Rails Jobs usando delayed jobs

https://github.com/CodevivorsG1/Expo

Equipo

CODEVIVORS

+ Dulfrey Hernández
+ Marlon Noguera
+ Giovanni Albarracín
+ Diego Rodríguez
+ Nicolas Enciso

https://github.com/CodevivorsG1/Expo

Contenido

1. Definición de Active job
2. Funcionalidad de un Active Job
3. ActiveJob en rails
4. Usando la gema Delayed job
5. Práctica

https://github.com/CodevivorsG1/Expo

Definición ActiveJob

- Implementado desde Rails 4.2
- Son procesos que se ejecutan “por debajo” en BACKGROUND
- Tienen la posibilidad de ejecución paralela.
- Dos tipos: Lineal y asincrónicos

https://github.com/CodevivorsG1/Expo

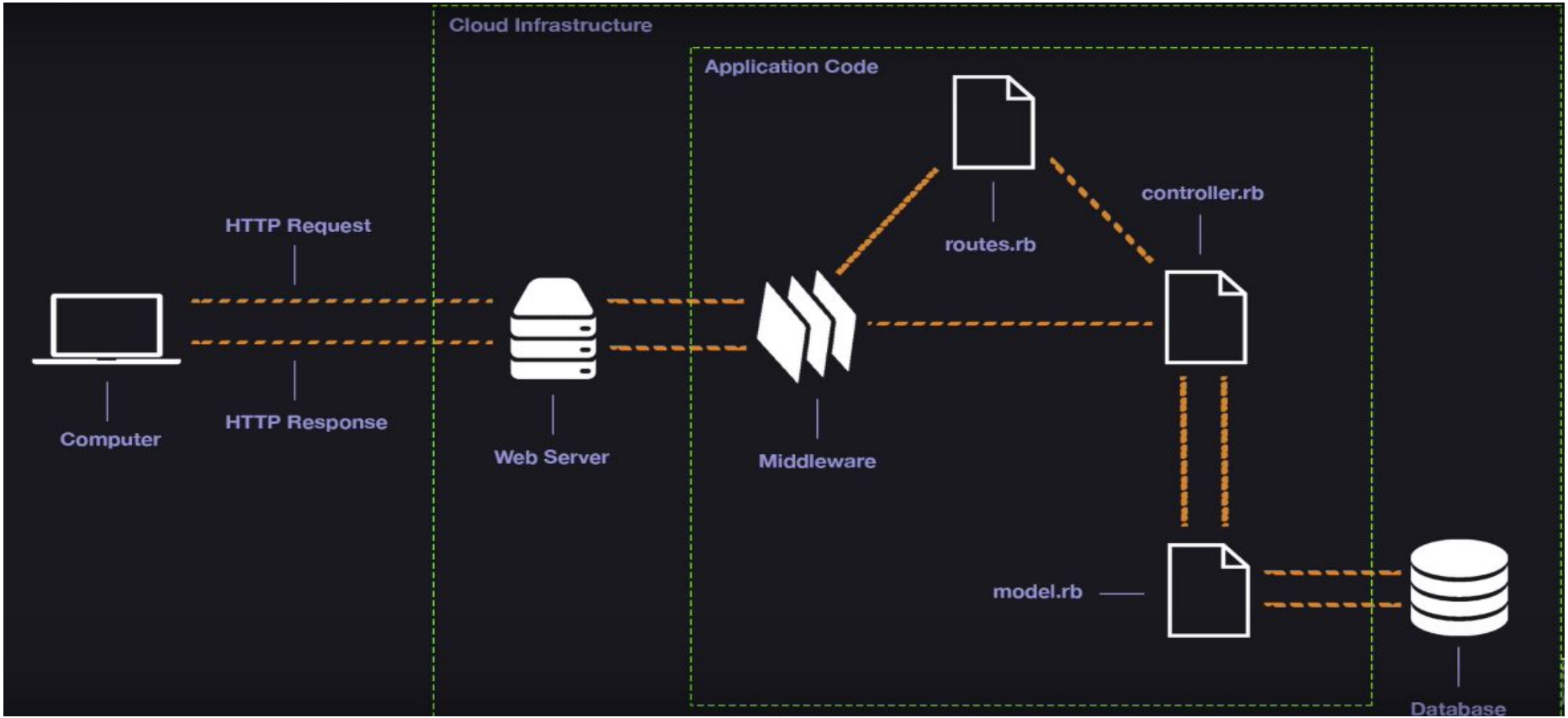
Funcionalidad de Active Job

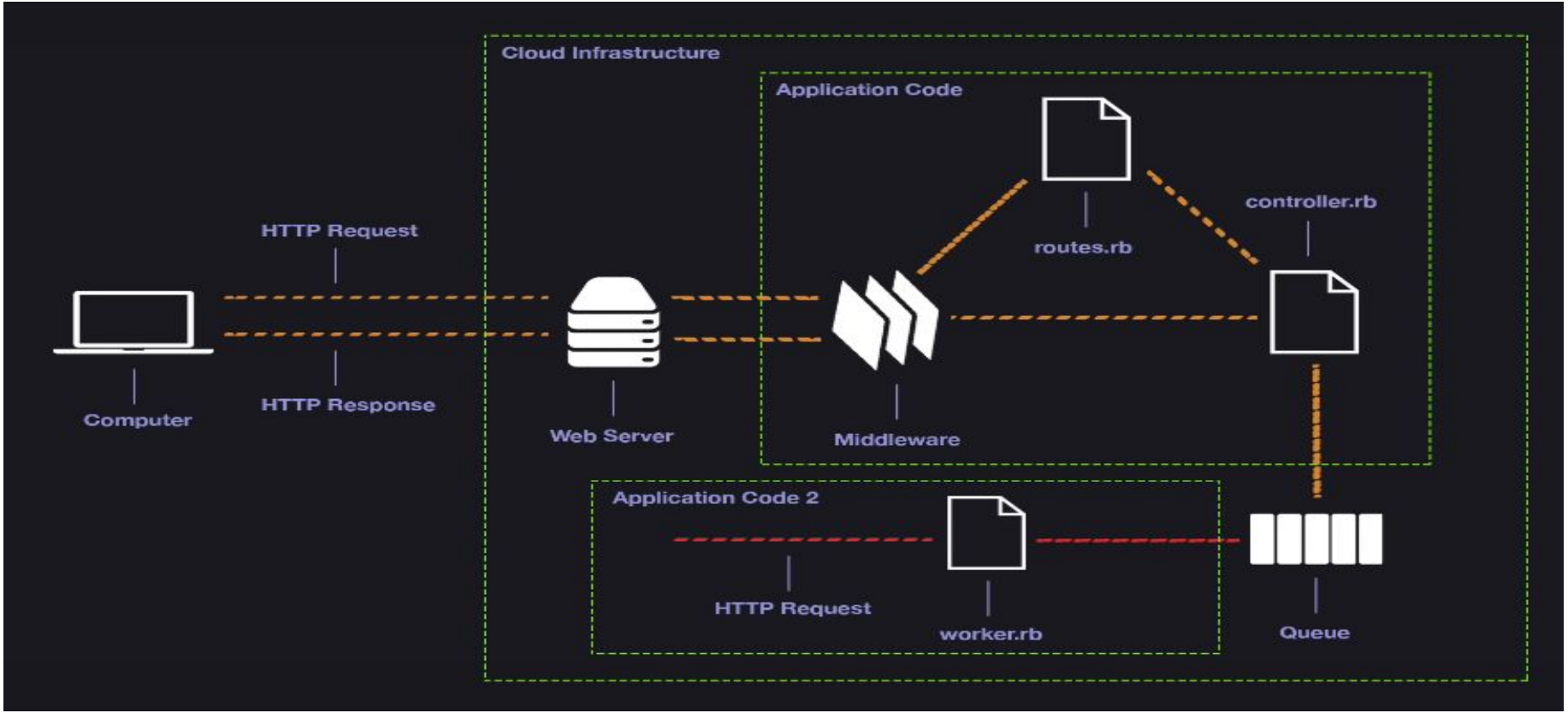
- Se usa para que no se ralentice la carga de la paǵina web, cuando se ejecutan muchos procesos (como muchos correos, sort data).
- Paralelismo, jobs hace que no se hagan procesos de forma lineal.
- Para que sea paralelo, se usan queues.
- Se les pueden añadir horarios para que se ejecute.

https://github.com/CodevivorsG1/Expo

Funcionalidades básicas

- Resuelve el problema de mucha espera por mucho proceso que debe hacer.
- Toma los procesos que debe ejecutar, y los subdivide.
- Cada trabajo subdividido, lo ejecuta en paralelo.
- Muy usado en: big data, reportes, gráficas con muchos datos


Cloud Infrastructure
Application Code
routes.rb
controller.rb
HTTP Request
HTTP Response
Computer
Web Server
Middleware
model.rb
Database


Cloud Infrastructure
Application Code
routes.rb
controller.rb
HTTP Request
HTTP Response
Computer
Web Server
Middleware
Application Code 2
HTTP Request
worker.rb
Queue

https://github.com/CodevivorsG1/Expo

ActiveJob en rails

- Se tienen varias formas de implementar los Jobs, con varias gemas, y no se necesita reescribir jobs si se quieren usar varias gemas a la vez.
- Se usa el generate de rails, facilitando todo el proceso.

    - Delayed Job
    - Sucker Punch
    - Resque
    - Sidekiq

Funciones adicionales de un Job

- Se le puede decir cuando ejecutarse: ahora mismo, en una semana etc.
- Se puede añadir a una cola junto con más jobs (--queue armandoDataset)
- Se puede asociar con “demonios”
- Mejora sustancialmente la calidad de tiempos de carga.

Implementación: ¿Qué se necesita?

- Gema delayed_job_active_record
- Versión de rails 4.2+
- Gema rake para hacer migration (versiones actuales usan rails --comando en vez de rake)
- Si se quieren usar funcionalidades extra propias de DB, usar redis como gestor de DB

¿Qué vamos a crear?

Como se dijo en los usos más comunes, enviar correos es uno de los más básicos.

La idea es crear una cola o queue de jobs, donde cada job es un email que se manda. En programas web sencillos, no es necesario, pero, imaginen el caso de una tienda online como MercadoLibre, con millones de cuentas y enviado de correos por segundo.

https://github.com/CodevivorsG1/Expo

Paso 1: Crear un proyecto nuevo rails

```
$ rails new ejemplo
```

https://github.com/CodevivorsG1/Expo

Paso 2: Añadir al gemfile la gema delayed job

```
#Cola donde se envian los jobs
gem 'delayed_job_active_record'
```

```
/ejemplo$ bundle install --gemfile=Gemfile
```

```
$ bundle install
```

https://github.com/CodevivorsG1/Expo

Paso 3: Generar el delayed job en el proyecto

```
/ejemplo$ rails generate delayed_job:active_record
```

```
Running via Spring preloader in process 22420
      create  bin/delayed_job
       chmod  bin/delayed_job
      create  db/migrate/20180418050314_create_delayed_jobs.rb
```

```
/ejemplo$ rails db:migrate
```

Paso 4: Indicarle a rails la queue donde se mandan los jobs

En config/application.rb :

```
config.active_job.queue_adapter = :delayed_job
```

```
config.action_mailer.default_url_options = { host: 'example.com'}
```

https://github.com/CodevivorsG1/Expo

Paso 5: Crear el scaffold de la tabla de persona para el email

```
/ejemplo$ rails g scaffold person name:string email:string
```

```
/ejemplo$ rails db:migrate
```

https://github.com/CodevivorsG1/Expo

Paso 6: Crear el scaffold del mailer

```
/ejemplo$ rails g mailer welcome notify
```

Paso 7: Añadir los parámetros para envío

En app/mailers/welcome_mailer.rb :

```
class WelcomeMailer < ApplicationMailer
  def notify(person)
    @person = person

    mail to: person.email, subject: "Bienvenido a la web"
  end
end
```

Paso 8: Poner la plantilla html de lo que se envía por correo

En app/views/layout/mailer.text.erb :

```
<html>
    <body>
        <h1>Bienvenido a nuestra web en rails</h1>
        <%= yield %>
    </body>

</html>
```

Paso 9: Modificar el html del mailer

En app/views/layouts/welcome_mailer/notify.html.erb :

```
<p>
  Hola <%= @person.name %>
</p>
```

https://github.com/CodevivorsG1/Expo

Paso 10: Hacer un preview del correo

test/mailers/previews/welcome_mailer_preview.rb:

```ruby
# Preview all emails at http://localhost:3000/rails/mailers/welcome_mailer
class WelcomeMailerPreview < ActionMailer::Preview

  # Preview this email at http://localhost:3000/rails/mailers/welcome_mailer/notify
  def notify
    WelcomeMailer.notify Person.new(name: 'Sample user', email: 'sample@mail.com')
  end

end
```

```
/ejemplo$ rails db:migrate
```

https://github.com/CodevivorsG1/Expo

Paso 11:
Modificar en el controller que envíe cuando crea

```
def create
  @person = Person.new(person_params)

  respond_to do |format|
    if @person.save
      WelcomeMailer.notify(@person).deliver_now
      format.html { redirect_to @person, notice: 'Person was successfully created.' }
      format.json { render :show, status: :created, location: @person }
```

Iniciar rails s, crear un usuario, comprobar en consola el envío

Paso 11: Configurar el smtp , simulación de uno

En config/application.rb:

```
config.action_mailer.delivery_method = :smtp
```

Paso 12: Modificamos el controller para que los jobs (mails) se pongan en cola

En app/controller/people_controller.rb:

```
if @person.save
  WelcomeMailer.notify(@person).deliver_later!
```

Paso 13: Comprobar creando usuarios, y liberación de ejecución de los jobs que están en delayed_jobs

```
$ rails jobs:workoff
```

Consideraciones especiales

Pueden haber problemas con el servidor smtp que trae rails, por lo que se puede usar en su lugar la gema mailcatcher

```
$ gem install mailcatcher
```

Luego se modifica la configuración en config/application.rb

Usando mailcatcher como server smtp

```
    config.action_mailer.smtp_settings = {
      address: 'localhost',
      port: 1025
    }
  end
end
```

Finalmente se puede entrar a localhost:1080 para ver el dashboard del mail catcher y así comprobar el flujo de los correos que se quedan en el queue de delayed_job

https://github.com/CodevivorsG1/Expo

Referencias

Documentación oficial:
http://guides.rubyonrails.org/active_job_basics.html

Gema mailcatcher:
https://mailcatcher.me/

Blog rails:
http://blog.andolasoft.com/2013/04/4-simple-steps-to-implement-delayed-job-in-rails.html

Otro blog rails:
https://blog.codeship.com/how-to-use-rails-active-job/