Analytics Overview

Google Analytics for Firebase provides free, unlimited reporting on up to 500 distinct events. The SDK automatically captures certain key events and user properties, and you can define your own custom events to measure the things that uniquely matter to your business.

Analytics surfaces data about user behavior in your iOS and Android and Web apps, enabling you to make better decisions about your app or game and marketing optimization. View crash data, notification effectiveness, deep-link performance, in-app purchase data, and more.

Check the official page for more information.

Setup

Before starting to use any Firebase extensions, you are required to follow some initial configuration steps. However if you've already done these for any of the other modules you can skip this configuration section and go straight to using the API functions.

- Create Project
- Platform Setup

Functions

The following functions are given for working with Analytics extension:

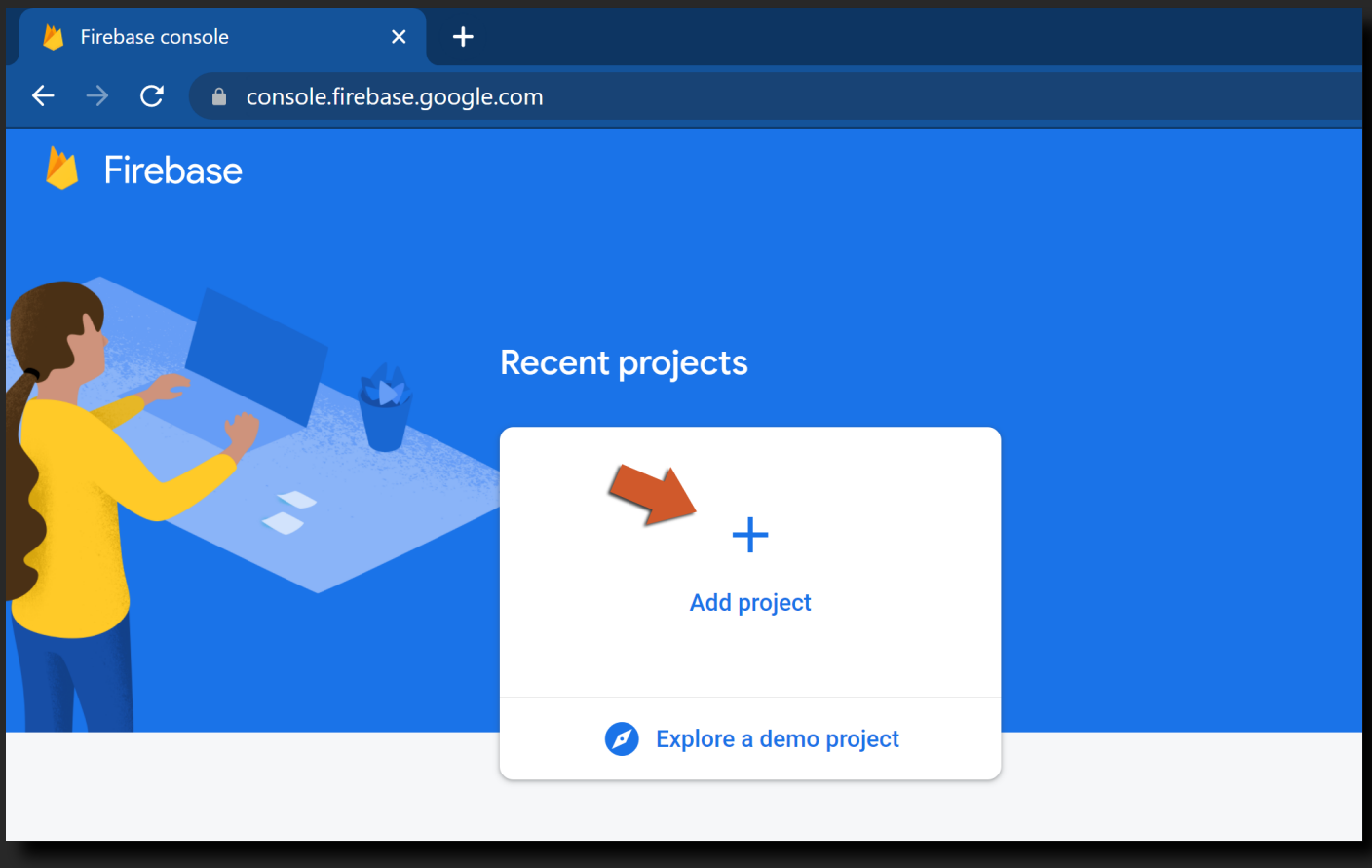
- FirebaseAnalytics_LogEvent

- FirebaseAnalytics_ResetAnalyticsData
- FirebaseAnalytics_SetAnalyticsCollectionEnabled
- FirebaseAnalytics_SetDefaultEventParameters
- FirebaseAnalytics_SetSessionTimeoutDuration
- FirebaseAnalytics_SetUserId
- FirebaseAnalytics_SetUserProperty

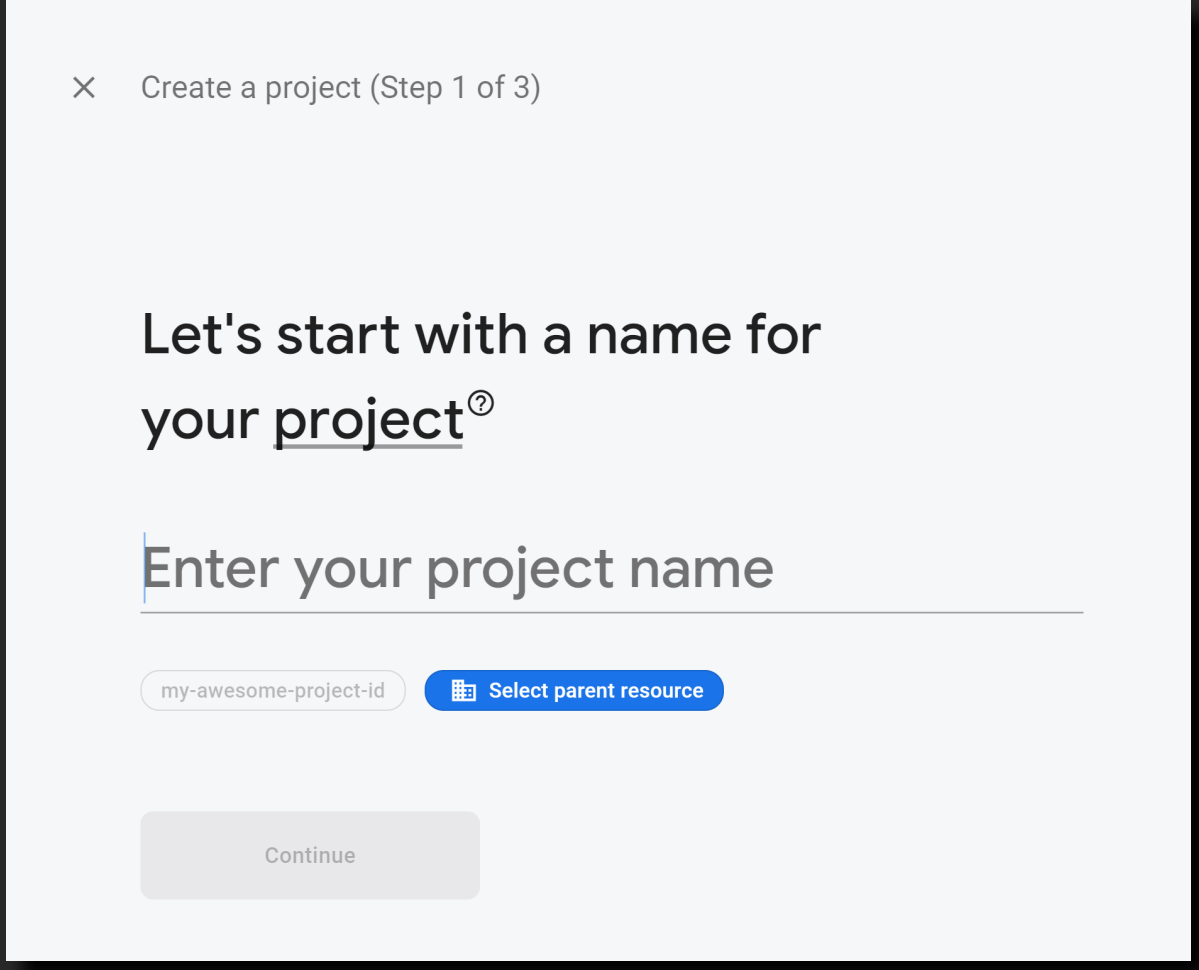
Create Project

Before working with any Firebase functions, you must set up your Firebase project:

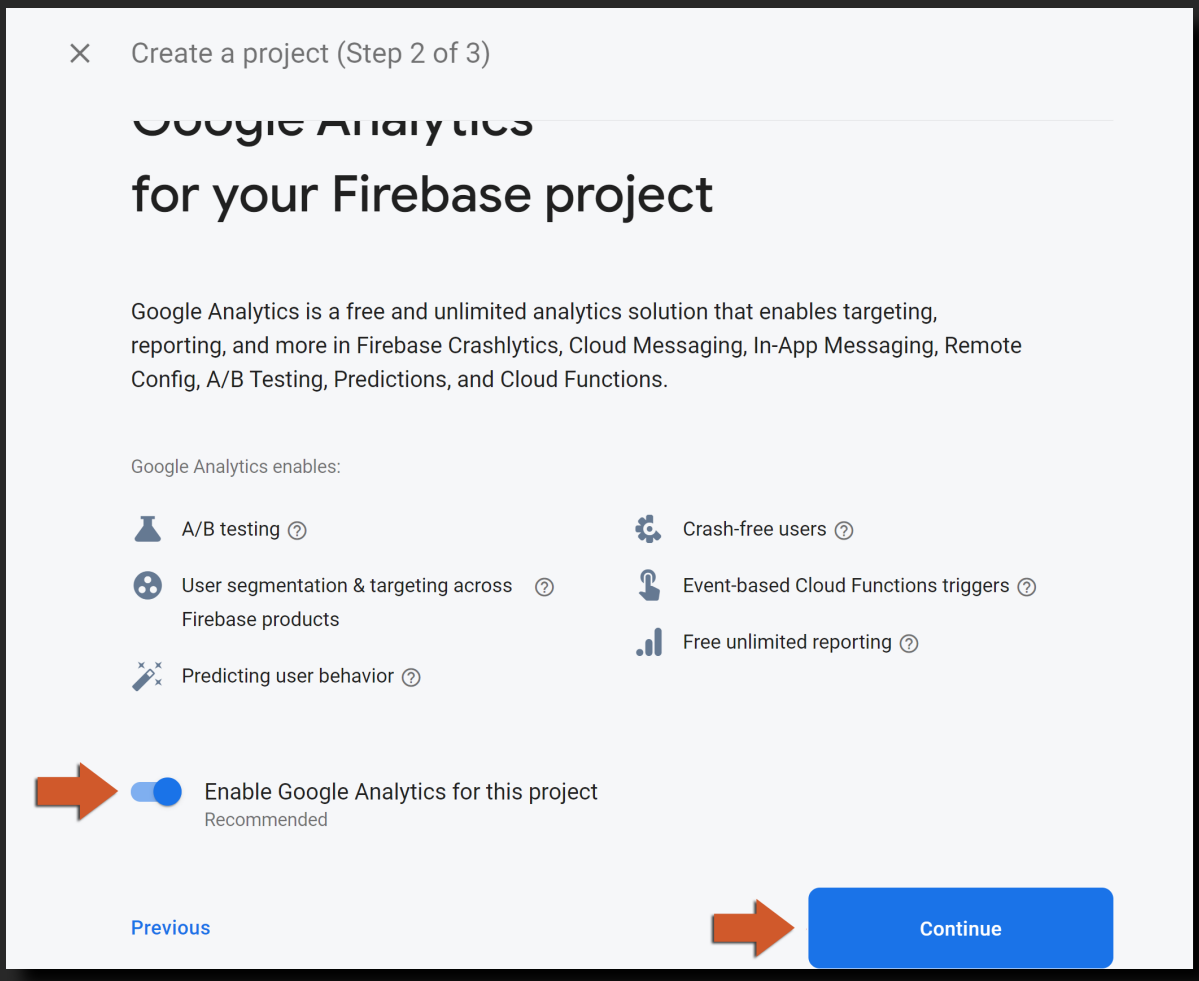
1. Go to the **Firebase Console** web site.
2. Click on **Add Project** to create your new project.



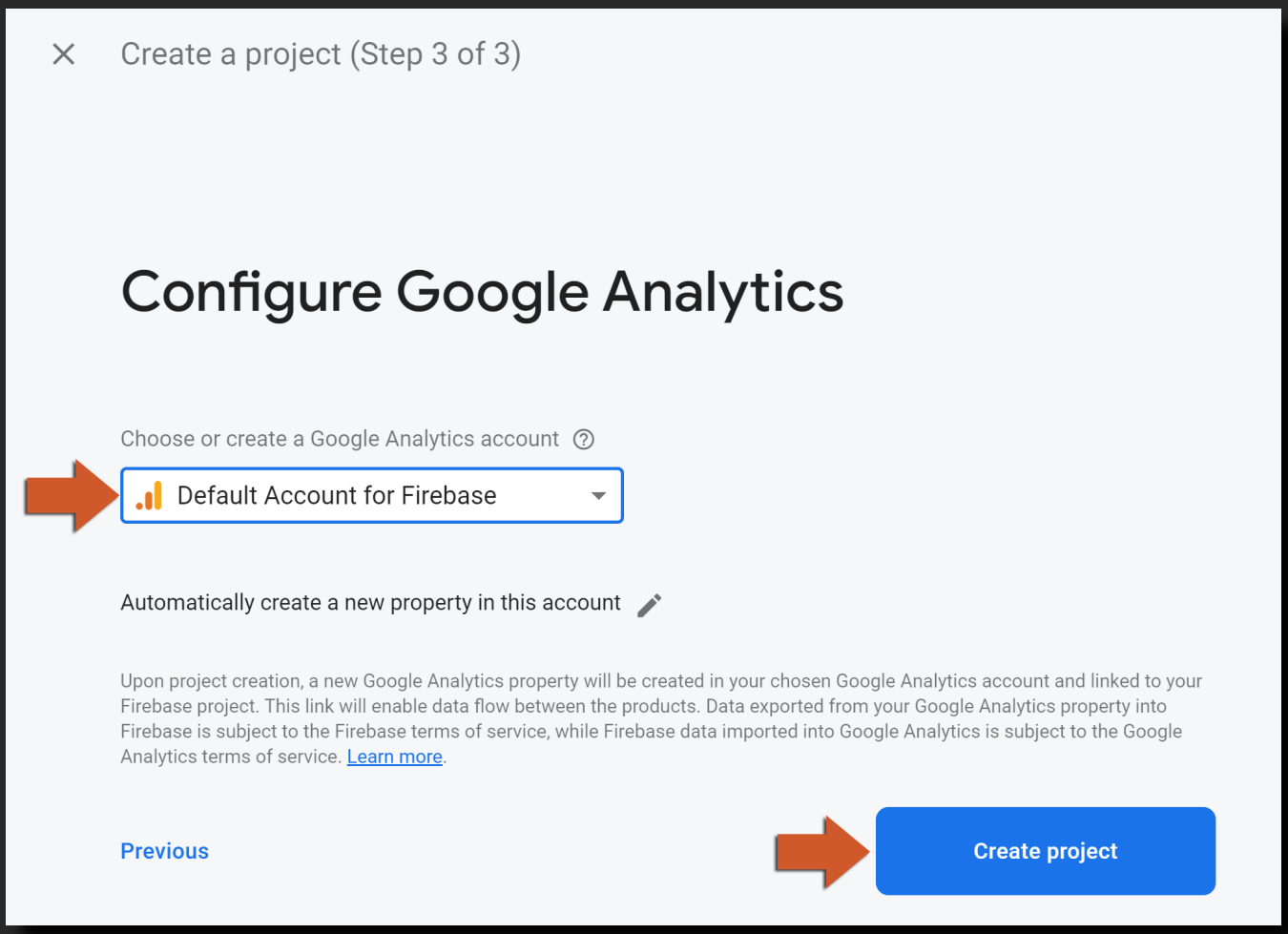
3. Enter a name for your project and click on the **Continue** button.



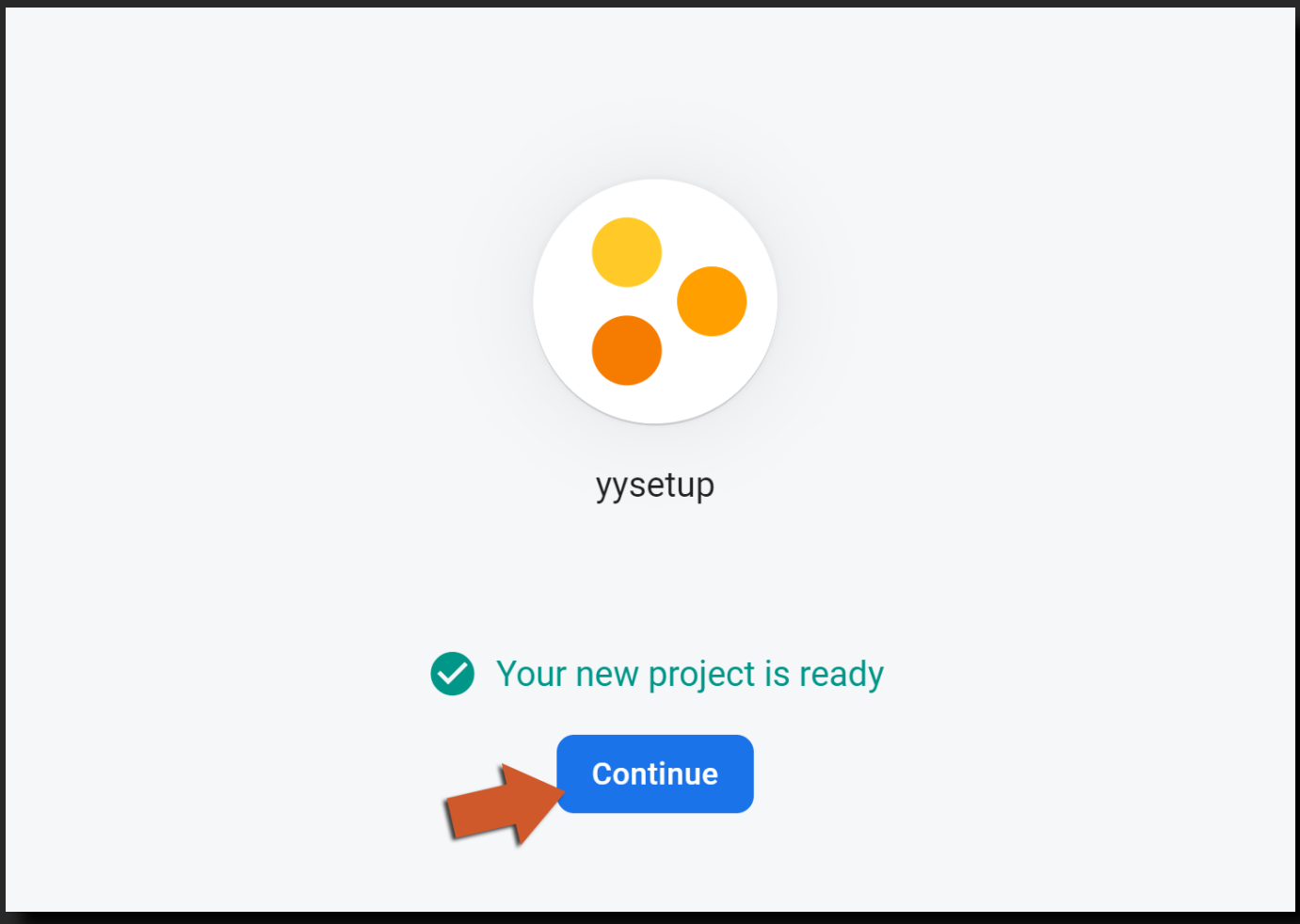
4. On the next page, make sure that **Enable Google Analytics for this project** is enabled and then click the **Continue** button:



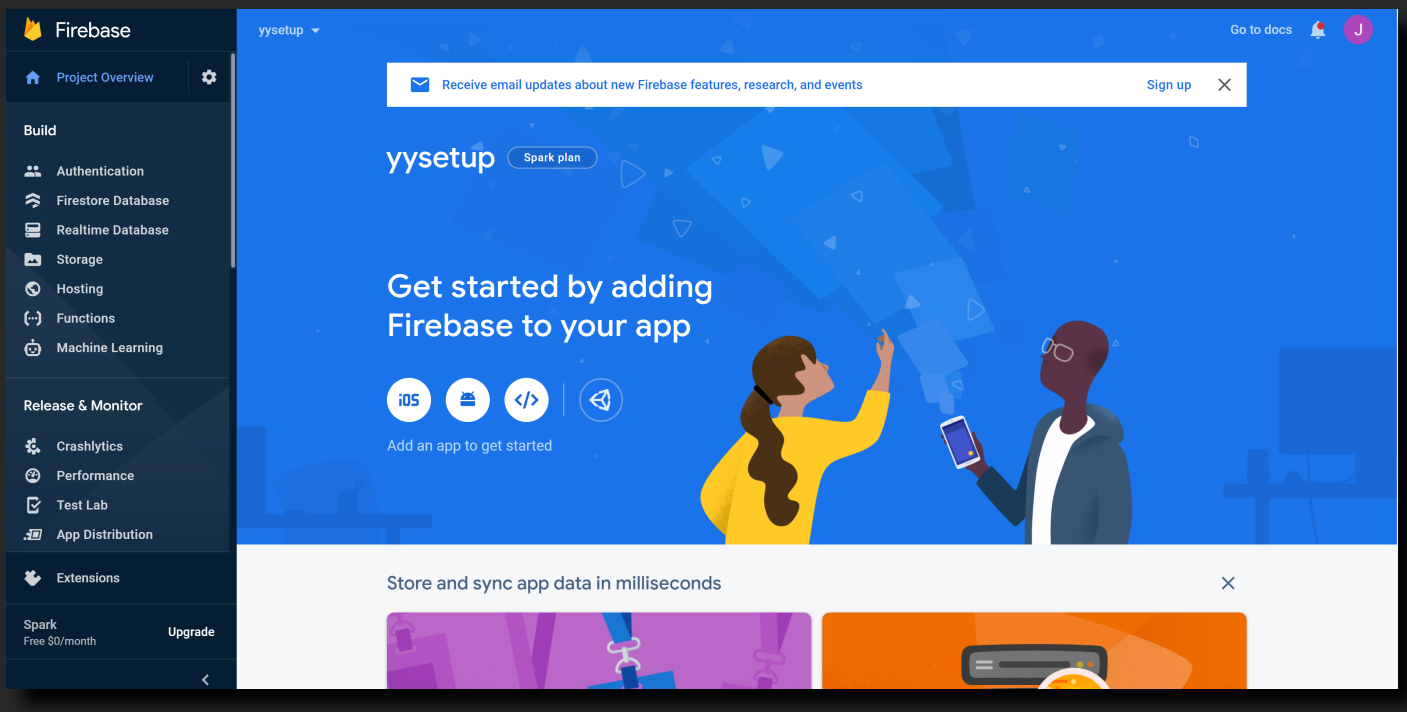
5. Select your account and click the **Create project** button:



6. Wait a moment until you project is created; after a few moments you should see the page shown below:



7. You will now be taken to your new project's home page:



8. Continue your adventure with the Firebase extensions provided for GameMaker!

Platform Setup

Firebase Analytics implementation uses SDK dependencies and therefore is only available on the **Android**, **iOS** and **Web** targets. In this section we will cover the required setup necessary to start using the Analytics extension on your game.

Select your target platform below and follow the simple steps to get your project up and running (you only need follow this topics once per project):

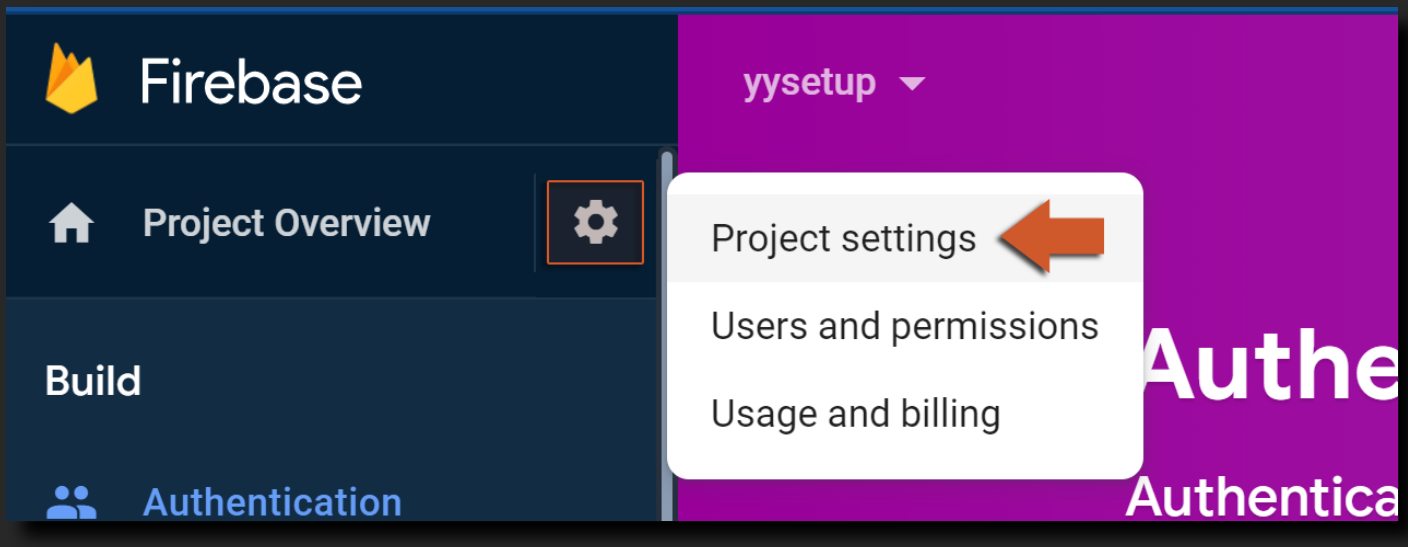
- Android Setup
- iOS Setup
- Web Setup

Android Setup

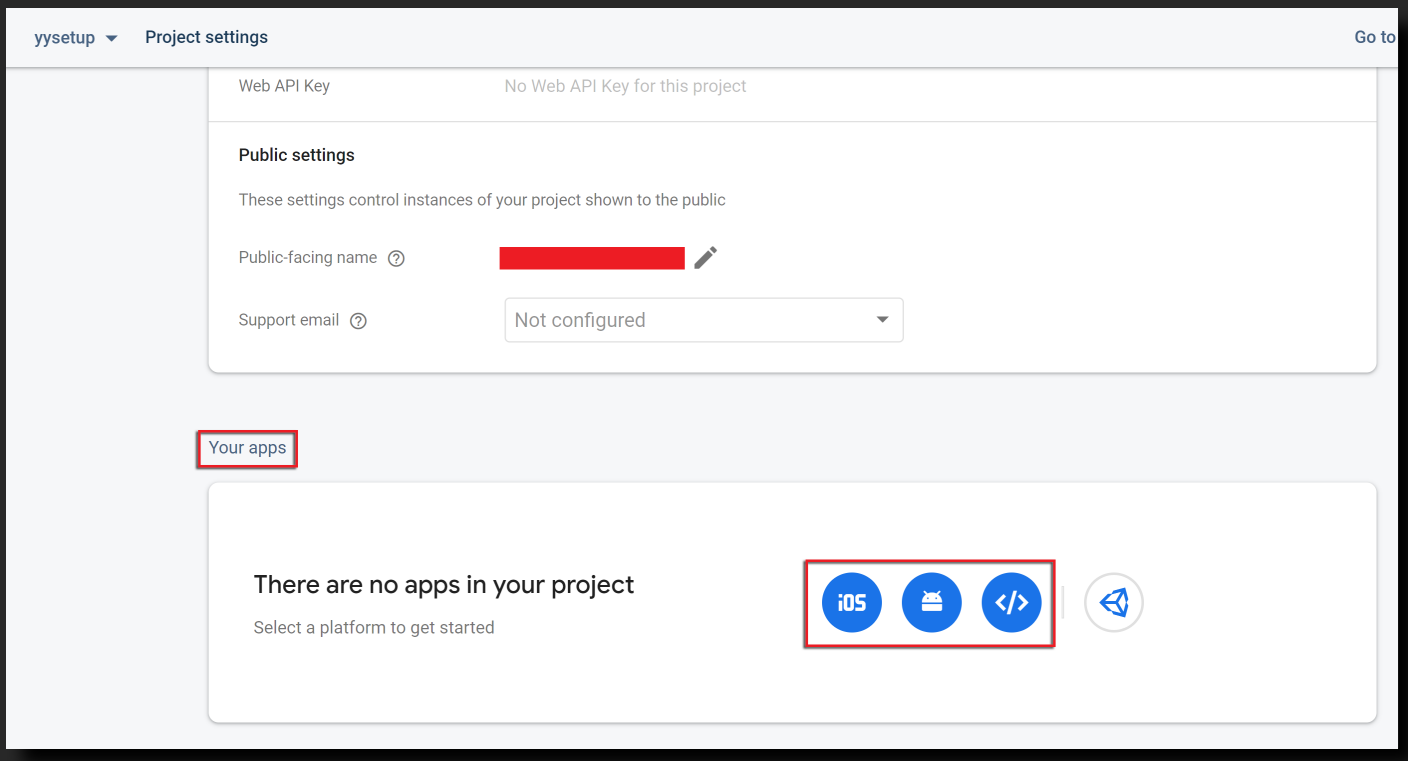
This setup is necessary for all the Firebase modules using Android and needs to be done once per project, and basically involves importing the **google-services.json** file into your project.

IMPORTANT Please refer to **this Helpdesk article** for instructions on setting up an Android project.

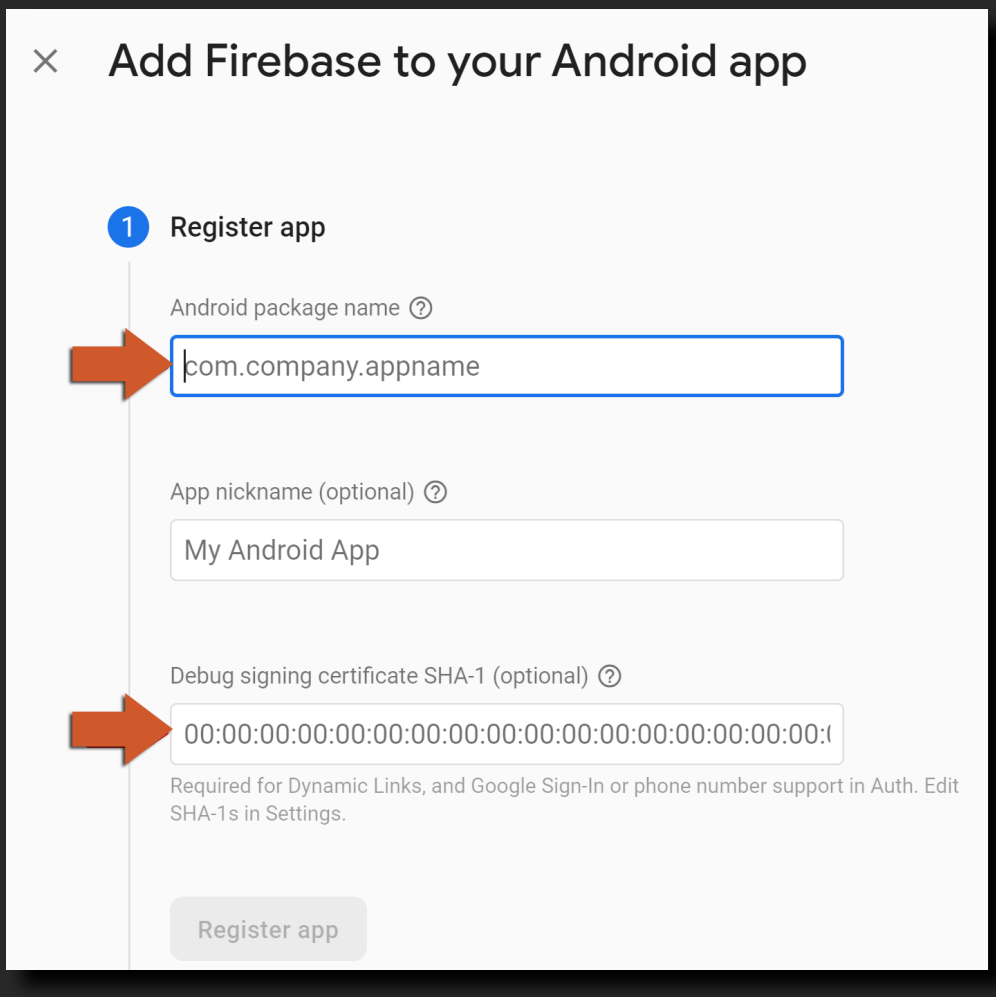
1. Click the **Settings** icon (next to Project Overview) and select **Project settings:**



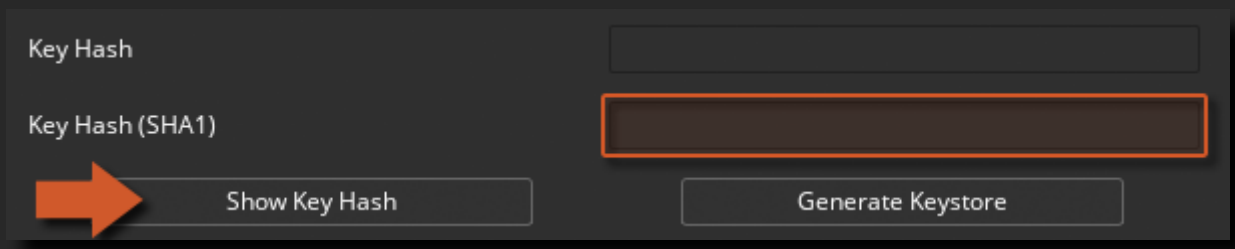
2. Now go to the **Your apps** section and click on the **Android** button:



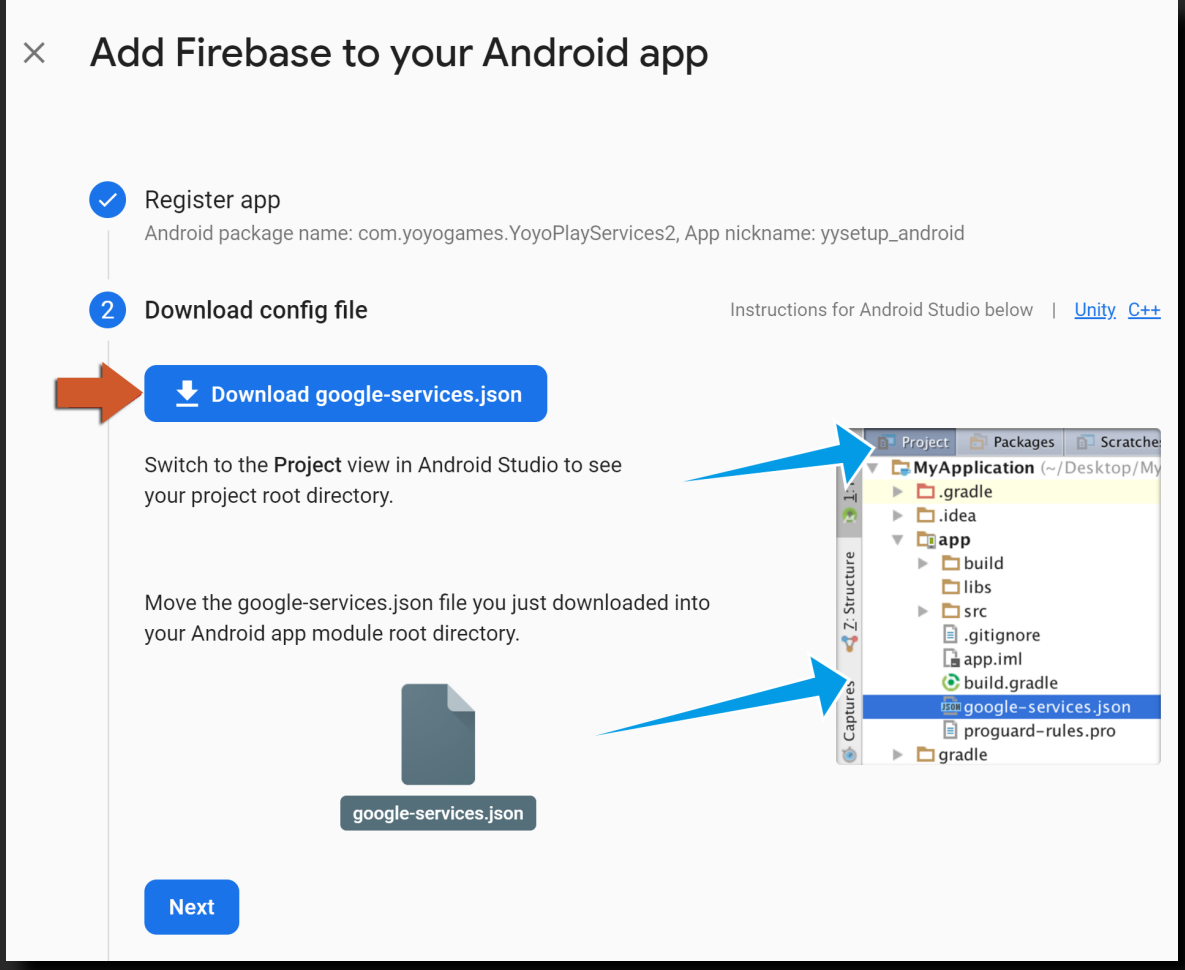
3. On this screen you need enter your **Package name** (required), **App nickname** (optional) and **Debug signing certificate SHA-1** (required if you are using Firebase Authentication).



You can get your package name from the **Android Game Options**, and your **Debug signing certificate SHA-1** from the **Android Preferences** (under Keystore):

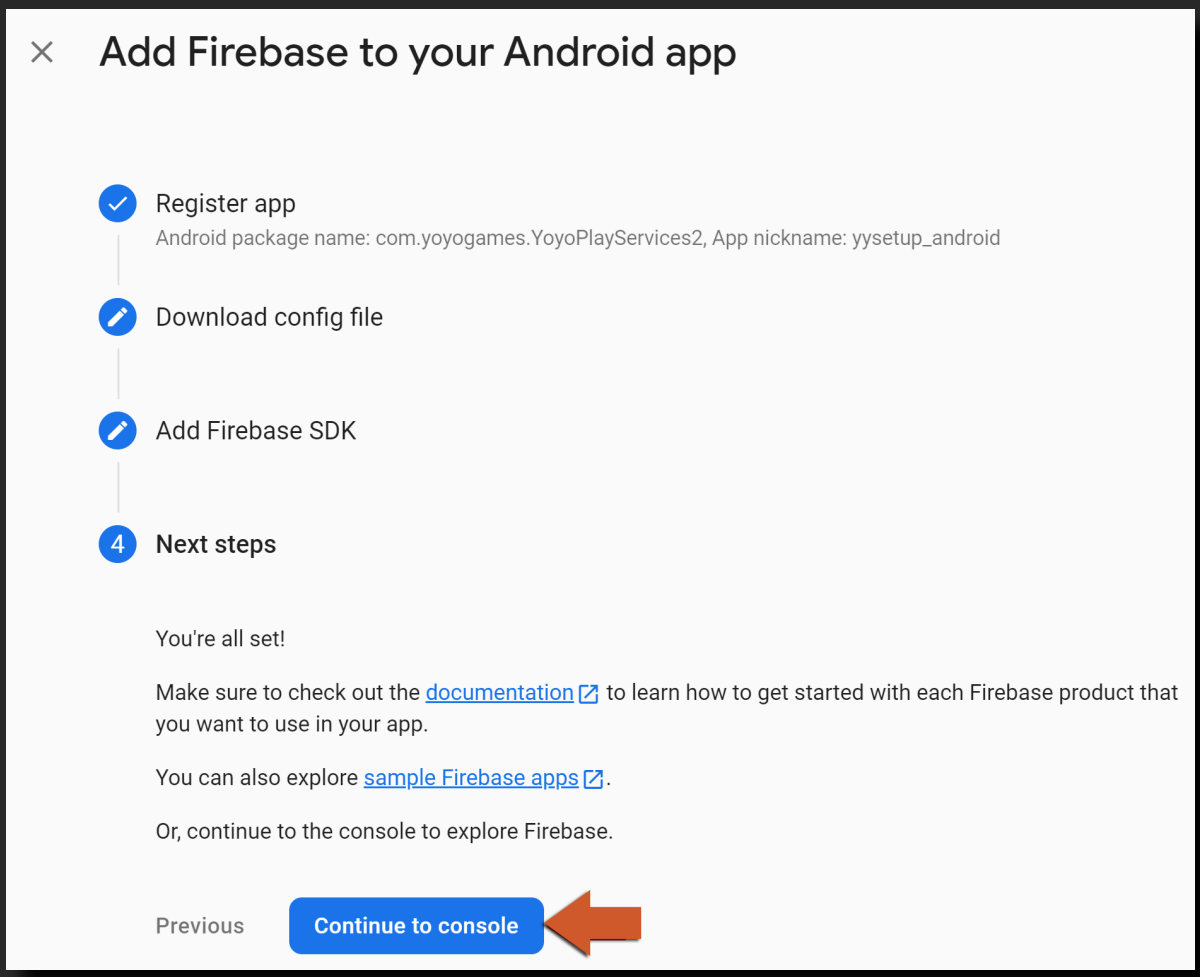


4. Click on **Download google-services.json** (make sure to save this file as we will need it in subsequent steps).

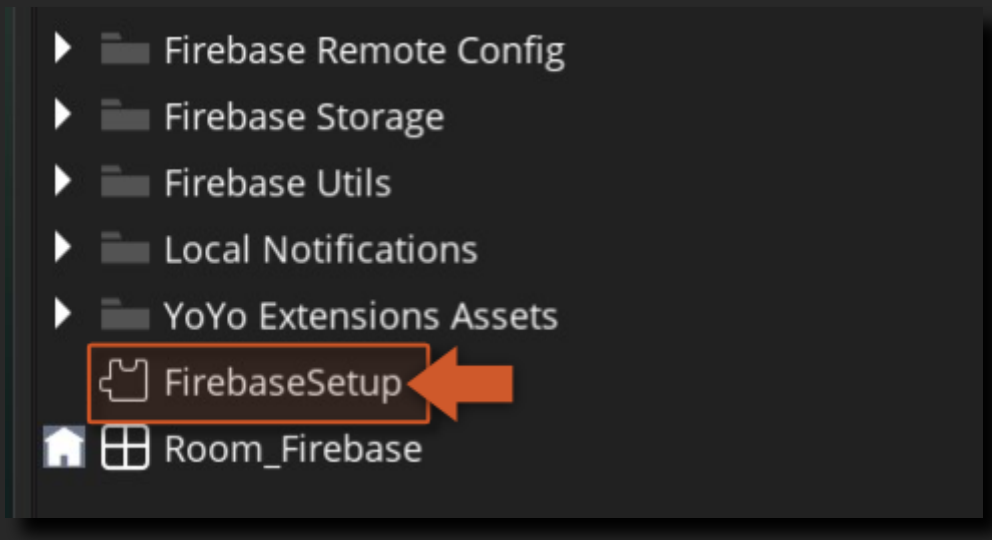


5. Ignore this screen, as this is already done in the extension.

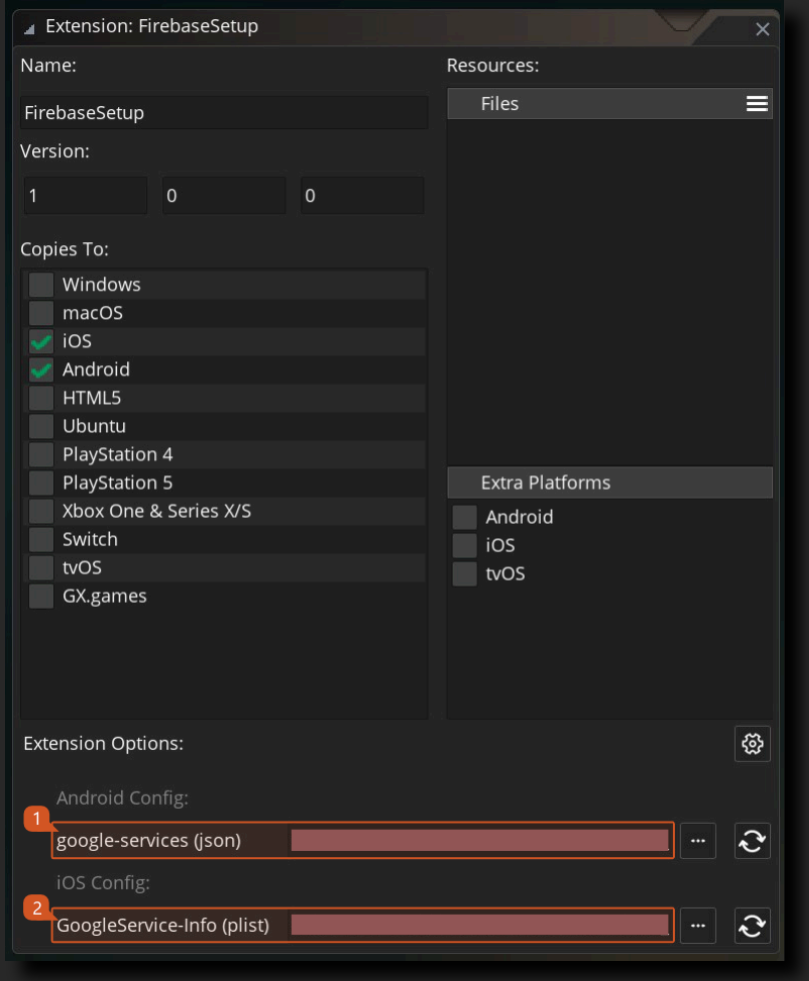
6. Click on the **Continue to console** button.



7. Now go into GameMaker, double click the extension **FirebaseSetup** asset.



8. In the extension panel just fill in the paths for the correct files (Android and/or iOS).



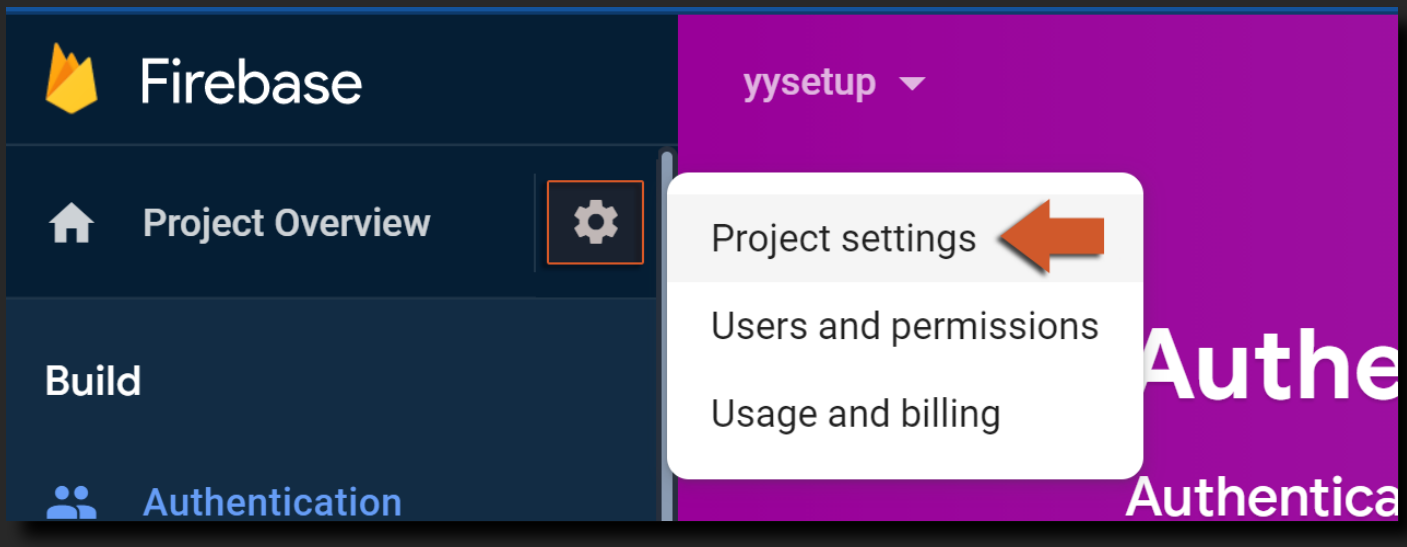
9. You have now finished the main setup for all Firebase Android modules!

iOS Setup

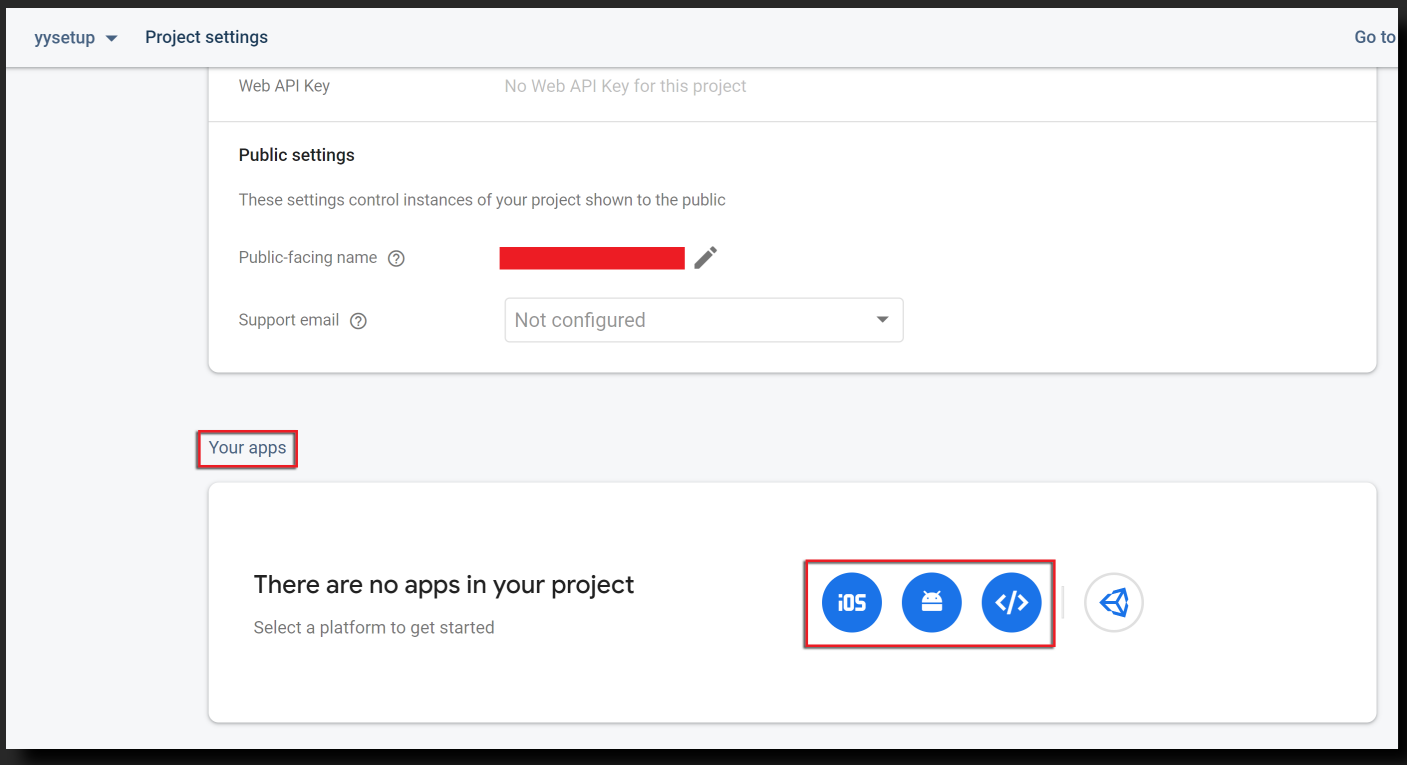
This setup is necessary for all the Firebase modules using iOS and needs to be done once per project, and basically involves importing the **GoogleServices-Info.plist** file into your project.

IMPORTANT Please refer to **this Helpdesk article** for instructions on setting up an iOS project.

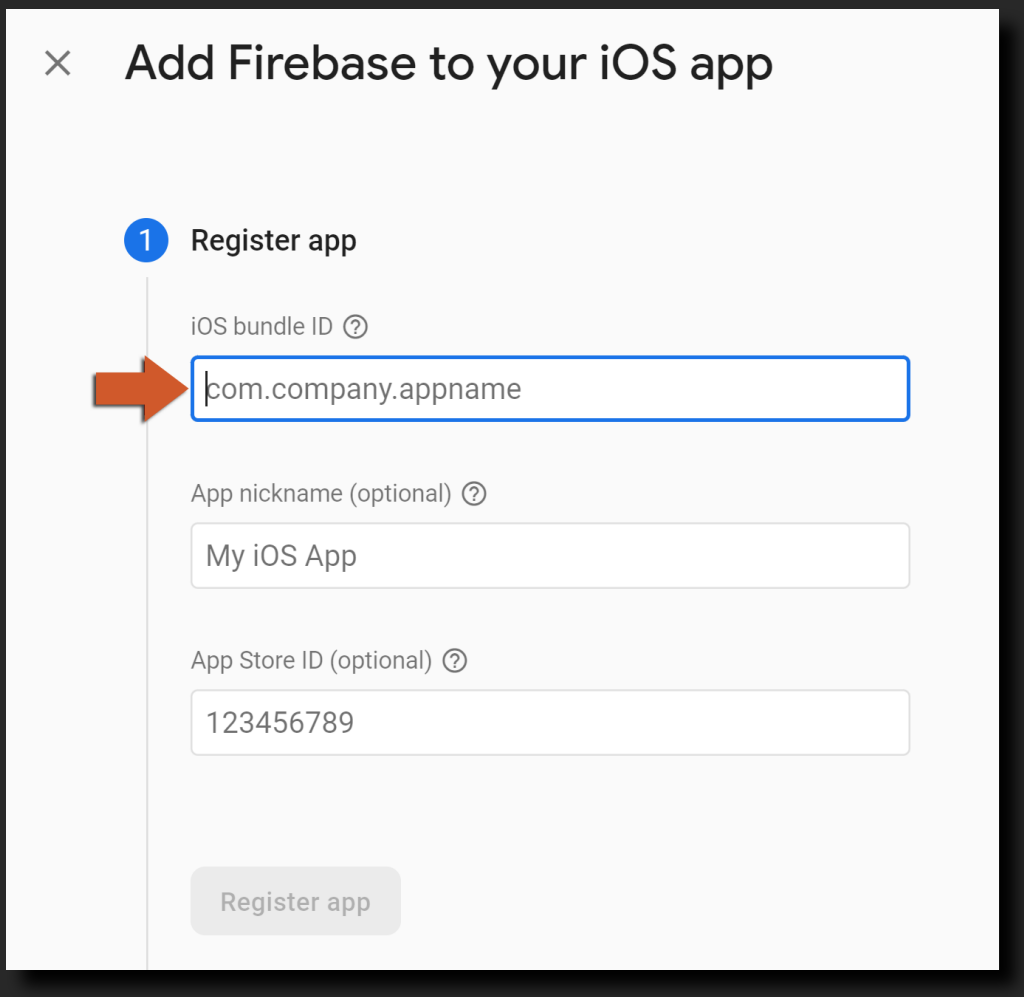
1. Click the **Settings** icon (next to Project Overview) and select **Project settings:**



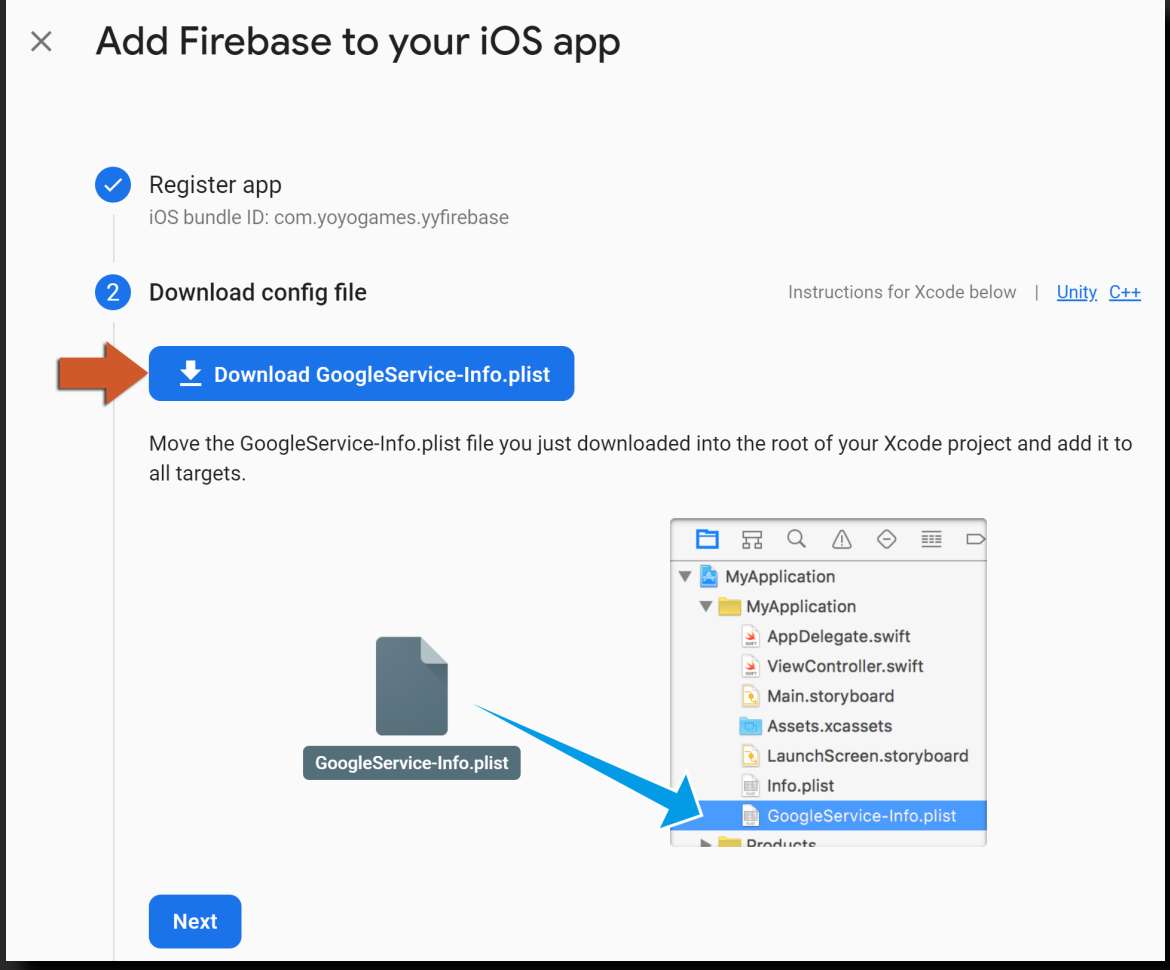
2. Now go to the **Your apps** section and click on the **iOS** button:



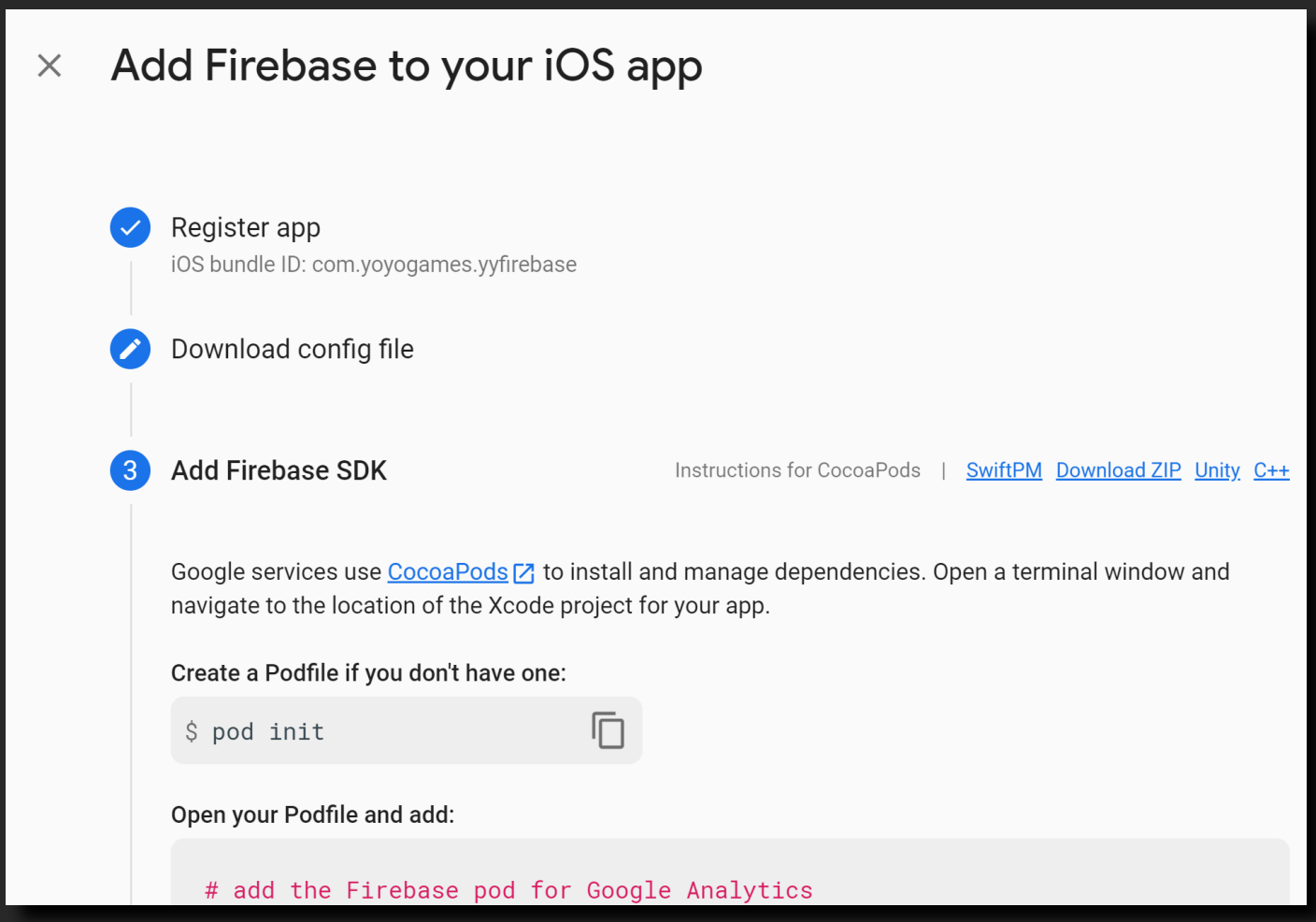
3. Fill the form with your **iOS Bundle ID**, **App nickname** and **AppStore ID** (last two are optional).



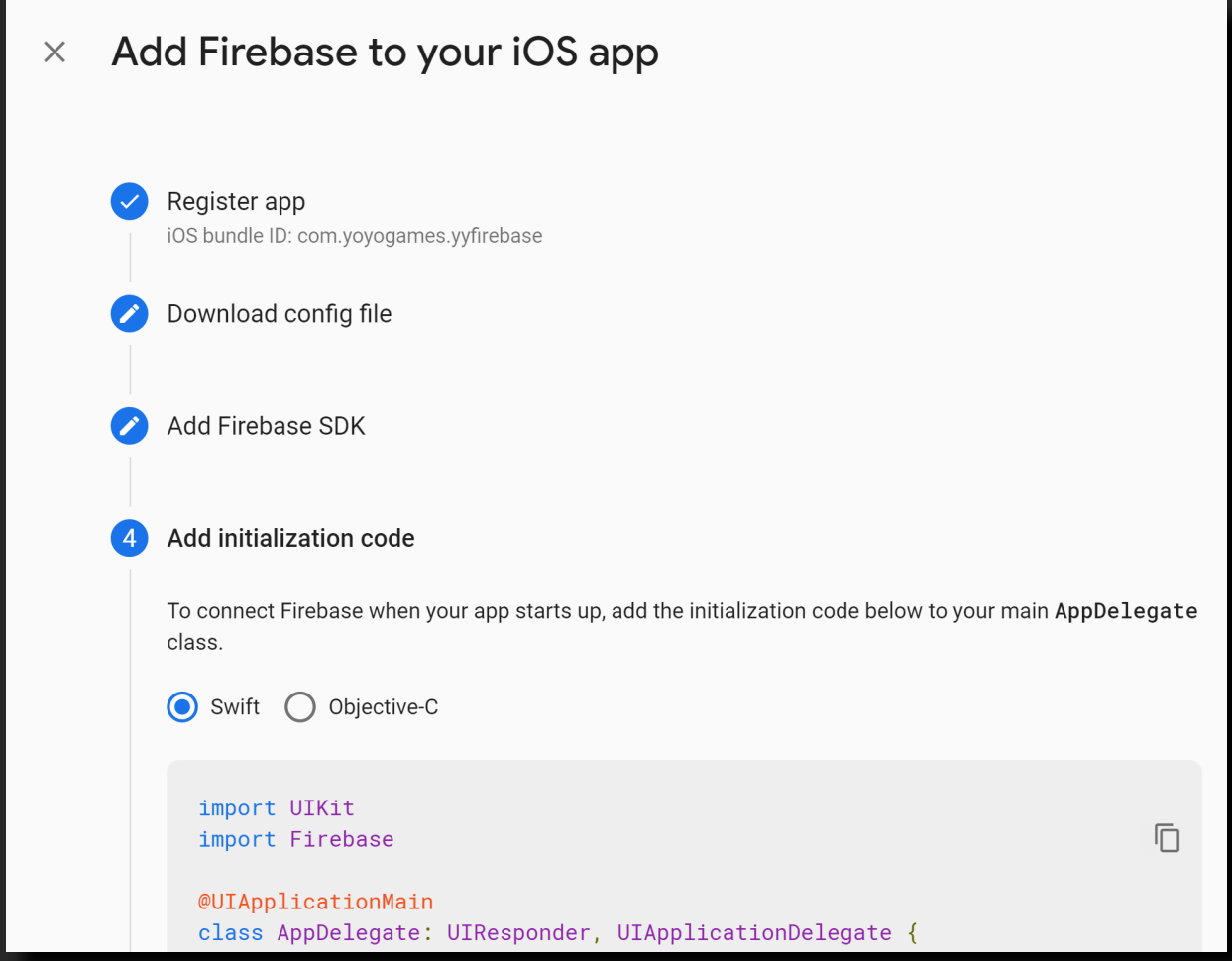
4. Click on **Download GoogleService-info.plist** (make sure to save this file as we will need it in subsequent steps).



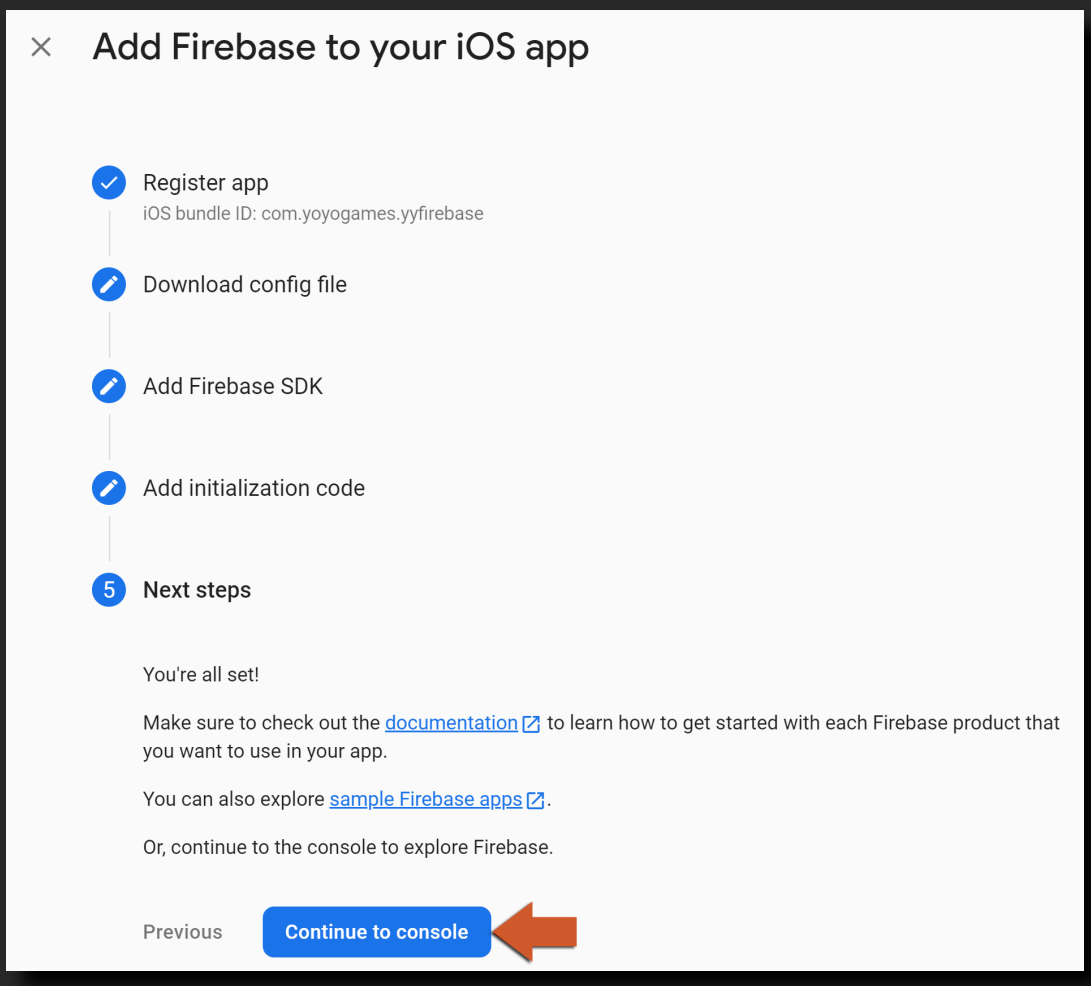
5. Ignore this screen, as this is already done in the extension.



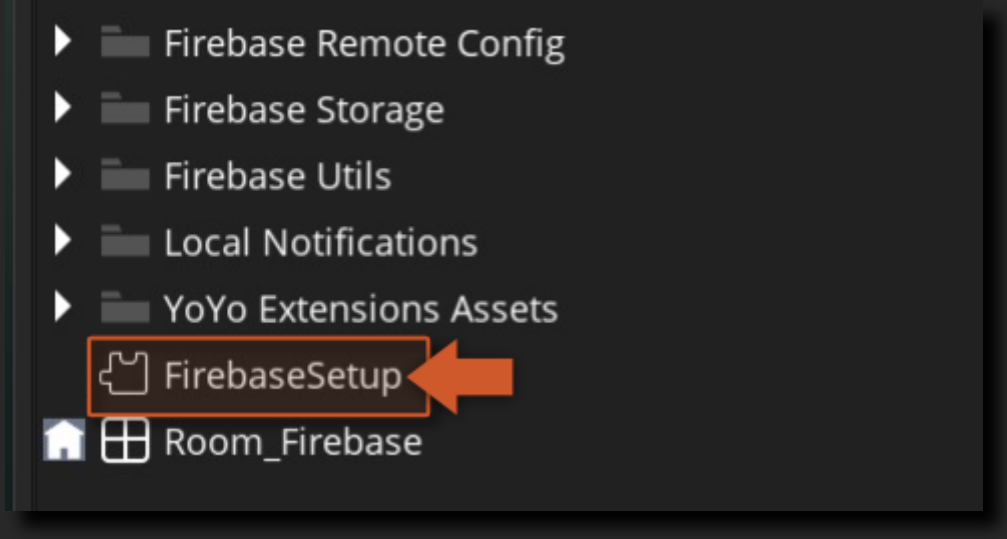
6. Ignore this screen as well, as this is also done in the extension.



7. Click on the **Continue to console** button:



8. Now go into GameMaker, double click the extension **FirebaseSetup** asset.



9. In the extension panel just fill in the paths for the correct files (Android and/or iOS).

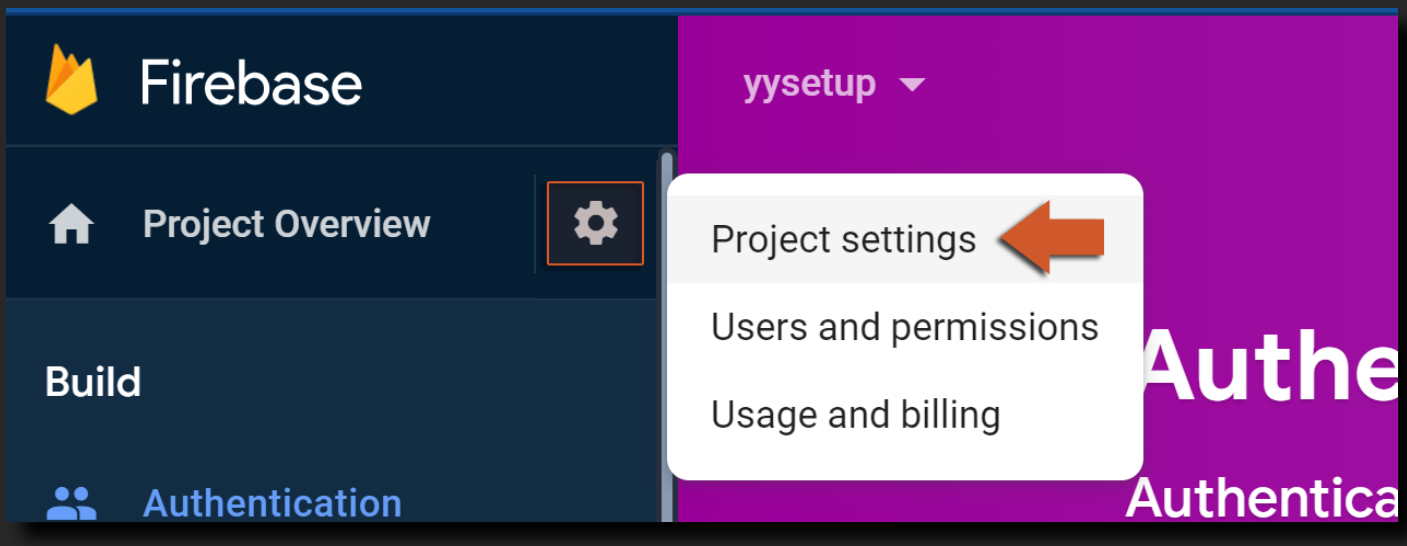
10. Make sure to set up **CocoaPods** for your project *unless* you are only using the REST API in an extension (if one is provided -- not all extensions provide a REST API) or the Firebase Cloud Functions extension (which only uses a REST API).

11. You have now finished the main setup for all Firebase iOS modules!

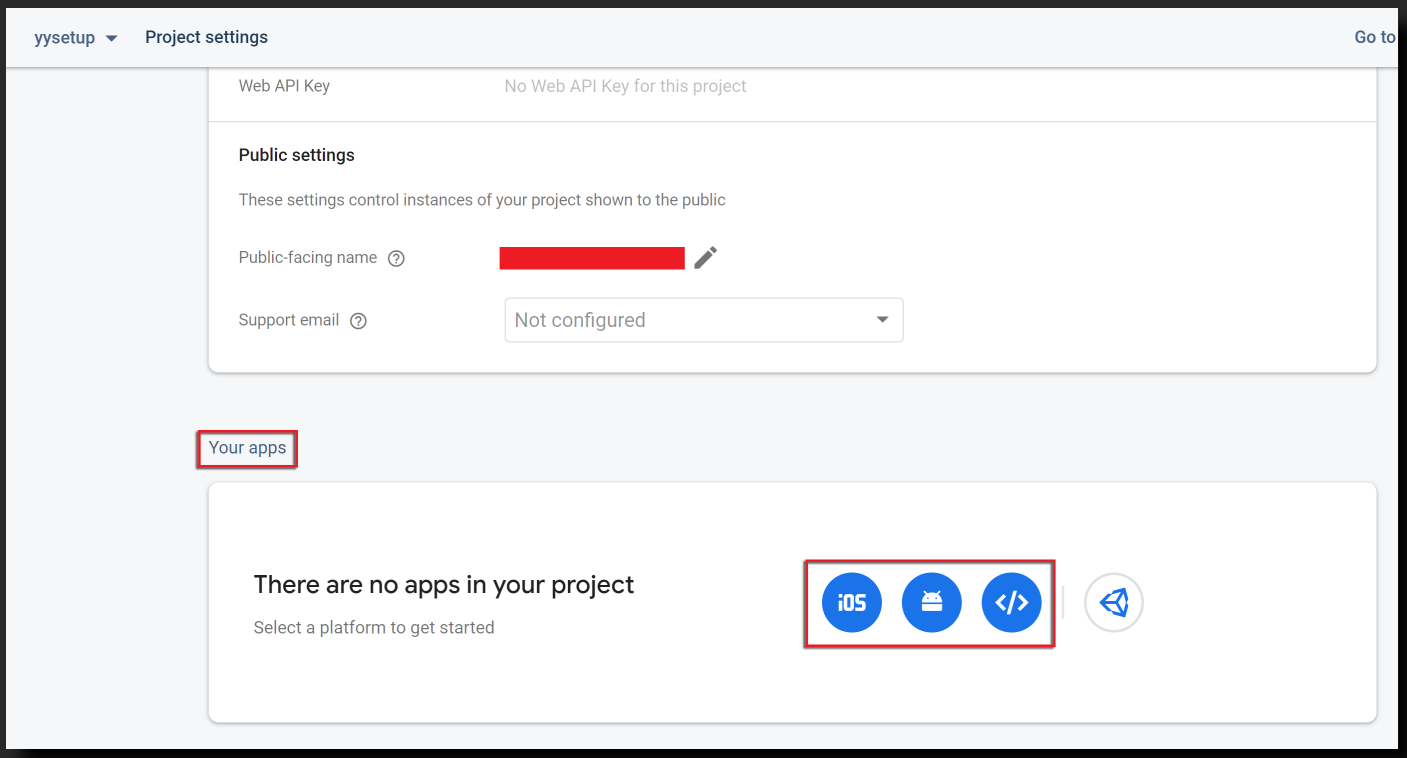
Web Setup

This setup is necessary for all the Firebase modules using Web export and needs to be done once per project, and basically involves adding Firebase libraries and your Firebase values to the **index.html** file in your project.

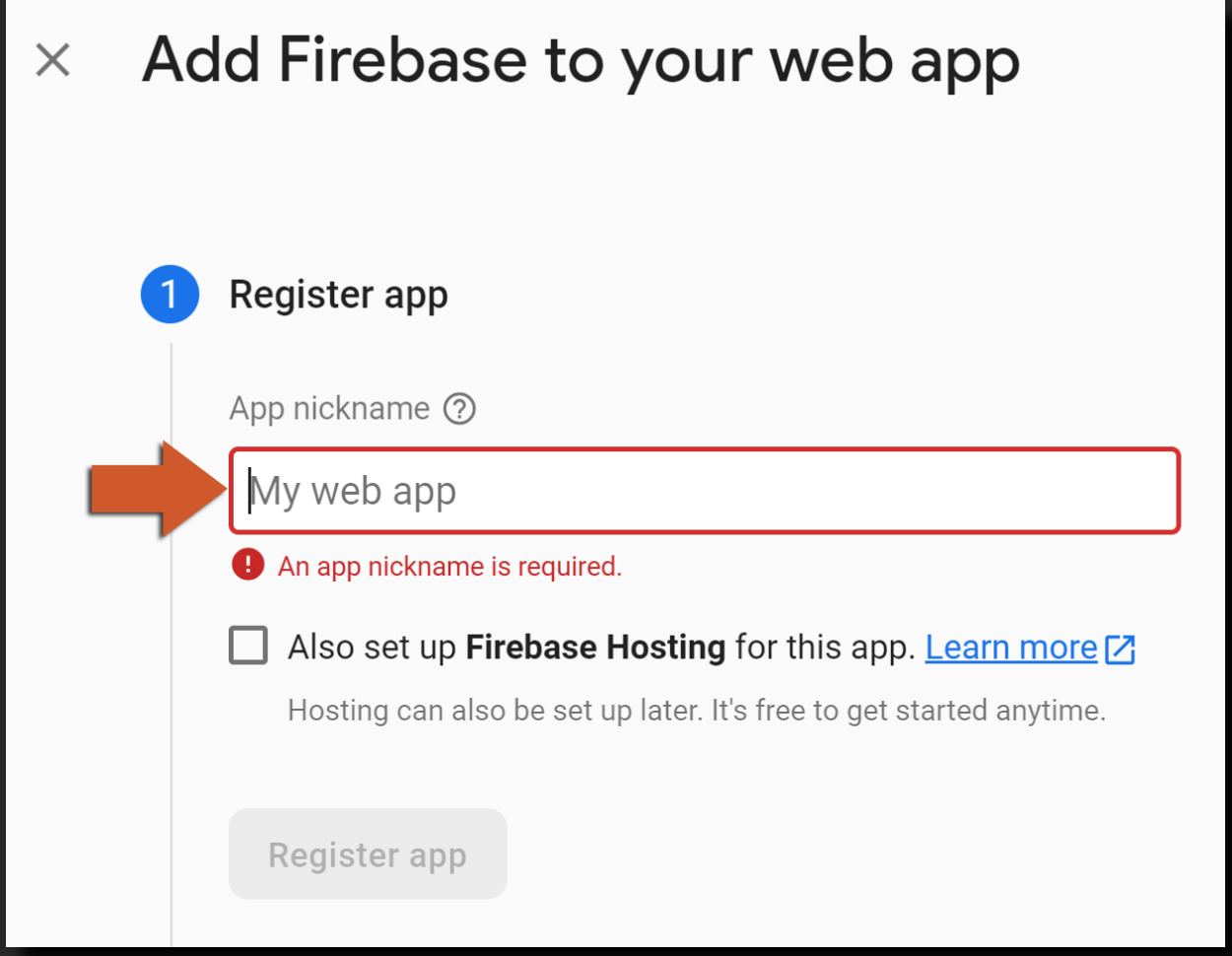
1. Click the **Settings** icon (next to Project Overview) and then **Project settings:**



2. Now go to the **Your apps** section and click on the **Web** (</>) button:



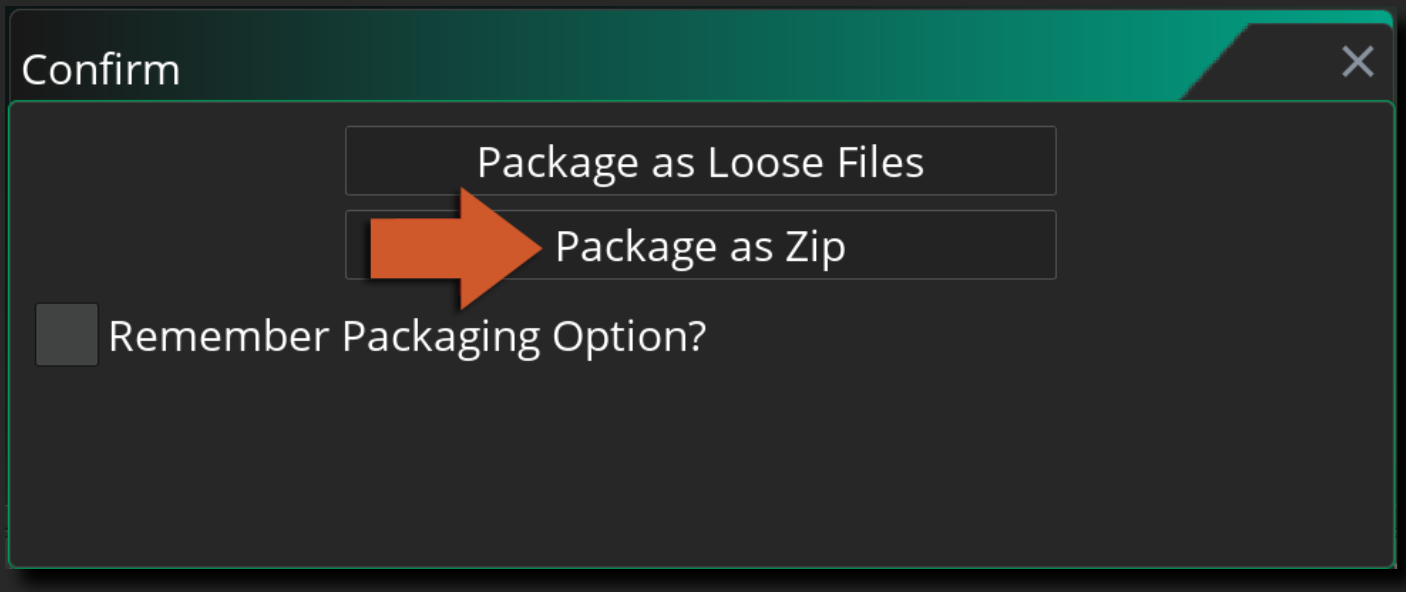
3. Enter your App nickname (required):



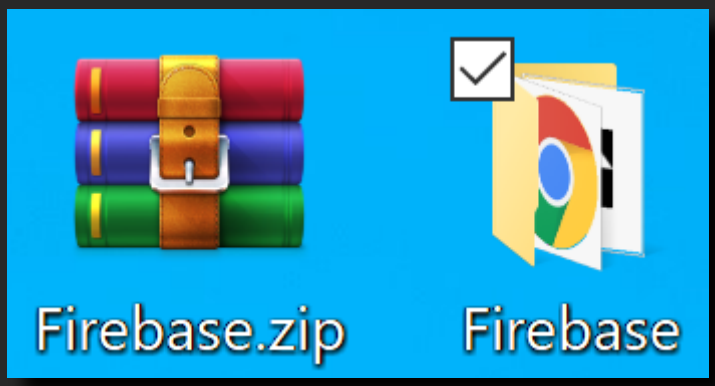
4. On this screen, just copy the **firebaseConfig** struct:

5. Now go back into GameMaker Studio 2 and **build** your project.

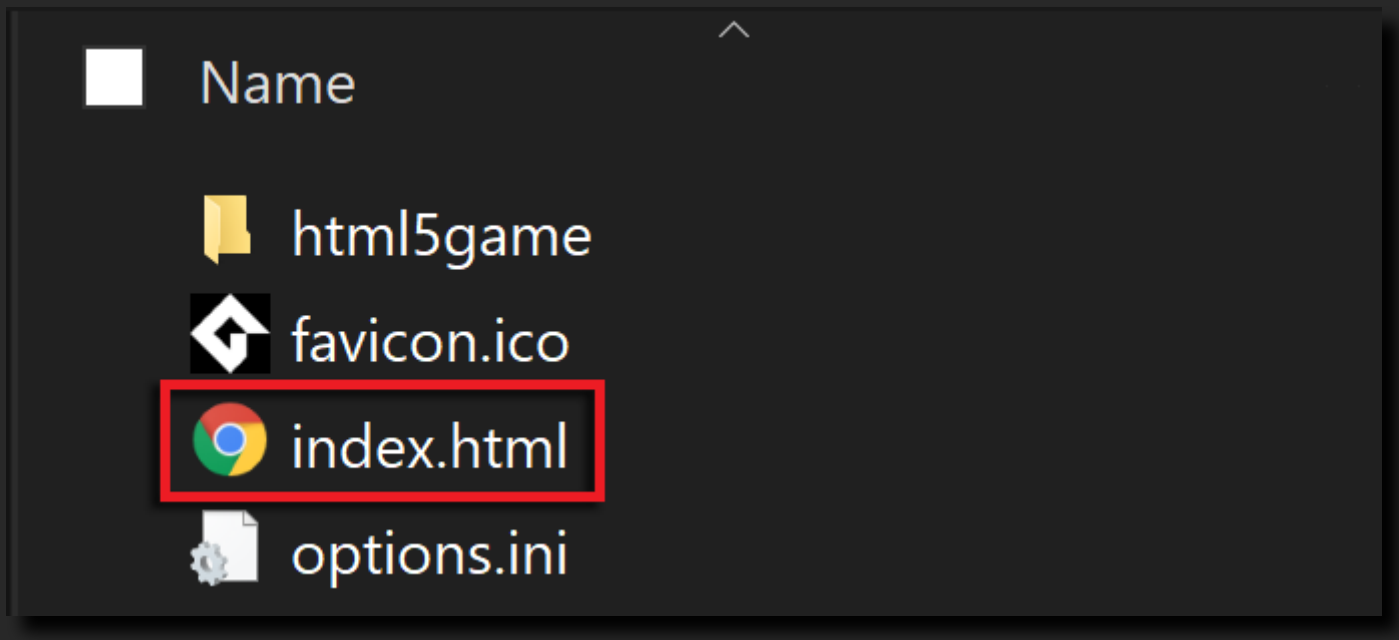
6. Choose the **Package As Zip** option:



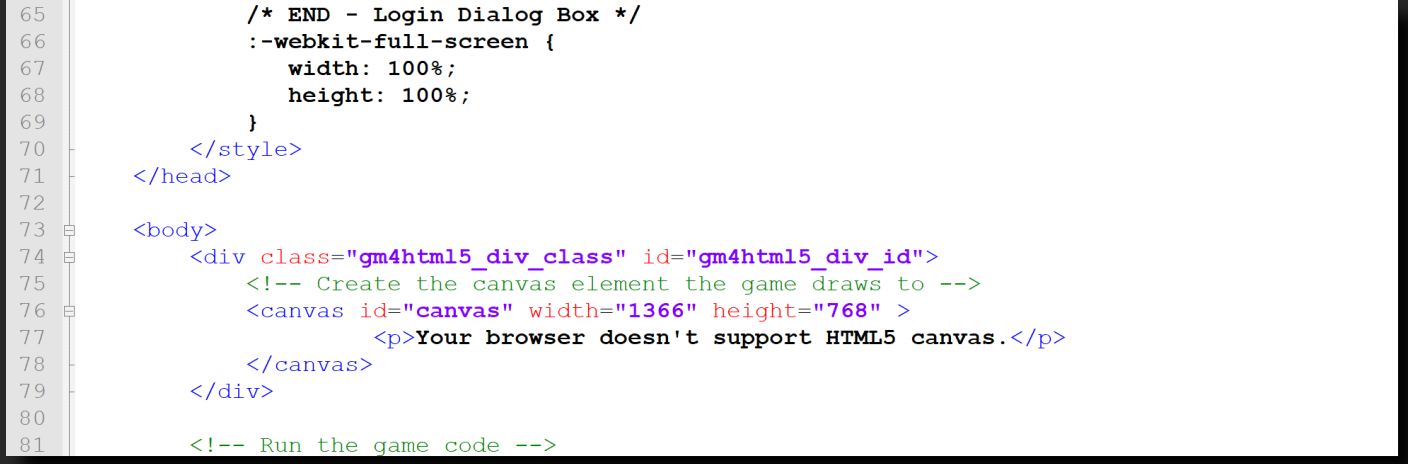
7. Locate the created package and **extract it** into a folder.



8. Open the extracted folder and look for an **index.html** file.



9. Open the **index.html** file in Notepad++ or Visual Studio Code (or any other text editor you prefer).



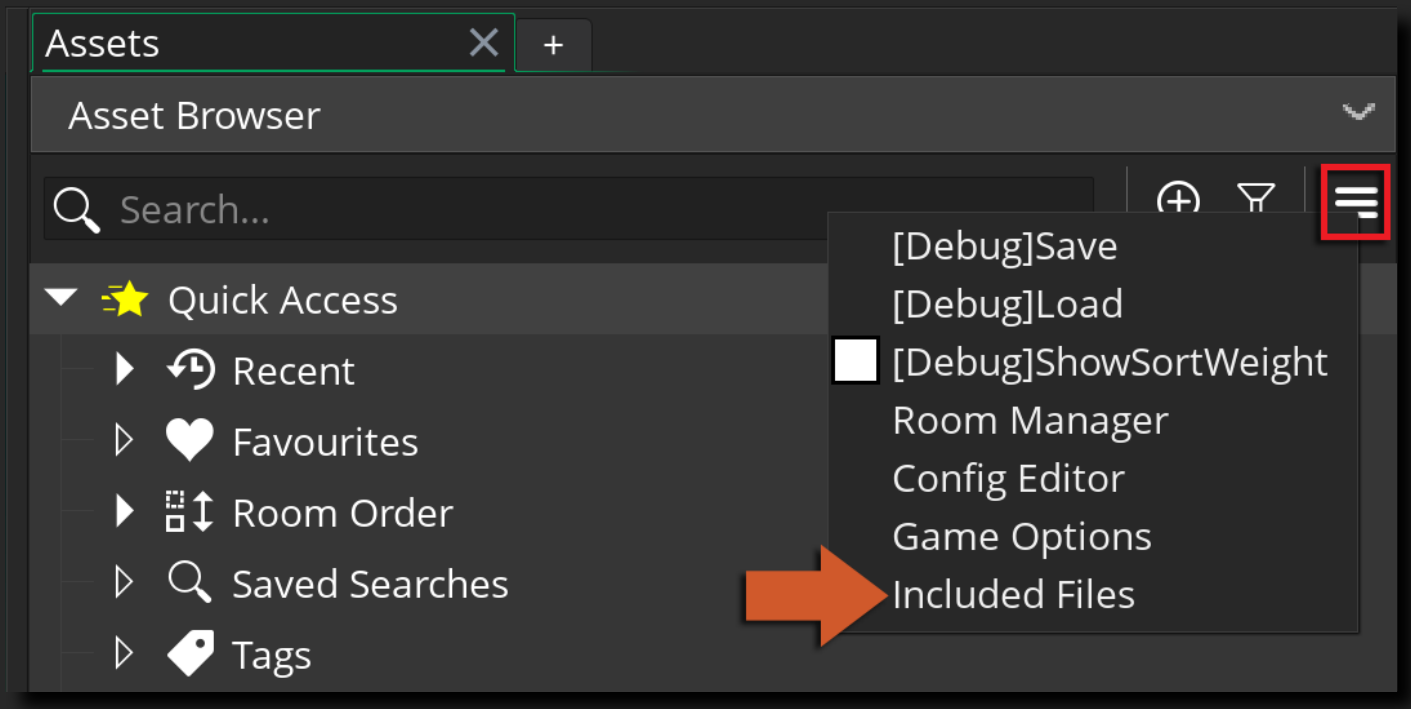
10. Now we need to add the following code between the `</head>` and `<body>` tags (line 72 in the **html.index** image above):

```
<script src="https://www.gstatic.com/firebasejs/8.9.1/firebase-
app.js"></script>
<script src="https://www.gstatic.com/firebasejs/8.9.1/firebase-
analytics.js"></script>
<script src="https://www.gstatic.com/firebasejs/8.9.1/firebase-
auth.js"></script>
<script src="https://www.gstatic.com/firebasejs/8.9.1/firebase-
database.js"></script>
<script src="https://www.gstatic.com/firebasejs/8.9.1/firebase-
firestore.js"></script>
<script src="https://www.gstatic.com/firebasejs/8.9.1/firebase-remote-
config.js"></script>

<script>
  const firebaseConfig = {
    apiKey: "",
    authDomain: "",
    databaseURL: "",
    projectId: "",
    storageBucket: "",
    messagingSenderId: "",
    appId: "",
    measurementId: ""
  };
  firebase.initializeApp(firebaseConfig);

</script>
```

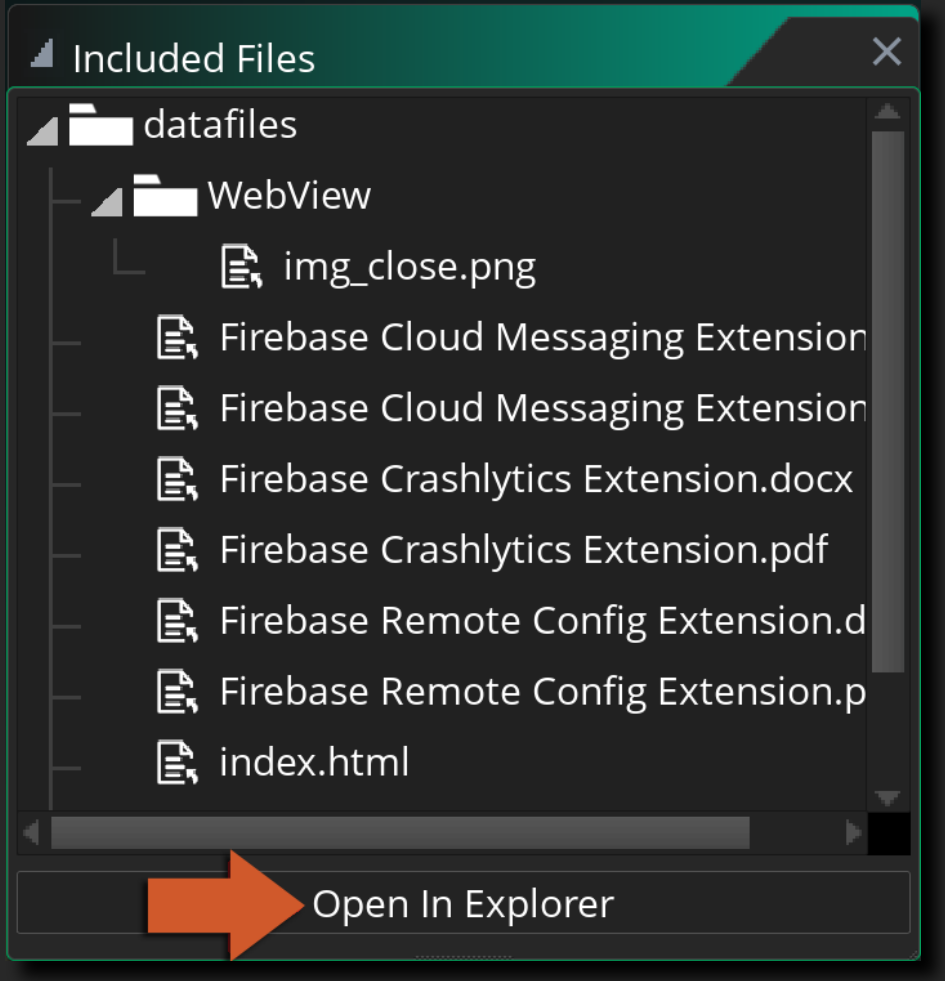
11. Replace the **const firebaseConfig** part with the one copied in step 4:

```
                height: 100%;
            }
        </style>
    </head>

<!-- Firebase -->
<script src="https://www.gstatic.com/firebasejs/8.9.1/firebase-app.js"></script>
<script src="https://www.gstatic.com/firebasejs/8.9.1/firebase-analytics.js"></script>
<script src="https://www.gstatic.com/firebasejs/8.9.1/firebase-auth.js"></script>
<script src="https://www.gstatic.com/firebasejs/8.9.1/firebase-database.js"></script>
<script src="https://www.gstatic.com/firebasejs/8.9.1/firebase-firestore.js"></script>
<script src="https://www.gstatic.com/firebasejs/8.9.1/firebase-remote-config.js"></script>


<script>

  // Your web app's Firebase configuration
  // For Firebase JS SDK v7.20.0 and later, measurementId is optional
  const firebaseConfig = {
    apiKey: "[redacted]",
    authDomain: "[redacted]",
    databaseURL: "[redacted]",
    projectId: "[redacted]",
    storageBucket: "[redacted]",
    messagingSenderId: "[redacted]",
    appId: "[redacted]",
    measurementId: "[redacted]"
  };

  firebase.initializeApp(firebaseConfig);

</script>



    <body>
        <div class="gm4html5_div_class" id="gm4html5_div_id">
```

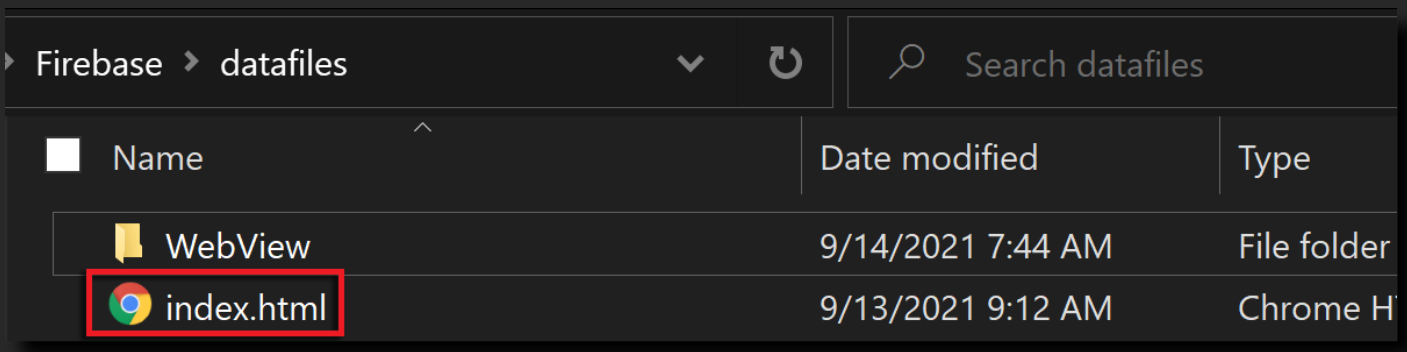
12. Go back into GameMaker and open your **Included Files** folder.



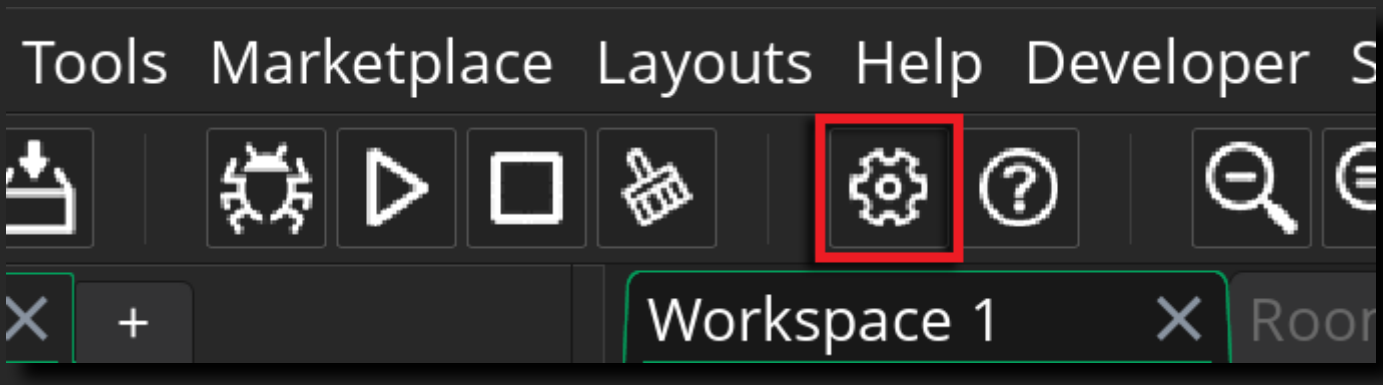
13. Press the **Open in Explorer** button:



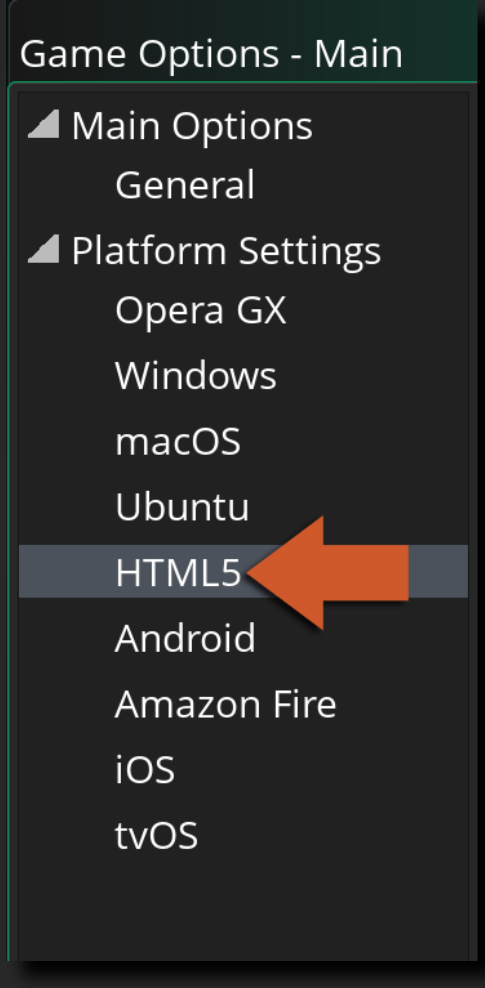
14. Place your **index.html** file inside the folder that opens (/datafiles).



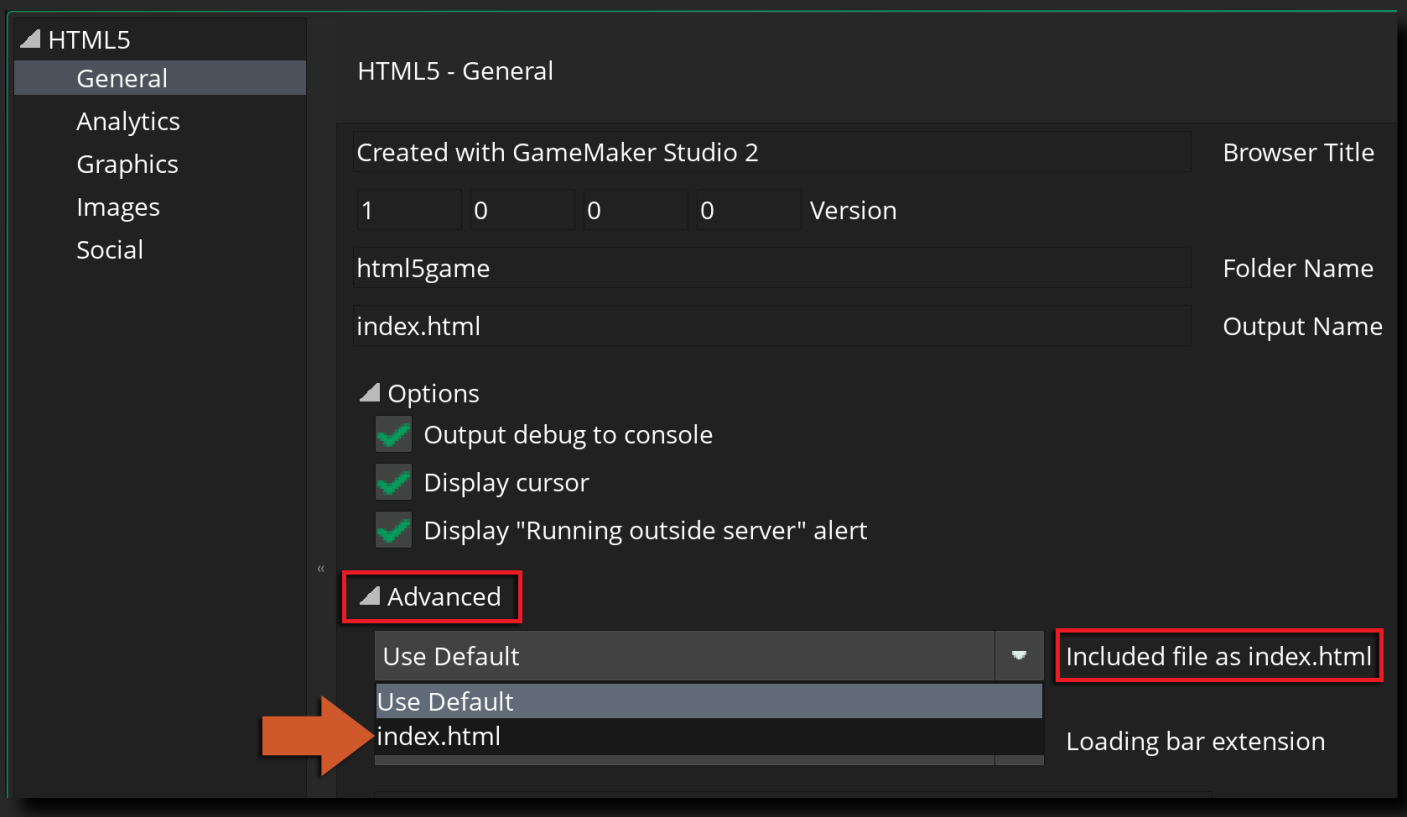
15. Back in GameMaker, click on the **Game Options button.**



16. Go into the **HTML5** platform settings



17. In the **Advanced** section go to the "Include file as index.html" dropdown and select the **index.html** option (this is the file we have just added to the included files).



18. Press **Apply** and the main setup for all Firebase Web modules is finished!

FirebaseAnalytics_LogEvent

This function logs an app event. The event can have up to 25 parameters. Note that events with the same name must have the same parameters. Up to 500 event names are supported. Using predefined FirebaseAnalytics.Event and/or FirebaseAnalytics.Param is recommended for optimal reporting.

Syntax:

```
FirebaseAnalytics_LogEvent(event, params)
```

Argument	Tyoe	Description
event	string	The event data (available events reference)
params	string	The data as a JSON formatted string (available params reference)

Returns:

```
N/A
```

Example:

```
var _data = { screen_name : room_get_name(room) };

FirebaseAnalytics_LogEvent("screen_view", json_stringify(_data))
```

In the code sample above we will submit a log to the event `"screen_view"` (see event details), so first create a data struct with a parameter named `"screen_name"` (this is one of the allowed parameters) and convert it to a string using the json_stringify function.

FirebaseAnalytics_ResetAnalyticsData

This function clears all analytics data for the app from the current device and resets the app instance ID.

> **NOTE** This will not clear analytics previously collected by the app that have already been pushed to the server.

Syntax:

```
FirebaseAnalytics_ResetAnalyticsData()
```

Returns:

```
N/A
```

Example:

```
if (global.stopDataCollection)
{
    FirebaseAnalytics_ResetAnalyticsData();
    FirebaseAnalytics_SetAnalyticsCollectionEnabled(false);
}
```

The code above will check if a global variable is set to true (`global.stopDataCollection`) and if so we will clear the local collected data for this extension and disable the analytics data collection (using the function FirebaseAnalytics_SetAnalyticsCollectionEnabled).

FirebaseAnalytics_SetAnalyticsCollectionEnabled

This function is used to enable or disable analytics collection for the app on the current device. This setting is persisted across app sessions. By default it is enabled.

Syntax:

```
FirebaseAnalytics_SetAnalyticsCollectionEnabled(enable)
```

Argument	Type	Description
enable	boolean	Whether or not analytics collection should be enabled.

Returns:

```
N/A
```

Example:

```
if (global.stopDataCollection)
{
    FirebaseAnalytics_ResetAnalyticsData();
    FirebaseAnalytics_SetAnalyticsCollectionEnabled(false);
}
```

The code above will check if a global variable is set to true and if so we will clear the local collected data (using the function FirebaseAnalytics_ResetAnalyticsData) for this extension and disable the analytics data collection.

FirebaseAnalytics_SetDefaultEventParameters

This function sets "default" parameters that will be provided with every event logged from the SDK, including automatic ones. The values passed in the parameters bundle will be added to the map of default event parameters. These parameters persist across app runs. They are of lower precedence than event parameters, so if an event parameter and a parameter set using this function have the same name, the value of the event parameter will be used. The same limitations on event parameters apply to default event parameters.

Syntax:

```
FirebaseAnalytics_SetDefaultEventParameters(parameters)
```

Argument	Type	Description
parameters	string	A **JSON formatted** string of a struct to be added to **every** event. These parameters will be added to the default event parameters, replacing any existing parameter with the same name. Valid parameter values are **string** and **double**. Setting a key's value to `pointer_null` will clear that parameter.

Returns:

```
N/A
```

Example:

```
var parameters = {
    playerName: "Hero",
    bossName: pointer_null,
};

FirebaseAnalytics_SetDefaultEventParameters(json_stringify(parameters))
```

With the code above we are setting the default parameters to be used on all further calls to FirebaseAnalytics_LogEvent. First we create a struct (`parameters`) with a `"playerName"` and

`"bossName"` keys, the boss name is set to `pointer_null` meaning it will be removed from the set of default event parameters.

FirebaseAnalytics_SetSessionTimeoutDuration

This function sets the duration of inactivity (in milliseconds) that terminates the current session. The default value is 1800000 (30 minutes).

Syntax:

```
FirebaseAnalytics_SetSessionTimeoutDuration(milliseconds)
```

Argument	Description
milliseconds	Session timeout duration in milliseconds.

Returns:

```
N/A
```

Example:

```
FirebaseAnalytics_SetSessionTimeoutDuration(3600000) //1 hour
```

The code above sets the timeout duration to 1 hour, meaning that if no events are logged within this timeout the session will be terminated.

FirebaseAnalytics_SetUserId

This function sets the user ID property. This feature must be used in accordance with Google's Privacy Policy.

Syntax:

```
FirebaseAnalytics_SetUserId(userID)
```

Argument	Description
userID	The user ID string to assign to the user of this app on this device, which must be non-empty and no more than 256 characters long. Setting the ID to `pointer_null` removes the user ID.

Returns:

```
N/A
```

Example:

```
FirebaseAnalytics_SetUserId("myUser123");
```

The above code will set the user ID to the string `"myUser123"`.

FirebaseAnalytics_SetUserProperty

This function sets a user property to the given value. Up to 25 user property names are supported. Once set, user property values persist throughout the app life cycle and across sessions.

Syntax:

```
FirebaseAnalytics_SetUserProperty(key,value)
```

Argument	Description
key	The name of the user property to set. Should contain 1 to 24 **alphanumeric** characters or underscores and must start with an **alphabetic** character. IMPORTANT The `"firebase_"`, `"google_"` and `"ga_"` prefixes are reserved and should not be used for user property names.
value	The value of the user property. Values can be up to 36 characters long. Setting the value to `pointer_null` removes the user property.

Returns:

```
N/A
```

Example:

```
FirebaseAnalytics_SetUserProperty("location","UK");
```

The code above will set an user property `"location"` to the value `"UK"`.