

UD 5: Firebase Realtime Database y Xamarin.Forms

Desarrollo de Interfaces

Índice

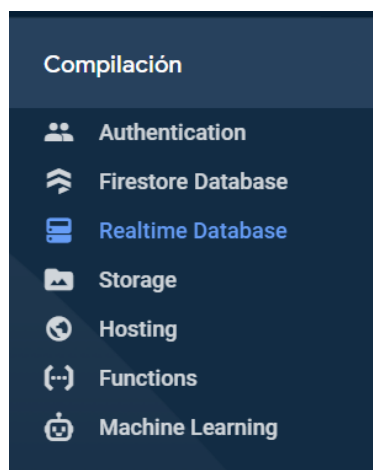
1. Firebase Realtime Database	1
2. Firebase Realtime Database en Visual Studio (Xamarin.Forms)	3

1. Firebase Realtime Database


Firebase Realtime Database nos ofrece la posibilidad de generar nuestra base de datos en la nube mediante JSON y sincronización en tiempo real para cada dispositivo conectado. Para poder hacer uso de esta base de datos necesitaremos tener una cuenta de Google y acceder a la url: <https://console.firebase.google.com/>

Dentro de esta página web tenemos que agregar un proyecto, darle un nombre, establecer si queremos utilizar google analytics en nuestro proyecto y esperar a que este se genere.

Una vez dentro de nuestro proyecto, en el apartado “Compilación” (menú lateral izquierdo), elegiremos la “Realtime Database” para su gestión.



Lo primero será generar nuestra base de datos que crearemos por defecto sin definir ninguna tabla ni nada (hay que recordar que trabaja como JSON). Una vez tengamos la base de datos creada, nos interesará la url de nuestra base de datos que necesitaremos más adelante.

 <https://desarrollo-interfaces-default-rtdb.europe-west1.firebaseio.com/>

Aquí podemos ver un ejemplo de la url generada para un proyecto llamado Desarrollo Interfaces en el servidor europe-west.

Además de esta dirección, nos hace falta el secreto de la base de datos que podremos obtener desde la misma consola de administración de Firebase. Para ello haremos click en el engranaje de la parte superior izquierda y seleccionaremos “Configuración de proyecto”.

Dentro de este apartado, en la parte superior elegiremos la pestaña “Cuentas de servicio” y seleccionaremos el apartado “Secretos de la base de datos”. Nos aparecerá una lista de nuestras bases de datos con su secreto oculto (se puede mostrar en cualquier momento sin problema).

The screenshot shows the 'Database Secrets' page in the Firebase Admin SDK. The left sidebar has a key icon for 'SDK de Firebase Admin' and a document icon for 'Secretos de la base de datos'. Below the sidebar, it says 'Credenciales heredadas' and 'Todas las cuentas de servicio' with a gear icon and '4 cuentas de servicio'. The main content area has a warning box: 'Actualmente, los secretos de la base de datos son obsoletos y usan un generador de tokens de Firebase heredado. Actualiza el código fuente con el SDK de administrador de Firebase.' with a 'Más información' link. Below this, it says 'Crea tokens de autenticación personalizados para la base de datos con un generador de tokens de Firebase heredado. Debe haber al menos un secreto en todo momento. Más información'. There is an 'Agregar secreto' button. A table lists the database secrets:

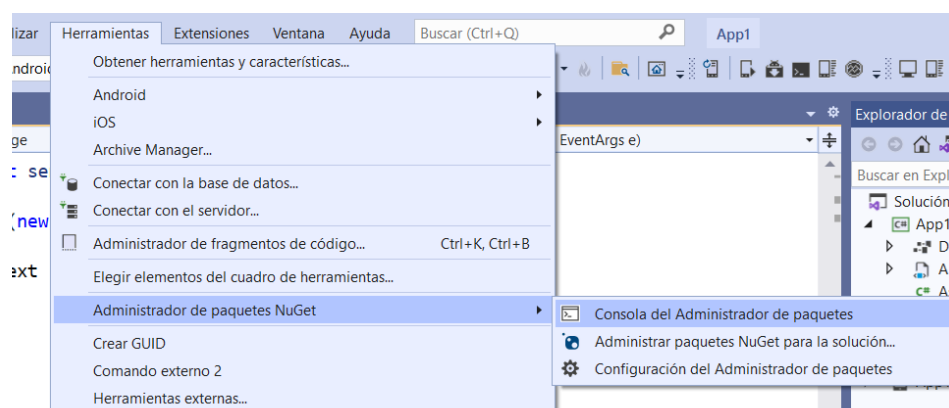
Base de datos	Secreto
desarrollo-interfaces-default-rtdb	S2vGG09hBiGdm7QxomtCZNHQREHNGb1hhL5oBGkN

Con estos datos ya podremos ir a nuestro proyecto de Visual Studio de Xamarin.Forms para configurarlo y trabajar con la base de datos que acabamos de crear y los datos para poder utilizarla.

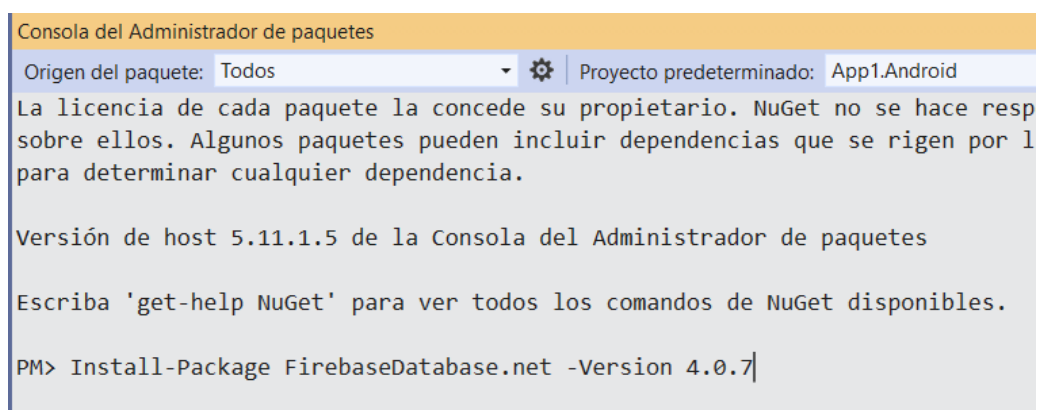
2. Firebase Realtime Database en Visual Studio (Xamarin.Forms)

La forma más sencilla de instalar los paquetes necesarios para trabajar con Firebase Database es mediante NuGet y el administrador de paquetes. En el siguiente enlace tenéis información sobre cómo instalarlo: <https://www.nuget.org/packages/FirebaseDatabase.net/>





En nuestro caso, generamos el proyecto de Xamarin.Forms y seleccionamos “Herramientas” > “Administrador de paquetes NuGet” > “Consola del Administrador de paquetes”.



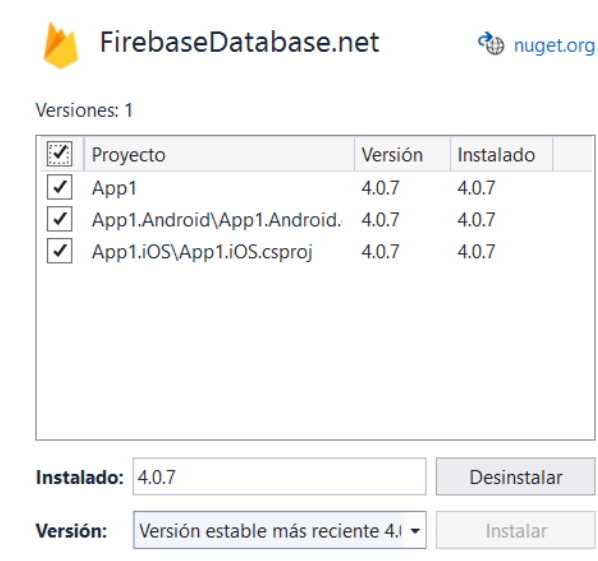
Entonces se nos abrirá una consola e introduciremos el comando para instalar el paquete de Firebase Database: **Install-Package FirebaseDatabase.net -Version 4.0.7**. Ahora reiniciamos el Visual Studio.



En este punto ya tendremos Firebase Database dentro de Visual Studio. Para confirmarlo, haremos click derecho en nuestra solución > “Administrar paquetes NuGet para la solución”. En esta ventana deberá ser similar a la siguiente (lo importante es que aparezca FirebaseDatabase.net).

	FirestoreDatabase.net por Step Up Labs, Inc.	4.0.7
	Complex C# library for Firestore Realtime Database built on top of Firestore REST API. Among others it supports following:	
	NETStandard.Library por Microsoft	2.0.3
	A set of standard .NET APIs that are prescribed to be used and supported together. 18a36291e48808fa7ef2d00a764ceb1ec95645a5	
	Xamarin.Essentials por Microsoft	1.7.1
	Xamarin.Essentials: a kit of essential API's for your apps	
	Xamarin.Forms por Microsoft	5.0.0.2337
	Build native UIs for iOS, Android, UWP, macOS, Tizen and many more from a single, shared C# codebase	

Es aconsejable actualizar dichos paquetes si no lo están ya. En este punto tendremos Firestore Database en la solución pero deberemos incluirlo en nuestros proyectos para poder utilizarlo. De modo que seleccionamos el paquete de FirestoreDatabase.net y en la parte de la derecha, seleccionamos todos los proyectos e instalamos la versión reciente más estable.



Con Firestore dentro de nuestro proyecto, ya podremos hacer uso de sus elementos para incluir dicha base de datos dentro de nuestra aplicación. En concreto utilizaremos el objeto FirestoreClient para utilizar la Realtime Database de Firestore tanto para hacer peticiones de los datos como añadir nuevos (hay que recordar que trabaja como si fuera JSON). A continuación tenéis la definición del objeto

```
public FirebaseClient fc = new FirebaseClient(FirestoreClient, new FirebaseOptions {
    AuthTokenAsyncFactory = () => Task.FromResult(FirestoreSecret) });
```

FirestoreClient será la url de nuestra base de datos y FirestoreSecret el secreto de dicha BD.

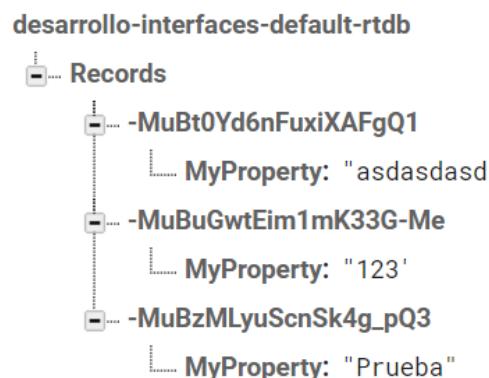
Para incluir datos a nuestra base de datos, utilizaremos la definición de un objeto propio que hará de modelo para las “tablas” de la base de datos (los objetos JSON que representan esos datos). Aquí tenéis un ejemplo con un objeto propio y la petición para crear ese elemento en la Base de Datos.

```
4 referencias
public class MyDatabaseRecord
{
    1 referencia
    public string MyProperty { get; set; }
}
```

Aquí tenéis la definición básica de un objeto con un campo de tipo string.

```
fc.Child("Records").PostAsync(new MyDatabaseRecord
{
    MyProperty = recordData.Text
});
```

Aquí tenéis la subida de los datos a la base de datos, almacenando esos elementos “MyDatabaseRecord” en ese conjunto llamado “Records”.



Captura de la base de datos con esos Records creados mediante la definición del objeto “MyDatabaseRecord”.