Solución Desafío - Usando un controlador

Descripción

Recordando el Desafío I, dado el siguiente controlador, refactorizar el código, de tal forma que la lógica de negocio pase al modelo correspondiente.

```
class CartsController < ApplicationController
 before_action :authenticate_user!

 def update
   product = params[:cart][:product_id]
   quantity = params[:cart][:quantity]

   current_order.add_product(product, quantity)

   redirect_to root_url, notice: "Product added successfuly"
 end

 def show
   @order = current_order
 end

 def pay_with_paypal
   order = Order.find(params[:cart][:order_id])

   #price must be in cents
   price = order.total * 100

   response = EXPRESS_GATEWAY.setup_purchase(price,
     ip: request.remote_ip,
     return_url: process_paypal_payment_cart_url,
     cancel_return_url: root_url,
     allow_guest_checkout: true,
     currency: "USD"
  )

   payment_method = PaymentMethod.find_by(code: "PEC")
   Payment.create(
     order_id: order.id,
      payment_method_id: payment_method.id,
      state: "processing",
      total: order.total,
      token: response.token
   )

    redirect_to EXPRESS_GATEWAY.redirect_url_for(response.token)
  end



  def process_paypal_payment
    details = EXPRESS_GATEWAY.details_for(params[:token])
    express_purchase_options =
     {
        ip: request.remote_ip,
        token: params[:token],
        payer_id: details.payer_id,
        currency: "USD"
     }

    price = details.params["order_total"].to_d * 100

    response = EXPRESS_GATEWAY.purchase(price, express_purchase_options)
    if response.success?
      payment = Payment.find_by(token: response.token)
      order = payment.order

      #update object states
      payment.state = "completed"
      order.state = "completed"

      ActiveRecord::Base.transaction do
        order.save!
        payment.save!
      end
    end
  end
end
```

Asumir

- current_order siempre entrega una Order válida.

Requerimientos

1. Implementar los tests unitarios para los modelos.
2. Mover el código desde el controlador hacia los modelos.
3. Refactorizar el código.

Al finalizar

1. Cada método debe cumplir con el principio de Single Responsibility, incluso en el controlador. Modificar los métodos que sean necesarios.

Solución

1. Aplicando los tests:

```
class OrderTest < ActiveSupport::TestCase

 test 'pass total price to cents' do
user = User.create(email: "user@example.com", password: "12345678")
   order = Order.create(user_id: user.id, total: 100)
   assert_equal order.total_in_cents, 10000
 end

 test 'order creates a payment' do
   user = User.create(email: "user@example.com", password: "12345678")
   order = Order.create(user_id: user.id, total: 100)
   PaymentMethod.create(name: "Paypal Express Checkout", code: "PEC")

   order.create_payment("PEC", "token_00112233")
   assert_equal order.payments.count, 1
 end

 test 'new payment have processing state' do
   user = User.create(email: "user@example.com", password: "12345678")
   order = Order.create(user_id: user.id, total: 100)
   PaymentMethod.create(name: "Paypal Express Checkout", code: "PEC")

   order.create_payment("PEC", "token_00112233")

   assert_equal order.payments.last.state, "processing"
 end

 test 'order is marked as completed' do
   user = User.create(email: "user@example.com", password: "12345678")
   order = Order.create(user_id: user.id, total: 100)
   order.complete!
   assert_equal order.state, "completed"
 end
end
```

```
class PaymentTest < ActiveSupport::TestCase

 test 'payment is marked as completed' do
   user = User.create(email: "user@example.com", password: "12345678")
   order = Order.create(user_id: user.id, total: 100)

   pm = PaymentMethod.create(name: "Paypal Express Checkout", code:
"PEC")

   payment = Payment.create(order_id: order.id, payment_method_id:
pm.id, state: "processing")

   payment.complete!

   assert_equal payment.state, "completed"
 end

 test 'complete payments' do
   user = User.create(email: "user@example.com", password: "12345678")
   order = Order.create(user_id: user.id, total: 100)

   pm = PaymentMethod.create(name: "Paypal Express Checkout", code:
"PEC")

   payment = Payment.create(order_id: order.id, payment_method_id:
pm.id, state: "processing")
```

```
   payment.close!

   assert_equal payment.state, "completed"
   assert_equal payment.order.state, "completed"
 end
end
```

2. Implementación de código en los modelos:

```
class Order < ApplicationRecord

 #...

 def total_in_cents
   total * 100
 end

 def create_payment(pm_code, token)
   payments << Payment.create(
     payment_method_id: PaymentMethod.find_by(code: pm_code).id,
     state: "processing",
     total: total,
     token: token
  )
 end

 def complete!
   update_attributes({state: "completed"})
 end
end
```

```
class Payment < ApplicationRecord
 belongs_to :order
 belongs_to :payment_method


 def complete!
   update_attributes({state: "completed"})
 end

 def close!
   ActiveRecord::Base.transaction do
```

```
      complete!
      order.complete!
    end
  end
end
```

La implementación de cada método en el modelo ya está optimizada, pero podría darse el caso que se necesite refactorizar según el caso por cada alumno.

3. El controlador, refactorizado queda de la siguiente manera:

```
class CartsController < ApplicationController
  before_action :authenticate_user!
  before_action :get_order, only: [:pay_with_paypal]

  def update
    product = params[:cart][:product_id]
    quantity = params[:cart][:quantity]

    current_order.add_product(product, quantity)

    redirect_to root_url, notice: "Product added successfuly"
  end

  def show
    @order = current_order
  end

  def pay_with_paypal

    response = EXPRESS_GATEWAY.setup_purchase(@order.total_cents,
      ip: request.remote_ip,
      return_url: process_paypal_payment_cart_url,
      cancel_return_url: root_url,
      allow_guest_checkout: true,
      currency: "USD"
   )

    @order.create_payment("PEC", response.token)

    redirect_to EXPRESS_GATEWAY.redirect_url_for(response.token)
  end
```

```
 def process_paypal_payment
   @details = EXPRESS_GATEWAY.details_for(params[:token])

   response = EXPRESS_GATEWAY.purchase(processed_price,
express_purchase_options)

   if response.success?
     payment = Payment.find_by(token: response.token)
     payment.close!
   end
 end

 def express_purchase_options
  {
     ip: request.remote_ip,
     token: params[:token],
     payer_id: @details.payer_id,
     currency: "USD"
  }
 end

 def processed_price
   @details.params["order_total"].to_d * 100
 end

 def get_order
   @order = Order.find(params[:cart][:order_id])
 end
end
```