Desenvolvedor Ruby

1. Escreva uma classe Ruby que é inicializada com uma string e contém um método que retorna um hash que mapeia os caracteres da string (convertidos para minúsculos) para a sua contagem de ocorrências na string.

Por exemplo, com a string "Abóbora" seria retornado:

```
{ 'a' => 2, 'b' => 2, 'ó' => 1, 'o' => 1, 'r' => 1 }
```

Deve-se fornecer também testes unitários para o código.

2. Você deve implementar uma classe que interpreta um comando simples em Ruby. Os comandos possíveis são strings de operações aritméticas básicas e possuem os seguintes formatos:

```
ADD N1, N2
SUB N1, N2
MUL N1, N2
DIV N1, N2
```

N1 e N2 são sempre números, não podem ser expressões. O separador decimal deve sempre ser o '.'.

Exemplos:

```
ADD 12, 45
SUB 12, -37.5
MUL 0, +346
DIV 234, -34.56
```

Ao executar um comando o resultado deve ser retornado. Por exemplo:

```
> Command.new('ADD 12, 45').execute
57
```

Se um comando for inválido, deve dar exceção. Forneça as classes utilizadas com os testes unitários.

3. Dado um projeto Rails, abaixo descrevemos os models Customer e Order, o controller de customers e a view da sua action index. O código está simplificado, omitimos partes que não interessam para as questões a seguir.

```
# app/models/customer.rb
class Customer < ApplicationModel
  belongs_to :operator
  has_many :locations
  has_many :orders
end

# app/models/order.rb
class order < ApplicationModel
  belongs_to :customer
  enum status: { pending: 0, success: 1, failed: 2, rolled_back: 3 }
end

# app/controllers/customer_controller.rb
class CustomersController < ApplicationController
  def index
    @customers = Customer.order(:name)
  end
end
```

```
<!-- app/views/customer/index.html.erb -->
<table>
  <thead>
    <tr>
      <th>Cliente</th></th>
      <th>Operador</th>
      <th>Locais</th>
    </tr>
  </thead>
  <tbody>
    <%  @customers.each do |customer| %>
      <tr>
        <td><%= customer.name %></td>
        <td><%= customer.operator.name %></td>
        <td><%= customer.locations.map(&:name).join(', ') %></td>
      </tr>
    <%  end %>
  </tbody>
</table>
```

3.1. Descreva o problema com a query executada na action index e reescreva a query, corrigindo o problema.

3.2. O controller de customers somente possui as actions CRUD, geradas pelo scaffold (tirando a de index, as demais foram omitidas da listagem acima). A configuração em config/routes.rb é a seguinte:

```
  resources :customers
```

É necessario adicionar uma nova action chamada block. Como o nome já diz, essa action vai bloquear um customer. Altere a configuração de config/routes.rb para considerar essa nova action.

3.3. Precisamos agora ter uma página HTML para listar os pedidos (orders) de um determinado customer. Descreva como você resolveria esse problema. Forneça o código necessário para exemplificar.

Segue a tabela dos pedidos (simplificada, sem índices, FKs, etc):

```
# db/schema.rb
create_table "orders" do |t|
  t.integer  "operator_id", null: false
  t.integer  "customer_id", null: false
  t.integer  "location_id", null: false
  t.decimal  "value", precision: 10, scale: 2, null: false
  t.integer  "status", default: 0, null: false
end
```