Facultad de Ingeniería y Arquitectura

Vistas ASP.NET MVC

Ing. Jose Emerson Aguilar

Almacenar archivos en Firebase Storage

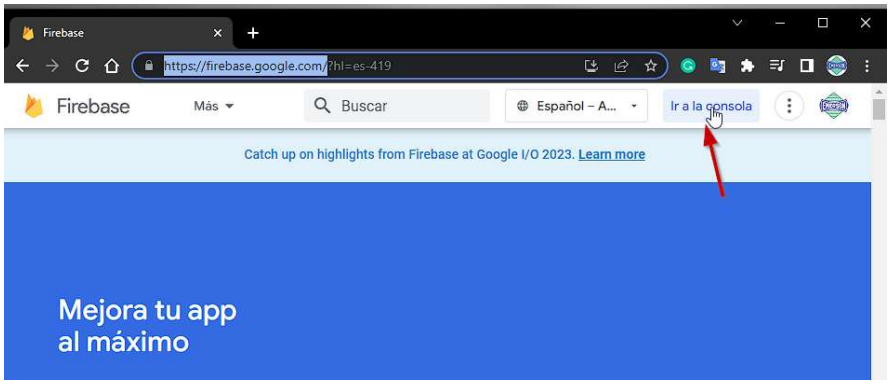
Google Firebase Storage

Cloud Storage para Firebase es un servicio de almacenamiento de objetos potente, simple y rentable construido para el escalamiento de Google.

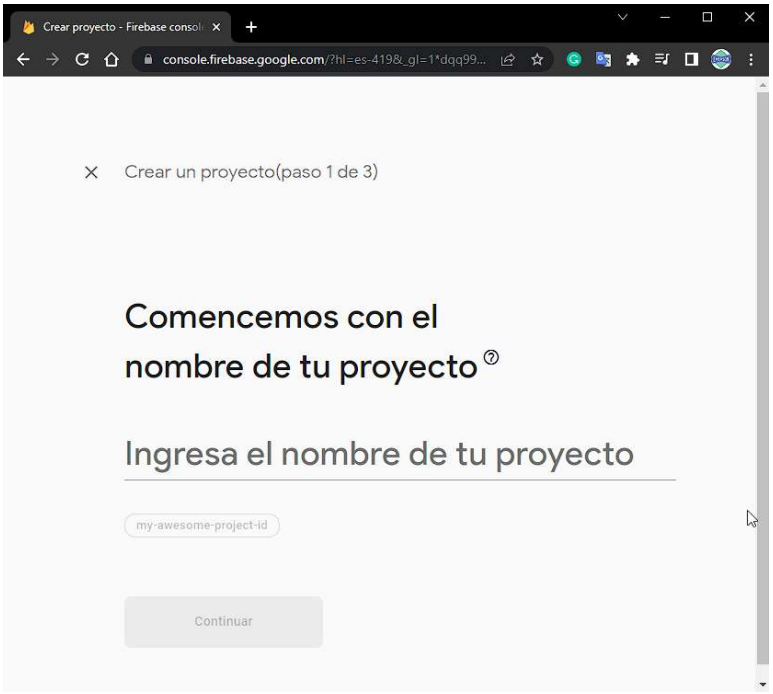
En esta guía utilizaremos dicho servicio para almacenar archivos desde proyecto ASP.MVC Core en Firebase Storage.

Configuración de Firebase Storage

Ingresamos a sitio de Firebase (https://firebase.google.com/) y abrimos la consola

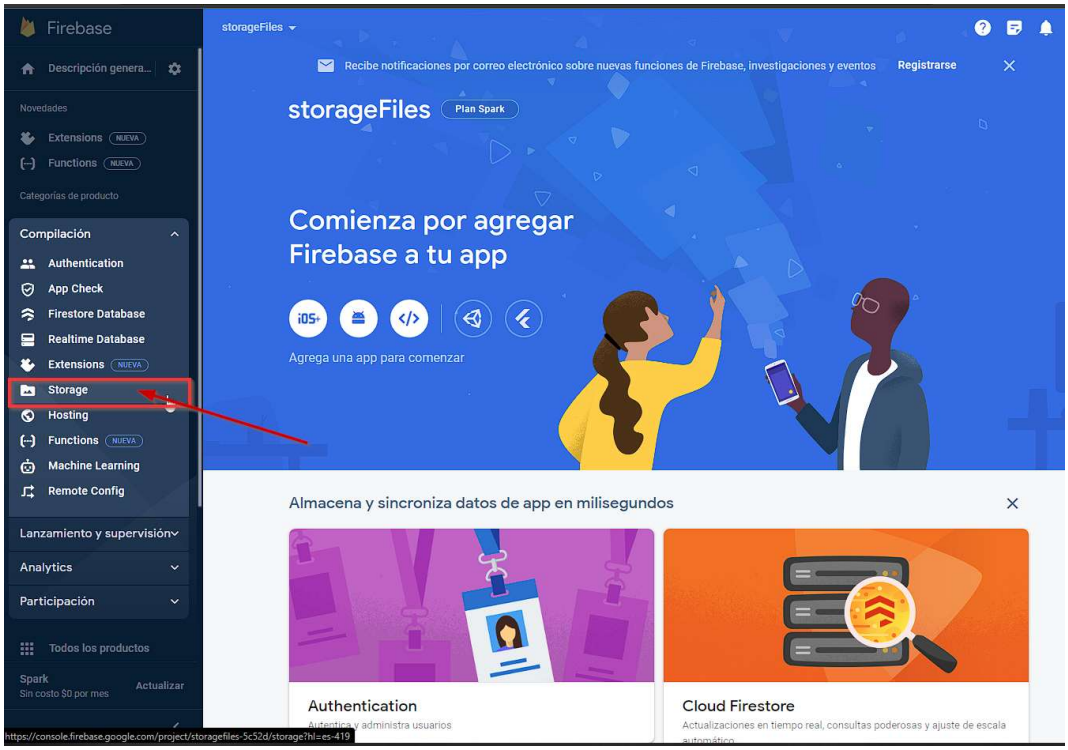


Agregamos un nuevo proyecto, indicando como primer paso el nombre del proyecto

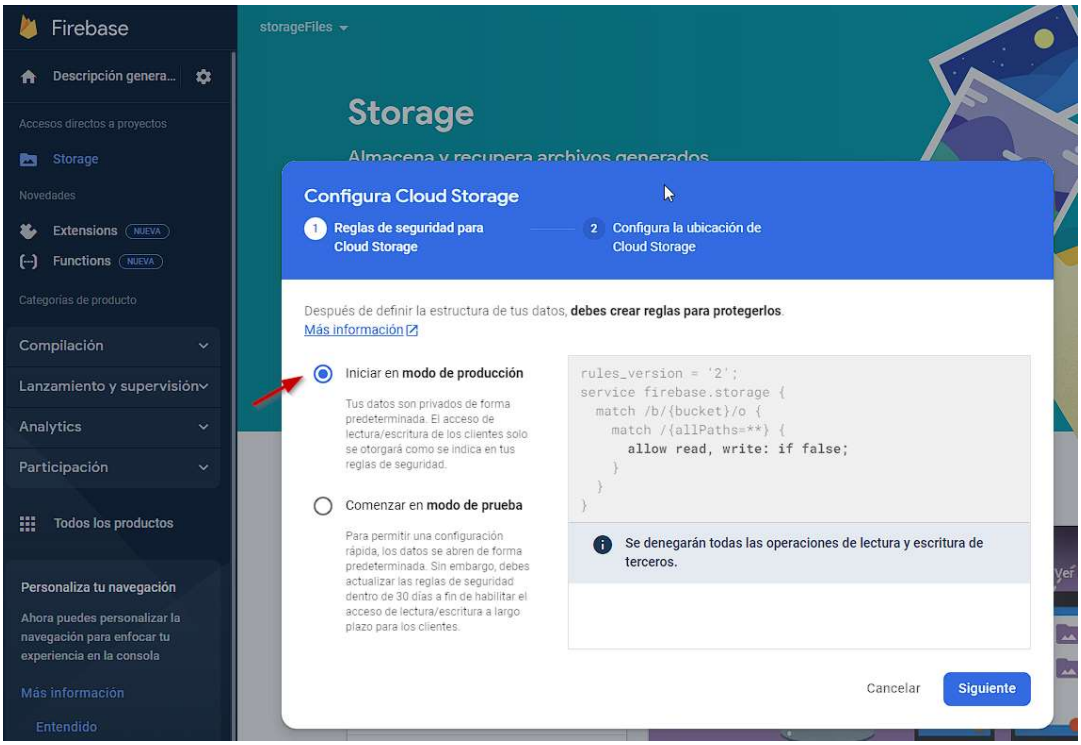


Almacenamiento de Archivos en Firebase

Con el proyecto creado, procedemos a configurar el espacio para almacenamiento de Firebase, para ello en el menú de COMPILACION seleccionamos STORAGE



Posteriores damos click en el botón de Comenzar y seleccionamos la opción de Iniciar en modo de producción, damos en Siguiente y posterior en Listo

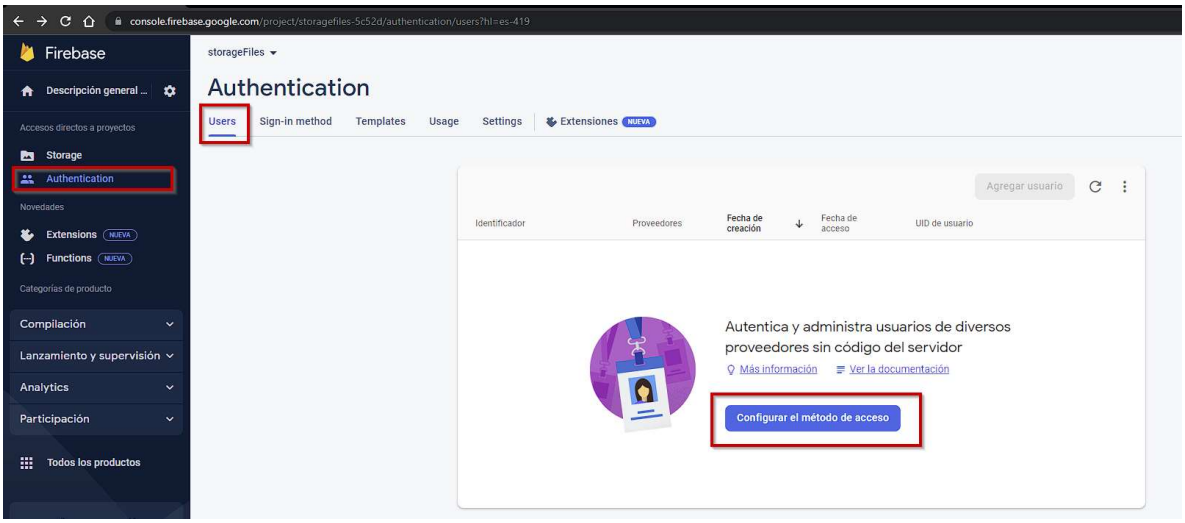


Almacenamiento de Archivos en Firebase

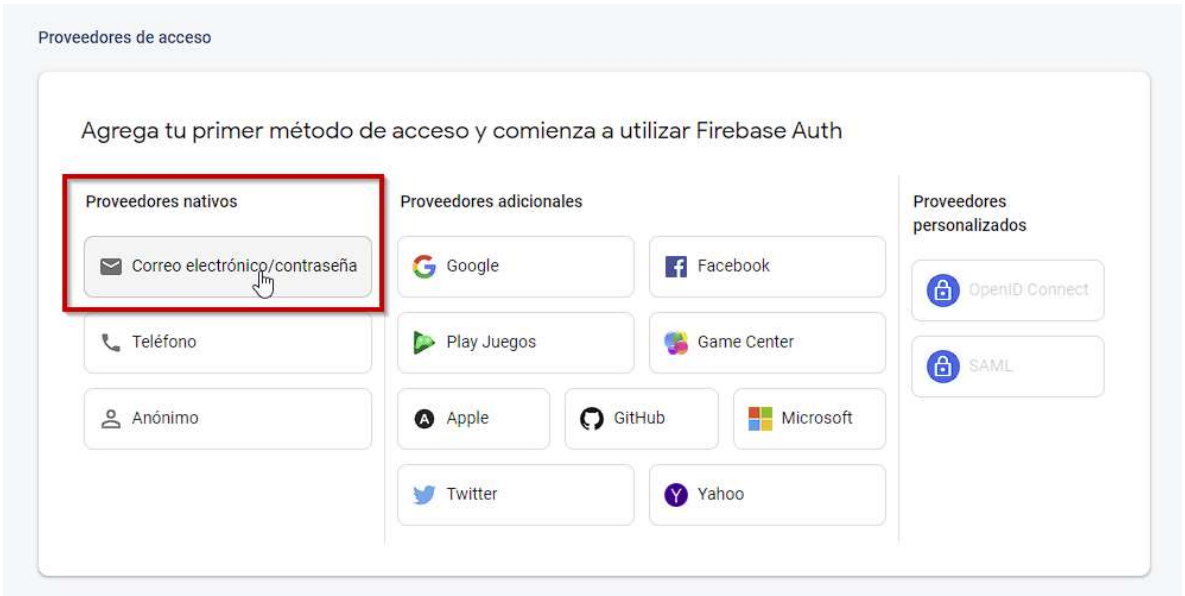
Variables de conexión

Para poder cargar archivos en Firebase Storage, se necesitan cuatro valores de configuración que se utilizaran en el método de C# que cargara el archivo, los cuales son Email y Clave de autenticación de Firebase, Ruta donde se almacenaran los archivos y el API KEY de nuestro proyecto de Firebase.

Para obtener el EMAIL y CLAVE para autenticación de Firebase, para ello nos dirigimos a la pestaña de Authentication, en User y le damos Configurar el método de acceso.

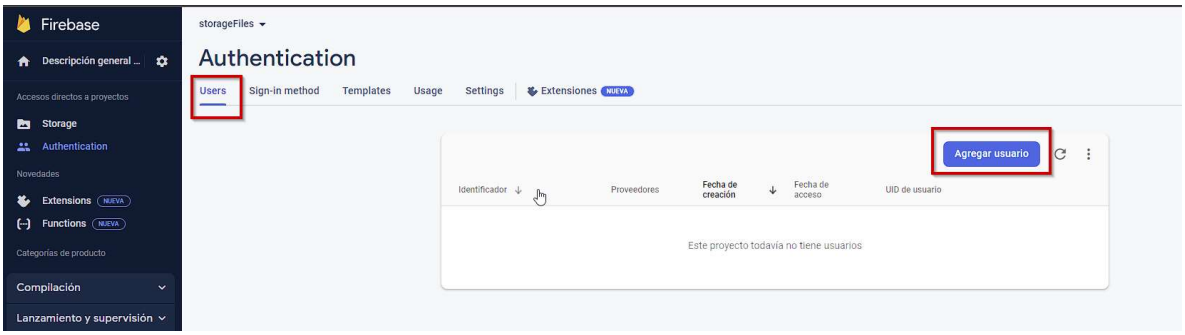


Seleccionamos de proveedores nativos, la opción de Correo electrónico/contraseña



Se habilita la opción de Correo Electrónico y luego regresamos a User y agregamos un usuario.

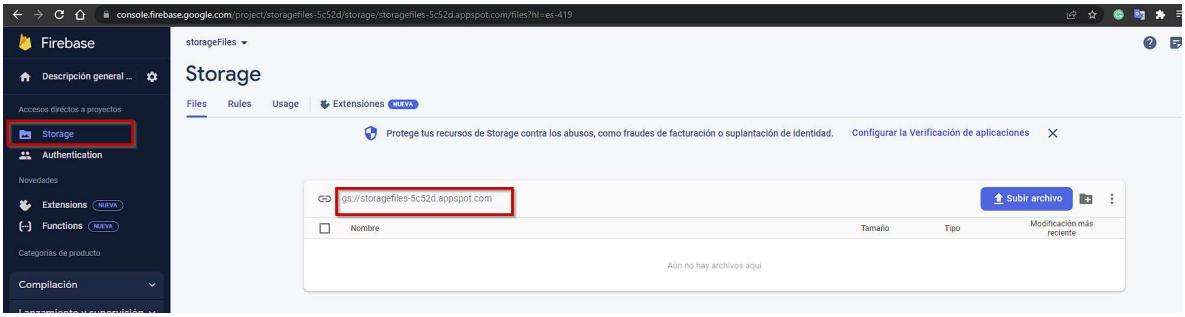
Almacenamiento de Archivos en Firebase



Asignamos correo electrónico existente y establecemos la contraseña

Este usuario es el que usaremos en el controlador para conectarnos.

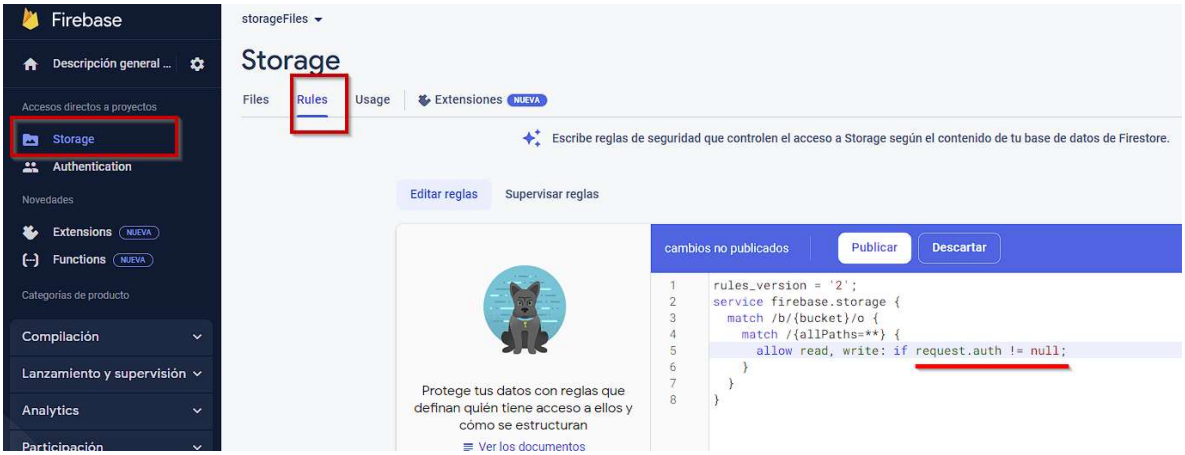
Continuamos y ubicamos la ruta del servicio storage.



Almacenamiento de Archivos en Firebase

Por último necesitamos el api_key para conectarnos al proyecto de Firebase, para ello nos vamos a Authentication y en generales, copiamos toda la API Key

Por último, se necesita cambiar una regla en la configuración del Storage para que nos permita conectarnos con el usuario creado, para ello en la opción de Storage, en la pestaña de Rules modificaremos la línea 5, donde cambiamos el valor de true a request.auth != null

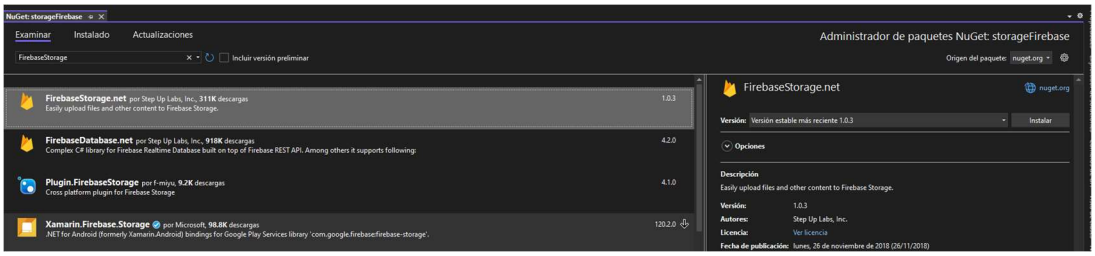


Damos click en Publicar para guardar los cambios, y con esto podemos iniciar la configuración del método en el proyecto ASP.NET MVC Core.

Almacenamiento de Archivos en Firebase

Implementación en proyecto MVC Core

En el proyecto ASP.NET MVC Core que deseamos utilizar Firebase, debemos instalar los paquetes necesario, para ello en el administrador de paquetes NuGet, buscamos Firebase.auth y FirebaseStorage.net



Formulario

Creamos un formulario que contenga un input de tipo file

El código html sería el siguiente:

```
@{
    ViewData["Title"] = "Carga de Archivos";
}

<div class="text-center">
    <h1 class="display-4">Welcome Dev</h1>
    <p>Learn about <a href="https://docs.microsoft.com/aspnet/core">Cargar archivos a Firebase</a>.</p><br/>

    <form asp-action="SubirArchivo" method="post" enctype="multipart/form-data">

        <div class="form-group">
            <label for="archivo">Seleccione el archivo a cargar</label>
            <input type="file" class="form-control" id="archivo" name="archivo">
        </div>

        <button type="submit" class="btn btn-primary mb-2">Subir Archivo</button>

    </form>

    <br/>

</div>
```

Almacenamiento de Archivos en Firebase

Debemos de asignar el **asp-action** indicando el nombre del método al cual enviaremos el archivo, el tipo de método, en este caso post y en este caso, como se manejan archivos debemos indicar el **enctype** como **multipart/form-data**.

El input tipo file debe tener como nombre el parámetro a recibir en el método.

Método de subir archivos

El método en este caso se llama SubirArchivo que recibirá un parámetro de tipo IFormFile

```
[HttpPost]
0 referencias
public async Task<ActionResult> SubirArchivo(IFormFile archivo)
{



    return RedirectToAction("VerImagen");

}
```

Creamos las variables que contendrán los cuatro valores que configuramos en Firebase en la sección anterior, como lo son email, clave, ruta y api_key

```
[HttpPost]
0 referencias
public async Task<ActionResult> SubirArchivo(IFormFile archivo)
{
    //Leemos el archivo indicado
    Stream archivoASubir = archivo.OpenReadStream();

    //Configuramos la conexion hacia firebase
    string email = "asp@mail.com";   //Correo para autenticar en firebase
    string clave = "delunoalnueve";  //Contraseña para autenticar en firebase
    string ruta = "storagefiles-5c52d.appspot.com";   //Lugar donde se guardan los archivos
    string api_key = "AIzaSyB8QPVFCvNm                    ';//Identificador del proyecto firebase a utilizar en el proyecto MVC


    return RedirectToAction("VerImagen");

}
```

Posterior agregamos la autenticación a Firebase, para ello indicamos primero la api_key de nuestro proyecto

```
var auth = new FirebaseAuthProvider(new FirebaseConfig(api_key));
```

Seguido por el email y clave

```
var autenticarFireBase = await auth.SignInWithEmailAndPasswordAsync(email, clave);
```

Creamos un token de cancelación y el token del usuario que se acaba de autenticar requeridos al momento de enviar el archivo.

```
var cancellation = new CancellationTokenSource();
var tokenUser = autenticarFireBase.FirebaseToken;
```

Almacenamiento de Archivos en Firebase

Se procede a crear la configuración del envió al storage donde indicamos:

- Ruta del storage
- Token del usuario autenticado
- Carpeta que utilizaremos en la ruta del storage
- Nombre del archivo
- El archivo por subir
- Y el token de cancelación

```
var tareaCargarArchivo = new FirebaseStorage( ruta,
                                              new FirebaseStorageOptions
                                              {
                                                  AuthTokenAsyncFactory = () => Task.FromResult(tokenUser), ThrowOnCancel = true
                                              }
                                            ).Child("Arhivos")
                                            .Child(archivo.FileName)
                                            .PutAsync(archivoASubir, cancellation.Token);
```

Ejecutamos la carga hacia el storage con el siguiente código

```
var urlArchivoCargado = await tareaCargarArchivo;
```

Esta tarea, al finalizar retorna la URL del archivo almacenado, la cual podemos guardar en algún campo de la base de datos y posterior recuperarlo para usarlo como src de algún elemento img o simplemente para que el usuario pueda descargar el archivo.

Los servicios de Firebase se pueden utilizar desde diferentes lenguajes de programación, la documentación oficial la pueden encontrar en https://firebase.google.com/docs/storage?hl=es-419