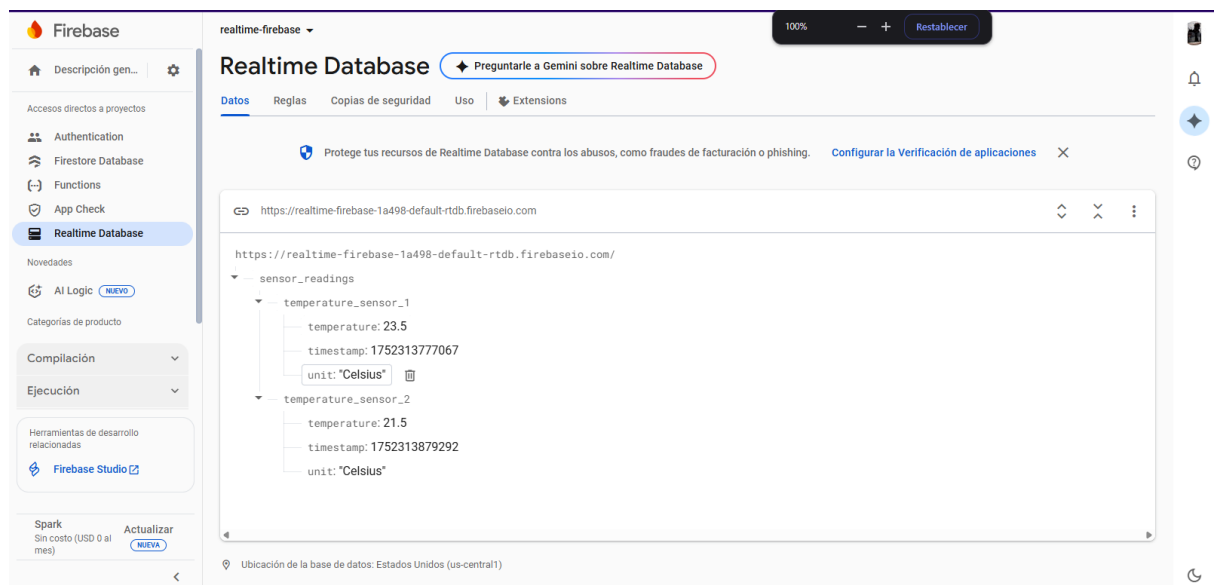
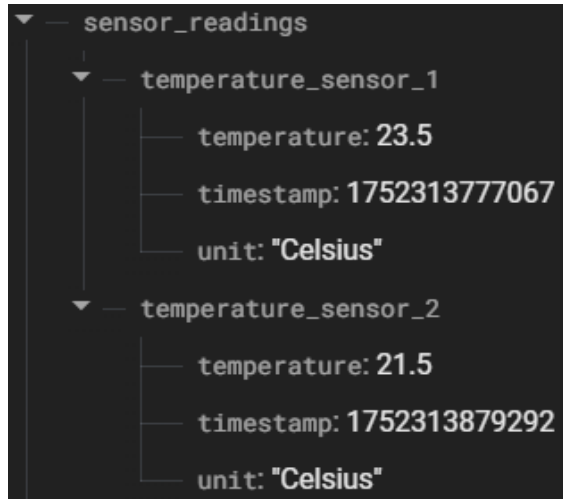


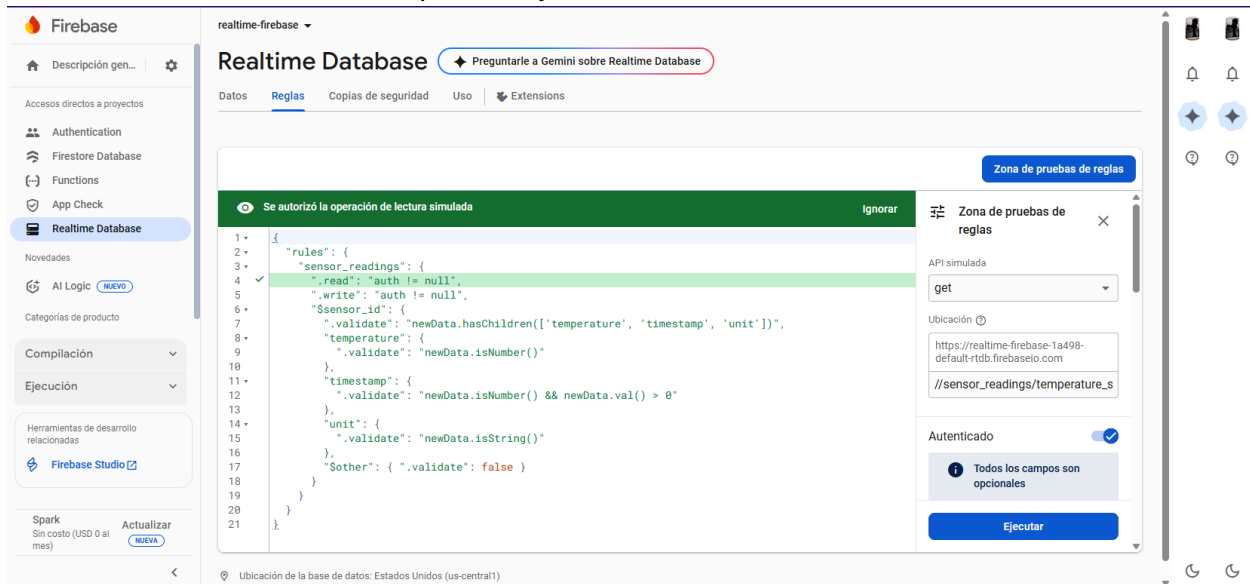
**Asignatura:** Herramientas de construcción de Software  
**Carrera:** Ingeniería de Software  
**Docente:** Harry Carpio S.  
**Fecha:** Julio del 2025  
**Nombre:** Daniel Villarroel, Joseph Naranjo

## Instrucciones

1. Crear una cuenta en [Firebase](#), iniciar un nuevo proyecto.
2. Crear una base de datos en Realtime Database, crearla en modo de prueba.
3. Investigar y escribir dos registros con el siguiente formato:



**Asignatura:** Herramientas de construcción de Software  
**Carrera:** Ingeniería de Software  
**Docente:** Harry Carpio S.  
**Fecha:** Julio del 2025  
**Nombre:** Daniel Villarroel, Joseph Naranjo



## Entregables

1. Código fuente usado para escribir en Realtime Database.

```
function actualizarSensor(idSensor, temperatura) {
  const db = getDatabase();
  const sensorRef = ref(db, `sensor_readings/${idSensor}`);

  set(sensorRef, {
    temperature: temperatura,
    timestamp: Date.now(),
    unit: "Celsius"
  });
}

actualizarSensor("temperature_sensor_1", 24.0);
```

2. Captura completa de la consola web de Firebase, debe verse claramente el nombre de su proyecto.

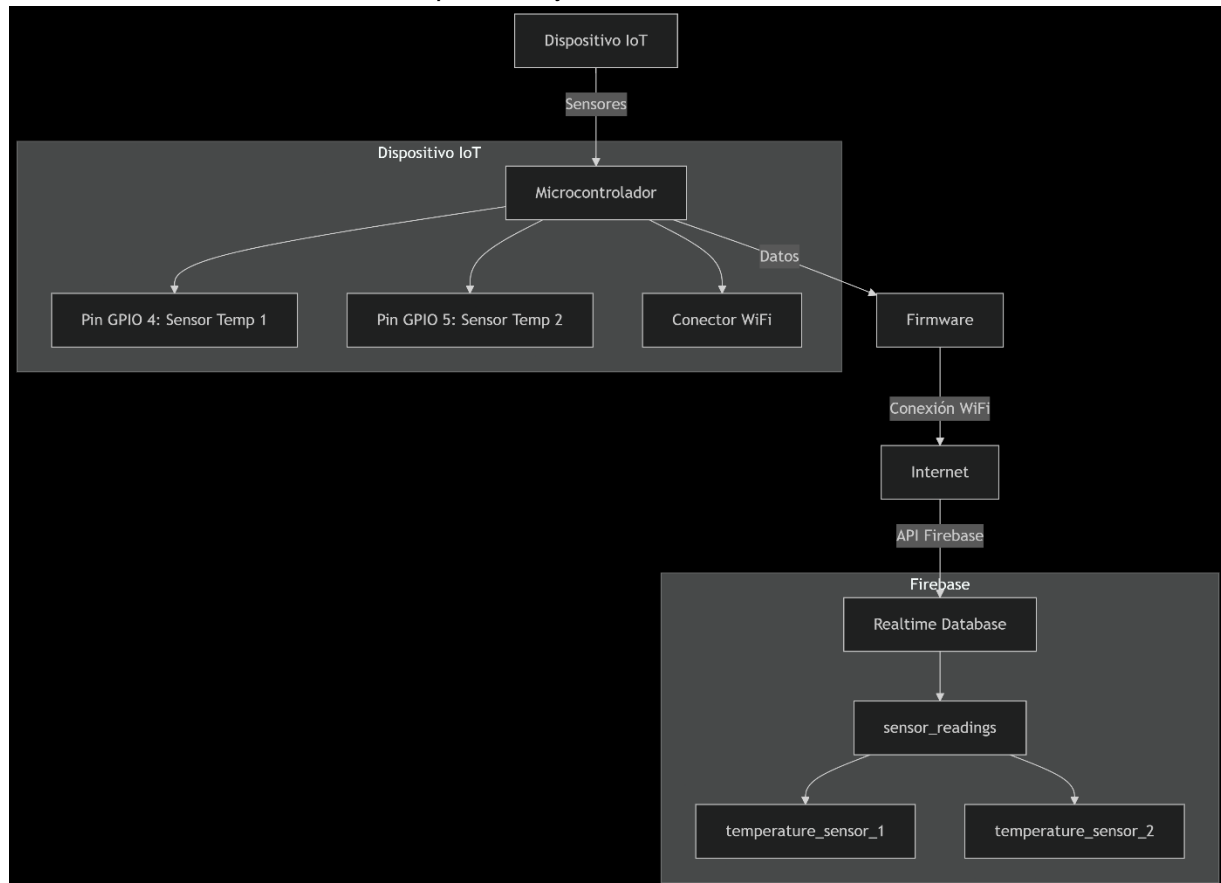
**Asignatura:** Herramientas de construcción de Software  
**Carrera:** Ingeniería de Software  
**Docente:** Harry Carpio S.  
**Fecha:** Julio del 2025  
**Nombre:** Daniel Villarroel, Joseph Naranjo

The screenshot shows the Firebase Realtime Database Rules editor. The left sidebar contains the Firebase logo and navigation links for various services. The main area displays the 'Reglas' (Rules) tab for the 'realtime-firebase' database. A green notification bar at the top indicates that a simulated read operation was authorized. The central pane shows a JSON configuration for the database rules, including a 'sensor\_readings' node with read and write permissions, and validation rules for 'sensor\_id', 'temperature', 'timestamp', and 'unit'. The right sidebar features a 'Zona de pruebas de reglas' (Rules Test Zone) with a simulated API endpoint set to 'get', a location field containing the database URL and path, and an 'Ejecutar' (Execute) button. A status message at the bottom of the right sidebar states 'Todos los campos son opcionales' (All fields are optional).

```
1 {
2   "rules": {
3     "sensor_readings": {
4       ".read": "auth != null",
5       ".write": "auth != null",
6       "sensor_id": {
7         ".validate": "newData.hasChildren(['temperature', 'timestamp', 'unit'])",
8         "temperature": {
9           ".validate": "newData.isNumber()"
10        },
11        "timestamp": {
12          ".validate": "newData.isNumber() && newData.val() > 0"
13        },
14        "unit": {
15          ".validate": "newData.isString()"
16        },
17        "$other": { ".validate": false }
18      }
19    }
20  }
21 }
```

3. Diagrama que muestre dónde estaría ubicado su código si contara con un dispositivo IoT.

**Asignatura:** Herramientas de construcción de Software  
**Carrera:** Ingeniería de Software  
**Docente:** Harry Carpio S.  
**Fecha:** Julio del 2025  
**Nombre:** Daniel Villarroel, Joseph Naranjo



4. Explicación de qué ocurre si se trata de escribir más datos para un mismo sensor.

a) **La escritura se permite** si incluye exactamente los campos:

- temperature (número)
- timestamp (número > 0)
- unit (texto)

b) **Se rechaza** si:

- Falta algún campo
- Algún campo tiene tipo incorrecto
- Se incluyen campos extra no permitidos

c) **Los nuevos datos reemplazan a los anteriores** del mismo sensor.

d) **Solo usuarios autenticados** (`auth != null`) pueden leer o escribir.

**Asignatura:** Herramientas de construcción de Software

**Carrera:** Ingeniería de Software

**Docente:** Harry Carpio S.

**Fecha:** Julio del 2025

**Nombre:** Daniel Villarroel, Joseph Naranjo