

INVESTIGACIÓN

MATERIA	Sistemas Avanzados de Bases de Datos	NRC	9768	
CARRERA	Ingeniería de Software	DOCENTE:		Alexis Estevez
PERIODO ACADÉMICO	Mayo 2023 – septiembre 2023	FECHA		18-08-2023
TEMA	IMPLEMENTACIÓN DE LENGUAJE PYTHON EN EL MANEJO DE OBJETOS			
ESTUDIANTE(S)	Stalin Bladimir Rivera Vega			

A. OBJETIVOS

- ❖ Integrar Python con Firebase para el manejo eficiente de objetos en la base de datos.
- ❖ Optimizar el código Python para asegurar un rendimiento eficiente en la manipulación de objetos y consultas a la base de datos en Firebase.

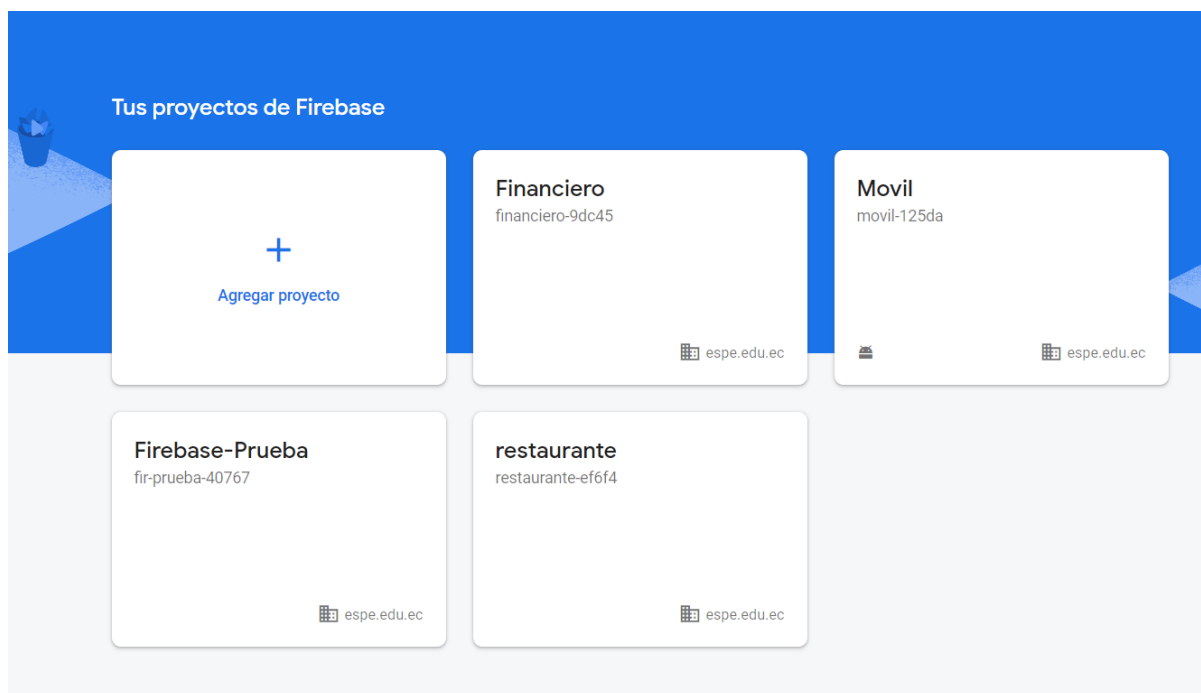
B. INTRODUCCIÓN

En un entorno digital cada vez más interconectado, la implementación efectiva de lenguajes de programación se convierte en un factor esencial para el éxito de las aplicaciones. En este contexto, la elección de Python como lenguaje de desarrollo y su integración con la base de datos en Firebase adquieren una importancia crucial. Python, conocido por su simplicidad y versatilidad, ofrece un terreno fértil para la creación de aplicaciones poderosas y eficientes.

C. DESARROLLO

1. Primer paso: Primero tener una cuenta en firebase

INVESTIGACIÓN



2. Crear un proyecto con el nombre que sea necesario para el proyecto.

× Crear un proyecto(paso 1 de 3)

Comencemos con el nombre de tu proyecto[?]

Nombre del proyecto

Financiero

 financiero-4d472

 Seleccionar recurso superior

Continuar

INVESTIGACIÓN

3. Desactivamos la opción de google Analytics

× Crear un proyecto(paso 2 de 2)

Google Analytics es una solución de analítica ilimitada y gratuita que permite usar la segmentación, los informes y otras funciones en Firebase Crashlytics, Cloud Messaging, In-App Messaging, Remote Config, A/B Testing y Cloud Functions.

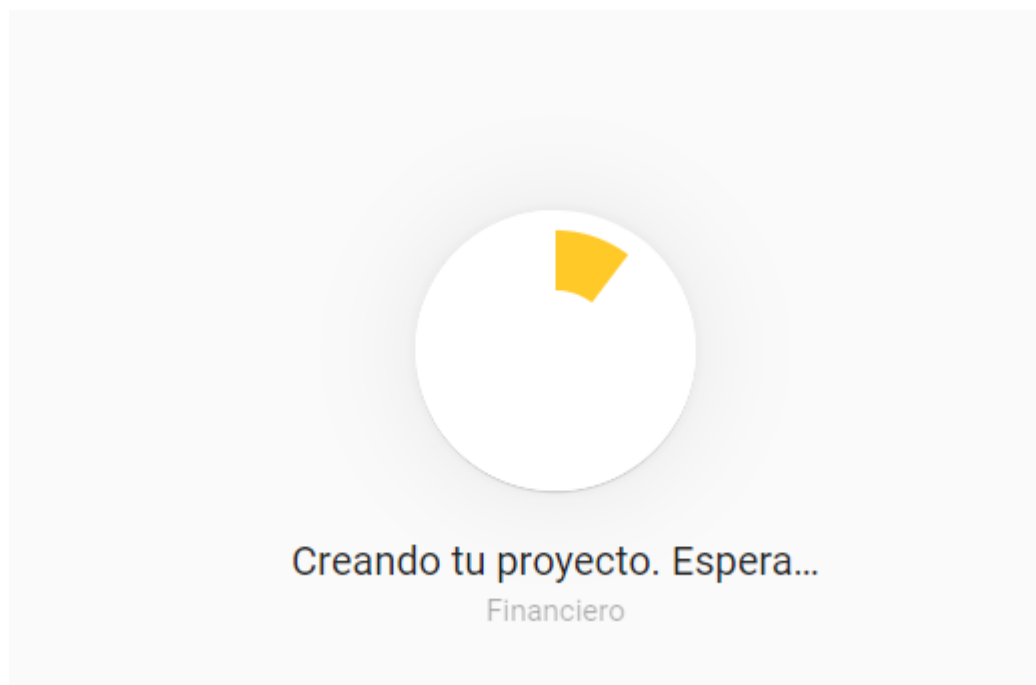
Google Analytics habilita las siguientes funciones:

× Pruebas A/B [?]	× Usuarios que no experimentan fallas [?]
× Segmentación de usuarios y orientación a ellos en los productos de Firebase [?]	× Activadores de Cloud Functions basados en eventos [?]
	× Informes ilimitados y gratuitos [?]

☐ Habilitar Google Analytics para este proyecto
Recomendado

[Anterior](#)[Crear proyecto](#)

4. Creamos el proyecto dentro de firebase.

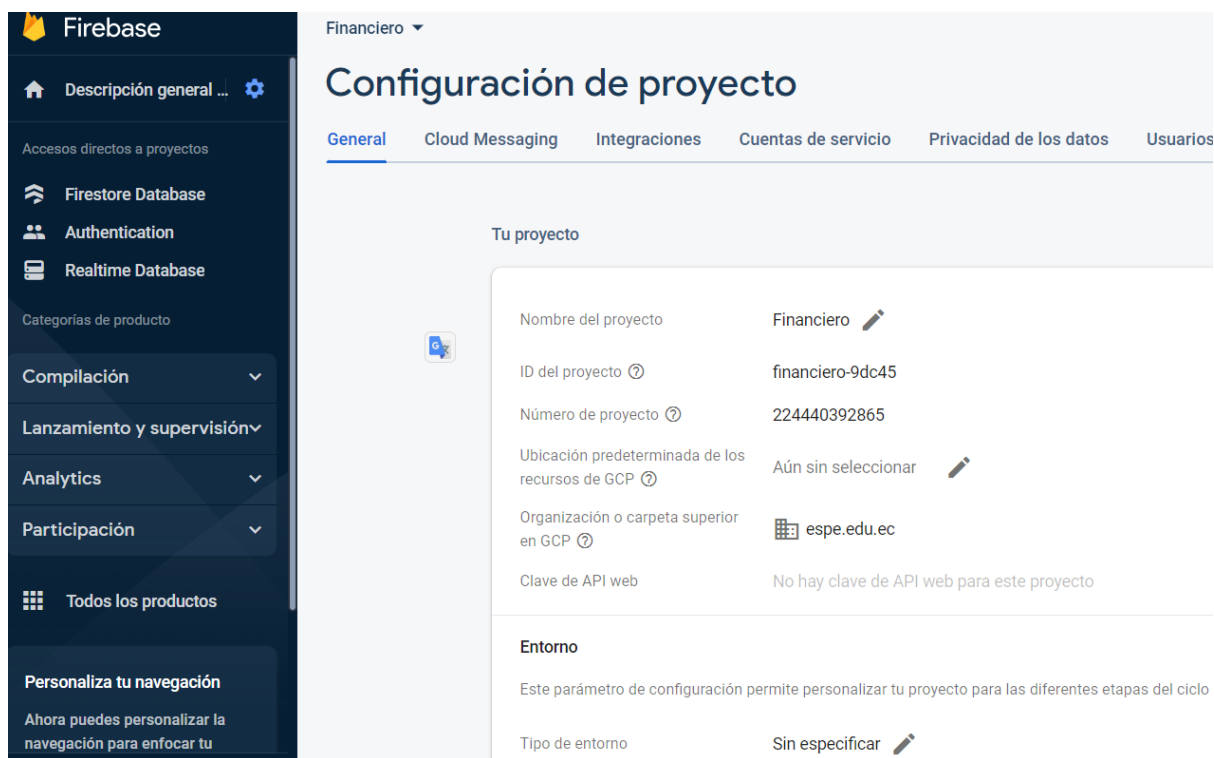


5. Después vamos al apartado de realtime database para obtener el enlace.

INVESTIGACIÓN

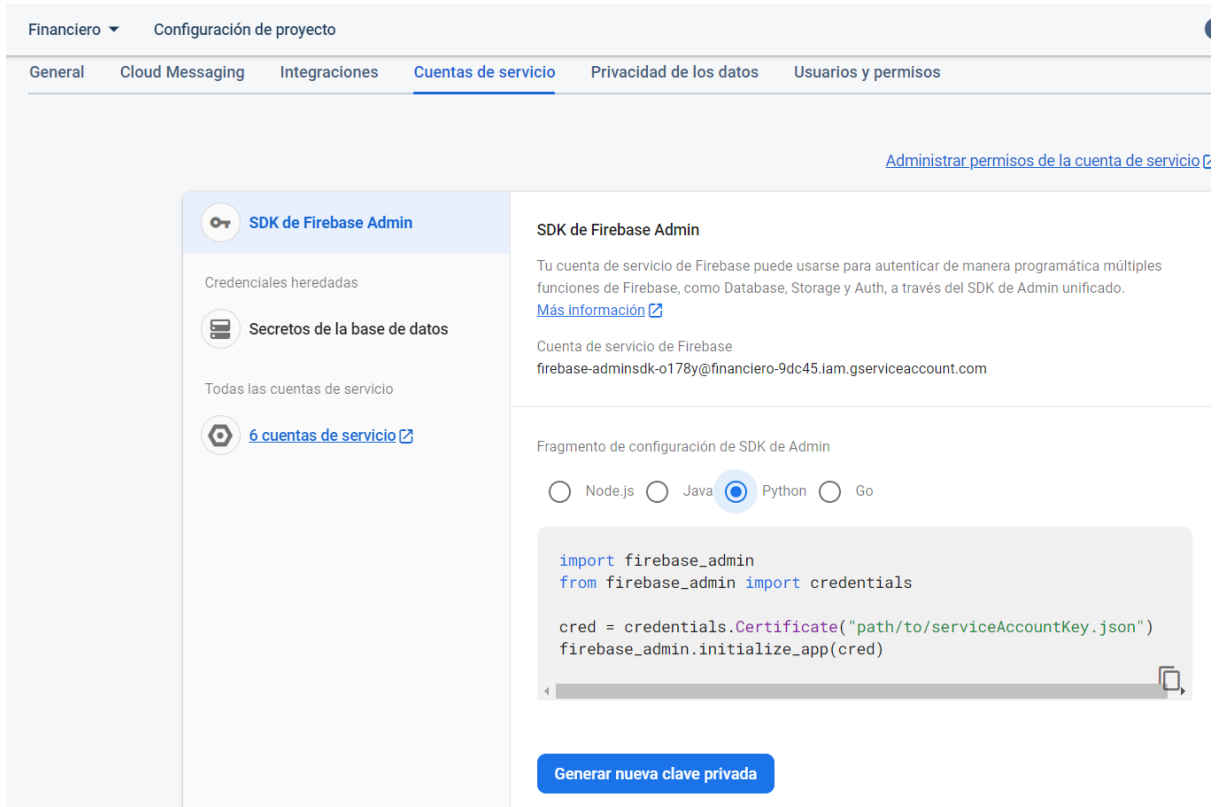


6. A continuación presionamos la opción de descripción general y presionamos la opción de configuración de proyecto, después de eso vamos al apartado de cuenta de servicio.



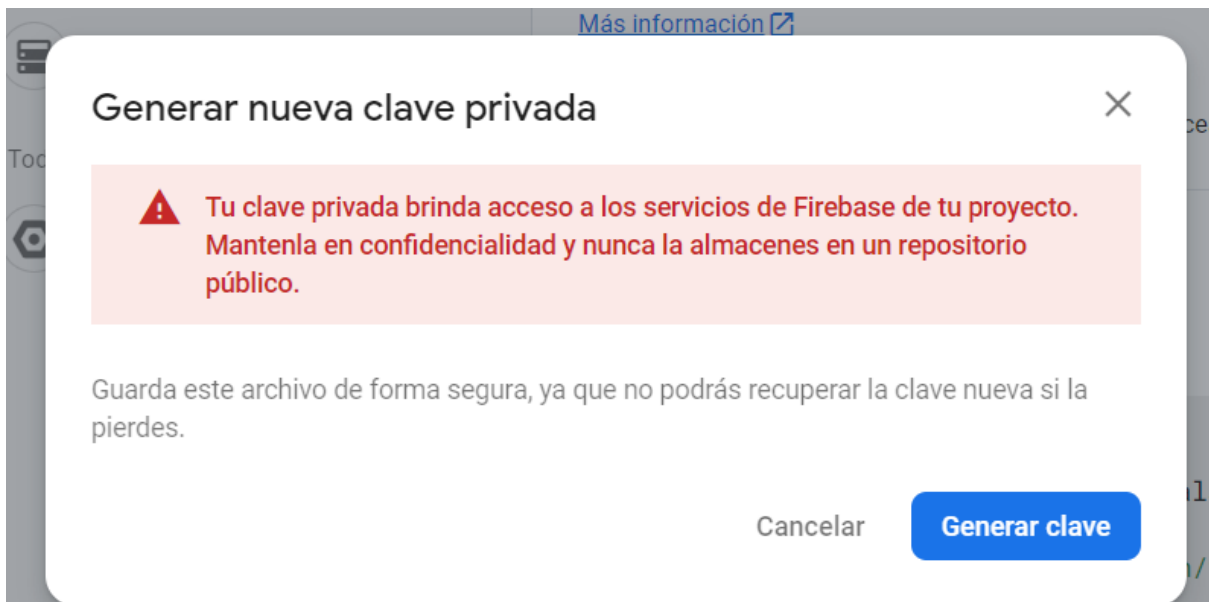
7. Después de estar en cuenta de servicio presionamos en generar nueva clave privada.

INVESTIGACIÓN



The screenshot shows the 'Configuración de proyecto' (Project Configuration) page for a project named 'Financiero'. The 'Cuentas de servicio' (Service Accounts) tab is selected. On the left sidebar, under 'SDK de Firebase Admin', there are links for 'Credenciales heredadas', 'Secretos de la base de datos', and '6 cuentas de servicio'. The main content area is titled 'SDK de Firebase Admin' and explains that the service account can be used for programmatic authentication. It shows the service account email 'firebase-adminsdk-o178y@financiero-9dc45.iam.gserviceaccount.com'. Below this, there's a section for 'Fragmento de configuración de SDK de Admin' with radio buttons for Node.js, Java, Python (selected), and Go. A code block shows the Python initialization code. At the bottom, there is a blue button labeled 'Generar nueva clave privada'.

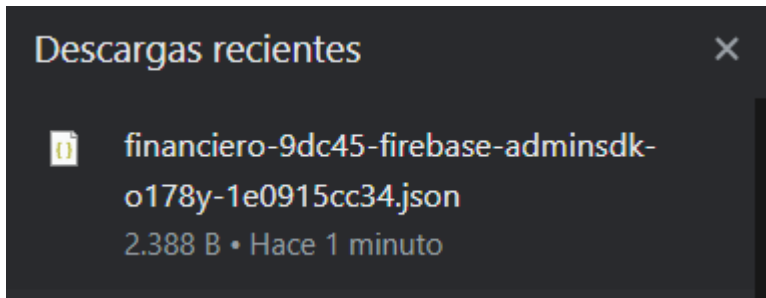
8. Presionamos el boton azul en la opción de Generar Clave.



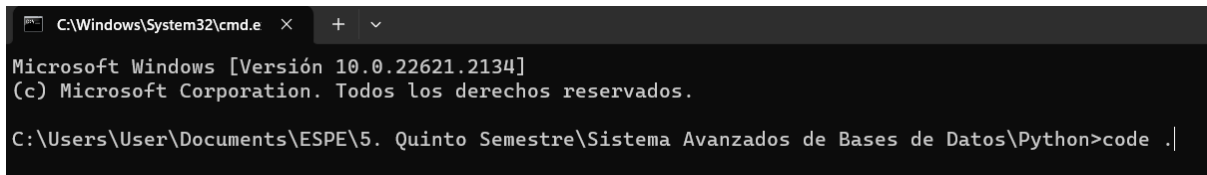
The screenshot shows a modal dialog box titled 'Generar nueva clave privada'. It contains a warning icon and text: 'Tu clave privada brinda acceso a los servicios de Firebase de tu proyecto. Mantenla en confidencialidad y nunca la almacenes en un repositorio público.' Below this, it says 'Guarda este archivo de forma segura, ya que no podrás recuperar la clave nueva si la pierdes.' At the bottom right, there are two buttons: 'Cancelar' and a blue 'Generar clave' button.

9. El siguiente paso es descargar la clave privada.

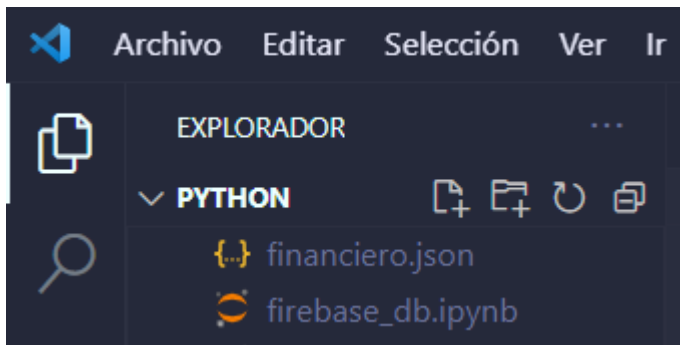
INVESTIGACIÓN



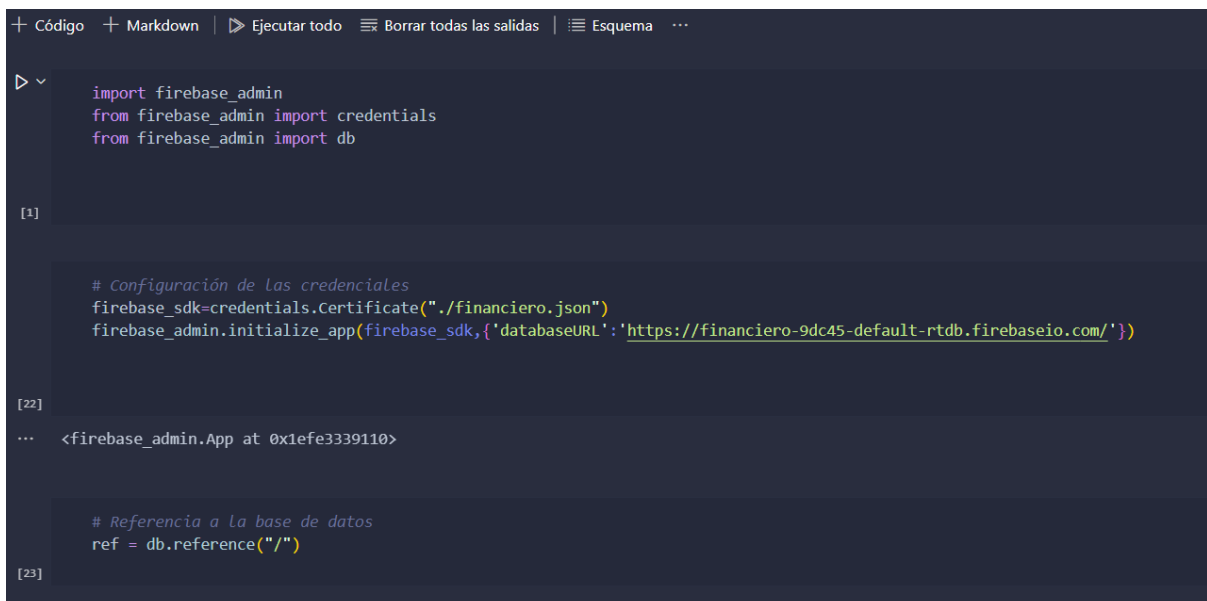
10. Abrimos la carpeta donde vamos a trabajar en el proyecto y presionamos el comando `code .` para posteriormente abrir la carpeta donde vamos hacer el proyecto..



11. Cargamos el archivo de json y también creamos un archivo con extensión ipynb.



12. Hacemos la conexión de firebase con python.



INVESTIGACIÓN

13. Después hacemos el código con los requerimientos que nos pidió el docente facilitador de sistemas avanzados de bases de datos.

Crear Usuario

```
# Crear un nuevo usuario con sus datos en las colecciones relacionadas
nuevo_usuario = {
  "Usuarios": {
    "ID_4": {
      "DATOS GENERALES": {
        "Nombre": "Stalin",
        "Apellido": "Rivera",
        "Celular": "987654321",
        "Dirección": "Avenida Principal"
      },
      "DATOS TÉCNICOS": {
        "Zona": "Zona A",
        "Servicio": "Fibra Óptica",
        "Serie ONT": "67890",
        "IP": "192.168.1.10",
        "Ancho de Banda": "1000 Mbps"
      },
      "DATOS FINANCIEROS": {
        "Plan": "Plan Premium",
        "Inventarios de pagos": {
          "Enero": 2500,
          "Febrero": 1800,
          "Marzo": 2200,
          "Abril": 2400
        }
      }
    },
    "ID_5": {
      "DATOS GENERALES": {
        "Nombre": "Carlos",
        "Apellido": "López",
        "Celular": "765432109",
        "Dirección": "Calle España"
      },
      "DATOS TÉCNICOS": {
        "Zona": "Zona B",
        "Servicio": "Banda Ancha",
        "Serie ONT": "54321",

```

INVESTIGACIÓN

```
        "IP": "192.168.2.20",
        "Ancho de Banda": "500 Mbps"
    },
    "DATOS FINANCIEROS": {
        "Plan": "Plan Estándar",
        "Inventarios de pagos": {
            "Enero": 1500,
            "Febrero": 1200,
            "Marzo": 1300,
            "Abril": 1400
        }
    }
}

ref.update(nuevo_usuario)
```

Información General

```
# Nueva información para el número de celular
nuevo_celular = "55555555"

# Ubicación del dato a actualizar
id_usuario = "ID_1"
ruta_celular = f"Usuarios/{id_usuario}/DATOS GENERALES/Celular"

# Actualizar el número de celular
ref.update({ruta_celular: nuevo_celular})

print("Número de celular actualizado con éxito.")

# Leer todos los datos de la colección "Usuarios"
usuarios_ref = ref.child("Usuarios").get()

if usuarios_ref:
    for id_usuario, datos_usuario in usuarios_ref.items():
        print(f"ID de Usuario: {id_usuario}")
        for categoria, datos in datos_usuario.items():
            print(f"  {categoria}:")
            for clave, valor in datos.items():
                print(f"    {clave}: {valor}")

# Leer todos los datos del usuario con ID "ID_1"
id_usuario = "ID_4"
usuario_ref = ref.child("Usuarios").child(id_usuario).get()
```


INVESTIGACIÓN

```
if usuario_ref:
    print(f"Datos del Usuario {id_usuario}:")
    for categoria, datos in usuario_ref.items():
        print(f"  {categoria}:")
        for clave, valor in datos.items():
            print(f"    {clave}: {valor}")
else:
    print(f"No se encontraron datos para el Usuario {id_usuario}.")
#Leer dato del celular de ID_1
id_usuario = "ID_1"
dato_a_leer = "Celular"

ruta_dato = f"Usuarios/{id_usuario}/DATOS GENERALES/{dato_a_leer}"
valor_dato = ref.child(ruta_dato).get()

if valor_dato:
    print(f"{dato_a_leer} del Usuario {id_usuario}: {valor_dato}")
else:
    print(f"No se encontró el dato {dato_a_leer} para el Usuario {id_usuario}.")
# ID del usuario que deseas eliminar
id_usuario_a_eliminar = "ID_4"

# Ruta a la referencia del usuario a eliminar
usuario_ref = ref.child("Usuarios").child(id_usuario_a_eliminar)

# Eliminar el usuario
usuario_ref.delete()

print(f"Usuario con ID {id_usuario_a_eliminar} eliminado exitosamente.")
```

Creación de los métodos CRUD dentro del código

```
# Funciones para las operaciones CRUD y consultas
def create_user(user_id, user_data):
    user_ref = ref.child('usuarios').child(user_id)
    user_ref.set(user_data)

def update_user(user_id, updated_data):
```

INVESTIGACIÓN

```
user_ref = ref.child('usuarios').child(user_id)
user_ref.update(updated_data)

def delete_user(user_id):
    user_ref = ref.child('usuarios').child(user_id)
    user_ref.delete()

def get_user_payments(user_id):
    payments_ref = ref.child('usuarios').child(user_id).child('pagos')
    return payments_ref.get()

def get_consolidated_payments_by_month(month):
    consolidated_ref = ref.child('consolidado').child(month)
    return consolidated_ref.get()

# Ejemplos de uso
if __name__ == "__main__":
    user_id = 'user123'
    user_data = {
        # ...
    }

    create_user(user_id, user_data)

    updated_data = {
        # ...
    }
    update_user(user_id, updated_data)

    delete_user(user_id)

    user_payments = get_user_payments(user_id)
    print(user_payments)

    consolidated_payments = get_consolidated_payments_by_month('Enero')
    print(consolidated_payments)
```

14. Finalmente observamos los datos que se crearon dentro de Firebase con diferentes tipos de objetos.

INVESTIGACIÓN

 <https://financiero-9dc45-default-rtdb.firebaseio.com><https://financiero-9dc45-default-rtdb.firebaseio.com/>

- ▶ Usuarios
- ▶ combined_data
- ▶ financial_data
- ▶ technical_data
- ▶ users

15. Visualizamos los datos dentro de un json.

```
{
  "Usuarios": {
    "ID_1": {
      "DATOS GENERALES": {
        "Celular": "555555555"
      }
    },
    "ID_5": {
      "DATOS FINANCIEROS": {
        "Inventarios de pagos": {
          "Abril": 1400,
          "Enero": 1500,
          "Febrero": 1200,
          "Marzo": 1300
        },
        "Plan": "Plan Estándar"
      },
      "DATOS GENERALES": {
        "Apellido": "López",
        "Celular": "765432109",
        "Dirección": "Calle España",
        "Nombre": "Carlos"
      },
      "DATOS TÉCNICOS": {
        "Ancho de Banda": "500 Mbps",
```

INVESTIGACIÓN

```
"IP": "192.168.2.20",
"Serie ONT": "54321",
"Servicio": "Banda Ancha",
"Zona": "Zona B"
}
},
"combined_data": {
  "user5": {
    "financial_id": "financial5",
    "technical_id": "technical5",
    "user_id": "user5"
  },
  "user6": {
    "financial_id": "financial6",
    "technical_id": "technical6",
    "user_id": "user6"
  }
},
"financial_data": {
  "financial5": {
    "financial_id": "financial5",
    "payment_inventories": [
      "Enero",
      "Febrero",
      "Marzo",
      "Abril",
      "Mayo",
      "Junio"
    ],
    "plan": "Plan Gold"
  },
  "financial6": {
    "financial_id": "financial6",
    "payment_inventories": {
      "Abril": 170,
      "Agosto": 220,
      "Enero": 120,
      "Febrero": 150,
      "Julio": 210,
      "Junio": 190,
      "Marzo": 180,
```

INVESTIGACIÓN

```
    "Mayo": 200
  },
  "plan": "Plan Silver"
}
},
"technical_data": {
  "technical5": {
    "bandwidth": "50 Mbps",
    "ip": "192.168.3.4",
    "ont_series": "ONT-987",
    "service": "Service Y",
    "technical_id": "technical5",
    "zone": "Zone E"
  },
  "technical6": {
    "bandwidth": "100 Mbps",
    "ip": "192.168.1.2",
    "ont_series": "ONT-123",
    "service": "Service X",
    "technical_id": "technical6",
    "zone": "Zone A"
  }
},
"users": {
  "user5": {
    "address": "123 Pine St",
    "first_name": "Charlie",
    "last_name": "Brown",
    "phone": "5555555555",
    "user_id": "user5"
  },
  "user6": {
    "address": "789 Oak St",
    "first_name": "Alice",
    "last_name": "Smith",
    "phone": "5555555555",
    "user_id": "user6"
  }
}
}
```

INVESTIGACIÓN**D. CONCLUSIONES**

Al implementar Python con Firebase, se logró una conexión sólida entre la aplicación y la base de datos, permitiendo un manejo fluido de objetos y una comunicación en tiempo real. Esto simplifica la manipulación de datos y mejora la experiencia del usuario.

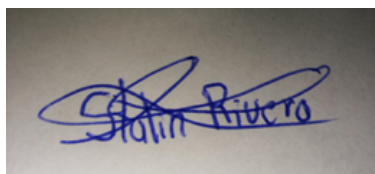
Mediante la optimización del código Python, se obtuvo una mejora notable en la velocidad de acceso y manipulación de objetos en la base de datos de Firebase. Esto resulta en tiempos de respuesta más cortos y una aplicación más ágil en general.

Para mantener la eficiencia en el manejo de objetos con Firebase y Python, es crucial seguir buenas prácticas de programación y mantener el código limpio y estructurado. Además, es recomendable utilizar las bibliotecas y herramientas adecuadas para trabajar con Firebase en Python, como "firebase-admin" para un acceso seguro a la base de datos y "pyrebase" para facilitar la interacción con la API de Firebase.

E. BIBLIOGRAFÍAS

1. Firebase. (2023). medium. Obtenido de https://firebase.google.com/?gad=1&gclid=Cj0KCQjwrfymBhCTARIsADXTbkpFI9tkwOO79ZM8tJwmQooD8fnH8DxyOkjwrXfJclBg_1ouMJQ3ogaAtI9EALw_wcB&gclsrc=aw.ds&hl=es-419

Firma(s)



Nombre(s) Estudiante(s): Stalin Bladimir Rivera Vega