Fetching all comments:

|  |
| --- |
| async function commentList() {  const comment = await Comment  .find()  .select('name');  console.log(comment);  }  //createUser("rakib","mail@y.se")  //createComment("somecomments","5e0e0b35d90ca352acd6436a" )  commentList(); |

Fetching comments with username(ID):

|  |
| --- |
| async function commentList() {  const comment = await Comment  .find()  .select('name user');  console.log(comment);  }  //createUser("rakib","mail@y.se")  //createComment("somecomments","5e0e0b35d90ca352acd6436a" )  commentList(); |

Fetching comments with user documents:

async function commentList() {

const comment = await Comment

.find()

//same name as comment model user object

.populate("user")

.select('name user');

console.log(comment);

}

//createUser("rakib","mail@y.se")

//createComment("somecomments","5e0e0b35d90ca352acd6436a" )

commentList();

Läs mer :<https://mongoosejs.com/docs/populate.html>

fetching specifik data från user Model( ex. name only) :

.populate("user", "name -\_id")

Denormalisation : Man refererar docs

const userSchema = new mongoose.Schema({

name: String,

email: String

})

const User = mongoose.model('User',userSchema);

const Comment= mongoose.model('Comment', new mongoose.Schema({

name: String,

**user:userSchema** //user docs

/\* istället för {

type:mongoose.Schema.Types.ObjectId,

ref:"User"

} \*/

}));

async function createUser(name, email) {

const user = new User({

name,

email

});

const result = await user.save();

console.log(result);

}

async function createComment(name, user) {

const comment = new Comment({

name,

user

})

const result = await comment.save();

console.log(result);

}

async function commentList() {

const comment = await Comment

.find()

.populate("user", "name -\_id")

.select('name user');

console.log(comment);

}

/**/man ska behöva redigera från parent docs om man**

**//vill redigera child docs:**

**async function updateUser(commentId){**

**const comment= await Comment.findById(commentId);**

**comment.user.name="Axel"**

**comment.save()**

**console.log(comment.user.name)**

**}**

//createUser("rakib","mail@y.se")

createComment("somecomments",{name:"someone" , email:"someone@email.se"} )

//commentList();

updateUser("5e0f23f342fdcb1474d6337e")

Övningar:

* Inbygga comments docs in i user docs
* Skapa en function för att uppdatera inbyggda comments docs. Gör uppdaterat värde dynamisk(lägg till två parametrar till ex updateUser(userId, updatedComment))
* Skapa en funktion för att radera inbyggda comments docs
* Skapa en Schema validering för comment egenskapen. till ex. max comment length.

extra:

* Spara alla comments i en comments array

ObjectId:

|  |
| --- |
| till ex : \_id:"5e0f3add45cd04426ce37a61"  24 tecken, varje 2 tecken representerar 1 byte.  12 bytes data för unik id.  16m unika data.  Läs mer : <https://docs.mongodb.com/manual/reference/method/ObjectId/>  Validera objectID.  if(!mongoose.Types.ObjectId.isValid(commentId)) return “error meddelandet”; |