	<p align="center">Centro Federal de Educação Tecnológica de Minas Gerais Campus VIII – Varginha Curso Técnico em Informática</p>	
<p><i>Disciplina</i> Lab. Aplicações Web I</p>	<p align="center">Criação do Backend para autenticação do usuário</p>	<p><i>Professor</i> Lázaro Eduardo da Silva</p>

Na aula de hoje vamos criar um projeto AdonisJS com Typescript e implementar o backend da nossa aplicação. Nesta primeira parte da implementação do backend, vamos implementar a criação e autenticação dos usuários, guardando os dados no banco de dados.

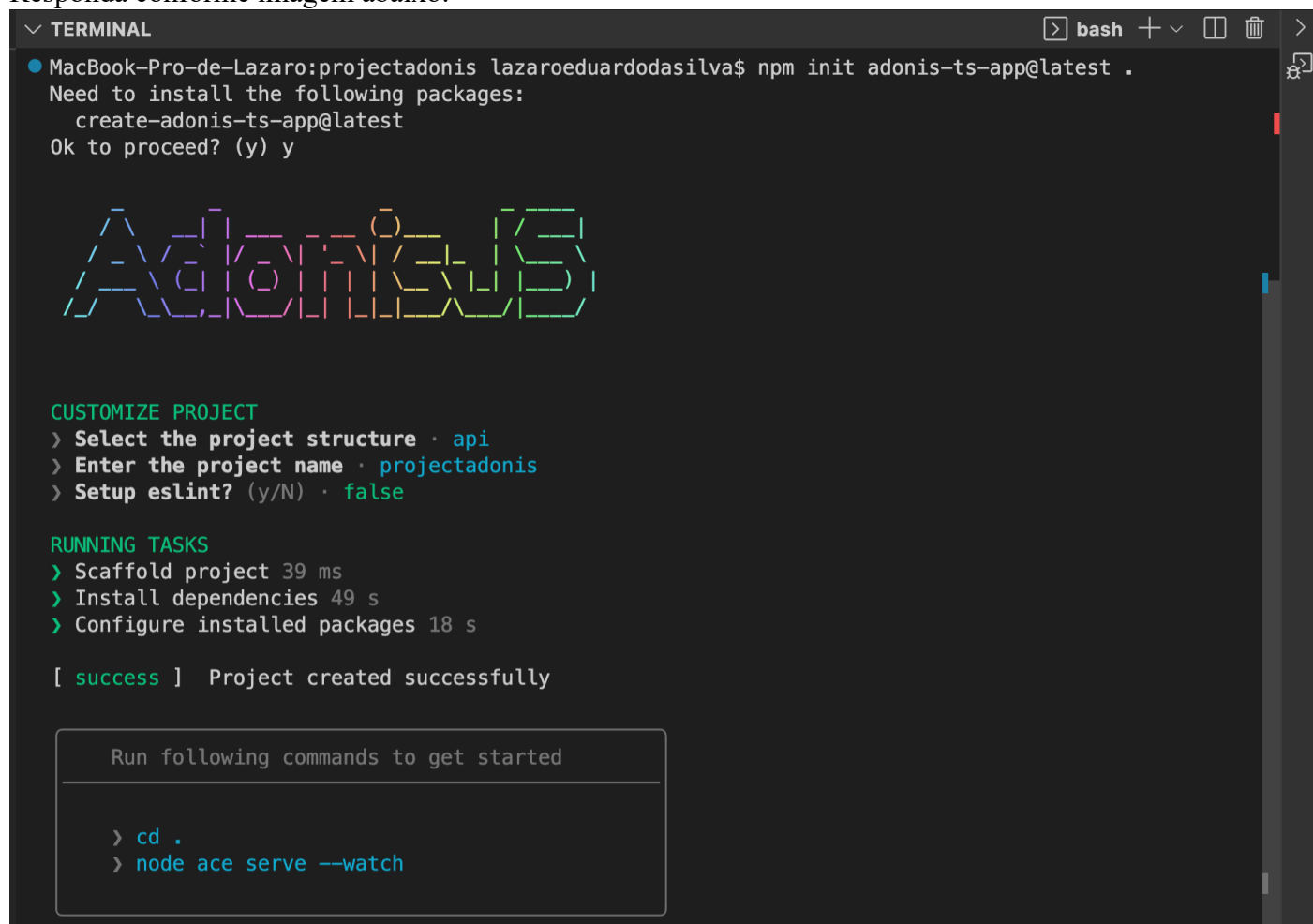
Para começar o projeto, inicialmente, crie uma nova pasta, abra o Visual Studio Code, abra a pasta criado no Visual Studio Code, abra o terminal e verifique a versão do nodejs que está instalado no computador que você está utilizando. Para isso, execute o comando abaixo:

```
node -v
```

A versão deve ser 14 ou maior. Caso seja, execute o comando abaixo para que o projeto AdonisJS seja criado:

```
npm init adonis-ts-app@latest .
```

Este comando irá iniciar a criação do projeto AdonisJS que terá algumas perguntas para serem respondidas. Responda conforme imagem abaixo:



```

TERMINAL
MacBook-Pro-de-Lazaro:projectadonis lazaroeduardodasilva$ npm init adonis-ts-app@latest .
Need to install the following packages:
  create-adonis-ts-app@latest
Ok to proceed? (y) y

  AdonisJS

CUSTOMIZE PROJECT
> Select the project structure · api
> Enter the project name · projectadonis
> Setup eslint? (y/N) · false

RUNNING TASKS
> Scaffold project 39 ms
> Install dependencies 49 s
> Configure installed packages 18 s

[ success ] Project created successfully

Run following commands to get started

> cd .
> node ace serve --watch


```

Ao final da instalação é apresentado o comando que inicia a execução da API

```
node ace serve --watch
```

Ao executá-lo, deve aparecer a tela abaixo que indica que a API pode ser acessada no endereço:

```
http://127.0.0.1:3333
```

	<p align="center">Centro Federal de Educação Tecnológica de Minas Gerais Campus VIII – Varginha Curso Técnico em Informática</p>		
<p><i>Disciplina</i> Lab. Aplicações Web I</p>	<p align="center">Criação do Backend para autenticação do usuário</p>	<p align="right"><i>Professor</i> Lázaro Eduardo da Silva</p>	

```

MacBook-Pro-de-Lazaro:projectadonis lazaroeduardodasilva$ node ace serve --watch
[ info ] building project...
[ info ] starting http server...
[19:37:03.839] INFO (projectadonis/2894): started server on 0.0.0.0:3333
[ info ] watching file system for changes

Server address: http://127.0.0.1:3333
Watching filesystem for changes: YES

```

O endereço do site da ferramenta que explica esta instalação é:

<https://docs.adonisjs.com/guides/installation>

Utilizaremos o banco de dados mysql para armazenar os dados do usuário nesta aplicação, para tal, precisamos instalar os pacotes de Database conforme documentação que está no endereço:

<https://docs.adonisjs.com/guides/database/introduction>

Para tal, abra um novo terminal e execute o comando abaixo para instalar o ORM Lucid:

```
npm install @adonisjs/lucid
```

Posteriormente, para configurar o acesso ao banco de dados deve-se executar o comando:

```
node ace configure @adonisjs/lucid
```

Ao executar o comando, algumas perguntas serão realizadas, para mudar as opções utilize as setas do teclado, para marcar alguma opção pressione espaço, para confirmar pressione enter. Responda conforme imagem abaixo:

```


MacBook-Pro-de-Lazaro:projectadonis lazaroeduardodasilva$ node ace configure @adonisjs/lucid
> Select the database driver you want to use · mysql, pg
CREATE: config/database.ts
UPDATE: .env,.env.example
[ wait ] Installing: luxon, mysql, pg .
CREATE: database/factories/index.ts
UPDATE: tsconfig.json { types += "@adonisjs/lucid" }
UPDATE: .adonisrc.json { commands += "@adonisjs/lucid/build/commands" }
UPDATE: .adonisrc.json { providers += "@adonisjs/lucid" }
[ info ] The package wants to display readable instructions for the setup
> Select where to display instructions · browser
CREATE: ace-manifest.json file

```

Selecionamos mysql e postgres porque vamos utilizar mysql no computador local e postgres quando formos publicar o projeto online.

As instruções para configurar o acesso ao banco de dados devem ser abertas em uma janela do navegador ou no terminal.

O arquivo que precisa ser configurado é o .env que está na raiz do projeto. Para executarmos este projeto no computador do laboratório, será necessário criar um banco de dados e configurar as variáveis:

	<p align="center">Centro Federal de Educação Tecnológica de Minas Gerais Campus VIII – Varginha Curso Técnico em Informática</p>	
<p><i>Disciplina</i> Lab. Aplicações Web I</p>	<p align="center">Criação do Backend para autenticação do usuário</p>	<p><i>Professor</i> Lázaro Eduardo da Silva</p>

DB_CONNECTION=
 MYSQL_HOST=
 MYSQL_PORT=
 MYSQL_USER=
 MYSQL_PASSWORD=
 MYSQL_DB_NAME=

Para implementarmos a criação e autenticação do usuário, vamos seguir a documentação que está no endereço:
<https://docs.adonisjs.com/guides/auth/introduction>

Para tal, no terminal execute o comando abaixo para instalar o pacote de autenticação:

```
npm install @adonisjs/auth
```

Posteriormente, para configurar a autenticação deve-se executar o comando:

```
node ace configure @adonisjs/auth
```

Ao executar o comando, algumas perguntas serão realizadas, para mudar as opções utilize as setas do teclado, para marcar alguma opção pressione espaço, para confirmar pressione enter. Responda conforme imagem abaixo:


```

● MacBook-Pro-de-Lazaro:projectadonis lazaroeuardodasilva$ node ace configure @adonisjs/auth
> Select provider for finding users · lucid
> Select which guard you need for authentication (select using space) · api
> Enter model name to be used for authentication · User
> Create migration for the users table? (y/N) · true
> Select the provider for storing API tokens · database
> Create migration for the api_tokens table? (y/N) · true
CREATE: app/Models/User.ts
CREATE: database/migrations/1660862735924_users.ts
CREATE: database/migrations/1660862735926_api_tokens.ts
CREATE: contracts/auth.ts
CREATE: config/auth.ts
CREATE: app/Middleware/Auth.ts
CREATE: app/Middleware/SilentAuth.ts
UPDATE: tsconfig.json { types += "@adonisjs/auth" }
UPDATE: .adonisrc.json { providers += "@adonisjs/auth" }
CREATE: ace-manifest.json file

```

A configuração da autenticação, cria duas migrations, que são classes responsáveis por criar as tabelas no nosso banco de dados. Antes de criar as tabelas, vamos fazer pequenas alterações na migration de usuários.

Acrescente o nome do usuário e coloque a restrição de único no e-mail, caso não exista, conforme imagem abaixo:

	<p align="center">Centro Federal de Educação Tecnológica de Minas Gerais Campus VIII – Varginha Curso Técnico em Informática</p>	
<p><i>Disciplina</i> Lab. Aplicações Web I</p>	<p align="center">Criação do Backend para autenticação do usuário</p>	<p><i>Professor</i> Lázaro Eduardo da Silva</p>

```

TS 1660862735924_users.ts ×
database > migrations > TS 1660862735924_users.ts > ...
1  import BaseSchema from '@ioc:Adonis/Lucid/Schema'
2
3  export default class UsersSchema extends BaseSchema {
4    protected tableName = 'users'
5
6    public async up() {
7      this.schema.createTable(this.tableName, (table) => {
8        table.increments('id').primary()
9        table.string('name', 80).notNullable()
10       table.string('email', 255).unique().notNullable()
11       table.string('password', 180).notNullable()
12       table.string('remember_me_token').nullable()
13
14       /**
15        * Uses timestampz for PostgreSQL and DATETIME2 for MSSQL
16        */
17       table.timestamp('created_at', { useTz: true }).notNullable()
18       table.timestamp('updated_at', { useTz: true }).notNullable()
19     })
20   }
21
22   public async down() {
23     this.schema.dropTable(this.tableName)
24   }
25 }

```

Para criar as tabelas definidas nas migrations execute o comando abaixo no terminal:

`node ace migration:run`

Após executar este comando, acesse o phpMyAdmin e verifique se as tabelas foram criadas no banco de dados configurado no arquivo .env. Caso tenha criado, siga para o próximo passo, caso não, chame o professor.

Outra configuração necessária para que a modificação que fizemos funcione corretamente é no arquivo User.ts na pasta app/Models acrescentando a informação do campo name:

```
app > Models > TS User.ts > ...
1  import { DateTime } from 'luxon'
2  import Hash from '@ioc:Adonis/Core/Hash'
3  import { column, beforeSave, BaseModel } from '@ioc:Adonis/Lucid/Orm'
4
5  export default class User extends BaseModel {
6    @column({ isPrimary: true })
7    public id: number
8
9    @column()
10   public name: string
11
12   @column()
13   public email: string
14
15   @column({ serializeAs: null })
16   public password: string
17
18   @column()
19   public rememberMeToken?: string
20
21   @column.dateTime({ autoCreate: true })
22   public createdAt: DateTime
23
24   @column.dateTime({ autoCreate: true, autoUpdate: true })
25   public updatedAt: DateTime
26
27   @beforeSave()
28   public static async hashPassword(User: User) {
29     if (User.$dirty.password) {
30       User.password = await Hash.make(User.password)
31     }
32   }
33 }
```

Observe no arquivo acima que é realizado um hash na senha para que ela não seja guardada no banco de dados de forma aberta.

Você deve ter percebido que as configurações são realizadas através do comando node ace. Ele é uma CLI (Interface de Linha de Comando) que é responsável por criar e configurar os recursos do framework. Vamos utilizá-lo para criar alguns arquivos importantes para a implementação da autenticação. O primeiro que vamos criar será utilizado para validação dos campos para registro dos usuários, execute o comando abaixo:

```
node ace make:validator RegisterUser
```

Este arquivo possui uma classe na qual, devemos configurar a regra de validação e as mensagens de erro, caso alguma regra não foi satisfeita.

Para tal, deve-se importar a variável rules no início do arquivo, preencher o schema e as custom messages conforme imagem abaixo:



Centro Federal de Educação Tecnológica de Minas Gerais
Campus VIII – Varginha
Curso Técnico em Informática

Disciplina
Lab. Aplicações Web I


**Criação do Backend para
autenticação do usuário**

Professor
Lázaro Eduardo da Silva

RegisterUserValidator.ts X

app > Validators > RegisterUserValidator.ts > ...

```
1  import { schema, rules, CustomMessages } from '@ioc:Adonis/Core/Validator'
2  import type { HttpContextContract } from '@ioc:Adonis/Core/HttpContext'
3
4  export default class RegisterUserValidator {
5    constructor(protected ctx: HttpContextContract) { }
6
7    /*
8     * Define schema to validate the "shape", "type", "formatting" and "integrity" of data.
9     *
10    * For example:
11    * 1. The username must be of data type string. But then also, it should
12    *    not contain special characters or numbers.
13    *    ```
14    *    schema.string({}, [ rules.alpha() ])
15    *    ```
16    *
17    * 2. The email must be of data type string, formatted as a valid
18    *    email. But also, not used by any other user.
19    *    ```
20    *    schema.string({}, [
21    *      rules.email(),
22    *      rules.unique({ table: 'users', column: 'email' }),
23    *    ])
24    *    ```
25    */
26    public schema = schema.create({
27      name: schema.string({}, [
28        rules.required()
29      ]),
30      email: schema.string({}, [
31        rules.required(),
32        rules.email(),
33        rules.unique({ table: 'users', column: 'email' })
34      ]),
35      password: schema.string({}, [
36        rules.required(),
37        rules.minLength(4)
38      ])
39    })
40
41    /**
42     * Custom messages for validation failures. You can make use of dot notation `(.)`
43     * for targeting nested fields and array expressions `(*)` for targeting all
44     * children of an array. For example:
45     *
46     * {
47     *   'profile.username.required': 'Username is required',
48     *   'scores.*.number': 'Define scores as valid numbers'
49     * }
50     *
51     */
52    public messages: CustomMessages = {
53      required: "0 {{field}} é obrigatório para se registrar!!!",
54      'email.unique': "E-mail já cadastrado!!!",
55      'minLength': "Tamanho de senha inválida"
56    }
57  }
```

	<p align="center">Centro Federal de Educação Tecnológica de Minas Gerais Campus VIII – Varginha Curso Técnico em Informática</p>	
<p><i>Disciplina</i> Lab. Aplicações Web I</p>	<p align="center">Criação do Backend para autenticação do usuário</p>	<p><i>Professor</i> Lázaro Eduardo da Silva</p>

Vamos utilizar o ace desta vez para criar o controller que utilizaremos para realizar o registro e a autenticação do usuário. Para isso, deve-se executar o comando abaixo no terminal:

```
node ace make:controller Auth
```


Ao executar este comando, deve ser criado o arquivo na pasta app/Controllers/Http/AuthController.ts. O conteúdo desse arquivo deve ser editado conforme imagem abaixo:

```
AuthController.ts ×
app > Controllers > Http > AuthController.ts > ...
1  import type { HttpContextContract } from '@ioc:Adonis/Core/HttpContext'
2  import User from 'App/Models/User'
3  import RegisterUserValidator from 'App/Validators/RegisterUserValidator'
4
5  export default class AuthController {
6
7      public async register({ request }: HttpContextContract) {
8          const data = await request.validate(RegisterUserValidator)
9          const user = await User.create(data)
10         return user
11     }
12
13     public async login({ request, auth, response }: HttpContextContract) {
14         try {
15             const { email, password } = request.all()
16             const token = await auth.use('api').attempt(email, password, {
17                 expiresIn: '1day'
18             })
19             const user = await User.findByOrFail("email", email)
20             return { token, user }
21         } catch (error) {
22             response.status(401).send("Login ou senha incorretos!!!")
23         }
24     }
25
26 }
```

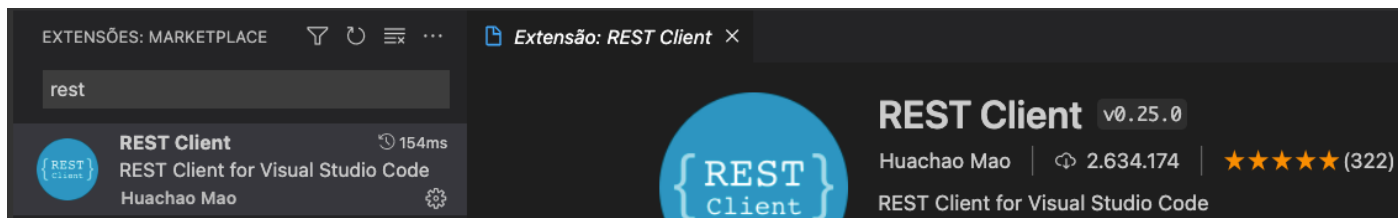
A última configuração que precisa ser realizada é no arquivo que fica na pasta start/routes.ts. Este arquivo

```
routes.ts ×
start > routes.ts > ...
21  import Route from '@ioc:Adonis/Core/Route'
22
23  Route.get('/', async () => {
24      return { hello: 'world' }
25  })
26
27  Route.post("/register", "AuthController.register")
28  Route.post("/login", "AuthController.login")
```

contém as rotas da aplicação. Estas rotas indicam, de acordo com a URL solicitada, qual função de qual controler deve ser executada. Este arquivo já possui um conteúdo, no qual, as duas últimas linhas devem ser acrescentadas:

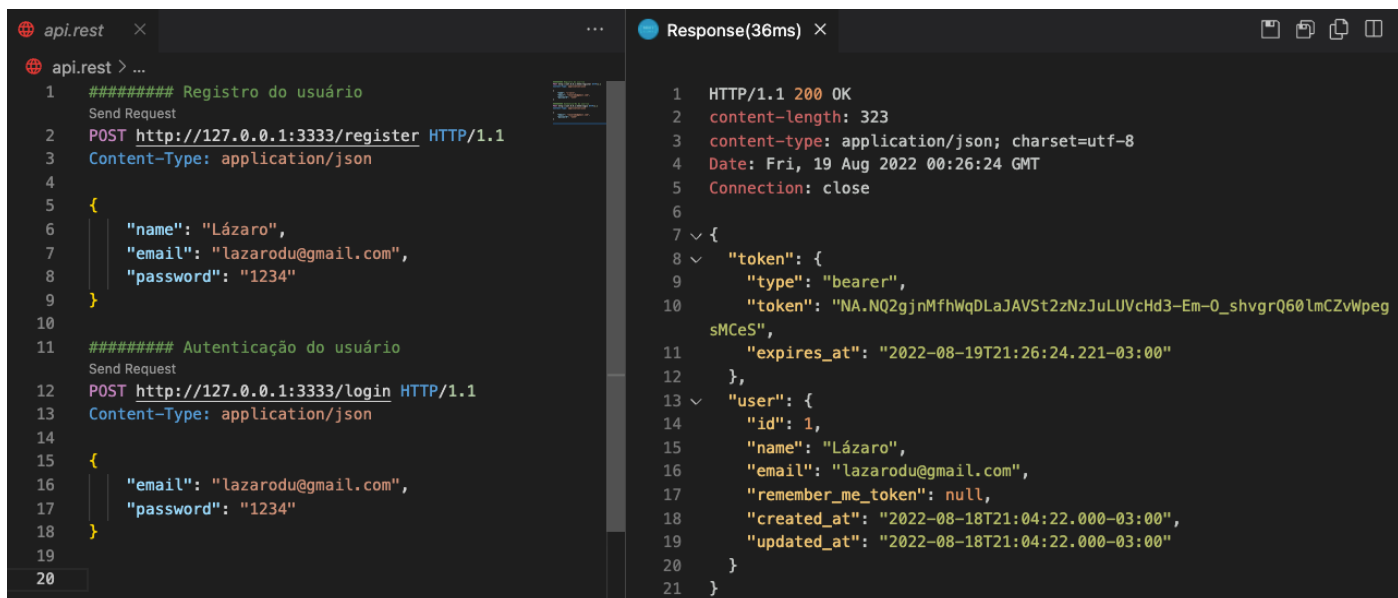
	<p align="center">Centro Federal de Educação Tecnológica de Minas Gerais Campus VIII – Varginha Curso Técnico em Informática</p>	
<p><i>Disciplina</i> Lab. Aplicações Web I</p>	<p align="center">Criação do Backend para autenticação do usuário</p>	<p><i>Professor</i> Lázaro Eduardo da Silva</p>

Estamos com as implementações realizadas. O próximo passo é testar nossa API. Para testar a API você deve instalar a extensão REST Client.



Esta extensão permite configurarmos um arquivo um arquivo na raiz do nosso projeto para fazer as requisições para a nossa API implementada.

Com o servidor rodando, crie um arquivo chamado api.rest na raiz do seu projeto e escreva o seu conteúdo conforme imagem abaixo, quadro da esquerda. O Send Request não precisa ser digitado, ele deve aparecer, assim que você finalizar a escrita dos comandos. Seja atento com o espaçamento e as quebras de linha, pois eles interferem na execução do teste.



Para executar uma requisição, clique no Send Request. A primeira requisição, acessa a rota register da nossa aplicação e envia os dados name, email e password. Ela deve criar um usuário com essas credenciais no banco de dados.

Após executá-la, acesse sua tabela user utilizando o phpMyAdmin para verificar se o conteúdo foi inserido no banco de dados.

Clique no Send Request da segunda requisição para testar a autenticação. O token retornado deve ser utilizado em todas as requisições que precisam estar autenticadas.

Estudaremos requisições autenticadas em uma atividade próxima.

Bom trabalho!!!