

# Dia 3: Programació C# amb MongoDB

Xavier Sala Pujolar



# MongoDB

{  
Universitat  
de Girona  
}

Abril 2021



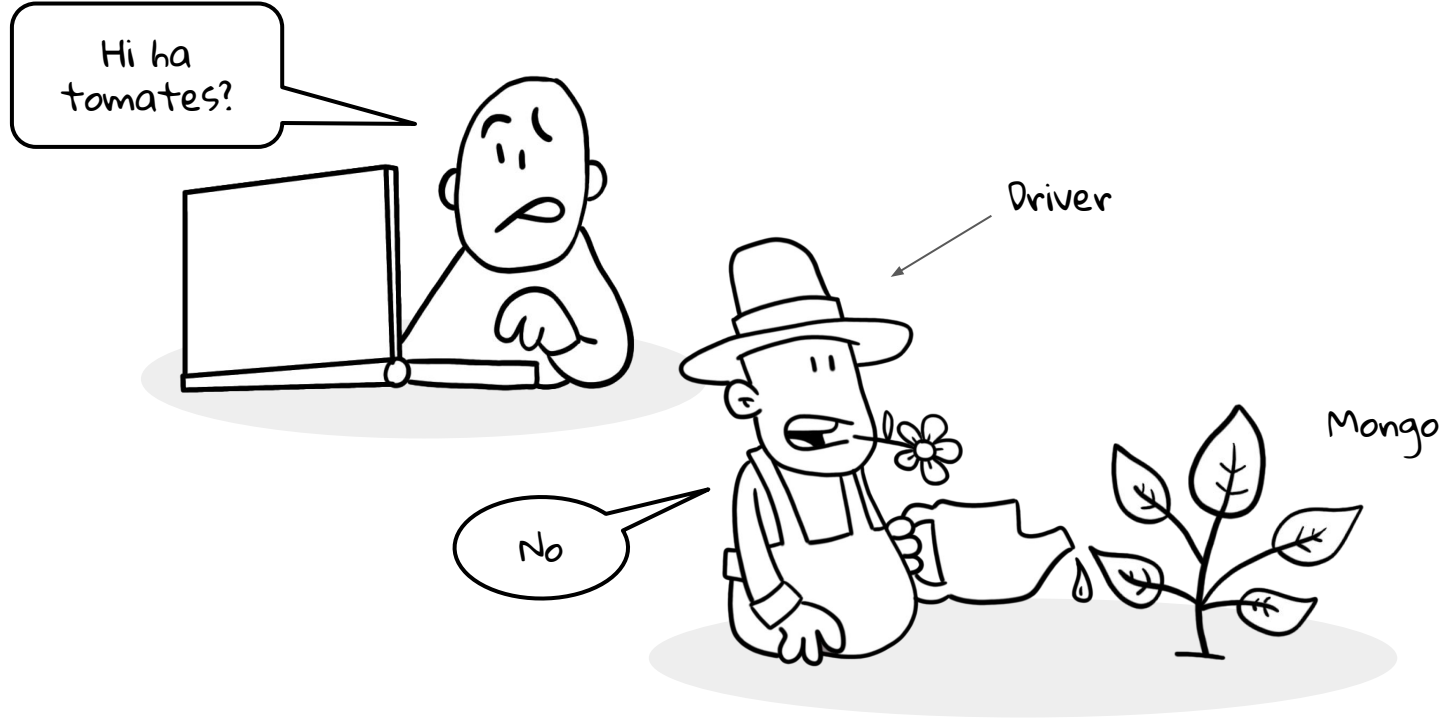
MongoDb disposa de llibreries d'accés oficials per d'una gran quantitat de llenguatges de programació:

*C, C++, C#, Go, Java, Node.js, PHP, Python, Ruby, Rust, Scala, Swift*

També n'hi ha de “Community”

# Accés des de llenguatges de programació

---



# Driver

---

Només cal afegir  
els paquets nuGet  
que calen

NuGet



MongoDB.Driver

MongoDB.Bson

MongoDB.Driver.GridFs

# Connexió amb el Servidor

---

```
var client = new MongoClient();
```

```
mongodb://username:password@host:port/defaultdb
```

```
mongodb://192.168.0.1, mongodb://192.168.0.2
```

# Objectes Base de Dades

---

```
var client = new MongoClient();  
  
var db =  
    client.GetDatabase("vendes");
```



No cal que la base de  
dades existeixi,

# Col·leccions

---

```
var client = new MongoClient();  
  
var db =  
    client.GetDatabase("vendes");  
  
var col =  
    db.GetCollection<BsonDocument>("users");
```



No cal que la  
col·lecció existeixi

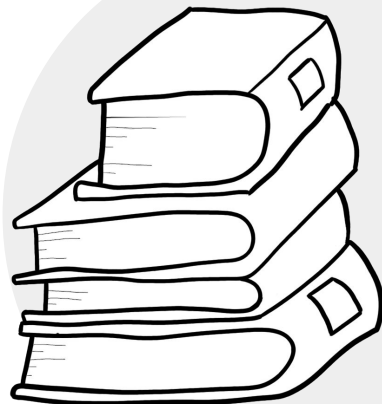
# BsonDocument

---

Per treballar  
amb dades amb  
estructura  
variable

LazyBsonDocument

RawBsonDocument



BsonDocument



```
{  
  nom: "Pere",  
  cognoms: "Pi",  
  edat: 40,  
  adreça: {  
    carrer: "Nou",  
    municipi: "Girona"  
  },  
  idiomes: [ "català", "anglès", "francès" ]  
}
```



```
var idiomes = new List<String> {  
    "català", "anglès", "francès"  
};  
  
var doc = new BsonDocument()  
    .Add("nom", "Pere")  
    .Add("cognoms", "Pi")  
    .Add("Edat", 40)  
    .Add("adreça", new BsonDocument()  
        .Add("carrer", "Nou")  
        .Add("municipi", "Girona")  
    )  
    .Add("idiomes", new BsonArray(idiomes));
```

Forma "fàcil" de  
definir documents  
BSON

```
{  
  nom: "Pere",  
  cognoms: "Pi",  
  edat: 40,  
  adreça: {  
    carrer: "Nou",  
    municipi: "Girona"  
  },  
  idiomes: [ "català", "anglès", "francès" ]  
}
```



```
var doc = new BsonDocument  
{  
  { "nom", "Pere" },  
  { "cognoms", "Pi" },  
  { "edat", 40 },  
  { "adreça" : new BsonDocument  
    {  
      {"carrer", "Nou"},  
      {"municipi", "Girona"}  
    }  
  },  
  { "idiomes", new BsonArray  
    {  
      "català", "anglès", "francès"  
    }  
  },  
};
```



Es poden convertir  
els valors als tipus  
corresponent



```
var doc = new BsonDocument
{
    { "nom", "Pere" },
    { "edat", 40 },
};

Console.WriteLine(doc["nom"]);

var nom = doc["nom"].AsString;
var edat = doc["edat"].AsInt32;

Assert.True(doc.Contains("nom"));
Assert.True(doc.ContainsValue("Pere"));
```

# Operacions

---

Els noms dels mètodes  
solen coincidir amb el  
seu equivalent en la  
consola

Find  
FindOne InsertMany  
InsertOne  
UpdateMany UpdateOne  
DeleteMany DeleteOne  
ReplaceOne



I venen acompanyats de  
les corresponents  
**versions asíncrones**

**FindAsync**  
FindOneAsync    InsertManyAsync  
InsertOneAsync  
UpdateManyAsync    UpdateOneAsync  
DeleteManyAsync    DeleteOneAsync  
ReplaceOneAsync





```
db.col  
  .find({"ofici": "Pagès"})
```

*Podem interrogar la  
col·lecció amb les  
operacions Mongo*

```
var filter = new BsonDocument().Add("ofici", "Pagès")  
var peres = coleccio.Find(filter);
```



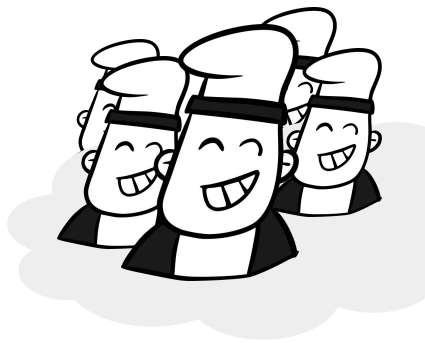
Els resultats són  
"fluent", es poden  
integrar amb el  
llenguatge

```
var filter = new BsonDocument().Add("ofici", "Pagès")  
var primerPere = coleccio.Find(filter).FirstOrDefault();
```

.ToList()

.ToCursor()

.ToEnumerable()



Es poden processar  
resultats sense  
esperar a rebre'ls  
tots

```
var filter = new BsonDocument().Add("ofici", "Pagès")  
var primerPere = coleccio.Find(filter)  
    .ForEachAsync( d => Console.WriteLine(d) );
```



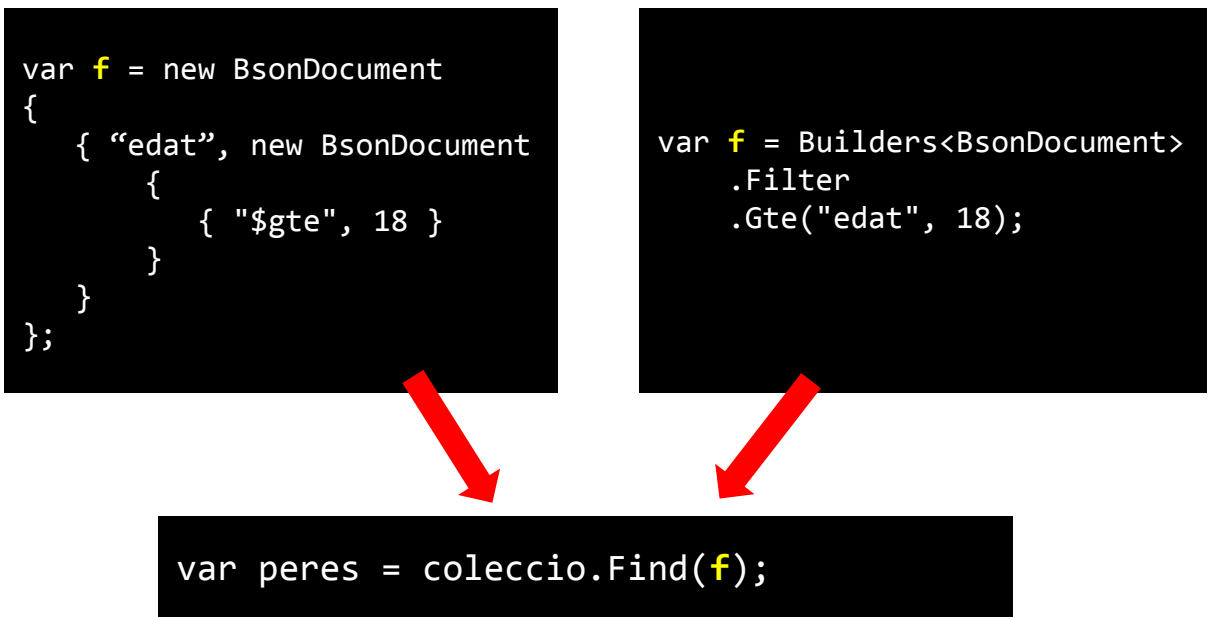
# Builders

---

Els builders fan  
més fàcil definir  
filtres,  
projeccions,  
operacions Update  
...

```
var f = new BsonDocument
{
    { "edat", new BsonDocument
        {
            { "$gte", 18 }
        }
    }
};
```

```
var f = Builders<BsonDocument>
    .Filter
    .Gte("edat", 18);
```



```
var peres = coleccio.Find(f);
```

Si més no, més  
fàcil de llegir ...



```
var s = new BsonDocument  
{  
  { "nom", 1 }  
};
```

```
var s = Builders<BsonDocument>  
  .Sort  
  .Ascending("nom");
```

```
var peres = coleccio.Find(f).Sort(s);
```

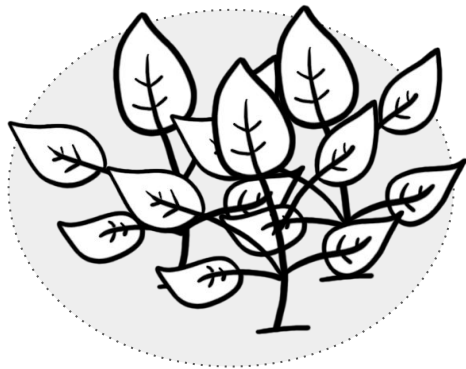


```
var s = new BsonDocument  
{  
    { "_id", 0 }  
};
```

```
var s = Builders<BsonDocument>  
    .Projection  
    .Exclude("_id");
```

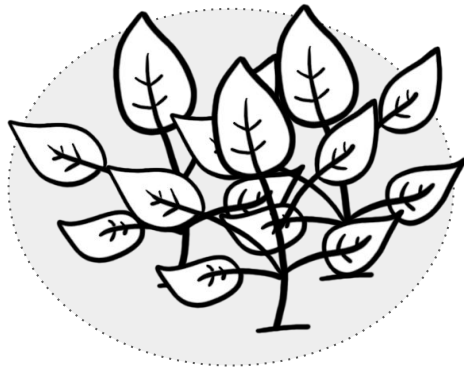
```
var peres = coleccion.Find(f)  
    .Project(s);
```

L'Aggregation  
funciona de la  
mateixa forma



```
var agrupa = new BsonDocument ...  
var ordena = new BsonDocument ...  
  
var resultats = col.Aggregate(  
    new BsonDocument[]  
    {  
        agrupa,  
        ordena  
    }  
).ToList();
```

Es pot usar fent  
servir expressions  
Lambda

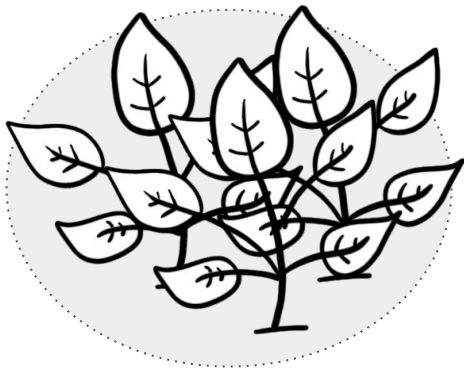


En les  
expressions  
es pot usar  
C#

```
var results = col.Aggregate()  
    .Group(g => g["adreça.poblacio"],  
           r => new  
               {  
                   poble = r.Key,  
                   suma = r.Count()  
               })  
    .Match(c => c.suma > 10);
```

Group és el que  
canvia més

Generació del  
document



Criteri d'agrupació

```
.Group(g => g["adreça.poblacio"],  
       r => new  
       {  
           poble = r.Key,  
           suma = r.Count()  
       }  
)
```

# Estructures predefinides

---

```
public class Usuaris
{
    [BsonId]
    public string Id {get; set;}

    [BsonElement("usuari")]
    public Username {get; set;}

    [BsonElement("grups")]
    public List<string> Grups {get; set;}
}

var col =
    db.GetCollection<Usuaris>("users");
```



Si té estructura, es  
pot mapejar  
automàticament



Les col·leccions  
i els seus  
resultats es  
poden integrar  
amb Linq

```
var informatics = col.AsQueryable<Usuaris>()  
    .Where(u => u.Grups.Contains("Informàtica"))  
    .Select(c => c.Username)  
    .Distinct()  
    .ToList()
```



**LINQ**  
Microsoft  
.NET





Les no estructurades  
també es poden  
integrar amb Linq

```
var informatics = col.AsQueryable<Usuaris>()  
    .Where(u => u.Grups.Contains("Informàtica"))  
    .Select(c => c.Username)  
    .Distinct()  
    .ToList()
```

```
var informatics = col.AsQueryable()  
    .Where(d => d["grups"] == "Informàtica")  
    .Select(d => d["username"])  
    .Distinct()  
    .ToList();
```

