1. El que corre es Python **mibot.py**
2. import logging
3. import mysql.connector
4. from telegram import Update
5. from telegram.ext import (
6. ApplicationBuilder,
7. CommandHandler,
8. MessageHandler,
9. filters,
10. ContextTypes,
11. ConversationHandler,
12. )
13. # Conexión a la base de datos
14. db = mysql.connector.connect(
15. host="localhost",
16. user="root",
17. password="1234",  # Cambiar si usás contraseña
18. database="mi\_base"
19. )
20. # Estados
21. ELEGIR, BUSCAR\_NOMBRE, BUSCAR\_CODIGO, BUSCAR\_ZONA, ELEGIR\_UNIDAD = range(5)
22. # Logs
23. logging.basicConfig(
24. format='%(asctime)s - %(name)s - %(levelname)s - %(message)s',
25. level=logging.INFO
26. )
27. # Menú principal
28. async def start(update: Update, context: ContextTypes.DEFAULT\_TYPE):
29. await update.message.reply\_text(
30. "Hola, soy tu bot de consulta. ¿En qué unidad estás interesada?\n"
31. "1) Buscar por nombre\n"
32. "2) Buscar por código\n"
33. "3) Buscar por zona (provincia) \n"
34. "4) No, ya terminé"
35. )
36. return ELEGIR
37. # Elegir opción
38. async def elegir(update: Update, context: ContextTypes.DEFAULT\_TYPE):
39. opcion = update.message.text.strip()
40. if opcion == "1":
41. await update.message.reply\_text("Ingrese el nombre de la unidad:")
42. return BUSCAR\_NOMBRE
43. elif opcion == "2":
44. await update.message.reply\_text("Ingrese el código de la unidad:")
45. return BUSCAR\_CODIGO
46. elif opcion == "3":
47. await update.message.reply\_text("Ingrese la zona o provincia:")
48. return BUSCAR\_ZONA
49. elif opcion == "4":
50. await update.message.reply\_text("¡Me alegro! Vuelve pronto escribiendo /start para usar cualquiera de las 3 opciones 😊")
51. return ConversationHandler.END
52. else:
53. await update.message.reply\_text("Opción no válida. Intente con 1, 2, 3 o 4.")
54. return ELEGIR
55. # Buscar por nombre
56. async def buscar\_nombre(update: Update, context: ContextTypes.DEFAULT\_TYPE):
57. nombre = update.message.text.strip()
58. cursor = db.cursor()
59. cursor.execute("SELECT \* FROM unidades WHERE nombre LIKE %s", (f"%{nombre}%",))
60. result = cursor.fetchone()
61. if result:
62. codigo, nombre, zona, jefe = result[1], result[2], result[3], result[4]
63. await update.message.reply\_text(
64. f"✅ Información de la unidad {nombre}:\n"
65. f"- Código: {codigo}\n- Zona: {zona}\n- Jefe: {jefe}"
66. )
67. else:
68. await update.message.reply\_text("No se encontró ninguna unidad con ese nombre.")
69. await update.message.reply\_text(
70. "¿Desea hacer otra consulta?\n"
71. "1) Buscar por nombre\n"
72. "2) Buscar por código\n"
73. "3) Buscar por zona\n"
74. "4) No, ya terminé"
75. )
76. return ELEGIR
77. # Buscar por código
78. async def buscar\_codigo(update: Update, context: ContextTypes.DEFAULT\_TYPE):
79. codigo = update.message.text.strip()
80. cursor = db.cursor()
81. cursor.execute("SELECT \* FROM unidades WHERE codigo = %s", (codigo,))
82. result = cursor.fetchone()
83. if result:
84. codigo, nombre, zona, jefe = result[1], result[2], result[3], result[4]
85. await update.message.reply\_text(
86. f"✅ Información del código {codigo}:\n"
87. f"- Unidad: {nombre}\n- Zona: {zona}\n- Jefe: {jefe}"
88. )
89. else:
90. await update.message.reply\_text("No se encontró ninguna unidad con ese código.")
91. await update.message.reply\_text(
92. "¿Desea hacer otra consulta?\n"
93. "1) Buscar por nombre\n"
94. "2) Buscar por código\n"
95. "3) Buscar por zona\n"
96. "4) No, ya terminé"
97. )
98. return ELEGIR
99. # Buscar por zona
100. async def buscar\_zona(update: Update, context: ContextTypes.DEFAULT\_TYPE):
101. zona = update.message.text.strip()
102. cursor = db.cursor()
103. cursor.execute("SELECT \* FROM unidades WHERE zona LIKE %s", (f"%{zona}%",))
104. results = cursor.fetchall()
105. if results:
106. context.user\_data["resultados"] = results
107. mensaje = "📍 Unidades en esa zona:\n"
108. for idx, r in enumerate(results, start=1):
109. mensaje += f"{idx}) {r[2]}\n"
110. mensaje += "Elija la unidad (1, 2, etc):"
111. await update.message.reply\_text(mensaje)
112. return ELEGIR\_UNIDAD
113. else:
114. await update.message.reply\_text("No se encontraron unidades en esa zona.")
115. await update.message.reply\_text(
116. "¿Desea hacer otra consulta?\n"
117. "1) Buscar por nombre\n"
118. "2) Buscar por código\n"
119. "3) Buscar por zona\n"
120. "4) No, ya terminé"
121. )
122. return ELEGIR
123. # Elegir unidad dentro de la zona
124. async def elegir\_unidad(update: Update, context: ContextTypes.DEFAULT\_TYPE):
125. try:
126. indice = int(update.message.text.strip()) - 1
127. datos = context.user\_data["resultados"][indice]
128. codigo, nombre, zona, jefe = datos[1], datos[2], datos[3], datos[4]
129. await update.message.reply\_text(
130. f"✅ Información de {nombre}:\n"
131. f"- Código: {codigo}\n- Zona: {zona}\n- Jefe: {jefe}"
132. )
133. except:
134. await update.message.reply\_text("Opción inválida.")
135. await update.message.reply\_text(
136. "¿Desea hacer otra consulta?\n"
137. "1) Buscar por nombre\n"
138. "2) Buscar por código\n"
139. "3) Buscar por zona\n"
140. "4) No, ya terminé"
141. )
142. return ELEGIR
143. # Cancelar
144. async def cancelar(update: Update, context: ContextTypes.DEFAULT\_TYPE):
145. await update.message.reply\_text("Consulta cancelada.")
146. return ConversationHandler.END
147. # Crear app
148. app = ApplicationBuilder().token("7233780519:AAEzGE8daJpNvFnO6SK0X26wZvJ5b5rDHI0").build()
149. # Conversación
150. conv\_handler = ConversationHandler(
151. entry\_points=[CommandHandler("start", start)],
152. states={
153. ELEGIR: [MessageHandler(filters.TEXT & ~filters.COMMAND, elegir)],
154. BUSCAR\_NOMBRE: [MessageHandler(filters.TEXT & ~filters.COMMAND, buscar\_nombre)],
155. BUSCAR\_CODIGO: [MessageHandler(filters.TEXT & ~filters.COMMAND, buscar\_codigo)],
156. BUSCAR\_ZONA: [MessageHandler(filters.TEXT & ~filters.COMMAND, buscar\_zona)],
157. ELEGIR\_UNIDAD: [MessageHandler(filters.TEXT & ~filters.COMMAND, elegir\_unidad)],
158. },
159. fallbacks=[CommandHandler("cancelar", cancelar)],
160. )
161. app.add\_handler(conv\_handler)
162. app.run\_polling()

* Tengo mi bdd en **worbench**
* Prendo los 2 y mando al telegram **/start**

Ahora para dejarlo activo puedo usar : **render es gratis**

**🌐 Para que tu bot funcione en todos lados (24 hs, desde cualquier lugar):**

Necesito **dos cosas alojadas en un lugar que esté siempre encendido**:

1. **Tu código del bot (mibot.py)**  
   — ahí está la lógica que responde los mensajes de Telegram.
2. **Tu base de datos (mi\_base)**  
   — donde están las unidades, códigos, jefes, etc.

**🔸 Dónde podés alojarlas**

No hace falta que compres una máquina física.  
Podés usar **un servidor virtual en la nube**, por ejemplo:

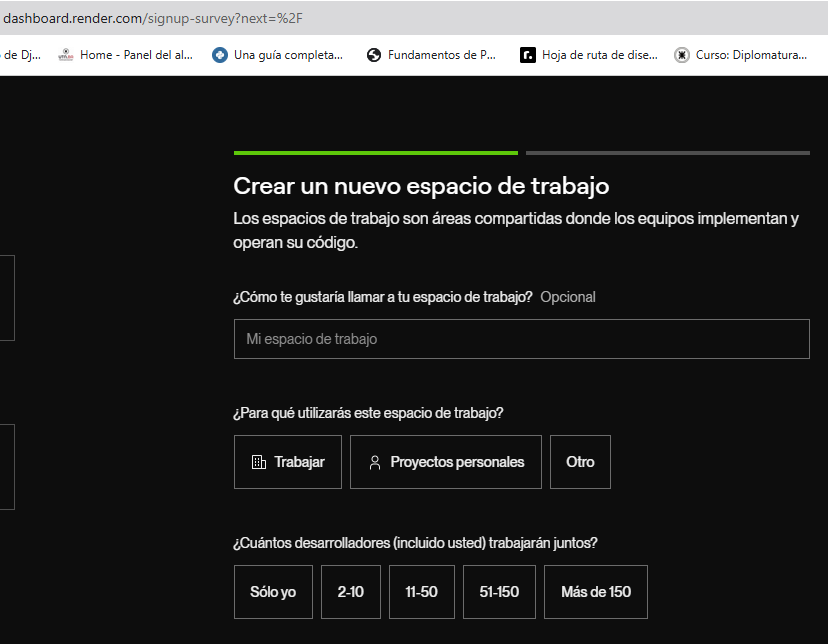
* 🟣 **Render** (gratis y fácil para Python)
* 🟢 **Railway** o **Clever Cloud** (tienen MySQL gratis)
* 🔵 **PythonAnywhere** (barato y estable)

RENDER:

[**https://dashboard.render.com/signup-survey?next=%2F**](https://dashboard.render.com/signup-survey?next=%2F)

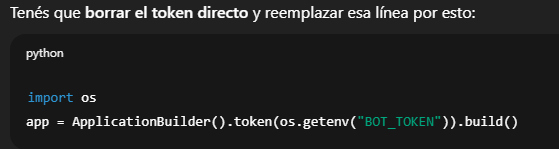
**usuario mail: joan…2016**

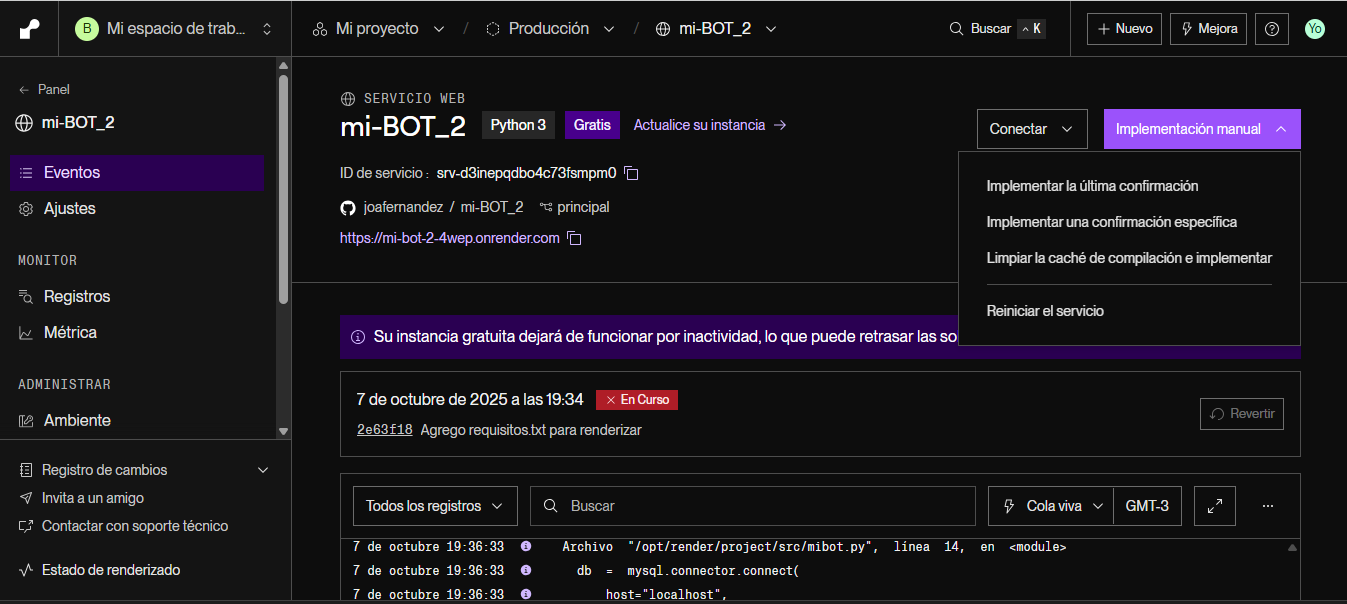
**contraseña: Emilce-30F**



Para render primero 

Oculto y en el código reemplazo por





Ahora tengo q migrar de worbench la bdd a una nube:

BUENO TUVE Q HECER UN QUILOMBO

<https://dashboard.render.com/web/srv-d3inepqdbo4c73fsmpm0/logs>

<https://railway.com/project/ecdd407f-674f-4b18-bd4d-cdab2722d2c6/service/b2369c73-76ad-4ccf-b089-3a092887ca94/database?environmentId=11e9afae-75c7-4bc3-8279-a3d28bb40e26&state=table&table=unidades>

