Comandos realizados en clases 27-03-2025:

Para agregar usuarios a nuestra bbdd y almacenar los ids en nuestra variable usuarios.

```
const usuarios = db.usuarios.insertMany([
  { nombre: “Carlos Pérez”, correo: “carlos@mail.com},
  { nombre: "María González", correo: "maria@mail.com" },
  { nombre: "José Fernández", correo: "jose@mail.com" },
  { nombre: "Ana López", correo: "ana@mail.com" },
  { nombre: "Daniel Rojas", correo: "daniel@mail.com" }
]).insertedIds;
```

Para agregar tableros en nuestra bbdd y almacenar los ids en nuestra variable tableros. Como tenemos los ids en usuarios, podemos relacionarlos de manera directa al crear estos nuevos datos.

```
const tableros = db.tableros.insertMany([
  { nombre: "Desarrollo Web", creadoPor: usuarios[0] },
  { nombre: "Marketing Digital", creadoPor: usuarios[1] },
  { nombre: "Gestión de Proyectos", creadoPor: usuarios[2] }
]).insertedIds;
```

Para agregar listas en nuestra bbdd… :

```
const listas = db.listas.insertMany([
  { nombre: "Pendientes", tableroId: tableros[0] },
  { nombre: "En Progreso", tableroId: tableros[0] },
  { nombre: "Finalizado", tableroId: tableros[1] },
  { nombre: "Ideas", tableroId: tableros[2] }
]).insertedIds;
```

Para insertar tarjetas… :

```
const tarjetas = db.tarjetas.insertMany([
  { titulo: "Configurar servidor", listaId: listas[0], usuariosAsignados: [usuarios[0], usuarios[2]],
    prioridad: "alta", fechaVencimiento: new Date("2024-04-10") },
  { titulo: "Diseñar página web", listaId: listas[1], usuariosAsignados: [usuarios[1]], prioridad: "media",
    fechaVencimiento: new Date("2024-04-12") },
  { titulo: "Revisión de UX/UI", listaId: listas[2], usuariosAsignados: [usuarios[3]], prioridad: "baja",
    fechaVencimiento: new Date("2024-04-20") }
]).insertedIds;
```

Para insertar comentarios… :

```
db.comentarios.insertMany([
  { texto: "No olvides configurar HTTPS", tarjetaId: tarjetas[0], usuarioId: usuarios[1], fechaCreacion:
    new Date() },
  { texto: "El diseño se ve muy bien", tarjetaId: tarjetas[1], usuarioId: usuarios[2], fechaCreacion: new
    Date() },
  { texto: "Añadir más detalles", tarjetaId: tarjetas[2], usuarioId: usuarios[3], fechaCreacion: new Date()
}]);
```

Creando colección con timeseries para almacenar nuestros registros de actividad:

```
db.createCollection("registros_actividad", {
  timeseries: {
    timeField: "fecha",
    metaField: "tarjetaId",
    granularity: "minutes"
  }
});
```

- **timeField:** especifica el campo en el que se almacenan los valores de tiempo. En nuestro caso, sería el campo de fecha.
- **metaField:** especifica un campo que agrupa los eventos relacionados entre sí. En este caso, utilizamos *tarjetaId*.
- **granularity:** define la granularidad de las fechas. Puede ser *seconds, minutes, hours*, dependiendo de la precisión que necesitemos.

Finalmente insertamos datos en nuestra colección registros_actividad…, pero antes, recuperaremos los valores de usuarios y tarjetas para poder ingresar y relacionar los datos directamente:

```
const usuarios = db.usuarios.find().toArray();
const tarjetas = db.tarjetas.find().toArray();

db.registros_actividad.insertMany([
  { accion: "Creó tarjeta", tarjetaId: tarjetas[0]._id, usuarioId: usuarios[0]._id, fecha: new Date() },
  { accion: "Asignó usuario a tarjeta", tarjetaId: tarjetas[1]._id, usuarioId: usuarios[1]._id, fecha: new
    Date() },
  { accion: "Cambiado estado a 'En Progreso'", tarjetaId: tarjetas[2]._id, usuarioId: usuarios[2]._id,
    fecha: new Date() },
  { accion: "Creó tarjeta", tarjetaId: tarjetas[3]._id, usuarioId: usuarios[0]._id, fecha: new Date() },
  { accion: "Asignó usuario a tarjeta", tarjetaId: tarjetas[0]._id, usuarioId: usuarios[2]._id, fecha: new
    Date() },
  { accion: "Cambiado estado a 'Completada'", tarjetaId: tarjetas[1]._id, usuarioId: usuarios[1]._id,
    fecha: new Date() },
  { accion: "Cambiado estado a 'En Progreso'", tarjetaId: tarjetas[3]._id, usuarioId: usuarios[0]._id,
    fecha: new Date() },
  { accion: "Creó tarjeta", tarjetaId: tarjetas[2]._id, usuarioId: usuarios[1]._id, fecha: new Date() },
  { accion: "Asignó usuario a tarjeta", tarjetaId: tarjetas[3]._id, usuarioId: usuarios[2]._id, fecha: new
    Date() },
  { accion: "Cambiado estado a 'Completada'", tarjetaId: tarjetas[0]._id, usuarioId: usuarios[0]._id,
    fecha: new Date() }
]);
```

Consultas realizadas en clases

Obtener todos los usuarios:

db.usuarios.find()

Obtener un usuario por su ID:

db.usuarios.findOne({ _id: ObjectId(“ID_DEL_USUARIO”) })

Buscar tarjetas con prioridad “alta”:

db.tarjetas.find({ prioridad: “alta” })

Buscar tarjetas asignadas a un usuario específico:

db.tarjetas.find({ usuariosAsignados: ObjectId(“ID_DEL_USUARIO”) })

Contar cuántas actividades ha hecho cada usuario:

```
db.registros_actividad.aggregate([
  { $group: { _id: “$usuarioId”, totalActividades: { $sum: 1 } } }
])
```

Complejizamos la consulta anterior agregando más etapas para traer los datos de los usuarios:

```
db.registros_actividad.aggregate([
  {
     $group: {
       _id: “$usuarioId”,
       totalActividades: { $sum: 1 }
     }
  },
  {
    $lookup: {
       from: “usuarios”,
       localField: “_id”,
       foreignField: “_id”,
       as: “usuario”
     }
  },
  {
     $project: {
       name: { $arrayElemAt: [“$usuario.nombre”, 0] },
       totalActividades: 1
     }
])
```