

Segurança em Primeiro Lugar

Como construir a camada de autorização para sua aplicação Rails



switch
dreams

O que é autorização?

Ação ou resultado de autorizar; conceder permissão para que alguém faça alguma coisa; concessão.

<https://www.dicio.com.br/autorizacao/>

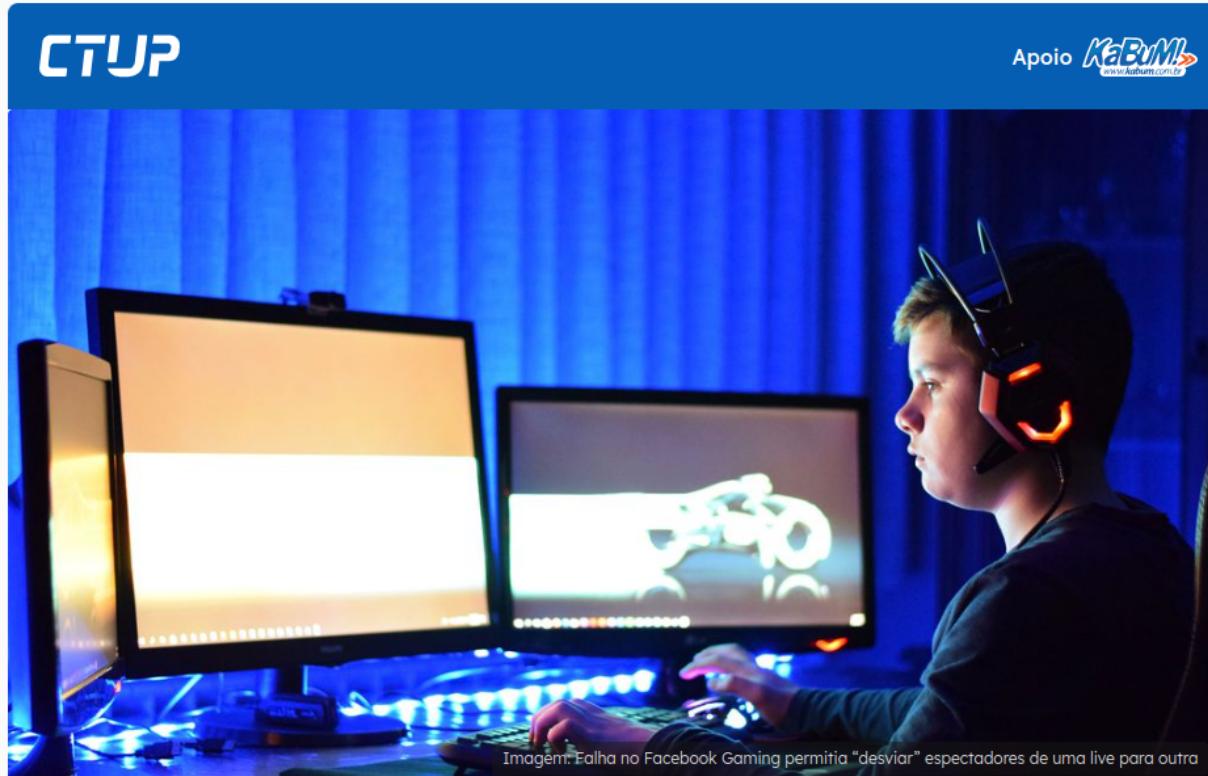
No nosso mundo?

Como conceder e verificar acesso de um usuário dos nossos sistemas?

Por que é tão importante para a
segurança das aplicações?

Falha no Facebook Gaming permitia “desviar” espectadores de uma live para outra

Por [Felipe Demartini](#) | 06 de Março de 2020 às 11h18



Essa é uma vulnerabilidade extremamente simples de ser explorada, mas bem complicada para quem tem a tarefa de proteger as aplicações

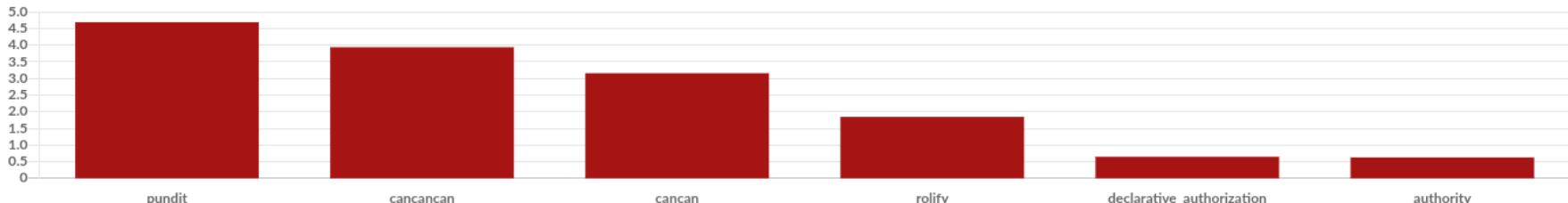
Gabriel Pato

Uma boa camada de autorização
é essencial para a segurança sua
aplicação rails

Qual melhor ferramenta para
fazer isso?

User Authorization

Narrow down the visibility and accessibility of operations for users based on roles and similar access control patterns



Full
 Compact
 Table
 Order by Project Score

	PROJECT SCORE	DOWNLOADS	STARS	FORKS	FIRST RELEASE	LATEST RELEASE	REVERSE DEPENDENCIES
pundit	4.69	59,766,145	8,104	601	2012-11-19	2023-07-17	108
cancancan	3.94	59,167,463	5,476	605	2014-02-19	2023-03-05	★ 163
cancan	3.16	12,713,381	6,289	★ 742	2009-11-16	2013-05-07	126
rolify	1.85	15,752,231	3,130	395	2011-06-03	2023-02-02	18
declarative_authorization	0.65	1,323,824	1,244	189	2009-10-12	2013-03-10	6
authority	0.63	7,378,844	1,215	66	2012-03-12	2017-02-07	1
action_policy	0.56	2,792,517	1,290	85	2018-02-12	2024-01-17	3

Gem Action Policy

- Necessita escrever menos código
- Performance (Cache)
- Testabilidade
- Flexibilidade
- Convenção com o Rails
- I18n

<-- Pundit com o poder do Suco -->

Como eu posso criar essa
camada de autorização?

Vamos para o código!

To Do list

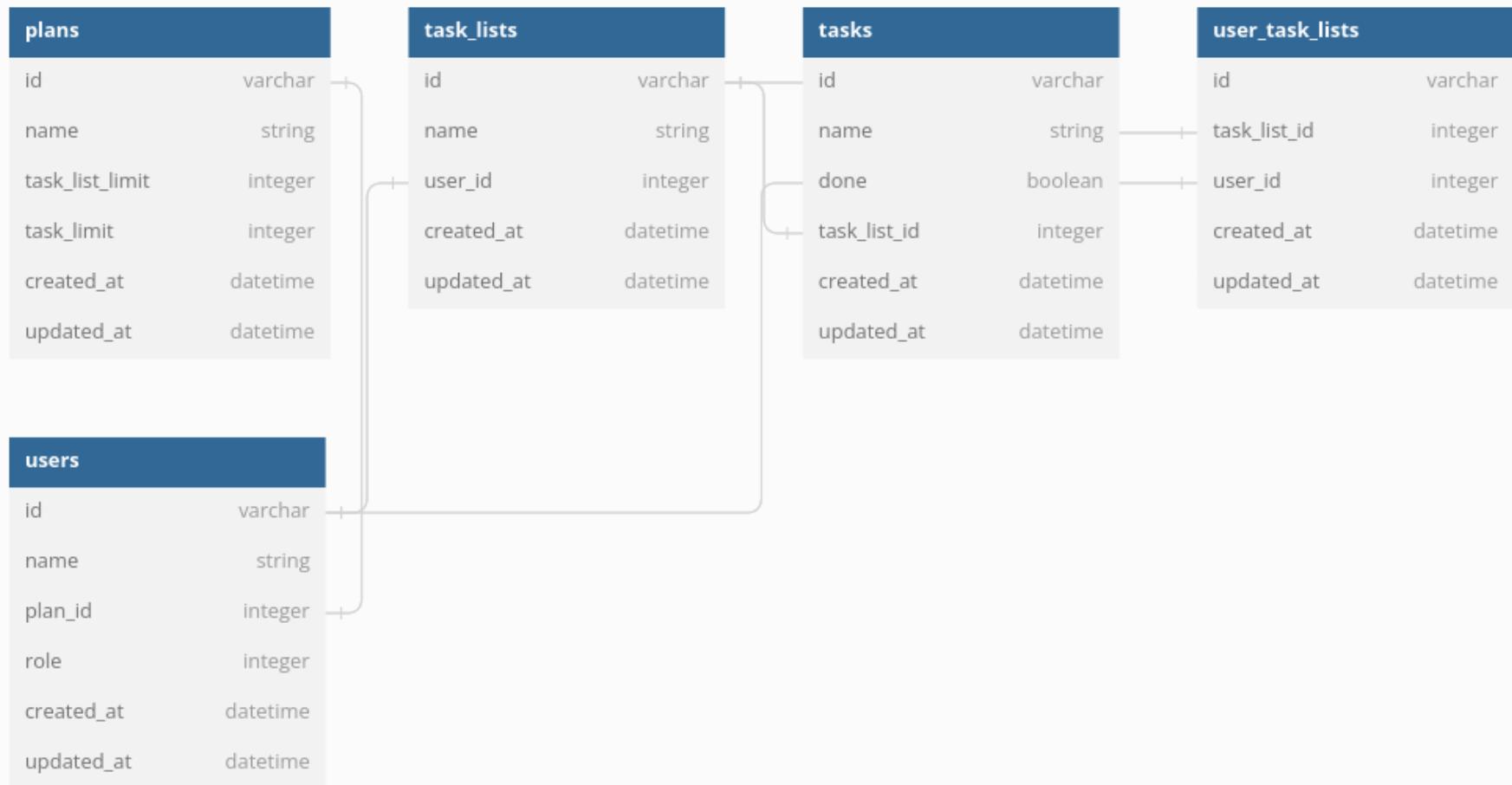
- Rates
- Points
- Leases
- Security
- Training
- Tailoring



SWIPE

Features

- A aplicação possui admins e usuários normais
- Um usuário pode criar uma lista de tarefas
- É possível convidar um usuário para ver a lista
- Existem planos que limitam a quantidade de listas e de tarefas por listas



```
# Sem utilizar o action policy
class TaskListsController < ApplicationController
  before_action :set_task_list, only: [:show]

  def index
    @task_lists = current_user.admin? ? Task.all :
      TaskList.left_outer_joins(:user_task_lists)
      .where('task_lists.user_id = ? OR user_task_lists.user_id = ?', user.id, user.id)
  end

  def show
    if !current_user.admin? and @task_list.user_id != current_user.id and !@task_list.users.includes?(current_user)
      unauthorized
    end
    @tasks = @task_list.tasks
  end

  def create
    if !current_user.admin? and current_user.task_lists.count >= current_user.plan.task_list_limit
      unauthorized(message: "Você chegou no limite de lista de tarefas do seu plano")
    end
    @task_list = TaskList.create!(task_list_params)
    rescue StandardError => e
      flash[:error] = e.full_message
      redirect_to task_lists_path
  end
end
```

Como refatorar com o Action Policy?

Resumo para Autorização

- Quais dados dentre esses recursos meu usuário tem acesso? (Scopes)
- Dado esse recurso meu usuário pode executar tal ação? (Rules)

1 - Criar a Policy

```
# Base class for application policies
class ApplicationPolicy < ActionPolicy::Base
  pre_check :allow_admins

  private

    # Define shared methods useful for most policies.
    # For example:
    #
    def owner?
      record.user_id == user.id
    end

    def allow_admins
      allow! if user.admin?
    end
end
```

Nessa TaskList o meu usuário pode executar tal ação?

```
class TaskListPolicy < ApplicationPolicy
  def index?
    true
  end

  def show?
    owner? || record.users.include?(user)
  end

  def manage?
    owner?
  end

  def create?
    user.task_lists.count < user.plan.task_list_limit
  end
end
```

Quais TaskList meu usuário tem acesso?

```
class TaskListPolicy < ApplicationPolicy
  # ...

  relation_scope do |relation|
    next relation if user.admin?

    relation.left_outer_joins(:user_task_lists)
      .where('task_lists.user_id = ? OR user_task_lists.user_id = ?', user.id, user.id)
  end
end
```

2 - Refatorar a Controller

```
class TaskListsController < ApplicationController
  before_action :set_task_list, only: [:show]

  def index
    @task_lists = current_user.admin? ? Task.all :
      TaskList.left_outer_joins(:user_task_lists)
      .where('task_lists.user_id = ? OR user_task_lists.user_id = ?', user.id, user.id)
  end

  def show
    if !current_user.admin? and @task_list.user_id != current_user.id and !@task_list.users.includes?(current_user)
      unauthorized
    end
    @tasks = @task_list.tasks
  end

  def create
    if !current_user.admin? and current_user.task_lists.count >= current_user.plan.task_list_limit
      unauthorized(message: "Você chegou no limite de lista de tarefas do seu plano")
    end
    @task_list = TaskList.create!(task_list_params)
  rescue StandardError => e
    flash[:error] = e.full_message
    redirect_to task_lists_path
  end
end
```

~~~

```
class TaskListsController < ApplicationController
  before_action :set_task_list, only: [:show]
  before_action -> { authorize! @task_list, with: TaskListPolicy }

  def index
    @task_lists = authorized_scope(TaskList.all)
  end

  def show
    @tasks = @task_list.tasks
  end

  def create
    @task_list = TaskList.create!(task_list_params)
    rescue StandardError => e
      flash[:error] = e.full_message
      redirect_to task_lists_path
    end
  end
```

# Resumo

- Nas controllers: ``authorize!`` como `guard_clause`
- Filtrar recursos que o usuário tem acesso: ``authorized_scope``
- Nas views: ``allowed_to?``

# Testes

# Antes

```
describe "GET /task_lists" do
  before do
    get "/tasks_lists"
  end

  context "quando não possui usuário logado" do
  end
  context "quando o usuário é admin" do
  end
  context "quando o usuário é normal" do
  end
end
```

# Depois

```
# Add this to your spec_helper.rb / rails_helper.rb
require "action_policy/rspec/dsl"

describe TaskPolicy do
  let(:user) { build_stubbed :user }
  let(:record) { build_stubbed :task_list }
  let(:context) { { user: user } }

  describe_rule :show? do
    failed "quando o usuário não tiver relação com a lista de tarefas"

    succeed "quando o usuário é admin" do
      let(:user) { build_stubbed :user, role: :admin }
    end

    succeed "quando o usuário é dono da lista de tarefas" do
      let(:user) { build_stubbed :user, role: :admin }
    end
  end
end
```

# Depois

```
describe "GET /task_lists" do
  before do
    get "/tasks_lists"
  end

  it "renderiza a lista de tarefa com sucesso" do
    #....
  end
end
```



# Obrigado!

Pedro Augusto Ramalho Duarte

