

**INSTITUTO FEDERAL**

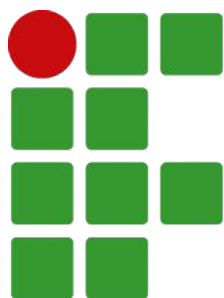
Paraná

Campus Paranaguá

# AdonisJs - Associações e Validações

Desenvolvimento Web

Prof. Diego Stiehl



**INSTITUTO FEDERAL**

Paraná

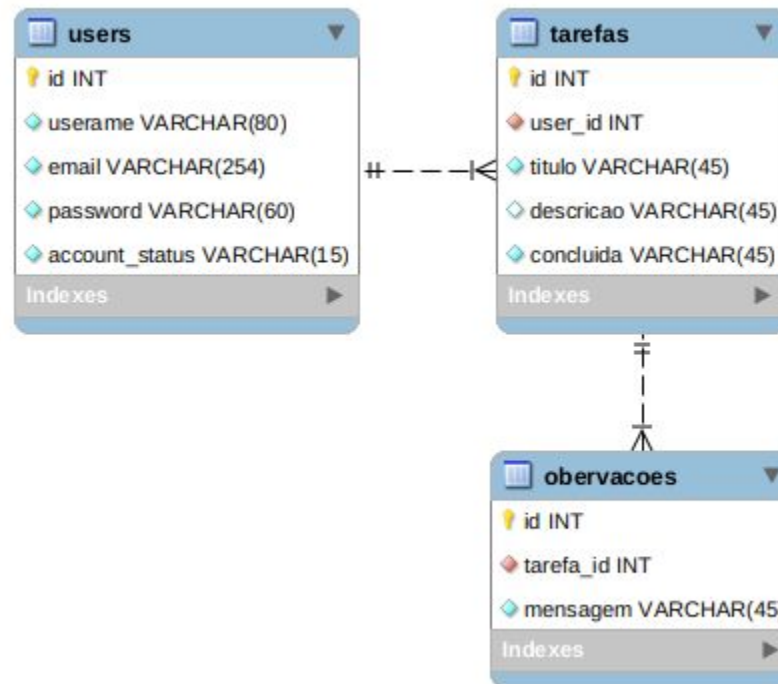
Campus Paranaguá

Vinculando diferentes models

# **Associações**

# Associações

- Vamos supor que queremos adicionar Observações para cada Tarefa
  - Uma **Tarefa** possui muitas **Observações**



# Observações

---

- Precisamos
  - Criar um novo model ← Observacao
  - Alterar o model de Tarefa
  - Criar sua migração
  - Criar uma rota ← /tarefas/:id/observacao
  - Habilitar cadastro e consulta
    - Alterar views
    - Alterar TarefaController

# Criar model/migration

- Execute

```
adonis make:model Observacao -m
```

- Irá criar:
  - app/Models/Observacao.js
  - database/migrations/9999999999999999\_observaca  
o\_schema.js



# Migration

```
class ObservacaoSchema extends Schema {  
  up() {  
    this.create('observacoes', table => {  
      table.increments();  
      table.text('mensagem').nullable();  
      table.integer('tarefa_id').unsigned().references('id').inTable('tarefas');  
      table.timestamps();  
    });  
  }  
  
  down() {  
    this.drop('observacoes');  
  }  
}
```

# Model Observacao

---

```
class Observacao extends Model {  
  static get table() {  
    return 'observacoes';  
  }  
  
  tarefa() {  
    return this.belongsTo('App/Models/Tarefa');  
  }  
}
```

# Model Tarefa

```
class Tarefa extends Model {  
    // ... Resto da classe  
    observacoes() {  
        return this.hasMany('App/Models/Observacao');  
    }  
}
```



# Adicionar Rota

---

- `start/routes.js`


```
Route.post('/tarefas/:id/observacao', 'TarefaController.observacao')  
  .as('tarefas.observacao')  
  .middleware(['auth', 'protegeTarefa']);
```

# show/observacao

```
const Observacao = use('App/Models/Observacao');
```

```
class TarefaController {  
  // ... Resto da classe
```


Carrega observações  
automaticamente (faz SQL para nós)

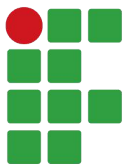


```
  async show({ view, tarefa }) {  
    await tarefa.load('observacoes');  
    const observacoes = tarefa.getRelated('observacoes').rows;  
    return view.render('tarefas.show', { tarefa, observacoes });  
  }
```

```
  async observacao({ request, response, tarefa }) {  
    const observacaoData = request.only(['mensagem']);  
    observacaoData.tarefa_id = tarefa.id;  
    await Observacao.create(observacaoData);  
    response.route('tarefas.show', { id: tarefa.id });  
  }  
}
```

Cria nova  
observação  
vinculada com  
a tarefa da URL





INSTITUTO  
FEDERAL

# show.edge

Pegar o código em:

<https://gist.github.com/seccomiro/9feb531ed05f6fb3f36f86a0ee7658eb>

## Fazer Outras Coisas

Lá em casa.

Editar

Remover

Voltar

Observação importante

Outra observação

Não esquecer

Etc

Observações já  
cadastradas da Tarefa

## Nova Observação

Descrição

Formulário para inclusão  
de novas observações

Adicionar

# Lista de Tarefas

## Minhas Tarefas

Nova Tarefa

☐ Fazer Outras Coisas

4

☐ Fazer alguma coisa

3

☐ Nada

Contagem de observações

# Lista de Tarefas

## TarefaController

```
async index({ request, response, view, auth }) {  
  const busca = request.input('busca') || '';  
  const tarefas = (await auth.user  
    .tarefas()  
    .withCount('observacoes')  
    .where('titulo', 'like', `%${busca}%`)  
    .orderBy('concluida')  
    .orderBy('updated_at', 'desc')  
    .fetch()).rows;  
  return view.render('tarefas.index', { tarefas });  
}
```

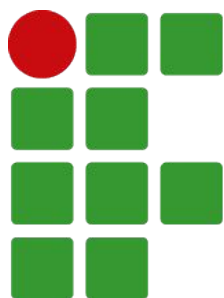
<https://gist.github.com/seccomiro/c046c8576cb5cb39b4d0a5f7a7869ce7>

COMPLETO



## index.edge

```
<li class="list-group-item d-flex justify-content-between align-items-center"  
  style="font-size: 150%">  
  <div>  
    ... Título e caixa de seleção (nada muda)  
  </div>  
  @if(tarefa.observacoes_count > 0)  
    <span class="badge badge-warning">{{ tarefa.observacoes_count }}</span>  
  @endif  
</li>
```



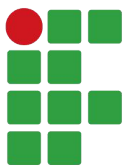
**INSTITUTO FEDERAL**

Paraná

Campus Paranaguá

Verificando dados de formulários

# Validações



# Validação

---

- Precisamos validar nossos formulários
  - Procurar por erros por parte dos usuários
- Se quisermos podemos fazer isto dentro de cada ação do controller
  - Gera repetição de código
  - Incha o controller
  - Dificulta a leitura dos métodos

# Validation Provider

---

- AdonisJs permite a validação de dados através de um Validation Provider
- Ele já oferece algumas regras prontas
  - Campo obrigatório, tamanho de string, atributo único, se um e-mail é válido, caracteres alfanuméricos, ...
- Caso validação **falhe**:
  - Retorna para formulário, injetando mensagens



# Criar Validator

- Executar:

```
adonis make:validator SalvarTarefa
```

- Irá criar:
  - app/Validators/SalvarTarefa.js

```
'use strict';
```

```
class SalvarTarefa {  
  get rules() {  
    return {  
      // validation rules  
    };  
  }  
}
```

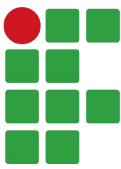
Retorna objeto com  
as regras de validação

```
module.exports = SalvarTarefa;
```

# Validar Rotas

- Passar validação em store e update

```
Route.resource('tarefas', 'TarefaController')  
  .middleware('auth')  
  .middleware(  
    new Map([['show', 'edit', 'update', 'destroy'], ['protegeTarefa']])  
  )  
  .validator(new Map([['store', 'update'], ['SalvarTarefa']]));
```



# SalvarTarefa.js

```
'use strict';
```

```
class SalvarTarefa {  
  get rules() {  
    const { id } = this.ctx.params;  
    return {  
      titulo: `required|min:5|max:30|unique:tarefas,titulo,id,${id}`,  
      descricao: 'required'  
    };  
  }  
  
  get messages() {  
    return {  
      'titulo.required': 'Por favor informe o título da tarefa',  
      'titulo.min': 'O título deve ter pelo menos 5 caracteres',  
      'titulo.max': 'O título deve ter no máximo menos 30 caracteres',  
      'titulo.unique': 'Tarefa já cadastrada',  
      'descricao.required': 'Por favor informe uma descrição'  
    };  
  }  
  
  get validateAll() {  
    return true;  
  }  
}
```

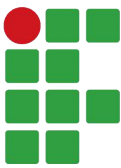
Regras para validação do título

Regras para validação da descrição

Mensagens customizadas para  
cada regra de cada atributo

Indica para executar todas as  
validações, mesmo que uma falhe

```
module.exports = SalvarTarefa;
```



# form.edge

```
<div class="form-group">
  <label for="tarefa_titulo">Título</label>
  <input type="text" class="form-control" id="tarefa_titulo"
    name="titulo" value="{{ old('titulo', tarefa.titulo || '') }}">
  @if(hasErrorFor('titulo'))
    <small class="form-text text-danger">{{ getErrorFor('titulo') }}</small>
  @endif
</div>
<div class="form-group">
  <label for="tarefa_descricao">Descrição</label>
  <textarea class="form-control" id="tarefa_descricao"
    name="descricao" rows="3">{{ old('descricao', tarefa.descricao || '') }}</textarea>
  @if(hasErrorFor('descricao'))
    <small class="form-text text-danger">{{ getErrorFor('descricao') }}</small>
  @endif
</div>
```

# Validador Customizado

```
'use strict';
```

```
// unicaNaoConcluida (código do slide anterior)
```

```
class SalvarTarefa {  
  get rules() {  
    const { auth, params } = this.ctx;  
    const { id } = params;  
    return {  
      titulo: `required|min:5|max:30|unicaNaoConcluida:${auth.user.id},${id}`,  
      descricao: 'required'  
    };  
  }  
  get messages() {  
    return {  
      // ...OUTRAS  
      'titulo.unicaNaoConcluida': 'Você já tem uma tarefa pendente com este título',  
    };  
  }  
  // ...validateAll()  
}
```

```
module.exports = SalvarTarefa;
```

# Validador Customizado

```
const Tarefa = use('App/Models/Tarefa');
const Validator = use('Validator');

const unicaNaoConcluida = async (data, field, message, args, get) => {
  const value = get(data, field);
  if (!value) {
    return;
  }
  const [userId, tarefaId] = args;
  // Retorna a contagem de tarefas não concluídas com mesmo título deste usuário
  const contagem = (await Tarefa.query()
    .where('user_id', userId).whereNot('id', tarefaId).where('concluida', false)
    .where('titulo', value).count('* as contagem'))[0].contagem;
  if (contagem > 0) {
    throw message;
  }
};

Validator.extend('unicaNaoConcluida', unicaNaoConcluida);
```

# Validador Customizado

```
'use strict';  
  
// unicaNaoConcluida (código do slide anterior)  
  
class SalvarTarefa {  
  get rules() {  
    const { auth, params } = this.ctx;  
    const { id } = params;  
    return {  
      titulo: `required|min:5|max:30|unicaNaoConcluida:${auth.user.id},${id}`,  
      descricao: 'required'  
    };  
  }  
  
  get messages() {  
    return {  
      // ...OUTRAS  
      'titulo.unicaNaoConcluida': 'Você já tem uma tarefa pendente com este título',  
    };  
  }  
  
  // ...validateAll()  
}
```

```
module.exports = SalvarTarefa;
```