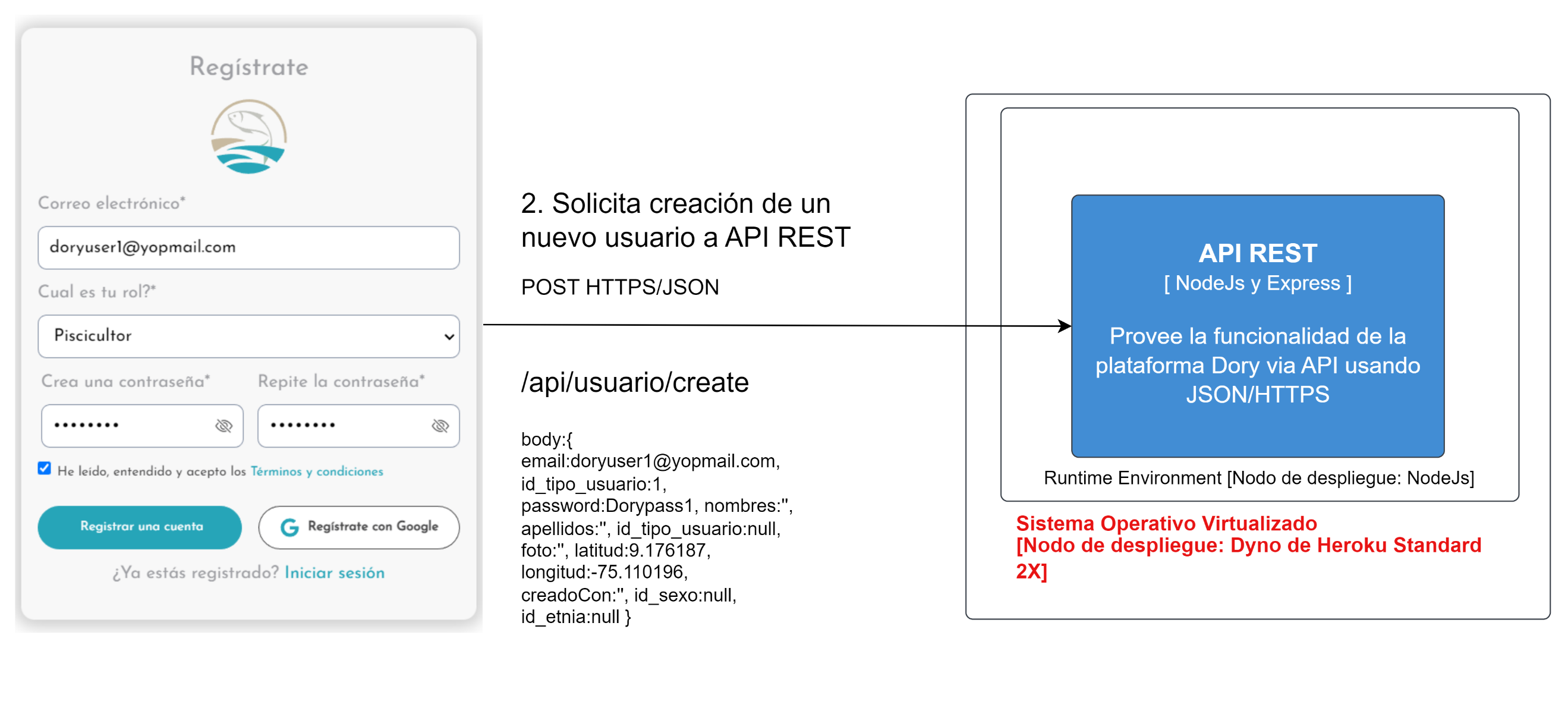


Imagen 8. Formulario solicitándole al API REST la creación de un nuevo usuario

En este paso se envía una solicitud HTTP desde el formulario de registro con datos en formato JSON usando el cliente HttpClient de Angular. El endpoint al que se envía la solicitud es **/api/usuario/create**. Se le envían los datos que el usuario ingreso en el formulario en el body de la solicitud.

body:{

email:doryuser1@yopmail.com,

id\_tipo\_usuario:1,

password:Dorypass1,

nombres:'',

apellidos:'',

id\_tipo\_usuario:null,

foto:'',

latitud:9.176187,

longitud:-75.110196,

creadoCon:'',

id\_sexo:null,

id\_etnia:null

}

Ahora, miremos que pasa cuando llega la solicitud al API REST



Imagen 9. API REST recibiendo la solicitud HTTP que viene del formulario

Ampliemos el cuadro del API REST de la imagen 10 para ver más de cerca lo que pasa dentro.

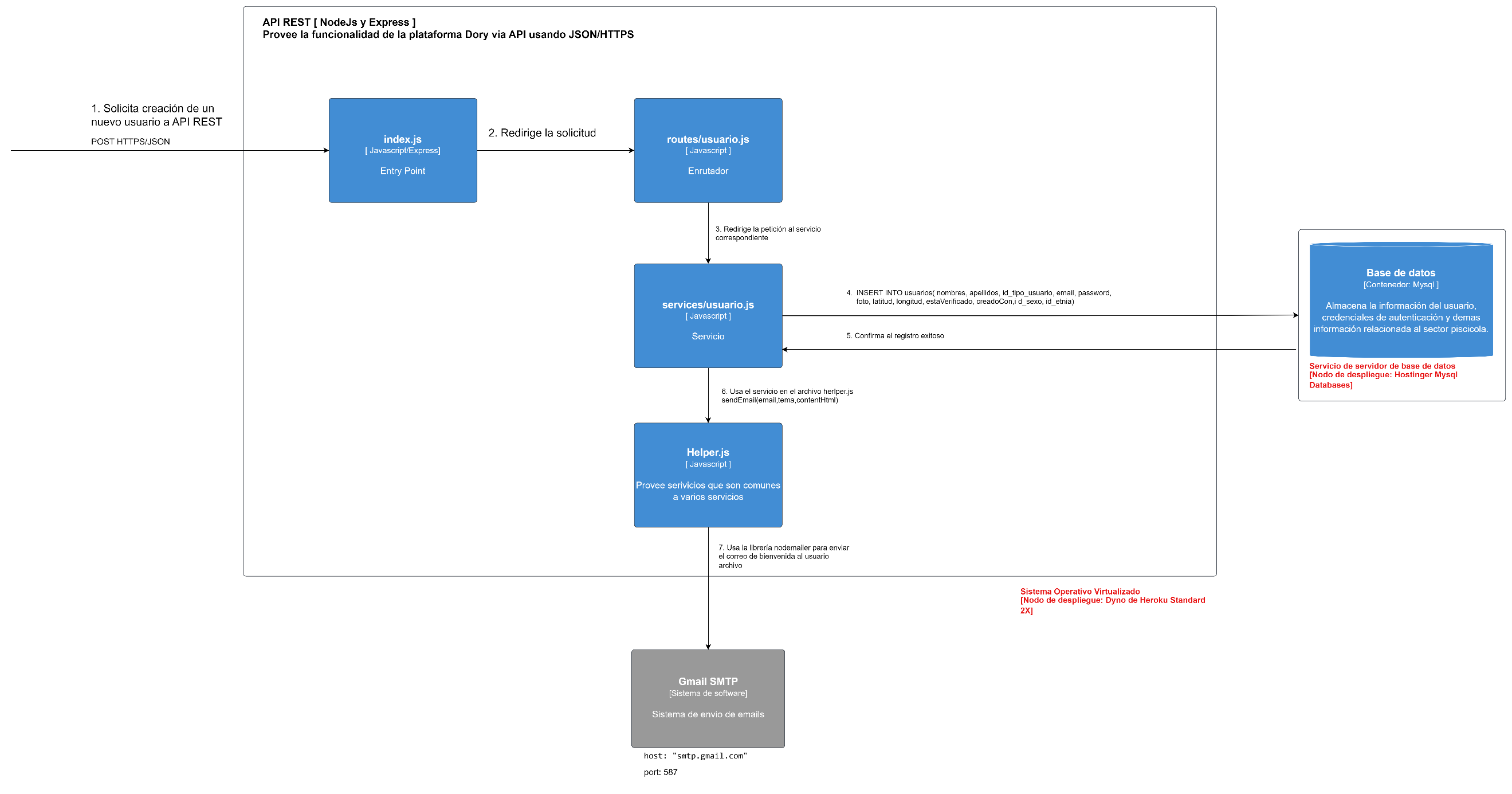


Imagen 10. Zoom del API REST

La solicitud es recibida en punto de entrada(index.js). Aquí a través de express se conoce cuál es el endpoint que el formulario está solicitando y se decide a que enrutador enviar la petición. Como el endpoint solicitado es POST /api/usuario/create y como este contiene el fragmento /api/usuario/ se decide enviar la petición a enrutador routes/usuario.js en el punto 2. Miremos lo que sucede dentro del enrutador:

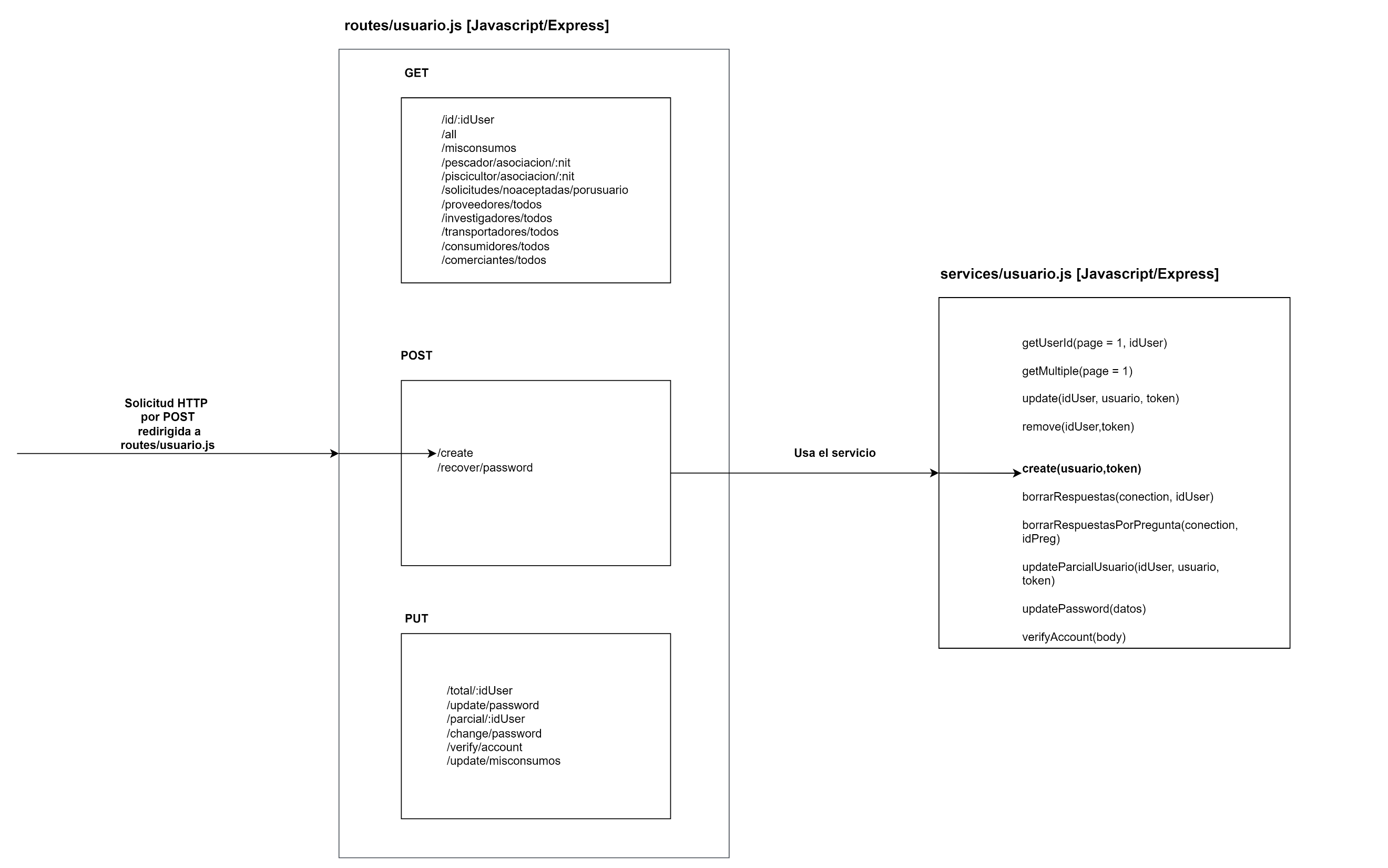


Imagen 11. Funcionamiento interno del enrutador routes/usuario.js

La función del enrutador es seleccionar el servicio que se encargara de resolver la petición. El enrutador contiene múltiples servicios los cuales están agrupados por verbos HTTP(GET, POST, PUT y DELETE). Como la solicitud se hizo usando el verbo POST el enrutador escogerá el servicio solo del grupo POST. Pero como solo debe escoger un solo servicio seleccionará el que este asociado con el endpoint que se está consumiendo. En este caso el escoge el servicio que esta asociado con el fragmento de ruta /create. Por lo tanto el enrutador extrae el body y lo envia al servicio services/usuario.js(punto 3) haciendo el llamado al método create del mismo.

El servicio services/usuario.js en el método create recibe los datos del usuario. Este verifica que todos los datos estén completos. Luego arma la siguiente consulta SQL y la envía usando el Driver Mysql2(https://www.npmjs.com/package/mysql2):

INSERT INTO usuarios(nombres,apellidos,id\_tipo\_usuario,email,password,foto,latitud,longitud,estaVerificado,creadoCon,id\_sexo,id\_etnia) VALUES (?,?,?,?,?,?,?,?,?,?,?,?)

Esta consulta permite insertar los datos ingresados por el usuario en el formulario de registro en la tabla usuarios.

La base de datos responde con un mensaje de éxito cuando realiza la inserción del registro en la tabla usuarios (punto 5).

Luego cuando la inserción del registro es exitosa se arma un correo de bienvenida para el usuario y se le adiciona una URL para verificación de la cuenta.



Imagen . Mensaje de bienvenida en la bandeja de entrada del usuario registrado.

Esta es la URL que se envía en el correo de bienvenida:

${process.env.DORY\_WEB\_APP\_URL}/verify-account?token=${token}

Esta compuesta por los siguientes elementos

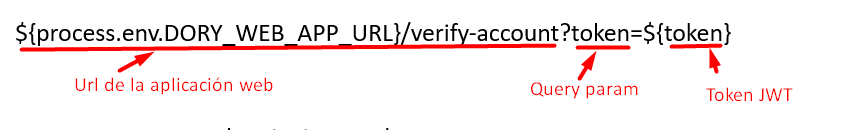


Imagen . Estructura de la url de verificación enviada el correo del usuario registrado.

Process.env.DORY\_WEB\_APP\_URL: Esta es una variable de entorno donde se encuentra almacenada la url de la aplicación web Dory.

/verify-account: Es el complemento de la url de la aplicación web Dory

token: Es un query param. Es decir, un parámetro que enviaremos por la url a la aplicación y cuyo valor es un token JWT.

Cuando el contenido del correo está preparado es enviado por el servicio services/usuario.js utilizando el servicio sendEmail(email,tema,contentHtml) del modulo herper.js como se muestra en la imagen 10 punto 7. Como se puede observar se le envía por parámetros el email receptor, el tema o asunto y el contenido del mensaje en formato html.

Además de estos datos al servicio sendEmail se le a configurado por defecto los siguientes parámetros:

{

host: "smtp.gmail.com",

port: 587,

secure: false,

auth: {

user: "plataforma.piscicola@gmail.com", //dirección desde donde se enviara el correo

pass: process.env.GOOGLE\_PASSWORD\_APP, //contraseña de esta cuenta

}

El servicio sendEmail utiliza la librería nodemailer y usando los datos anteriores solicita el envío del correo al servidor smtp.gmail.com en el puerto 587 como se indica en los campos host y port respectivamente.

e. El usuario API-REST informa que el registro fue exitoso.

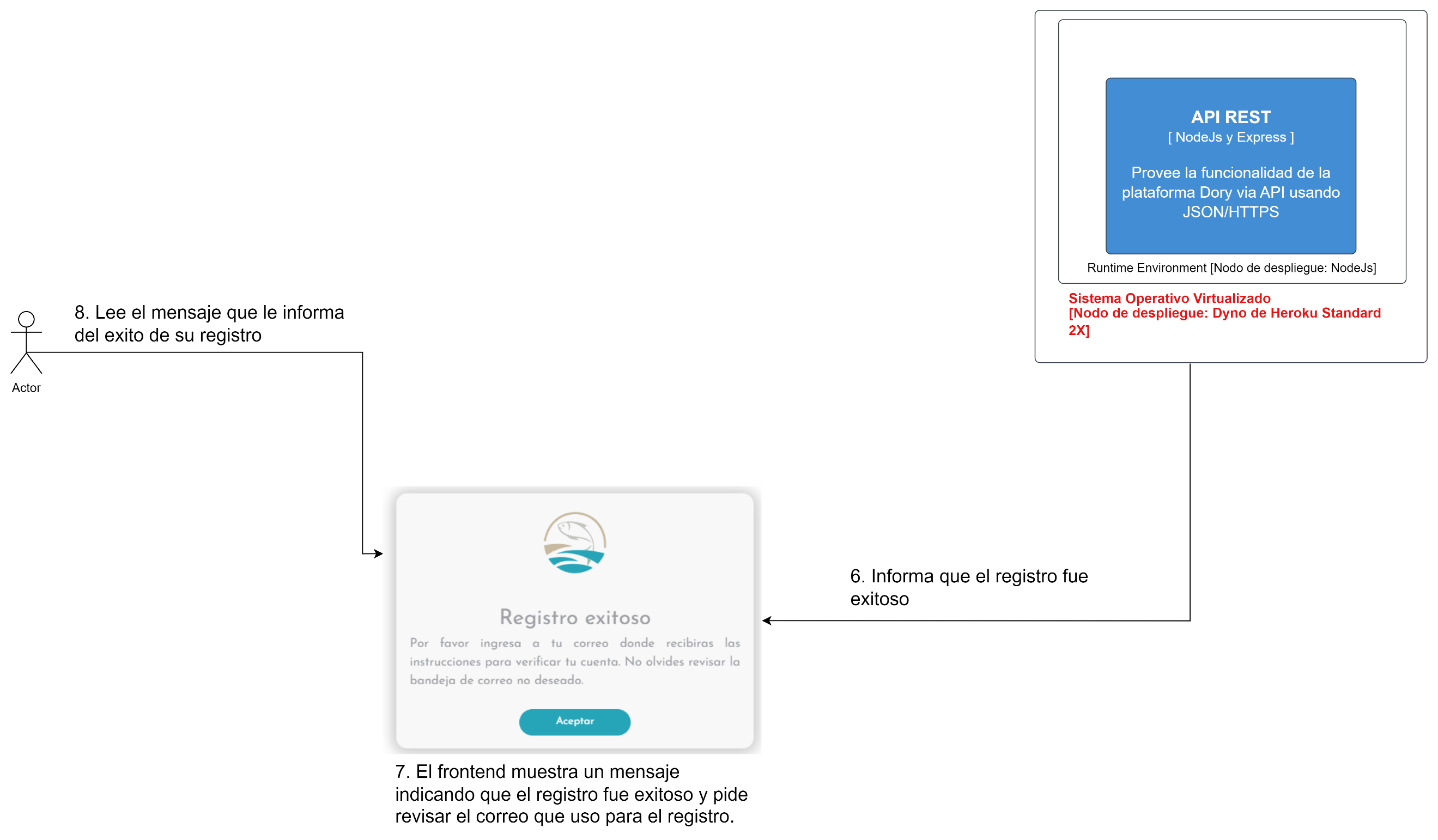


Imagen . Confirmación registro exitoso desde el api rest hacia el usuario registrado.

f. El usuario ingresa a su bandeja de entrada y revisa el correo de bienvenida

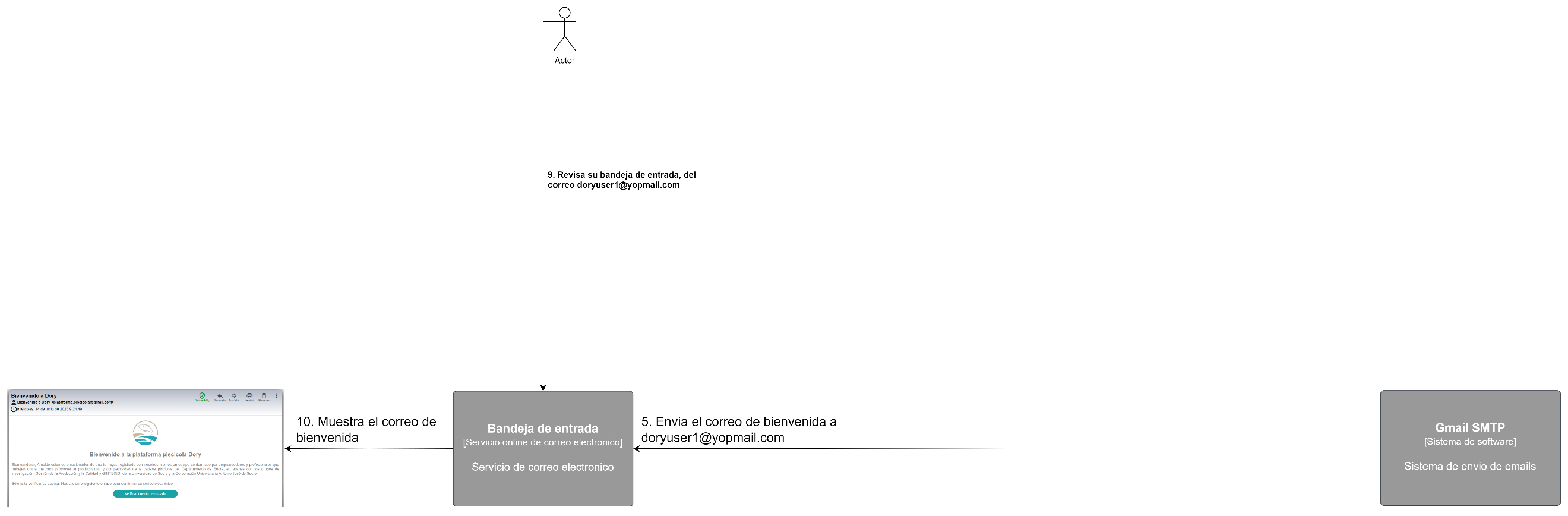


Imagen . Usuario ingresa a la bandeja de entrada para revisar el correo de bienvenida.

f. El usuario presiona el botón “Verificar cuenta de usuario”

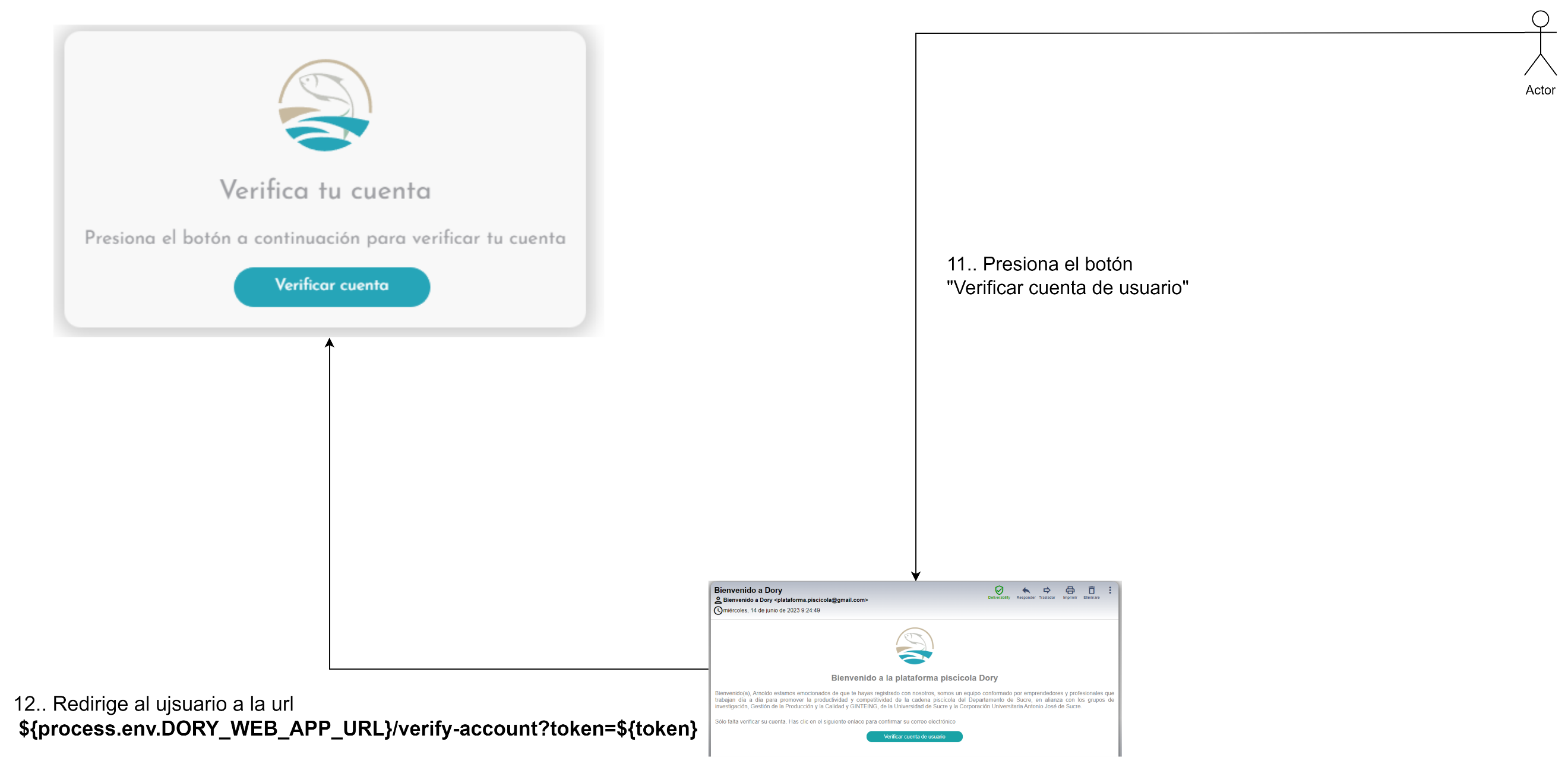
****

Imagen . Usuario presionando el botón verificar cuenta.

Al presionar el botón “Verificar cuenta de usuario” se le abre la dirección ${process.env.DORY\_WEB\_APP\_URL}/verify-account?token=${token} y esta dirige a la vista del punto 12 que lo invita a verificar la cuenta.

Cuando el usuario presiona el botón “Verificar cuenta” se extrae el token que viene en el query param token en ?token=${token}. Luego se le solicita al endpoint /api/usuario/verify/account y enviándole el token JWT en el body de la petición:

Body: {

token: token

}

Lo que hace el endpoint es validar que el token sea valido y cambiar el estado del campo ***estaVerificado*** de la tabla ***usuarios*** *a 1.* De esta manera queda completado el proceso.

**2. Conocer el método de registro usado por Google**

En la aplicación web Dory, también se implementó un método de registro utilizando los servicios de autenticación de Google. Los pasos para el registro con Google son los siguientes:

a. El usuario accede a la opción de registro con Google en la aplicación.

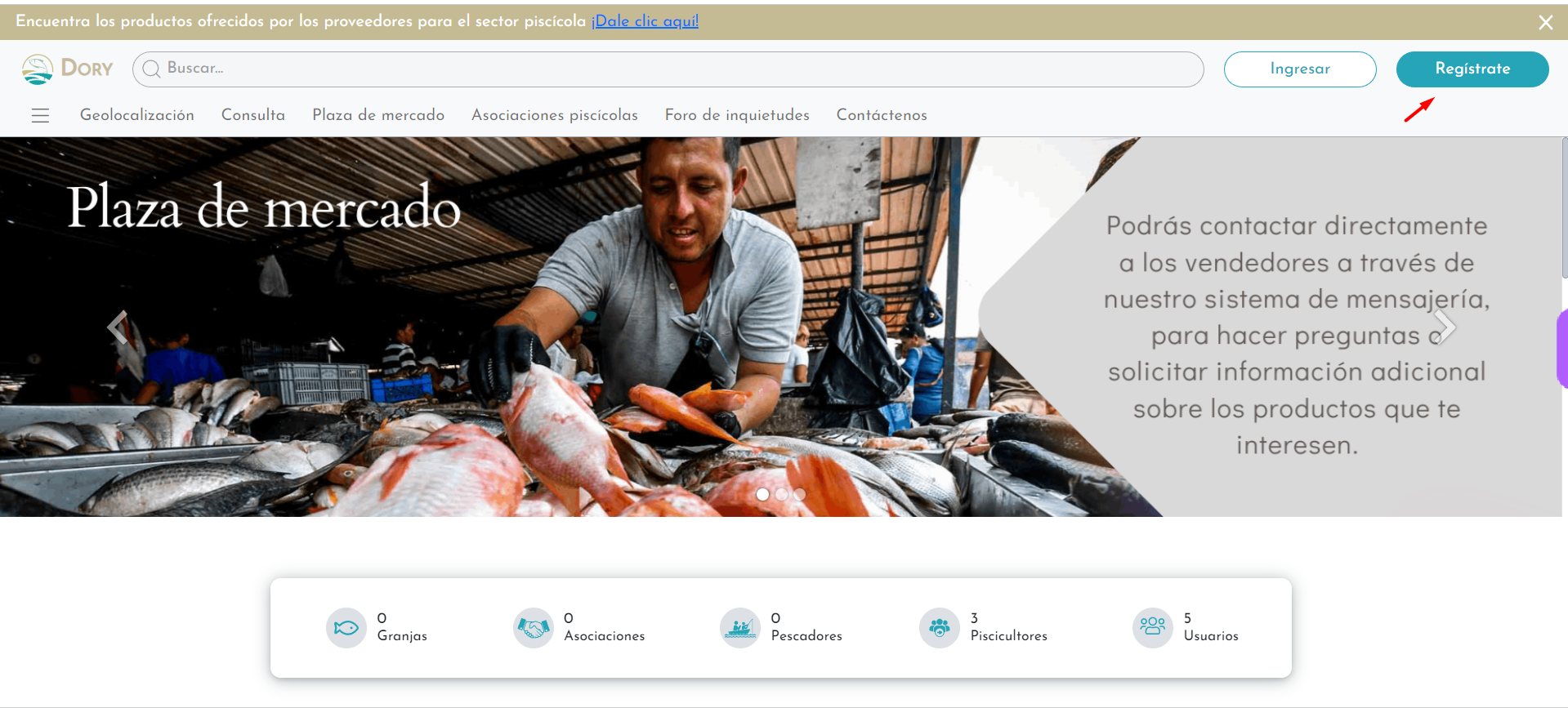


Imagen . Botón de registro en la página de la plataforma Dory.

b. Se muestra un botón o enlace que permite al usuario iniciar sesión con su cuenta de Google.

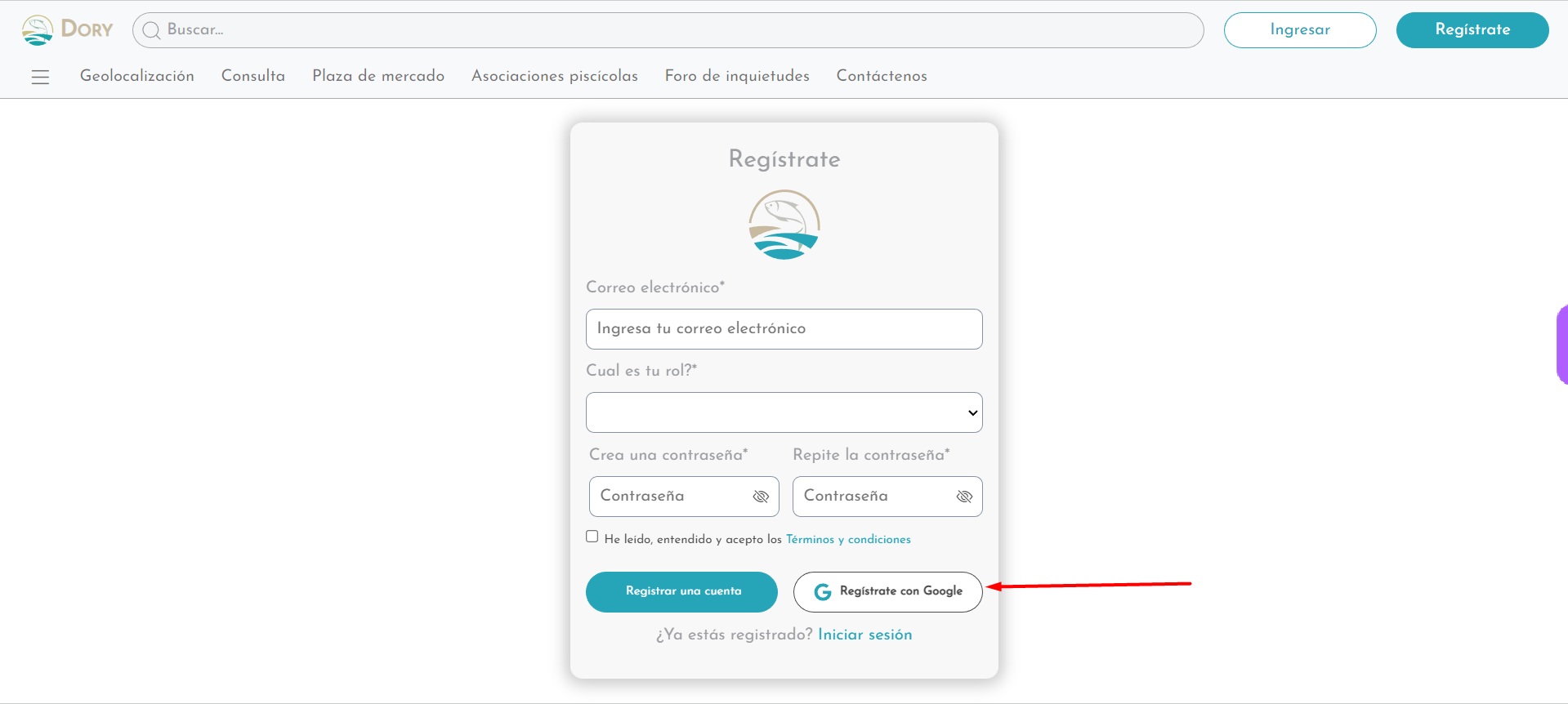


Imagen . Botón de Google en el formulario de registro.

c. Al hacer clic en el botón o enlace, se abre una ventana emergente o se redirige al usuario a la página de inicio de sesión de Google

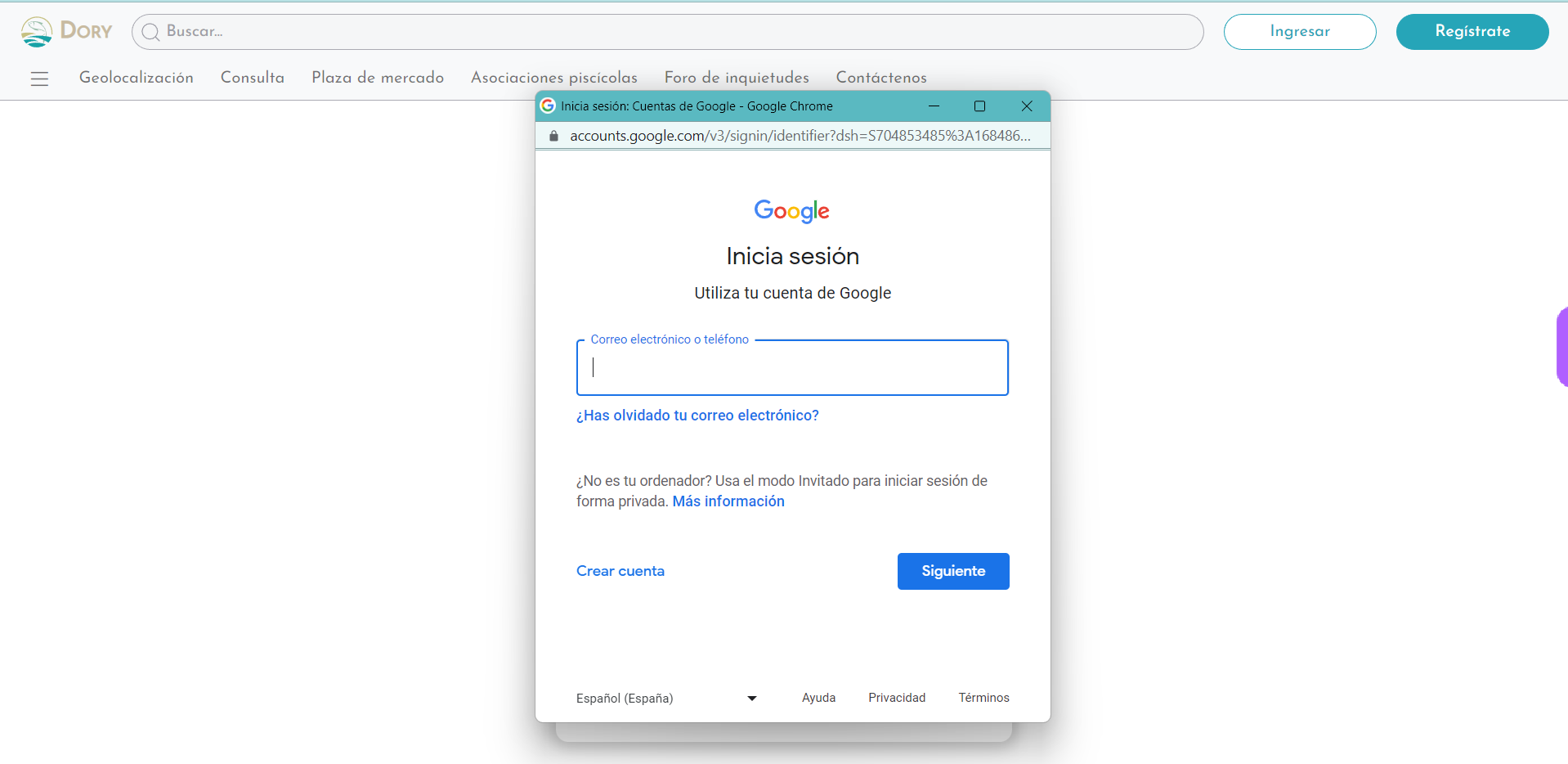


Imagen . Ventana de inicio de sesión.

d. El usuario ingresa sus credenciales de inicio de sesión de Google.

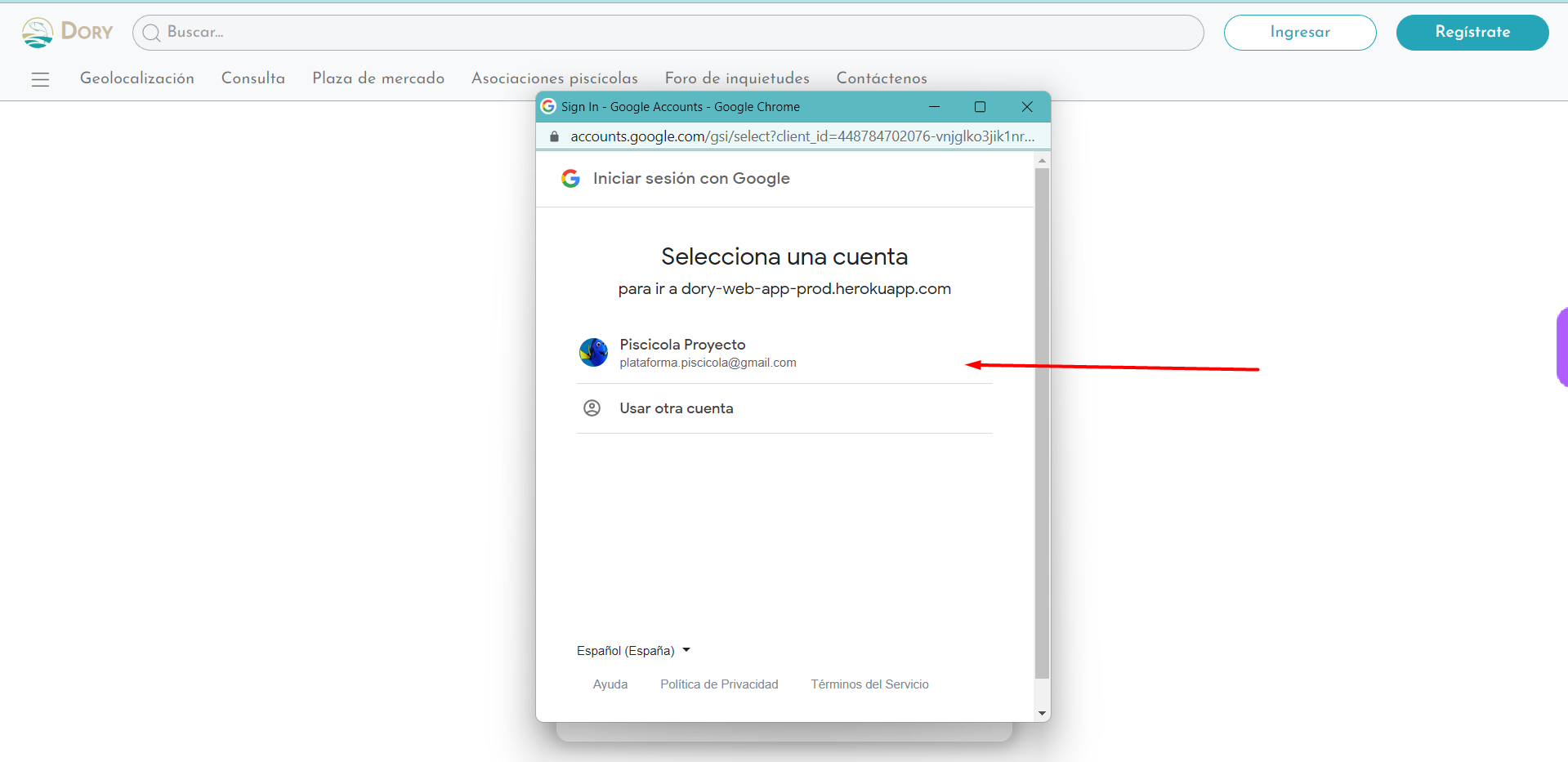


Imagen . Ventana de Google para seleccionar su cuenta para autenticar.

d. Desde el componente RegistroComponent se hace un llamado al api de Google. En la siguiente imagen se muestra el proceso completo desde que el usuario presiona el botón de Google.

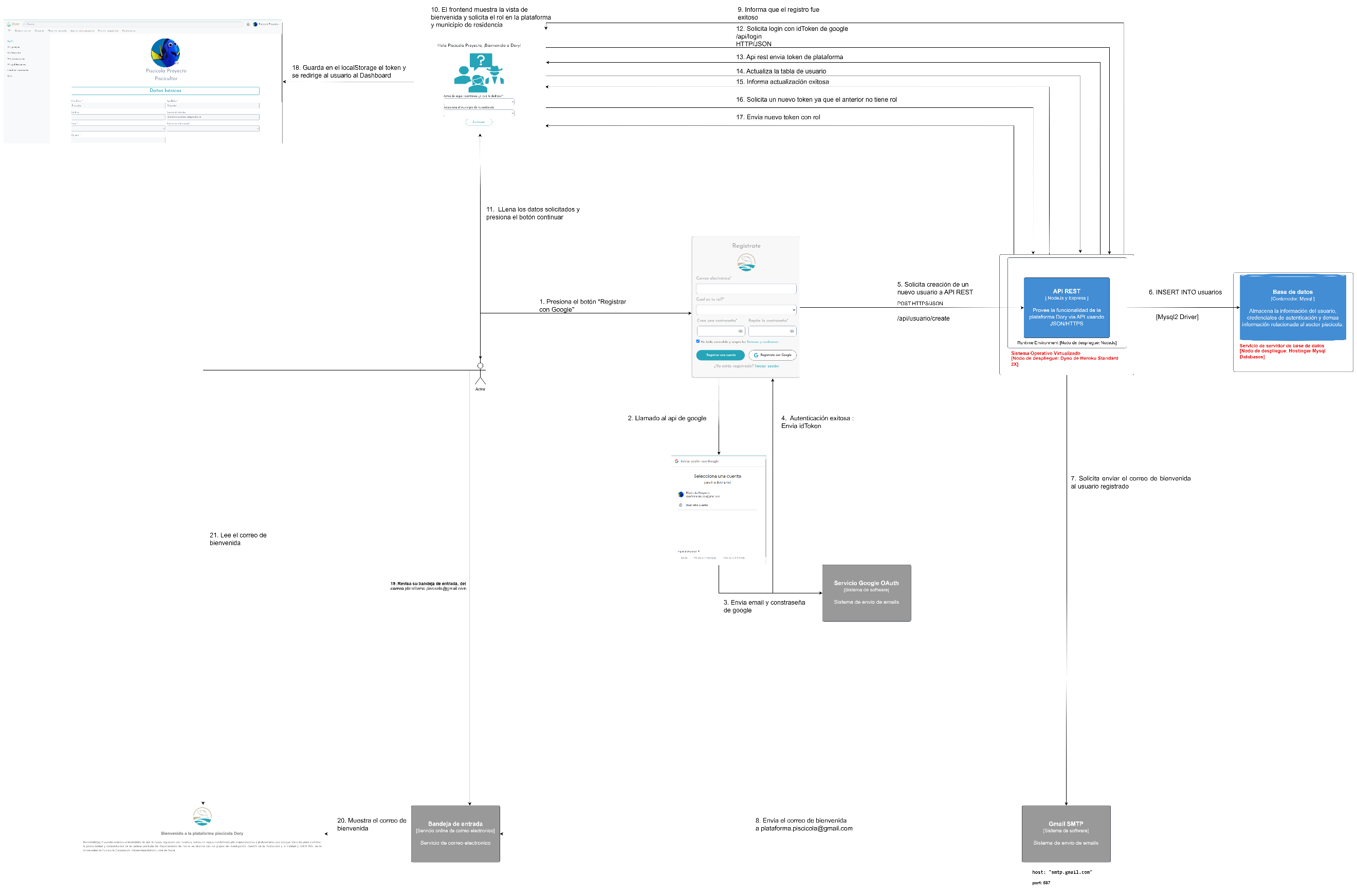
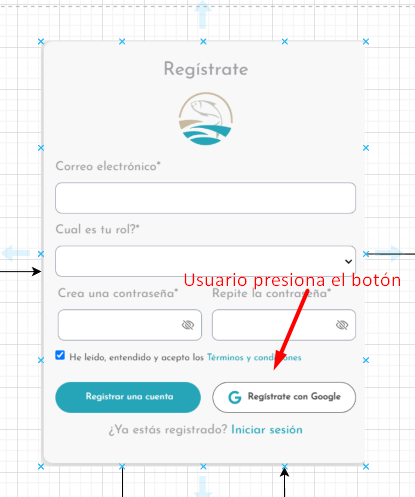


Imagen Proceso de registro completo usando una cuenta de Google.

A continuación, se explicará paso a paso la imagen 21:

**1. Presiona el botón "Registrar con Google"**



El usuario presiona el botón de Google.

**2. Llamado al api de Google**

Se realiza un llamado a el api de Google para que muestre la ventana de autenticación de Google.

**3. Envía email y contraseña de Google**

El usuario ingresa el email y la contraseña de su cuenta de Google

**4. Autenticación exitosa: Envía idToken**

Si la autenticación fue exitosa el servicio de autenticación de Google envía como respuesta un idToken el cual tiene el siguiente aspecto:  
  
eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9..eyJpYXQiOiAxMjQzNDM0MjU0LCAiZXhwIjogMTI0NDUzNDIyMiwgInN1YiI6ICIxNTY0NTY0NjU0NjU0MzIxNjQ2MTY1IiwgImF1ZCI6ICI2NDU0NjU0NjQ2NC12bmhoc2Rmc2Rmc2RmNDM1ZmRmZDR0cmZmZmdnODM0c2FwcHMtZ29vZ2xldXNlcmNvbnRlbnQuY29tIiwgImlzcyI6ICJodHRwczovL2FjY291bnRzLmdvb2dsZS5jb20iLCAibmJmIjogMjYyMjMyMzI1NiwgImp0aSI6ICI2N2FhamhnNzY3

Este token tiene tres partes las cuales se separan con puntos.

La primera parte es el header:

eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9

Esta contiene la siguiente información la cual nos dice que tipo de token es y con que algoritmo está firmado:

{

"alg": "RS256",

"kid": "7987987987897978674654564465489798746549",

"typ": "JWT"

}

La segunda parte contiene el payload o carga útil



{

"iss": "https://accounts.google.com",

"nbf": 2622323256,

"aud": "645465465464-

vnhghyfyfr7fsdgfd567tghjgjgg7834s.apps.googleusercontent.com",

"sub": "1564564654654321646165",

"email": "plataforma.piscicola@gmail.com",

"email\_verified": true,

"azp": "448784702076-vnjsdfsdfdsfdsf435fdfd4trfffg34s.apps.googleusercontent.com",

"name": "Piscicola Proyecto",

"picture": "https://lh3.googleusercontent.com/a/AAcHTtcpkEl7xpAToAhpQU1Jkh79sM3KX-dGY84C9wkq=s96-c",

"given\_name": "Piscicola",

"family\_name": "Proyecto",

"iat": 1243434254,

"exp": 1244534222,

"jti": "67aajhg7676guy7gujg87g8ug8uguighjg8t7gb2"

}

A este token se le puede extraer la data que tiene en su payload. Esta contiene información del usuario como del nombre, apellido, foto y si la cuenta esta o no verificada, entre otros datos. Esta data la usaremos en el punto 5 para crear el usuario.

La tercera parte del token es la firma y se usa para comprobar que el token es válido:

eyJpYXQiOiAxMjQzNDM0MjU0LCAiZXhwIjogMTI0NDUzNDIyMiwgInN1YiI6ICIxNTY0NTY0NjU0NjU0MzIxNjQ2MTY1IiwgImF1ZCI6ICI2NDU0NjU0NjQ2NC12bmhoc2Rmc2Rmc2RmNDM1ZmRmZDR0cmZmZmdnODM0c2FwcHMtZ29vZ2xldXNlcmNvbnRlbnQuY29tIiwgImlzcyI6ICJodHRwczovL2FjY291bnRzLmdvb2dsZS5jb20iLCAibmJmIjogMjYyMjMyMzI1NiwgImp0aSI6ICI2N2FhamhnNzY3

**5. Solicita creación de un nuevo usuario a API REST**

En este paso se envía una solicitud HTTP desde el componente de registro con datos en formato JSON usando el cliente HttpClient de Angular. El endpoint al que se envía la solicitud es **/api/usuario/create**. Se le envían los datos que son extraídos del payload del token de Google en el body de la solicitud.

body:{

email:plataforma.piscicola@gmail.com,

id\_tipo\_usuario:null,

password:null,

nombres:'Piscicola',

apellidos:'Proyecto',

id\_tipo\_usuario:null,

foto:' https://lh3.googleusercontent.com/a/AAcHTtcpkEl7xpAToAhpQU1Jkh79sM3KX-dGY84C9wkq=s96-c',

latitud:9.176187,

longitud:-75.110196,

creadoCon:'google',

id\_sexo:null,

id\_etnia:null

}

**6. INSERT INTO usuarios**

Consulte mas arriba la explicacion que se le da a la imagen 10 para comprender que pasa dentro de del API-REST.

El servicio services/usuario.js en el método create recibe los datos del usuario. Este verifica que todos los datos estén completos. Luego arma la siguiente consulta SQL y la envía usando el Driver Mysql2(https://www.npmjs.com/package/mysql2):

INSERT INTO usuarios(nombres,apellidos,id\_tipo\_usuario,email,password,foto,latitud,longitud,estaVerificado,creadoCon,id\_sexo,id\_etnia) VALUES (?,?,?,?,?,?,?,?,?,?,?,?)

Esta consulta permite insertar los datos de la cuenta de Google del usuario en la tabla usuarios de la base de datos de la plataforma Dory(***process.env.DB\_NAME)***

**7. Solicita enviar el correo de bienvenida al usuario registrado**

La base de datos responde con un mensaje de éxito cuando realiza la inserción del registro en la tabla usuarios(imagen 3).

Cuando el usuario es registrado se le envía un correo de bienvenida. Para esto se prepara el contenido html del correo y luego es enviado en el servicio services/usuario.js usando el método sendEmail(email,tema,contentHtml) del módulo herper.js como se muestra en la imagen 10 punto 7. Como se puede observar se le envía por parámetros el email receptor, el tema o asunto y el contenido del mensaje en formato html.

Además de estos datos al servicio sendEmail se le ha configurado por defecto los siguientes parámetros:

{

host: "smtp.gmail.com",

port: 587,

secure: false,

auth: {

user: "plataforma.piscicola@gmail.com", //dirección desde donde se enviará el correo

pass: process.env.GOOGLE\_PASSWORD\_APP, //contraseña de esta cuenta

}

El servicio sendEmail utiliza la librería nodemailer y usando los datos anteriores solicita el envío del correo al servidor smtp.gmail.com en el puerto 587 como se indica en los campos host y port respectivamente.

**8. Envía el correo de bienvenida a plataforma.piscicola@gmail.com**

En este punto el servicio Gmail SMTP recibe la petición para enviar el correo en el puerto 587 y lo envía a la bandeja de entrada del correo de destino. En este caso el correo del usuario registrado.

**9. Informa que el registro fue exitoso**

El API-REST envía un mensaje al cliente (aplicación web Dory) indicando que el registro fue exitoso. Esto lo hace a través del envió de un código de respuesta 200. El cual el cliente interpretara como un resultado positivo o exitoso.

**10. El frontend muestra la vista de bienvenida y solicita el rol en la plataforma y municipio de residencia**

La aplicación web Dory envía al usuario a la ruta /welcome, esta lo lleva a una página de bienvenida y le solicita unos datos necesarios para este usuario dentro de la plataforma.

**11. LLena los datos solicitados y presiona el botón continuar**

El usuario llena los datos solicitados y presiona el botón.

**12. Solicita login con idToken de google**

La aplicación web realiza un llamdo al endpoint POST /api/login/google en una solicitud HTTP/JSON enviándole en el body el idToken de google de la siguiente forma:

{

token:idToken

}

**13. API-REST envia token de plataforma**

El API-REST responde con un token de la plataforma Dory

{

token:eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiaWF0IjoxNTE2MjM5MDIyfQ.SflKxwRJSMeKKF2QT4fwpMeJf36POk6yJV\_adQssw5c

}

El token de la plataforma es muy necesario para consumir los servicios de esta. Por eso es necesario intercambiar el token de Google por uno de la plataforma.

**14. Actualiza la tabla de usuario**Se llama al endpoint PUT /api/usuario/parcial/{id} y se le envía por la ruta el id del usuario a actualizar y en el body de la solicitud se le envían los campos de la tabla usuarios de la base de datos de la plataforma Dory que se quieren actualizar. En este caso como queremos actualizar el rol y el municipio entonces el body resultante es el siguiente:

{

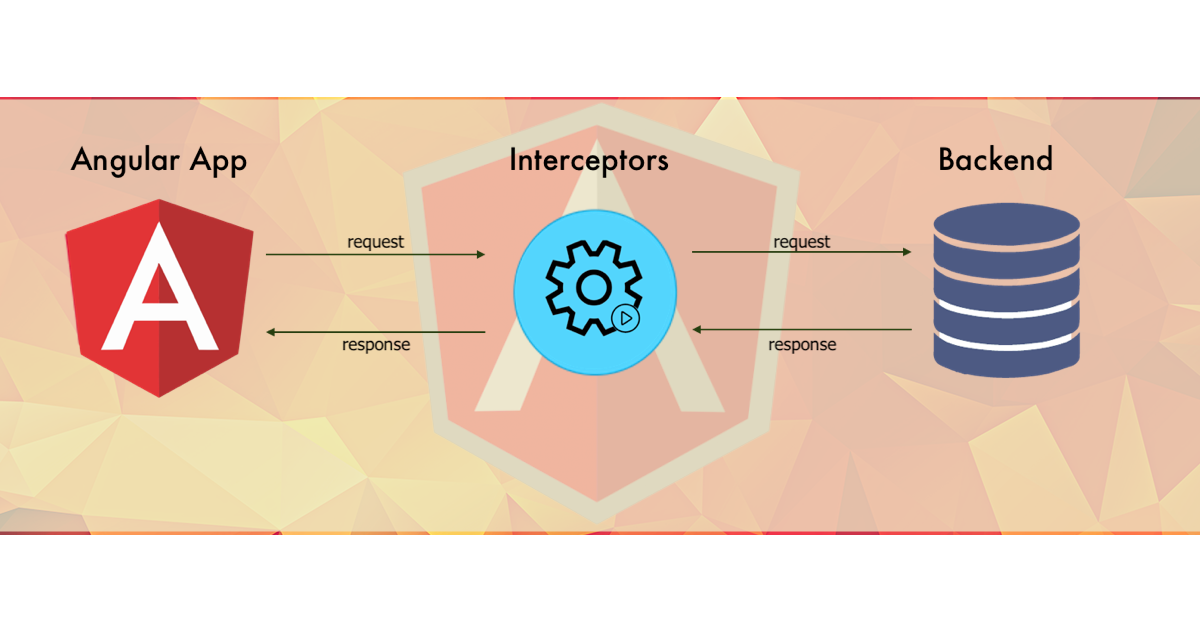
id\_tipo\_usuario:2 //tipo de usuario(rol) Proveedor

id\_municipio:70001 //Sincelejo

}

Lo que quiere decir que este usuario quedará con un rol de Proveedor y se le asignará en el municipio de Sincelejo.

La solicitud antes de salir de la aplicación web Dory pasara por un mecanismo de Angular llamado interceptor el cual se encargará de adicionar el token que se obtuvo en el punto 13 a esta solicitud de actualización de usuario para que el backend después de confirmar que es un token valido autorice la actualización de la tabla de usuarios.



**15. Informa actualización exitosa**

El API-REST envía un mensaje al cliente (aplicación web Dory) indicando que la actualización del registro fue exitosa. Esto lo hace a través del envió de un código de respuesta 200. El cual el cliente interpretara como un resultado positivo o exitoso.

**16. Solicita un nuevo token ya que el anterior no tiene rol**

Se solicita un nuevo token ya que el token enviado por el API-REST en el punto 13 no tiene el rol del usuario. Ya que este este fue generado antes de que el usuario llenara estos datos en la vista de bienvenida.



Por lo tanto se debe pedir igual que en el punto 12 nuevamente al endpoint POST /api/login/google en una solicitud HTTP/JSON enviándole en el body el idToken de google de la siguiente forma:

{

token:idToken

}

**17. Envía el nuevo token con rol**

El API-REST responde con un token de la plataforma Dory, pero esta ves con el campo rol en el payload del token

{

token: eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiaWF0IjoxNTE2MjM5MDIyLCJyb2wiOiJQcm92ZWVkb3IifQ.\_oyb-BuunKSsj1zISRquygBF9nFUcWOASO6f6iKo98E

}

**18. Guarda en el localStorage el token y se redirige al usuario al Dashboard**

Este token enviado en paso 17 se almacena en el localStorage del navegador donde el usuario esta accediendo a la aplicación web Dory para posterios uso. Esto quiere decir que antes de enviar una petición o solitud HTTP desde la aplicación pasara por el interceptor y este buscara el token en el localStorage y lo adicionara en header Authorization y anteponiéndole la palabra Bearer:

Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiaWF0IjoxNTE2MjM5MDIyLCJyb2wiOiJQcm92ZWVkb3IifQ.\_oyb-BuunKSsj1zISRquygBF9nFUcWOASO6f6iKo98E

**El punto 19, 20 y 21 no requiere explicación**