NOVIEMBRE 2023

ARMARIO 2.0

YULIA VERETENNIKOVA
JORDI TEIXIDÓ

ÍNDICE:

1. Prefacio

Hoy en día cada persona tiene 148 piezas de ropa según Capsulewardrobedata.com

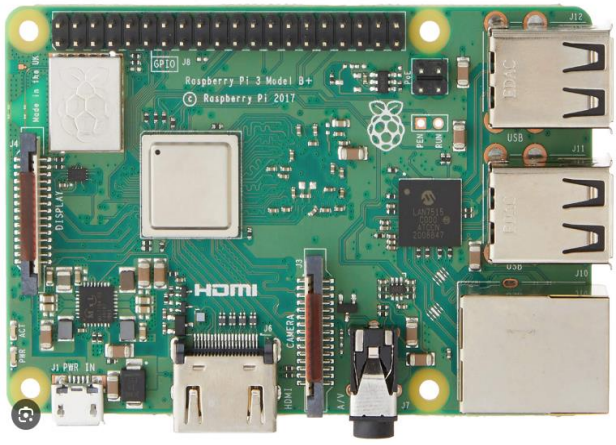
Aunque nuestro cerebro tiene una capacidad de 2,5 peta bytes de memoria, hay muchas cosas que suceden cada día y el problema de olvidar de lo que tienes en el armario es muy común, sobre todo si tienes hijos.

Este proyecto consta de Raspberry PI, etiquetas RFID, lector RFID, recogida de datos en la nube (Firebase), su almacenamiento y comunicación con la Raspberry y también de un armario simulado con un modelo 3D creado en Tinkercad.

A nivel de usuario el proyecto Vestuario 2.0 ayuda a comprender hábitos y preferencias del usuario y simplificar la analítica y toma de decisiones.

2. Material para el Proyecto

Raspberry PI 3



Es una placa de microcontrolador con el sistema operativo Linux que nos permitirá activar el lector RFID y programar tags/etiquetas y enviar la información recibida de lector a la base de dados y comunicarse con ella.

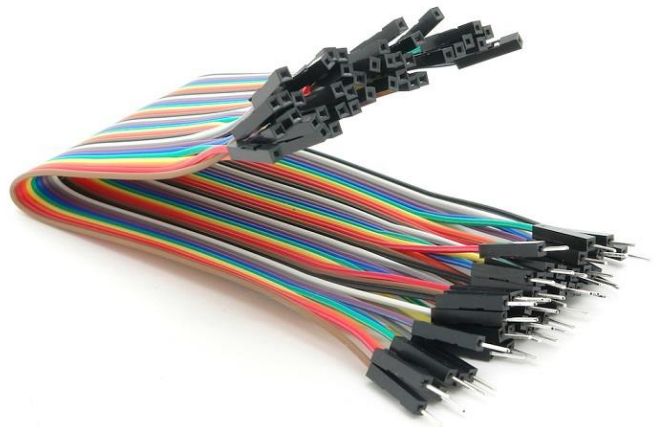
Módulo FRID-RC522

Este módulo opera con frecuencia de 13.56 MHz, nos permite comunicar con el microcontrolador Raspberry PI 3 y también grabar y leer la información de las etiquetas u otros dispositivos pasivos RFID de 13.56 MHz. Maneja el ISO14443-A y soporta el algoritmo de encriptación MIFARE y Quick CRYPTO1.

Detecta los tags RFID de 0 a 30 cm.

La tensión de alimentación es de 3.3V.

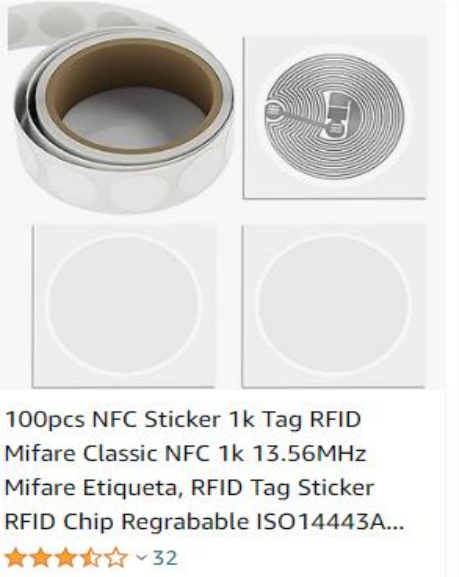
Cables Dupont



Cables Dupont sirven para conectar componentes y transferir señales eléctricas de cualquier parte de la placa de prototipos.

Necesitamos 7 cables macho-hembra para el proyecto actual.

Tags (Etiquetas) RFID 13.56 MHz



Los tags pueden ser tarjetas, etiquetas, claves, discos que representan un sistema de almacenamiento donde la memoria está dividida en bloques, con mecanismos simple para el acceso a la información.

MIFARE Classic 1K dispone de 1024 bytes de memoria divididos en 16 sectores de 64 bytes, cada uno protegido por dos claves llamadas A y B. Cada una puede ser programada individualmente para permitir o bloquear operaciones lectura o escritura.

Cada sector reserva una cierta memoria para las claves A y B, por lo que este espacio normalmente no puede ser empleado para guardar datos, lo que reduce la cantidad de memoria disponible en una MIFARE Classic 1K a 752 bytes.

La memoria EEPROM de las tarjetas MIFARE Classic puede soportar más de 100.000 ciclos de escritura, y pueden mantener la memoria durante más de 10 años sin recibir alimentación.

Los tags MIFARE Classic emplean el estándar ISO/IEC 14443 Type A y funcionan con frecuencia 13.56 MHz.

Para el proyecto actual usamos etiquetas **Mifare Classic ISO14443-A** que tienen una base adhesiva.

Servicio Cloud para almacenar los dados y consultarlos/modificarlos

Firebase es una herramienta que permite crear una base de datos noSQL. La BBDD está alojada en la nube y facilita amanecer, sincronizar y buscar dados en tiempo real.

La usamos para almacenar la información de cada pieza de ropa que disponemos y también para poder ver la estadística del uso del vestuario.

Programa de Creación de Diseño 3D

Tinkercad es una herramienta que permite crear modelos tridimensionales basados en geometría solida constructiva. Tiene capacidad de simular escenarios reales.

3. Fases del Trabajo:

1) Montar el circuito con el lector RFID
2) Activar el módulo SPI en la Raspberry
3) Instalar la librería MFRC522.py a la Raspberry
4) Comprobar si el lector está activado creando un programa read.py y ver si lee la tarjeta del kit inicial.
5) Crear un programa write.py para poder programar la tarjeta y las etiquetas.
6) Crear un programa para establecer comunicación instantánea entre los dados registrados por la Raspberry y la base de dados Firebase.

Montar el circuito RFID

Los orígenes de la tecnología RFID se remontan al año 1920 en el instituto de Tecnología de Massachusetts (MIT).

Durante la 2ª Guerra Mundial los británicos empezaron a usar la tecnología de reconocimiento de aeroplanos como amigos o enemigos.

La tecnología RFID funciona de la siguiente manera: el lector lanza una señal RF mediante la que el tag RFID se activa y emite una respuesta.

Cada RFID tag tiene un código de identificación único que se puede personalizar.

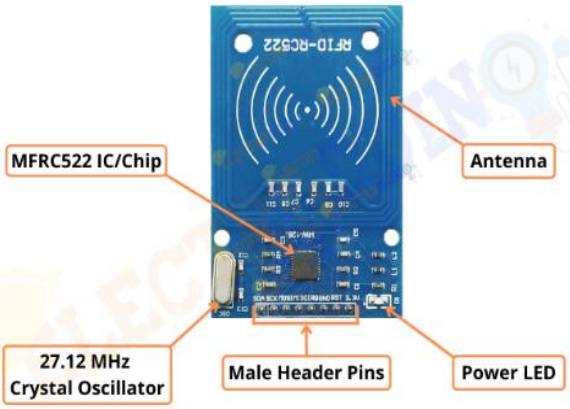
Depende del rango de frecuencia, los tags pueden operar en diferentes sistemas, de baja frecuencia, alta frecuencia, ultra alta frecuencia.

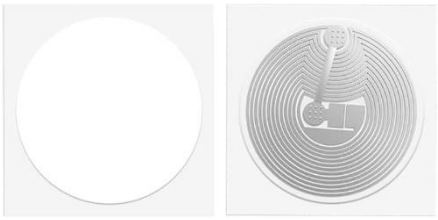
Bandas de frecuencia utilizadas en *RFID*

Banda de frecuencias	Descripción	Rango
125 kHz	LF (Baja Frecuencia)	Hasta 50 cm.
13,56 MHz	HF (Alta Frecuencia)	De 8 cm.
400 MHz - 1.000 MHz	UHF (Ultra Alta Frecuencia)	De 3 a 10 m.
2,45 GHz - 5,4 GHz	Microondas	Más de 10 m.

Los sistemas RFID pueden transmitir su propia señal con la información almacenada en el chip que tiene potencia propia. En este caso el sistema RFID es **activo**.

En contrario, los sistemas **RFID pasivos** no cuentan con una fuente de alimentación propia que permite minimizar el tamaño del dispositivo RFID. El corriente eléctrico del dispositivo se activa cuando acercamos un lector RFID que envía energía a la antena de la etiqueta. Esto es el método que elegimos para nuestro proyecto.





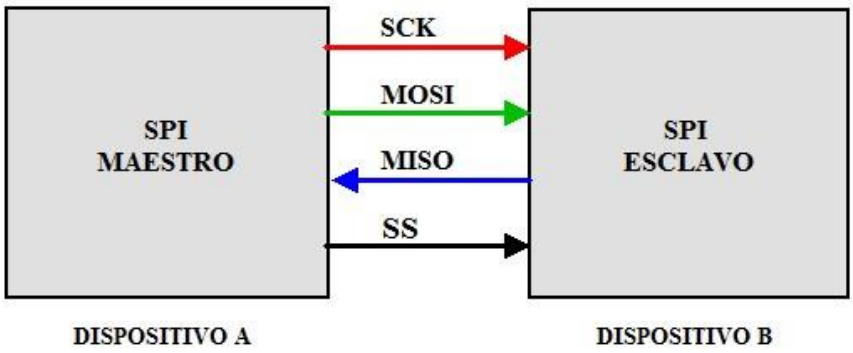
Lector RFID Tag RFID

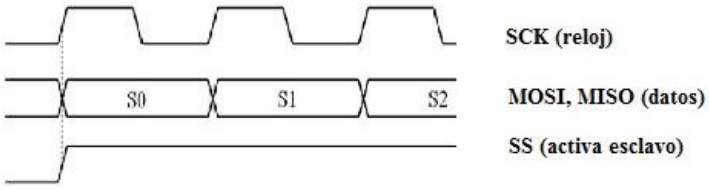
El módulo lector RFID-RC522 del proyecto actual utiliza 3.3V DC como voltaje de alimentación y se controla a través de protocolo SPI y por esto esta compatible con diferentes microcontroladores como en nuestro caso, con Raspberry PI 3.

El protocolo SPI (**Syncronous Peripheral Interface**) está utilizado para la comunicación serial entre dispositivos. El SPI fue inicialmente creado por Motorola y adoptado posteriormente por diferentes fabricantes, como Microchip y Atmel.

Los dispositivos SPI se comunican entre sí utilizando un bus de 4 señales (**MOSI, MISO, SCK, SS**) y un esquema maestro/esclavo, en el cual el maestro inicia el protocolo de trasmisión de los datos.

El RC522 utiliza el sistema de modulación y demodulación para todo tipo de dispositivos pasivos de frecuencia 13.56 MHz.





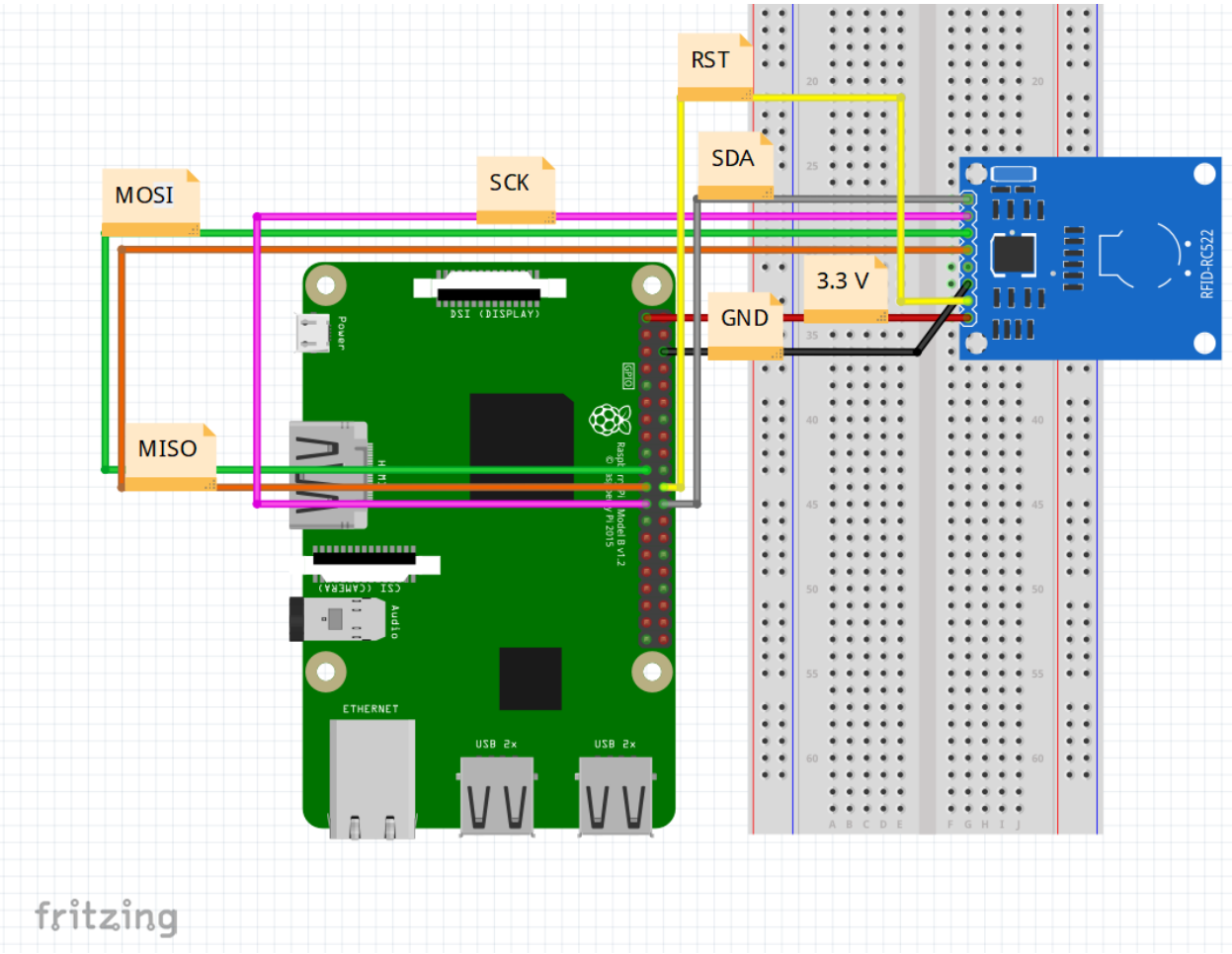
MOSI = MASTER OUT, SLAVE IN
MISO = MASTER IN, SLAVE OUT
SCK = SERIAL CLOCK
SS = SLAVE SELECT
SPI = SYNCRONOUS PERIPHERAL INTERFACE

ESTÁNDAR DE COMUNICACIÓN SPI

Montamos el circuito con el lector RFID conectando los siguientes pines del lector RFID y GPIOs (o pines) de la Raspberry utilizando 7 cables Dupont macho-hembra:

RFID	**BCM**	**BOARD**
3.3 V	3.3 V	PIN 1
GND	GND	PIN 6
SDA	GPIO 8	PIN 24
MISO	GPIO 9	PIN 21
MOSI	GPIO 10	PIN 19
SCK	GPIO 11	PIN 23
RST	GPIO 25	PIN 22

La simulación **Fritzing** es la siguiente:



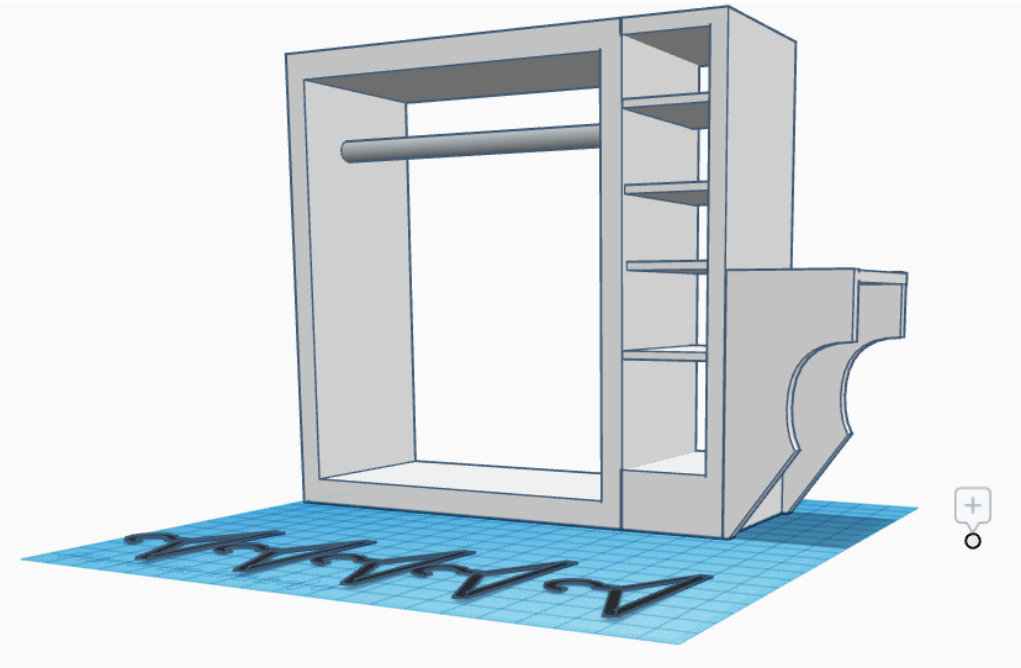
Comprobamos si el lector RFID funciona pasando dos pequeños programas (**READ.py y WRITE.py**) a la Raspberry para poder leer y guardar información en los tags. Activamos el **protocolo SPI**.

```python
import RPi.GPIO as GPIO
from mfrc522 import SimpleMFRC522

reader = SimpleMFRC522()

while True:
    try:
        id, text = reader.read()
        print(id)
        print(text)
    finally:
        GPIO.cleanup()

```

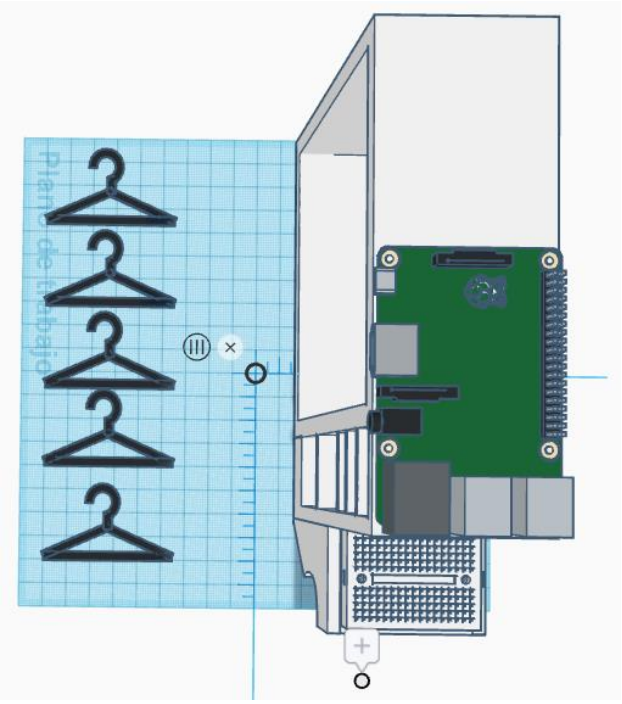
```python
import RPi.GPIO as GPIO
from mfrc522 import SimpleMFRC522

reader = SimpleMFRC522()

try:
        text = input('New data:')
        print("Now place your tag to write")
        reader.write(text)
        print("Written")
finally:
        GPIO.cleanup()

```

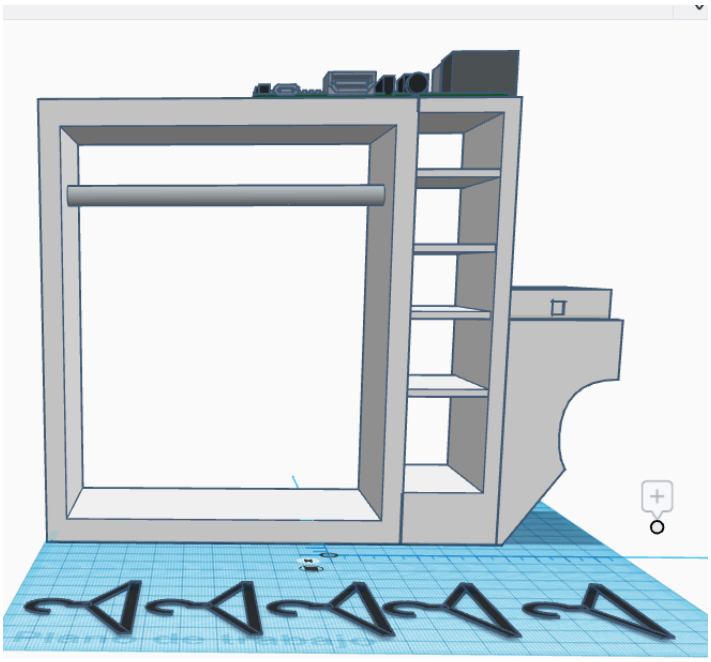
Haciendo el test del circuito, veamos el número del tag leído y la información grabada por nosotros:

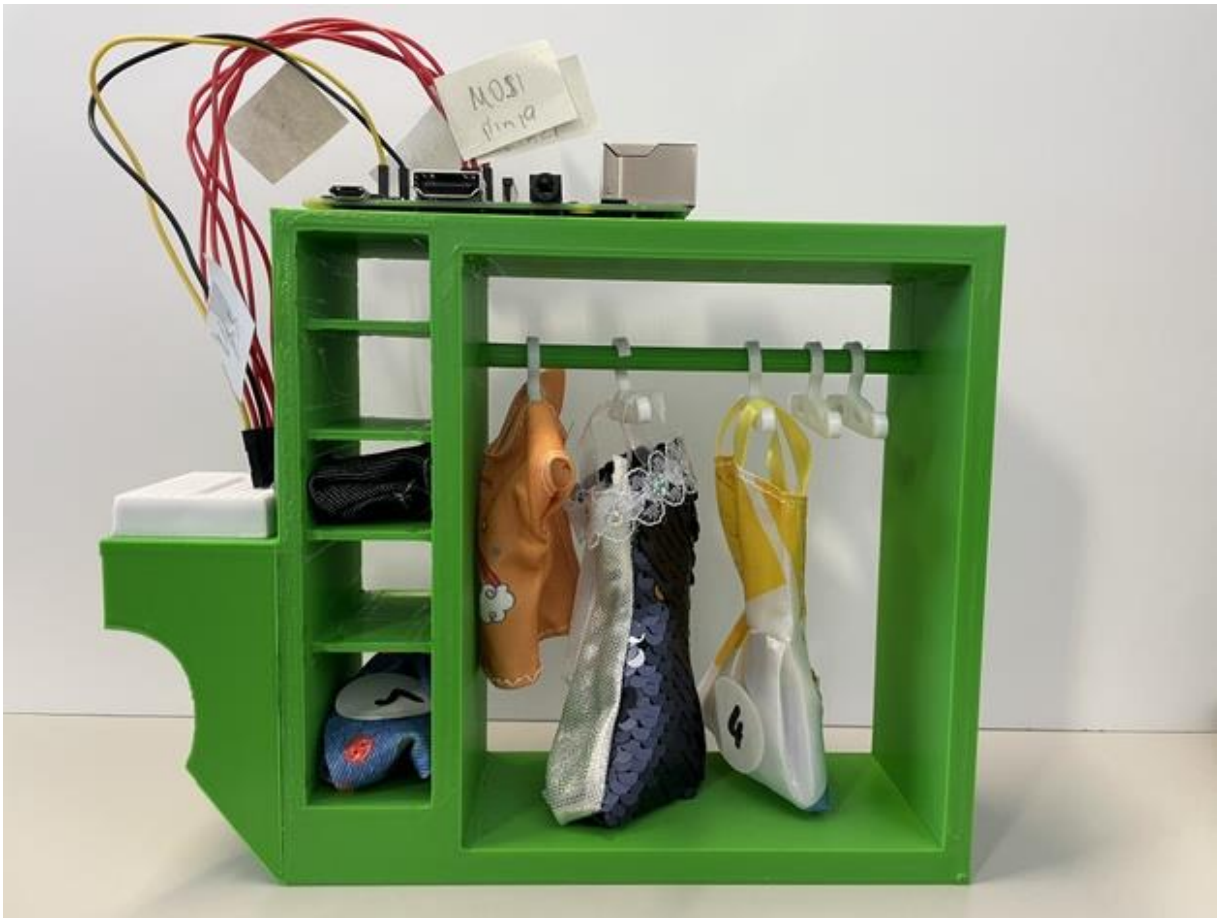
Creamos el **modelo 3D del vestuario** con el programa Tinkercad en línea.



Para asegurarnos la colocación de la Breadboard y la Raspberry, hacemos la simulación final ajustando las dimensiones.







Creamos un programa para recibir los dados de la Raspberry al Cloud.

Como que nuestro proyecto está hecho con el microcontrolador Raspberry PI3, usamos el lenguaje Python para programar la base de dados.

El programa nos permitirá:

1) Registrar una nueva pieza del vestuario y grabar esta información en la etiqueta RFID, pasar la información de la pieza nueva a la Firebase.
2) Leer la información grabada en la etiqueta RFID.
3) Modificar la información grabada en la etiqeuta RFID.
4) Ver las estadísticas de la pieza mas y menos usada de todas registradas en la Firebase.
5) Borrar la pieza del armario.

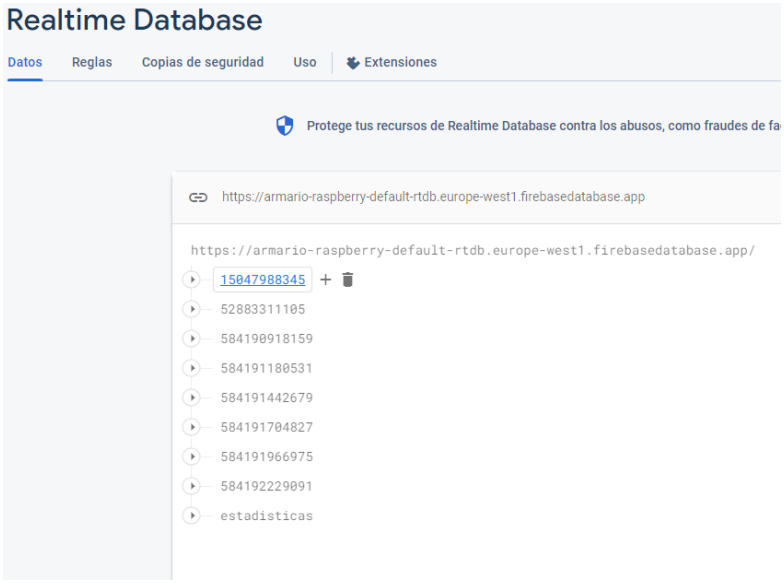
Importamos las bibliotecas del lector RFID (mfrc522 y SimpleMFRC522), pyrebase, json en la Raspberry.

Modo 1 Console – Registrar la Prenda

El registro de la prenda se hace con la Raspberry acercando el tag al lector:

```
Press intro to continue
MENU: C.R.U.D
***************
1- Crear
2- Leer/Consultar
3- Update/Modificar
4- Borrar/Delete
5- Salir
Elija opción: 1
AUTH ERROR!!
AUTH ERROR(status2reg & 0x08) != 0
El código leido es: 584191966975
Introduzca el nombre de la prenda: vestido
Introduzca el color de la prenda: lentejuelas negro
Intoduzca la temporada de la prenda: fiesta
--PRENDA--
Código:584191966975
nombre:vestido
color:lentejuelas negro
temporada:fiesta
usos:0
*********
{"codigo": 584191966975, "nombre": "vestido", "color": "lentejuelas negro", "tem
porada": "fiesta", "usos": 0}
****setField(584191966975, {'codigo': 584191966975, 'nombre': 'vestido', 'color'
: 'lentejuelas negro', 'temporada': 'fiesta', 'usos': 0})
Press intro to continue
```

Las dados se registran en la Firebase:



Modo 2 Console - Leer/Consultar

```
MENU: C.R.U.D
***************
1- Crear
2- Leer/Consultar
3- Update/Modificar
4- Borrar/Delete
5- Salir
Elija opción: 2
AUTH ERROR!!
AUTH ERROR(status2reg & 0x08) != 0
fromJSON()
JSON=OrderedDict([('codigo', 584191180531), ('color', 'naranja'),
color=naranja
--PRENDA--
Código:584191180531
nombre:camiseta
color:naranja
temporada:otoño
usos:0
```

Las dados se registran en la Firebase:

Modo 2 - Uso

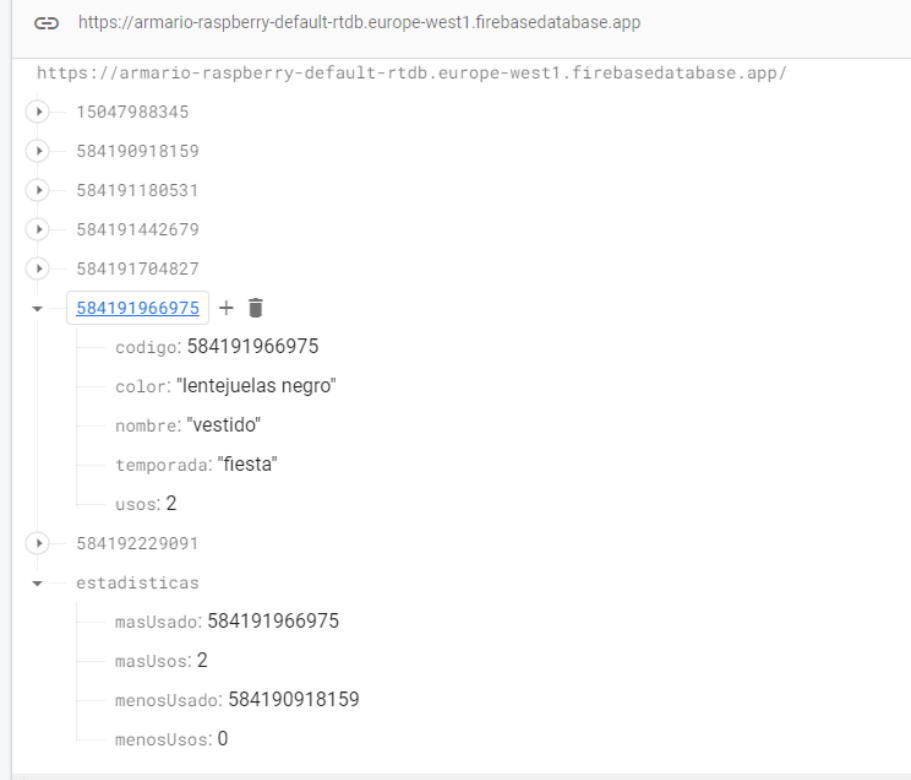
Cada vez cuando la etiqueta pasa por el lector, la información de “Uso” se registra en la etiqueta y también en la Firebase:

```
MENU: PRINCIPAL
**************
1- Mode console
2- Mode use
3- Mode estadisticas
Elija opción: 2
AUTH ERROR!!
AUTH ERROR(status2reg & 0x08) != 0
fromJSON()
JSON=OrderedDict([('codigo', 584191180531), ('color', 'naranja'), ('nombre', 'camiseta'), ('te
os', 0)])
color=naranja
****setField(584191180531, {'codigo': 584191180531, 'nombre': 'camiseta', 'color': 'naranja',
sos': 1})
```

Modo 3 – Estadísticas

Este modo nos enseña la prenda más usada y la prenda menos usada de las registradas en la Firebase:

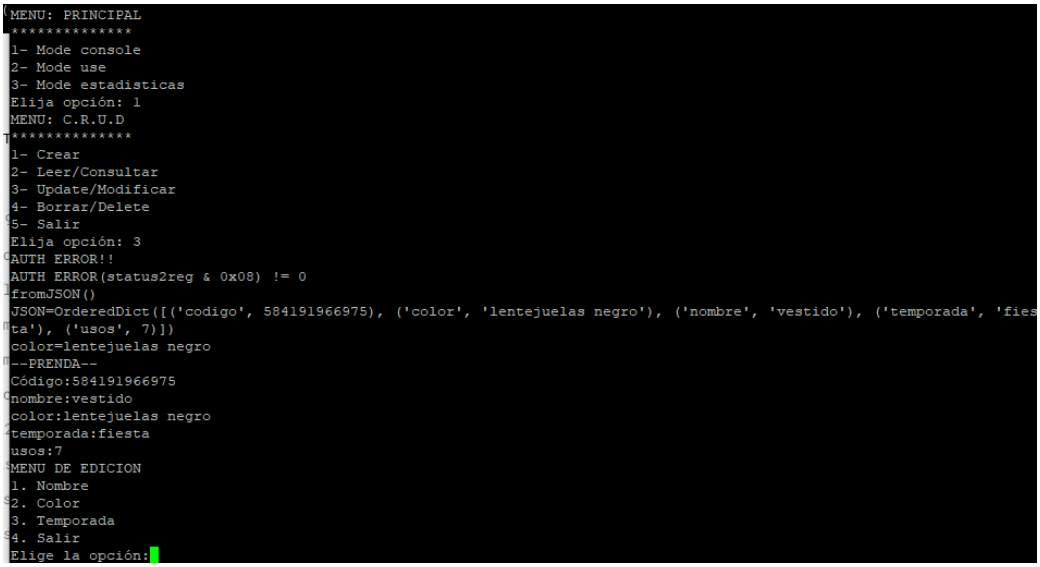
```
JSON=OrderedDict([('codigo', 584191966975), ('color', 'lentejuelas negro'), ('nombre', 'vestido'), ('temporada', 'fies
ta'), ('usos', 1)])
color=lentejuelas negro
****setField(584191966975, {'codigo': 584191966975, 'nombre': 'vestido', 'color': 'lentejuelas negro', 'temporada': 'f
iesta', 'usos': 2})
****setField(estadisticas, {'masUsos': 2, 'masUsado': 584191966975, 'menosUsos': 0, 'menosUsado': 584190918159})
```

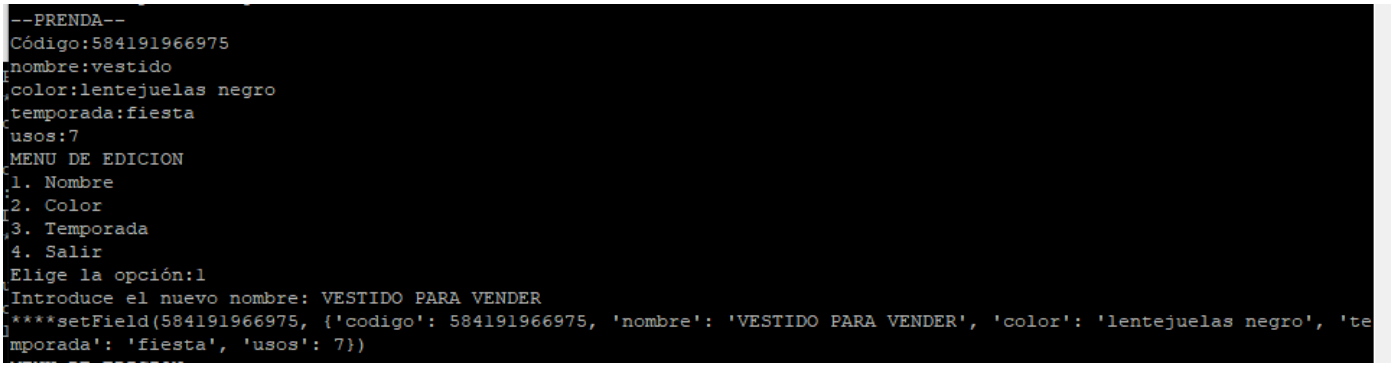


Modo 3 Update/Modificar

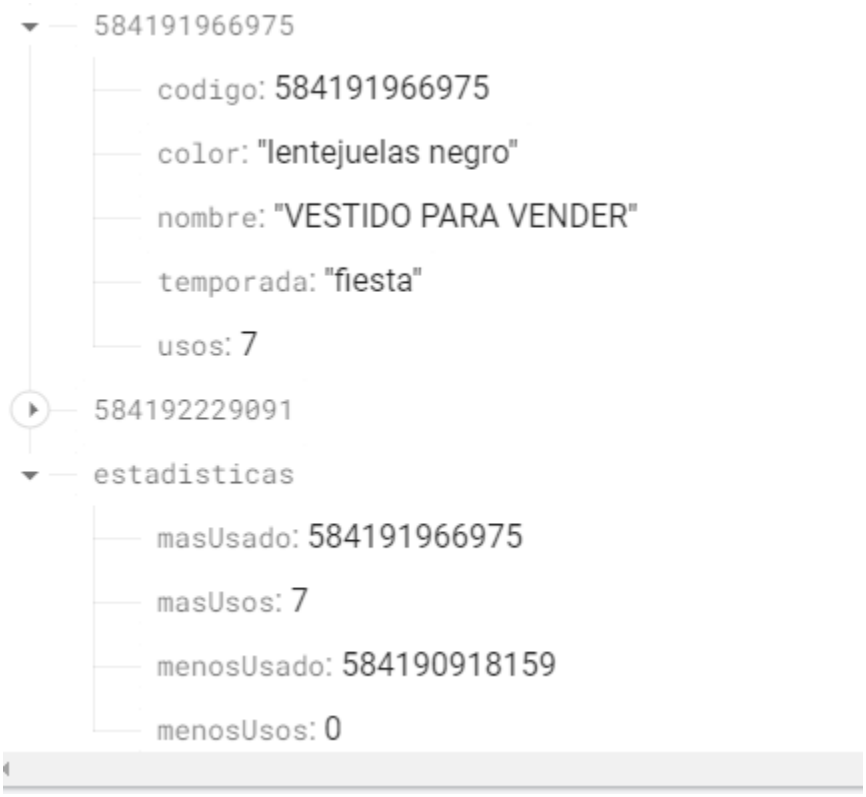
Este modo nos permite editar la información sobre la prenda registrada.

Cambiamos el nombre de la prenda de “vestido” a “VESTIDO PARA VENDER”





Los cambios se registran en la Firebase:



Modo Borrar/Delete:

Borramos la prenda desde la Raspberry

```
MENU: C.R.U.D
**************
1- Crear
2- Leer/Consultar
3- Update/Modificar
4- Borrar/Delete
5- Salir
Elija opción: 4
AUTH ERROR!!
AUTH ERROR(status2reg & 0x08) != 0
MENU: C.R.U.D
**************
1- Crear
2- Leer/Consultar
3- Update/Modificar
4- Borrar/Delete
5- Salir
Elija opción:
```

La prenda desaparece de la BBDD:

Programación

```python
from mfrc522 import SimpleMFRC522
import pyrebase
import time
import datetime
import json

class Firebase:

    @property
    def ready(self):
        return self._ready
    @ready.setter
    def ready(self, v):
        self._ready = v

    def __init__(self):
        try:
            self.ready = False
            self.configurate()
            self.connect()
            self.ready = True
        except:
            print("excepcion in init de Firebase")

    def configurate(self):
        #project = "armario-raspberry"
        try:

            configuration = {
                "apiKey": "apiKey",
                "authDomain": "armario-raspberry.firebaseapp.com",
                "databaseURL": "https://armario-raspberry-default-rtdb.europe-west1.firebasedatabase.app/",
                "storageBucket": "armario-raspberry.appspot.com"
            }
            self.configuration = configuration
        except:
            result = False
        else:
            result = True
        finally:
            return result

    def connect(self):
        try:
            self.firebase = pyrebase.initialize_app(self.configuration)
            self.db = self.firebase.database()
        except:
```

```
            result = False
        else:
            result = True
        finally:
            return result

    def setField(self, fields, data):
        print(f"****setField({fields}, {data})")
        try:
            self.splitFields(fields).set(data)
        except:
            result = False
        else:
            result = True
        finally:
            return True

    def splitFields(self, fields):
        try:
            field=self.db.child("/")
            fields=fields.split(".")
            for fld in fields:
                field=field.child(fld)
        except:
            field = None
        finally:
            return field

    def updateField(self, fields, data):
        try:
            if self.getField(fields)==None:
                self.setField(fields, data)
            else:
                self.setField(fields, data)
                #self.splitFields(fields).update(data)
        except:
            self.splitFields(fields).update(data)
            result = False
        else:
            result = True
        finally:
            return result

    def disconnect(self):
        try:
```

```
            self.firebase = None
        except:
            result = False
        else:
            result = True
        finally:
            return result

    def getField(self, fields):
        try:
            result = self.splitFields(fields).get().val()
        except:
            result = None
        else:
            pass
        finally:
            return result

    def removeField(self,fields):
        try:
            self.splitFields(fields).remove()
        except:
            result = False
        else:
            result = True
        finally:
            return result

    def listFields(self,fields):
        try:
            result = self.splitFields(fields).get()
        except:
            result = None
        finally:
            return result

    def startThread(self):
        try:
            self.thread = t.ThreadFirebase(self, self.main)
            self.thread.start()
        except:
            result = False
        else:
            result = True
        finally:
            return result

```

```python
class Prenda:
    def __init__(self, codigo="", nombre="", color="", temporada=""):
        self.codigo = codigo
        self.nombre = nombre
        self.color = color
        self.temporada = temporada
        self.usos = 0
        #self.fecha = datetime.date.today()

    def aumentar_usos(self):
        #self['usos'] += 1
        self.usos += 1
        #self.actualizar_fecha()

    def actualizar_fecha(self):
        #self.fecha = datetime.date.today()
        pass

    def toJSON(self):
        return json.dumps(self.__dict__)

    def fromJSON(self, JSON):
        print("fromJSON()")
        print(f"JSON={JSON}")
        print(f"color={JSON['color']}")
        self.codigo = JSON['codigo']
        self.color = JSON['color']
        self.nombre = JSON['nombre']
        self.temporada = JSON['temporada']
        self.usos = JSON['usos']

    def toFB(self):
        return {"codigo":self.codigo,"nombre":self.nombre,"color":self.color,"temporada":self.temporada,"usos":self.usos}

    def fromDictionary(self, dictionary):
        self.codigo = dictionary.get('codigo')
        self.nombre = dictionary.get('nombre')
        self.color = dictionary.get('color')
        self.temporada = dictionary.get('temporada')
        self.usos = dictionary.get('usos')
        #self.fecha = dictionary.get('fecha')

    def show(self):
        print("--PRENDA--")
        print(f"C□digo:{self.codigo}")
```

```
185         print(f"nombre:{self.nombre}")
186         print(f"color:{self.color}")
187         print(f"temporada:{self.temporada}")
188         print(f"usos:{self.usos}")
189         #print(f"fecha:{self.fecha}")
190 
191 class Estadisticas:
192     def __init__(self):
193         self.masUsos = None
194         self.menosUsos = None
195         self.masUsado = None
196         self.menosUsado = None
197         #self.ultima = None
198         #self.antigua = None
199 
200     def toJSON(self):
201         return json.dumps(self.__dict__)
202 
203     def fromDictionary(self, dictionary):
204         self.masUsos = dictionary.get('masUsos')
205         self.menosUsos = dictionary.get('menosUsos')
206         self.menosUsado = dictionary.get('menosUsado')
207         self.masUsado = dictionary.get('masUsado')
208         #self.ultima = dictionary.get('ultima')
209         #self.antigua = dictionary.get('antigua')
210 
211     def fromJSON(self, JSON):
212         print("fromJSON()")
213         print(f"JSON={JSON}")
214         self.masUsos = JSON['masUsos']
215         self.menosUsos = JSON['menosUsos']
216         self.menosUsado = JSON['menosUsado']
217         self.masUsado = JSON['masUsado']
218 
219     def toFB(self):
220         return {"masUsos":self.masUsos,"masUsado":self.masUsado,"menosUsos":self.menosUsos, "menosUsado":self.menosUsado}
221 
222     def show(self):
223         print("*******ESTADISTICIAS*******")
224         print(f"masUsos={self.masUsos}")
225         print(f"masUsado={self.masUsado}")
226         print(f"menosUsos={self.menosUsos}")
227         print(f"menosUsado={self.menosUsado}")
228 
229 class Reader(SimpleMFRC522):
230     import RPi.GPIO as GPIO
231 
```

```
232     def __init__(self):
233         super().__init__()
234 
235     def my_write(self, data):
236         self.write(data)
237 
238     def my_read(self):
239         id, data = self.read()
240         return id, data
241 
242 class Main():
243     def __init__(self):
244         print("1 main")
245         self.setup()
246         print("2 main")
247         while True:
248             print("3 main")
249             self.loop()
250             print("4 main")
251 
252     def setup(self):
253         print("1 setup")
254         self.fb = Firebase()
255         print("2 setup")
256         self.reader = Reader()
257         print("3 setup")
258         self.estadisticas = Estadisticas()
259         print("4 setup")
260         estadisticas_JSON = self.fb.getField("estadisticas")
261         print("5 setup")
262         datos = self.fb.getField("/")
263         print("6 setup")
264         print("***************")
265         print("7 setup")
266         print(f"JSON={datos}")
267         print("8 setup")
268         time.sleep(1000)
269         print("9 setup")
270         datosFB = datos['estadisticas']
271         print("10 setup")
272         print("***************")
273         print("11 setup")
274         print(f"JSON={datosFB}")
275         print("12 setup")
276         time.sleep(1000)
```

```
277         print("13 setup")
278         self.estadisticas.fromJSON(estadisticas_JSON)
279 
280     def loop(self):
281         print("MENU: PRINCIPAL")
282         print("**************")
283         print("1- Mode console")
284         print("2- Mode use")
285         print("3- Mode estadisticas")
286         option = int(input("Elija opción: "))
287         if option == 1:
288             self.mode_console()
289         elif option == 2:
290             self.mode_use()
291         elif option == 3:
292             self.mode_estadisticas()
293         else:
294             print("opcion no valida")
295             print("vuelva a intentarlo")
296 
297     def mode_console(self):
298         end = False
299         while not end:
300             print("MENU: C.R.U.D")
301             print("**************")
302             print("1- Crear")
303             print("2- Leer/Consultar")
304             print("3- Update/Modificar")
305             print("4- Borrar/Delete")
306             print("5- Salir")
307             option = int(input("Elija opción: "))
308             if option == 1:
309                 self.create()
310             elif option == 2:
311                 self.read()
312             elif option == 3:
313                 self.update()
314             elif option == 4:
315                 self.delete()
316             elif option == 5:
317                 end = True
318             else:
319                 print("opcion no valida")
320                 print("vuelva a intentarlo ")
321 
```

```
322     def create(self):
323         codigo, data = self.reader.my_read()
324         print(f"El código leido es: {codigo}")
325         nombre = input("Introduzca el nombre de la prenda: ")
326         color = input("Introduzca el color de la prenda: ")
327         temporada = input("Intoduzca la temporada de la prenda: ")
328         prenda = Prenda(codigo, nombre, color, temporada)
329         prenda.show()
330         print("*********")
331         print(prenda.toJSON())
332         self.fb.setField(str(codigo), prenda.toFB())
333         #self.reader.my_write(nombre)
334         input("Press intro to continue ")
335 
336     def read(self):
337         codigo, data = self.reader.my_read()
338         prenda_JSON = self.fb.getField(str(codigo))
339         prenda = Prenda()
340         prenda.fromJSON(prenda_JSON)
341         prenda.show()
342 
343     def update(self):
344         codigo, data = self.reader.my_read()
345         prenda_JSON = self.fb.getField(str(codigo))
346         prenda = Prenda()
347         prenda.fromJSON(prenda_JSON)
348         prenda.show()
349         opcion = None
350         while opcion != 4:
351             print("MENU DE EDICION")
352             print("1. Nombre")
353             print("2. Color")
354             print("3. Temporada")
355             print("4. Salir")
356             opcion = int(input("Elige la opción:"))
357             if opcion == 1:
358                 nombre = input("Introduce el nuevo nombre: ")
359                 prenda.nombre = nombre
360             elif opcion == 2:
361                 color = input("Introduce el nuevo color: ")
362                 prenda.color = color
363             elif opcion == 3:
364                 temporada = input("Introduce el nueva temporada: ")
365                 prenda.temporada = temporada
366             self.fb.setField(str(prenda.codigo), prenda.toFB())
367 
368     def delete(self):
```

```
369             codigo, data = self.reader.my_read()
370             self.fb.removeField(str(codigo))
371             #self.reader.my_write(prenda.nombre)
372 
373         def mode_use(self):
374             try:
375                 while True:
376                     codigo, data = self.reader.my_read()
377                     #LANZAR SONIDO BEEP
378                     #aumento el numero de veces
379                     prenda_JSON = self.fb.getField(str(codigo))
380                     prenda = Prenda()
381                     prenda.fromJSON(prenda_JSON)
382                     prenda.aumentar_usos()
383                     #actualizo FB y la Tarjeta
384                     self.fb.setField(str(codigo), prenda.toFB())
385                     self.remake_estadisticas()
386                     #self.reader.my_write(prenda)
387                     time.sleep(2.5)
388             except KeyboardInterrupt:
389                 pass
390 
391         def remake_estadisticas(self):
392             datos = self.fb.getField("/")
393             datosFB = datos['estadisticas']
394             first = True
395             estadisticas = Estadisticas()
396             estadisticas.fromDictionary(datosFB)
397             for codigo in datos:
398                 if first:
399                     self.estadisticas.masUsos = datos[codigo]['usos']
400                     self.estadisticas.masUsado = datos[codigo]['codigo']
401                     self.estadisticas.menosUsos = datos[codigo]['usos']
402                     self.estadisticas.menosUsado = datos[codigo]['codigo']
403                     #self.estadisticas.ultima = datos[codigo]['fecha']
404                     #self.estadisticas.antigua = datos[codigo]['fecha']
405                     first = False
406                 if codigo != "estadisticas":
407                     prenda = Prenda()
408                     prenda.fromDictionary(datos[codigo])
409                     if prenda.usos>self.estadisticas.masUsos:
410                         self.estadisticas.masUsos = prenda.usos
411                         self.estadisticas.masUsado = prenda.codigo
412                     if prenda.usos<self.estadisticas.menosUsos:
413                         self.estadisticas.menosUsos = prenda.usos
```

```
414                         self.estadisticas.menosUsado = prenda.codigo
415                     """
416                     if prenda.fecha>self.estadisticas.ultima:
417                         self.estadisticas.ultima = prenda.fecha
418                     if prenda.fecha<self.estadisticas.antigua:
419                         self.estadisticas.antigua = prenda.fecha
420                     """
421             """
422             for codigo in datos:
423                 if codigo != "estadisticas" and datos[codigo]['usos'] == 0:
424                     estadisticas.algo = str(datos[codigo]['codigo'])+";"
425             """
426             self.fb.setField("estadisticas", self.estadisticas.toFB())
427 
428         def mode_estadisticas(self):
429             self.remake_estadisticas()
430 
431 
432 
433     if __name__ == "__main__":
434         print("aqui")
435         main = Main()
436 
437 
438 
439 
```

WEBGRAFIA:

https://medium.com/@askwonder/how-much-information-does-the-human-brain-learn-every-day-92deaad459a6

https://pimylifeup.com/raspberry-pi-rfid-rc522/

https://blog.nuoplanet.com/rfid

https://vicentferrer.com/que-es-rfid-y-como-funciona/#:~:text=La%20Historia%20de%20RFID&text=Esta%20tecnolog%C3%ADa%20(La%20IFF)%20fue,el%20Estado%20Sovi%C3%A9tico%20en%201945

https://www.nxp.com/docs/en/data-sheet/MFRC522.pdf

https://www.rfidinc.com/pub/media/pdfs/13.56_MHz_HF_Data_Sheet_v10.16.pdf