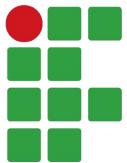


INSTITUTO FEDERAL
Paraná
Campus Paranaúá

Framework AdonisJs

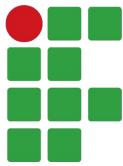
Desenvolvimento Web

Prof. Diego Stiehl



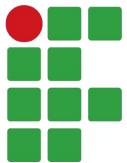
Express

- O Framework Express nos permite construir qualquer aplicação web
- Ele é pouco opinativo
 - Foco na simplicidade
 - Desenvolvedor escolhe suas ferramentas
- Pode gerar mais trabalho quando a aplicação começa a crescer
 - Código repetido, escrever bibliotecas, tratamento manual de erros, ...



AdonisJs

- AdonisJS é um framework Node.js para web baseado no modelo MVC
 - Arquitetura mais rígida e definida
 - Baseada em boas práticas conhecidas
 - Inspiração: Framework Laravel (PHP)
 - Convenção sobre configuração
 - Extensível
 - Permite focar na regra de negócio da aplicação
 - Possui um framework de persistência embutido
 - Projeto já nasce com uma estrutura funcional



Instalação

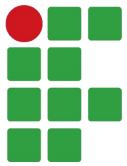
- Via NPM

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

- Vai fornecer o executável adonis
 - Ferramenta de linha de comando

- Testar

```
adonis --version
```



Criando Projeto

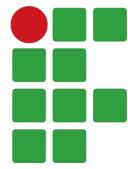
- Vamos refazer o exemplo da lista de tarefas
- No terminal:

adonis new info-todo-list

```
↳ adonis new info-todo-list
[1/6] 🔒 Requirements matched [node & npm]
[2/6] ✎ Ensuring project directory is clean [info-todo-list]
[3/6] 📁 Cloned [adonisjs/adonis-fullstack-app]
[4/6] 📦 Dependencies installed
[5/6] 📜 Environment variables copied [.env]
[6/6] 🔑 Key generated [adonis key:generate]

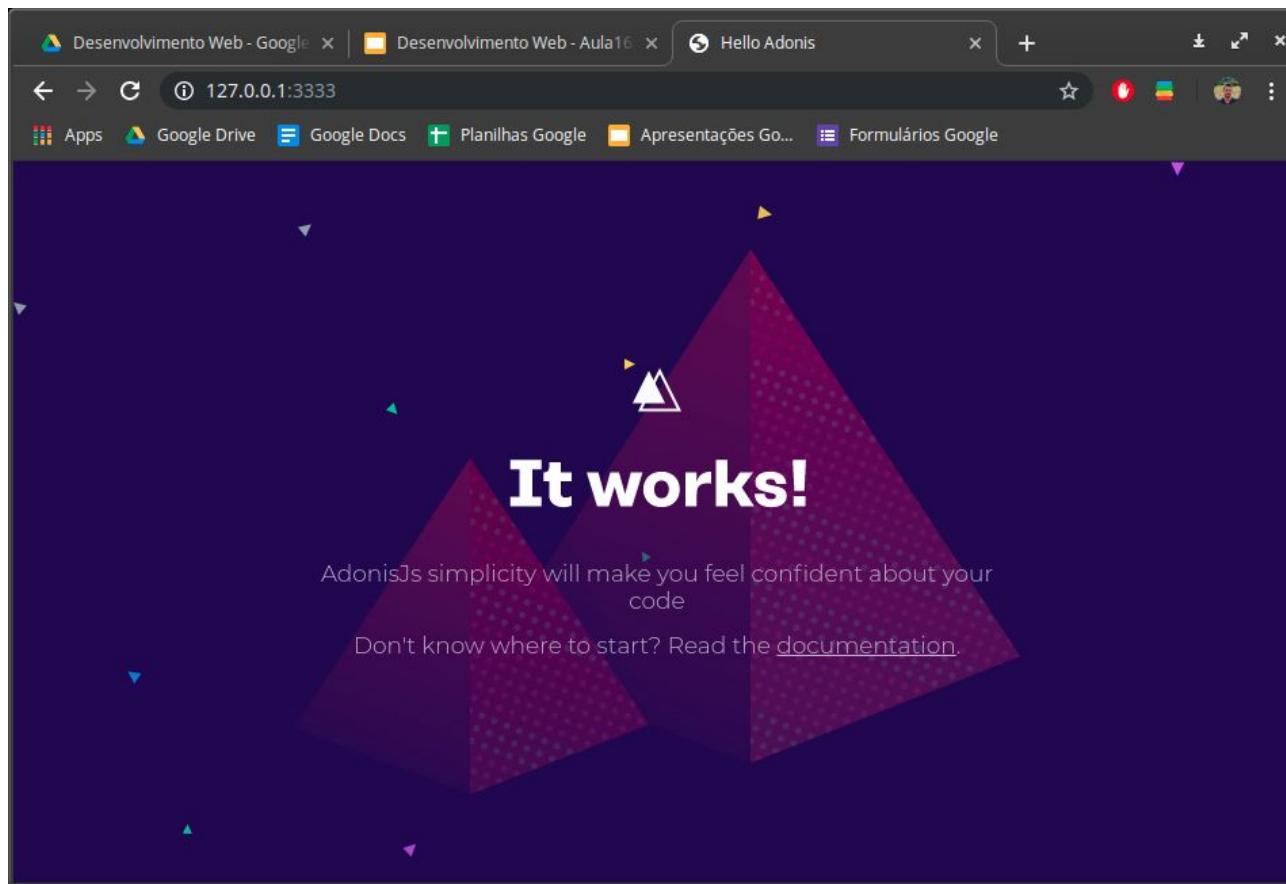
🚀 Successfully created project
👉 Get started with the following commands

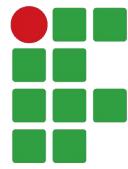
$ cd info-todo-list
$ adonis serve --dev
```



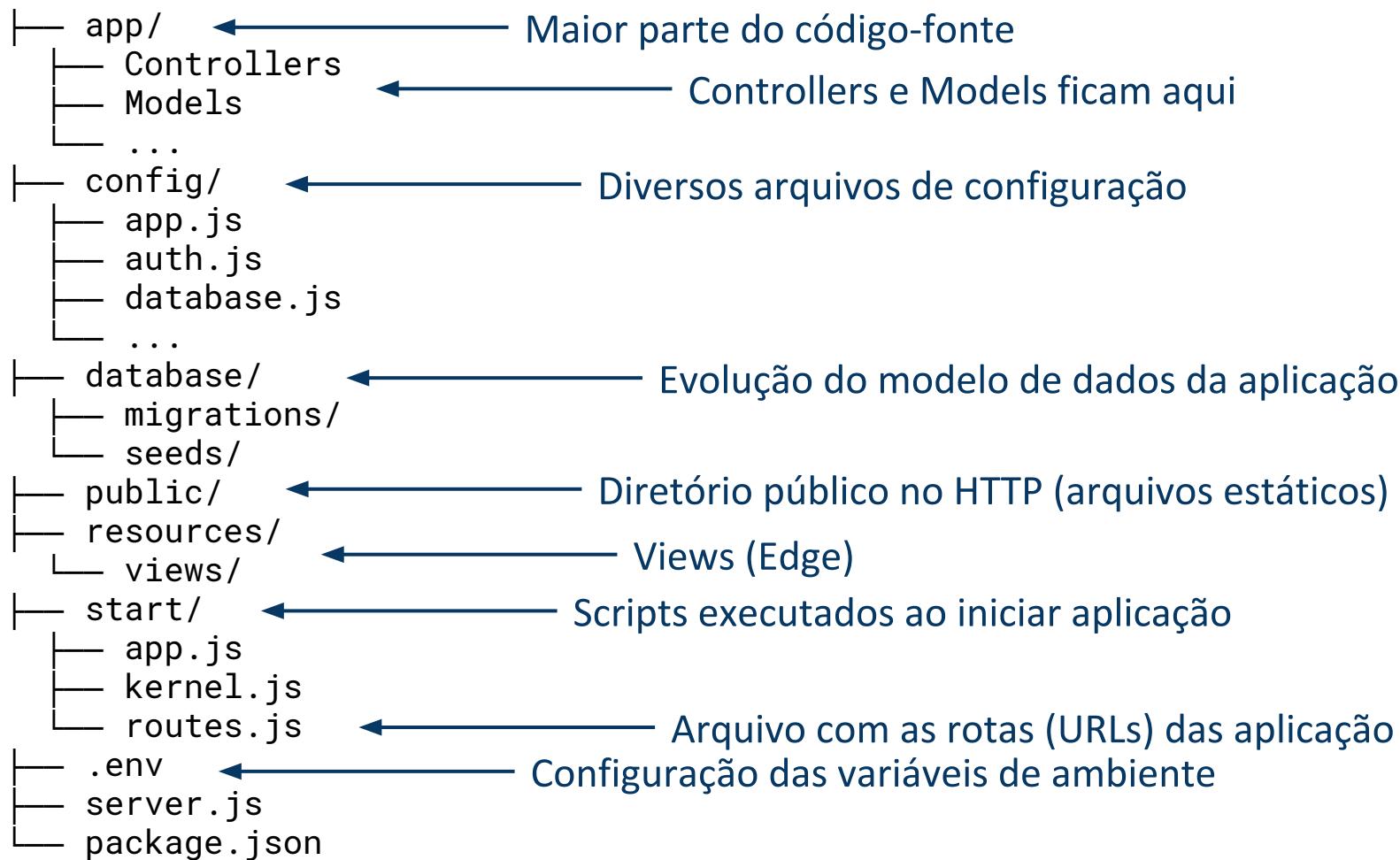
Iniciando Aplicação

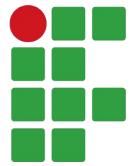
adonis serve --dev





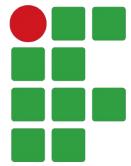
Estrutura do Projeto (resumo)





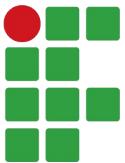
Modelo MVC

- A estrutura geral de aplicações AdonisJs é bem parecida com a que tentamos montar em nossa aplicação com Express
 - Só que com muito mais recursos prontos
- Baseado no **modelo MVC**
 - Models ← Dados
 - Views ← Páginas
 - Controllers ← Define e agrupa ações
 - Routes ← Mapeia URLs para ações de controllers



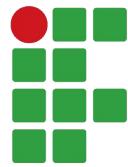
Banco de Dados

- Uma aplicação AdonisJs já nasce fortemente ligada a um banco de dados
- O próprio framework controla a evolução do modelo de dados
- Evolução do BD com migrações
 - Não precisamos escrever SQL
- Independência de banco de dados
 - Suporta → PostgreSQL, MySQL, SQLite3, MariaDB, Oracle, MSSQL



SQLite3

- Por ora, vamos usar o SQLite3
 - Banco pequeno
 - Único arquivo
 - Auto-contido
 - Embutível
- SQLite não serve para produção
 - Mas é muito bom para desenvolvimento e testes rápidos



Configurando SQLite3

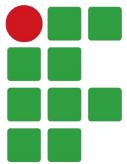
- Abrir arquivo `.env`
- Verificar entrada `DB_CONNECTION`
- Deve estar assim:

`DB_CONNECTION=sqlite`

- Instalar biblioteca

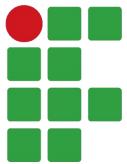
`npm install sqlite3 --save`

- O resto da configuração já está feita no arquivo **config/database.js**



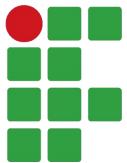
Migrações

- O AdonisJs utiliza migrações para evoluir gradualmente nosso banco de dados
- Arquivos no diretório:
 - **database/migrations/**
 - São executados em ordem (como um script SQL)
- A aplicação já vem com algumas
 - Verifique a estrutura delas
 - Método up() e down()
- Logo criaremos uma nossa



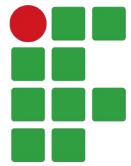
Model

- Models são a versão orientada a objetos dos dados do BD (definido nas migrações)
- Utiliza um sub-framework
 - Lucid (models, migrações, consultas, ...)
- São criados com classes
 - Dentro do diretório **app/Models/**
- Verifique os models já existentes
 - Equivalentes às migrações



Controller

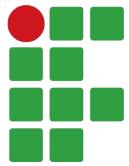
- Controllers definem as ações da aplicação
 - Requisição, resposta, parâmetros, ...
- São criados com classes
 - Dentro do diretório **app/Controllers/Http/**
- Cada método definido pode se tornar uma ação da aplicação



Geradores de Código

- Notícia boa
 - Não precisamos criar tuuuudo na mão
- O AdonisJs vem com vários geradores
- Execute no terminal:

```
adonis --help
```



adonis --help

```
↳ adonis --help
Usage:
  command [arguments] [options]

Global Options:
  --env          Set NODE_ENV before running the commands
  --no-ansi      Disable colored output
  --version      output the version number

Available Commands:
  addon          Create a new AdonisJs addon
  install        Install Adonisjs provider from npm/yarn and run post install instructions
  new            Create a new AdonisJs application
  repl            Start a new repl session
  seed            Seed database using seed files
  serve           Start Http server

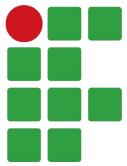
key
  key:generate   Generate secret key for the app

make
  make:command   Make a new ace command
  make:controller Make a new HTTP or Websocket channel controller
  make:ehandler   Make a new global exception handler
  make:exception  Make a new exception
  make:hook       Make a new lucid model hook
  make:listener   Make a new event or redis listener
  make:middleware Make a new HTTP or Ws Middleware
  make:migration  Create a new migration file
  make:model     Make a new lucid model
  make:provider   Make a new provider
  make:seed       Create a database seeder
  make:trait     Make a new lucid trait
  make:view      Make a view file

migration
  migration:refresh Refresh migrations by performing rollback and then running from start
  migration:reset   Rollback migration to the first batch
  migration:rollback Rollback migration to latest batch or to a specific batch number
  migration:run     Run all pending migrations
  migration:status  Check migrations current status

route
  route:list      List all registered routes

run
  run:instructions Run instructions for a given module
```



make:model

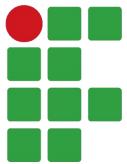
- Execute no terminal:

```
adonis make:model --help
```

Options:

-m, --migration Generate **migration** for the model
-c, --controller Generate **resourceful controller** for the model

- Além do model, o comando pode criar uma migração e um controller com ações para um CRUD completo



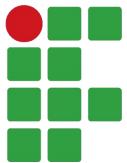
make:model

- Executar:

```
adonis make:model Tarefa -mc
```

- Arquivos criados

- app/Models/Tarefa.js
- database/migrations/999999999999_tarefa_schema.js
- app/Controllers/Http/TarefaController.js



Migração

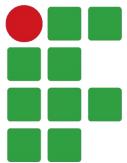
```
'use strict'

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

class TarefaSchema extends Schema {
  up () {
    this.create('tarefas', (table) => {
      table.increments()
      table.string('titulo', 100).notNullable();
      table.text('descricao');
      table.boolean('concluida').defaultTo(false);
      table.integer('user_id').unsigned().references('id').inTable('users'); // para uso futuro
      table.timestamps()
    })
  }

  down () {
    this.drop('tarefas')
  }
}

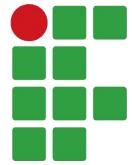
module.exports = TarefaSchema
```



Model

- O model gerado fica vazio mesmo
 - Por enquanto
- O framework consegue “adivinar” os atributos e outras características
 - Com base nas migrações

```
class Tarefa extends Model {  
}
```

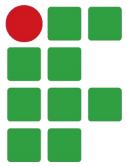


Executar Migração

- Uma migração só “vale” depois de executada
- Executar:

```
adonis migration:run
```

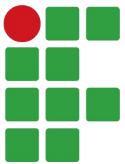
- O comando verifica todas migrações ainda não executadas
 - Executa somente elas
 - Tabelas são criadas no BD



TarefaController

- Classe criada pelo gerador make:auth
- Métodos pronto para uso

```
class TarefaController {  
    async index({ request, response, view }) {}  
    async create({ request, response, view }) {}  
    async store({ request, response }) {}  
    async show({ params, request, response, view }) {}  
    async edit({ params, request, response, view }) {}  
    async update({ params, request, response }) {}  
    async destroy({ params, request, response }) {}  
}
```

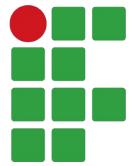


Rota

- Rotas mapeiam URLs para controllers
- Ficam no arquivo:
 - **start/routes.js**
- Exemplo de rota:

```
Route.get('tarefas', 'TarefaController.index');
```

- Vai gerar:
 - **http://servidor/tarefas** ← GET



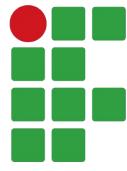
Resourceful Routes

- Como criamos um **resourceful controller**
 - Podemos criar Resourceful Routes

```
Route.resource('tarefas', 'TarefaController')
```

- É equivalente a fazer:

```
Route.get('tarefas', 'TarefaController.index').as('tarefas.index')
Route.post('tarefas', 'TarefaController.store').as('tarefas.store')
Route.get('tarefas/create', 'TarefaController.create').as('tarefas.create')
Route.get('tarefas/:id', 'TarefaController.show').as('tarefas.show')
Route.put('tarefas/:id', 'TarefaController.update').as('tarefas.update')
Route.patch('tarefas/:id', 'TarefaController.update')
Route.get('tarefas/:id/edit', 'TarefaController.edit').as('tarefas.edit')
Route.delete('tarefas/:id', 'TarefaController.destroy').as('tarefas.destroy')
```



Rotas da aplicação

- Como saber todas as rotas registradas?

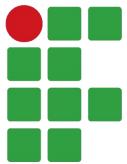
`adonis route:list`

Route	Verb(s)	Handler	Middleware	Name	Domain
/tarefas	HEAD, GET	TarefaController.index		tarefas.index	
/tarefas/create	HEAD, GET	TarefaController.create		tarefas.create	
/tarefas	POST	TarefaController.store		tarefas.store	
/tarefas/:id	HEAD, GET	TarefaController.show		tarefas.show	
/tarefas/:id/edit	HEAD, GET	TarefaController.edit		tarefas.edit	
/tarefas/:id	PUT, PATCH	TarefaController.update		tarefas.update	
/tarefas/:id	DELETE	TarefaController.destroy		tarefas.destroy	
/tarefas/:id/concluida	HEAD, GET	TarefaController.concluida		tarefas.concluida	

↑
URL

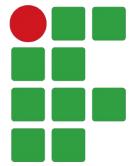
↑
Controller.action

↑
Apelido da Rota



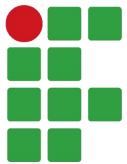
Views

- O AdonisJs usa um motor de template bem parecido com HTML puro
 - Edge
- Tem os mesmos recursos que o Pug
- Diferentemente do Pug, não precisamos reescrever o HTML com outra sintaxe



Controller e Views

- Abra este link:
<https://gist.github.com/seccomiro/1401b811a8d0877b801bee3d9fcd4d23>
- Copie o conteúdo de TarefaController.js
- Criar diretório /resources/views/**layout**/
 - Copiar **master.edge**
- Criar diretório /resources/views/**tarefas**/
 - Copiar restante das views



Views

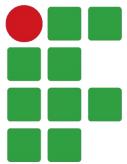
- master.edge é o template da página
 - Ele pode definir seções para as views usarem

```
@!section('conteudo')
```

- Os outros arquivos.edge são views
 - Estendem o template e usam as seções

```
@layout('layout.master')
```

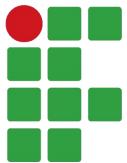
```
@section('conteudo')  
    {{-- Conteúdo único da view --}}  
@endsection
```



Views

- Para escrever no HTML, usamos `{{ }}`
- O Edge fornece acesso a alguns métodos helpers, que usamos com `@`
- Exemplo:

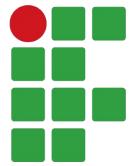
```
@each(tarefa in tarefas)
  @if(tarefa.concluida)
    <del>{{ tarefa.titulo }}</del>
  @else
    {{ tarefa.titulo }}
  @endif
@endeach
```



Controller

- Todo método do controller recebe um objeto contendo o contexto da requisição
- Podemos extrair e utilizar:
 - request
 - response
 - params
 - view
 - Dentre outros

```
async edit({ params, request, response, view }) {  
    // Código da ação  
}
```

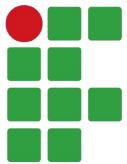


Render / Redirect

- As ações no controller normalmente vão renderizar uma view
- ```
async create({ request, response, view }) {
 const tarefa = new Tarefa();
 return view.render('tarefas.create', { tarefa });
}
```
- Parâmetros para a view  
/views/tarefas/create.edge
- Ou vai redirecionar para uma outra URL

```
async destroy({ params, request, response }) {
 let tarefa = await Tarefa.find(params.id);
 const success = await tarefa.delete();
 response.route('tarefas.index');
}
```

Apelido da rota  
Ver **adonis route:list**



# Lucid

- O Lucid (framework de persistência) fornece muitos métodos para consulta, inserção, etc

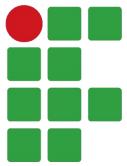
```
// Importa model Tarefa do Lucid
const Tarefa = use('App/Models/Tarefa');

// Retorna um objeto de Tarefa
Tarefa.find(params.id);

// Retorna uma lista de tarefas ordenada
Tarefa.query()
 .orderBy('concluida')
 .orderBy('updated_at', 'desc')
 .fetch();

// Cria uma tarefas no banco com base em um objeto
Tarefa.create(tarefaData);

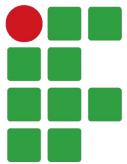
// Se tarefa for uma Tarefa, salva ela no banco (insert ou update)
tarefa.save();
```



# async / await

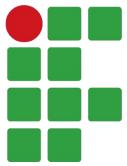
- Como acesso ao BD pode demorar, precisamos fazer de maneira assíncrona
- O AdonisJs implementa isso com Promises
  - Promise é recurso do JavaScript
    - Dizemos que uma função é assíncrona (async)
    - Chamadas de BD devem ser aguardadas (await)

```
async destroy({ params, request, response }) {
 let tarefa = await Tarefa.find(params.id);
 await tarefa.delete();
 response.route('tarefas.index');
}
```



# Ação

- Para fazer o link de concluída / não concluída
  - Criar rota
  - Criar método no controller
  - Adaptar link na view index.edge



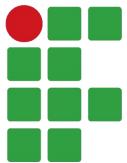
# Criando Rota

- Rota
  - Esta rota foge do padrão CRUD
  - Precisamos especificá-la

```
Route.get('/tarefas/:id/concluida', 'TarefaController.concluida').as(
 'tarefas.concluida'
)
```

The diagram illustrates the components of a route definition. It shows three parts of the code with corresponding labels:

- URL**: Points to the URL pattern `/tarefas/:id/concluida`.
- Controller.action**: Points to the controller method `TarefaController.concluida`.
- Apelido da rota**: Points to the alias `tarefas.concluida`.

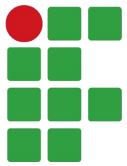


# Método

- Criar em TarefaController

```
async concluida({ params, request, response }) {
 let tarefa = await Tarefa.find(params.id);
 tarefa.concluida = !tarefa.concluida;
 await tarefa.save();
 response.route('tarefas.index');
}
```

- Encontra a tarefa pelo id da URL, inverte a propriedade concluida, salva no BD e redireciona de volta para a index

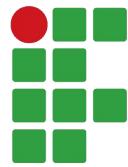


# Atualizar Link

- Atualize o link já existente em index.edge

```
<a
 class="text-dark"
 href="{{ route('tarefas.concluida', { id: tarefa.id }) }}"
 >
 <i class="far fa{{ tarefa.concluida ? '-check' : '' }}-square"></i>

```



# Mais Informações

- Documentação do AdonisJs:
  - <https://adonisjs.com/docs>
- Adonis 4 Tutorial - Learn Adonis 4 in this Crash Course:
  - <https://coursetro.com/posts/code/170/Adonis-4-Tutorial---Learn-Adonis-4-in-this-Crash-Course>