Informe Turismo

Profesor: Ing. Fernando Zapata
Integrantes: Luciano Toneatti, Ian Olmedo, Reynier Lopez, Camila Choque y Martin Navarro

1) **Planteo de Necesidades**
   Nuestro objetivo es desarrollar un sistema centralizado que permita a los usuarios gestionar todos los aspectos de su viaje en un solo sitio web. Los usuarios registrados podrán acceder fácilmente a servicios clave, como la reserva de alojamientos y vehículos, la compra de boletos de vuelo, y la exploración de destinos.

   Este sistema debe facilitar la planificación personalizada del viaje, permitiendo crear itinerarios completos y listas de destinos de interés. También se busca que el sitio web ofrezca métodos de pago seguros y diversos, un soporte al cliente eficiente y una interfaz accesible y adaptada para cualquier dispositivo. En resumen, la plataforma no solo cubrirá las necesidades básicas de la organización, sino que también fomentará una experiencia de viaje enriquecedora, confiable y accesible para todo tipo de usuarios.

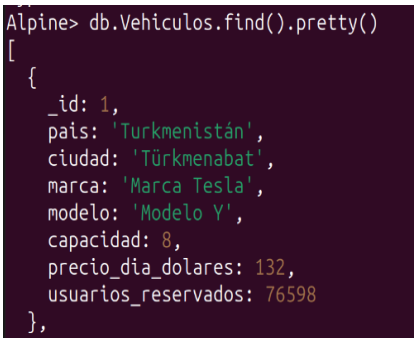
2) **Modelado estructura**
   https://www.figma.com/design/UcPFhsXNhmeIVClqH3FCyJ/DBDII?node-id=0-1&t=ypLMqF8ldu1pfz9e-0

3) **Modelo de implementación**
   El Proyecto lo tenemos dividido en 6 diferentes “Collections” llamadas:
   - Alojamientos
   - Vuelos
   - Vehículos
   - Home
   - Atracciones
   - Usuarios

```
Alpine> show collections
Alojamientos
Atracciones
Home
Usuario
Vehiculos
Vuelos
```

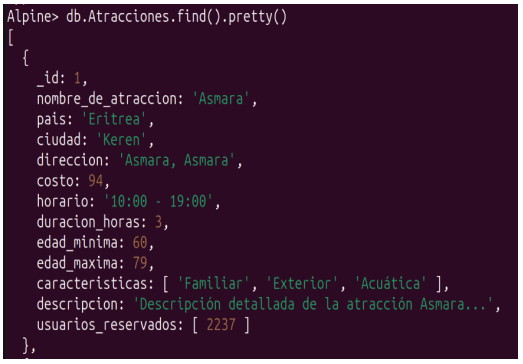
Además agregamos “Relaciones” llamadas:

- Un Usuario — Varios Vehículos
  En la colección Vehículos hay un campo llamado “usuarios_reservados” en el que se guarda el Usuario que reservó el vehículo.



- Varios Usuarios — Varios Vuelos
  En la colección Usuarios hay un campo llamado “vuelos_reservados” en el que se guardan todos los vuelos que reservó el usuario. Y en la colección Vuelos hay un campo llamado “pasajeros” en los que se guardan todos los Usuarios que toman ese vuelo.

- Varios Usuarios — Varias Atracciones
  En la colección Usuarios hay un campo llamado “atracciones_reservadas” en el que se guardan todas las atracciones que reservó el usuario. Y en la colección Atracciones hay un campo llamado “usuarios_reservados” en los que se guardan todos los Usuarios que toman esa atracción.



- Varios Usuarios — Varios Alojamientos
  En la colección Usuarios hay un campo llamado “alojamientos_reservados” en el que se guardan todas los alojamientos que reservó el usuario. Y en la colección Alojamientos hay un campo llamado “usuarios_reservados” en los que se guardan todos los Usuarios que toman ese alojamiento.

4) Carga de datos(SCRIPTS)

Para cargar una gran cantidad de datos utilizamos Scripts hechos en Python, uno para cada colección. Con esto nos permitirá poblar la base de datos con millones de registros en menos de 2 minutos y de esta forma podemos probar su funcionamiento y velocidad.

Usuarios:

```python
paises = list(paises_ciudades.keys())


try:
    usuarios_bulk = []
    for i in range(500000):
        pais = random.choice(paises)
        ciudad = random.choice(paises_ciudades[pais])
        usuarios_bulk.append({
            "_id": i+1,
            "nombre": random.choice(nombres),
            "apellido": random.choice(apellidos),
            "DNI": random.randint(1000000, 9999999),
            "correo electronico": f"usuario{i}@example.com",
            "contraseña": f"{random.choice(['TPL60XZY9YW', 'ABC123XYZ', 'MYPASSWORD'])}",
            "pais": pais,
            "ciudad": ciudad,
            "nacimiento": datetime(random.randint(1950, 2010), random.randint(1, 12), random.randint(1, 28)),
            "alojamientos_reservados": [random.randint(1,1000000) for _ in range(random.randint(0,10))],
            "atraciones_reservadas": [random.randint(1,1500000) for _ in range(random.randint(0,10))],
            "vuelos_reservados": [random.randint(1,500000) for _ in range(random.randint(0,10))]
        })

        if len(usuarios_bulk) == 500000:
            db.Usuario.insert_many(usuarios_bulk)
            usuarios_bulk = []

            # Mensaje de progreso cada 100,000 usuarios insertados
            if i % 100000 == 0:
                print(f"{i} usuarios insertados...")

    if usuarios_bulk:
        db.Usuario.insert_many(usuarios_bulk)

    print("Colección Usuario poblada con 1,000,000 documentos.")

except Exception as e:
    print(f"Error durante la inserción: {e}")
```

Alojamientos:

```python
client = MongoClient('mongodb://localhost:27017/')
db = client['Alpine']

try:

    # Poblar colección Alojamiento (10 documentos)
    alojamiento_bulk = []
    for i in range(1000000):
        pais = random.choice(list(paises_ciudades.keys()))
        ciudad = random.choice(paises_ciudades[pais])
        direccion = random.choice(paises_direcciones[pais])
        alojamiento_bulk.append({
            "_id": i + 1,
            "pais": pais,
            "ciudad":ciudad,
            "direccion": direccion,
            "duracion_dias": random.randint(1, 30),
            "estacionamiento": random.choice([True, False]),
            "precio_dolares": random.randint(33, 527),
            "telefono": f"{random.randint(1000000000, 9999999999)}",
            "descripcion": f"Descripción detallada del alojamiento {direccion}...",
            "usuarios_reservados": [random.randint(1,500000) for _ in range(random.randint(1,10))]
        })

        if len(alojamiento_bulk) == 1000000:
            db.Alojamientos.insert_many(alojamiento_bulk)
            alojamiento_bulk = []

    if alojamiento_bulk:
        db.Alojamientos.insert_many(alojamiento_bulk)


except Exception as e:
    print(f"Error durante la inserción: {e}")
```

Atracciones:

```python
client = MongoClient('mongodb://localhost:27017/')
db = client['Alpine']

try:
    atracciones_bulk = []
    for i in range(1500000):
        pais = random.choice(list(paises_ciudades.keys()))
        ciudad = random.choice(paises_ciudades[pais])
        atraccion = random.choice(paises_atracciones[pais])
        direccion = random.choice(paises_direcciones[pais])
        atracciones_bulk.append({
            "_id": i+1,
            "nombre_de_atraccion": atraccion,
            "pais": pais,
            "ciudad": ciudad,
            "direccion": direccion,
            "costo": random.randint(8, 112),
            "horario": f"{random.randint(9, 17)}:00 - {random.randint(18, 22)}:00",
            "duracion_horas": random.randint(1, 5),
            "edad_minima": random.randint(8, 90),
            "edad_maxima": random.randint(8, 90),
            "caracteristicas": random.sample(caracteristicas, random.randint(1, len(caracteristicas))),
            "descripcion": f"Descripción detallada de la atracción {atraccion}...",
            "usuarios_reservados": [random.randint(1,500000) for _ in range(random.randint(1,10))]
        })

        if len(atracciones_bulk) == 1500000:
            db.Atracciones.insert_many(atracciones_bulk)
            atracciones_bulk = []

    if atracciones_bulk:
        db.Atracciones.insert_many(atracciones_bulk)

    print("Colección Atracciones poblada con 10,000,000 documentos.")

except Exception as e:
    print(f"Error durante la inserción: {e}")
```

Vehículos:

```python
client = MongoClient('mongodb://localhost:27017/')
db = client['Alpine']


paises = list(paises_ciudades.keys())


try:
    vehiculos_bulk = []
    for i in range(500000):
        pais = random.choice(paises)
        ciudad = random.choice(paises_ciudades[pais])
        vehiculos_bulk.append({
            "_id": i+1,
            "pais": pais,
            "ciudad": ciudad,
            "marca": f"Marca {random.choice(['Toyota', 'Ford', 'Honda', 'Tesla', 'BMW'])}",
            "modelo": f"Modelo {random.choice(['X', 'Y', 'Z', 'GT', 'S'])}",
            "capacidad": random.randint(2, 9),
            "precio_dia_dolares": random.randint(30, 500),
            "usuarios_reservados": random.randint(1,500000)
        })

        if len(vehiculos_bulk) == 500000:
            db.Vehiculos.insert_many(vehiculos_bulk)
            vehiculos_bulk = []

    if vehiculos_bulk:
        db.Vehiculos.insert_many(vehiculos_bulk)

    print("Colección Vehiculos poblada con 10,000 documentos.")

except Exception as e:
    print(f"Error durante la inserción: {e}")
```

Home:

```
client = MongoClient('mongodb://localhost:27017/')
db = client['Alpine']

try:
    home_bulk = []
    categorias = ["Mas Visitados", "Mejores Atracciones", "Mejores Alojamientos", "Ofertas"]
    for i in range(4):
        home_bulk.append({
            "nombre": categorias[i],
            "imagen": [f" /home/usser/assets/img{i}-{j}.png" for j in range(15)]
        })

        if len(home_bulk) == 4:
            db.Home.insert_many(home_bulk)
            home_bulk = []

    if home_bulk:
        db.Home.insert_many(home_bulk)

    print("Colección Home poblada con 5 documentos.")

except Exception as e:
    print(f"Error durante la inserción: {e}")
```

Vuelos:

```
client = MongoClient('mongodb://localhost:27017/')
db = client['Alpine']

paises = list(paises_ciudades.keys())

try:
    vuelos_bulk = []
    for i in range(500000):
        pais = random.choice(paises)
        ciudad = random.choice(paises_ciudades[pais])
        aeropuerto = random.choice(paises_aeropuertos[pais])
        origen = random.choice(paises)
        destino = random.choice(paises)
        while origen == destino:
            destino = random.choice(pais)
        fecha_salida = datetime.now() + timedelta(days=random.randint(1, 365))
        fecha_llegada = fecha_salida + timedelta(hours=random.randint(1, 24))
        vuelos_bulk.append({
            "_id": i+1,
            "aeropuerto": aeropuerto,
            "pais": pais,
            "ciudad": ciudad,
            "origen": origen,
            "destino": destino,
            "fecha_salida": fecha_salida.strftime("%Y-%m-%d"),
            "fecha_llegada": fecha_llegada.strftime("%Y-%m-%d"),
            "hora_salida": fecha_salida.strftime("%H:%M"),
            "hora_llegada": fecha_llegada.strftime("%H:%M"),
            "aerolinea": random.choice(aerolineas),
            "precio_dolares": random.randint(20, 1293),
            "asientos_disponibles": random.randint(0, 213),
            "pasajeros": [random.randint(1,500000) for _ in range(random.randint(1,10))]
        })
        if len(vuelos_bulk) == 500000:
            db.Vuelos.insert_many(vuelos_bulk)
            vuelos_bulk = []

    if vuelos_bulk:
        db.Vuelos.insert_many(vuelos_bulk)
except Exception as e:
    print(f"Error durante la inserción: {e}")
```

5) **Elaboración de consultas avanzadas**

Hemos realizado unas 20 consultas, algunas son más sencillas y otras más complejas. Todas las consultas se detallan a continuación:

1. Usuarios que han reservado alojamientos

```bash
db.Alojamientos.aggregate([
  {
    $lookup: {
      from: "Usuario",
      localField: "usuarios_reservados",
      foreignField: "_id",
      as: "usuarios_reservados_info"
    }
  },
  {
    $unwind: "$usuarios_reservados_info"
  },
  {
    $project: {
      pais: 1,
      ciudad: 1,
      direccion: 1,
      nombre_usuario: "$usuarios_reservados_info.nombre",
      apellido_usuario: "$usuarios_reservados_info.apellido",
      DNI: "$usuarios_reservados_info.DNI"
    }
  }
])
```

2. Usuarios que han reservado atracciones

```
db.Atracciones.aggregate([
  {
    $lookup: {
      from: "Usuario",
      localField: "usuarios_reservados",
      foreignField: "_id",
      as: "usuarios_reservados_info"
    }
  },
  {
    $unwind: "$usuarios_reservados_info"
  },
  {
    $project: {
      nombre_atraccion: 1,
      pais: 1,
      ciudad: 1,
      nombre_usuario: "$usuarios_reservados_info.nombre",
      apellido_usuario: "$usuarios_reservados_info.apellido",
      DNI: "$usuarios_reservados_info.DNI"
    }
  }
])
```

3. Usuarios que han reservado vuelos

```
db.Vuelos.aggregate([
  {
    $lookup: {
      from: "Usuario",
      localField: "pasajeros",
      foreignField: "_id",
      as: "usuarios_reservados_info"
    }
  },
  {
    $unwind: "$usuarios_reservados_info"
  },
  {
    $project: {
      aeropuerto: 1,
      pais: 1,
      ciudad: 1,
      origen: 1,
      destino: 1,
      fecha_salida: 1,
      nombre_usuario: "$usuarios_reservados_info.nombre",
      apellido_usuario: "$usuarios_reservados_info.apellido",
      correo_usuario: "$usuarios_reservados_info.correo electronico"
    }
  }
])
```

4. Usuarios que han reservado vehículos

```
db.Vehiculos.aggregate([
  {
    $lookup: {
      from: "Usuario",
      localField: "usuarios_reservados",
      foreignField: "_id",
      as: "usuarios_reservados_info"
    }
  },
  {
    $unwind: "$usuarios_reservados_info"
  },
  {
    $project: {
      marca: 1,
      modelo: 1,
      capacidad: 1,
      precio_dia_dolares: 1,
      nombre_usuario: "$usuarios_reservados_info.nombre",
      apellido_usuario: "$usuarios_reservados_info.apellido",
      correo_usuario: "$usuarios_reservados_info.correo electronico"
    }
  }
])
```

5. Consulta para alojamientos con duracion mayor a 7 dias

```
db.Alojamientos.find({"duracion_dias": {"$gt": 7}})
```

6. Consulta para encontrar todos los alojamientos en Arabia Saudita

```
db.Alojamientos.find({"pais": "Arabia Saudita"})
```

7. Consulta para encontrar todas las atracciones con un cosoto menor a 50 dolares

```
db.Atracciones.find({"costo": {"$lt": 50}})
```

8. Consulta para encontrar usuarios de República Checa

```
db.Usuario.find({"pais": "República Checa"})
```

9. Consulta para encontrar todos los vehículos con una capacidad mayor a 5 personas

```
db.Vehiculos.find({"capacidad": {"$gt": 5}})
```

10. Consulta para encontrar todos los vuelos desde España a Francia

```
db.Vuelos.find({"origen": "España", "destino":"Francia"})
```

11. Consulta para encontrar todas las atracciones familiares

```Bash
db.Atracciones.find({"caracteristicas": "Familiar"})
```

12. Consulta para encontrar todos los usuarios que el nombre empiece con la letra "A"

```Bash
db.Usuario.find({"nombre": {"$regex": "^A"}})
```

13. Consulta para encontrar el promedio de precio de los alojamientos por país

```
db.Alojamientos.aggregate([{"$group": {"_id": "$pais", "promedio_precio": {"$avg": "$precio_dolares"}}}])
```

14. Consulta para encontrar todos los alojamientos con estacionamiento y un precio menor a 300 dólares

```
db.Alojamientos.find({"estacionamiento": true, "precio_dolares": {"$lt": 300}})
```

15. Consulta para encontrar todas las atracciones familiares y nocturnas en Madrid

```
db.Atracciones.find({"ciudad": "Madrid", "caracteristicas": {"$all": ["Familiar", "Nocturna"]}})
```

16. Consulta para encontrar todos los vehículos con una capacidad mayor a 4 personas y un precio por día entre 150 y 250 dólares

```
db.Vehiculos.find({"capacidad": {"$gt": 4}, "precio_dia_dolares": {$gte: 150, $lte: 250}})
```

17. Consulta para encontrar todas las atracciones nocturnas y acuáticas en Francia con un costo menor a 80 dólares

```
db.Atracciones.find({"pais": "Francia", "caracteristicas": {"$all": ["Nocturna", "Acuática"]}, "costo": {"$lt": 80}})
```

18. Consulta para encontrar todas las atracciones nocturnas y acuáticas en Italia con un costo menor a 70 dólares

```
db.Atracciones.find({"pais": "Italia", "caracteristicas": {"$all": ["Nocturna", "Acuática"]}, "costo": {"$lt": 70}})
```

19. Consulta para encontrar todos los alojamientos con estacionamiento en Paraguay

```
db.Alojamientos.find({"pais": "Paraguay", "estacionamiento": true})
```

20. Consulta Obtener Información del Usuario y sus Reservas

```bash
db.Usuario.aggregate([
  {
    $match: { _id: 20 }
  },
  {
    $lookup: {
      from: "Vuelos",
      localField: "vuelos_reservados",
      foreignField: "_id",
      as: "vuelos_reservados_info"
    }
  },
  {
    $lookup: {
      from: "Atracciones",
      localField: "atraciones_reservadas",
      foreignField: "_id",
      as: "atraciones_reservadas_info"
    }
  },
  {
    $lookup: {
      from: "Alojamientos",
      localField: "alojamientos_reservados",
      foreignField: "_id",
      as: "alojamientos_reservados_info"
    }
  }
])
```

6) Indización

En todas las colecciones hemos añadido al menos un índice, a continuación se mostraran todos los índices añadidos:

```
Alpine> db.Alojamientos.getIndexes()
[
  { v: 2, key: { _id: 1 }, name: '_id_' },
  { v: 2, key: { pais: 1 }, name: 'consulta_por_paises' },
  { v: 2, key: { precio_dolares: 2 }, name: 'consulta_por_precio' }
]
```

```
Alpine> db.Atracciones.getIndexes()
[
  { v: 2, key: { _id: 1 }, name: '_id_' },
  { v: 2, key: { pais: 3 }, name: 'consulta_por_paises_atracciones' },
  { v: 2, key: { costo: 4 }, name: 'consulta_por_costo_atracciones' },
  { v: 2, key: { ciudad: 5 }, name: 'consulta_por_ciudad_atracciones' }
]
```

```
Alpine> db.Usuario.getIndexes()
[
  { v: 2, key: { _id: 1 }, name: '_id_' },
  { v: 2, key: { apellido: 6 }, name: 'consulta_por_apellidos' }
]
```

```
Alpine> db.Vehiculos.getIndexes()
[
  { v: 2, key: { _id: 1 }, name: '_id_' },
  { v: 2, key: { pais: 7 }, name: 'consulta_por_pais_vehiculo' }
]
```

```
Alpine> db.Vuelos.getIndexes()
[
  { v: 2, key: { _id: 1 }, name: '_id_' },
  { v: 2, key: { pais: 8 }, name: 'consulta_por_pais_vuelos' }
]
```

```
Alpine> db.Home.getIndexes()
[ { v: 2, key: { _id: 1 }, name: '_id_' } ]
```

7) **Backup**

Exportación de Colecciones:

mongoexport -d “BaseDeDatos” -c “Colección” -o “ArchivoDeSalida”

mongoimport –db “BaseDeDatos” –collection “NombreColeccion” –file “NombreArchivo”

mongodump –db “BaseDeDatos” –out “RutaDeSalida”