

15-3 分享相关 API

创建分享

我的分享列表

查看分享

保存到自己到网盘

创建分享

- 创建数据迁移表



Bash | 复制代码

```
1 npx sequelize migration:generate --name=share
```

- 定义 database / migrations / 目录下生成的数据迁移文件

```
1  "use strict";
2
3  /** @type {import('sequelize-cli').Migration} */
4  module.exports = {
5    async up(queryInterface, Sequelize) {
6      const { INTEGER, STRING, DATE, ENUM, TEXT } = Sequelize;
7      return queryInterface.createTable("share", {
8        id: {
9          type: INTEGER(20),
10         primaryKey: true,
11         autoIncrement: true,
12       },
13       sharedurl: {
14         type: STRING,
15         allowNull: true,
16         defaultValue: "",
17         comment: "分享链接",
18       },
19       file_id: {
20         type: INTEGER,
21         allowNull: false,
22         defaultValue: 0,
23         comment: "文件id",
24         references: {
25           model: "file",
26           key: "id",
27         },
28         onDelete: "cascade",
29         onUpdate: "restrict", // 更新时操作
30       },
31       iscancel: {
32         type: INTEGER(1),
33         allowNull: false,
34         defaultValue: 0,
35         comment: "是否取消分享",
36       },
37       user_id: {
38         type: INTEGER,
39         allowNull: false,
40         defaultValue: 0,
41         comment: "用户id",
42         references: {
43           model: "user",
44           key: "id",
45         },
46       },
47     },
48   };
49 }
```

```
46         onDelete: "cascade",
47         onUpdate: "restrict", // 更新时操作
48     },
49     created_time: DATE,
50     updated_time: DATE,
51 });
52 },
53 },
54 async down(queryInterface, Sequelize) {
55     return queryInterface.dropTable("share");
56 },
57 };
```

- 执行数据库变更



Bash | 复制代码

```
1 npx sequelize db:migrate
```

- model 包创建 share.js 数据模型

```
1  "use strict";
2  const crypto = require("crypto");
3  module.exports = (app) => {
4    const { STRING, INTEGER, DATE, ENUM, TEXT } = app.Sequelize;
5
6    const Share = app.model.define("share", {
7      id: {
8        type: INTEGER(20),
9        primaryKey: true,
10       autoIncrement: true,
11     },
12     sharedurl: {
13       type: STRING,
14       allowNull: true,
15       defaultValue: "",
16       comment: "分享链接",
17     },
18     file_id: {
19       type: INTEGER,
20       allowNull: false,
21       defaultValue: 0,
22       comment: "文件id",
23       references: {
24         model: "file",
25         key: "id",
26       },
27       onDelete: "cascade",
28       onUpdate: "restrict", // 更新时操作
29     },
30     iscancel: {
31       type: INTEGER(1),
32       allowNull: false,
33       defaultValue: 0,
34       comment: "是否取消分享",
35     },
36     user_id: {
37       type: INTEGER,
38       allowNull: false,
39       defaultValue: 0,
40       comment: "用户id",
41       references: {
42         model: "user",
43         key: "id",
44       },
45       onDelete: "cascade",
```

```
46     onUpdate: "restrict", // 更新时操作
47   },
48   created_time: DATE,
49   updated_time: DATE,
50 });
51
52 Share.associate = function (models) {
53   // 关联文件
54   Share.belongsTo(app.model.File);
55 };
56
57 return Share;
58 };
59
```

- 控制器: app/controller/share.js

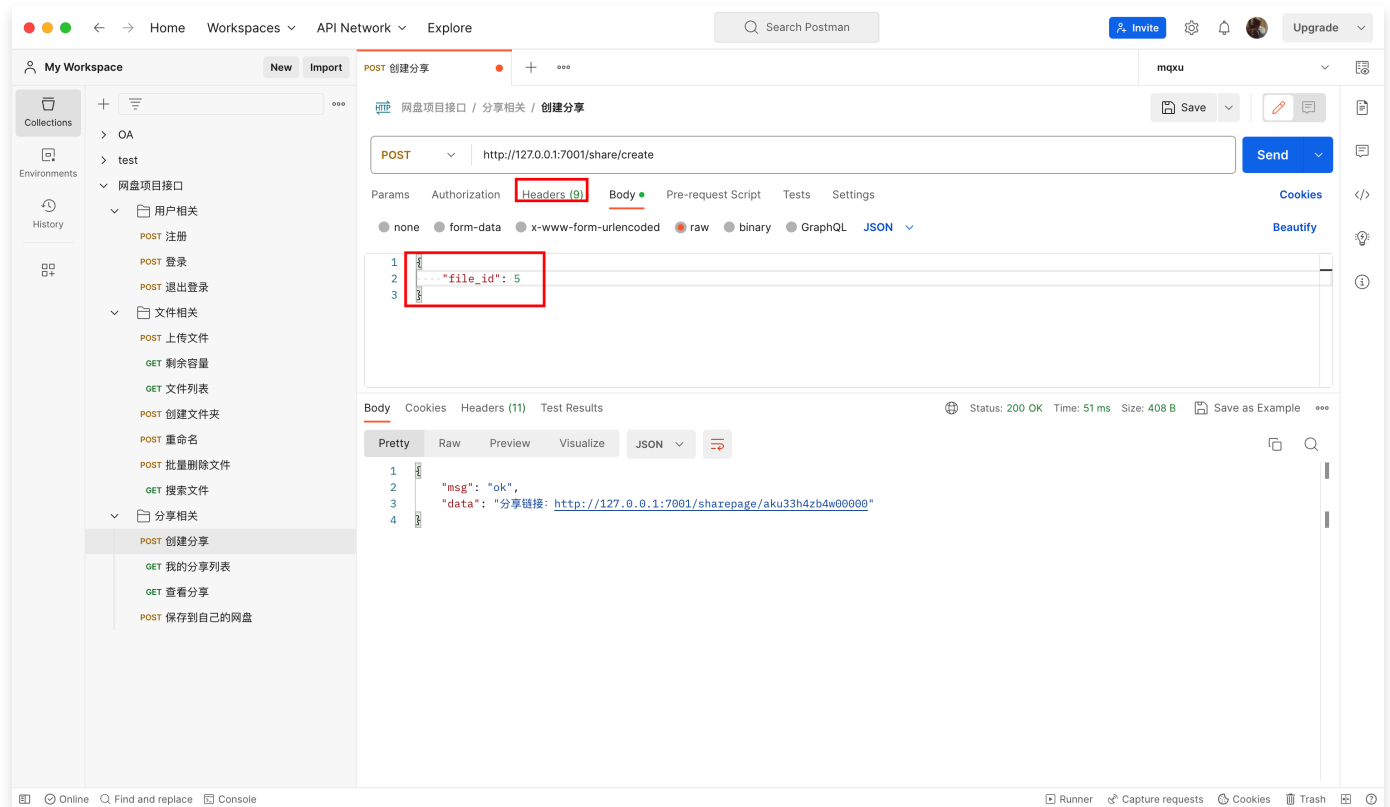
```
1  "use strict";
2
3  const Controller = require("egg").Controller;
4
5  class ShareController extends Controller {
6    // 创建分享
7    async create() {
8      const { ctx, app, service } = this;
9      let user_id = ctx.authUser.id;
10
11      ctx.validate({
12        file_id: {
13          type: "int",
14          required: true,
15          desc: "文件ID",
16        },
17      });
18
19      let { file_id } = ctx.request.body;
20
21      let f = await app.model.File.findOne({
22        where: {
23          id: file_id,
24          user_id,
25        },
26      });
27
28      if (!f) {
29        return ctx.throw(404, "文件不存在");
30      }
31
32      let sharedurl = ctx.genID(15);
33
34      let s = await app.model.Share.create({
35        sharedurl,
36        file_id,
37        iscancel: 0,
38        user_id,
39      });
40
41      let url = "http://127.0.0.1:7001/sharepage/" + sharedurl;
42      ctx.apiSuccess("分享链接: " + url);
43    }
44  }
45
```

```
46 module.exports = ShareController;
```

- 路由

```
1 // 创建分享
2 router.post("/share/create", controller.share.create);
```

测试，带 token 传需要分享的文件的主键



```
commit ba2dd0f0dedffd70c7ef26f10e0a8120e405898 (HEAD -> main)
Author: mqxu <moqi1977@gmail.com>
Date: Tue Jul 4 23:48:41 2023 +0800
```

创建分享

我的分享列表

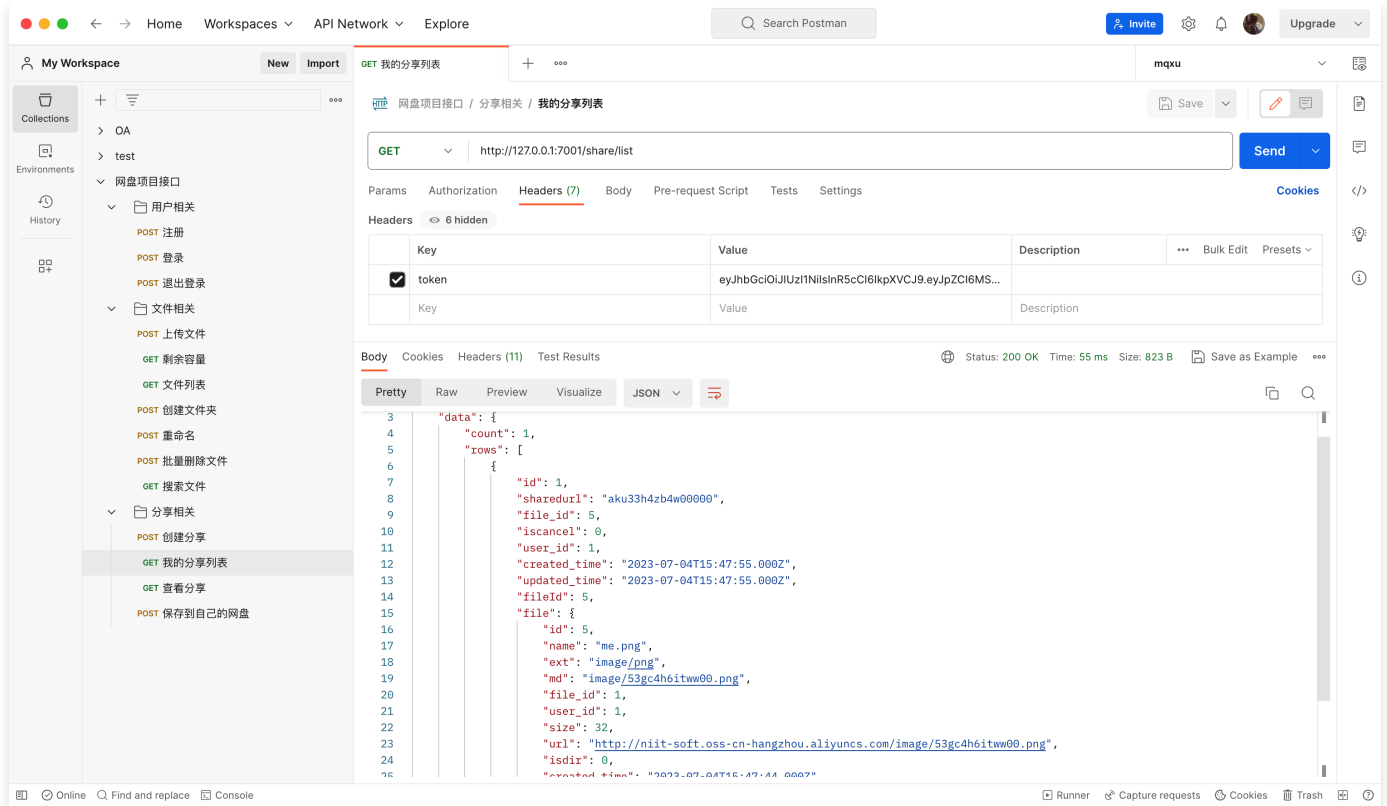
- 控制器: app/controller/share.js

```
1 // 我的分享列表
2 async list() {
3   const { ctx, app } = this
4   const user_id = ctx.authUser.id
5
6   let list = await app.model.Share.findAndCountAll({
7     where: {
8       user_id,
9     },
10    include: [
11      {
12        model: app.model.File,
13      },
14    ],
15  })
16
17   ctx.apiSuccess(list)
18 }
```

- 路由: app/router.js

```
1 // 我的分享列表
2 router.get('/share/list', controller.share.list);
```

- 测试带 token 请求, 可以看到刚才分享过的文件



```
commit dabe25e7b6ebd6ab8f04466ef607b30e08ac1a04 (HEAD -> main)
Author: mxqu <moqi1977@gmail.com>
Date: Tue Jul 4 23:52:00 2023 +0800
```

我的分享列表

查看分享

- 控制器: app/controller/share.js

```
1 // 查看分享
2 async read() {
3   const { ctx, app, service } = this
4   let sharedurl = ctx.params.sharedurl
5   if (!sharedurl) {
6     return ctx.apiFail('非法参数')
7   }
8
9   let file_id = ctx.query.file_id
10
11   // 分享是否存在
12   let s = await service.share.isExist(sharedurl)
13
14   let where = {
15     user_id: s.user_id,
16   }
17
18   if (!file_id) {
19     where.id = s.file_id
20   } else {
21     where.file_id = file_id
22   }
23
24   let rows = await app.model.File.findAll({
25     where,
26     order: [['isdir', 'desc']],
27   })
28
29   ctx.apiSuccess(rows)
30 }
```

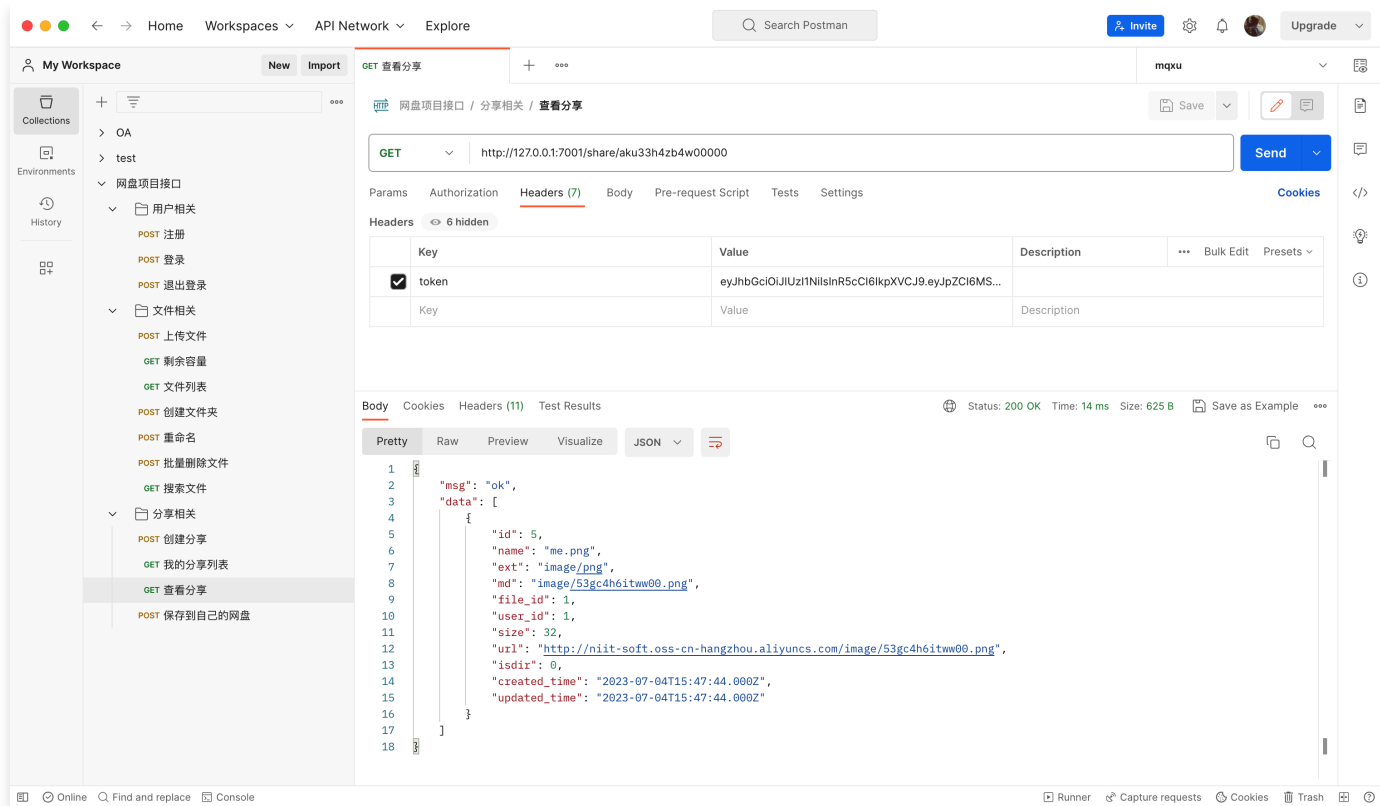
- 服务: app/service/share.js

```
1  'use strict'
2
3  const Service = require('egg').Service
4
5  class ShareService extends Service {
6    async isExist(sharedurl, options = {}) {
7      let s = await this.app.model.Share.findOne({
8        where: {
9          sharedurl,
10         iscancel: 0,
11       },
12       ...options,
13     })
14
15     if (!s) {
16       return this.ctx.throw(404, '该分享已失效')
17     }
18
19     return s
20   }
21 }
22
23 module.exports = ShareService
```

- 路由: app/router.js

```
1  // 查看分享
2  router.get('/share/:sharedurl', controller.share.read);
```

- 测试, 带上之前创建成功的分享链接作为路径参数



```
commit 5cb75d55088d95eaeadb8925d010ffd3d8498f98 (HEAD -> main)
```

```
Author: mxqu <moqi1977@gmail.com>
```

```
Date: Tue Jul 4 23:56:43 2023 +0800
```

查看分享

保存到自己到网盘

控制器：app/controller/share.js

```
1 // 保存到自己的网盘
2 async saveToSelf() {
3   const { ctx, app, service } = this;
4   let current_user_id = ctx.authUser.id;
5
6   ctx.validate({
7     dir_id: {
8       type: "int",
9       required: true,
10      desc: "目录ID",
11    },
12    sharedurl: {
13      type: "string",
14      required: true,
15      desc: "分享标识",
16    },
17  });
18
19   let { dir_id, sharedurl } = ctx.request.body;
20
21   // 分享是否存在
22   let s = await service.share.isExist(sharedurl, {
23     include: [
24       {
25         model: app.model.File,
26       },
27     ],
28   });
29   if (s.user_id === current_user_id) {
30     return ctx.apiSuccess("本人分享, 无需保存");
31   }
32
33   // 文件是否存在
34   if (dir_id > 0) {
35     await service.file.isDirExist(dir_id);
36   }
37
38   // 查询该分享目录下的所有数据
39   let getAllFile = async (obj, dirId) => {
40     let data = {
41       name: obj.name,
42       ext: obj.ext,
43       md: obj.md,
44       file_id: dirId,
45       user_id: current_user_id,
```

```

46         size: obj.size,
47         isdir: obj.isdir,
48         url: obj.url,
49     };
50     // 判断当前用户剩余空间
51     if (ctx.authUser.total_size - ctx.authUser.used_size < data.size) {
52         return ctx.throw(400, "你的可用内存不足");
53     }
54
55     // 直接创建
56     let o = await app.model.File.create(data);
57
58     // 更新user表的使用内存
59     ctx.authUser.used_size = ctx.authUser.used_size + parseInt(data.size
60 );
61     await ctx.authUser.save();
62
63     // 目录
64     if (obj.isdir) {
65         // 继续查询下面其他的数据
66         let rows = await app.model.File.findAll({
67             where: {
68                 user_id: obj.user_id,
69                 file_id: obj.id,
70             },
71         });
72         rows.forEach((item) => {
73             getAllFile(item, o.id);
74         });
75         return;
76     }
77 };
78
79     await getAllFile(s.file, dir_id);
80
81     ctx.apiSuccess("ok");
82 }

```

- 路由: app/router.js

```

1 // 保存到自己网盘
2 router.post('/share/save_to_self', controller.share.saveToSelf);

```

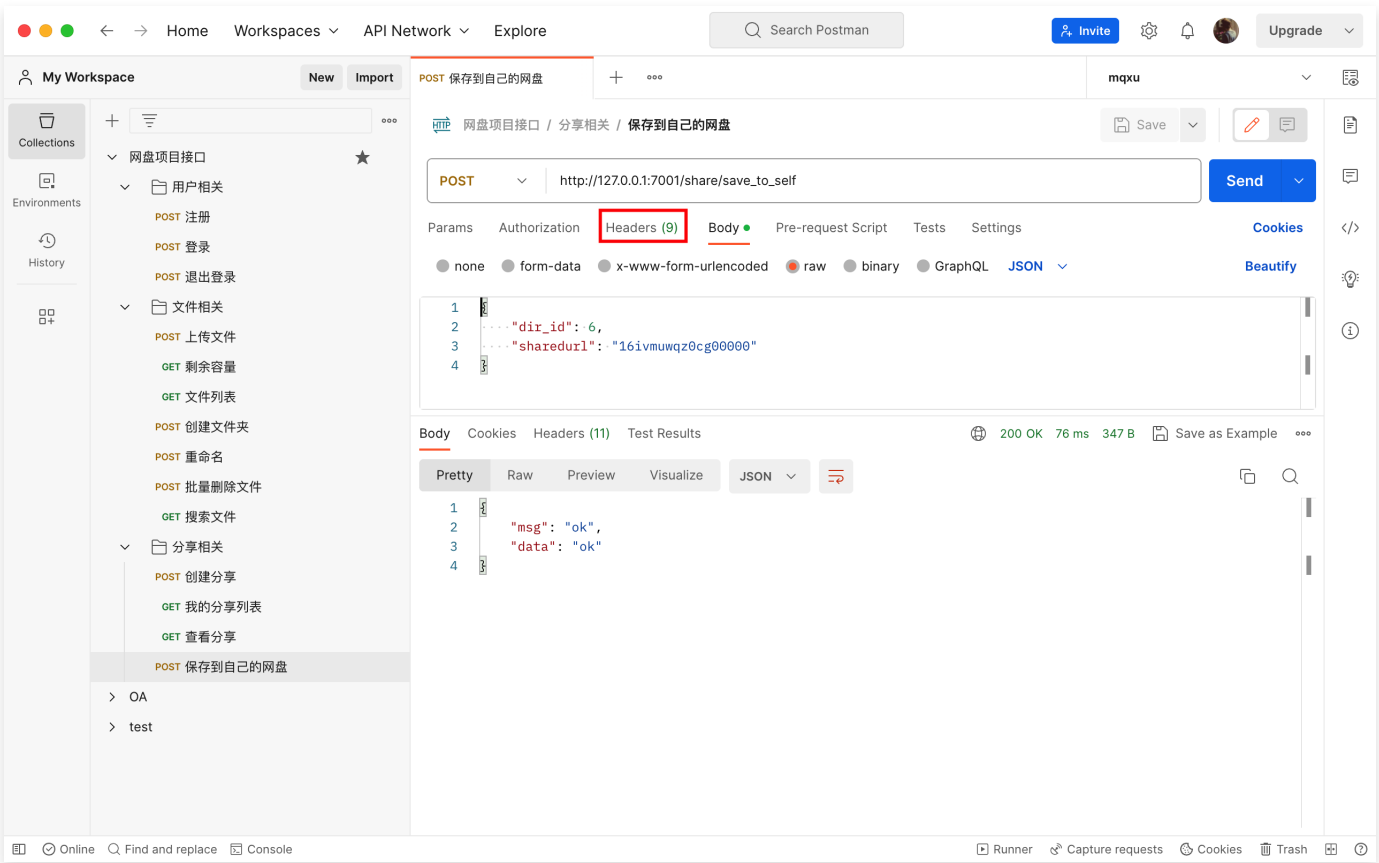
测试

注册一个用户，登录，后续操作都带上这个用户的 token

新建一个目录

用这个 2 号用户发起“保存到自己网盘”的请求，需要传递两个参数

- 保存到自己哪个目录
- 分享的 url



数据库

id	name	ext	md	file_id	user_id	size	url	isdir	created_time	updated_time
1	image			0	1	0		1	2023-07-04 23:02:37	2023-07-04 23:02:37
5	me.png	image/png	image/53gc4h6itww00.pn	1	1	32	http://niit-soft.oss-cn-hangzhou.aliyu	0	2023-07-04 23:47:44	2023-07-04 23:47:44
6	avatar			0	2	0		1	2023-07-05 11:59:40	2023-07-05 11:59:40
7	img11.jpg	image/jpeg	image/hmqpt6nspgw00.jp	1	1	18	http://niit-soft.oss-cn-hangzhou.aliyu	0	2023-07-05 12:04:01	2023-07-05 12:04:01
8	img9.jpg	image/jpeg	image/4fv0l7y1nb400.jpg	1	1	18	http://niit-soft.oss-cn-hangzhou.aliyu	0	2023-07-05 12:04:07	2023-07-05 12:04:07
9	img7.jpg	image/jpeg	image/8lclh7fdpos00.jpg	1	1	17	http://niit-soft.oss-cn-hangzhou.aliyu	0	2023-07-05 12:04:16	2023-07-05 12:04:16
10	img9.jpg	image/jpeg	avatar/e2rig0mqd8g00.jp	6	2	58	http://niit-soft.oss-cn-hangzhou.aliyu	0	2023-07-05 12:23:09	2023-07-05 12:23:09
11	img16.jpg	image/jpeg	avatar/ahew14vjfoo00.jpg	6	2	60	http://niit-soft.oss-cn-hangzhou.aliyu	0	2023-07-05 12:23:21	2023-07-05 12:23:21
12	img7.jpg	image/jpeg	image/8lclh7fdpos00.jpg	6	2	17	http://niit-soft.oss-cn-hangzhou.aliyu	0	2023-07-08 16:06:20	2023-07-08 16:06:20

```
commit e19b7fc22c5e936f7056a079b434f9fabfb15a9c (HEAD -> main)
Author: mqxu <moqi1977@gmail.com>
Date: Sat Jul 8 16:08:54 2023 +0800

保存到自己到网盘
```