EL CAMINO DE LA INSTRUMENTACIÓN

Patricio Mac Adden

SINAPTIA

EL INICIO DEL CAMINO

- Apps en mantenimiento
  - Problemas de performance
    - Memory leaks
    - Slow actions
  - Recursos escasos (hardware)
    - Poca memoria
  - Recursos escasos (software)
    - APM free tier
    - No APM

IMPOSIBLE IDENTIFICAR PROBLEMAS

QUÉ OPCIONES HAY?

- Muchos APMs comerciales:
  - NewRelic
  - AppSignal
  - Datadog
  - Airbrake
  - Rollbar
- DIY
  - ActiveSupport::Notifications y ActiveSupport::ErrorReporter
  - OpenTelemetry

PERO ANTES... QUÉ ES INSTRUMENTAR?

Instrumentar es modificar el código de nuestra app de manera que podamos analizarla.

Ejemplos de instrumentación:

- Logging
- Profiling
- Error reporting

Es el paso anterior a la observabilidad (que veremos más adelante).

ActiveSupport::Notifications
y
ActiveSupport::ErrorReporter

ActiveSupport::Notifications

Es la API de instrumentación de Rails.

Tiene dos componentes:

- Instrumenters: notifican la ocurrencia de un evento
- Subscribers: consumen eventos y actúan en consecuencia

```ruby
# Para instrumentar un evento
ActiveSupport::Notifications.instrument("render", extra: :information) do
  render plain: "Foo"
end

# Para consumir un evento
ActiveSupport::Notifications.subscribe("render") do |event|
  event.name          # => "render"
  event.duration      # => 10 (in milliseconds)
  event.payload       # => { extra: :information }
  event.allocations   # => 1826 (objects)
end
```

ActiveSupport::ErrorReporter

Interface para servicios de Error Reporting. Es un mecanismo similar a Notifications.

Tiene como objetivo:

- Simplificar el error-handling code
- Eliminar middleware o monkey patching de terceros para reportar errores

```ruby
# Pasar de esto:
begin
  do_something
rescue SomethingIsBroken => error
  MyErrorReportingService.notify(error)
end

# a esto:
Rails.error.handle(SomethingIsBroken) do
  do_something
end

# Entonces podemos crear un subscriber
class ErrorSubscriber
  def report(error, handled:, severity:, context:, source: nil)
    MyErrorReportingService.report_error(error, context: context, handled: handled, level: severity)
  end
end

# y suscribirlo
Rails.error.subscribe(ErrorSubscriber.new)
```

ESTA BUENO, PERO... SE QUEDA CORTO

- No hay instrumentación por defecto: hay que instrumentar todo manualmente
- Hay “pocos” eventos
- No hay info detallada de los recursos de hardware (mem, cpu, gc, etc)
- No hay métricas de los eventos instrumentados

SI QUEREMOS LOGRAR ALGO COMO CUALQUIER APM, HAY QUE DESARROLLAR MUCHO

OPENTELEMETRY

OPENTELEMETRY

- Es un observability framework, diseñado para recolectar, procesar y exportar telemetry data (signals):
  - Metrics
  - Traces
  - Logs
- Vendor and tool agnostic
  - Open-Source: Jaeger, Zipkin, Prometheus, etc
  - Comerciales: Datadog, NewRelic, etc
- SDKs para muchos lenguajes, frameworks y librerías: zero-code instrumentation
- **Distributed tracing**

CONCEPTOS DE OBSERVABILITY

Observability es entender el sistema desde afuera sin necesariamente conocer el funcionamiento interno ⇒ para eso tiene que estar bien instrumentado.

Logs: lo que ya conocemos. No necesariamente relacionados a un request y generalmente falta información contextual.

Spans: trackea una operación específica. Un trace es un árbol de spans, correspondiente a un request a la app.

Metrics: agregaciones numéricas en un período de tiempo sobre la app o infra.

OPENTELEMETRY Y RUBY

- opentelemetry-sdk: implementación de referencia
  - Collect, analyze and export traces, metrics and logs
- opentelemetry-instrumentation-all: meta-paquete de instrumentación mantenida por la comunidad. Instrumenta:
  - Rails (activerecord, activestorage, actionmailer, activejob, etc)
  - Rake
  - Sinatra
  - delayed_job
  - pg, mysql, redis
  - Y más
- opentelemetry-exporter-otlp: exporta signals usando OpenTelemetry Protocol
  - Jaeger, Zipkin
  - Datadog

DEMO: OPENTELEMETRY + JAEGER + PROMETHEUS

NO ES SUFICIENTE... NI PARA CUALQUIERA

- Sólo guarda métricas de los traces
- Setup complejo

SOLIDTELEMETRY

SOLIDTELEMETRY

- Database-backed OpenTelemetry
- OpenTelemetry + APM integrado ⇒ Droga de entrada a OpenTelemetry
  - Traces
  - Metrics
  - Errors (tipo Rollbar)
  - Performance (tipo AppSignal)
- En vez de exportar OTLP, exporta a ActiveRecord
- Compatible con:
  - Postgres, MySQL, Sqlite3
  - Rails >= 7

DEMO: SOLIDTELEMETRY

Q&A

GRACIAS!