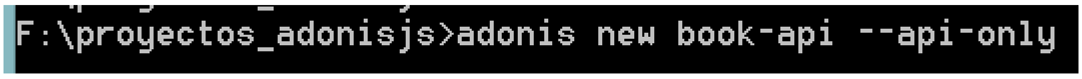
PROYECTO ADONISJS

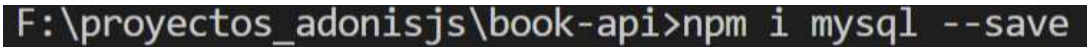
➔Creando proyecto



```
F:\proyectos_adonisjs>adonis new book-api --api-only
```

Configuración y base de datos

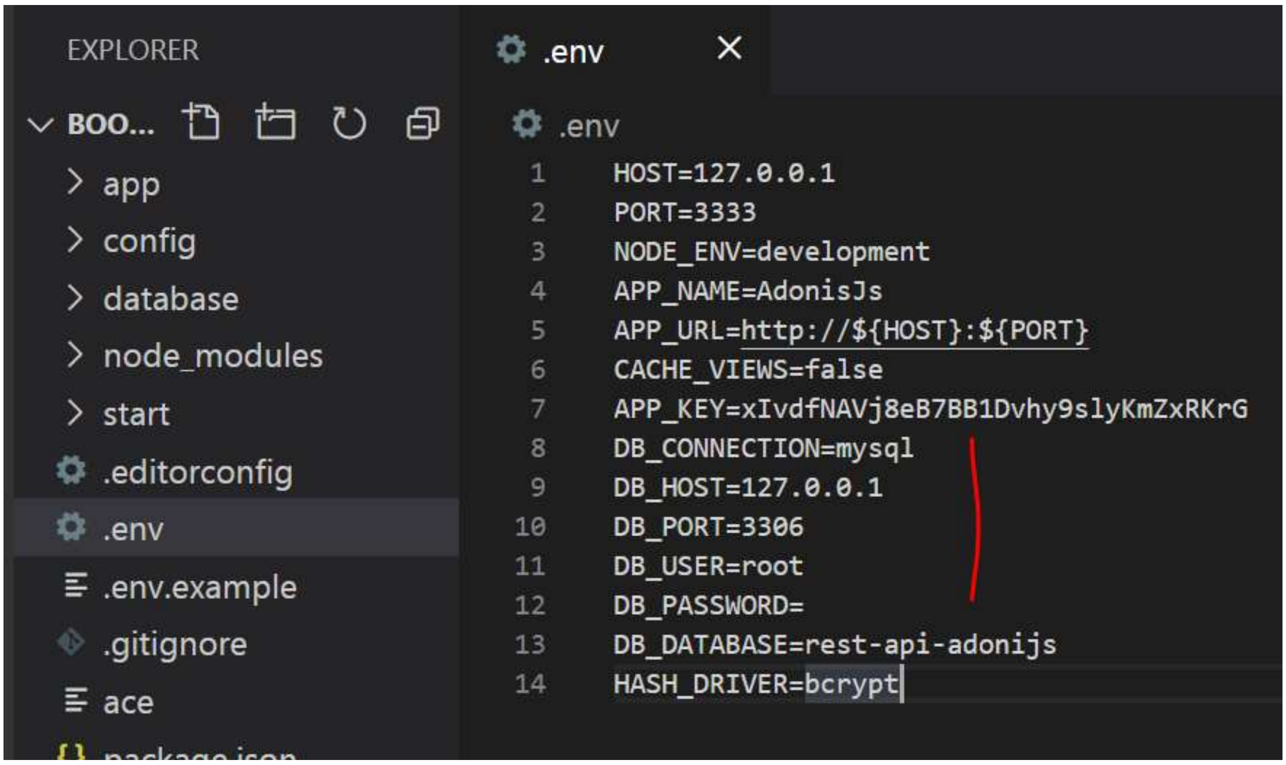
➔Instalando mysql al proyecto



```
F:\proyectos_adonisjs\book-api>npm i mysql --save
```

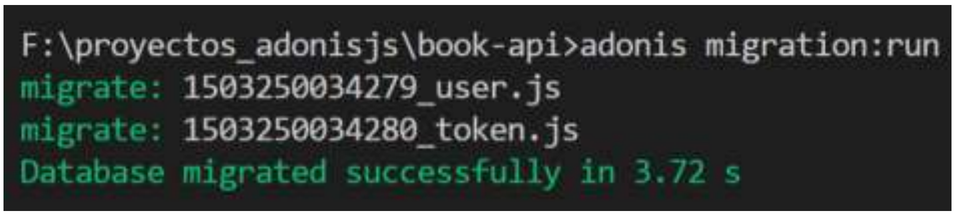
➔ Configurar entorno mysql y migraciones

En el archivo "env"



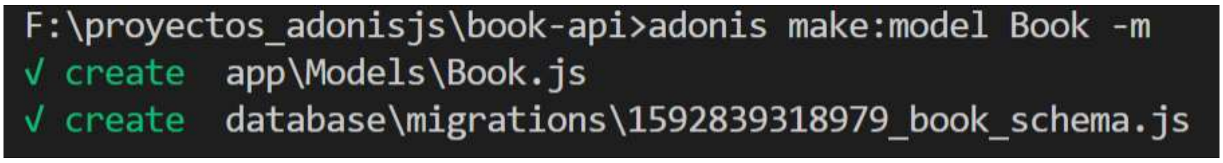
```
HOST=127.0.0.1
PORT=3333
NODE_ENV=development
APP_NAME=AdonisJs
APP_URL=http://${HOST}:${PORT}
CACHE_VIEWS=false
APP_KEY=xIvdfNAVj8eB7BB1Dvhy9slyKmZxRKrG
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_USER=root
DB_PASSWORD=
DB_DATABASE=rest-api-adonijs
HASH_DRIVER=bcrypt
```

Migrar, user y token



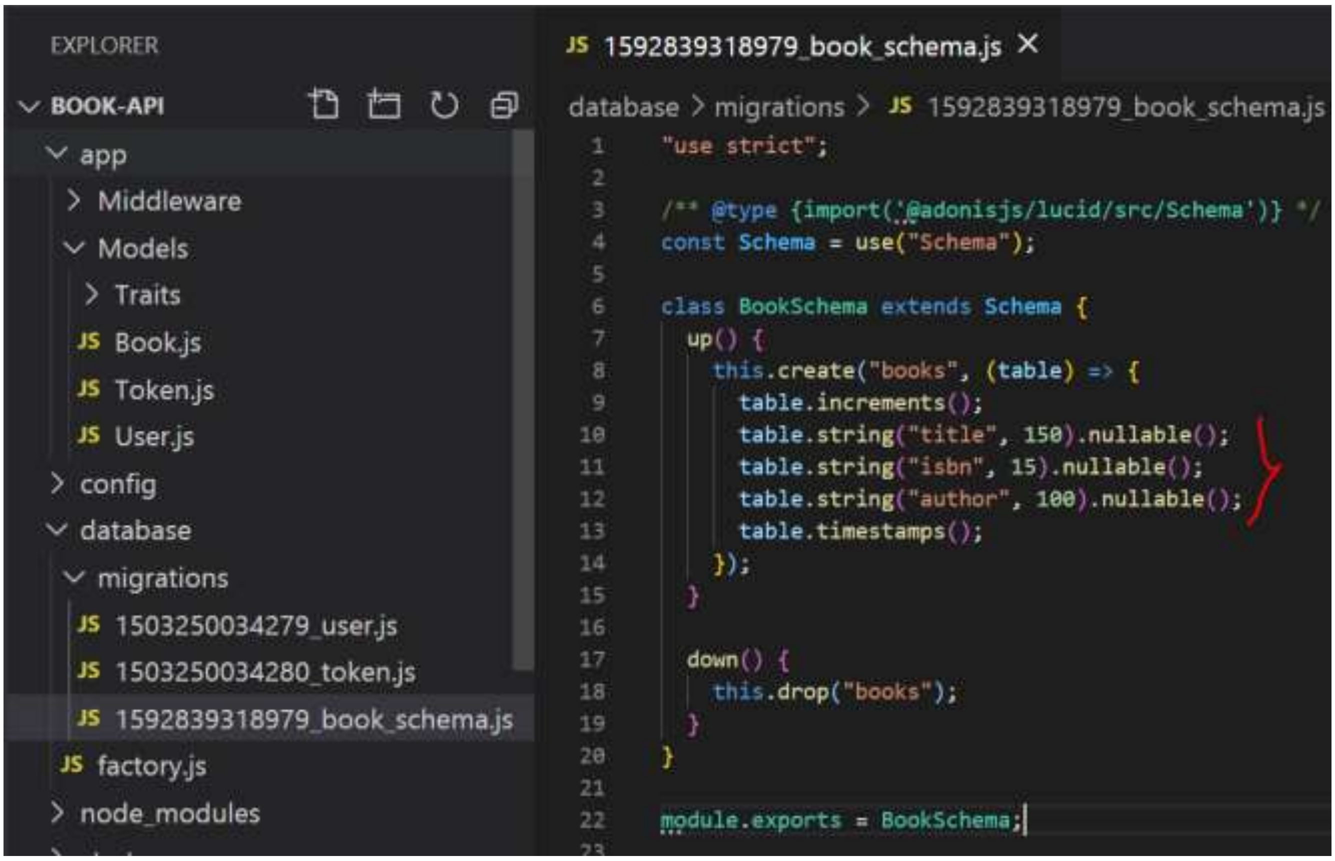
```
F:\proyectos_adonisjs\book-api>adonis migration:run
migrate: 1503250034279_user.js
migrate: 1503250034280_token.js
Database migrated successfully in 3.72 s
```

➔ Desarrollar el modelo y la migracion Book



```
F:\proyectos_adonisjs\book-api>adonis make:model Book -m
√ create  app\Models\Book.js
√ create  database\migrations\1592839318979_book_schema.js
```

Añadiendo campos en la esquema “Book”:



```
"use strict";

/** @type {import('@adonisjs/lucid/src/Schema')} */
const Schema = use("Schema");

class BookSchema extends Schema {
  up() {
    this.create("books", (table) => {
      table.increments();
      table.string("title", 150).nullable();
      table.string("isbn", 15).nullable();
      table.string("author", 100).nullable();
      table.timestamps();
    });
  }

  down() {
    this.drop("books");
  }
}

module.exports = BookSchema;
```

Configurar autenticación

Se puede usar el tipo de autenticación configurando el archivo auth (config/ayth.js)

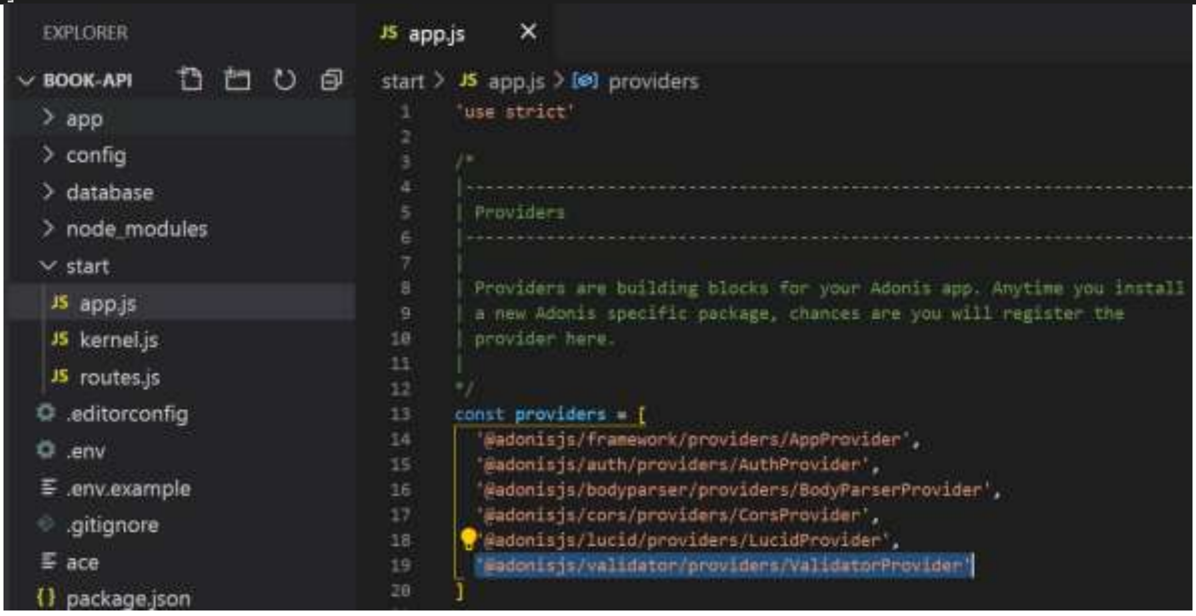
Configurar validacion

Instalando los validadores:

```
F:\proyectos_adonisjs\book-api>adonis install @adonisjs/validator
```

Registrando el provider:

```
const providers = [
  '@adonisjs/framework/providers/AppProvider',
  '@adonisjs/auth/providers/AuthProvider',
  '@adonisjs/bodyparser/providers/BodyParserProvider',
  '@adonisjs/cors/providers/CorsProvider',
  '@adonisjs/lucid/providers/LucidProvider',
  '@adonisjs/validator/providers/ValidatorProvider'
]
```



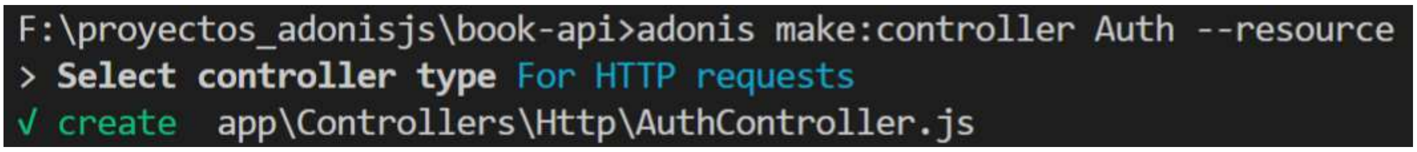
```
'use strict'

/*
|--------------------------------------------------------------------------
| Providers
|--------------------------------------------------------------------------
|
| Providers are building blocks for your Adonis app. Anytime you install
| a new Adonis specific package, chances are you will register the
| provider here.
|
*/
const providers = [
  '@adonisjs/framework/providers/AppProvider',
  '@adonisjs/auth/providers/AuthProvider',
  '@adonisjs/bodyparser/providers/BodyParserProvider',
  '@adonisjs/cors/providers/CorsProvider',
  '@adonisjs/lucid/providers/LucidProvider',
  '@adonisjs/validator/providers/ValidatorProvider'
]
```

Autenticacion y registro de usuarios

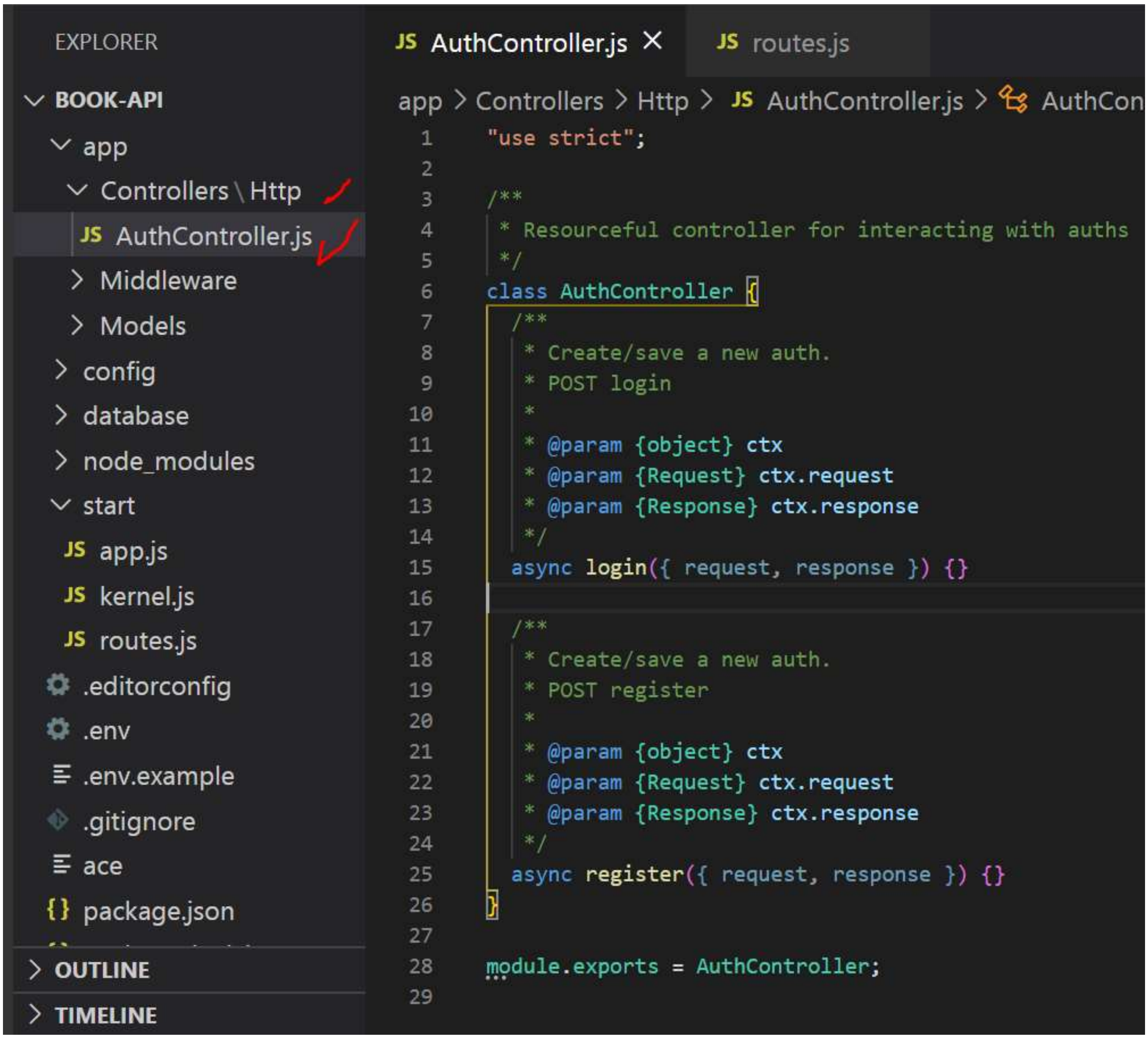
➔Rutas y controlador para autenticacion

Creando controlador:

adonis make:controller Auth --resource → Crea todos los métodos en el controlador



```
F:\proyectos_adonisjs\book-api>adonis make:controller Auth --resource
> Select controller type For HTTP requests
√ create  app\Controllers\Http\AuthController.js
```

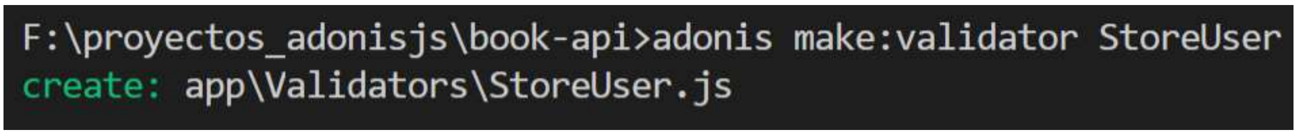


```
"use strict";

/**
 * Resourceful controller for interacting with auths
 */
class AuthController {
  /**
   * Create/save a new auth.
   * POST login
   *
   * @param {object} ctx
   * @param {Request} ctx.request
   * @param {Response} ctx.response
   */
  async login({ request, response }) {}

  /**
   * Create/save a new auth.
   * POST register
   *
   * @param {object} ctx
   * @param {Request} ctx.request
   * @param {Response} ctx.response
   */
  async register({ request, response }) {}
}

module.exports = AuthController;
```

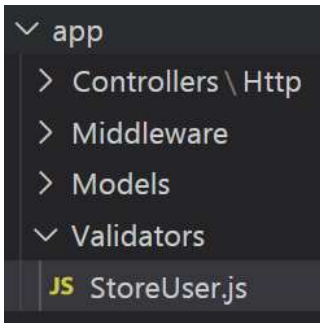
Creando las rutas:

```
"use strict";

/** @type {typeof import('@adonisjs/framework/src/Route/Manager')} */
const Route = use("Route");

Route.get("/", () => {
  return { greeting: "Hello world in JSON" };
});

/* Creando grupo de rutas */
Route.group(() => {
  Route.post("login", "AuthController.login");
  Route.post("register", "AuthController.register");
}).prefix("api/v1");
```

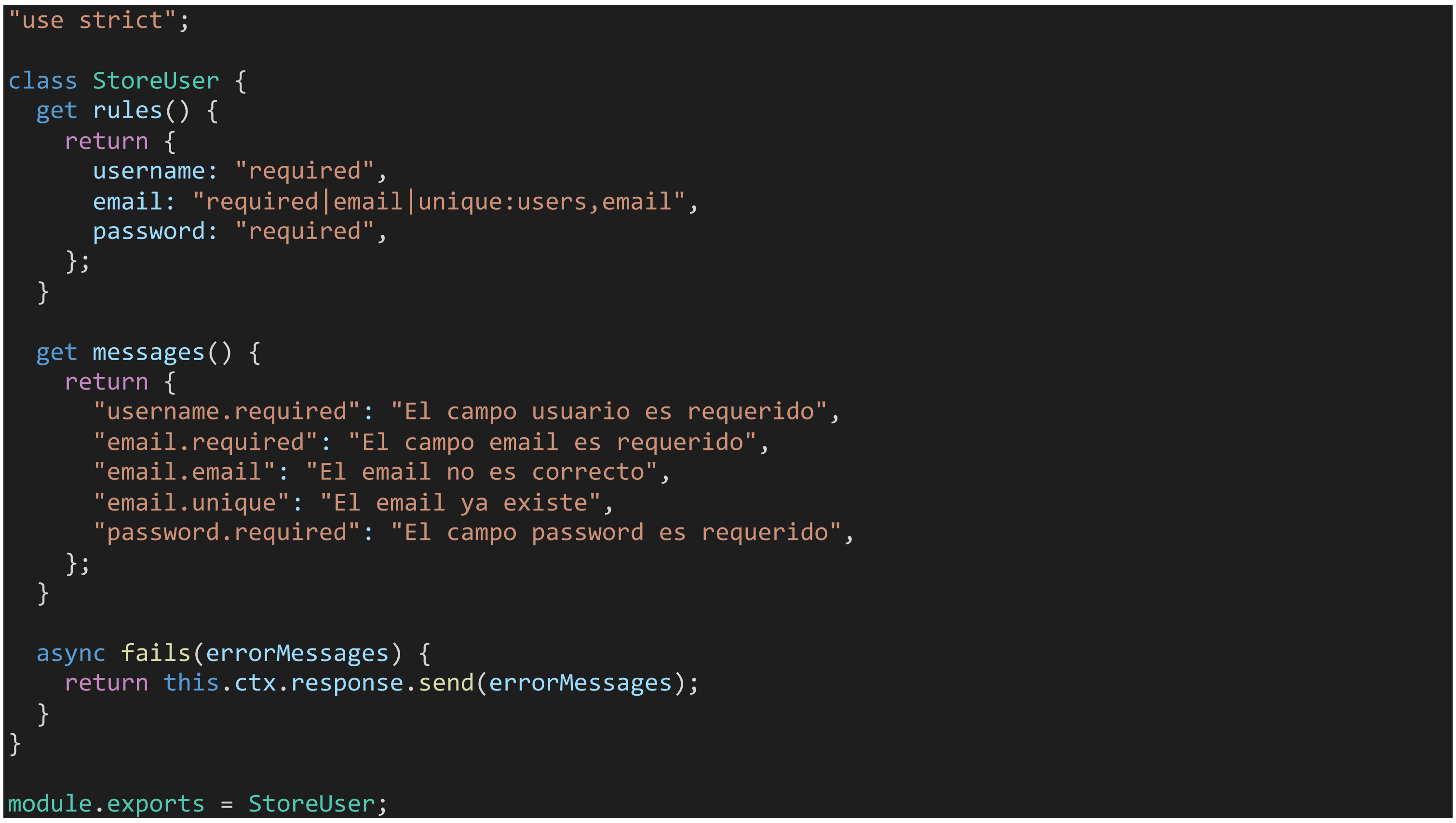
➔Validacion para registrar usuarios a través de rutas



```
F:\proyectos_adonisjs\book-api>adonis make:validator StoreUser
create: app\Validators\StoreUser.js
```

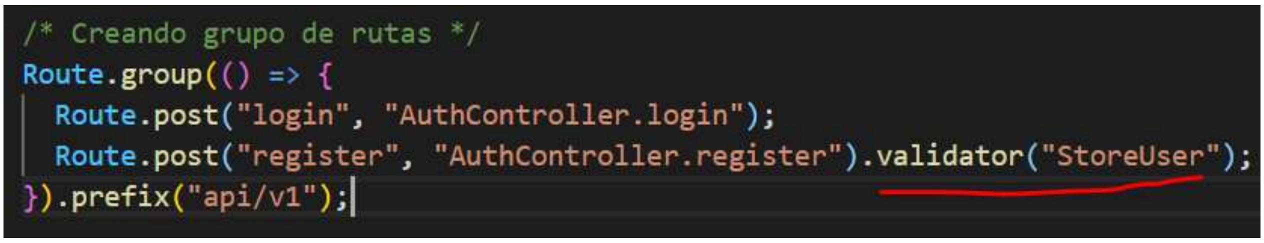


```
app
  Controllers\Http
  Middleware
  Models
  Validators
    StoreUser.js
```



```
"use strict";

class StoreUser {
  get rules() {
    return {
      username: "required",
      email: "required|email|unique:users,email",
      password: "required",
    };
  }

  get messages() {
    return {
      "username.required": "El campo usuario es requerido",
      "email.required": "El campo email es requerido",
      "email.email": "El email no es correcto",
      "email.unique": "El email ya existe",
      "password.required": "El campo password es requerido",
    };
  }

  async fails(errorMessages) {
    return this.ctx.response.send(errorMessages);
  }
}

module.exports = StoreUser;
```

Usar el validador en las ruta de registro:



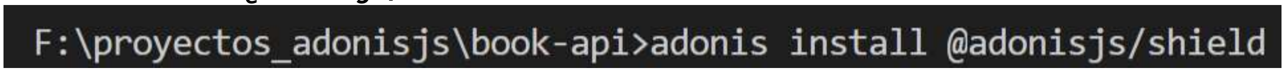
```
/* Creando grupo de rutas */
Route.group(() => {
  Route.post("login", "AuthController.login");
  Route.post("register", "AuthController.register").validator("StoreUser");
}).prefix("api/v1");
```

➔Filtro para la proteccion CSRF

CSRF: el Cross Site Request Forgery (CSRF o XSRF) es un tipo de ataque que se suele usar para estafas por Internet. Los delincuentes se apoderan de una sesión autorizada por el usuario (session riding) para realizar actos dañinos. El proceso se lleva a cabo mediante solicitudes HTTP.

Instalando “shield”:

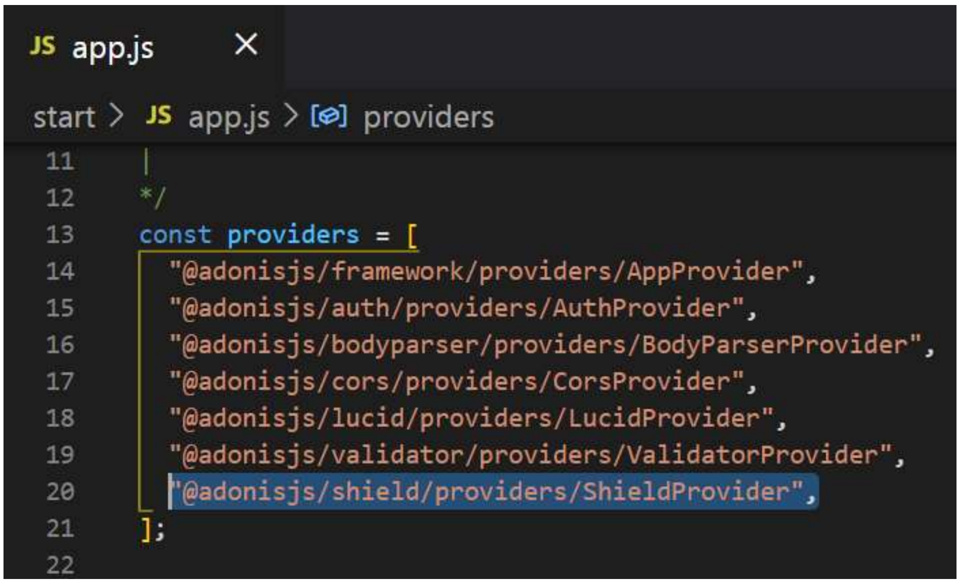
```
adonis install @adonisjs/shield
```



```
F:\proyectos_adonisjs\book-api>adonis install @adonisjs/shield
```

Agregar el provider en “start/app.js”:

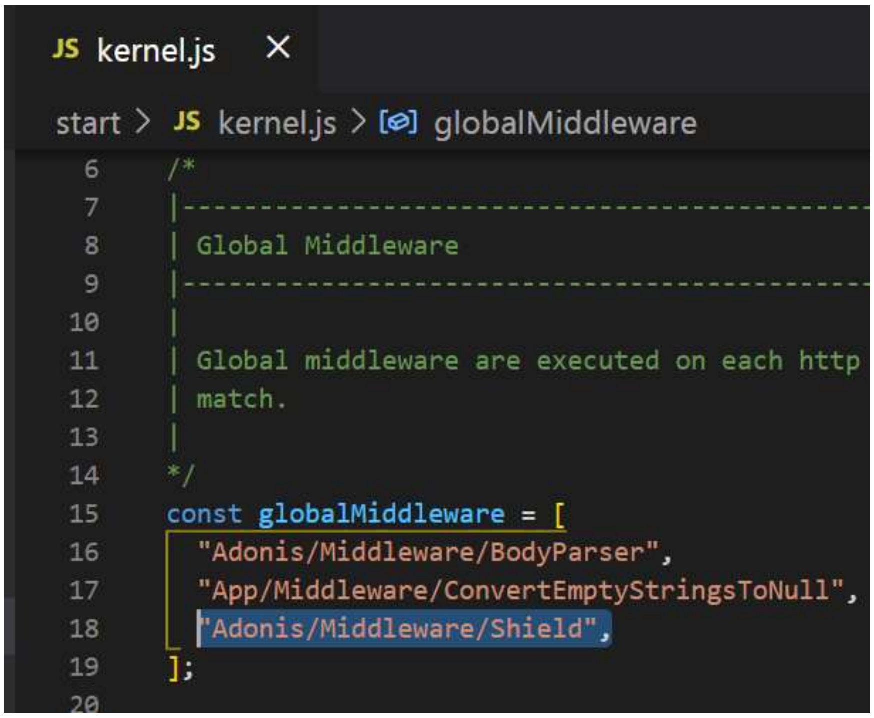
```
'@adonisjs/shield/providers/ShieldProvider'
```



```
/*
const providers = [
  "@adonisjs/framework/providers/AppProvider",
  "@adonisjs/auth/providers/AuthProvider",
  "@adonisjs/bodyparser/providers/BodyParserProvider",
  "@adonisjs/cors/providers/CorsProvider",
  "@adonisjs/lucid/providers/LucidProvider",
  "@adonisjs/validator/providers/ValidatorProvider",
  "@adonisjs/shield/providers/ShieldProvider",
];
```

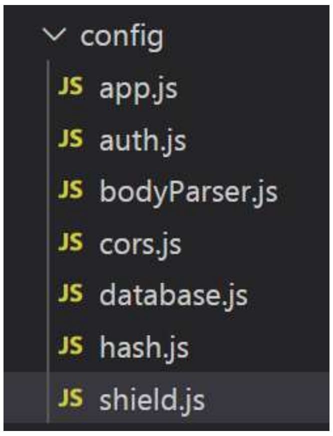
Registrar middleware en “start/kernel.js”

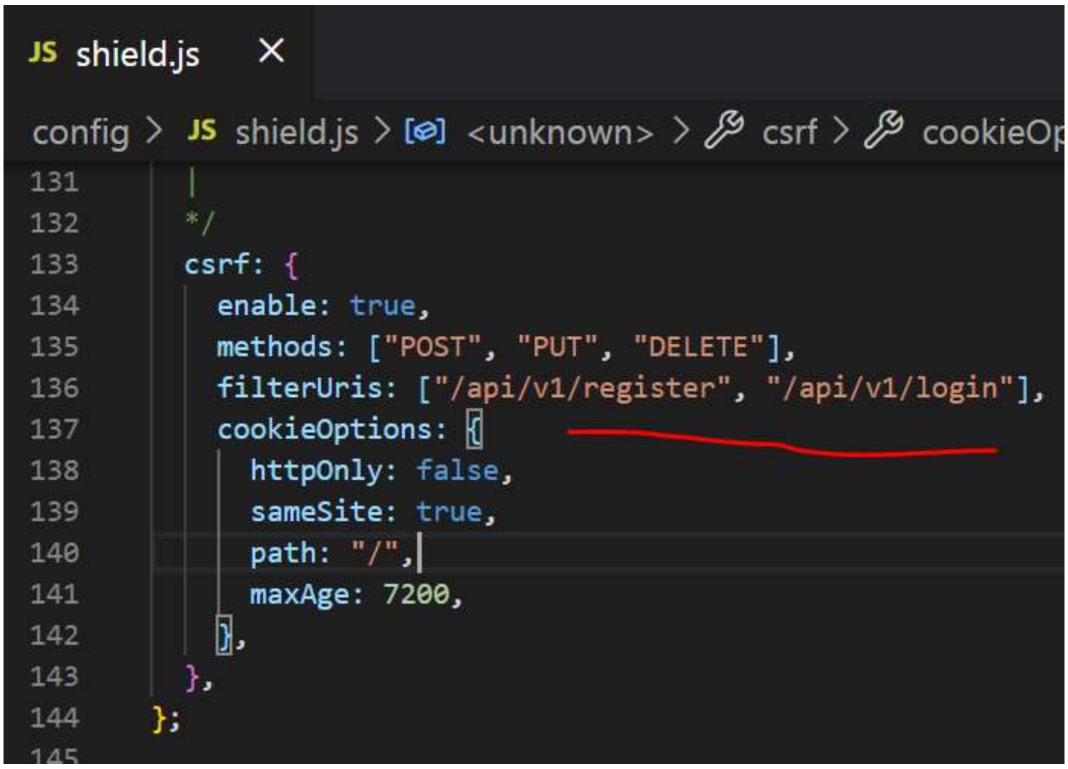
```
const globalMiddleware = [

  'Adonis/Middleware/Shield'

]
```



```
/*
|--------------------------------------------------
| Global Middleware
|--------------------------------------------------
|
| Global middleware are executed on each http
| match.
|
*/
const globalMiddleware = [
  "Adonis/Middleware/BodyParser",
  "App/Middleware/ConvertEmptyStringsToNull",
  "Adonis/Middleware/Shield",
];
```

Configurar csrf en “`config/shield.js`”





```
    |
    */
    csrf: {
      enable: true,
      methods: ["POST", "PUT", "DELETE"],
      filterUris: ["/api/v1/register", "/api/v1/login"],
      cookieOptions: {
        httpOnly: false,
        sameSite: true,
        path: "/",
        maxAge: 7200,
      },
    },
};
```

➔Registrar usuarios y login

```
"use strict";

const User = use("App/Models/User");

/**
 * Resourceful controller for interacting with auths
 */
class AuthController {
  /**
   * Create/save a new auth.
   * POST login
   *
   * @param {object} ctx
   * @param {Request} ctx.request
   * @param {Response} ctx.response
   * @param {Auth} ctx.auth
   */
  async login({ request, response, auth }) {
    const { email, password } = request.all();
    const user = await auth.attempt(email, password);
    return response.json(user);
  }
  /**
   * Create/save a new auth.
   * POST register
   *
   * @param {object} ctx
   * @param {Request} ctx.request
   * @param {Response} ctx.response
   */
  async register({ request, response }) {
    const data = request.only(["username", "email", "password"]);
    const user = await User.create({
      username: data.username,
      email: data.email,
      password: data.password,
    });
    // Generando token luego de crear nuevo usuario
    /* await auth.generate(user); */
    return response.status(200).json(user);
  }
}

module.exports = AuthController;
```

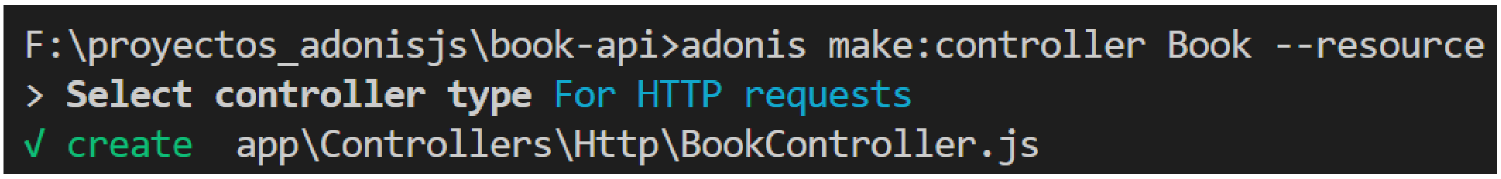
Rutas:

```
"use strict";

/** @type {typeof import('@adonisjs/framework/src/Route/Manager')} */
const Route = use("Route");

Route.get("/", () => {
  return { greeting: "Hello world in JSON" };
});

/* Creando grupo de rutas */
Route.group(() => {
  Route.post("login", "AuthController.login");
  Route.post("register", "AuthController.register").validator("StoreUser");
}).prefix("api/v1");
```

Crud books utilizando tokens

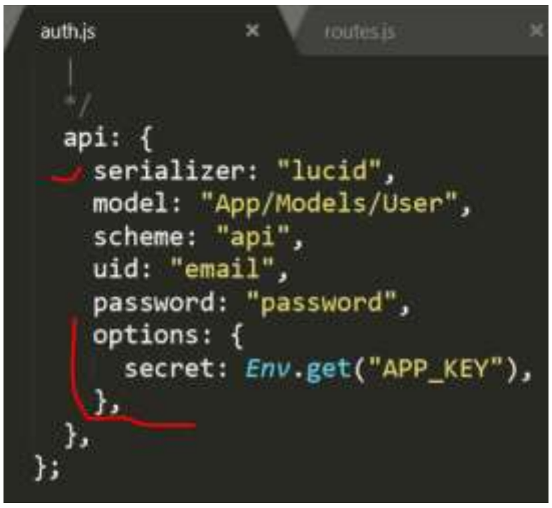
➔Rutas restful Books y método para obtener todos los recursos

Creando controlador:



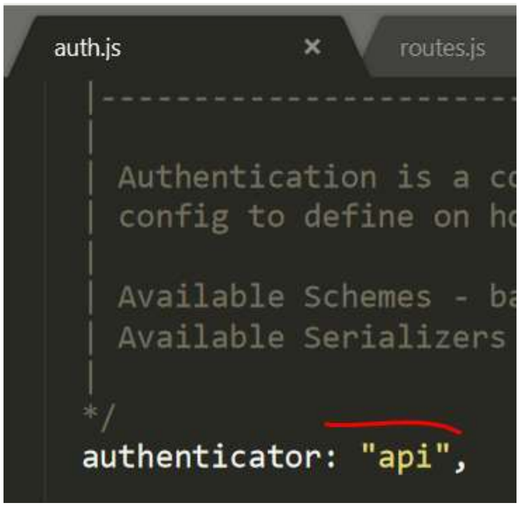
```
F:\proyectos_adonisjs\book-api>adonis make:controller Book --resource
> Select controller type For HTTP requests
√ create  app\Controllers\Http\BookController.js
```

En config/auth:



auth.js ×   routes.js ×

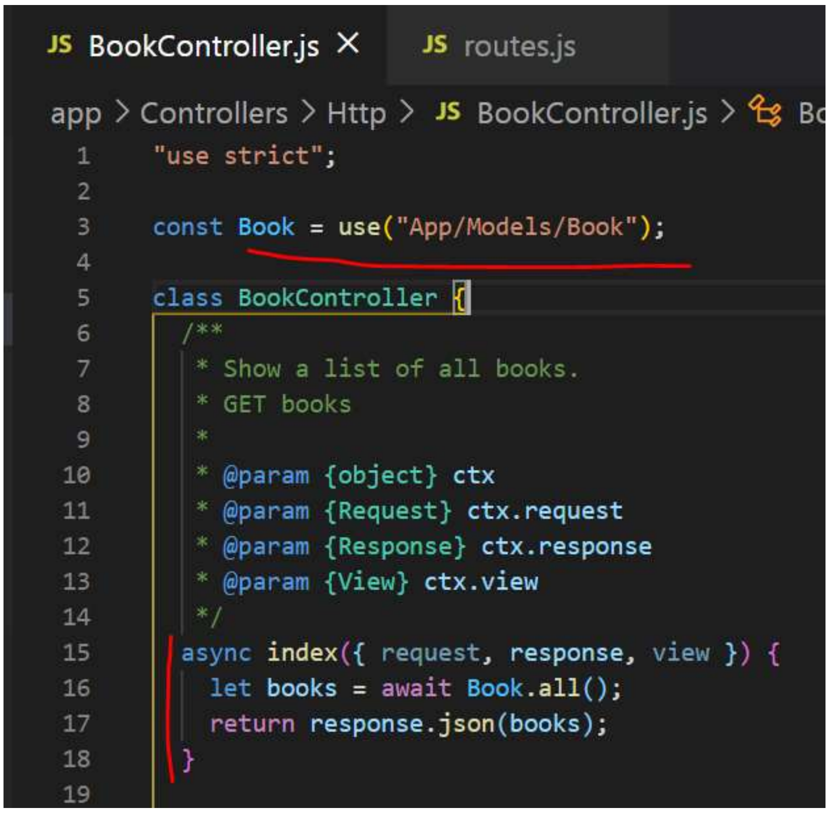
```
    */
  api: {
    serializer: "lucid",
    model: "App/Models/User",
    scheme: "api",
    uid: "email",
    password: "password",
    options: {
      secret: Env.get("APP_KEY"),
    },
  },
};
```



auth.js ×   routes.js

```
  |-----------------------
  |
  | Authentication is a co
  | config to define on ho
  |
  | Available Schemes - ba
  | Available Serializers
  |
  */
  authenticator: "api",
```

Método index del controlador:



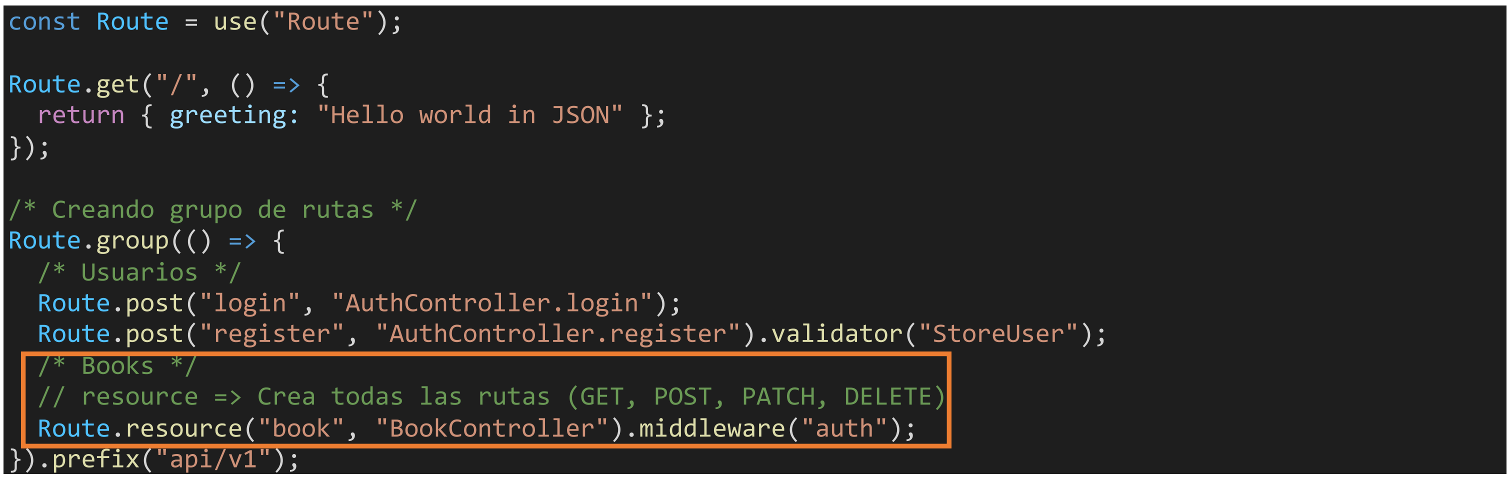
BookController.js ×   routes.js

app > Controllers > Http > BookController.js > Bo

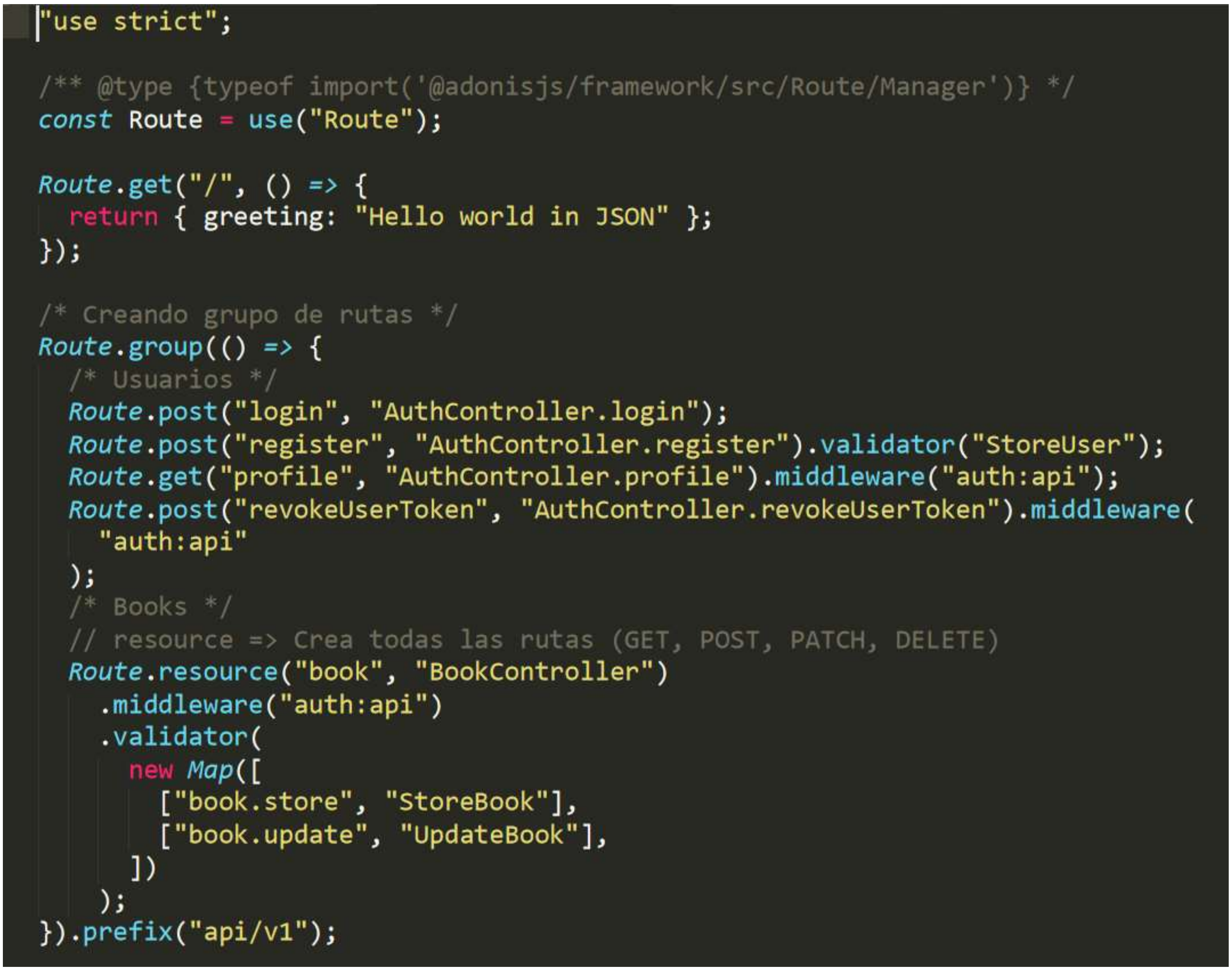
```
"use strict";

const Book = use("App/Models/Book");

class BookController {
  /**
   * Show a list of all books.
   * GET books
   *
   * @param {object} ctx
   * @param {Request} ctx.request
   * @param {Response} ctx.response
   * @param {View} ctx.view
   */
  async index({ request, response, view }) {
    let books = await Book.all();
    return response.json(books);
  }

```

Rutas:

```
"use strict";

/** @type {typeof import('@adonisjs/framework/src/Route/Manager')} */
```

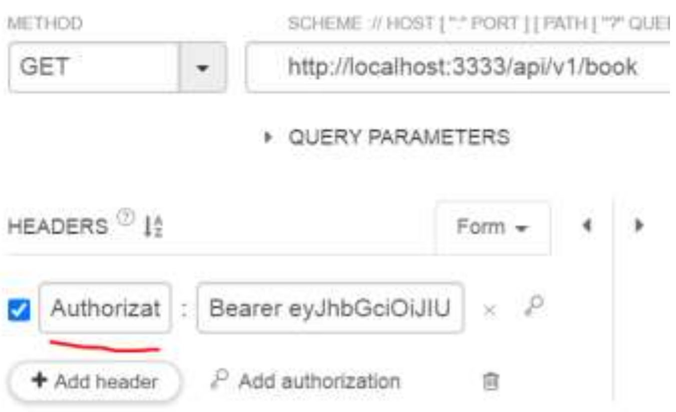


```
const Route = use("Route");

Route.get("/", () => {
  return { greeting: "Hello world in JSON" };
});

/* Creando grupo de rutas */
Route.group(() => {
  /* Usuarios */
  Route.post("login", "AuthController.login");
  Route.post("register", "AuthController.register").validator("StoreUser");
  /* Books */
  // resource => Crea todas las rutas (GET, POST, PATCH, DELETE)
  Route.resource("book", "BookController").middleware("auth");
}).prefix("api/v1");
```



```
"use strict";

/** @type {typeof import('@adonisjs/framework/src/Route/Manager')} */
const Route = use("Route");

Route.get("/", () => {
  return { greeting: "Hello world in JSON" };
});

/* Creando grupo de rutas */
Route.group(() => {
  /* Usuarios */
  Route.post("login", "AuthController.login");
  Route.post("register", "AuthController.register").validator("StoreUser");
  Route.get("profile", "AuthController.profile").middleware("auth:api");
  Route.post("revokeUserToken", "AuthController.revokeUserToken").middleware(
    "auth:api"
  );
  /* Books */
  // resource => Crea todas las rutas (GET, POST, PATCH, DELETE)
  Route.resource("book", "BookController")
    .middleware("auth:api")
    .validator(
      new Map([
        ["book.store", "StoreBook"],
        ["book.update", "UpdateBook"],
      ])
    );
}).prefix("api/v1");
```

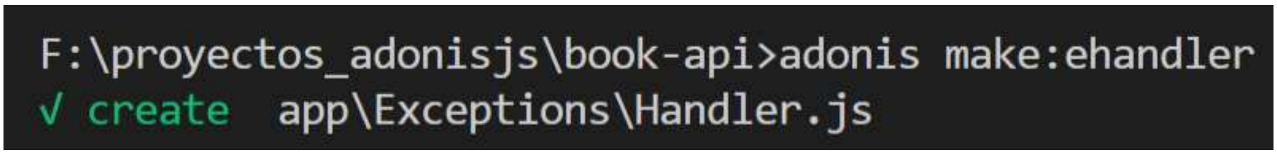
Prueba:



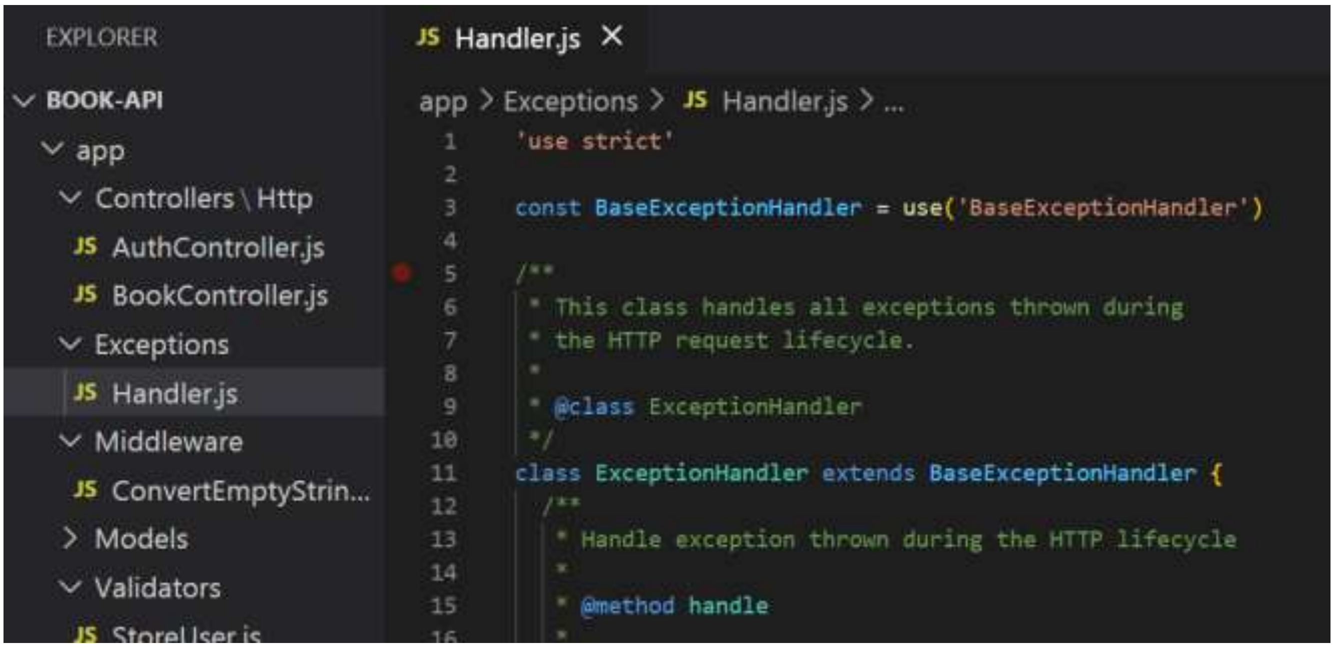
```
METHOD            SCHEME :// HOST [ ":" PORT ] [ PATH [ "?" QUE
GET               http://localhost:3333/api/v1/book

                  QUERY PARAMETERS

HEADERS                                   Form

☑ Authorizat : Bearer eyJhbGciOiJIU

+ Add header      Add authorization
```

➔Manejo de excepciones global

adonis make:ehandler



```
F:\proyectos_adonisjs\book-api>adonis make:ehandler
√ create  app\Exceptions\Handler.js
```

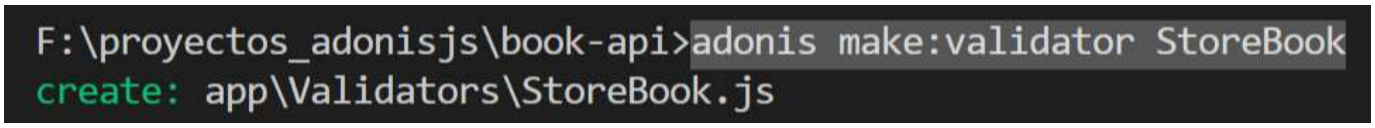


```
EXPLORER                      JS Handler.js ×
BOOK-API                      app > Exceptions > JS Handler.js > ...
  app                         'use strict'
    Controllers \ Http
      JS AuthController.js    const BaseExceptionHandler = use('BaseExceptionHandler')
      JS BookController.js
    Exceptions                /**
      JS Handler.js            * This class handles all exceptions thrown during
    Middleware                 * the HTTP request lifecycle.
      JS ConvertEmptyStrin...  *
    Models                     * @class ExceptionHandler
    Validators                 */
      JS StoreUser.js         class ExceptionHandler extends BaseExceptionHandler {
                                /**
                                 * Handle exception thrown during the HTTP lifecycle
                                 *
                                 * @method handle
                                 *
```

```
"use strict";

const BaseExceptionHandler = use("BaseExceptionHandler");

/**
 * This class handles all exceptions thrown during
 * the HTTP request lifecycle.
 *
 * @class ExceptionHandler
 */
class ExceptionHandler extends BaseExceptionHandler {
  /**
   * Handle exception thrown during the HTTP lifecycle
   *
   * @method handle
   *
   * @param  {Object} error
   * @param  {Object} options.request
   * @param  {Object} options.response
   *
   * @return {void}
   */
  async handle(error, { request, response }) {
    response.status(error.status).json({ msg: error.message });
  }
}

module.exports = ExceptionHandler;
```

➔Validadores para crear y actualizar Books

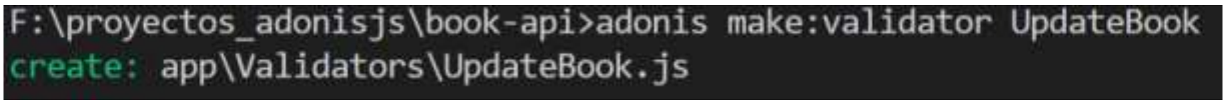
adonis make:validator StoreBook



```
F:\proyectos_adonisjs\book-api>adonis make:validator StoreBook
create: app\Validators\StoreBook.js
```

```
"use strict";

class StoreBook {
  get rules() {
    return {
      title: "required|unique:books,title",
      isbn: "required|unique:books,isbn",
      author: "required",
    };
  }

  get messages() {
    console.log("ENTRA A LOS MENSAJES DE ERROR");
    return {
      "title.required": "El título es requerido",
      "title.unique": "El título ya existe",
      "isbn.required": "El campo email es requerido",
      "isbn.unique": "El isbn ya existe",
      "author.required": "El autor es requerido",
    };
  }

  async fails(errorMessages) {
    return this.ctx.response.send(errorMessages);
  }
}

module.exports = StoreBook;
```

Además, se requiere un validador para la actualización del libro:



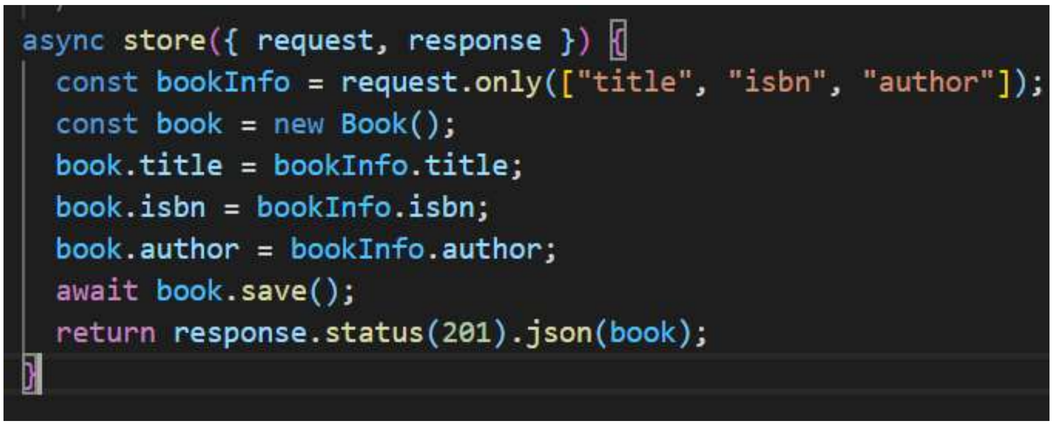
```
F:\proyectos_adonisjs\book-api>adonis make:validator UpdateBook
create: app\Validators\UpdateBook.js
```

```
"use strict";

class UpdateBook {
  get rules() {
    const bookId = this.ctx.params.id;
    return {
      title: `required|unique:books,title,id,${bookId}`,
      isbn: `required|unique:isbn,id,${bookId}"`,
      author: "required",
    };
  }

  get messages() {
    console.log("ENTRA A LOS MENSAJES DE ERROR");
    return {
      "title.required": "El título es requerido",
      "title.unique": "El título ya existe",
      "isbn.required": "El campo email es requerido",
      "isbn.unique": "El isbn ya existe",
      "author.required": "El autor es requerido",
    };
  }

  async fails(errorMessages) {
    return this.ctx.response.send(errorMessages);
  }
}

module.exports = UpdateBook;
```

En la ruta:

```
"use strict";

/** @type {typeof import('@adonisjs/framework/src/Route/Manager')} */
const Route = use("Route");

Route.get("/", () => {
  return { greeting: "Hello world in JSON" };
});

/* Creando grupo de rutas */
Route.group(() => {
  /* Usuarios */
  Route.post("login", "AuthController.login");
  Route.post("register", "AuthController.register").validator("StoreUser");
  /* Books */
  // resource => Crea todas las rutas (GET, POST, PATCH, DELETE)
  Route.resource("book", "BookController")
    .middleware("auth")
    .validator(
      new Map([
        ["book.store", "StoreBook"],
        ["book.update", "UpdateBook"],
      ])
    );
}).prefix("api/v1");
```

➔Dar de alta Books

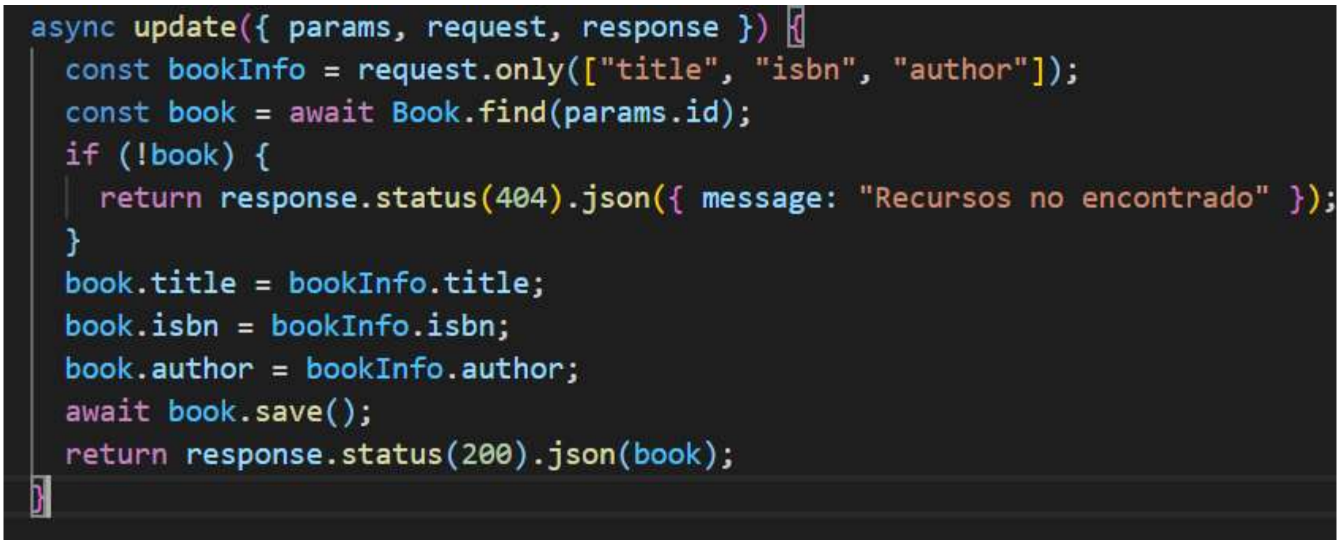


```
async store({ request, response }) {
  const bookInfo = request.only(["title", "isbn", "author"]);
  const book = new Book();
  book.title = bookInfo.title;
  book.isbn = bookInfo.isbn;
  book.author = bookInfo.author;
  await book.save();
  return response.status(201).json(book);
}
```

Restringir la devolución de datos del libro, en el modelo:

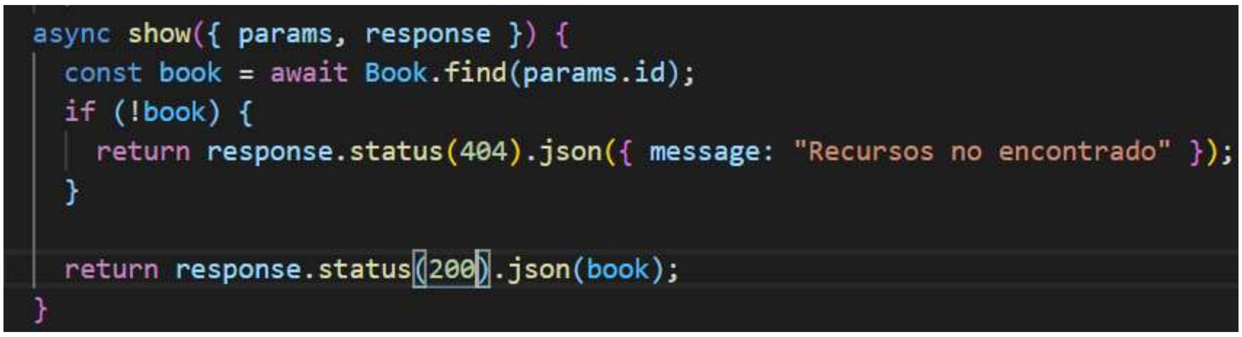
```
"use strict";

/** @type {typeof import('@adonisjs/lucid/src/Lucid/Model')} */
const Model = use("Model");

class Book extends Model {
  static get visible() {
    return ["title", "isbn", "author", "created_at"];
  }
}

module.exports = Book;
```

➔Actualizar Books



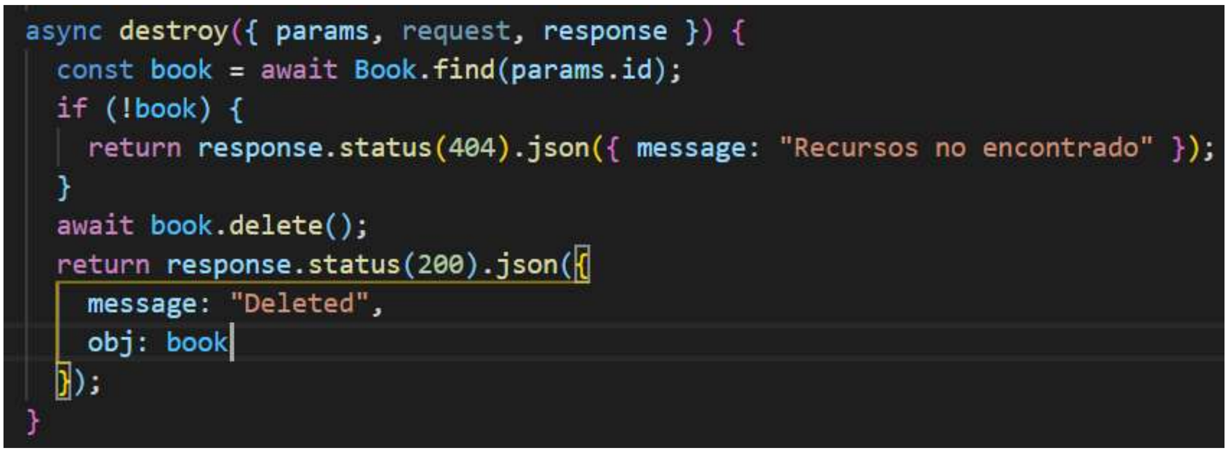
```
async update({ params, request, response }) {
  const bookInfo = request.only(["title", "isbn", "author"]);
  const book = await Book.find(params.id);
  if (!book) {
    return response.status(404).json({ message: "Recursos no encontrado" });
  }
  book.title = bookInfo.title;
  book.isbn = bookInfo.isbn;
  book.author = bookInfo.author;
  await book.save();
  return response.status(200).json(book);
}
```

➔Buscar Books por id



```
async show({ params, response }) {
  const book = await Book.find(params.id);
  if (!book) {
    return response.status(404).json({ message: "Recursos no encontrado" });
  }

  return response.status(200).json(book);
}
```

➔Eliminar Books



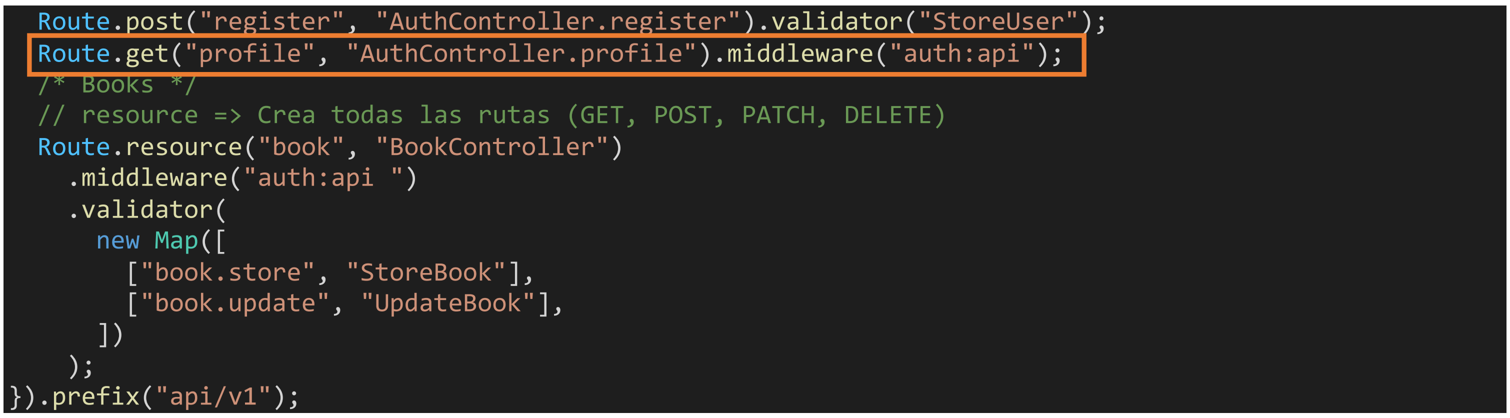
```
async destroy({ params, request, response }) {
  const book = await Book.find(params.id);
  if (!book) {
    return response.status(404).json({ message: "Recursos no encontrado" });
  }
  await book.delete();
  return response.status(200).json({
    message: "Deleted",
    obj: book
  });
}
```

<u>Obtener usuario por token y revocar tokens</u>

➔Obtener usuario por el token

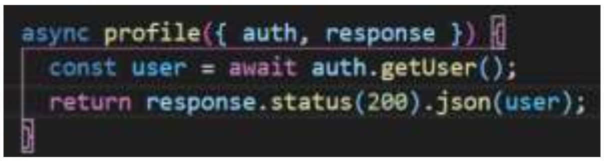
Creando la ruta que devolverá el perfil del usuario con sus datos:

```
Route.group(() => {
  /* Usuarios */
  Route.post("login", "AuthController.login");
```



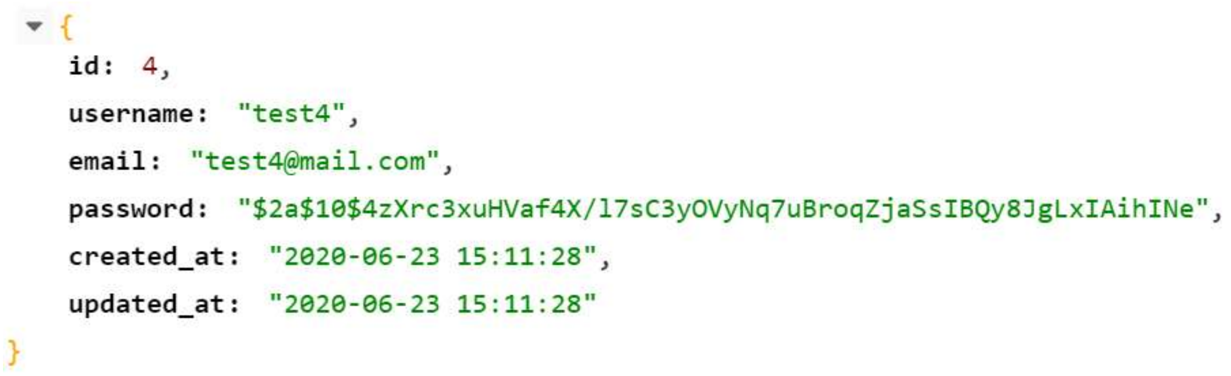
```
  Route.post("register", "AuthController.register").validator("StoreUser");
  Route.get("profile", "AuthController.profile").middleware("auth:api");
  /* Books */
  // resource => Crea todas las rutas (GET, POST, PATCH, DELETE)
  Route.resource("book", "BookController")
    .middleware("auth:api ")
    .validator(
      new Map([
        ["book.store", "StoreBook"],
        ["book.update", "UpdateBook"],
      ])
    );
}).prefix("api/v1");
```

Creando el método en AuthController:



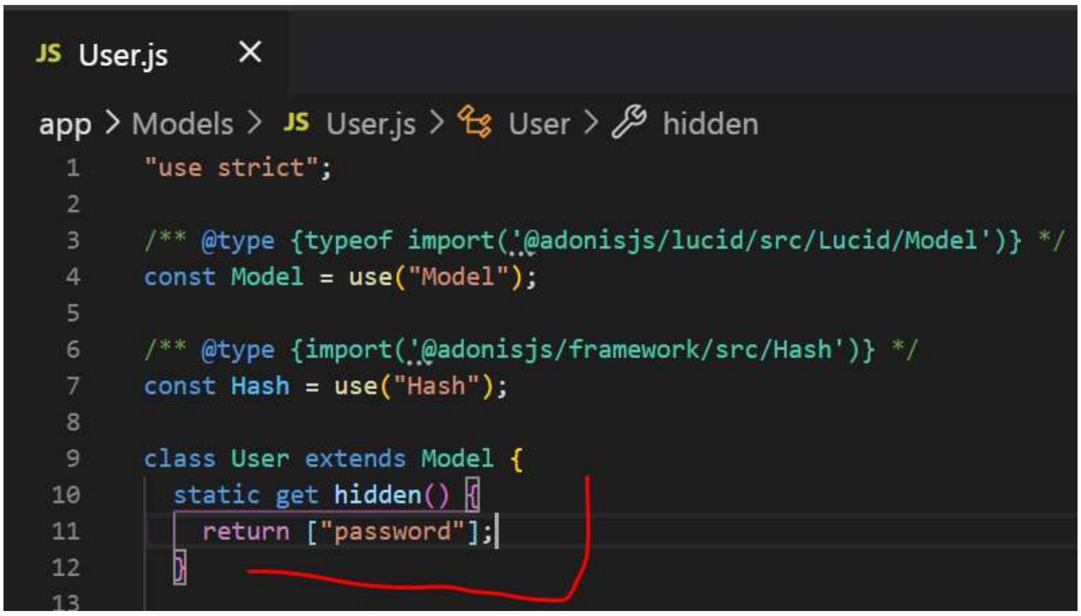
```
async profile({ auth, response }) {
  const user = await auth.getUser();
  return response.status(200).json(user);
}
```

Resultado:



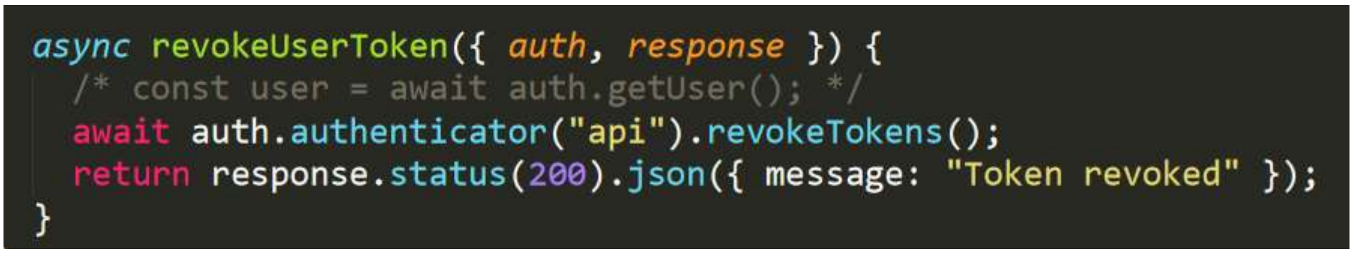
```
▾ {
    id: 4,
    username: "test4",
    email: "test4@mail.com",
    password: "$2a$10$4zXrc3xuHVaf4X/l7sC3yOVyNq7uBroqZjaSsIBQy8JgLxIAihINe",
    created_at: "2020-06-23 15:11:28",
    updated_at: "2020-06-23 15:11:28"
}
```

Modificando el modelo para que no devuelva el password del usuario:



```
"use strict";

/** @type {typeof import('@adonisjs/lucid/src/Lucid/Model')} */
const Model = use("Model");

/** @type {import('@adonisjs/framework/src/Hash')} */
const Hash = use("Hash");

class User extends Model {
  static get hidden() {
    return ["password"];
  }

```

➔Revocar tokens



```
async revokeUserToken({ auth, response }) {
  /* const user = await auth.getUser(); */
  await auth.authenticator("api").revokeTokens();
  return response.status(200).json({ message: "Token revoked" });
}
```