Pré-Requisitos

```
node
npm
docker
```

Instalação do framework Adonis.js

```
sudo npm i -g @adonisjs/cli
```

> após a instalação criar projeto com adonijs

Criando projeto

```
adonis new my-api --api-only
```

> O my-api é o nome que dei ao meu projeto,o parâmetro --api-only no fim significa que não queremos a parte de views do Adonis já que vamos servir os dados do nosso app via REST para o ReactJS e o React Native.
>
> após o termino da criação do projeto siga os camandos

```
$ cd my-api
$ adonis serve --dev
```

criando docker para mariadb

```
$ docker pull mariadb
```

> $ docker run -p < PORTA-DOCKER:PORTAMAQUINA > --name some-mariadb -e MYSQL_ROOT_PASSWORD=my-secret-pw -d mariadb:tag

> $ docker run --name some-mariadb -e MYSQL_ROOT_PASSWORD=my-secret-pw -d mariadb:tag
> ... onde some-mariadb é o nome que você deseja atribuir ao seu contêiner, my-secret-pw é a senha a ser definida para o usuário root do MySQL e tag é a tag que especifica a versão do MySQL desejada. Veja a lista acima para tags relevantes.

```
docker run -p 3306:3306 --name mariadb -e MYSQL_ROOT_PASSWORD=secret -d mariadb:latest
```

Instalar módulo do mariadb no adonisjs

```
$ npm install mysql
```

Rodando Migrações

```
adonis migration:run
```

Criando Controller

adonis make:controller Auth

escolha a opção

> For HTTP requests

> edite o arquivo app/Controllers/Http/AuthController.js da seguinte forma

```
const User = use('App/Models/User')


class AuthController {
•    async register({ *request*  }){
•        const data = *request*.only(['username', 'email', 'password'])
•
•        const user = await User.create(data)


•        return user
•    }
```

```
    async authenticate({ *request*, *auth* }){
        const { email, password } = *request*.all();

        const token = await *auth*.attempt(email, password);

        return token;

    }
}

module.exports = AuthController
```

adonis make:controller App

escolha a opção

> For HTTP requests

edite o arquivo app/Controllers/Http/AppController.js da seguinte forma

```
class AppController {

    index(){
        return 'hello World';
    }
}

module.exports = AppController
```

edite o arquivo start/routes.js da seguinte forma

Route.post('/register', "AuthController.register")

Route.post('/authenticate', "AuthController.authenticate")

Route.get('/app', "AppController.index").middleware(["auth"]);

Testando as Rotas

Instale o Insominia

> https://insomnia.rest/download/

Na Rota

```
POST \
  --url http://127.0.0.1:3333/register \
  --json '{
    "username":"nome",
    "email": "email@email.com",
    "password":"123456"
}'
```

Resposta

```
{
  "username": "nome",
  "email": "email@email.com",
  "password": "$2a$10$WgCkdbaXgtKoB88W38xoq.LNoxK7EMIOE0NmFMPYN6k8jzAWmA7yq",
  "created_at": "2020-09-21 22:49:16",
  "updated_at": "2020-09-21 22:49:16",
  "id": 1
}
```

Na Rota

```
curl --request POST \
  --url http://127.0.0.1:3333/authenticate \
  --json '{
    "email": "email@email.com",
    "password":"123456"
}'
```

Resposta

```
{
  "type": "bearer",
  "token":
"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1aWQiOjEsImlhdCI6MTYwMDc0MTQ5NH0.sLWVevA
LtIXrmtDzXyl-wnr9Dk45wAceFR4HwidevaU",
  "refreshToken": null
}
```

Na Rota

```
curl --request GET \
  --url http://127.0.0.1:3333/app \
  --header 'content-type: application/json' \
  --data '{
    "email": "email@email.com",
    "password":"123456"
}'
```

caso não esteja autenticado

InvalidJwtToken

E_INVALID_JWT_TOKEN: jwt must be provided

Show all frames

REQUEST DETAILS

URI	**/app**
REQUEST METHOD	GET
HTTP VERSION	1.1
CONNECTION	

HEADERS

HOST	**127.0.0.1:3333**
USER-AGENT	insomnia/2020.4.0
CONTENT-TYPE	application/json
ACCEPT	/
CONTENT-LENGTH	61