

Modelling Data and Advanced Mongoose

Locations (Geospatial Data)

Sy: 1

```
6 const tourSchema = new mongoose.Schema(  
7   {  
92     startLocation: {  
93       // GeoJSON  
94       type: {  
95         type: String,  
96         default: 'Point',  
97         enum: ['Point'],  
98       },  
99       coordinates: [Number],  
100      address: String,  
101      description: String,  
102    },  
103    locations: [  
104      {  
105        type: {  
106          type: String,  
107          default: 'Point',  
108          enum: ['Point'],  
109        },  
110        coordinates: [Number],  
111        address: String,  
112        description: String,  
113        day: Number,  
114      },  
115    ],  
116    guides: [  
117      {  
118        type: mongoose.Schema.ObjectId,  
119        ref: 'User',  
120      },  
121    ], Start Location için kullanılan GeoJson veri tipi.  
122  }, Locations dizisi için de GeoJson veri tipi kullanılmakta.  
123 );
```

Modelling Data and Advanced Mongoose

Populating 1

Sy: 2

tourSchema.guides Property

```
6 const tourSchema = new mongoose.Schema()
7 {
8     • • •
116     guides: [
117         {
118             type: mongoose.Schema.ObjectId,
119             ref: 'User',
120         },
121     ],
122 },
127 );
```

tourSchema find populate

```
1 tourSchema.pre(/^find/, function (next) {
2     this.populate({
3         path: 'guides',
4         select: '-__v -passwordChangedAt',
5     });
6
7     next();
8 });
```

Get Tour

The screenshot shows a Postman interface with a GET request to `api/v1/tours/5c88fa8cf4afda39709c2951`. The response body is a JSON object representing a tour document. The tour has an ID of `5c88fa8cf4afda39709c2951`, a name of "The Forest Hiker", a duration of 5, a max group size of 25, a difficulty level of "easy", and a price of 397. It is marked as not secret (`secretTour: false`). The tour's guides are listed as an array, with one guide's details highlighted: an ID of `682cb8aaad40925ec430c628`, a role of "guide", a name of "User", and an email of "guide@jonas.io".

```
26 {
27     "secretTour": false,
28     "guides": [
29         {
30             "role": "guide",
31             "_id": "682cb8aaad40925ec430c628",
32             "name": "User",
33             "email": "guide@jonas.io"
34         },
35         {
36             "_id": "5c88fa8cf4afda39709c2951",
37             "name": "The Forest Hiker",
38             "duration": 5,
39             "maxGroupSize": 25,
40             "difficulty": "easy",
41             "price": 397,
```

Tour şeması üzerinde oluşturulan **guides property'si** User şeması ile ilişki kurmak için kullanılan bir alandır. Tour şeması üzerinde bulunan **guides** özelliği turda görevli kullanıcıların **id** bilgilerini tutar. Guides ve id üzerinden kurulan ilişki ile turda görevli kullanıcı bilgileri alınabilir. Tour bilgileri üzerinde yapılan her **find** ile başlayan sorgularda Tour.guides üzerinde tutulan idler üzerinden **this.populate()** metodu işletilerek kullanıcıların diğer bilgileri getirilir.

Modelling Data and Advanced Mongoose

Review Schema

Sy: 3

reviewSchema

```
1 const reviewSchema = new mongoose.Schema(  
2 {  
3   review: {  
4     type: String,  
5     required: [true, 'Review can not be empty!'],  
6   },  
7   rating: {  
8     type: Number,  
9     min: 1,  
10    max: 5,  
11  },  
12  createdAt: {  
13    type: Date,  
14    default: Date.now,  
15  },  
16  tour: {  
17    type: mongoose.Schema.ObjectId,  
18    ref: 'Tour',  
19    required: [true, 'Review must belong to a tour.'],  
20  },  
21  user: {  
22    type: mongoose.Schema.ObjectId,  
23    ref: 'User',  
24    required: [true, 'Review must belong to a user'],  
25  },  
26},  
27{  
28  toJSON: { virtuals: true },  
29  toObject: { virtuals: true },  
30};  
31);  
32  
33 reviewSchema.pre(/^find/, function (next) {  
34   this.populate({  
35     path: 'tour',  
36     select: 'name',  
37   }).populate({  
38     path: 'user',  
39     select: 'name photo',  
40   });  
41  
42   next();  
43});
```

reviewController

```
1 exports.getAllReviews = catchAsync(async (req, res, next) => {  
2   const reviews = await Review.find();  
3  
4   res.status(200).json({  
5     status: 'success',  
6     results: this.getAllReviews.length,  
7     data: {  
8       reviews,  
9     },  
10   });  
11 };  
12  
13 exports.createReview = catchAsync(async (req, res, next) => {  
14   const newReview = await Review.create(req.body);  
15  
16   res.status(201).json({  
17     status: 'success',  
18     data: {  
19       review: newReview,  
20     },  
21   });  
22 });
```

Create New Review

HTTP Jonas Natours 2025 / Review / Create New Review

POST {{URL}} api/v1/reviews

Params Authorization Headers (11) Body Scripts Settings

Body

```
1 {  
2   "review": "Loved it!",  
3   "rating": 4,  
4   "tour": "5c88fa8cf4afda39709c2951",  
5   "user": "682da32af55f7ce393e30f41"  
6 }
```

Body Cookies (1) Headers (21) Test Results

{} JSON Preview Visualize

```
1 {  
2   "status": "success",  
3   "data": {  
4     "review": {  
5       "_id": "682da3baf55f7ce393e30f47",  
6       "review": "Loved it!",  
7       "tour": "5c88fa8cf4afda39709c2951",  
8       "user": "682da32af55f7ce393e30f41",  
9       "createdAt": "2025-05-21T09:58:18.510Z",  
10      "__v": 0,  
11      "id": "682da3baf55f7ce393e30f47"  
12    }  
13  }
```

Get Review

GET {{URL}} api/v1/reviews

Params Authorization Headers (9) Body Scripts Settings

Body

This request does not have a body

{} JSON Preview Visualize

```
31 {  
32   "tour": {  
33     "guides": [  
34       {  
35         "role": "guide",  
36         "id": "682cb8aaad40925ec430c628",  
37         "name": "User",  
38         "email": "guide@jonas.io"  
39       },  
40       {  
41         "role": "guide",  
42         "id": "5c88fa8cf4afda39709c2951",  
43         "name": "The Forest Hiker",  
44         "durationWeeks": null,  
45         "id": "5c88fa8cf4afda39709c2951"  
46     },  
47     "user": {  
48       "id": "682da32af55f7ce393e30f41",  
49       "name": "User"  
50     },  
51     "createdAt": "2025-05-21T09:58:18.510Z",  
52     "__v": 0,  
53     "id": "682da3baf55f7ce393e30f47"  
54   },  
55 }
```

Turlar hakkında yapılan görüşleri **Review Schema** üzerinden veritabanına yazılır. Girilen bir **review** üzerinde **tourId** ve **userId** bulundurmak tour ve user dökümanlarının gereksiz büyümесini engeller. Gerektiğinde review üzerinde bulunan tourId ve userId ile ilgili dökümanların bilgileri **populate** metoduyla çekilebilir.

Modelling Data and Advanced Mongoose

Populate Fault!

Sy: 4

Get A Tour

```
HTTP Jonas Natours 2025 / Tour / Get Requests / Get A Tour

GET https://jonasnatur.com/api/v1/tours/5c88fa8cf4afda39709c2951

Params Authorization Headers (9) Body Scripts Settings
None form-data x-www-form-urlencoded raw binary GraphQL

Body Cookies (1) Headers (21) Test Results

{} JSON ▾ ▶ Preview Visualize ▾
{
    "tour": {
        "id": "5c88fa8cf4afda39709c2951",
        "imageCover": "tour-1-cover.jpg",
        "locations": [...],
        "slug": "the-forest-hiker",
        "__v": 0,
        "durationWeeks": 0.7142857142857143,
        "reviews": [
            {
                "id": "682da3baf55f7ce393e30f47",
                "review": "Loved it!",
                "tour": {
                    "id": "5c88fa8cf4afda39709c2951",
                    "imageCover": "tour-1-cover.jpg",
                    "locations": [...],
                    "slug": "the-forest-hiker",
                    "__v": 0,
                    "durationWeeks": 0.7142857142857143,
                    "reviews": [
                        {
                            "id": "682da32af55f7ce393e30f41",
                            "name": "User"
                        }
                    ],
                    "user": {
                        "id": "682da32af55f7ce393e30f41",
                        "name": "User"
                    },
                    "createdAt": "2025-05-21T09:58:18.510Z"
                }
            }
        ]
    }
}
```

reviewSchema Find Middleware

```
33 reviewSchema.pre(/^find/, function (next) {
34     this.populate({
35         path: 'tour',
36         select: 'name',
37     }).populate({
38         path: 'user',
39         select: 'name photo',
40     });
41
42     next();
43 });


```

reviewSchema Middleware Fix

```
1 reviewSchema.pre(/^find/, function (next) {
2     this.populate({
3         path: 'user',
3         select: 'name photo',
4     });
5
6     next();
7 });
8 }
```

```
HTTP Jonas Natours 2025 / Tour / Get Requests / Get A Tour

GET https://jonasnatur.com/api/v1/tours/5c88fa8cf4afda39709c2951

Params Authorization Headers (9) Body Scripts Settings
None form-data x-www-form-urlencoded raw binary GraphQL

Body Cookies (1) Headers (21) Test Results

{} JSON ▾ ▶ Preview Visualize ▾
{
    "tour": {
        "id": "5c88fa8cf4afda39709c2951",
        "imageCover": "tour-1-cover.jpg",
        "locations": [...],
        "slug": "the-forest-hiker",
        "__v": 0,
        "durationWeeks": 0.7142857142857143,
        "reviews": [
            {
                "id": "682da3baf55f7ce393e30f47",
                "review": "Loved it!",
                "tour": {
                    "id": "5c88fa8cf4afda39709c2951"
                }
            }
        ]
    }
}
```

tourSchema Middleware Fix (Virtual Property)

```
1 // Virtual populate
2 tourSchema.virtual('reviews', {
3     ref: 'Review',
4     foreignField: 'tour',
5     localField: '_id',
6 });


```

tourControl.getTour()

```
1 exports.getTour = catchAsync(async (req, res, next) => {
2     const tour = await Tour.findById(req.params.id).populate('reviews');
3
4     if (!tour) {
5         return next(new AppError('No tour found with that ID', 404));
6     }
7
8     res.status(200).json({
9         status: 'success',
10        data: {
11            tour,
12        },
13    });
14 });


```

Review Schema üzerinde **tour** ve **user** için **populate** işlemi uygulanarak bir **tour** dokumanı çekildiğinde, **tour** içinde bulunan **review** bilgisi üzerinden yeniden **tour** bilgisi gelir. Bilginin bu şekilde sürekli tekrar etmesi istenen bir durum değildir. Bu durumu düzeltmek için **reviewSchema** üzerinde bulunan **find middleware** üzerindeki **tour populate** silinir. **tourSchema** üzerinde gereksiz alan barındırmamak için **review** dökümanına ait bir bilgi saklanmaz. Fakat **tour** üzerinde **review** bilgisini görüntülemek gerekiyinde bu durum çözmek için **virtual property** tanımlanabilir. **Virtual Property** hafızada çalışan sanal bir alan olduğu için bilgisayarda yer kaplamaz. **tourSchema** üzerinde **reviews** için **virtual property** hazırlandıktan sonra, **tourControl.getTour()** metodunda **populate('reviews')** uygulanır.

Modelling Data and Advanced Mongoose

Simple Nested Routers

Sy: 5

Sign Up New User

```
POST {{URL}} api/v1/users/signup
```

Params Authorization Headers (9) Body Scripts Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

Beautify

```
1 {
  "name": "jonas",
  "email": "hello@jonas.io",
  "password": "pass1234",
  "passwordConfirm": "pass1234"
}
```

Body Preview Visualize 201 Created 286 ms 1.5 KB Save Response

```
{ } JSON
```

```
1 {
  "status": "success",
  "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZC16IjY4MmlxNjhNTFhZGQxNTg3NDExZWUzZSIsImhdCI6MTc0Nzg1MDg5MSwiZXhwIjoxNzU1NjI20DkxfQ.6XnJBqAjqmqlugEJ13bnablFnDb04eP0V36D6G2bexk",
  "data": {
    "user": {
      "role": "user",
      "active": true,
      "_id": "682e168b51add1587411ee3e",
      "name": "jonas",
      "email": "hello@jonas.io",
      "__v": 0
    }
  }
}
```

Get A Tour

```
GET {{URL}} api/v1/tours/5c88fa8cf4afda39709c2951
```

Body Preview Visualize 200 OK 12 ms 2.77 KB Save Response

```
{ } JSON
```

```
26 {
  "secretTour": false,
  "guides": [],
  "_id": "5c88fa8cf4afda39709c2951",
  "name": "The Forest Hiker",
  "duration": 5,
  "maxGroupSize": 25,
  "difficulty": "easy",
  "price": 397,
  "summary": "Breathtaking hike through the Canadian Banff National Park",
}
```

Create New Review on Tour

```
POST {{URL}} api/v1/tours/5c88fa8cf4afda39709c2951/reviews
```

Body Preview Visualize 201 Created 22 ms 1.2 KB Save Response

```
{ } JSON
```

```
1 {
  "rating": 5,
  "review": "Best tour in the world!"
}
```

```
1 {
  "status": "success",
  "data": {
    "review": {
      "_id": "682e17ecf77e55c84643676",
      "rating": 5,
      "review": "Best tour in the world!",
      "tour": "5c88fa8cf4afda39709c2951",
      "user": "682e168b51add1587411ee3e",
      "createdAt": "2025-05-21T18:14:04.012Z",
      "__v": 0,
      "id": "682e17ecf77e55c84643676"
    }
  }
}
```

tourRoute :tourId

/tours/:tourId/reviews

```
1 const express = require('express');
2 const authController = require('../controllers/authController');
3 const reviewController = require('../controllers/reviewController');
4
5 const router = express.Router();
6
7 // POST /tour/234fad4/reviews
8 // GET /tour/234fad4/reviews
9 // GET /tour/234fad4/reviews/94887fda
10
11 router
12   .route('/:tourId/reviews')
13   .post(
14     authController.protect,
15     authController.restrictTo('user'),
16     reviewController.createReview,
17   );
18
19 module.exports = router;
```

1 exports.protect = catchAsync(async (req, res, next) => {
 ...
 // GRANT ACCESS TO PROTECTED ROUTE
 req.user = currentUser;
 next();
});

createReview

```
1 const Review = require('../models/reviewModel');
2 const catchAsync = require('../utils/catchAsync');
3
4 exports.createReview = catchAsync(async (req, res, next) => {
5   // Allow nested routers
6   if (!req.body.tour) req.body.tour = req.params.tourId;
7   if (!req.body.user) req.body.user = req.user.id;
8
9   const newReview = await Review.create(req.body);
10
11 res.status(201).json({
12   status: 'success',
13   data: {
14     review: newReview,
15   },
16 });
17
18
19 _id: ObjectId('682e17ecf77e55c84643676')
20 rating: 5
21 review: "Best tour in the world!"
22 tour: ObjectId('5c88fa8cf4afda39709c2951')
23 user: ObjectId('682e168b51add1587411ee3e')
24 createdAt: 2025-05-21T18:14:04.012+00:00
25 __v: 0
```

Bir review kaydını **/tours/:tourId/reviews** route'ü üzerinden yapmak gereklidir. review kaydı bu şekilde yapılınca review için gerekli olan tourId verisi route üzerinden alınır ve tourId manuel olarak girilmez.

createReview metodu incelendiğinde review için gerekli olan tourId ve userId bilgilerinin nasıl alındığı görülmektedir.

tourId **/tours/:tourId/reviews** üzerinden params ile userId protect fonksiyonu ile login olan kullanıcı bilgileri üzerinden alınmaktadır.

Modelling Data and Advanced Mongoose

Express Nested Routes

Sy: 6

Tour Routes

```
routes > JS tourRoutes.js > ...
27   authController.protect,
28   authController.restrictTo('admin', 'lead-guide'),
29   tourController.deleteTour,
30 );
31
32 // POST /tour/234fad4/reviews
33 // GET /tour/234fad4/reviews
34 // GET /tour/234fad4/reviews/94887fda
35
36 router
37   .route('/:tourId/reviews')
38   .post(
39     authController.protect,
40     authController.restrictTo('user'),
41     reviewController.createReview,
42   );
43
44 module.exports = router;
```

Review Routes

```
routes > JS reviewRoutes.js > ...
You, 23 hours ago | 1 author (You)
1 const express = require('express');
2 const reviewController = require('../controllers/reviewController');
3 const authController = require('../controllers/authController');
4
5 const router = express.Router();
6
7 router
8   .route('/')
9   .get(reviewController.getAllReviews)
10  .post(
11    authController.protect,
12    authController.restrictTo('user'),
13    reviewController.createReview,
14  );
15
16 module.exports = router;
```

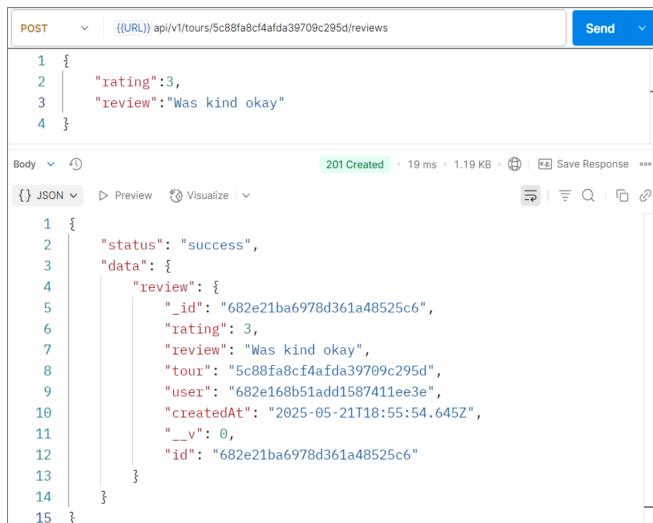
Tour Routes Fix

```
1 const express = require('express');
2 const reviewRouter = require('../routes/reviewRoutes');
3
4 const router = express.Router();
5
6 router.use('/:tourId/reviews', reviewRouter);
7
8 module.exports = router;
```

Review Routes Fix

```
1 const express = require('express');
2 const reviewController = require('../controllers/reviewController');
3 const authController = require('../controllers/authController');
4
5 const router = express.Router({ mergeParams: true });
6
7 router
8   .route('/')
9   .get(reviewController.getAllReviews)
10  .post(
11    authController.protect,
12    authController.restrictTo('user'),
13    reviewController.createReview,
14  );
15
16 module.exports = router;
```

Create a review on tour



```
POST {{URL}} api/v1/tours/5c88fa8cf4afda39709c295d/reviews
Body
1 {
  "rating":3,
  "review":"Was kind okay"
}
201 Created
```

```
1 {
  "status": "success",
  "data": {
    "review": {
      "_id": "682e21ba6978d361a48525c6",
      "rating": 3,
      "review": "Was kind okay",
      "tour": "5c88fa8cf4afda39709c295d",
      "user": "682e168b51add1587411ee3e",
      "createdAt": "2025-05-21T18:55:54.645Z",
      "__v": 0,
      "id": "682e21ba6978d361a48525c6"
    }
  }
}
```

Get tour



```
GET {{URL}} api/v1/tours/5c88fa8cf4afda39709c295d
200 OK
```

```
1 {
  "name": "The City Wanderer",
  "coordinates": [
    -122.408865,
    37.787825
  ],
  "_id": "5c88fa8cf4afda39709c295d",
  "description": "San Francisco",
  "day": 5,
  "slug": "the-city-wanderer",
  "__v": 0,
  "durationWeeks": 1.2857142857142858,
  "reviews": [
    {
      "_id": "682e21ba6978d361a48525c6",
      "rating": 3,
      "review": "Was kind okay",
      "tour": "5c88fa8cf4afda39709c295d",
      "user": {
        "_id": "682e168b51add1587411ee3e",
        "name": "jonas"
      }
    }
  ]
}
```

Üstte bulunan simple nested routes çözümü **tourRoutes** ve **reviewRoutes** üzerinde kod tekrarı oluşturur. Bu istenilen bir durum değildir. Bu durumun çözümü için **router.use()** metodu kullanılır. **tourRoutes** üzerinde bulunan **router.use()** metodunun ilk parametresi route adresi, ikinci parametresi kullanılacak router metodudur(**reviewRouter**). **reviewRoutes** üzerinde router nesnesi oluşturulurken **express.Router()** metodu **mergeParams:true** değerini alarak router birleşimini gerçekleştirir.

Modelling Data and Advanced Mongoose

Create Reviews on Tour

Sy: 7

Post Review on Tour

```
1 //POST: {{URL}}api/v1/tours/5c88fa8cf4afda39709c295d/reviews
2 {
3   "rating": 4,
4   "review": "very good tour"
5 }
```

tourRoute

```
1 const express = require('express');
2 const reviewRouter = require('../routes/reviews');
3
4 const router = express.Route;
5
6 router.use('/:tourId/reviews', reviewRouter);
7
8 module.exports = router;
```

reviewRoute

```
1 const express = require('express');
2 const reviewController = require('../controllers/reviews');
3 const authController = require('../controllers/auth');
4
5 const router = express.Router({ mergeParams: true });
6
7 // POST /tour/234fad4/reviews
8 // GET /tour/234fad4/reviews
9 // POST /reviews
10
11 router
12   .route('/')
13   .get(reviewController.getAllReviews)
14   .post(
15     authController.protect,
16     authController.restrictTo('user'),
17     reviewController.createReview,
18   );
19
20 module.exports = router;
```

reviewController.createReview()

```
1 exports.createReview = catchAsync(async (req, res, next) => {
2   // Allow nested routers
3   if (!req.body.tour) req.body.tour = req.params.tourId;
4   if (!req.body.user) req.body.user = req.user.id;
5
6   const newReview = await Review.create(req.body);
7
8   res.status(201).json({
9     status: 'success',
10    data: {
11      review: newReview,
12    },
13  });
14});
```

MongoDB

```
_id: ObjectId('68315308bfd2a56e2adbc387')
rating : 4
review : "very good tour"
tour : ObjectId('5c88fa8cf4afda39709c295d')
user : ObjectId('68315303bfd2a56e2adbc384')
createdAt : 2025-05-24T05:03:04.526+00:00
```

tours route'ı üzerinden gönderilen reviews post'u öncelikle tourRoute üzerinden yakalanır ve burada bulunan router.use() metodu ile bu talep reviewRoute'una iletirilir. tourRoute üzerinden gönderilen parametreler mergeParams:true parametresi ile birleştirilerek tourId değeri reviewRoute üzerinden alınır. reviewRoute üzerinden post işlemi öncelikle protect() metoduya başlar. protect() metodu kullanıcının login olup olmadığına bakar. Eğer kullanıcı login oldussa user nesnesi protect() metodu üzerinden creatReview() metoduna gönderilir.

createReview() metodu tourRoute() metodu üzerinden gelen parametre verisindeki tourId ve protect() metodundan gelen userId verisini alarak kullanıcı review bilgisini veritabanına yazar.

Modelling Data and Advanced Mongoose

Get Reviews on Tour

Sy: 8

{URL}api/v1/tours/5c88fa8cf4afda39709c295d/reviews

```
GET      ({URL}) api/v1/tours/5c88fa8cf4afda39709c295d/reviews Send
tourRoute
1 const express = require('express');
2 const reviewRouter = require('../routes/reviews');
3
4 const router = express.Router();
5
6 router.use('/:tourId/reviews', reviewRouter);
7
8 module.exports = router;
```

reviewController.getAllReviews()

```
1 exports.getAllReviews = catchAsync(async (req, res, next) => {
2   let filter = {};
3   if (req.params.tourId) filter = { tour: req.params.tourId };
4
5   // { tour: '5c88fa8cf4afda39709c295d' }
6
7   const reviews = await Review.find(filter);
8
9   res.status(200).json({
10     status: 'success',
11     results: reviews.length,
12     data: {
13       reviews,
14     },
15   });
16 });
```

```
GET      ({URL}) api/v1/tours/5c88fa8cf4afda39709c295d/reviews Send
Params Auth Headers (9) Body Scripts Settings Cookies
Body Params Auth Headers (9) Body Scripts Settings Cookies
200 OK 34 ms 1.46 KB Save Response ...
{} JSON Preview Visualize
2 {"status": "success",}
3 // {URL}api/v1/tours/5c88fa8cf4afda39709c295d/reviews
4 {
5   "results": 2,
6   "reviews": [
7     {
8       "_id": "682e251ec3ecb67e7a17f1a6",
9       "rating": 3,
10      "review": "Was kind okay",
11      "tour": "5c88fa8cf4afda39709c295d",
12      "user": {
13        "_id": "682da32af55f7ce393e30f41",
14        "name": "User"
15      },
16    },
17    {
18      "_id": "68315308bfd2a56e2adbc387",
19      "rating": 4,
20      "review": "very good tour",
21      "tour": "5c88fa8cf4afda39709c295d",
22      "user": {
23        "_id": "68315303bfd2a56e2adbc384",
24        "name": "tuna"
25      }
26    }
27  ]
28 }
29 }
30 }
31 }
```

reviewRoute

```
1 const express = require('express');
2 const reviewController = require('../controllers/reviews');
3 const authController = require('../controllers/auth');
4
5 const router = express.Router({ mergeParams: true });
6
7 // POST /tour/234fad4/reviews
8 // GET /tour/234fad4/reviews
9 // POST /reviews
10
11 router
12   .route('/')
13   .get(reviewController.getAllReviews)
14   .post(
15     authController.protect,
16     authController.restrictTo('user'),
17     reviewController.createReview,
18   );
19
20 module.exports = router;
```

reviewSchema

```
1 const reviewSchema = new mongoose.Schema(
2   {
3     review: {
4       type: String,
5       required: [true, 'Review can not be empty!'],
6     },
7     rating: {
8       type: Number,
9       min: 1,
10      max: 5,
11    },
12    tour: {
13      type: mongoose.Schema.ObjectId,
14      ref: 'Tour',
15      required: [true, 'Review must belong to a tour.'],
16    },
17    user: {
18      type: mongoose.Schema.ObjectId,
19      ref: 'User',
20      required: [true, 'Review must belong to a user'],
21    },
22    • • •
23  },
24  {
25    • • •
26  }
27 );
```

get() metoduna gelene kadar post() metodunda bulunan adımlar aynı şekildedir.

tourRoute üzerinden gelen tourId Review scheması üzerinden sorgu yapabilmek için filter isimli nesneye dönüştürülür ve find() metoduna argüman olarak gönderilir.

Bu işlemin sonucu olarak bir tour'a bağlı olan review bilgileri nested route yardımıyla yapılır.