Universidad Tecnológica de la Mixteca

Ingeniería en Computación

Grupo: 602

Programación Web II

Correos

Correos

Enviar correo de recuperación

Cuando olvidamos nuestra contraseña, usualmente tenemos un botón que nos permite recuperarla, de esta manera logramos recuperarla mediante un correo que se nos envía. Por ello tenemos que darle funcionalidad al botón Recuperando contraseña.

```
<div class="row">
   <div class="col s12 center">
      <a (click)="logueo()" class=" modal-action modal-close waves-effect waves-green
btn-flat">Acceso</a>
   </div>
</div>
<div class="row">
   <div class="col s12 center">
      <a  (click)="modalCambiarContrasenya()" >Olvidé mi contraseña!</a>
   </div>
</div>
```

Imagen 1. Botón recuperar (proyectoUTM/src/app/app.login.html)

Para ello creamos una función “modalCambiarContrasenya()” en login.ts

```
modalCambiarContrasenya()
{
  console.log("modalCambiarContrasenya");
  $('#modalCambiarContrasenya').modal({ dismissible: false });
  $('#modalCambiarContrasenya').modal('open');
}
```

Imagen 2. Función modalCambiarContrasenya y enviarCorreo (proyectoUTM/src/app/app.login.ts)

Aquí creamos una función que abre un modal al dar clic, y nos sirven para mostrar un formulario oculto sobre la pantalla que se está visualizando, al principio se pone en *‘false’* para que no sea visible, pero cuando damos clic en el botón, busca en el html de login el i*d=recuperar, y* se abre una ventana emergente, este modal nos muestra el formulario que nos va a ayudar a recuperar la contraseña.

Imagen 3. Formulario recuperando contraseña

```
<div id="modalCambiarContrasenya" class="modal">
   <div class="modal-content">
       <div class="row" style="margin-bottom: 0px !important;">
           <div class="col s12">
               <h5 class="center">Recuperar contraseña</h5>
           </div>
       </div>
       <div class="row" style="margin-bottom: 0px !important;">
           <div class="col s12">
               <p >Revisa tu correo electrónico,revisa el correo no deseado.</p>
           </div>
       </div>
       <div style="border: .5px solid #78909c; padding: 10px;">
           <div class="row">
               <div class="col s12">
                   <label>Correo electrónico</label>
                   <input required type="email" [(ngModel)]="usuario.correo"
placeholder="correo" value="profesor.correo">
               </div>
           </div>
           <div class="row">
               <div class="col s6 center">
                   <a (click)="cambiarContrasenya()" class=" modal-action modal-close
waves-effect waves-green btn-flat">Enviar</a>
               </div>
               <div class="col s6 center">
                   <a class=" modal-action modal-close waves-effect waves-green
btn-flat">Cancelar</a>
               </div>
           </div>
       </div>
   </div>
</div>
```

Imagen 4. Formulario modalCambiarContrasenya (proyectoUTM/src/app/app.login.html)

Creamos un formulario “modalCambiarContrasenya” donde tendremos el campo correo para guardar la dirección. Al dar clic en el botón enviar, llamamos a la función cambiarContrasenya. En dicha función mediante un json, el cual solo contiene un campo (correo), se envía a la función del servicio.

```
 cambiarContrasenya()
 {

this.correoService.enviarCorreoRecuperarContrasenya(this.usuario).subscribe((resUsuari
o: any) =>
   {
     console.log(resUsuario);
   },err => console.error(err));
 }
```

Imagen 5. Función cambiarContrasenya *(proyectoUTM/src/app/app.login.ts)*

Creamos el servicio llamado “CorreoService”, aquí utilizamos otra nueva ruta llamada “API_URI_CORREOS” (ver Imagen 7), la cual se debe de declarar en el archivo enviroment.ts, esta ejecutará un servidor el cuál se encargará de enviar los correos (ver Imagen 6, línea 14). Enviamos la ruta del nuevo servidor (app), y mandamos el json (correo).

```
export class CorreoService
{
 constructor(private http: HttpClient) { }
 enviarCorreoRecuperarContrasenya(body:any)
 {
                                                                                return
this.http.post(`${environment.API_URI_CORREOS}/enviarCorreoRecuperarContrasenya/`,body
);
 }
 decodificarMail(token:any)
 {
   let dato={"token":token};
   return this.http.post(`${environment.API_URI_CORREOS}/decodificarMail`, dato);
 }
}
```

Imagen 6. Servicio correo (proyectoUTM/src/app/services/recuperar.service.ts)

```
export const environment = {
 production: false,
 API_URI: "http://localhost:3000",
 API_URI_CORREOS: "http://localhost:3001",
};
```

Imagen 7. Servidor para correos (proyectoUTM/src/enviroments/enviroment.ts)

Modificamos el archivo package.json para ejecutar el demonio del nuevo servidor (app). Donde añadimos el script "correo", de esta manera podremos ejecutarlo con;

npm run correo.

```
 "scripts": {
   "build": "tsc -w",
   "dev": "nodemon src/build/index.js",
   "correo": "nodemon src/build/app.js"
 },
```

Imagen 8. Modificar package.json (proyectoUTM/server/package.json)

Debemos crear en el server un nuevo archivo llamado "app.ts". El cual es el servidor que enviará correos y en donde se inicia el mismo.

```
import express, { Application } from 'express';
import morgan from 'morgan';
import cors from 'cors';
import jwt from 'jsonwebtoken';
import dotenv from 'dotenv';
import pool from "./database";
const correoAcceso = require('./correoAcceso');
class Server
{
   public app: Application;
   constructor()
   {
       dotenv.config();
       this.app = express();
       this.config();
       this.routes();
   }
   config(): void
   {

       this.app.use(express.urlencoded({limit: '50mb',parameterLimit: 100000,extended:
false}));
       this.app.use(express.json({limit: '50mb'}));
```

```
        this.app.set('port', process.env.PORT || 3001);
        this.app.use(morgan('dev'));
        this.app.use(cors());
        this.app.use(express.urlencoded({ extended: false }));
    }
    routes(): void
    {
        this.app.post('/enviarCorreoRecuperarContrasenya', (req, res) =>
        {
            correoAcceso(req.body);
        });
        this.app.post('/decodificarMail', async (req, res) =>
        {
            let decodificado;
            try
            {
                decodificado = jwt.verify(req.body.token, process.env.TOKEN_SECRET ||
'prueba');
                const result1 =  await this.queryProfesor(decodificado) as any;
                if(result1.length==0)
                    res.json(0);
                else
                    res.json( result1[0] );

            }
            catch(err) {    res.json( 0 );  }
        });
    }
    queryProfesor = (decodificado:any) =>
    {
        return new Promise((resolve, reject)=>
        {
            let consulta='SELECT * FROM profesores WHERE correo="'+decodificado+'"';
            pool.query(consulta, (error:any, results:any)=>
            {
                if(error)
                    return reject(error);
                return resolve(results);
            });
        });
    };
    start()
    {
        this.app.listen(this.app.get('port'), () =>
        {
            console.log(`Listening on port ${this.app.get('port')}`);
        });
    }
}
const server = new Server();
server.start();
```

Imagen 9. Servidor app que servirá para el envío de correos

Aquí debemos hacer solicitudes en formato json, es por ello que colocamos en la función ***config*** los códigos correspondientes. Para poder escuchar las peticiones mediante el demonio, ejecutamos el método start.

En esta parte lo más importante es el apartado **routes()**, ya que este es llamado por el servicio del cliente *(correoService.ts),* al igual que mandamos los datos obtenidos al método que enviará el correo *(ver Imagen 9, línea 32),* para ello renombramos e instanciamos el funcionamiento del método ***decodificarMail'***.

Ahora creamos el archivo “correoAcceso”, el cuál será el encargado de enviar el correo y darle el formato.

```
var email = require("emailjs/email");
import jwt from 'jsonwebtoken';
import dotenv from 'dotenv';
dotenv.config();
module.exports = (formulario: any) =>
{
   const token : string = jwt.sign(formulario.correo, process.env.TOKEN_SECRET || 'prueba');
   var server = email.server.connect({
       user:   "desarrollo@correo.com",
       password:"prueba",
       host:   "smtp.gmail.com",
       ssl: true,
   });
   var message: any ={};
       message =
       {
           from: "Desarrollo UTM <erik@mixteco.utm.mx>",
           to: formulario.correo,
           bcc: "",
           subject: "Cambio de contraseña",
           attachment: [
               { data: `
               En la siguiente liga podrás cambiar tu contraseña:
               <a href="http://localhost:4200/recuperar/${token}" >ACEPTAR</a>
               <br><br>

                   `, alternative: true }
           ]
       };
   server.send(message, function(err:any, message:any) { console.log(1); });
}
```

Imagen 10. Archivo correoAcceso (proyectoUTM/server/src/correoAcceso.ts)

Comenzamos importando la librería mediante una variable email. Después de ello hacemos un módulo en el cual vamos a conectar al email con el servidor, y pondremos los campos de la cuenta del usuario y su contraseña, al igual que el servidor del correo “Gmail”, cabe destacar que cada tipo servidor de correo tiene una configuración diferente, estas configuraciones varían de acuerdo al servidor.

Declaramos una variable “message” y la inicializamos vacía. Esta variable contendrá todo el formato de un correo, como se muestra en la imagen.

Finalmente se envía el correo.

Imagen 11. Correo recibido