REALMS

GitHub Repository: hakula139 / REALMS

REALMS Establishes A Library Management System, written in Go, using a MySQL database.

Getting started

0. Prerequisities

To set up the environment, you need to have the following dependencies installed.

- Go 1.14 or above
- GNU make 4.0 or above
- MySQL 5.7 or above / MariaDB 10.4 or above

For Windows, try MinGW-w64.

1. Installation

First, you need to obtain the REALMS package.

```
git clone https://github.com/hakula139/REALMS.git
cd REALMS
```

Then you can build the project using Make.

```
make build
```

For MinGW-w64 on Windows, use the command below.

```
mingw32-make build
```

You should see the following output, which indicates a successful installation.

```
build: realms done.
build: realmsd done.
```

2. Usage

2.1 realmsd

Run `realmsd` using the command below, and the server will listen to port `7274` by default.

```
./bin/realmsd
```

`realmsd` will open a database connection to a MySQL database, originally at `root:Hakula@tcp(localhost:3306)/library`. You can modify the configuration in the config file `./configs/db_config.json`. There's no need to manually create a database named `library`, as it'll be created automatically in advance.

2.2 realms

To interact with the back end, here's a simple CLI tool, namely, `realms`. Though, it's not necessarily required, since you can easily build another front end with the RESTful APIs, a guide to which will be provided later.

Run `realms` using the command below.

```
./bin/realms
```

You should see the welcome message.

```
Welcome to REALMS! Check the manual using the command 'help'.
>
```

To get started with REALMS, use the command `help` to show all available commands.

```
> help
```

```
COMMANDS:
   Public:
      help           Shows a list of commands
      exit           Quit

      login          Log in to your library account
      logout         Log out of your library account
      status         Shows the current login status

      show books     Shows all books in the library
      show book      Shows the book of given ID
      find books     Finds books by title / author / ISBN

   Admin privilege required:
      add book       Adds a new book to the library
      update book    Updates data of a book
      remove book    Removes a book from the library

      add user       Adds a new user to the database
      update user    Updates data of a user
      remove user    Removes a user from the database
      show users     Shows all users in the library
      show user      Shows the user of given ID

   User privilege required:
      me             Shows the current logged-in user

      borrow book    Borrows a book from the library
      return book    Returns a book to the library
      check ddl      Checks the deadline to return a book
      extend ddl     Extends the deadline to return a book
      show list      Shows all books that you've borrowed
      show overdue   Shows all overdue books that you've borrowed
      show history   Shows all records
```

It's quite easy to understand how these commands work, nevertheless we're going to talk about them in the next chapter.

3. REST API

Here we'll demonstrate the usage of these RESTful APIs by example.

3.1 Log in

3.1.1 Request

Method: `POST /login`
Content-Type: `multipart/form-data`
CLI command: `login`

In `realms`:

```
> login
Enter Username: Hakula
Enter Password:
```

You'll be required to enter your username and password (FYI, the password is invisible while typing). There's no `signup` in REALMS, so a user account can only be acquired from an admin.

To authenticate a user's credentials, REALMS uses the session. In the implementation of `realms`, a session cookie is used to store the essential information, and the cookies are handled by cookiejar.

On the server-side, the password will be hashed using bcrypt before save.

3.1.2 Response

Status: `200 OK`
Content-Type: `application/json`

```
{"data": true}
```

In case of a successful login, you'll receive a welcome message.

```
Welcome Hakula!
```

Otherwise, an error will be returned. If the server didn't return a response, `realms` will print the following message.

```
cli: failed to make an http request, did you start realmsd?
```

Other possible error messages are shown below.

```
auth: user not exist
auth: incorrect password
auth: already logged in
auth: failed to save session
```

3.2 Log out

3.2.1 Request

Method: `GET /logout`
CLI command: `logout`

In `realms`:

```
> logout
```

3.2.2 Response

Status: `200 OK`
Content-Type: `application/json`

```
{"data": true}
```

In case of a successful logout, you'll receive a success message.

```
Successfully logged out!
```

Otherwise, an error will be returned. Possible error messages are shown below.

```
auth: invalid session token, have you logged in?
auth: failed to save session
```

3.3 Show current logged-in user

3.3.1 Request

Method: `GET /user/me`
CLI command: `me`

In `realms`:

```
> me
```

User privilege is required, which means you have to login before doing this operation.

3.3.2 Response

Status: `200 OK`
Content-Type: `application/json`

```
{"data": 3}
```

Normally, your user ID will be returned.

```
Current user ID: 3
```

If you're not logged in, you'll receive an error message below.

```
auth: unauthorized
```

3.4 Show the current login status

3.4.1 Request

Method: `GET /status`
CLI command: `status`

In `realms`:

```
> status
```

3.4.2 Response

Status: `200 OK`
Content-Type: `application/json`

```
{"data": true}
```

If the current user is logged in, you'll see the following message.

```
Online
```

3.5 Add a new book

3.5.1 Request

Method: `POST /admin/books`
Content-Type: `application/json`
CLI command: `add book`

```
{
  "title": "CS:APP",
  "author": "Randal E. Bryant",
  "publisher": "Pearson",
  "isbn": "978-0134092669"
}
```

In `realms`:

```
> add book
Title (required): CS:APP
Author (optional): Randal E. Bryant
Publisher (optional): Pearson
ISBN (optional): 978-0134092669
```

Admin privilege is required. In REALMS, we use `level` to indicate a user's privilege, which is a property of the user model. When a user makes a request, the server will check if he/she has admin privilege. If not, an Unauthorized Error will be returned.

You'll be required to input the necessary information of the book, and the `title` field should not be blank, or an error will be returned. To skip an optional field in `realms`, simply press Enter.

On the server-side, the following message will be written to log using zap. The default path to the log file is `./logs/realmsd.log`, which can be modified in the config file `./configs/log_config.json`.

```
{"level":"info","time":"2020-05-04T01:19:11.206+0800","msg":"Added book 20"}
```

3.5.2 Response

Status: `200 OK`
Content-Type: `application/json`

```
{
  "data": {
    "id": 20,
    "title": "CS:APP",
    "author": "Randal E. Bryant",
    "publisher": "Pearson",
    "isbn": "978-0134092669"
  }
}
```

The complete information of the added book will be returned, since you may want to display it in your front-end application. For the sake of simplicity, here `realms` will just print the book ID.

```
Successfully added book 20
```

If not authorized (i.e. you're not an admin), you'll receive an error message below.

```
auth: unauthorized
```

3.6 Update data of a book

3.6.1 Request

Method: `PATCH /admin/books/:id`
Content-Type: `application/json`
CLI command: `update book`

```
{
  "title": "Computer Systems",
  "author": "Randal E. Bryant, David R. O'Hallaron",
}
```

In `realms`:

```
> update book
Book ID: 20
Title (optional): Computer Systems
Author (optional): Randal E. Bryant, David R. O'Hallaron
Publisher (optional):
ISBN (optional):
```

Admin privilege is required.

Here `:id` refers to the book ID, which `realms` will prompt the user for input at the beginning.

Simply sending a request including just the fields that you want to update is fine, and empty values will be omitted. Still, there's an input checker for all inputs on the server-side, which will validate your request body to prevent invalid requests.

The following message will be written to log.

```
{"level":"info","time":"2020-05-04T01:41:57.908+0800","msg":"Updated book 20"}
```

3.6.2 Response

Status: `200 OK`
Content-Type: `application/json`

```
{
  "data": {
    "id": 20,
    "title": "Computer Systems",
    "author": "Randal E. Bryant, David R. O'Hallaron",
    "publisher": "Pearson",
    "isbn": "978-0134092669"
  }
}
```

The updated data will be returned.

```
Successfully updated book 20
```

Possible error messages are shown below.

```
auth: unauthorized
database: book not found
```

3.7 Remove a book

3.7.1 Request

Method: `DELETE /admin/books/:id`
Content-Type: `application/json`
CLI command: `remove book`

```
{"message": "Book lost"}
```

In `realms`:

```
> remove book
Book ID: 5
Explanation (optional): Book lost
```

Admin privilege is required.

The `message` field is optional, which is the explanation why you remove the book.

The following message will be written to log.

```
{"level":"info","time":"2020-05-04T02:13:00.956+0800","msg":"Removed book 5 with explanation: Book lost"}
```

Or if there's no explanation:

```
{"level":"info","time":"2020-05-04T02:13:00.956+0800","msg":"Removed book 5"}
```

3.7.2 Response

Status: `200 OK`
Content-Type: `application/json`

```
{"data": true}
```

Since the book has already been removed, there's no need to return its information.

```
Successfully removed book 5
```

Possible error messages are shown below.

```
auth: unauthorized
database: book not found
```

3.8 Show all books

3.8.1 Request

Method: `GET /books`
CLI command: `show books`

In `realms`:

```
> show books
```

3.8.2 Response

Status: `200 OK`
Content-Type: `application/json`

```json
{
  "data": [
    {
      "id": 12,
      "title": "A Byte of Python",
      "author": "Swaroop C H",
      "publisher": "",
      "isbn": ""
    },
    {
      "id": 20,
      "title": "Computer Systems",
      "author": "Randal E. Bryant, David R. O'Hallaron",
      "publisher": "Pearson",
      "isbn": "978-0134092669"
    },
    {
      "id": 22,
      "title": "Operating Systems: Three Easy Pieces",
      "author": "Andrea C. Arpaci-Dusseau, Remzi H. Arpaci-Dusseau",
      "publisher": "CreateSpace Independent Publishing Platform",
      "isbn": "978-1985086593"
    }
  ]
}
```

We expect the following output, aligned in table style.

```
ID      Title                     Author                    Publisher                 ISBN
-----------------------------------------------------------------------------------------------------------------------
12      A Byte of Python          Swaroop C H
20      Computer Systems          Randal E. Bryant, David   Pearson                   978-0134092669
22      Operating Systems: Thre   Andrea C. Arpaci-Dussea   CreateSpace Independent   978-1985086593
```

Here the column width can be customized in `realms`, which is `25` by default. Overflowed content will be hidden.

If there's no book found, `realms` will print the following message.

```
No books found
```

In the implementation of `realms`, when handling distinct responses, generally we use an `interface{}` to represent the unknown data type (which can be `bool`, `int`, `string`, `map[string]interface{}`). However, when it comes to this response, the returned data is in fact a `[]map[string]interface{}`, which cannot be asserted directly. So here's a workaround.

```go
// ShowBooks shows all books in the library
func ShowBooks() error {
  // ...

  // Extracts the data
  // dataBody is of type interface{}
  if dataBody, ok := data["data"]; ok {
    books := dataBody.([]interface{}) // asserts to []interface{} first
    printBooks(books)
  }
  return nil
}

func printBooks(books []interface{}) {
  // ...
  for _, elem := range books {
    book := elem.(map[string]interface{}) // asserts the elements later
    // ...
  }
}
```

3.9 Show the book of given ID

3.9.1 Request

Method: `GET /books/:id`
CLI command: `show book`

In `realms`:

```
> show book
Book ID: 20
```

3.9.2 Response

Status: `200 OK`
Content-Type: `application/json`

```json
{
  "data": {
    "id": 20,
    "title": "Computer Systems",
    "author": "Randal E. Bryant, David R. O'Hallaron",
    "publisher": "Pearson",
    "isbn": "978-0134092669"
  }
}
```

Output:

```
Book 20
   Title:     Computer Systems
   Author:    Randal E. Bryant, David R. O'Hallaron
   Publisher: Pearson
   ISBN:      978-0134092669
```

Possible error messages are shown below.

```
database: book not found
```

3.10 Find books by title / author / ISBN

3.10.1 Request

Method: `POST /books/find`
Content-Type: `application/json`
CLI command: `find books`

To search by title (fuzzy, case-insensitive):

```
{"title": "system"}
```

To search by author (exact, case-insensitive):

```
{"author": "Randal E. Bryant, David R. O'Hallaron"}
```

To search by ISBN (exact, case-insensitive):

```
{"isbn": "978-0134092669"}
```

To search by title and author:

```
{
  "title": "foo",
  "author": "bar"
}
```

etc.

In `realms`:

```
> find books
Title (optional): system
Author (optional):
ISBN (optional):
```

3.10.2 Response

Status: `200 OK`
Content-Type: `application/json`

```
{
  "data": [
    {
      "id": 20,
      "title": "Computer Systems",
      "author": "Randal E. Bryant, David R. O'Hallaron",
      "publisher": "Pearson",
      "isbn": "978-0134092669"
    },
    {
      "id": 22,
      "title": "Operating Systems: Three Easy Pieces",
      "author": "Andrea C. Arpaci-Dusseau, Remzi H. Arpaci-Dusseau",
      "publisher": "CreateSpace Independent Publishing Platform",
      "isbn": "978-1985086593"
    }
  ]
}
```

Output:

```
ID      Title                    Author                   Publisher                ISBN
-----------------------------------------------------------------------------------------------------------
20      Computer Systems         Randal E. Bryant, David  Pearson                  978-0134092669
22      Operating Systems: Thre  Andrea C. Arpaci-Dussea  CreateSpace Independent  978-1985086593
```

If there's no book found, `realms` will print the following message.

```
No books found
```

3.11 Add a new user

3.11.1 Request

Method: `POST /admin/users`
Content-Type: `application/json`
CLI command: `add user`

```
{
  "username": "Guest",
  "password": "123456",
  "level": 1
}
```

In `realms`:

```
> add user
Enter Username: Guest
Enter Password:
Enter Password again:
(1: User, 2: Admin, 3: Super Admin)
Enter Privilege Level: 1
```

Admin privilege is required.

The username should be unique. As for the privilege level:

Level	Privilege
1	User
2	Admin
3	Super Admin

The following message will be written to log.

```
{"level":"info","time":"2020-05-05T14:44:18.319+0800","msg":"Added user 11"}
```

3.11.2 Response

Status: `200 OK`
Content-Type: `application/json`

```
{
  "data": {
    "id": 11,
    "username": "Guest",
    "password": "$2a$10$wUGgnk03qDQwQNg0c722GuUm4oGbcG5GpC9vAqgAKxbfJ3jt8usYq",
    "level": 1
  }
}
```

Output:

```
Successfully added user 11
```

Possible error messages are shown below.

```
auth: unauthorized
database: username already exists
```

3.12 Update data of a user

3.12.1 Request

Method: `PATCH /admin/users/:id`
Content-Type: `application/json`
CLI command: `update user`

```
{
  "password": "000000",
  "level": 2
}
```

In `realms`:

```
> update user
User ID: 11
Enter Password:
Enter Password again:
(1: User, 2: Admin, 3: Super Admin)
Enter Privilege Level: 2
```

Admin privilege is required.

Here `:id` refers to the user ID. The `level` field is optional.

The following message will be written to log.

```
{"level":"info","time":"2020-05-05T15:11:07.467+0800","msg":"Updated user 11"}
```

3.12.2 Response

Status: `200 OK`
Content-Type: `application/json`

```
{
  "data": {
    "id": 11,
    "username": "Guest",
    "password": "$2a$10$AKXBbTkngAwdW8SQXkswu.5mgOMcJZB80YtVz6M3pA2nK8UIjOxCO",
    "level": 2
  }
}
```

Output:

```
Successfully updated user 11
```

Possible error messages are shown below.

```
auth: unauthorized
database: user not found
```

3.13 Remove a user

3.13.1 Request

Method: `DELETE /admin/users/:id`
CLI command: `remove user`

In `realms`:

```
> remove user
User ID: 11
```

Admin privilege is required.

The following message will be written to log.

```
{"level":"info","time":"2020-05-05T15:20:12.451+0800","msg":"Removed user 11"}
```

3.13.2 Response

Status: `200 OK`
Content-Type: `application/json`

```
{"data": true}
```

Output:

```
Successfully removed user 11
```

Possible error messages are shown below.

```
auth: unauthorized
database: user not found
```

3.14 Show all users

3.14.1 Request

Method: `GET /admin/users`
CLI command: `show users`

In `realms`:

```
> show users
```

Admin privilege is required.

3.14.2 Response

Status: `200 OK`
Content-Type: `application/json`

```
{
  "data": [
    {
      "id": 3,
      "username": "Hakula",
      "password": "$2a$10$XEh0dNu4eNOJqXaf0Z.dVeHceZOU7gOaOqI8tXdy9dVXyskBFP5Hm",
      "level": 3
    },
    {
      "id": 5,
      "username": "Alukah",
      "password": "$2a$10$NogyoGcBYGDbOmjwI8L6Iui303oq4A2bEx7HFQitfsLxweU2BxoDK",
      "level": 1
    }
  ]
}
```

It's obvious that we don't need to display the hashed passwords here.

```
ID      Username                  Level
---------------------------------------
3       Hakula                    3
5       Alukah                    1
```

Possible error messages are shown below.

```
auth: unauthorized
```

3.15 Show the user of given ID

3.15.1 Request

Method: `GET /admin/users/:id`
CLI command: `show user`

In `realms`:

```
> show user
User ID: 3
```

Admin privilege is required.

3.15.2 Response

Status: `200 OK`
Content-Type: `application/json`

```
{
  "data": {
    "id": 3,
    "username": "Hakula",
    "password": "$2a$10$XEh0dNu4eNOJqXaf0Z.dVeHceZOU7gOaOqI8tXdy9dVXyskBFP5Hm",
    "level": 3
  }
}
```

Output:

```
User 3
   Username: Hakula
   Level:    3
```

Possible error messages are shown below.

```
auth: unauthorized
database: user not found
```

3.16 Borrow a book

3.16.1 Request

Method: `POST /user/books/:id`
Content-Type: `application/json`

CLI command: `borrow book`

```
{"borrow_date": "2020-01-01T12:00:00Z"}
```

In `realms`:

- Debug mode: `true`

```
> borrow book
Book ID: 20
(Format: yyyy-mm-dd)
Borrow date: 2020-01-01
(Format: hh:mm:ss)
Borrow time: 12:00:00
```

- Debug mode: `false`

```
> borrow book
Book ID: 20
```

User privilege is required.

Here we add a debug mode for testing purposes, since it's impossible to keep waiting for several weeks until the borrowed book is overdue. When debug mode is enabled, a user can input the borrowing date manually, otherwise it will be set to the current date and time.

The following message will be written to log.

```
{"level":"info","time":"2020-05-05T15:50:00.395+0800","msg":"User 5 borrowed book 20"}
```

3.16.2 Response

Status: `200 OK`
Content-Type: `application/json`

```
{
  "data": {
    "id": 15,
    "user_id": 5,
    "book_id": 20,
    "return_date": "2020-01-15T12:00:00Z",
    "extend_times": 0,
    "real_return_date": null
  }
}
```

The default return date is `14` days after the borrowing date. You may change it in the config file `./configs/library_config.json`.

```
Successfully borrowed book 20
Your return date is: 2020-01-15T12:00:00Z
```

If a user has more than `3` overdue books not returned (the limit may also be customized), his/her account will be suspended, which means he/she is no longer allowed to borrow another book before returning the overdue books. In that case, an error will be returned.

```
library: too many overdue books
```

If the book has already been borrowed by the current user before, here comes another error.

```
library: book already borrowed
```

Other possible error messages are shown below.

```
auth: unauthorized
database: book not found
```

3.17 Return a book

3.17.1 Request

Method: `DELETE /user/books/:id`
CLI command: `return book`

In `realms`:

```
> return book
Book ID: 20
```

User privilege is required.

In the implementation of `realmsd`, the record is soft deleted, which means it will not be actually removed from the table. Instead, we use a `delete_at` column to store the time when the record is removed (the book is returned). Therefore, these records are temporarily ignored in most queries, but can still be obtained using the command `show history`, which we will talk about later.

The following message will be written to log.

```
{"level":"info","time":"2020-05-05T17:18:07.279+0800","msg":"User 5 returned book 20"}
```

3.17.2 Response

Status: `200 OK`
Content-Type: `application/json`

```
{"data": true}
```

Output:

```
Successfully returned book 20
```

If the book has not been borrowed by the current user before, an error will be returned.

```
library: book not borrowed
```

Other possible error messages are shown below.

```
auth: unauthorized
```

Why there's not a `book not found` error here? It's to prevent the case that an admin removed a book which had been borrowed, and now the user who borrowed it wants to return it back.

3.18 Check the deadline to return a book

3.18.1 Request

Method: `GET /user/books/:id`
CLI command: `check ddl`

In `realms`:

```
> check ddl
Book ID: 22
```

User privilege is required.

3.18.2 Response

Status: `200 OK`
Content-Type: `application/json`

```
{
  "data": {
    "id": 30,
    "user_id": 5,
    "book_id": 22,
    "return_date": "2020-04-15T12:00:00Z",
    "extend_times": 0,
    "real_return_date": null
  }
}
```

Output:

```
Record 30
   Book ID:     22
   Return Date: 2020-04-15T12:00:00Z
   Extended:    0/3
```

Possible error messages are shown below.

```
auth: unauthorized
library: book not borrowed
```

3.19 Extend the deadline to return a book

3.19.1 Request

Method: `PATCH /user/books/:id`
CLI command: `extend ddl`

In `realms`:

```
> extend ddl
Book ID: 22
```

User privilege is required.

3.19.2 Response

Status: `200 OK`
Content-Type: `application/json`

```json
{
  "data": {
    "id": 30,
    "user_id": 5,
    "book_id": 22,
    "return_date": "2020-04-22T12:00:00Z",
    "extend_times": 1,
    "real_return_date": null
  }
}
```

By default, the return date is extended by `7` days per request, and a user can extend the deadline for at most `3` times. You may change them in the config file `./configs/library_config.json`.

```
Record 30
   Book ID:     22
   Return Date: 2020-04-22T12:00:00Z
   Extended:    1/3
```

If a user tries to extend for more than `3` times, an error will be returned.

```
library: extended too many times
```

Other possible error messages are shown below.

```
auth: unauthorized
library: book not borrowed
```

3.20 Show all books that you've borrowed

3.20.1 Request

Method: `GET /user/books`
CLI command: `show list`

In `realms`:

```
> show list
```

User privilege is required.

3.20.2 Response

Status: `200 OK`
Content-Type: `application/json`

```json
{
  "data": [
    {
      "id": 2,
      "user_id": 5,
      "book_id": 12,
      "return_date": "2019-02-10T18:00:00Z",
      "extend_times": 3,
      "real_return_date": null
    },
    {
      "id": 30,
      "user_id": 5,
      "book_id": 22,
      "return_date": "2020-04-22T12:00:00Z",
      "extend_times": 1,
      "real_return_date": null
    },
    {
      "id": 32,
      "user_id": 5,
      "book_id": 20,
      "return_date": "2020-05-20T12:00:00Z",
      "extend_times": 0,
      "real_return_date": null
    }
  ]
}
```

Output:

```
ID      Book ID   Return Date           Extended
------------------------------------------------
2       12        2019-02-10T18:00:00Z  3/3
30      22        2020-04-22T12:00:00Z  1/3
32      20        2020-05-20T12:00:00Z  0/3
```

If there's no record found, `realms` will print the following message.

```
No records found
```

Possible error messages are shown below.

```
auth: unauthorized
```

3.21 Show all overdue books that you've borrowed

3.21.1 Request

Method: `GET /user/overdue`
CLI command: `show overdue`

In `realms`:

```
> show overdue
```

User privilege is required.

3.21.2 Response

Status: `200 OK`
Content-Type: `application/json`

```json
{
  "data": [
    {
      "id": 2,
      "user_id": 5,
      "book_id": 12,
      "return_date": "2019-02-10T18:00:00Z",
      "extend_times": 3,
      "real_return_date": null
    },
    {
      "id": 30,
      "user_id": 5,
      "book_id": 22,
      "return_date": "2020-04-22T12:00:00Z",
      "extend_times": 1,
      "real_return_date": null
    }
  ]
}
```

The records will be ordered by return date in ascending order.

```
ID      Book ID   Return Date           Extended
------------------------------------------------
2       12        2019-02-10T18:00:00Z  3/3
30      22        2020-04-22T12:00:00Z  1/3
```

Possible error messages are shown below.

```
auth: unauthorized
```

3.22 Show all your records

3.22.1 Request

Method: `GET /user/history`
CLI command: `show history`

In `realms`:

```
> show history
```

User privilege is required.

3.22.2 Response

Status: `200 OK`
Content-Type: `application/json`

```
{
  "data": [
    {
      "id": 32,
      "user_id": 5,
      "book_id": 20,
      "return_date": "2020-05-20T12:00:00Z",
      "extend_times": 0,
      "real_return_date": null
    },
    {
      "id": 30,
      "user_id": 5,
      "book_id": 22,
      "return_date": "2020-04-22T12:00:00Z",
      "extend_times": 1,
      "real_return_date": null
    },
    {
      "id": 15,
      "user_id": 5,
      "book_id": 20,
      "return_date": "2020-01-15T12:00:00Z",
      "extend_times": 0,
      "real_return_date": null
    },
    {
      "id": 2,
      "user_id": 5,
      "book_id": 12,
      "return_date": "2019-02-10T18:00:00Z",
      "extend_times": 3,
      "real_return_date": null
    }
  ]
}
```

The records will be ordered by record ID in descending order. Here returned date is equal to `real_return_date` in the response, which is the same as `delete_at` in the database. That's why we use a soft delete.

```
ID      Book ID   Return Date           Extended  Returned Date
----------------------------------------------------------------------
32      20        2020-05-20T12:00:00Z  0/3       N/A
30      22        2020-04-22T12:00:00Z  1/3       N/A
15      20        2020-01-15T12:00:00Z  0/3       2020-05-05T17:18:07Z
2       12        2019-02-10T18:00:00Z  3/3       N/A
```

Possible error messages are shown below.

```
auth: unauthorized
```

TODO

- [ ] Add unit tests

Contributors

- **Hakula Chen**<i@hakula.xyz> - Fudan University

License