UNIVERSIDAD TÉCNICA DEL NORTE
FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS
CARRERA DE SOFTWARE
ASIGNATURA: FABRICA DE SOFTWARE NIVEL: 8

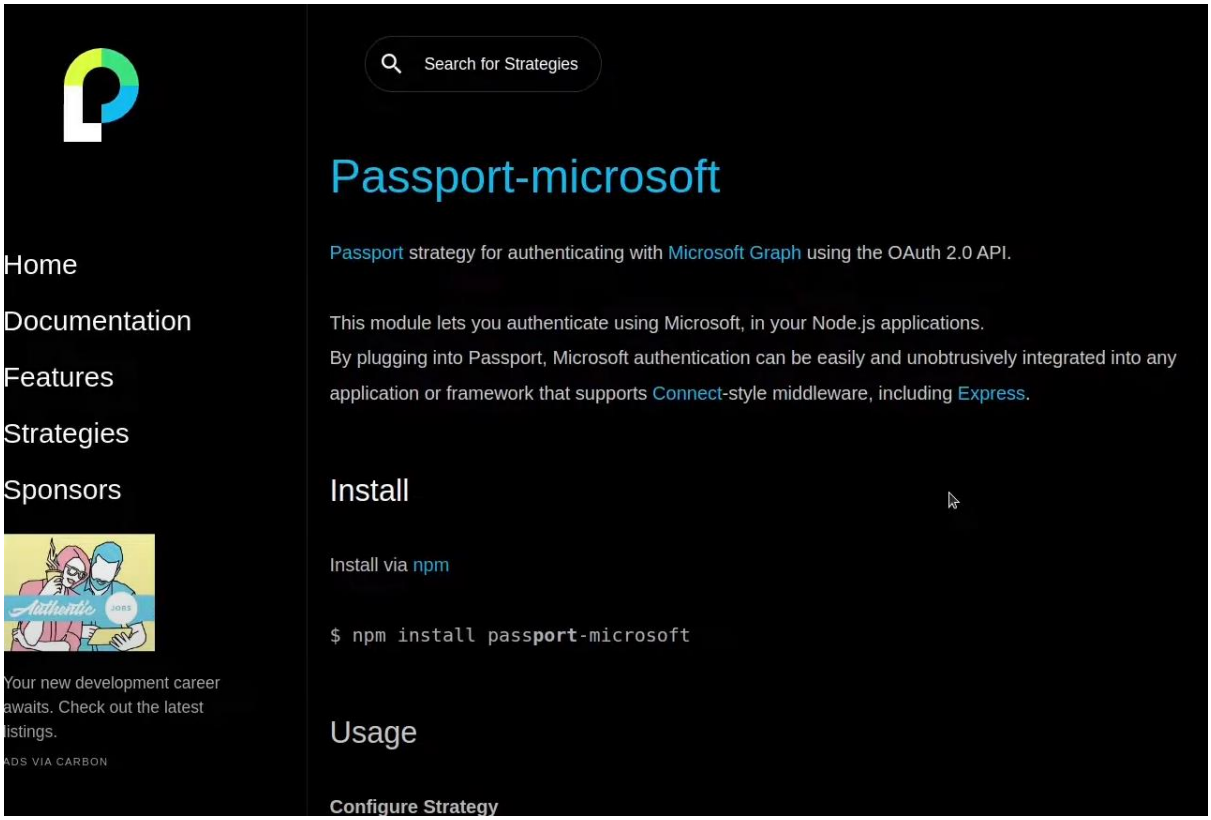
Nombre del proyecto:	Back UTN
Fecha:	08/01/2024
Tema:	Guía de desarrollador correo federado con las credenciales de Microsoft
Objetivo de esta actividad:	Explicación de la implementación del uso del correo Microsoft en la aplicación Back UTN

INDICACIONES:

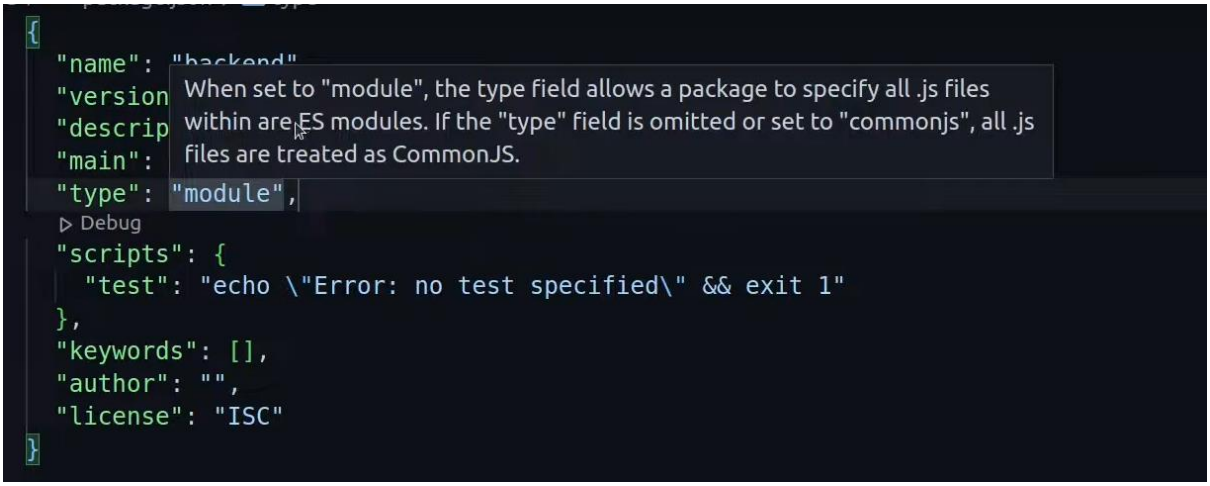
Para realizar la implementación del correo federado es necesario, tener una cuenta educativa asociada con Microsoft y una credencial de Azure para la gestión de permisos para poder usar esta funcionalidad.

DESARROLLO:

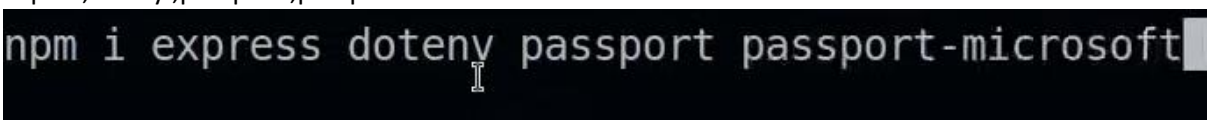
1. Instalar la librería Passport-Microsoft:



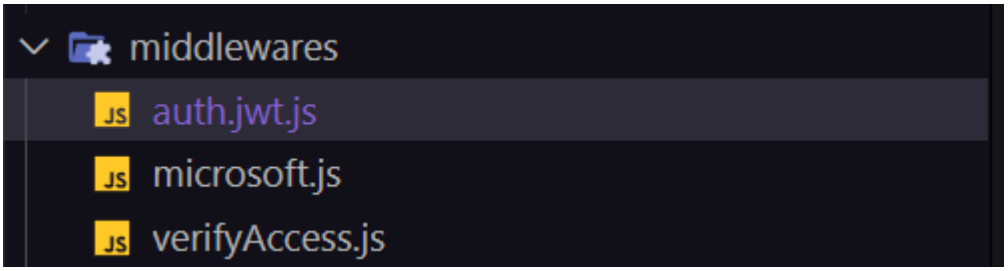
2. Configurar el packege.json para usar módulos ES modules se mira a continuación el dato a ingresar:



3. Luego de iniciar la aplicación se procedera a instalar las librerías necesarias en este caso son: expres, donty ,passport ,passport-microsoft :



4. Luego se define una carpeta que va a ser un intermediario entre nuestra app, y la app de Microsoft:



5. Luego se verificar el código, aquí se maneja la lógica para poder extraer los datos para ser ingresados dentro de la base de datos a continuación el código:

```
// Import necessary modules
import passport from "passport";
import { Strategy as MicrosoftStrategy } from "passport-microsoft";
import { config } from "dotenv";
import { client } from "../database/database.js";
import { getUserDesByID } from "../controllers/usersController.js";

// Load environment variables from a .env file
config();

// Serialization: Convert the user object to an identifier (user.id)
passport.serializeUser(function (user, done) {
  if (user && user.id) {
    done(null, user.id);
  } else {
    console.error("Invalid user object:", user);
    done(new Error("Invalid user object"), null);
  }
});

// Deserialization: Convert the identifier back to a user object
passport.deserializeUser(async function (id, done) {
  try {
    // Fetch user details based on the identifier
    const user = await getUserDesByID(id);

    if (!user) {
      console.error("User not found");
      return done(null, false);
    }
    console.log("User is:", user.user_full_name);

    // Fetch role and module assignments
    const assignmentsQueryResult = await client.query(

        SELECT am.assignment_id, r.rol_name, u.user_full_name, m.module_name
        FROM assignments_modules AS am
        INNER JOIN roles AS r ON am.rol_id = r.rol_id
        INNER JOIN modules AS m ON am.module_id = m.module_id
        INNER JOIN users AS u ON am.user_id = u.user_id WHERE u.user_id = $1
      `,
      [user.user_id]
    );

    const roles = assignmentsQueryResult.rows.map((row) => row.rol_name);
    console.log("Roles:", roles);
    const modules = assignmentsQueryResult.rows.map((row) => row.module_name);
    console.log("Modules:", modules);

    // Pass roles and modules when deserialization is complete
    done(null, { user, roles, modules });
  } catch (error) {
    done(error, null);
  }
});

// Passport strategy for Microsoft authentication
passport.use(
  "auth-microsoft",
  new MicrosoftStrategy(
    {
      clientID: process.env.MICROSOFT_CLIENT_ID,
      clientSecret: process.env.MICROSOFT_CLIENT_SECRET,
      callbackURL: "https://app-backend-utn-2023.onrender.com/auth/microsoft/callback",
      scope: ["user.read", "calendars.read", "mail.read", "offline_access"],
      authorizationURL: "https://login.microsoftonline.com/8dbe1469-c79c-4e21-9d43-
ca65d9e9c475/oauth2/v2.0/authorize",
      tokenURL: "https://login.microsoftonline.com/8dbe1469-c79c-4e21-9d43-
ca65d9e9c475/oauth2/v2.0/token",
    },
    async function (req, accessToken, refreshToken, profile, done) {
      console.log(req);
      try {
        // Log the user details
        done(null, { profile, accessToken, refreshToken });
        // Check if the user exists in the database
        const userQueryResult = await client.query("SELECT * FROM users WHERE user_code = $1",
[profile.id]);
        let userFromDB;

        if (userQueryResult.rows.length > 0) {
          userFromDB = userQueryResult.rows[0];
          console.log("User already registered");
        } else {
          // If the user does not exist, insert them into the database
          const userData = {
            user_code: profile.id,
            user_full_name: profile.displayName,
            user_first_name: profile.name.givenName,
            user_last_name: profile.name.familyName,
            user_email: profile.emails && profile.emails.length > 0 ? profile.emails[0].value : "",
            user_phone_number: profile._json.mobilePhone || "",
            user_state: true,
            user_date_register: new Date(),
          };

          const insertResult = await client.query(

              INSERT INTO users (user_code, user_full_name, user_first_name, user_last_name,
user_email,
                user_phone_number, user_state, user_date_register)
              VALUES ($1, $2, $3, $4, $5, $6, $7, $8)
              ON CONFLICT (user_code) DO NOTHING
              RETURNING *;
            `,
            [
              userData.user_code,
              userData.user_full_name,
              userData.user_first_name,
              userData.user_last_name,
              userData.user_email,
              userData.user_phone_number,
              userData.user_state,
              userData.user_date_register,
            ]
          );

          userFromDB = insertResult.rows[0];
          console.log("User registered successfully");
        }
      } catch (error) {
        console.error("Error authenticating user", error);
        done(error, null);
      }
    }
  )
);
```

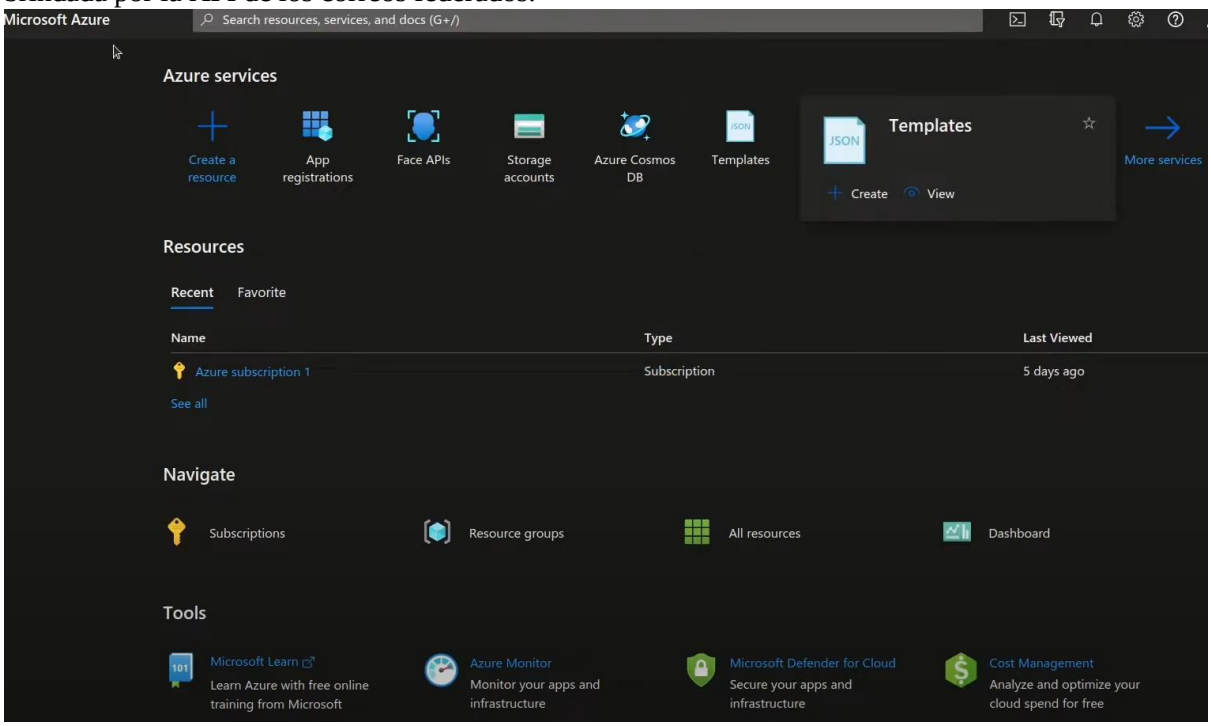
6. Se importa las librerías que van ha ser necesarias en este caso las siguientes:

```
import express from "express";
import session from "express-session";
import cors from "cors";
import passport from "passport";
```

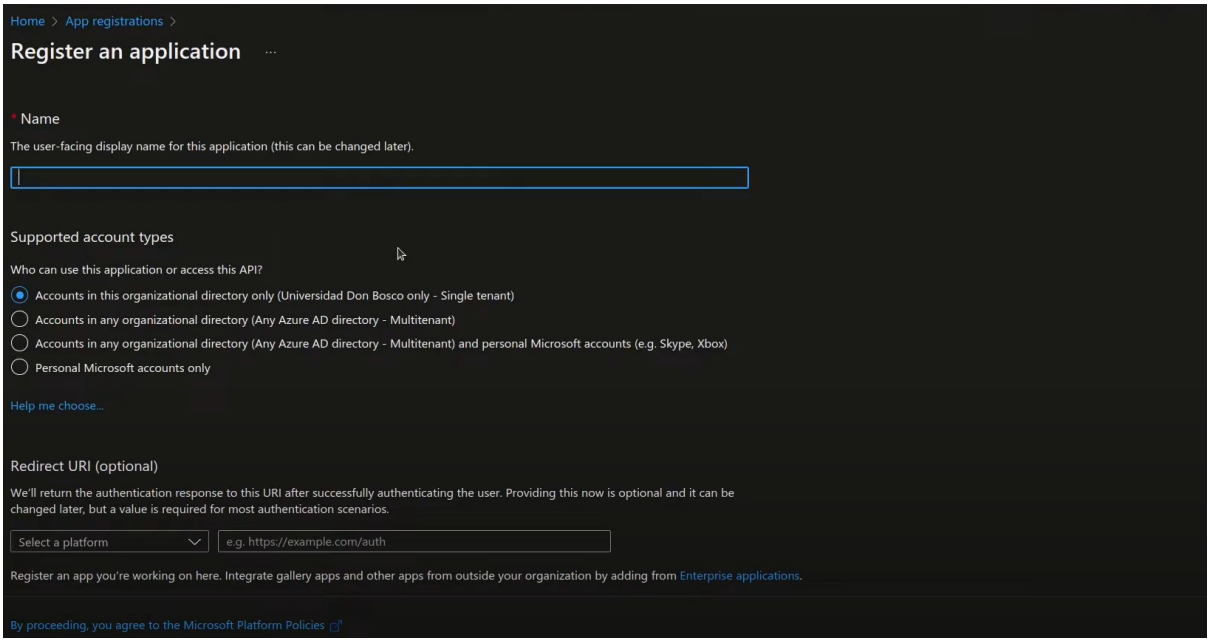
7. Luego procedemos a generar una ruta y importa el modulo a nuestro main o index:

```
import {loginRouter} from './routes/microsoft.js";
import {usersRouter} from "./routes/users.js";
import {rolesRouter} from "./routes/roles.js";
import {modulesRouter} from "./routes/modules.js";
import {assignments_modulesRouter} from "./routes/assignments_modules.js";
import {eventsRouter} from "./routes/events.js";
import {assignments_eventsRouter} from "./routes/assignments_events.js";
import {classroomRouter} from "./routes/classroom.js";
import {assignments_classRouter} from "./routes/assignments_class.js";
import {class_scoreRouter} from "./routes/class_score.js";
import {auditingRouter} from "./routes/auditing.js";
import "./middlewares/microsoft.js";
import {authorize} from "./middlewares/verifyAccess.js";
import {swaggerDocs as V1SwaggerDocs} from "./routes/swagger.js";
import {router} from './routes/login.routes.js'
import { getmRouter } from "./routes/getmodules.routes.js";
const port = process.env.PORT || 3000;
```

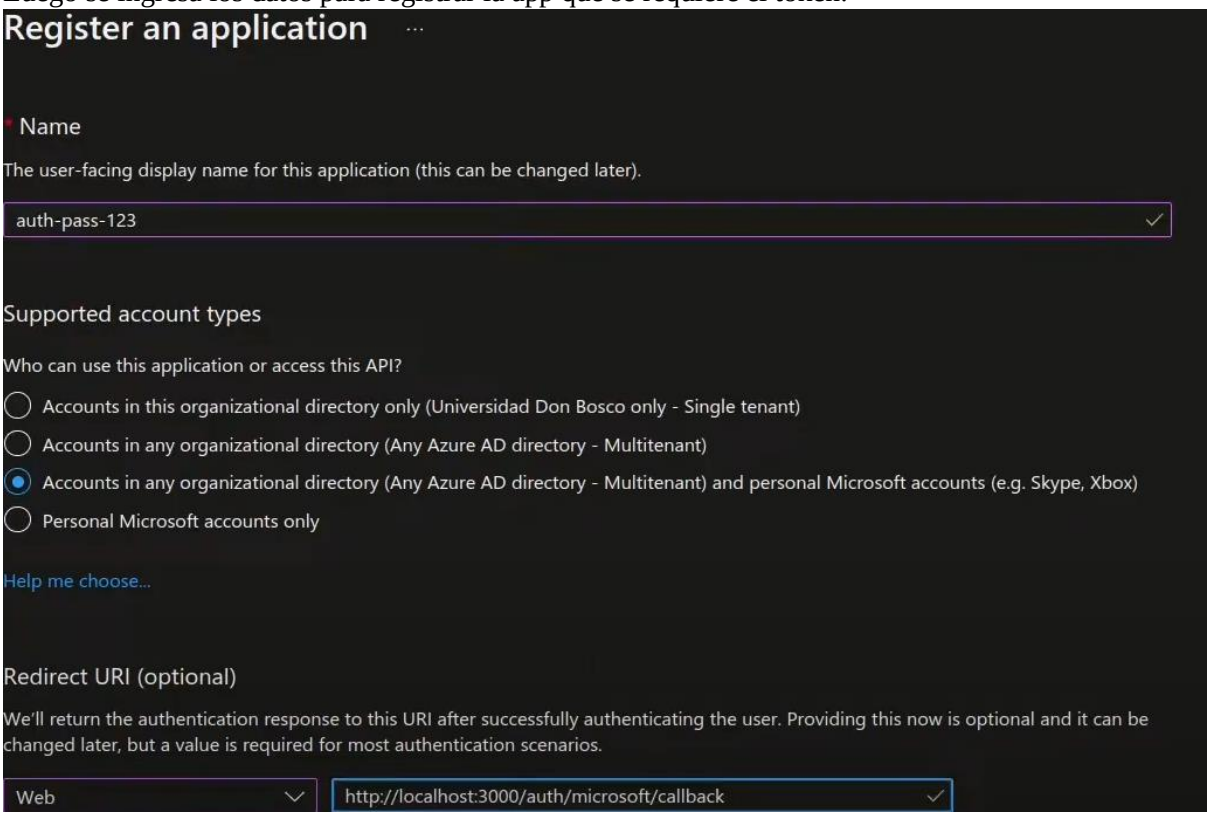
8. Luego vamos al portal de azure para poder permisos para acceder la aplicación a la información brindada por la API de los correos federados:



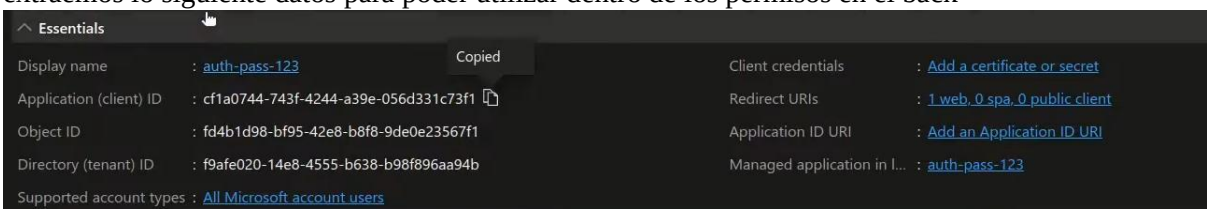
9. Luego colocamos en el buscador registrar nueva aplicación:



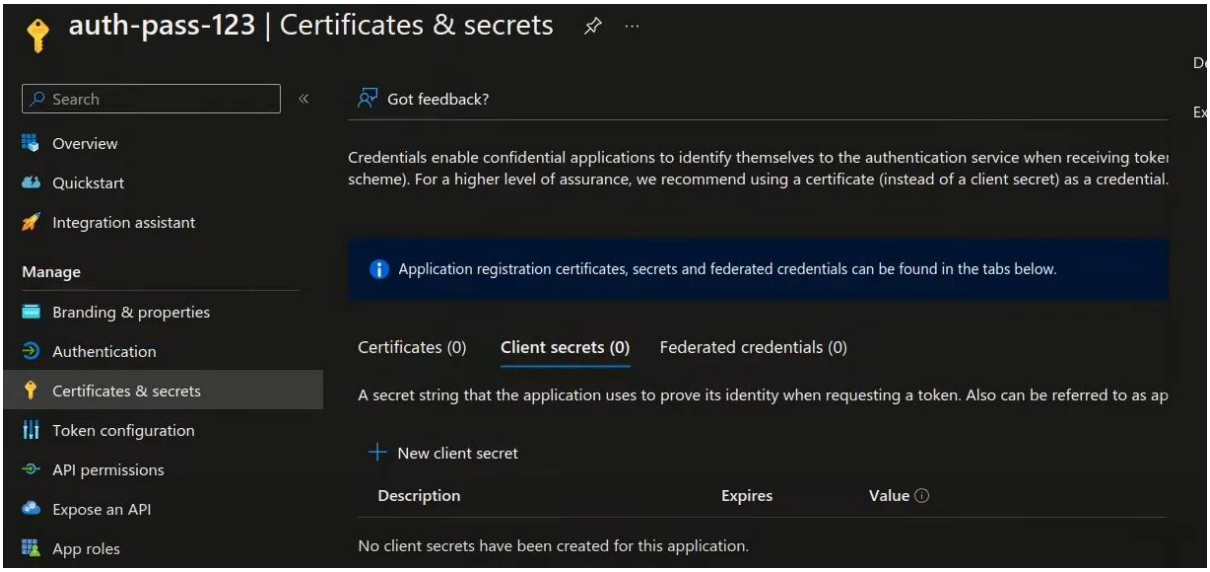
10. Luego se ingresa los datos para registrar la app que se requiere el token:



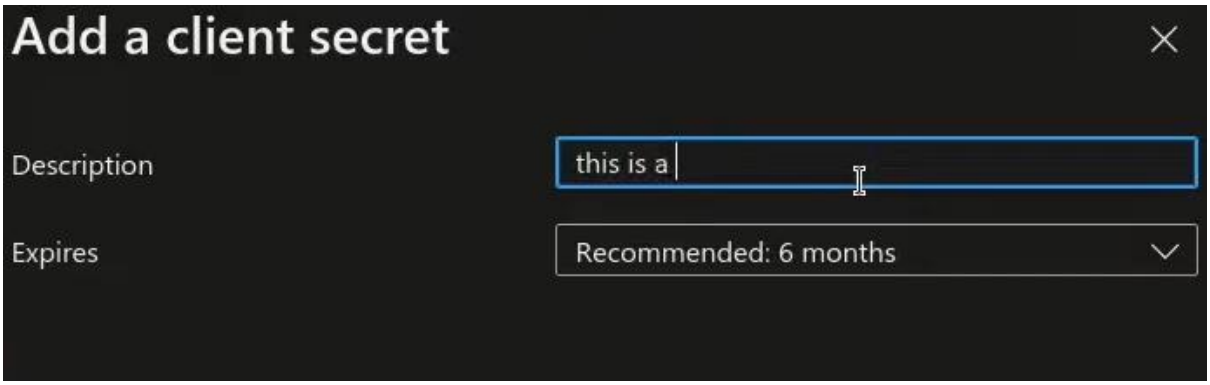
11. El siguiente paso para realizar correctamente la implementación con la API de backUTN extraemos lo siguiente datos para poder utilizar dentro de los permisos en el back



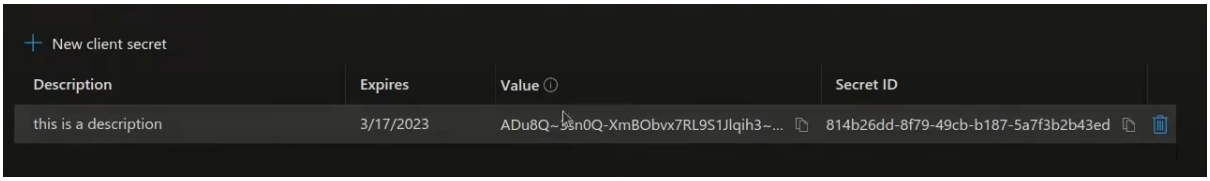
12. Luego se crea un certificado para nuestra app en la sección de Certificates & secrets:



13. Damos clic en nuevo cliente y registramos la aplicación:



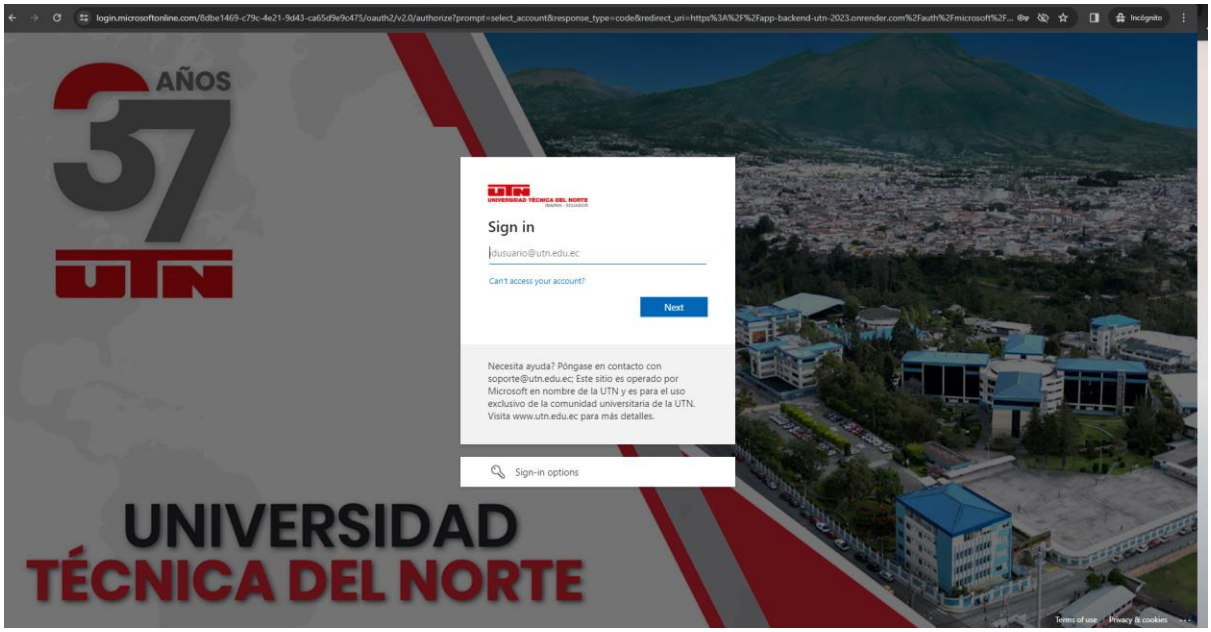
14. Una vez ya se haya registrado la aplicacion se genera la clave para el uso del correo federado:



15. Luego copiamos esa clave y se coloca junto con el Id del usuario, se realizaría de la siguiente forma:

```
1  MICROSOFT_CLIENT_ID=cf1a0744-743f-4244-a39e-056d331c73f1
2  MICROSOFT_CLIENT_SECRET=ADu8Q~3sn0Q-XmBObvx7RL9S1Jlqih3~Eix8saPA
```

16. Luego una vez realizado todo el proceso para acceder al login de Microsoft ocupamos la siguiente ruta: https://app-backend-utn-2023.onrender.com/auth/microsoft una vez se da el clic nos llevara a la pantalla de Microsoft:



17. Luego por último tenemos los datos devueltos por el usuario:


```
{"provider":"microsoft","name":{"familyName":"Hern6ndez M6ndez","givenName":"Nelson Adonis"},"id":"17b7e49b-5ff2-46d3-a79a-bc587b3d1b06","displayName":"Nelson Adonis Hern6ndez M6ndez","emails":
[{"type":"work","value":"HM211512@alumno.udb.edu.sv"}],"_raw":"{\"@odata.context\":\"https://graph.microsoft.com/v1.0/$metadata#users/$entity\",\"businessPhones\":[],\"displayName\":\"Nelson
\adonis Hern\\ufffdndez M\\ufffdndez\",\"givenName\":\"Nelson
\adonis\",\"jobTitle\":null,\"mail\":\"HM211512@alumno.udb.edu.sv\",\"mobilePhone\":null,\"officeLocation\":null,\"preferredLanguage\":null,\"surname\":\"Hern\\ufffdndez
4\\ufffdndez\",\"userPrincipalName\":\"HM211512@alumno.udb.edu.sv\",\"id\":\"17b7e49b-5ff2-46d3-a79a-bc587b3d1b06\"}","_json":
{"@odata.context":"https://graph.microsoft.com/v1.0/$metadata#users/$entity","businessPhones":[],"displayName":"Nelson Adonis Hern6ndez M6ndez","givenName":"Nelson
\adonis","jobTitle":null,"mail":"HM211512@alumno.udb.edu.sv","mobilePhone":null,"officeLocation":null,"preferredLanguage":null,"surname":"Hern6ndez
46ndez","userPrincipalName":"HM211512@alumno.udb.edu.sv","id":"17b7e49b-5ff2-46d3-a79a-bc587b3d1b06"}}
```