The pillars of integration

Rodrigo Boniatti

Developer at Codeminer 42

@boniattirodrigo

rodrigoboniatti.com

CODE MINER

FREAKING AWESOME

ESTE É NOSSO SOBRENOME

Communication issues

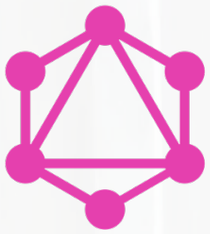
In tech it's not different

104000000 1000123

10400011R0100030 1000012

10400013000001P 011234511

- Ugly design;
- Unsafe;
- Hard to use and test;
- Bad documented;



Ecommerce
GraphQL

Ecommerce
Web Crawler

Ecommerce
XLS

Price comparison
system

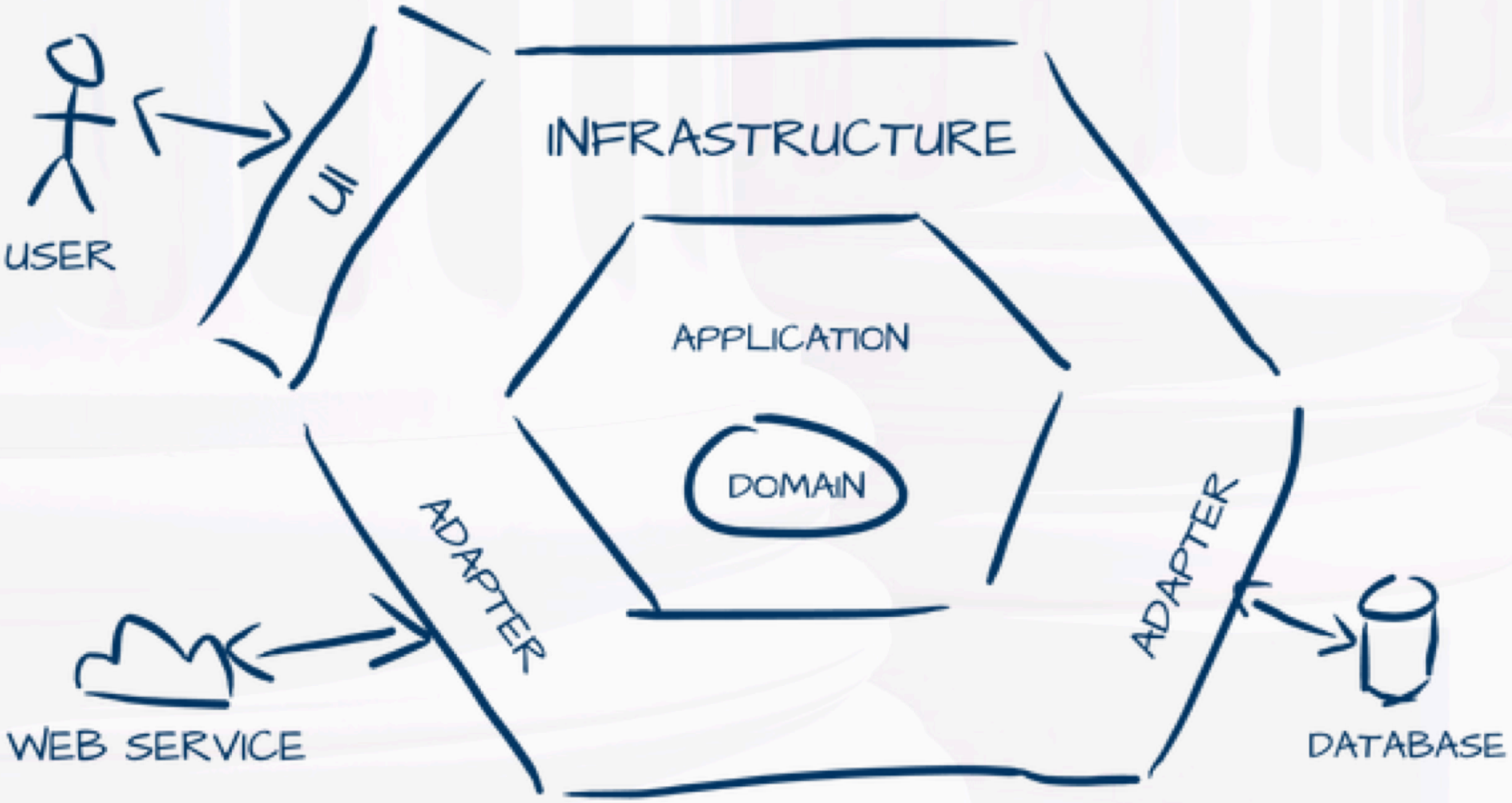
How can we improve our communication?

Architecture


Partner
Application


Partner
Anti-corruption layer
Application

Hexagonal architecture



Design patterns

Adapter

```
// Brazilian supplier
product = {
  nome: 'Sofá 3 lugares',
  preco: 1450,
  quantidade: 1
}
```

```
// German supplier
product = {
  name: 'Stuhl',
  preis: 200,
  menge: 1
}
```

```ruby
class BrazillianProductAdapter < ActiveModel::Serializer
  attributes :name, :price, :quantity

  def name
    object[:nome]
  end

  def price
    object[:preco]
  end

  def quantity
    object[:quantidade]
  end
end
```

```
product = {
   nome: 'Sofá 3 lugares',
   preco: 1450,
   quantidade: 1
}

BrazillianProductAdapter.new(product).as_json

{:name=>"Sofá 3 lugares", :price=>1450, :quantity=>1}
```

UbiquitousLanguage

Martin Fowler

31 October 2006

Ubiquitous Language is the term Eric Evans uses in Domain Driven Design for the practice of building up a common, rigorous language between developers and users. This language should be based on the Domain Model used in the software - hence the need for it to be rigorous, since software doesn't cope well with ambiguity.

Evans makes clear that using the ubiquitous language between in conversations with domain experts is an important part of testing it, and hence the domain model. He also stresses that the language (and model) should evolve as the team's understanding of the domain grows.

Strategy

```ruby
class Product < ApplicationRecord
  validates :name, :price, :quantity, :partner, presence: true
end
```

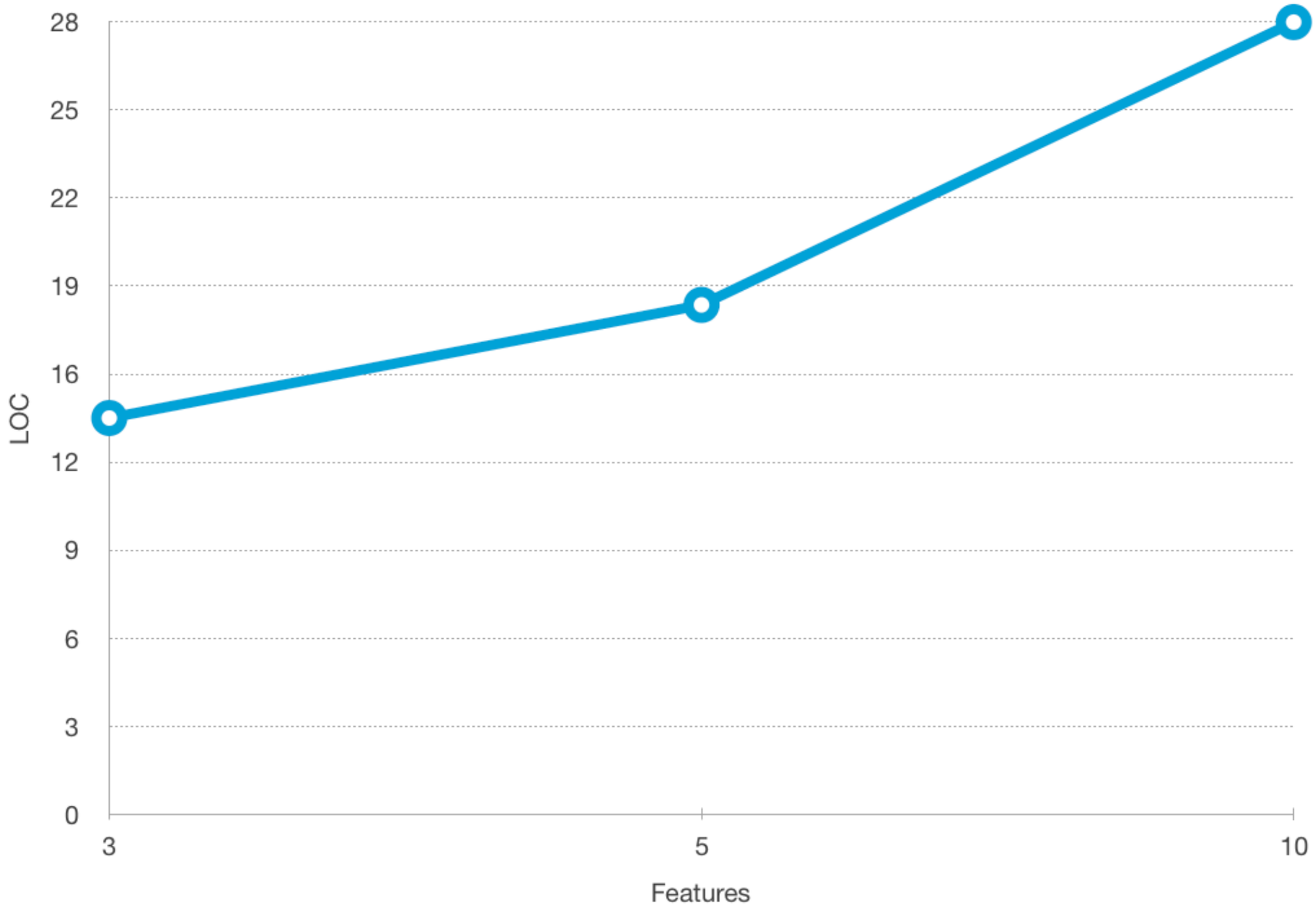
```ruby
class EcommerceA::TrackProductAccess
  def self.call(id)
  end
end

class EcommerceB::TrackProductAccess
  def self.call(id)
  end
end

class EcommerceC::TrackProductAccess
  def self.call(id)
  end
end
```

Smell

```
class TrackProductAccess
  def self.call(product)
    case product.partner
    when 'EcommerceA'
      EcommerceA::TrackProductAccess.call(product.id)
    when 'EcommerceB'
      EcommerceB::TrackProductAccess.call(product.id)
    when 'EcommerceC'
      EcommerceC::TrackProductAccess.call(product.id)
    else
      puts 'Partner not found'
    end
  end
end
```


28
25
22
19
16
12
9
6
3
0
LOC
3
5
10
Features

Refactored

```
class TrackProductAccess
  class << self
    def call(partner, product_id)
      partner_module = partner.constantize
      partner_module::TrackProductAccess.call(product_id)
    end
  end
end
```

Refactored

Dependency Injection

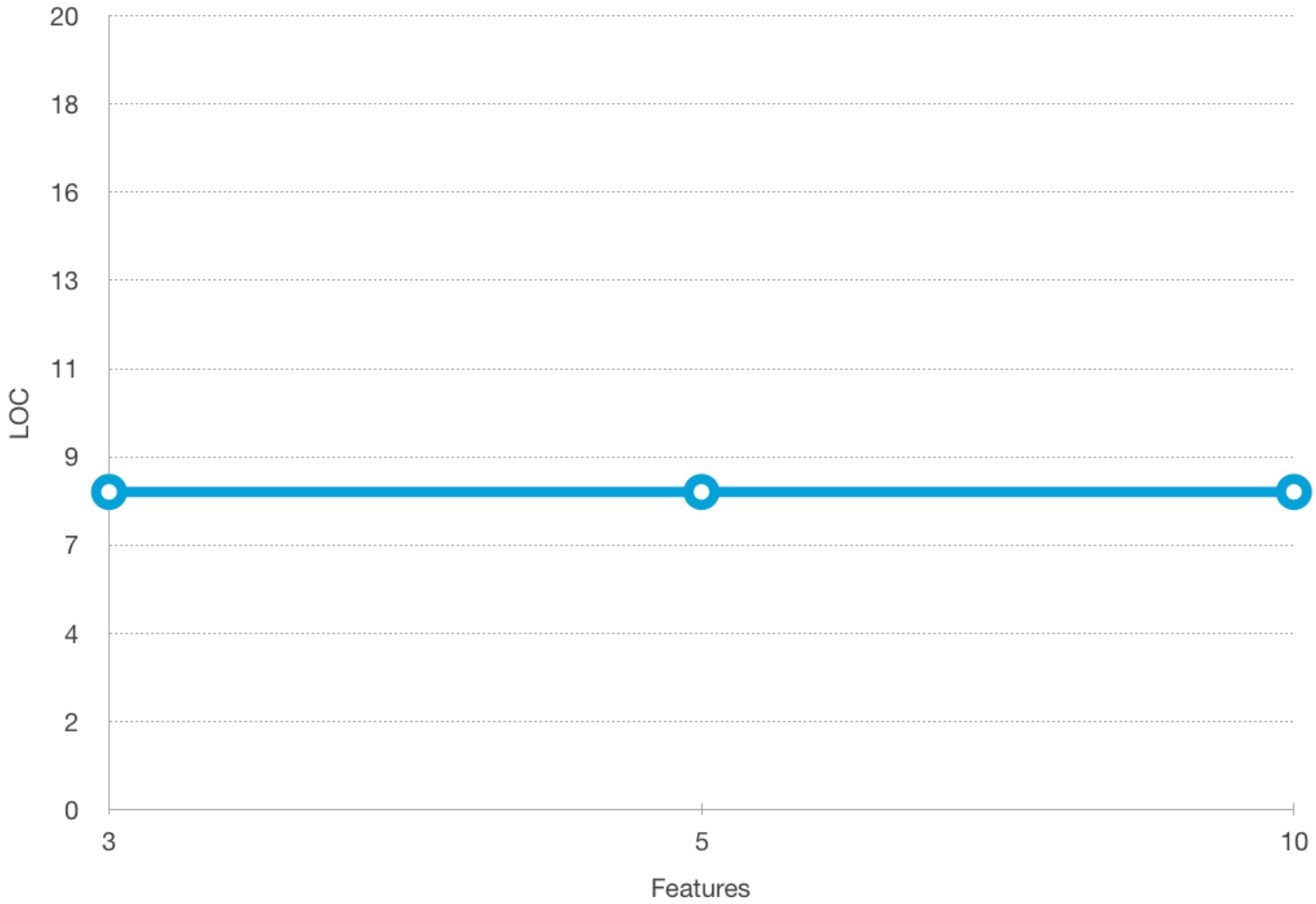
```
class TrackProductAccess
  class << self
    def call(partner, product_id)
      partner_module = partner.constantize
      partner_module::TrackProductAccess.call(product_id)
    end
  end
end
```

Refactored

Dependency Injection

```
class TrackProductAccess
  class << self
    def call(partner, product_id)
      partner_module = partner.constantize
      partner_module::TrackProductAccess.call(product_id)
    end
  end
end
```

Inversion of Control


20
18
16
13
11
9
7
4
2
0
LOC
3
5
10
Features

Partners configuration could be saved in a YML file

```ruby
module Ecommerce
  class TrackProductAccess
    def self.call(product_id)
      self.new(product_id).track
    end

    def track
      EcommerceHttpClient.post do |req|
        req.url "/track_user/#{@product_id}"
        req.body = configurations
      end
    end

    private

    def initialize(product_id)
      @product_id = product_id
    end

    def configurations
      {
        version: 3,
        port: 5500,
        security: true,
        anonymous_user: false,
        cors_enable: true
      }
    end
  end
end
```

Better

```
class TrackProductAccess
  def self.call(product_id, config = YAML.load_file('ecommerce_config.yml'))
    self.new(product_id, config).track
  end

  def track
    EcommerceHttpClient.post do |req|
      req.url "/track_user/#{@product_id}"
      req.body = @config
    end
  end

  private

  def initialize(product_id, config)
    @product_id = product_id
    @config = config
  end
end
```

Avoid hardcoded partners name

Smell

```
class Order < ApplicationRecord
  validates :stripe_status, presence: true
  validates :aws_s3_invoice_url, format: { with: URI.regexp }
end
```

Better

```
class Order < ApplicationRecord
  validates :payment_status, presence: true
  validates :invoice_url, format: { with: URI.regexp }
end
```

Message broker

Sidekiq

- every time the Sidekiq process crashes, any messages being processed are lost. You can avoid this with Sidekiq Pro's reliable fetch feature.

https://github.com/mperham/sidekiq/wiki/Problems-and-Troubleshooting#my-sidekiq-process-is-crashing-what-do-i-do


RabbitMQ

Advantages

- Persistence;
- Fault Resilience;
- Delivery acknowledgements;
- AMQP protocol;

Error tracking


Rollbar

Top 10 items in last 24 hours

view all

24hr Trend	Count		Title
	959	1,257	#3 Hull breach!
	985	696	#315 NoMethodError: undefined method `[]' for nil:NilClass
	10	2	#1132 KeyError: 'birthday'
	7	1,111	#6 some log message
	4,837	989	#316 ActionController::RoutingError: No route matches [GET] "/qffavsetvv"
	949	1,267	#317 NameError: undefined local variable or method `this_variable_has_not_been_set'
	4	980	#311 Uncaught ReferenceError: nonexistent_function is not defined
	1	1	#1134 another log message for people test test
	1	103	#338 ReferenceError: Can't find variable: nonexistent_function
	1	1	#1133 log message for warning

Top 5 active items new/reactivated last day

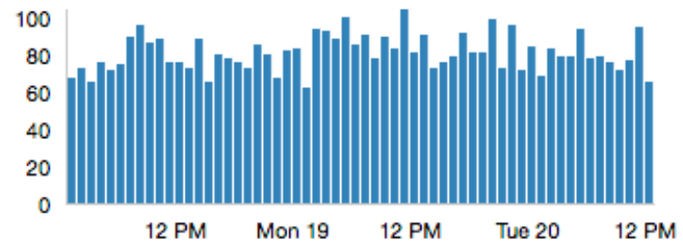
24hr Trend	Count		Title
	10	2	#1132 KeyError: 'birthday'
	1	1	#1134 another log message for people test test
	1	1	#1133 log message for warning
	1	1	#1130 some log messagep
	6	1	#1131 khjkjhk

Top 5 active items new/reactivated last week

7day Trend	Count		Title
	11,417	1,257	#3 Hull breach!
	10	2	#1132 KeyError: 'birthday'
	6	2	#1125 UnboundLocalError: local variable 'status' referenced before assignment
	3	1	#1129 TypeError: 'NoneType' object has no attribute '__getitem__'
	1	1	#1124 errors

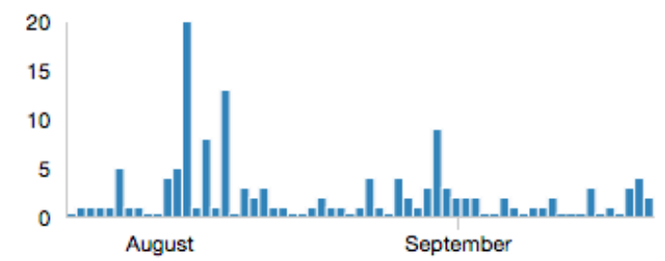
Any Environment

Hourly Error/Critical Occurrences



Daily Error/Critical Occurrences

Daily New/Reactivated Items



Account Usage

3,025,623 occurrences used (unlimited during trial).
Your free trial lasts until January 18, 2038. · See details

API Rejects

0 in the last 24 hours. View all.

Specs

VCR

- Cache your partners' response;
- Know when responses have changed;
- Easy mock.

What should we test in our partners?

GraphQL

GraphQL

- Avoid chaining requests;
- Documentation;
- Single request for many resources.

Webhooks

Webhooks

- Just call your API when something occurs;
- Easy to setup;
- Stop requests every X minutes.

What should we test in our partners?

- API Rate Limit;
- Documentation;
- GraphQL;
- Webhooks.

Have some response backups

Your software dictates the rules

OPEN CLOSED PRINCIPLE

Open Chest Surgery Is Not Needed When Putting On A Coat

Great developers think how others will use their tools

Questions?

Thanks ;)