

```

6  ✓ class Database:
7  ✓     def __init__(self, database, collection):
8      self.connect(database, collection)
9
10 ✓     def connect(self, database, collection):
11 ✓         try:
12             connectionString = "mongodb+srv://root:root@cluster0.ddrq2fn.mongodb.net/"
13             self.clusterConnection = pymongo.MongoClient(
14                 connectionString,
15                 tlsAllowInvalidCertificates=True
16             )
17             self.db = self.clusterConnection[database]
18             self.collection = self.db[collection]
19             print("Conectado ao banco de dados com sucesso!")
20 ✓         except Exception as e:
21             print(e)
22
23 ✓     def resetDatabase(self):
24 ✓         try:
25             self.db.drop_collection(self.collection)
26             self.collection.insert_many(dataset)
27             print("Banco de dados resetado com sucesso!")
28 ✓         except Exception as e:
29             print(e)

```

```

4  class MotoristaDAO:
5      def __init__(self, database):
6          self.db = database
7
8      def create_driver(self, nota: int, corridas: int):
9          try:
10             res = self.db.collection.insert_one({"nota": nota, "corridas": corridas})
11             print(f"Driver created with id: {res.inserted_id}")
12             return res.inserted_id
13         except Exception as e:
14             print(f"An error occurred while creating driver: {e}")
15             return None
16
17     def read_driver_by_id(self, id: str):
18         try:
19             res = self.db.collection.find_one({"_id": ObjectId(id)})
20             print(f"Driver found: {res}")
21             return res
22         except Exception as e:
23             print(f"An error occurred while reading driver: {e}")
24             return None
25
26     def update_driver(self, id: str, nota: int, corridas: int):
27         try:
28             res = self.db.collection.update_one({"_id": ObjectId(id)}, {"$set": {"nota": nota, "corridas": corridas}})
29             print(f"driver updated: {res.modified_count} document(s) modified")
30             return res.modified_count
31         except Exception as e:
32             print(f"An error occurred while updating driver: {e}")
33             return None
34
35     def delete_driver(self, id: str):
36         try:
37             res = self.db.collection.delete_one({"_id": ObjectId(id)})
38             print(f"driver deleted: {res.deleted_count} document(s) deleted")
39             return res.deleted_count
40         except Exception as e:
41             print(f"An error occurred while deleting driver: {e}")
42             return None

```

```

9  #Classes
10 class Passageiro:
11     def __init__(self, nome: str, documento: str):
12         self.nome = nome
13         self.documento = documento
14
15 class Corrida:
16     def __init__(self, nota: int, distancia: float, valor: float, passageiro: Passageiro):
17         self.nota = nota
18         self.distancia = distancia
19         self.valor = valor
20         self.passageiro = Passageiro
21
22 class Motorista:
23     def __init__(self, corridas: [Corrida], nota: int):
24         self.corridas = Corrida
25         self.nota = nota

```

relatorio_avaliativo_01.motoristas

1 1
DOCUMENTS INDEXES

Documents Aggregations Schema Indexes Validation

Filter ⓘ ⓘ Type a query: { field: 'value' }

Explain

Reset

Find

↩

Options ▶

➕ ADD DATA ▼

📄 EXPORT DATA ▼

1 - 1 of 1

↺

↻

↺

⋮

🔍

🏠

```

_id: ObjectId('651daf061d21bebb4539cd29')
Corrida: Object
  nota: 0
  Passageiro: Object
    nome: ""
    documento: ""
    distancia: 0
    valor: 0

```

```

20 class MotoristaCRUD(SimpleCLI):
21     def __init__(self, driver_model):
22         super().__init__()
23         self.driver_model = driver_model
24         self.add_command("create", self.create_driver)
25         self.add_command("read", self.read_driver)
26         self.add_command("update", self.update_driver)
27         self.add_command("delete", self.delete_driver)
28
29     def create_driver(self):
30         nota = int(input("Entre com a nota:"))
31         corrida = input("Entre com a corrida:")
32         self.driver_model.create_driver(nota, corrida)
33
34     def read_driver(self):
35         id = input("Enter the id: ")
36         driver = self.driver_model.read_driver_by_id(id)
37         if driver:
38             print(f"Titulo: {driver['titulo']}")
39             print(f"Autor: {driver['autor']}")
40             print(f"Ano: {driver['ano']}")
41             print(f"Preco: {driver['preco']}")
42
43     def update_driver(self):
44         id = input("Entre com o id: ")
45         titulo = input("Entre com o novo titulo: ")
46         autor = input("Entre com o novo autor: ")
47         ano = int(input("Entre com o novo ano: "))
48         preco = float(input("Entre com novo o preco: "))
49         self.driver_model.update_driver(id, titulo, autor, ano, preco)
50
51     def delete_driver(self):
52         id = input("Entre com o id: ")
53         self.driver_model.delete_driver(id)
54
55     def run(self):
56         print("Bem vindo ao driverCRUD!")
57         print("Comandos disponiveis: create, read, update, delete, quit")
58         super().run()

```

The dashboard displays two charts. The top chart, titled **_id**, has a y-axis labeled **objectid** and an x-axis showing days of the week (S, M, T, W, T, F, S) and time intervals (0:00, 6:00, 12:00, 18:00, 23:00). The bars show a peak on Wednesday. The bottom chart, titled **Corrida**, has a y-axis labeled **document** and an x-axis showing **distancia** (double) and **nota** (double). The bars show a peak on Wednesday.