REST API CON ADONISJS

Ver: https://adonisjs.com/

Creando el proyecto:

adonis new nombre_proyecto

```
F:\proyectos_adonisjs>adonis new rest-api
```

Se crea un proyecto completo, pero sólo queremos hacer un rest-api, entonces eliminamos el proyectos y creamos otro así:

adonis new rest-api --api-only

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

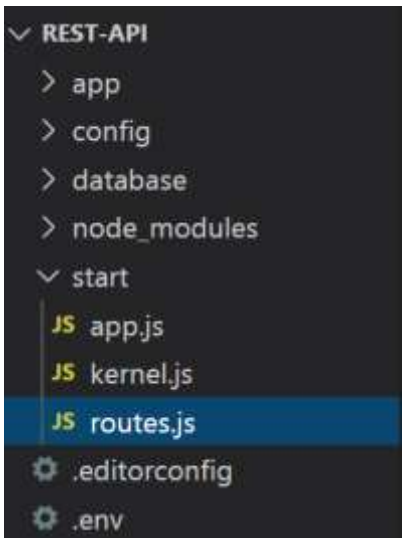
Se puede ejecutar el servidor:

adonis serve --dev

```
F:\proyectos_adonisjs\rest-api>adonis serve --dev

 SERVER STARTED
> Watching files for changes...

info: serving app on http://127.0.0.1:3333
```

Rutas y Usuarios

Rutas



routes.js

```
start > routes.js > ...
"use strict";

/*
|--------------------------------------------------------------------------
| Routes
|--------------------------------------------------------------------------
|
| Http routes are entry points to your web application. You can create
| routes for different URLs and bind Controller actions to them.
|
| A complete guide on routing is available here.
| http://adonisjs.com/docs/4.1/routing
|
*/

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

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

Route.post("users/register", ({ request }) => {
  return { message: "Regitraste un usuario" };
});
```

➔Creando un controlador:

adonis make:controller User

```
F:\proyectos_adonisjs\rest-api>adonis make:controller User
> Select controller type For HTTP requests
√ create  app\Controllers\Http\UserController.js
```

➔Asignar el controlador a la ruta correspondiente:

```
// Agrupando las rutas
Route.group(() => {
  Route.post("users/register", "UserController.store");
  Route.delete("users/register", "UserController.store");
  Route.put("users/register", "UserController.store");
}).prefix("api/v1");
```

Creando usuarios

Instalando sqlite:

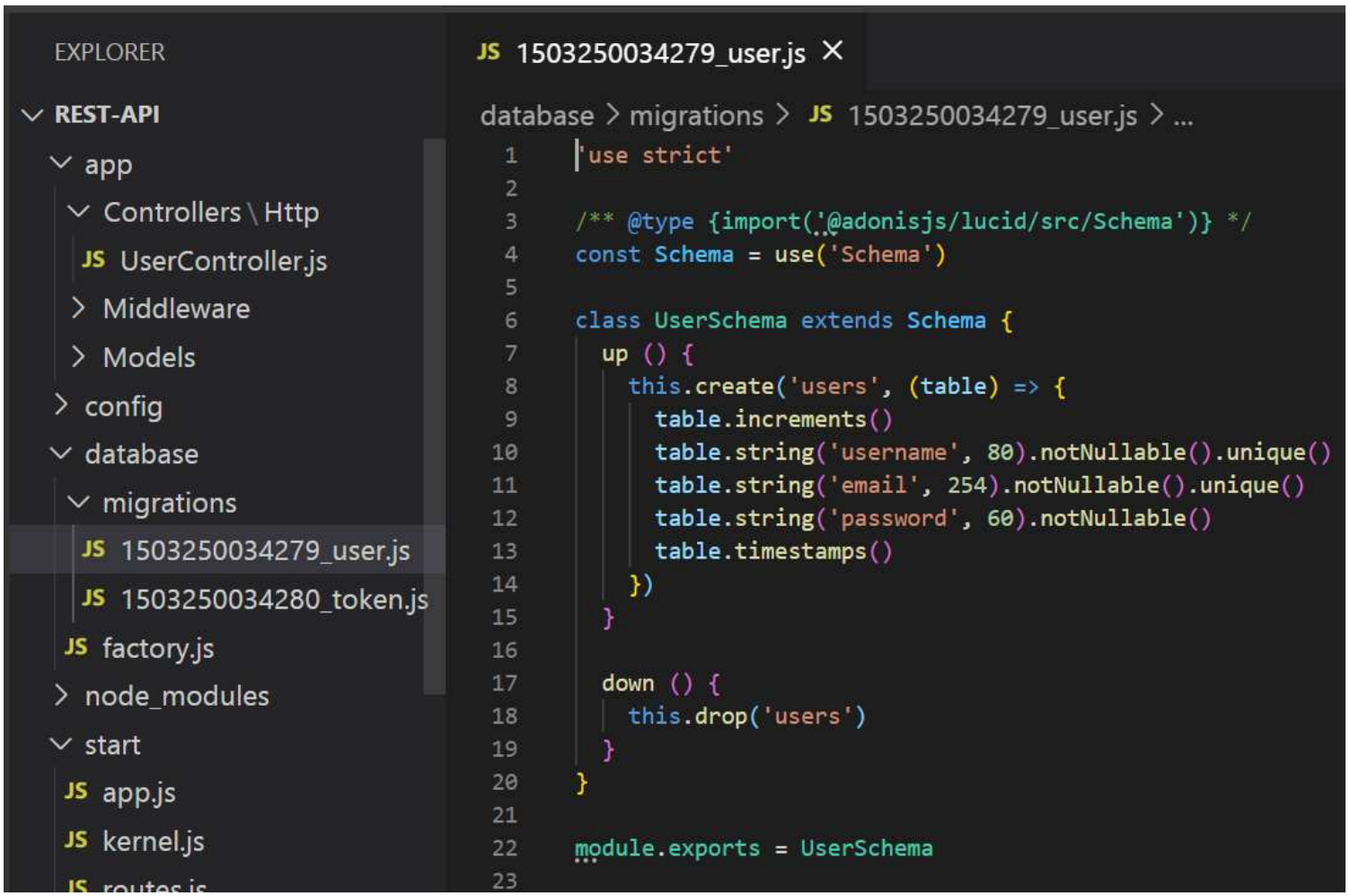
```
F:\proyectos_adonisjs\rest-api>npm install sqlite3 --save
```

Para crear las tablas, correr las migraciones:

adonis migration:run

```
F:\proyectos_adonisjs\rest-api>adonis migration:run
migrate: 1503250034279_user.js
migrate: 1503250034280_token.js
Database migrated successfully in 2.09 s
```

En las migraciones:



En el controlador:

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

class UserController {
  async store({ request }) {
    const { email, password } = request.all();
    const user = await User.create({ email, password, username: email });
    /* console.log(user); */
    return user;
  }
}

module.exports = UserController;
```

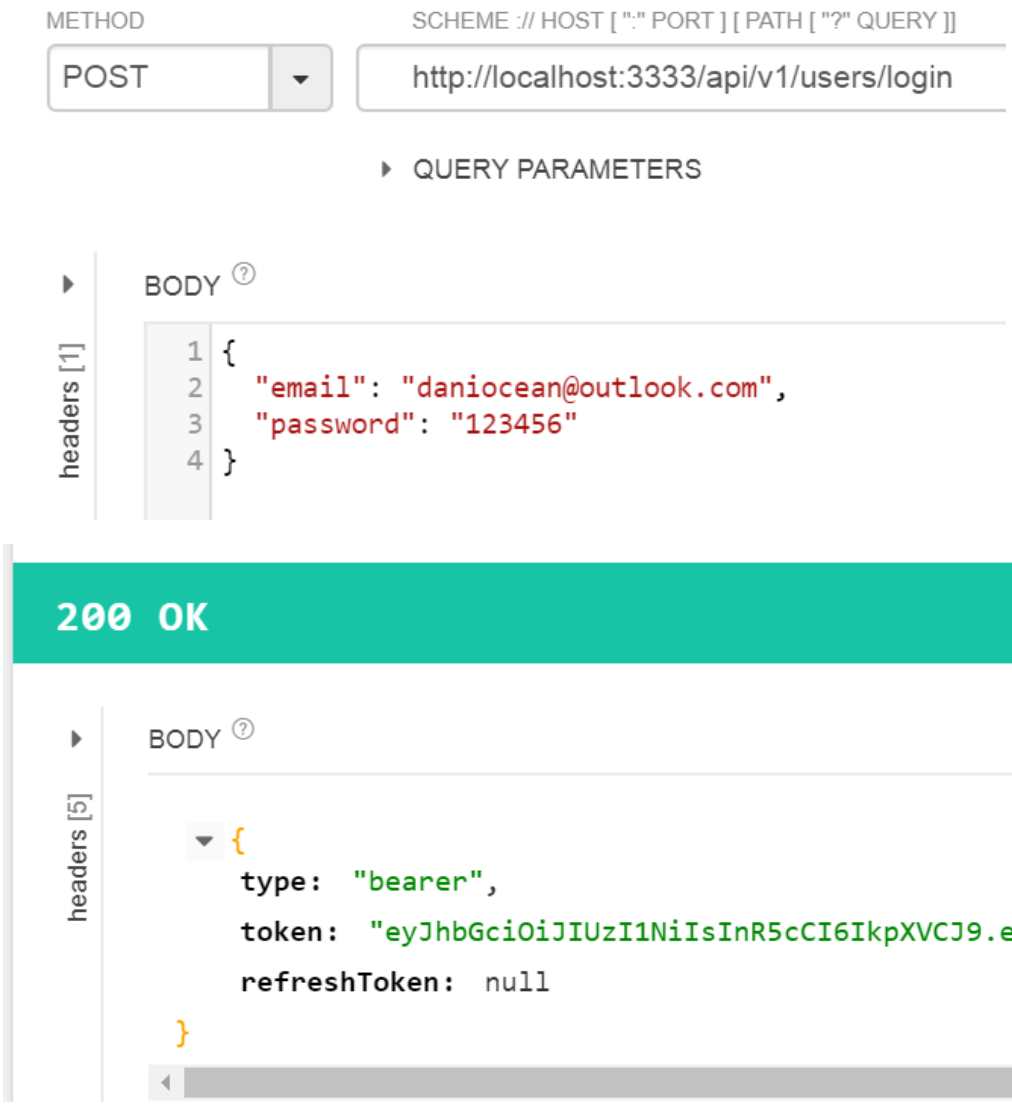
Login de Usuarios y JWT

Creando el método:

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

class UserController {
  async login({ request, auth }) {
    // Solicitar email y password
    const { email, password } = request.all();
    // Generando token si es correcto
    const token = await auth.attempt(email, password);
    return token;
  }

  async store({ request }) {
    const { email, password } = request.all();
    const user = await User.create({ email, password, username: email });
    /* console.log(user); */
    return user;
  }
}

module.exports = UserController;
```

Creando la ruta:

```
"use strict";

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

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

// Agrupando las rutas
Route.group(() => {
  Route.post("users/login", "UserController.login"); // login
  Route.post("users/register", "UserController.store"); // Registro
}).prefix("api/v1");
```



CREANDO OTRA INSTANCIA (Proyectos)

Creando el modelo Proyecto

```
F:\proyectos_adonisjs\rest-api>adonis make:model Proyecto -m
√ create  app\Models\Proyecto.js
√ create  database\migrations\1592600543208_proyecto_schema.js
```

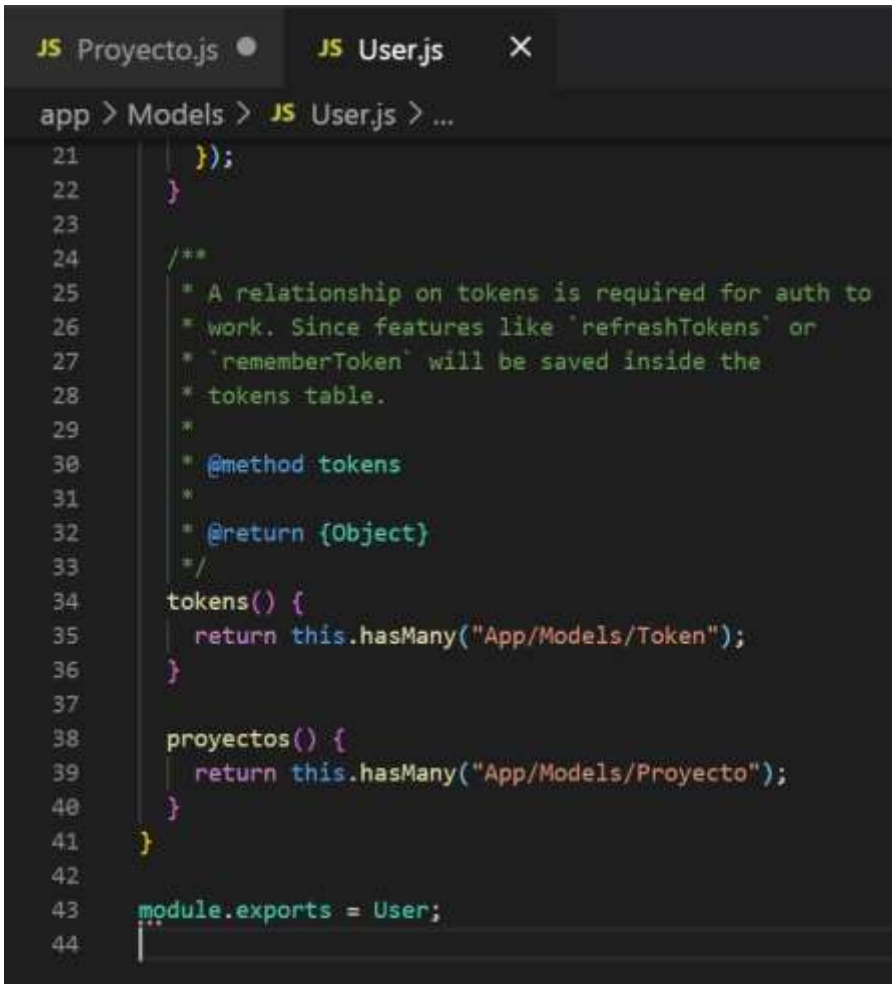
En la migración: https://adonisjs.com/docs/4.1/migrations

```
"use strict";

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

class ProyectoSchema extends Schema {
  up() {
    this.create("proyectos", (table) => {
      table.increments();
      table.integer("user_id").unsigned().references("id").inTable("users");
      table.string("name", 80).notNullable();
      table.timestamps();
    });
  }
```

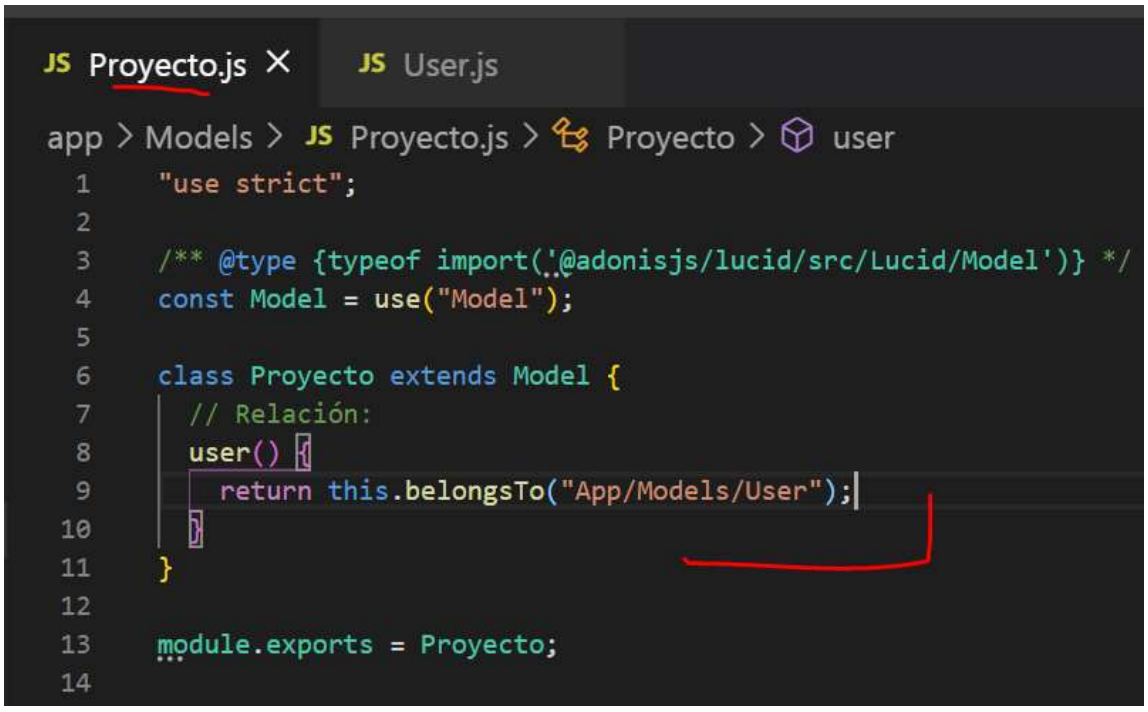
```
  down() {
    this.drop("proyectos");
  }
}

module.exports = ProyectoSchema;
```

➔Para acceder a todos los proyectos de un usuario específico, se hace uso de los modelos: en este caso:

En el modelo User:



En el modelo Proyecto (1 proyecto pertenece a un solo usuario, Belongs to):



Migrar:

```
F:\proyectos_adonisjs\rest-api>adonis migration:run
migrate: 1592600543208_proyecto_schema.js
Database migrated successfully in 2.9 s
```

Método Index

➔Creando el controlador:

```
F:\proyectos_adonisjs\rest-api>adonis make:controller Proyecto
> Select controller type For HTTP requests
√ create  app\Controllers\Http\ProyectoController.js
```

JS ProyectoController.js ×

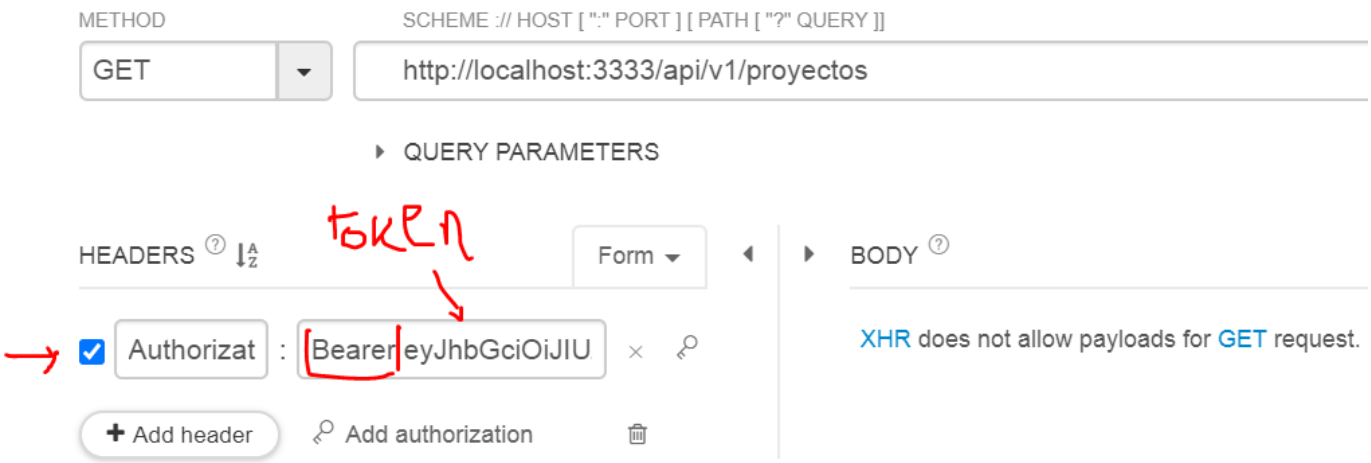
app > Controllers > Http > JS ProyectoController.js > Proyec

```
1   "use strict";
2
3   class ProyectoController {
4     /* Muestra todos los registros de la BD de proyectos
5     Sólo lo podrán ver los usuarios logueados*/
6     async index({ auth }) {
7       const user = await auth.getUser();
8       console.log(user);
9       return {
10        message: "Index de proyectos",
11        obj: user,
12      };
13    }
14  }
15
16  module.exports = ProyectoController;
17
```

➔Creando la ruta:

```
// Agrupando las rutas
Route.group(() => {
  /* Usuarios */
  Route.post("users/login", "UserController.login"); // login
  Route.post("users/register", "UserController.store"); // Registro
  /* Proyectos */
  Route.get("proyectos", "ProyectoController.index2").middleware("auth"); // Listar
}).prefix("api/v1");
```

➔Prueba:



➔Resultado:


```
{
  message: "Index de proyectos",
  obj: {
      id: 2,
      username: "daniocean@outlook.com",
      email: "daniocean@outlook.com",
      password: "$2a$10$6FUxe0Gg.wY1Ar0M3gS8jO8qCYaG4Q/hU6BrZP5u1T0AGlrcufGei",
      created_at: "2020-06-19 10:29:09",
      updated_at: "2020-06-19 10:29:09"
  }
}
```


➔Método que devuelve los proyectos del usuario logueado:

```
/* Muestra los proyectos del usuario logueado */
async index2({ auth }) {
  const user = await auth.getUser();
  return await user.proyectos().fetch();
}
```

Relación

Método Create

```
async create({ auth, request }) {
  const user = await auth.getUser();
  const data = request.only(["name"]);
  const proyecto = new Proyecto();
  proyecto.fill({ name: data.name });
  /* console.log(proyecto); */
  await user.proyectos().save(proyecto);

  return proyecto;
}
```

➔Ruta:

```
// Agrupando las rutas
Route.group(() => {
  /* Usuarios */
  Route.post("users/login", "UserController.login"); // login
  Route.post("users/register", "UserController.store"); // Registro
  /* Proyectos */
  Route.get("proyectos", "ProyectoController.index2").middleware("auth"); // Listar
  Route.post("proyectos", "ProyectoController.create").middleware("auth"); // Crear
}).prefix("api/v1");
```

Método Destroy

➔Ruta: Enviando un parámetro:

```
// Agrupando las rutas
Route.group(() => {
  /* Usuarios */
  Route.post("users/login", "UserController.login"); // login
  Route.post("users/register", "UserController.store"); // Registro
  /* Proyectos */
  Route.get("proyectos", "ProyectoController.index2").middleware("auth"); // Listar
  Route.post("proyectos", "ProyectoController.create").middleware("auth"); // Crear
  Route.delete("proyectos/:id", "ProyectoController.destroy").middleware("auth"); // Delete
}).prefix("api/v1");
```

➔Método en el controlador:

```
async destroy({ auth, response, params }) {
  const user = await auth.getUser();
  const { id } = params;
  const proyecto = await Proyecto.find(id);
  if (proyecto.user_id !== user.id) {
    return response.status(403).json({
      message: "No puede eliminar el proyecto, no es dueño",
    });
  }
  await proyecto.delete();
  return proyecto;
}
```

Servicios y Excepciones

➔Creando la carpeta “Services” en app

➔Creando una excepción:

```
F:\proyectos_adonisjs\rest-api>adonis make:exception AccesoProhidido
√ create  app\Exceptions\AccesoProhididoException.js
```

```
"use strict";

const { LogicalException } = require("@adonisjs/generic-exceptions");

class AccesoProhididoException extends LogicalException {
  /**
   * Handle this exception by itself
   */
  handle(error, { response }) {
    return response.status(403).json({
      error: "Acceso no permitido al recurso",
    });
  }
}

module.exports = AccesoProhididoException;
```

➔En el servicio creado:

```
const AccesoProhididoException = use("App/Exceptions/AccesoProhididoException");

class AutorizacionService {
  verificarPermiso(recurso, user) {
    if (recurso.user_id !== user.id) {
      throw new AccesoProhididoException();
    }
  }
}

module.exports = new AutorizacionService();
```

➔En el controlador, método destroy:

```
async destroy({ auth, response, params }) {
  const user = await auth.getUser();
  const { id } = params;
  const proyecto = await Proyecto.find(id);
  AutorizacionService.verificarPermiso(proyecto, user);
  await proyecto.delete();
  return proyecto;
}
```

➔Creando nueva excepción si no existe el ID:

```
F:\proyectos_adonisjs\rest-api>adonis make:exception RecursoNoEncontrado
√ create  app\Exceptions\RecursoNoEncontradoException.js
```

JS RecursoNoEncontradoException.js ✕ JS ProyectoController.js JS r

app > Exceptions > JS RecursoNoEncontradoException.js > ...

```
1   "use strict";
2
3   const { LogicalException } = require("@adonisjs/generic-exceptions");
4
5   class RecursoNoEncontradoException extends LogicalException {
6     /**
7      * Handle this exception by itself
8      */
9     handle(error, { response }) {
10      return response.status(404).json({
11        error: "Recurso no encontrado, no existe",
12      });
13    }
14  }
15
16  module.exports = RecursoNoEncontradoException;
```

➔Utilizando la nueva excepción en el servicio:

```
const AccesoProhididoException = use("App/Exceptions/AccesoProhididoException");
const RecursoNoEncontradoException = use(
  "App/Exceptions/RecursoNoEncontradoException"
);

class AutorizacionService {
  verificarPermiso(recurso, user) {
    if (!recurso) {
      throw new RecursoNoEncontradoException();
    }
    if (recurso.user_id !== user.id) {
      throw new AccesoProhididoException();
    }
  }
}

module.exports = new AutorizacionService();
```

Metodo Update

➔Creando el método en el controlador:

```
async update({ auth, params, request }) {
    const user = await auth.getUser();
    const { id } = params;
    const proyecto = await Proyecto.find(id);
    AutorizacionService.verificarPermiso(proyecto, user);
    proyecto.merge(request.only(["name"]));
    await proyecto.save();
    return proyecto;
  }
```

➔Creando la ruta:

```
// Agrupando las rutas
Route.group(() => {
  /* Usuarios */
  Route.post("users/login", "UserController.login"); // login
  Route.post("users/register", "UserController.store"); // Registro
  /* Proyectos */
  Route.get("proyectos", "ProyectoController.index2").middleware("auth"); // Listar
  Route.post("proyectos", "ProyectoController.create").middleware("auth"); // Crear
  Route.delete("proyectos/:id", "ProyectoController.destroy").middleware(
    "auth"
  ); // Delete
  Route.patch("proyectos/:id", "ProyectoController.update").middleware("auth"); // update
}).prefix("api/v1");
```

➔Prueba:

Creando tareas

- ⇨ **Creando tareas para los proyectos**

Creando el modelo de Tareas

Adonis make:model Tarea -m

En la migración:

```
"use strict";

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

class TareaSchema extends Schema {
  up() {
    this.create("tareas", (table) => {
      table.increments();
      table
        .integer("proyecto_id")
        .unsigned()
        .references("id")
        .inTable("proyectos");
      table.string("description", 255).notNullable();
      table.timestamps();
    });
  }

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

module.exports = TareaSchema;
```

➔Relacionar proyectos con tareas:

Proyecto:

```
"use strict";

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

class Proyecto extends Model {
  // Relaciones:
  user() {
    return this.belongsTo("App/Models/User");
  }

  tareas() {
    return this.hasMany("App/Models/Tarea");
  }
}

module.exports = Proyecto;
```

Tarea:

```
"use strict";

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

class Tarea extends Model {
  // Relaciones
  proyecto() {
    return this.belongsTo("App/Models/Proyecto");
  }
}

module.exports = Tarea;
```

➔Migrar

```
F:\proyectos_adonisjs\rest-api>adonis migration:run
migrate: 1592750574207_tarea_schema.js
Database migrated successfully in 2.26 s
```

Metodo Create para nuestras tareas

➔Creando controlador

```
F:\proyectos_adonisjs\rest-api>adonis make:controller Tarea
> Select controller type For HTTP requests
√ create  app\Controllers\Http\TareaController.js
```

```
"use strict";

const Proyecto = use("App/Models/Proyecto");
const Tarea = use("App/Models/Tarea");
const AutorizacionService = use("App/Services/AutorizacionService");

class TareaController {
  async create({ auth, request, params }) {
    const user = await auth.getUser();
    const data = request.only(["description"]);
    const { id } = params;
    const proyecto = await Proyecto.find(id);
    // ¿El usuario es dueño?
    AutorizacionService.verificarPermiso(proyecto, user);
    const tarea = new Tarea();
    tarea.fill({
      description: data.description,
    });
    await proyecto.tareas().save(tarea);
    return tarea;
  }
}

module.exports = TareaController;
```

➔Creando la ruta:

```
// Agrupando las rutas
Route.group(() => {
  /* Usuarios */
  Route.post("users/login", "UserController.login"); // login
  Route.post("users/register", "UserController.store"); // Registro
  /* Proyectos */
  Route.get("proyectos", "ProyectoController.index2").middleware("auth"); // Listar
  Route.post("proyectos", "ProyectoController.create").middleware("auth"); // Crear
  Route.delete("proyectos/:id", "ProyectoController.destroy").middleware(
    "auth"
  ); // Delete
  Route.patch("proyectos/:id", "ProyectoController.update").middleware("auth"); // update
  /* tareas */
  Route.post("proyectos/:id/tareas", "TareaController.create").middleware(
    "auth"
  ); // Registro
}).prefix("api/v1");
```

Listamos todas nuestras tareas

➔Creando método index en el controlador:

```
"use strict";

const Proyecto = use("App/Models/Proyecto");
const Tarea = use("App/Models/Tarea");
const AutorizacionService = use("App/Services/AutorizacionService");

class TareaController {
  async index({ auth, request, params }) {
    const user = await auth.getUser();
    const { id } = params; // del proyecto
    const proyecto = await Proyecto.find(id);
    // ¿El usuario es dueño?
    AutorizacionService.verificarPermiso(proyecto, user);
    return await proyecto.tareas().fetch();

  }

  async create({ auth, request, params }) {
    const user = await auth.getUser();
    const data = request.only(["description"]);
    const { id } = params;
    const proyecto = await Proyecto.find(id);
    // ¿El usuario es dueño?
    AutorizacionService.verificarPermiso(proyecto, user);
    const tarea = new Tarea();
    tarea.fill({
      description: data.description,
    });
    await proyecto.tareas().save(tarea);
    return tarea;
  }
}

module.exports = TareaController;
```

➔Creando la ruta

```
"use strict";

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

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

// Agrupando las rutas
Route.group(() => {
  /* Usuarios */
  Route.post("users/login", "UserController.login"); // login
  Route.post("users/register", "UserController.store"); // Registro
  /* Proyectos */
  Route.get("proyectos", "ProyectoController.index2").middleware("auth"); // Listar
  Route.post("proyectos", "ProyectoController.create").middleware("auth"); // Crear
  Route.delete("proyectos/:id", "ProyectoController.destroy").middleware(
    "auth"
  ); // Delete
  Route.patch("proyectos/:id", "ProyectoController.update").middleware("auth"); // update
  /* tareas */
  Route.get("proyectos/:id/tareas", "TareaController.index").middleware(
    "auth"
  ); // lista
  Route.post("proyectos/:id/tareas", "TareaController.create").middleware(
    "auth"
  ); // Registro
```

```
}).prefix("api/v1");
```

Clase Final Actualizar y Eliminar tareas

➔Agregando un campo a la BD en tareas:

```
class TareaSchema extends Schema {
  up() {
    this.create("tareas", (table) => {
      table.increments();
      table
        .integer("proyecto_id")
        .unsigned()
        .references("id")
        .inTable("proyectos");
      table.string("description", 255).notNullable();
      table.boolean("completed").defaultTo(false);
      table.timestamps();
    });
  }
```

```
F:\proyectos_adonisjs\rest-api>adonis migration:rollback
rollback: 1592750574207_tarea_schema.js
Rollback completed in 1.23 s

F:\proyectos_adonisjs\rest-api>adonis migration:run
migrate: 1592750574207_tarea_schema.js
Database migrated successfully in 1.34 s
```

➔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" };
});

// Agrupando las rutas
Route.group(() => {
  /* Usuarios */
  Route.post("users/login", "UserController.login"); // login
  Route.post("users/register", "UserController.store"); // Registro
  /* Proyectos */
  Route.get("proyectos", "ProyectoController.index2").middleware("auth"); // Listar
  Route.post("proyectos", "ProyectoController.create").middleware("auth"); // Crear
  Route.delete("proyectos/:id", "ProyectoController.destroy").middleware(
    "auth"
  ); // Delete
  Route.patch("proyectos/:id", "ProyectoController.update").middleware("auth"); // update
  /* tareas */
  Route.get("proyectos/:id/tareas", "TareaController.index").middleware("auth"); // lista
  Route.post("proyectos/:id/tareas", "TareaController.create").middleware(
    "auth"
  ); // Registro
  Route.patch("tareas/:id", "TareaController.update").middleware("auth"); // update
  Route.delete("tareas/:id", "TareaController.destroy").middleware("auth"); // delete
```

```
}).prefix("api/v1");
```

➔Creando los métodos:

```
"use strict";

const Proyecto = use("App/Models/Proyecto");
const Tarea = use("App/Models/Tarea");
const AutorizacionService = use("App/Services/AutorizacionService");

class TareaController {
  async index({ auth, request, params }) {
    const user = await auth.getUser();
    const { id } = params; // del proyecto
    const proyecto = await Proyecto.find(id);
    // ¿El usuario es dueño?
    AutorizacionService.verificarPermiso(proyecto, user);
    return await proyecto.tareas().fetch();
  }

  async create({ auth, request, params }) {
    const user = await auth.getUser();
    const data = request.only(["description"]);
    const { id } = params;
    const proyecto = await Proyecto.find(id);
    // ¿El usuario es dueño?
    AutorizacionService.verificarPermiso(proyecto, user);
    const tarea = new Tarea();
    tarea.fill({
      description: data.description,
    });
    await proyecto.tareas().save(tarea);
    return tarea;
  }

  async destroy({ auth, response, params }) {
    const user = await auth.getUser();
    const { id } = params;
    const tarea = await Tarea.find(id);
    // Proyecto que pertenece a la tarea
    const proyecto = await tarea.proyecto().fetch();
    AutorizacionService.verificarPermiso(proyecto, user);
    await tarea.delete();
    return tarea;
  }

  async update({ auth, params, request }) {
    const user = await auth.getUser();
    const { id } = params;
    const tarea = await Tarea.find(id);
    // Proyecto que pertenece a la tarea
    const proyecto = await tarea.proyecto().fetch();
    AutorizacionService.verificarPermiso(proyecto, user);
    await tarea.merge(request.only(["description", "completed"]));
    await tarea.save();
    return tarea;
  }
}

module.exports = TareaController;
```