INSTITUTO FEDERAL
Paraná
Campus Paranaguá

Ministério da Educação

TECNÓLOGO EM ANÁLISE E DESENVOLVIMENTO
Disciplina de Desenvolvimento Web II
Aula 07: Adonis JS - Autenticação e Autorização

Gil Eduardo de Andrade

Conceitos Preliminares

(https://adonisjs.com/)
(https://docs.adonisjs.com/guides/authentication/access-tokens-guard)
(https://docs.adonisjs.com/guides/security/authorization)

Introdução

No desenvolvimento de aplicações web modernas, segurança é fundamental. Sendo assim, os conceitos de autenticação e a autorização mostram-se cruciais. A autenticação permite verificar a identidade de um usuário, enquanto a autorização possibilita determinar quais ações esse usuário pode realizar.
O AdonisJS, oferece um ecossistema completo para lidar com ambos os conceitos, de forma organizada, utilizando conceitos e ferramentas, tais como: ***access tokens***, ***middleware auth***, ***Bouncer*** e ***Policies***, abordados a seguir.

Guards e Providers

- ***Guards*** são implementações completas de um tipo de login específico. O AdonisJS, oferece guards para ***session***, ***access tokens*** e ***basic auth***.
- ***Providers*** são responsáveis por buscar usuários e tokens de uma fonte de dados, como um banco de dados.

Access Tokens: Autenticação para APIs e SPAs

O ***access token guard*** é ideal para cenários onde o servidor não pode manter um estado de sessão persistente com o cliente, como em APIs consumidas por Single Page Applications (SPAs) ou aplicativos móveis. O AdonisJS utiliza *Opaque Access Tokens*, que são tokens seguros e sem informações decodificáveis publicamente, ao contrário dos JWTs.

Estrutura de um Opaque Access Token

Um opaque access token no AdonisJS é composto por:

- Um valor aleatório e criptograficamente seguro. Um sufixo de checksum (CRC32) para detecção de vazamentos.

- Um hash do token é armazenado no banco de dados para verificação.
- O valor final é codificado em Base64 e prefixado (por padrão, oat_).

Essa abordagem garante que, mesmo que um token seja exposto, ele não revele nenhuma informação sobre o usuário ou a sessão.

Configurando o Access Token Guard

Para utilizar o ***access token guard***, é necessário configurar um provedor de ***tokens*** no modelo ***User***. O AdonisJS já vem com um ***DbAccessTokensProvider*** que armazena os tokens em uma tabela no banco de dados.

```typescript
// app/models/user.ts
import { BaseModel } from '@adonisjs/lucid/orm'
import { DbAccessTokensProvider } from '@adonisjs/auth/access_tokens'
export default class User extends BaseModel {
      // ...
      static accessTokens = DbAccessTokensProvider.forModel(User, {
            expiresIn: '30 days',
            prefix: 'oat_',
            table: 'auth_access_tokens',
            type: 'auth_token',
            tokenSecretLength: 40,
      })
}
```

Emissão de Tokens

Após a autenticação bem-sucedida de um usuário (por exemplo, através de login e senha), um novo token pode ser gerado:

```typescript
import router from '@adonisjs/core/services/router'
import User from '#models/user'

router.post('login', async ({ request, auth }) => {
      const { email, password } = request.only(['email', 'password'])
      const user = await User.verifyCredentials(email, password)
      const token = await User.accessTokens.create(user)
      return {
            type: 'bearer',
```

```
            value: token.value!.release(),
            //O método release() expõe o valor do token
        }
})
```

O cliente deve, então, armazenar este token de forma segura e enviá-lo em cada requisição subsequente, geralmente no cabeçalho Authorization como um ***Bearer Token*** (credencial de segurança).

O Middleware auth : Protegendo Rotas
Uma vez que o sistema de autenticação está configurado, proteger as rotas é uma tarefa simples com o middleware auth . Este middleware intercepta as requisições e verifica se o usuário está autenticado de acordo com o guard configurado.
Primeiro, o middleware de inicialização ***@adonisjs/auth/initialize_auth_middleware*** deve ser registrado globalmente. Ele é responsável por criar uma instância do autenticador e disponibilizá-la no contexto da requisição (HttpContext).

Para proteger uma rota específica, basta aplicar o middleware auth :

TypeScript
```typescript
// start/routes.ts
import router from '@adonisjs/core/services/router'
const UsersController = () => import('#controllers/users_controller')

router.get('/profile', [UsersController, 'profile'])
      .use(middleware.auth())
```

No exemplo acima, qualquer requisição para rota ***"/profile"***, que não contenha um ***access token*** válido no cabeçalho *Authorization*, resultará em um erro 401 ***Unauthorized*** . O ***middleware auth*** utiliza o ***guard*** padrão definido no arquivo config/auth.ts . É possível especificar um guard diferente se necessário: ***.use(middleware.auth({ guards: ['api'] }))***.

Autorização: Policies

Após autenticar um usuário, o próximo passo é a autorização: determinar o que esse usuário pode ou não fazer na aplicação. Neste contexto, temos o conceito de ***policies***, que são classes criadas com o objetivo de agrupar toda a lógica de autorização relacionada a um recurso específico do sistema. Em outras palavras, as *policies* representam uma abordagem estruturada para a autorização, determinando o que um usuário autenticado pode ou não fazer na aplicação.

- Função Principal**:** as *policies* têm, como função principal, centralizar e organizar as regras de permissão para um recurso em um único local, tornando o código mais limpo e fácil de manter.

- Implementação: uma *policy* (ex: CursoPolicy) contém, noralmanente, métodos para cada ação possível no recurso (ex: `list`, `view`, `create`, `edit`, `delete`).

Controle de Acesso Baseado em Papéis

Para implementação do controle de acesso baseado em papéis (Role-Based Access Control) usando *Policies*, utilizaremos, nesta aula, a seguinte lógica:

- Definição dos Papéis (Roles): será criada uma classe de modelo *Role*, vinculada a tabela *roles*, contendo os possíveis papéis que um usuário possui na aplicação (neste caso "1 - Coordenador" e "2 - Professor"). Aos usuários cadastrados no sistema, será vinculado um papel específico, através da adição de uma chave estrangeira "*role_id*" na tabela *users*.

- Mapeamento de Permissões: será criado/definido um arquivo de mapeamento das permissões (*"/app/utils/permissions.ts"*), que permite associar os "*role_id*" a um conjunto específico de possíveis ações possíveis que usuário tem ou não permissão de efetuar para cada recurso da aplicação. ***OBS.:*** as permissões poderiam ser armazenadas numa tabela, caso a aplicação necessitasse de definições dinâmicas para as permissões (as permissões pudessem mudar ao longo do tempo, por um usuário do sistema, como um administrador, por exemplo).

- Verificação na *Policy*: cada método da Policy verifica se o usuário logado possui permissão para efetuar uma determinada ação sobre o recurso em questão. A *policy* retorna "*true*" quando o usuário possui permissão e *"false"* quando ele não possui.

Camada de Verificação de Permissão

A verificação de autorização (permissão) do usuário ocorre na Controller (pode ser vista como uma camada intermediária entre apresentação e aplicação). Para tal, e

considerando boas práticas de desenvolvimento, a classe base de controle será adaptada, recebendo, por injeção de dependência (de forma genérica), qual policy deve utilizar para efetuar o controle de acesso.
Essa abordagem garante que, após a autenticação, as ações do usuário sejam estritamente limitadas pelo papel atribuído, conforme as regras definidas no sistema de permissões.

INSERINDO PROCESSO DE AUTENTICAÇÃO

Access Token / Middleware Auth

(Aplicação da Aula Anterior)

OS PASSOS A SEGUIR SÃO UTILIZANDO DOCKER (Linux)
Continuação da Aplicação da Aula Anterior

---

1. Configurando Variáveis de Ambiente (BD) *“.env”*

TypeScript

```
DB_CONNECTION=mysql
MYSQL_HOST=db
MYSQL_PORT=3306
MYSQL_USER=root
MYSQL_PASSWORD=root
MYSQL_DB_NAME=adonis_app
```

---

1. Efetuando a Migração / Seeder

Comando

(no terminal de comando)

Shell

```
docker compose exec --user $(id -u):$(id -g) app node ace
                migration:fresh --seed
```

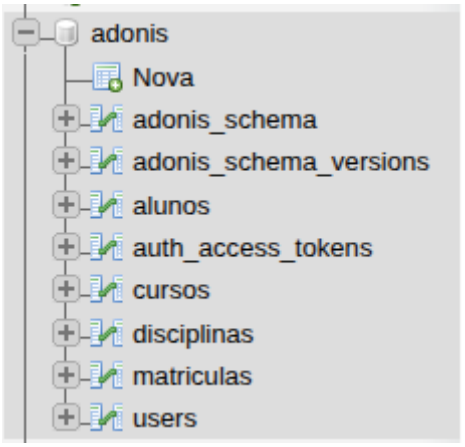


Tabela de Cursos

+ Opções

	id	nome	duracao	created_at	updated_at
Edita Copiar Apagar	1	Técnico em Informática	4	2025-09-07 19:57:00	2025-09-07 19:57:00
Edita Copiar Apagar	2	Técnólogo em Análise e Desenvolvimento	3	2025-09-07 19:57:00	2025-09-07 19:57:00

Tabela de Disciplinas

+ Opções

	id	nome	carga	curso_id	created_at	updated_at
Edita Copiar Apagar	1	Programação Paralela e Distribuída	2	2	2025-09-07 19:57:00	2025-09-07 19:57:00
Edita Copiar Apagar	2	Desenvolvimento Mobile	4	2	2025-09-07 19:57:00	2025-09-07 19:57:00
Edita Copiar Apagar	3	Linguagem de Programação	4	1	2025-09-07 19:57:00	2025-09-07 19:57:00
Edita Copiar Apagar	4	Matemática I	3	1	2025-09-07 19:57:00	2025-09-07 19:57:00

Tabela de Alunos

+ Opções

	id	nome	curso_id 1	created_at	updated_at
Edita Copiar Apagar	3	Ana Luisa (Técnico em Informática)	1	2025-09-07 19:57:00	2025-09-07 19:57:00
Edita Copiar Apagar	4	Luiz Eduardo (Técnico em Informática)	1	2025-09-07 19:57:00	2025-09-07 19:57:00
Edita Copiar Apagar	1	Ana Luisa (Técnólogo em Análise e Desenvolvimento)	2	2025-09-07 19:57:00	2025-09-07 19:57:00
Edita Copiar Apagar	2	Luiz Eduardo (Técnólogo em Análise e Desenvolvimen...	2	2025-09-07 19:57:00	2025-09-07 19:57:00

---

2. Criando *Service, Controller* e *Validator* de Autenticação

Comandos

(no terminal de comando)

Interfaces

None

```
- Criar a pasta "interfaces" dentro da pasta "app"
```

```
- Criar o arquivo "auth_interface.ts" na pasta "interfaces"
```

Repository

```
None
- Criar o arquivo "auth_repository.ts" dentro da pasta "app/repositories"
```

Service

```shell
docker compose exec --user $(id -u):$(id -g) app node ace make:service AuthService
```

Controller

```shell
docker compose exec --user $(id -u):$(id -g) app node ace make:controller AuthController
```

Validator

```shell
docker compose exec --user $(id -u):$(id -g) app node ace make:validator Auth
```

INTERFACES

Arquivo: "/app/interfaces/auth_interface.ts"

```typescript
export interface PayloadUser {
  fullName: string | undefined
  email: string
  password: string
}
```

```
export interface LoginInput {
  email: string
  password: string
}

export interface UserOutput {
  user: {
    id: number
    fullName: string | null
    email: string
    createdAt: Date | null
    updatedAt: Date | null
  }
}

export interface LoginOutput extends UserOutput {
  token: {
    type: string
    value: string
    expiresAt: Date | null
  }
}

export interface UserToken {
  name: string | null
  type: string
  abilities: string[]
  lastUsedAt: Date | null
  expiresAt: Date | null
  createdAt: Date | null
}
```

Arquivo: “package.json”

JSON

```
"imports": {
      "#controllers/*": "./app/controllers/*.js",
      "#exceptions/*": "./app/exceptions/*.js",
      ...
      "#config/*": "./config/*.js",
```

```
    "#repositories/*": "./app/repositories/*.js",
    "#interfaces/*": "./app/interfaces/*.js"
},
```

REPOSITORY

Arquivo: "/app/repositories/auth_repository.ts"

TypeScript

```typescript
import type { HttpContext } from '@adonisjs/core/http'
import User from '#models/user'
import type {
  PayloadUser,
  LoginInput,
  UserOutput,
  LoginOutput,
  UserToken,
} from '#interfaces/auth_interface'

export default class AuthRepository {
  async createUser(payload: PayloadUser): Promise<any> {
    const user = await User.create(payload)
    // user: usuário logado;
    // *: coringa, token pode acessar todas as rotas protegidas;
    const token = await User.accessTokens.create(user, ['*'], {
      name: 'Registration Token',
      expiresIn: '30 days', // define a validade do Token;
    })

    return {
      message: 'Usuário registrado com sucesso',
      user: {
        id: user.id,
        fullName: user.fullName,
        email: user.email,
        createdAt: user.createdAt,
      },
      token: {
        type: 'bearer', // padrão (OAuth 2.0) tipo de token que cliente
        value: token.value!.release(), // string limpa/utilizável (token)
```

```
      expiresAt: token.expiresAt, // data e hora da expiração do token;
    },
  }
}

async authenticateUser(payload: LoginInput): Promise<LoginOutput> {
  const user = await User.verifyCredentials(payload.email, payload.password)
  // Cria o token de acesso
  const token = await User.accessTokens.create(user, ['*'], {
    name: 'Login Token',
    expiresIn: '30 days',
  })

  return {
    user: {
      id: user.id,
      fullName: user.fullName,
      email: user.email,
      createdAt: user.createdAt,
      updatedAt: user.updatedAt,
    },
    token: {
      type: 'bearer',
      value: token.value!.release(),
      expiresAt: token.expiresAt,
    },
  }
}

async deauthenticateUser({ auth }: HttpContext): Promise<boolean> {
  const user = auth.getUserOrFail()
  const token = auth.user?.currentAccessToken

  if (token) {
    await User.accessTokens.delete(user, token.identifier)
    return true
  }

  return false
}
```

```
  async getUserAuthenticated({ auth }: HttpContext): Promise<UserOutput> {
    const user = auth.getUserOrFail()

    return {
      user: {
        id: user.id,
        fullName: user.fullName,
        email: user.email,
        createdAt: user.createdAt,
        updatedAt: user.updatedAt,
      },
    }
  }

  async getListTokens({ auth }: HttpContext): Promise<UserToken[]> {
    const user = auth.getUserOrFail()
    const tokens = await User.accessTokens.all(user)
    const tokensList: UserToken[] = tokens.map((token) => ({
      name: token.name,
      type: token.type,
      abilities: token.abilities,
      lastUsedAt: token.lastUsedAt,
      expiresAt: token.expiresAt,
      createdAt: token.createdAt,
    }))

    return tokensList
  }

   async createNewToken({ auth, request }: HttpContext): Promise<any>
{
    const user = auth.getUserOrFail()
    const { name, abilities, expiresIn } = request.only(['name', 'abilities', 'expiresIn'])

    const token = await User.accessTokens.create(user, abilities || ['*'], {
      name: name || 'API Token',
      expiresIn: expiresIn || '30 days',
    })

    return {
      message: 'Token criado com sucesso',
```

```
      token: {
        type: 'bearer',
        value: token.value!.release(),
        name: token.name,
        abilities: token.abilities,
        expiresAt: token.expiresAt,
      },
    }
  }
}
```

SERVICE

Arquivo: "/app/services/auth_service.ts"

TypeScript

```typescript
import type { HttpContext } from '@adonisjs/core/http'
import User from '#models/user'
import AuthRepository from '#repositories/auth_repository'
import { inject } from '@adonisjs/core'
import type {
  PayloadUser,
  LoginInput,
  UserOutput,
  LoginOutput,
  UserToken,
} from '#interfaces/auth_interface'

@inject()
export class AuthService {
  protected repository: AuthRepository

  constructor(repository: AuthRepository) {
    this.repository = repository
  }

  async register(payload: PayloadUser): Promise<any> {
    return this.repository.createUser(payload)
  }
```

```
  async login(payload: LoginInput): Promise<LoginOutput> {
    return this.repository.authenticateUser(payload)
  }

  async logout(httpContext: HttpContext): Promise<boolean> {
    return this.repository.deauthenticateUser(httpContext)
  }
  async user(httpContext: HttpContext): Promise<UserOutput> {
    return this.repository.getUserAuthenticated(httpContext)
  }

  async tokens(httpContext: HttpContext): Promise<UserToken[]> {
    return this.repository.getListTokens(httpContext)
  }

  async createToken(httpContext: HttpContext): Promise<any> {
    return this.repository.createNewToken(httpContext)
  }
}
```

VALIDATOR

Arquivo: "/app/validators/auth.ts"

TypeScript

```
/**
 * Validator para registro de usuário
 */
export const registerValidator = vine.compile(
  vine.object({
    fullName: vine.string().trim().minLength(2).maxLength(100).optional(),
    email: vine
      .string()
      .email()
      .normalizeEmail()
      .unique(async (db, value) => {
        const user = await db.from('users').where('email', value).first()
        return !user
      }),
    password: vine.string().minLength(8).maxLength(32),
```

```
  })
)
/**
 * Validator para login de usuário
 */
export const loginValidator = vine.compile(
  vine.object({
    email: vine.string().email().normalizeEmail(),
    password: vine.string(),
  })
)
```

CONTROLLER (Registro)

Arquivo: "/app/controllers/auth_controllers.ts"

TypeScript

```
import { inject } from '@adonisjs/core'
import type { HttpContext } from '@adonisjs/core/http'
import { loginValidator, registerValidator } from '#validators/auth'
import { AuthService } from '#services/auth_service'

@inject()
export default class AuthController {
  protected service: AuthService

  constructor(service: AuthService) {
    this.service = service
  }

  /* Efetuar Registro do Usuário */
  async register({ request, response }: HttpContext) {
    const payload = await request.validateUsing(registerValidator)
    const resp = await this.service.register(payload)

    return response.created({
      message: 'Usuário registrado com sucesso',
      resp,
    })
  }
```

```
/* Efetuar Autenticação do Usuário */
async login({ request, response }: HttpContext) {
  const { email, password } = await request.validateUsing(loginValidator)
  const resp = await this.service.login({ email, password })

  response.ok({
    message: 'Login realizado com sucesso',
    resp,
  })
}

async logout(httpcontext: HttpContext) {
  const { response } = httpcontext

  if (await this.service.logout(httpcontext)) {
    return response.ok({
      message: 'Logout Realizado com Sucesso',
    })
  }

  return response.unauthorized({
    message: 'Token Inválido',
  })
}

/* Retorna Informações do Usuário Autenticado */
async me(httpcontext: HttpContext) {
  const { response } = httpcontext
  const resp = await this.service.user(httpcontext)

  return response.ok(resp)
}

/* Listar Todos os Tokens do Usuário Autenticado */
async tokens(httpcontext: HttpContext) {
  const { response } = httpcontext
  const resp = await this.service.tokens(httpcontext)

  return response.ok(resp)
}
```

```
  /* Criar um novo token para o usuário autenticado */
  async createToken(httpcontext: HttpContext) {
    const { response } = httpcontext
    const resp = await this.service.createToken(httpcontext)

    return response.ok({
      message: 'Token Criado com Sucesso',
      resp,
    })
  }
}
```

---

3. Configurando Rotas e Middleware

Arquivo: "/start/routes.ts"

```
TypeScript
// Rotas de autenticação (públicas)
router.group(() => {
  router.post('/register', '#controllers/auth_controller.register')
  router.post('/login', '#controllers/auth_controller.login')
  // Rotas protegidas de autenticação
  router.post('/logout', '#controllers/auth_controller.logout')
      .use(middleware.auth())
  router.get('/me', '#controllers/auth_controller.me')
      .use(middleware.auth())
  router.get('/tokens', '#controllers/auth_controller.tokens')
      .use(middleware.auth())
  router.post('/tokens', '#controllers/auth_controller.createToken')
      .use(middleware.auth())
}).prefix('/auth')
```

---

4. Configurando o "*Global Exception Handler*"

Arquivo: "app/exceptions/handler.ts"

TypeScript

```
...
   /**
   * O método handle converte a exceção em uma resposta HTTP
   */
  async handle(error: unknown, ctx: HttpContext) {
    /**
     * Erros de Validação (VineJS)
     * O Adonis já trata isso automaticamente, mas você pode customizar aqui
     */
    if (error.code === 'E_VALIDATION_ERROR') {
      return ctx.response.status(422).send({
        errors: error.messages,
      })
    }

    /**
     * 2. Erros de Registro Não Encontrado (Lucid findOrFail)
     */
    if (error instanceof lucideErrors.E_ROW_NOT_FOUND) {
      return ctx.response.status(404).send({
        message: 'O recurso solicitado não foi encontrado no banco de dados.',
      })
    }

    /**
     * 3. Erros de Autenticação
     */
    if (error instanceof authErrors.E_UNAUTHORIZED_ACCESS) {
      return ctx.response.status(401).unauthorized({
        message: 'Você precisa estar logado e ter permissão para acessar este recurso.',
      })
    }

    /**
     * 4. Erros Personalizados (Custom Exceptions)
     * Se você criou uma exceção com 'node ace make:exception'
     */
    if (error.code === 'E_INSUFFICIENT_FUNDS') {
      return ctx.response.status(400).send({
        message: error.message,
      })
    }
```

```
    /* 5. Fallback para erros genéricos (500) */
    return super.handle(error, ctx)
  }
...
```

---

5. Executando e Testando a Aplicação

Comando

(no terminal de comando)

```
Shell
docker compose up
```

REGISTRAR USUÁRIO

(*register - no terminal de comando)*

```
Shell
curl -X POST http://localhost:12000/auth/register \
-H "Content-Type: application/json" \
-d '{"fullName": "João Silva", "email": "joao@exemplo.com",
"password": "senha123456"}'
```

```
 % curl -X POST http://localhost:12000/auth/register \
-H "Content-Type: application/json" \
-d '{"fullName": "João Silva", "email": "joao@exemplo.com", "password": "senha123456"}'
{"message":"Usuário registrado com sucesso","resp":{"message":"Usuário registrado com sucesso","user":{"
id":2,"fullName":"João Silva","email":"joao@exemplo.com","createdAt":"2026-04-13T18:23:07.387+00:00"},"t
oken":{"type":"bearer","value":"oat_MQ.MGdKSVJYejg3djhfSlNJWXpTNHNFZHYxaHVZYkJpRWx1X2FhZkhCZDMzOTE5ODM3M
TQ","expiresAt":"2026-05-13T18:23:07.400Z"}}}%
```

Token: oat_MQ.MGdKSVJYejg3djhfSlNJWXpTNHNFZHYxaHVZYkJpRWx1X2FhZkhCZDMzOTE5ODM3MTQ

(oat - Opaque Access Tokens)

ACESSAR ROTA PROTEGIDA “ME”

(*no terminal de comando)*

```
Shell
curl -X GET http://localhost:12000/auth/me \
```

```
-H "Authorization: Bearer
oat_MQ.MGdKSVJYejg3djhfSlNJWXpTNHNFZHYxaHVZYkJpRWx1X2FhZkhCZDMzOTE5OD
M3MTQ"
```

```
% curl -X GET http://localhost:12000/auth/me \
-H "Authorization: Bearer oat_MQ.MGdKSVJYejg3djhfSlNJWXpTNHNFZHYxaHVZYkJpRWx1X2FhZkhCZDMzOTE5ODM3MTQ"
{"user":{"id":2,"fullName":"João Silva","email":"joao@exemplo.com","createdAt":"2026-04-13T18:23:07.000+
00:00","updatedAt":"2026-04-13T18:23:07.000+00:00"}}%
```

[AUTORIZADO]

ACESSAR ROTA PROTEGIDA “TOKENS”
(*no terminal de comando)*

```
Shell
curl -X GET http://localhost:12000/auth/tokens \
-H "Authorization: Bearer
oat_MQ.MGdKSVJYejg3djhfSlNJWXpTNHNFZHYxaHVZYkJpRWx1X2FhZkhCZDMzOTE5OD
M3MTQ"
```

```
% curl -X GET http://localhost:12000/auth/tokens \
-H "Authorization: Bearer oat_MQ.MGdKSVJYejg3djhfSlNJWXpTNHNFZHYxaHVZYkJpRWx1X2FhZkhCZDMzOTE5ODM3MTQ"
[{"name":"Registration Token","type":"auth_token","abilities":["*"],"lastUsedAt":"2026-04-13T18:32:34.00
0Z","expiresAt":"2026-05-13T18:23:07.000Z","createdAt":"2026-04-13T18:23:07.000Z"}]%
```

[AUTORIZADO]

EFETUAR LOGOUT
(*logout - no terminal de comando)*

```
Shell
curl -X POST http://localhost:12000/auth/logout \
-H "Authorization: Bearer
oat_MQ.MGdKSVJYejg3djhfSlNJWXpTNHNFZHYxaHVZYkJpRWx1X2FhZkhCZDMzOTE5OD
M3MTQ"
```

```
% curl -X POST http://localhost:12000/auth/logout \
-H "Authorization: Bearer oat_MQ.MGdKSVJYejg3djhfSlNJWXpTNHNFZHYxaHVZYkJpRWx1X2FhZkhCZDMzOTE5ODM3MTQ"
{"message":"Logout Realizado com Sucesso"}%
```

[AUTORIZADO]

ACESSAR ROTA PROTEGIDA “ME”
(*no terminal de comando)*

```shell
curl -X GET http://localhost:12000/auth/me \
-H "Authorization: Bearer
oat_MQ.MGdKSVJYejg3djhfSlNJWXpTNHNFZHYxaHVZYkJpRWx1X2FhZkhCZDMzOTE5OD
M3MTQ"
```

```
% curl -X GET http://localhost:12000/auth/me \
-H "Authorization: Bearer oat_MQ.MGdKSVJYejg3djhfSlNJWXpTNHNFZHYxaHVZYkJpRWx1X2FhZkhCZDMzOTE5ODM3MTQ"
{"message":"Você precisa estar logado para acessar este recurso."}%
```

[NÃO AUTORIZADO]

EFETUAR LOGIN

(*login - no terminal de comando)*

```shell
curl -X POST http://localhost:12000/auth/login \
-H "Content-Type: application/json" \
-d '{"email": "joao@exemplo.com", "password": "senha123456"}'
```

```
% curl -X POST http://localhost:12000/auth/login \
-H "Content-Type: application/json" \
-d '{"email": "joao@exemplo.com", "password": "senha123456"}'
{"message":"Login realizado com sucesso","resp":{"user":{"id":2,"fullName":"João Silva","email":"joao@ex
emplo.com","createdAt":"2026-04-13T18:23:07.000+00:00","updatedAt":"2026-04-13T18:23:07.000+00:00"},"tok
en":{"type":"bearer","value":"oat_Mg.TE9yM2dvdDhMR2F5VktpaU9pdDhDdEYwQmZjc3h6RVh4eklVNHpNWDM0NjI4MzMzMTQ
","expiresAt":"2026-05-13T18:46:52.632Z"}}}%
```

Token: oat_Mg.TE9yM2dvdDhMR2F5VktpaU9pdDhDdEYwQmZjc3h6RVh4eklVNHpNWDM0NjI4MzMzMTQ

ACESSAR ROTA PROTEGIDA “ME”

(*no terminal de comando)*

```shell
curl -X GET http://localhost:12000/auth/me \
-H "Authorization: Bearer
oat_Mg.TE9yM2dvdDhMR2F5VktpaU9pdDhDdEYwQmZjc3h6RVh4eklVNHpNWDM0NjI4Mz
MzMTQ"
```

```
% curl -X GET http://localhost:12000/auth/me \
-H "Authorization: Bearer oat_Mg.TE9yM2dvdDhMR2F5VktpaU9pdDhDdEYwQmZjc3h6RVh4eklVNHpNWDM0NjI4MzMzMTQ"
{"user":{"id":2,"fullName":"João Silva","email":"joao@exemplo.com","createdAt":"2026-04-13T18:23:07.000+
00:00","updatedAt":"2026-04-13T18:23:07.000+00:00"}}%
```

[AUTORIZADO]

INSTITUTO FEDERAL
Paraná
Campus Paranaguá

Ministério da Educação

INSERINDO PROCESSO DE AUTORIZAÇÃO

Bouncer / Policies

1. Criando Modelo e Migração - Papel do Usuário (Role)

Comando

(no terminal de comando)

```shell
docker compose exec --user $(id -u):$(id -g) app node ace make:model Role -m
```

OBS.: renomear a "*migration*" de role, para que ela seja criada antes da migração do usuário.

Arquivo: "/database/migrations/…_create_roles_table.ts"

```typescript
async up() {
    this.schema.createTable(this.tableName, (table) => {
      table.increments('id')
      table.string('nome')
      table.timestamp('created_at')
      table.timestamp('updated_at')
    })
}
```

Arquivo: "/database/migrations/…_create_roles_table.ts"

```typescript
async up() {
    this.schema.createTable(this.tableName, (table) => {
      table.increments('id')
      table.string('nome')
      table.timestamp('created_at')
      table.timestamp('updated_at')
    })
}
```

Arquivo: "/app/models/role.ts

```typescript
import { RoleSchema } from '#database/schema'
import { hasMany } from '@adonisjs/lucid/orm'
import type { HasMany } from '@adonisjs/lucid/types/relations'
import User from '#models/user'

export default class Role extends RoleSchema {
  // Relacionamentos
  @hasMany(() => User)
  declare usuarios: HasMany<typeof User>
}
```

---

2. Atualizando a Migração *"User"*

Comando

(no terminal de comando)

Arquivo: "/database/migrations/…_create_users_table.ts"

```typescript
async up() {
    this.schema.createTable(this.tableName, (table) => {
      table.increments('id').notNullable()
      table.string('full_name').nullable()
      table.string('email', 254).notNullable().unique()
      table.string('password').notNullable()

      table.integer('papel_id').unsigned().references('id')
            .inTable('papels')

      table.timestamp('created_at').notNullable()
      table.timestamp('updated_at').nullable()
    })
  }
```

---

3. Atualizando a Modelo *"User"*

Comando

(no terminal de comando)

Arquivo: "/app/models/user.ts

```typescript
...
import { belongsTo } from '@adonisjs/lucid/orm'
import type { BelongsTo } from '@adonisjs/lucid/types/relations'
import Role from '#models/role'

export default class User extends compose(BaseModel, AuthFinder) {
  ...
  // Relacionamentos
  @belongsTo(() => Role)
  declare role: BelongsTo<typeof Role>
}
```

---

4. Criando *Seeders*: *Role* e *User*

Comando

(no terminal de comando)

```shell
docker compose exec --user $(id -u):$(id -g) app node ace
                    make:seeder 04_Role

docker compose exec --user $(id -u):$(id -g) app node ace
                    make:seeder 05_User
```

Arquivo: "/database/seeders/04_role_seeder.ts"

```typescript
import { BaseSeeder } from '@adonisjs/lucid/seeders'
import Role from '#models/role'

export default class extends BaseSeeder {
```

```
  async run() {
    // Write your database queries inside the run method
    await Role.createMany([
      {
        nome: 'Coordenador',
      },
      {
        nome: 'Professor',
      },
    ])
  }
}
```

Arquivo: "/database/seeders/05_user_seeder.ts"

TypeScript

```
import { BaseSeeder } from '@adonisjs/lucid/seeders'
import User from '#models/user'

export default class extends BaseSeeder {
  async run() {
    // Write your database queries inside the run method
    await User.createMany([
      {
        fullName: 'Gil Eduardo de Andrade',
        email: 'gil@gmail.com',
        password: '1234@5678',
        role_id: 1,
      },
      {
        fullName: 'Manuel Araújo Castro',
        email: 'manuel@gmail.com',
        password: '1234@5678',
        role_id: 2,
      },
    ])
  }
}
```

5. Efetuando a Migração / Seeder

Comando

(no terminal de comando)

```shell
docker compose exec --user $(id -u):$(id -g) app node ace migration:run

#ou

docker compose exec --user $(id -u):$(id -g) app node ace migration:fresh --seed
```

6. Construindo as Policies / Configurando Path

(Criando pasta e Arquivos)

- Criar a pasta ***"/app/policies"***
  - Dentro da nova pasta => Criar o arquivo ***"curso_policy.ts"***

Arquivo: "/app/policies/curso_policy.ts"

```typescript
import type User from '#models/user'
import { permissions } from '../utils/permissions.ts'

export default class CursoPolicy {
  list(user: User | undefined): boolean {
    if (!user) return false
    return permissions[user.roleId].listCurso
  }

  view(user: User | undefined): boolean {
    if (!user) return false
    return permissions[user.roleId].viewCurso
  }

  create(user: User | undefined): boolean {
    if (!user) return false
    return permissions[user.roleId].createCurso
  }

  edit(user: User | undefined): boolean {
    if (!user) return false
    return permissions[user.roleId].editCurso
  }

  delete(user: User | undefined): boolean {
    if (!user) return false
    return permissions[user.roleId].deleteCurso
  }
}
```

Adicionando o novo diretório “policies” ao alias “#”

Arquivo: “package.json”

```json
"imports": {
      "#controllers/*": "./app/controllers/*.js",
      "#exceptions/*": "./app/exceptions/*.js",
      ...
      "#config/*": "./config/*.js",
```

```
      "#repositories/*": "./app/repositories/*.js"
      "#policies/*": "./app/policies/*.js",
},
```

---

7. Criando *Utils* - Permissões

(criar pasta e arquivo diretamente no Visual Code)

- Criar a pasta ***"/app/utils"***
  - Dentro da nova pasta => Criar o arquivo ***"permissions.ts"***

Arquivo: "/app/utils/permissions.ts"

TypeScript

```typescript
export const permissions = [
  {
    listCurso: false,
    viewCurso: false,
    createCurso: false,
    editCurso: false,
    deleteCurso: false,
  },
  // COORDENADOR - 1
  {
    listCurso: true,
    viewCurso: true,
    createCurso: true,
    editCurso: true,
    deleteCurso: true,
  },
  // PROFESSOR - 2
  {
    listCurso: true,
    viewCurso: true,
    createCurso: true,
    editCurso: false,
    deleteCurso: false,
  },
]
```

8. Alterando "*BaseController*" e "*CursoController*"

Arquivo: "/app/controllers/bases_controller.ts"

TypeScript

```
import type { HttpContext } from '@adonisjs/core/http'
import type { BaseCrudInterface } from '#controllers/interfaces/base_crud_interface'
import type { ControllerValidators } from '#controllers/interfaces/validator_inerface'
import type CursoPolicy from '#policies/curso_policy';

export abstract class BasesController<
  T,
  P extends CursoPolicy,
  V extends ControllerValidators,
> implements BaseCrudInterface {
  // Recebemos o service e o schema do validator no construtor
  constructor(
    protected service: T,
    protected policy: P,
    protected validators: V
  ) {}

  async index({ response, auth }: HttpContext) {
    if (!this.policy.list(auth.user)) {
      return response.forbidden({ message: 'Você não tem permissão para listar esse recurso!' })
    }
    // const dados = await (this.service as any).indexWith(['alunos', 'disciplinas'])
    const dados = await (this.service as any).index()
    return response.status(200).json({
      message: 'OK',
      data: dados,
    })
  }

  async store({ request, response, auth }: HttpContext) {
    if (!this.policy.create(auth.user)) {
      return response.forbidden({ message: 'Você não tem permissão para criar esse recurso!' })
    }
    const payload = await request.validateUsing(this.validators.criar)
    const curso = await (this.service as any).store(payload)
    if (!curso) {
```

```
      return response.status(422).json({
        message: 'ERROR',
      })
    }
    return response.status(201).json({
      message: 'OK',
      data: curso,
    })
  }

  async show({ params, response, auth }: HttpContext) {
    if (!this.policy.view(auth.user)) {
      return response.forbidden({ message: 'Você não tem permissão para visualizar esse recurso!' })
    }
    const curso = await (this.service as any).show(Number(params.id))
    if (!curso) {
      return response.status(404).json({
        message: 'NOT FOUND',
      })
    }
    return response.status(200).json({
      message: 'OK',
      data: curso,
    })
  }

  async update({ params, request, response, auth }: HttpContext) {
    if (!this.policy.edit(auth.user)) {
      return response.forbidden({ message: 'Você não tem permissão para modificar esse recurso!' })
    }
    const payload = await request.validateUsing(this.validators.alterar)
    const curso = await (this.service as any).update(Number(params.id), payload)
    if (!curso) {
      return response.status(422).json({
        message: 'NOT FOUND',
      })
    }
    return response.status(202).json({
      message: 'OK',
      data: curso,
    })
```

```
  }

  async destroy({ params, response, auth }: HttpContext) {
    if (!this.policy.delete(auth.user)) {
      return response.forbidden({ message: 'Você não tem permissão para remover esse recurso!' })
    }
    const course = (this.service as any).delete(Number(params.id))
    if (!course) {
      return response.status(404).json({
        message: 'NOT FOUND',
      })
    }
    return response.status(200).json({
      message: 'OK',
      data: course,
    })
  }
}
```

Arquivo: "/app/controllers/cursos_controller.ts"

TypeScript

```
import { inject } from '@adonisjs/core'
import { CursoService } from '#services/curso_service'
import { criarCurso, alterarCurso } from '#validators/curso'
import { BasesController } from './bases_controller.ts'
import CursoPolicy from '#policies/curso_policy'

@inject() // Habilita injeção automática no Adonis
export default class UsersController extends BasesController<
  CursoService,
  CursoPolicy,
  { criar: typeof criarCurso, alterar: typeof alterarCurso },
> {
  constructor(
    protected service: CursoService,
    protected policy: CursoPolicy
  ) {
    super(service, policy, { criar: criarCurso, alterar: alterarCurso })
  }
```

```
}
```

---

9. Aplicando Middleware - Rotas de Cursos

Arquivo: "/app/start/routes.ts"

```TypeScript
...
router.resource('cursos', '#controllers/cursos_controller')
      .use('*', middleware.auth())
...
```

---

10. Executando e Testando a Aplicação

Comando

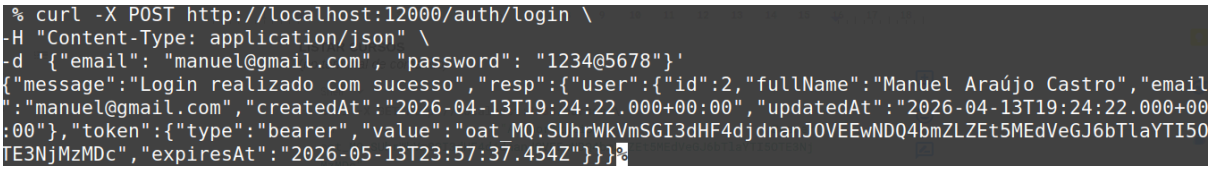
(no terminal de comando)

```Shell
docker compose up
```

EFETUAR LOGIN (Professor)

(*login - no terminal de comando)*

```Shell
curl -X POST http://localhost:12000/auth/login \
-H "Content-Type: application/json" \
-d '{"email": "manuel@gmail.com", "password": "1234@5678"}'
```



```
% curl -X POST http://localhost:12000/auth/login \
-H "Content-Type: application/json" \
-d '{"email": "manuel@gmail.com", "password": "1234@5678"}'
{"message":"Login realizado com sucesso","resp":{"user":{"id":2,"fullName":"Manuel Araújo Castro","email
":"manuel@gmail.com","createdAt":"2026-04-13T19:24:22.000+00:00","updatedAt":"2026-04-13T19:24:22.000+00
:00"},"token":{"type":"bearer","value":"oat_MQ.SUhrWkVmSGI3dHF4djdnanJOVEEwNDQ4bmZLZEt5MEdVeGJ6bTlaYTI5O
TE3NjMzMDc","expiresAt":"2026-05-13T23:57:37.454Z"}}}%
```

Token: oat_MQ.SUhrWkVmSGI3dHF4djdnanJOVEEwNDQ4bmZLZEt5MEdVeGJ6bTlaYTI5OTE3NjMzMDc

---

LISTAR CURSOS

(*no terminal de comando)*

```
Shell
curl -X GET http://localhost:12000/cursos \
-H "Authorization: Bearer oat_MQ.SUhrWkVmSGI3dHF4djdnanJOVEEwNDQ4bmZLZEt5MEdVeGJ6bTlaYTI5OTE3NjMzMDc"
```

```
% curl -X GET http://localhost:12000/cursos \
-H "Authorization: Bearer oat_MQ.SUhrWkVmSGI3dHF4djdnanJOVEEwNDQ4bmZLZEt5MEdVeGJ6bTlaYTI5OTE3NjMzMDc"
{"message":"OK","data":[{"id":2,"nome":"Técnólogo em Análise e Desenvolvimento","duracao":3,"createdAt"
"2026-04-13T19:24:22.000+00:00","updatedAt":"2026-04-13T19:24:22.000+00:00"},{"id":1,"nome":"Técnico em
Informática","duracao":4,"createdAt":"2026-04-13T19:24:22.000+00:00","updatedAt":"2026-04-13T19:24:22.0
0+00:00"}]}%
```

[AUTORIZADO]

CRIAR CURSOS

(*no terminal de comando)*

```
Shell
curl -X POST \
-H "Content-Type: application/json" \
-H "Authorization: Bearer oat_MQ.SUhrWkVmSGI3dHF4djdnanJOVEEwNDQ4bmZLZEt5MEdVeGJ6bTlaYTI5OTE3NjMzMDc" \
-d '{"nome": "Técnico em Mecânica", "duracao": 4}' \
http://localhost:12000/cursos
```

```
% curl -X POST \
-H "Content-Type: application/json" \
-H "Authorization: Bearer oat_MQ.SUhrWkVmSGI3dHF4djdnanJOVEEwNDQ4bmZLZEt5MEdVeGJ6bTlaYTI5OTE3NjMzMDc" \
-d '{"nome": "Técnico em Mecânica", "duracao": 4}' \
http://localhost:12000/cursos
{"message":"OK","data":{"nome":"Técnico em Mecânica","duracao":4,"createdAt":"2026-04-14T00:21:14.490+00
:00","updatedAt":"2026-04-14T00:21:14.490+00:00","id":3}}%
```

[AUTORIZADO]

EXIBIR CURSOS

(*no terminal de comando)*

```
Shell
curl -X GET http://localhost:12000/cursos/1 \
-H "Content-Type: application/json" \
```

```
-H                          "Authorization:                          Bearer
oat_MQ.SUhrWkVmSGI3dHF4djdnanJOVEEwNDQ4bmZLZEt5MEdVeGJ6bTlaYTI5OTE3Nj
MzMDc"
```

```
% curl -X GET http://localhost:12000/cursos/1 \
-H "Content-Type: application/json" \
-H "Authorization: Bearer oat_MQ.SUhrWkVmSGI3dHF4djdnanJOVEEwNDQ4bmZLZEt5MEdVeGJ6bTlaYTI5OTE3NjMzMDc"
{"message":"OK","data":{"id":1,"nome":"Técnico em Informática","duracao":4,"createdAt":"2026-04-13T19:24
:22.000+00:00","updatedAt":"2026-04-13T19:24:22.000+00:00"}}%
```

[AUTORIZADO]

ALTERAR CURSOS
(*no terminal de comando)*

```
Shell
curl -X PUT http://localhost:12000/cursos/1 \
-H "Content-Type: application/json" \
-H                          "Authorization:                          Bearer
oat_MQ.SUhrWkVmSGI3dHF4djdnanJOVEEwNDQ4bmZLZEt5MEdVeGJ6bTlaYTI5OTE3Nj
MzMDc" \
-d '{"nome": "Técnico em Meio Ambiente", "duracao": 4}'
```

```
% curl -X PUT http://localhost:12000/cursos/1 \
-H "Content-Type: application/json" \
-H "Authorization: Bearer oat_MQ.SUhrWkVmSGI3dHF4djdnanJOVEEwNDQ4bmZLZEt5MEdVeGJ6bTlaYTI5OTE3NjMzMDc" \
-d '{"nome": "Técnico em Meio Ambiente", "duracao": 4}'

{"message":"Você não tem permissão para modificar esse recurso!"}%
```

[NÃO AUTORIZADO]

REMOVER CURSOS
(*no terminal de comando)*

```
Shell
curl -X DELETE http://localhost:12000/cursos/2 \
-H "Content-Type: application/json" \
-H                          "Authorization:                          Bearer
oat_MQ.SUhrWkVmSGI3dHF4djdnanJOVEEwNDQ4bmZLZEt5MEdVeGJ6bTlaYTI5OTE3Nj
MzMDc"
```

```
% curl -X DELETE http://localhost:12000/cursos/2 \
-H "Content-Type: application/json" \
-H "Authorization: Bearer oat_MQ.SUhrWkVmSGI3dHF4djdnanJOVEEwNDQ4bmZLZEt5MEdVeGJ6bTlaYTI5OTE3NjMzMDc"
{"message":"Você não tem permissão para remover esse recurso!"}%
```

[NÃO AUTORIZADO]

EFETUAR LOGOUT
(*logout - no terminal de comando)*

```
Shell
curl -X POST http://localhost:12000/auth/logout \
-H "Authorization: Bearer oat_MQ.SUhrWkVmSGI3dHF4djdnanJOVEEwNDQ4bmZLZEt5MEdVeGJ6bTlaYTI5OTE3NjMzMDc"
```

```
 % curl -X POST http://localhost:12000/auth/logout \
-H "Authorization: Bearer oat_MQ.SUhrWkVmSGI3dHF4djdnanJOVEEwNDQ4bmZLZEt5MEdVeGJ6bTlaYTI5OTE3NjMzMDc"
{"message":"Logout Realizado com Sucesso"}%
```

[AUTORIZADO]

EFETUAR LOGIN (Coordenador)
(*login - no terminal de comando)*

```
Shell
curl -X POST http://localhost:12000/auth/login \
-H "Content-Type: application/json" \
-d '{"email": "gil@gmail.com", "password": "1234@5678"}'
```

```
 % curl -X POST http://localhost:12000/auth/login \
-H "Content-Type: application/json" \
-d '{"email": "gil@gmail.com", "password": "1234@5678"}'

{"message":"Login realizado com sucesso","resp":{"user":{"id":1,"fullName":"Gil Eduardo de Andrade","email":"gil@gmail.com","createdAt":"2026-04-13T19:24:22.000+00:00","updatedAt":"2026-04-13T19:24:22.000+00:00"},"token":{"type":"bearer","value":"oat_Mg.a0VKREl2alVXUG81ZS1GSGNkcU1vX0hwVFktTGg4TFJCYVd5R25YTTE1Nzk2MTM5NzA","expiresAt":"2026-05-14T12:23:41.716Z"}}}%
```

Token: oat_Mg.a0VKREl2alVXUG81ZS1GSGNkcU1vX0hwVFktTGg4TFJCYVd5R25YTTE1Nzk2MTM5NzA

LISTAR CURSOS
(*no terminal de comando)*

```
Shell
curl -X GET http://localhost:12000/cursos \
-H "Authorization: Bearer oat_Mg.a0VKREl2alVXUG81ZS1GSGNkcU1vX0hwVFktTGg4TFJCYVd5R25YTTE1Nzk2MTM5NzA"
```

```
 % curl -X GET http://localhost:12000/cursos \
-H "Authorization: Bearer oat_Mg.a0VKREl2alVXUG81ZS1GSGNkcU1vX0hwVFktTGg4TFJCYVd5R25YTTE1Nzk2MTM5NzA"
{"message":"OK","data":[{"id":3,"nome":"Técnico em Mecânica","duracao":4,"createdAt":"2026-04-14T00:21:1
4.000+00:00","updatedAt":"2026-04-14T00:21:14.000+00:00"},{"id":2,"nome":"Técnólogo em Análise e Desenvo
lvimento","duracao":3,"createdAt":"2026-04-13T19:24:22.000+00:00","updatedAt":"2026-04-13T19:24:22.000+0
0:00"},{"id":1,"nome":"Técnico em Informática","duracao":4,"createdAt":"2026-04-13T19:24:22.000+00:00","
updatedAt":"2026-04-13T19:24:22.000+00:00"}]}%
```

[AUTORIZADO]

CRIAR CURSOS
(*no terminal de comando)*

Shell

```
curl -X POST \
-H "Content-Type: application/json" \
-H "Authorization: Bearer oat_Mg.a0VKREl2alVXUG81ZS1GSGNkcU1vX0hwVFktTGg4TFJCYVd5R25YTTE1Nzk2MTM5NzA" \
-d '{"nome": "Técnico em Mecânica", "duracao": 4}' \
http://localhost:12000/cursos
```

```
 % curl -X POST \
-H "Content-Type: application/json" \
-H "Authorization: Bearer oat_Mg.a0VKREl2alVXUG81ZS1GSGNkcU1vX0hwVFktTGg4TFJCYVd5R25YTTE1Nzk2MTM5NzA" \
-d '{"nome": "Técnico em Mecânica", "duracao": 4}' \
http://localhost:12000/cursos
{"message":"OK","data":{"nome":"Técnico em Mecânica","duracao":4,"createdAt":"2026-04-14T12:28:08.912+00
:00","updatedAt":"2026-04-14T12:28:08.912+00:00","id":4}}%
```

[AUTORIZADO]

EXIBIR CURSOS
(*no terminal de comando)*

Shell

```
curl -X GET http://localhost:12000/cursos/1 \
-H "Content-Type: application/json" \
-H "Authorization: Bearer oat_Mg.a0VKREl2alVXUG81ZS1GSGNkcU1vX0hwVFktTGg4TFJCYVd5R25YTTE1Nzk2MTM5NzA"
```

```
 % curl -X GET http://localhost:12000/cursos/1 \
-H "Content-Type: application/json" \
-H "Authorization: Bearer oat_Mg.a0VKREl2alVXUG81ZS1GSGNkcU1vX0hwVFktTGg4TFJCYVd5R25YTTE1Nzk2MTM5NzA"
{"message":"OK","data":{"id":1,"nome":"Técnico em Informática","duracao":4,"createdAt":"2026-04-13T19:24
:22.000+00:00","updatedAt":"2026-04-13T19:24:22.000+00:00"}}%
```

[AUTORIZADO]

ALTERAR CURSOS
(*no terminal de comando)*

```shell
curl -X PUT http://localhost:12000/cursos/1 \
-H "Content-Type: application/json" \
-H "Authorization: Bearer oat_Mg.a0VKREl2alVXUG81ZS1GSGNkcU1vX0hwVFktTGg4TFJCYVd5R25YTTE1Nzk2MTM5NzA" \
-d '{"nome": "Técnico em Meio Ambiente", "duracao": 4}'
```

```
 % curl -X PUT http://localhost:12000/cursos/1 \
-H "Content-Type: application/json" \
-H "Authorization: Bearer oat_Mg.a0VKREl2alVXUG81ZS1GSGNkcU1vX0hwVFktTGg4TFJCYVd5R25YTTE1Nzk2MTM5NzA" \
-d '{"nome": "Técnico em Meio Ambiente", "duracao": 4}'
{"message":"OK","data":{"id":1,"nome":"Técnico em Meio Ambiente","duracao":4,"createdAt":"2026-04-13T19:24:22.000+00:00","updatedAt":"2026-04-14T12:29:58.712+00:00"}}%
```

[AUTORIZADO]

REMOVER CURSOS
(*no terminal de comando*)

```shell
curl -X DELETE http://localhost:12000/cursos/3 \
-H "Content-Type: application/json" \
-H "Authorization: Bearer oat_Mg.a0VKREl2alVXUG81ZS1GSGNkcU1vX0hwVFktTGg4TFJCYVd5R25YTTE1Nzk2MTM5NzA"
```

```
 % curl -X DELETE http://localhost:12000/cursos/3 \
-H "Content-Type: application/json" \
-H "Authorization: Bearer oat_Mg.a0VKREl2alVXUG81ZS1GSGNkcU1vX0hwVFktTGg4TFJCYVd5R25YTTE1Nzk2MTM5NzA"
{"message":"OK","data":{}}%
```

[AUTORIZADO]

REGISTRAR USUÁRIO (Agora com papel)

Atualizando Validator do Usuário

Arquivo: "/app/validators/auth.ts"

```typescript
...
vine.object({
    fullName: vine.string().trim().minLength(2).maxLength(100).optional(),
```

```
    email: vine
      .string()
      .email()
      .normalizeEmail()
      .unique(async (db, value) => {
        const user = await db.from('users').where('email', value).first()
        return !user
      }),
    password: vine.string().minLength(8).maxLength(32),
    roleId: vine.number().exists(async (db, value, field) => {
      const role = await db.from('roles').where('id', value).first()
      return !!role
    }),
  })
...
```

Atualizando Repositório do Usuário (Retornar campo “role”)

Arquivo: “/app/validators/auth.ts”

TypeScript

```
...
async createUser(payload: PayloadUser): Promise<any> {
    const user = await User.create(payload)
    // user: usuário logado;
    // *: coringa, token pode acessar todas as rotas protegidas;
    const token = await User.accessTokens.create(user, ['*'], {
      name: 'Registration Token',
      expiresIn: '30 days', // define a validade do Token;
    })

    return {
      message: 'Usuário registrado com sucesso',
      user: {
        id: user.id,
        fullName: user.fullName,
        roleId: user.roleId,
        email: user.email,
        createdAt: user.createdAt,
```

```
      },
      token: {
        type: 'bearer', // padrão (OAuth 2.0) tipo de token que cliente
        value: token.value!.release(), // string limpa e utilizável pelo cliente
        expiresAt: token.expiresAt, // data e hora da expiração do token;
      },
    }
  }
...
```

***OBS.:** a mesma dinâmica deve ser efetuada para outros métodos da classe, no exemplo foi feito apenas o método "create()".*

Executando Comando no Terminal

Shell

```
curl -X POST http://localhost:12000/auth/register \
-H "Content-Type: application/json" \
-H "Authorization: Bearer oat_Mg.a0VKREl2alVXUG81ZS1GSGNkcU1vX0hwVFktTGg4TFJCYVd5R25YTTE1Nzk2MTM5NzA" \
-d '{"fullName": "João Silva", "email": "joao@exemplo.com", "password": "senha123456", "roleId": "1"}'
```

```
% curl -X POST http://localhost:12000/auth/register \
-H "Content-Type: application/json" \
-H "Authorization: Bearer oat_Mg.a0VKREl2alVXUG81ZS1GSGNkcU1vX0hwVFktTGg4TFJCYVd5R25YTTE1Nzk2MTM5NzA" \
-d '{"fullName": "João Silva", "email": "joao@exemplo.com", "password": "senha123456", "roleId": "1"}'
{"message":"Usuário registrado com sucesso","resp":{"message":"Usuário registrado com sucesso","user":{"
id":8,"fullName":"João Silva","roleId":1,"email":"joao@exemplo.com","createdAt":"2026-04-14T12:51:06.051
+00:00"},"token":{"type":"bearer","value":"oat_OA.Ti1PSVZRRy03Y24zTHgtVGhjdkJZNW8yeUJ6UExXdWtsdi10TGs1Rz
QxOTIzNDgwNDk","expiresAt":"2026-05-14T12:51:06.061Z"}}}%
```

[AUTORIZADO]