

GET

http://localhost:3000/api/67130500099/courses

Docs

Params

Authorization

Headers (9)

Body ●

Scripts ●

Settings

Body

Cookies

Headers (7)

Test Results (1/1)

JSON

Preview

Visualize

```

1  {
2    "data": [
3      {
4        "id": 1,
5        "subject_id": 1,
6        "year": 2024
7      },
8      {
9        "id": 2,
10       "subject_id": 1,
11       "year": 2025
12     },
13     {
14       "id": 3,
15       "subject_id": 2,
16       "year": 2025
17     },
18   ]

```

GET

http://localhost:3000/api/67130500099/courses/1/students

Docs

Params

Authorization

Headers (9)

Body ●

Scripts ●

Settings

Body

Cookies

Headers (7)

Test Results (1/1)

JSON

Preview

Visualize

```

1  [
2    {
3      "id": 101,
4      "name": "Somchai"
5    },
6    {
7      "id": 103,
8      "name": "Preeda"
9    },
10   {
11     "id": 104,
12     "name": "Kitichai"
13   }
14 ]

```



REST API basics: CRUD, test & variable / **Update data** • From Update data

POST

http://localhost:3000/api/67130500099/students/102/courses

Docs Params Authorization Headers (9) **Body** Scripts Settings

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

```
1  {
2    "course_id": 3,
3    "grade": 3.5
4  }
```

Body Cookies Headers (7) Test Results (1/1)

JSON Preview Visualize

```
1  {
2    "course_id": 3,
3    "student_id": 102,
4    "grade": 3.5
5  }
```



REST API basics: CRUD, test & variable / **Update data** • From Update data

DELETE

http://localhost:3000/api/67130500099/students/102/courses/3

Docs Params Authorization Headers (9) **Body** Scripts Settings

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL **JSC**

```
1  {
2    "course_id": 3,
3    "grade": 3.5
4  }
```

Body Cookies Headers (4) Test Results (1/1)

Raw Preview Visualize

1

.env

```
7      # server with the prisma dev CLI command, when not choosing any non-default
10     # one found in a remote Prisma Postgres URL, does not contain any sensitive in
11     💡
12     DATABASE_URL="mysql://root:mysql@sit@localhost:3306/ass_67130500099"
```

schema.prisma

```
1  generator client {
2    provider = "prisma-client"
3  }
4
5  datasource db {
6    provider = "mysql"
7    url      = env("DATABASE_URL")
8  }
9
10 model course_student {
11   course_id Int    @db.UnsignedTinyInt
12   student_id Int   @db.UnsignedTinyInt
13   grade     Float
14   courses   courses @relation(fields: [course_id], references: [id], onDelete: NoAction, onUpdate: NoAction, map: "course_student_courses_id_fk")
15   students  students @relation(fields: [student_id], references: [id], onDelete: NoAction, onUpdate: NoAction, map: "course_student_students_id_fk")
16
17   @@id([course_id, student_id])
18   @@index([course_id], map: "idx_fk_course_id")
19   @@index([student_id], map: "idx_fk_student_id")
20 }
21
22 model courses {
23   id          Int          @id @unique @default(autoincrement()) @db.UnsignedTinyInt
24   subject_id  Int
25   year        Int          @db.Year
26   course_student course_student[]
27
28   @@index([subject_id], map: "idx_fk_subject_id")
29 }
30
31
32
33
34
35
36
37 model subjects {
38   id          Int          @id @unique @default(autoincrement()) @db.UnsignedTinyInt
39   subject_code String @unique @db.Char(6)
40   title       String @db.VarChar(40)
41   credit      Float        @default(3)
42 }
```

Courses-repository.js

```
JS courses-repository.js × JS courses-controller.js JS student-controller.js JS CourseSTD.js
1  const {PrismaClient} = require('@prisma/client');
2  const prisma : PrismaClient<PrismaClientOptions, ..., DefaultArgs> = new PrismaClient();
3
4  module.exports = {
5    findAll: async function (
6      pageRequest : {page: number, pageSize: number} = { page:1, pageSize:1 },
7      sortRe : {order: string, sortBy: string} = { sortBy:'id', order:'asc' },
8    ) : Promise<{...}> {
9      const { page : number , pageSize : number } = pageRequest
10     const { sortBy : string , order : string } = sortRe
11
12     const validOrder : string = order === 'desc' ? 'desc' : 'asc'
13
14     const totalItems : number = await prisma.courses.count()
15     const data : (GetFindResult<...>[]) = await prisma.courses.findMany( args: {
16       skip: (page - 1) * pageSize,
17       take: pageSize,
18       orderBy: {
19         [sortBy]: validOrder
20       }
21     })
22
23     return {
24       data,
25       page,
26       pageSize,
27       totalPages: Math.ceil( x: totalItems / pageSize)
28     }
29   },
30
31   findStudentsByCourseId: async function (courseId) : PrismaPromise<...> {
32     return prisma.course_student.findMany( args: {
33       where: { course_id : courseId },
34       include: {
35         students: true
36       }
37     })
38   }
39 }
40
```

student-repository.js

```
ses-controller.js  JS student-controller.js  JS CourseSTD.js  JS app.js  JS www  JS courses-service.js

1  const { PrismaClient } = require('@prisma/client');
2  const prisma : PrismaClient<PrismaClientOptions, ..., DefaultArgs> = new PrismaClient();
3
4  module.exports = {
5    addCourse: async function (courseId, stdId, grade) : Promise<Prisma__course_studentClient<...>> {
6      return prisma.course_student.create({
7        data: {
8          course_id: courseId,
9          student_id: stdId,
10         grade
11       }
12     });
13   },
14   removeCourse: async function (courseId, stdId) : Promise<Prisma__course_studentClient<...>> {
15     return prisma.course_student.delete( args: {
16       where: {
17         course_id_student_id: {
18           course_id: courseId,
19           student_id: stdId
20         }
21       }
22     });
23   },
24   findStudentsByCourseId: async function (courseId) : PrismaPromise<...> {
25     return prisma.course_student.findMany( args: {
26       where: { course_id: courseId },
27       include: { students: true }
28     });
29   }
30 }
```

courses-services.js

```
1  const courseRepo : {...} | {...} = require("../repositories/courses-repository");
2
3  async function getAllCourses(params) : Promise<...> { Show usages
4      return await courseRepo.findAll(params);
5  }
6
7  async function getStdInCourses(courseId) : Promise<...> { Show usages
8      return await courseRepo.findStudentsByCourseId(courseId);
9  }
10
11  module.exports = {getAllCourses, getStdInCourses}
```

students-services.js

```
const stdRepo : {...} | {...} = require("../repositories/student-repository");

async function getStdById(courseId) : Promise<...> { Show usages
    return await stdRepo.findStudentsByCourseId(courseId);
}

async function addCourseToStd(courseId, stdId, grade) : Promise<Prisma__course_studentClient<...>> { Show usages
    return await stdRepo.addCourse(courseId, stdId, grade);
}

async function removeCourse(courseId, stdId) : Promise<Prisma__course_studentClient<...>> { Show usages
    return await stdRepo.removeCourse(courseId, stdId);
}

module.exports = { getStdById, addCourseToStd, removeCourse };
```

courses-controller.js

```
JS courses-controller.js x JS student-controller.js JS CourseSTD.js JS app.js JS www JS courses-service.js JS student-service.js JS student-reposito v
1 const stdService :(.)(.) = require("../services/student-service");
2
3 async function addCourseSTD(req, res) :Promise<...> { Show usages
4   try {
5     const stdId :number = Number(req.params.id);
6     const { course_id, grade } = req.body;
7
8     if (isNaN(stdId) || !course_id || grade == null) {
9       return res.status(400).send({ error: "Missing or Invalid fields" });
10    }
11
12    const result :Prisma__course_studentClient<GetResult<...>, never, DefaultArgs, PrismaClientOptions> = await stdService.addCoursestd(course_id, stdId, grade);
13    res.status(200).json(result);
14
15  } catch (e) {
16    if (e.code === 'P2002') {
17      return res.status(409).send({ error: "Course already exists" });
18    }
19    res.status(500).json({ error: e.message });
20  }
21 }
22
```

```
JS courses-controller.js x JS student-controller.js JS CourseSTD.js JS app.js JS www JS cou
3   }
20 }
21 }
22
23 async function removeCourseSTD(req, res) :Promise<...> { Show usages
24   try {
25     const stdId :number = Number(req.params.id);
26     const course_id :number = Number(req.params.course_id);
27
28     if (isNaN(stdId) || isNaN(course_id)) {
29       return res.status(400).send({ error: "Missing or Invalid fields" });
30     }
31
32     await stdService.removeCourse(course_id, stdId);
33     res.status(204).send();
34
35   } catch (e) {
36     if (e.code === 'P2001') {
37       return res.status(404).json({ error: "course not found" });
38     }
39     res.status(500).json({ error: e.message });
40   }
41 }
42
43 module.exports = { addCourseSTD, removeCourseSTD };
44
```

student-controller.js

```
1  const stdService : Prisma__course_studentClient<GetResult<...>, never, DefaultArgs, PrismaClientOptions> = require("../services/student-service");
2
3  async function addCourseSTD(req, res) : Promise<...> { Show usages
4      try {
5          const stdId : number = Number(req.params.id);
6          const { course_id, grade } = req.body;
7
8          if (isNaN(stdId) || !course_id || grade == null) {
9              return res.status(400).send({ error: "Missing or Invalid fields" });
10         }
11
12         const result : Prisma__course_studentClient<GetResult<...>, never, DefaultArgs, PrismaClientOptions> = await stdService.addCoursestd(course_id, stdId, grade);
13         res.status(200).json(result);
14     } catch (e) {
15         if (e.code === 'P2002') {
16             return res.status(409).send({ error: "Course already exists" });
17         }
18         res.status(500).json({ error: e.message });
19     }
20 }
21
22
```

```
JS courses-controller.js JS student-controller.js × JS CourseSTD.js JS app.js JS www
3  async function addCourseSTD(req, res) : Promise<...> { Show usages
20      }
21  }
22
23
24  async function removeCourseSTD(req, res) : Promise<...> { Show usages
25      try {
26          const stdId : number = Number(req.params.id);
27          const course_id : number = Number(req.params.course_id);
28
29          if (isNaN(stdId) || isNaN(course_id)) {
30              return res.status(400).send({ error: "Missing or Invalid fields" });
31          }
32
33          await stdService.removeCourse(course_id, stdId);
34          res.status(204).send();
35
36      } catch (e) {
37          if (e.code === 'P2001') {
38              return res.status(404).json({ error: "course not found" });
39          }
40          res.status(500).json({ error: e.message });
41      }
42  }
43
44
45
46  module.exports = { addCourseSTD, removeCourseSTD };
47
```


Route/CourseSTD.js

```
JS courses-controller.js JS student-controller.js JS CourseSTD.js x JS app.js JS www JS courses-service.js
1  const express : e | () => core.Express = require('express')
2  const router : Router = express.Router()
3
4  const courseController : {...} | {...} = require('../controllers/courses-controller')
5  const stdController : {...} | {...} = require('../controllers/student-controller')
6
7  router.get( path: '/67130500099/courses', courseController.getCourse)
8  router.get( path: '/67130500099/courses/:id/students', courseController.getStdById)
9
10
11 router.post( path: '/67130500099/students/:id/courses', stdController.addCourseSTD)
12 router.delete( path: '/67130500099/students/:id/courses/:course_id', stdController.removeCourseSTD)
13
14 module.exports = router
```

app.js

```
JS courses-controller.js JS student-controller.js JS CourseSTD.js JS app.js x JS www JS courses-service.js
1  require('dotenv').config();
2  const express : e | () => core.Express = require('express');
3  const app : any | Express = express();
4  const router : Router | {...} = require('./routes/CourseSTD')
5
6  app.use(express.json());
7  app.use('/api', router);
8  app.use((req : Request<RouteParameters<...>, any, any, ParsedQs, Record<...>>, res : Response<any, Record<...>>) : void => {
9      res.status( code: 404 ).send( body: 'Not Found' );
10  })
11
12
13
14 module.exports = app;
15
```