Modelling Data and Advanced Mongoose

Locations (Geospatial Data)


```
 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     ],
122   },
127 );
```

Start Location için kullanılan **GeoJson** veri tipi.

Locations dizisi için de **GeoJson** veri tipi kullanılmakta.

Modelling Data and Advanced Mongoose

Populating 1

Sy: 2

tourSchema.guides Property

```
6   const tourSchema = new mongoose.Schema(
7     {
                  ● ● ●
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

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

Params Authorization Headers (9) Body Scripts Settings

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

This request does not

Body Cookies (1) Headers (21) Test Results

{} JSON Preview Visualize

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

```
const reviewSchema = new mongoose.Schema(
  {
    review: {
      type: String,
      required: [true, 'Review can not be empty!'],
    },
    rating: {
      type: Number,
      min: 1,
      max: 5,
    },
    createdAt: {
      type: Date,
      default: Date.now,
    },
    tour: {
      type: mongoose.Schema.ObjectId,
      ref: 'Tour',
      required: [true, 'Review must belong to a tour.'],
    },
    user: {
      type: mongoose.Schema.ObjectId,
      ref: 'User',
      required: [true, 'Review must belong to a user'],
    },
  },
  {
    toJSON: { virtuals: true },
    toObject: { virtuals: true },
  },
);

reviewSchema.pre(/^find/, function (next) {
  this.populate({
    path: 'tour',
    select: 'name',
  }).populate({
    path: 'user',
    select: 'name photo',
  });

  next();
});
```

reviewController

```
exports.getAllReviews = catchAsync(async (req, res, next) => {
  const reviews = await Review.find();

  res.status(200).json({
    status: 'success',
    results: this.getAllReviews.length,
    data: {
      reviews,
    },
  });
});

exports.createReview = catchAsync(async (req, res, next) => {
  const newReview = await Review.create(req.body);

  res.status(201).json({
    status: 'success',
    data: {
      review: newReview,
    },
  });
});
```

Create New Review

Jonas Natours 2025 / Review / **Create New Review**

POST {{URL}} api/v1/reviews

Params | Authorization | Headers (11) | Body | Scripts | Settings

none | form-data | x-www-form-urlencoded | raw | binary | Graph

```
{
    "review": "Loved it!",
    "raiting": 4,
    "tour": "5c88fa8cf4afda39709c2951",
    "user":"682da32af55f7ce393e30f41"
}
```

Body | Cookies (1) | Headers (21) | Test Results

{} JSON | Preview | Visualize

```
{
    "status": "success",
    "data": {
        "review": {
            "_id": "682da3baf55f7ce393e30f47",
            "review": "Loved it!",
            "tour": "5c88fa8cf4afda39709c2951",
            "user": "682da32af55f7ce393e30f41",
            "createdAt": "2025-05-21T09:58:18.510Z",
            "__v": 0,
            "id": "682da3baf55f7ce393e30f47"
        }
    }
}
```

Get Review

GET {{URL}} api/v1/reviews

Params | Authorization | Headers (9) | Body | Scripts | Settings

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

This request does not have a bo

Body | Cookies (1) | Headers (21) | Test Results

{} JSON | Preview | Visualize

```
        {
            "_id": "682da3baf55f7ce393e30f47",
            "review": "Loved it!",
            "tour": {
                "guides": [
                    {
                        "role": "guide",
                        "_id": "682cb8aaad40925ec430c628",
                        "name": "User",
                        "email": "guide@jonas.io"
                    }
                ],
                "_id": "5c88fa8cf4afda39709c2951",
                "name": "The Forest Hiker",
                "durationWeeks": null,
                "id": "5c88fa8cf4afda39709c2951"
            },
            "user": {
                "_id": "682da32af55f7ce393e30f41",
                "name": "User"
            },
            "createdAt": "2025-05-21T09:58:18.510Z",
            "__v": 0,
            "id": "682da3baf55f7ce393e30f47"
        }
```

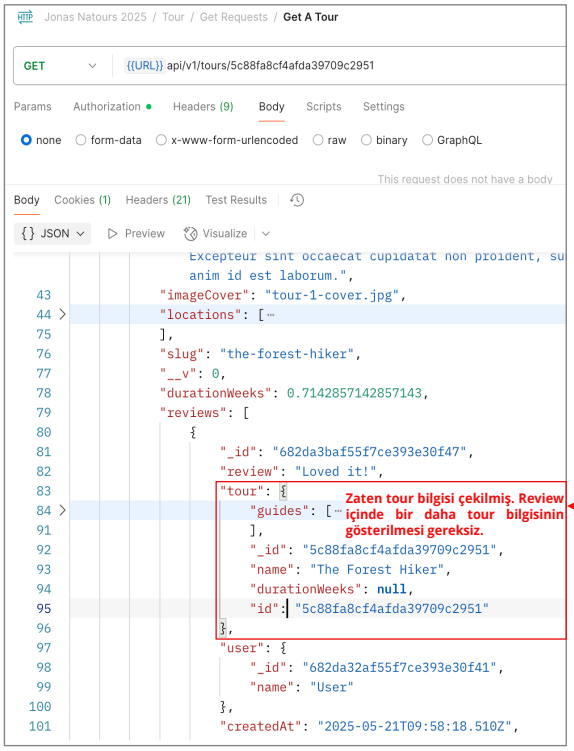
Turlar hakkında yapılan görüşleri **Review Schema** üzerinden veritabanına yazılır. Girilen bir **review** üzerinde **tourId** ve **userId** bulundurmak tour ve user dökümanlarının gereksiz büyümesini engeller. Gerektiğinde review üzerinde bulunan tourId ve userId ile ilgili dökümanların bilgileri **populate**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

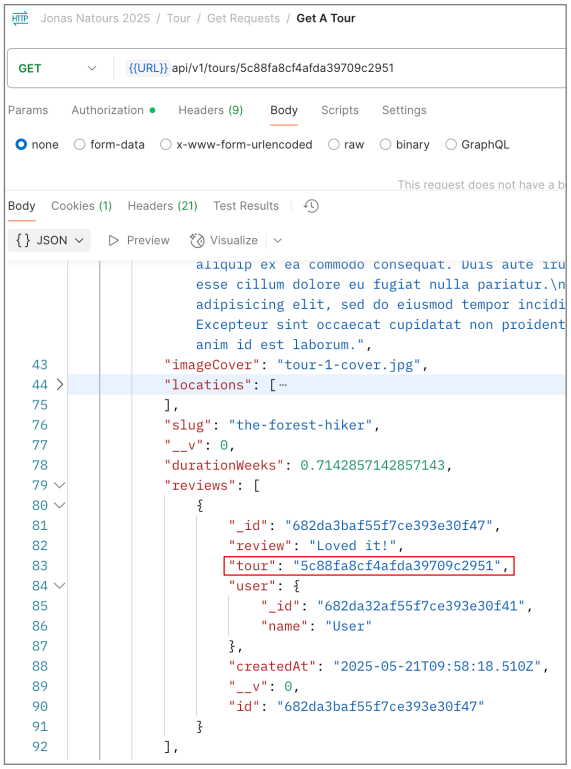
Modelling Data and Advanced Mongoose

Populate Fault!

Sy: 4

Get A Tour





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',
4      select: 'name photo',
5    });
6
7    next();
8  });
```

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 });
```

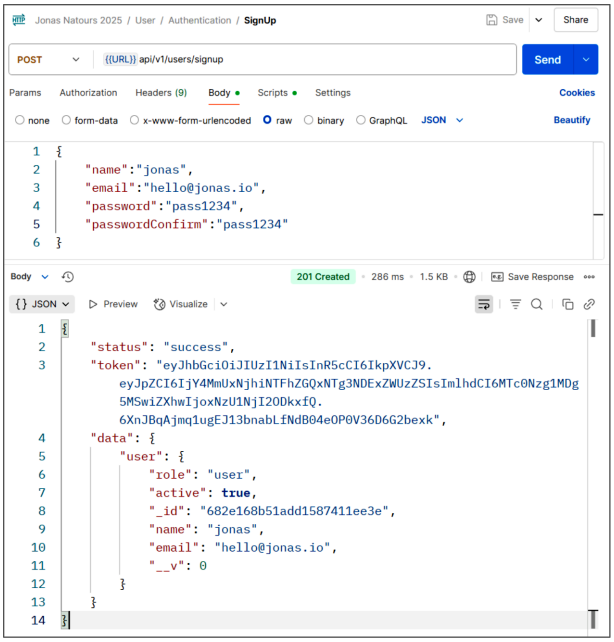
Review Schema *üzerinde* ***tour*** *ve* ***user*** *için populate işlemi uygulanarak bir* ***tour*** *dokumanı çekildiğinde, tour içinde bulunan review bilgisi üzerinden yeniden tour bilgisi gelir.* Bilginin bu şekilde sürekli tekrar etmesi istenen bir durum değildir. Bu durumu düzeltmek için **reviewSchema** üzerinde bulunan **find middleware** üzerinde ki **tour populate** silinir. tourSchema üzerinde gereksiz alan barındırmamak için review dökümanına ait bir bilgi saklanmaz. Fakat tour üzerinde review bilgisini görüntülemek gerektiğinde 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

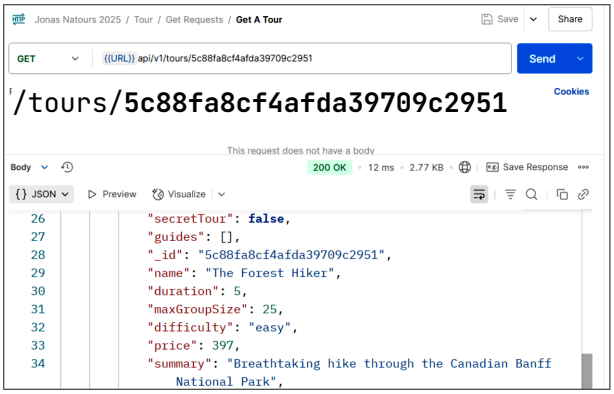
Simple Nested Routers

Sy: 5

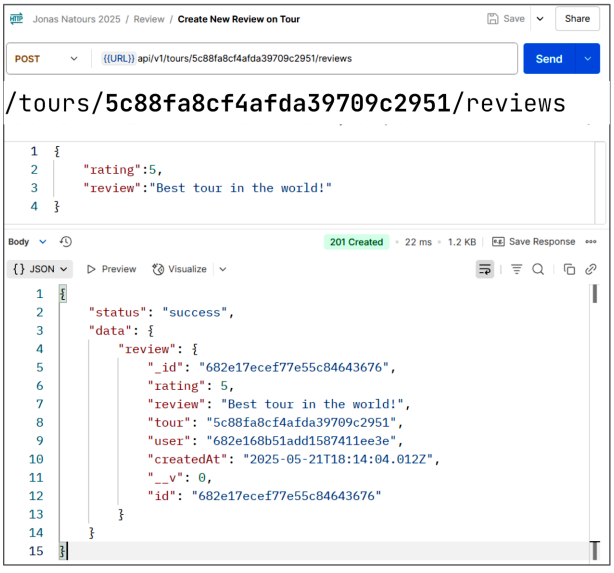
Sign Up New User



Get A Tour



Create New Review on Tour



tourRoute

/tours/5c88fa8cf4afda39709c2951/reviews

:tourId

```
const express = require('express');
const authController = require('./../controllers/authController');
const reviewController = require('./../controllers/reviewController');

const router = express.Router();

// POST /tour/234fad4/reviews
// GET /tour/234fad4/reviews
// GET /tour/234fad4/reviews/94887fda

router
  .route('/:tourId/reviews')
  .post(
    authController.protect,
    authController.restrictTo('user'),
    reviewController.createReview
  );

module.exports = router;
```

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

createReview

```
const Review = require('./../models/reviewModel');
const catchAsync = require('./../utils/catchAsync');

exports.createReview = catchAsync(async (req, res, next) => {
  // Allow nested routers
  if (!req.body.tour) req.body.tour = req.params.tourId;
  if (!req.body.user) req.body.user = req.user.id;

  const newReview = await Review.create(req.body);

  res.status(201).json({
    status: 'success',
    data: {
      review: newReview,
    },
  });
});
```

```
_id: ObjectId('682e17ecef77e55c84643676')
rating : 5
review : "Best tour in the world!"
tour : ObjectId('5c88fa8cf4afda39709c2951')
user : ObjectId('682e168b51add1587411ee3e')
createdAt : 2025-05-21T18:14:04.012+00:00
__v : 0
```

Bir review kaydını **/tours/:toudId/reviews** route'u üzerinden yapmak gerekir. review kaydı bu şekilde yapılırsa 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/:toudId/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 > ...

```
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
3   const authController = require('./../controllers/a
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/re
3  const authController = require('./../controllers/auth
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 Send

```
1  {
2    "rating":3,
3    "review":"Was kind okay"
4  }
```

Body 201 Created 19 ms 1.19 KB Save Response

{} JSON Preview Visualize

```
1  {
2    "status": "success",
3    "data": {
4      "review": {
5        "_id": "682e21ba6978d361a48525c6",
6        "rating": 3,
7        "review": "Was kind okay",
8        "tour": "5c88fa8cf4afda39709c295d",
9        "user": "682e168b51add1587411ee3e",
10       "createdAt": "2025-05-21T18:55:54.645Z",
11       "__v": 0,
12       "id": "682e21ba6978d361a48525c6"
13     }
14   }
15 }
```

Get tour

GET {{URL}} api/v1/tours/5c88fa8cf4afda39709c295d Send

```
60                  "coordinates": [
61                      -122.408865,
62                      37.787825
63                  ],
64                  "_id": "5c88fa8cf4afda39709c295e",
65                  "description": "San Francisco",
66                  "day": 5
67              }
68          ],
69          "slug": "the-city-wanderer",
70          "__v": 0,
71          "durationWeeks": 1.2857142857142858,
72          "reviews": [
73              {
74                  "_id": "682e21ba6978d361a48525c6",
75                  "rating": 3,
76                  "review": "Was kind okay",
77                  "tour": "5c88fa8cf4afda39709c295d",
78                  "user": {
79                      "_id": "682e168b51add1587411ee3e",
80                      "name": "jonas"
```

En üstte bulunan simple nesteded routes çözümü **tourRoutes** ve **reviewRoutes** üzerinde kod tekrarı oluşturur. Bu istenen 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

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

tourRoute

```
const express = require('express');
const reviewRouter = require('./../routes/rev

const router = express.Route;

router.use('/:tourId/reviews', reviewRouter);

module.exports = router;
```

reviewRoute

```
const express = require('express');
const reviewController = require('./../controllers/re
const authController = require('./../controllers/auth

const router = express.Router({ mergeParams: true });

// POST /tour/234fad4/reviews
// GET /tour/234fad4/reviews
// POST /reviews

router
  .route('/')
  .get(reviewController.getAllReviews)
  .post(
    authController.protect,
    authController.restrictTo('user'),
    reviewController.createReview,
  );

module.exports = router;
```

reviewController.createReview()

```
exports.createReview = catchAsync(async (req, res, next) => {
  // Allow nested routers
  if (!req.body.tour) req.body.tour = req.params.tourId;
  if (!req.body.user) req.body.user = req.user.id;

  const newReview = await Review.create(req.body);

  res.status(201).json({
    status: 'success',
    data: {
      review: newReview,
    },
  });
});
```

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

createReviw() 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

tourRoute

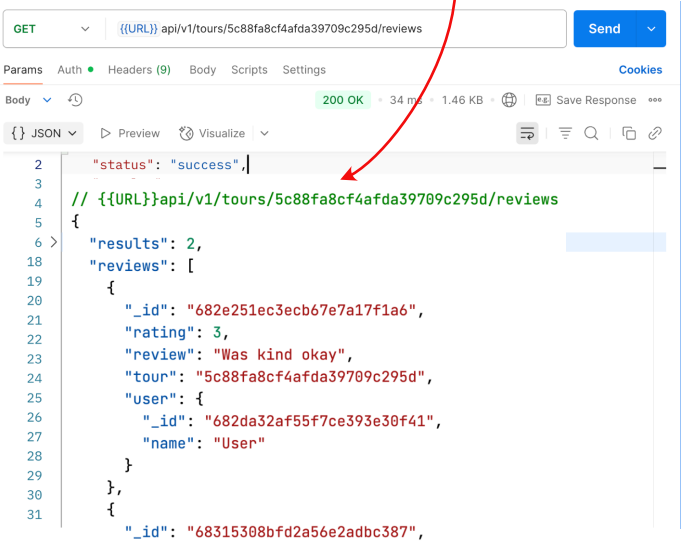
```
1  const express = require('express');
2  const reviewRouter = require('./../routes/rev
3
4  const router = express.Route;
5
6  router.use('/:tourId/reviews', reviewRouter);
7
8  module.exports = router;
```

reviewRoute

```
1  const express = require('express');
2  const reviewController = require('./../controllers/re
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.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 });
```



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     },
       . . .
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     },
       . . .
31 );
```

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

tourRoute üzerinden gelen **tourId Review scheması** üzerininden 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.