



10 najgorszych
rad dla developera
Ruby i Ruby on Rails





1



Moja rada:



Jacek

Moja rada:

PHP > Ruby



Jacek

Moja rada:

PHP > Ruby



Jacek



Moja rada:

Żartowałem



Jacek



Jacek Czarnecki

Rozwijam zespół IT w inFakt

- lubię niezdrowe przekąski
- lubię jeździć motocyklem
- mam kota

Budujemy **inFakt** prawie
10 lat



Jacek

Budujemy **inFakt** prawie
10 lat i zdążyliśmy
popełnić **dużo** błędów



Jacek

Padło pytanie:

Hej ! Developerzy !



Jacek

Padło pytanie:

Może pochwaliemy się
swoimi błędami?



Jacek

Padło pytanie:

Jakich najgorszych rad
możemy udzielić, żeby inni
developerzy popełniali
podobne błędy?



Jacek

Zebrałiśmy 40 „dobrych rad”, z których na dzisiaj wybraliśmy 10



Jacek



10 najgorszych
rad dla developera
Ruby i Ruby on Rails

1





Adrian Zięba

Ruby Developer

- fan dobrej kawy
- tata Zosi

Moja rada:



Adrian

Moja rada:

Wzoruj się na
istniejącym projekcie



Adrian

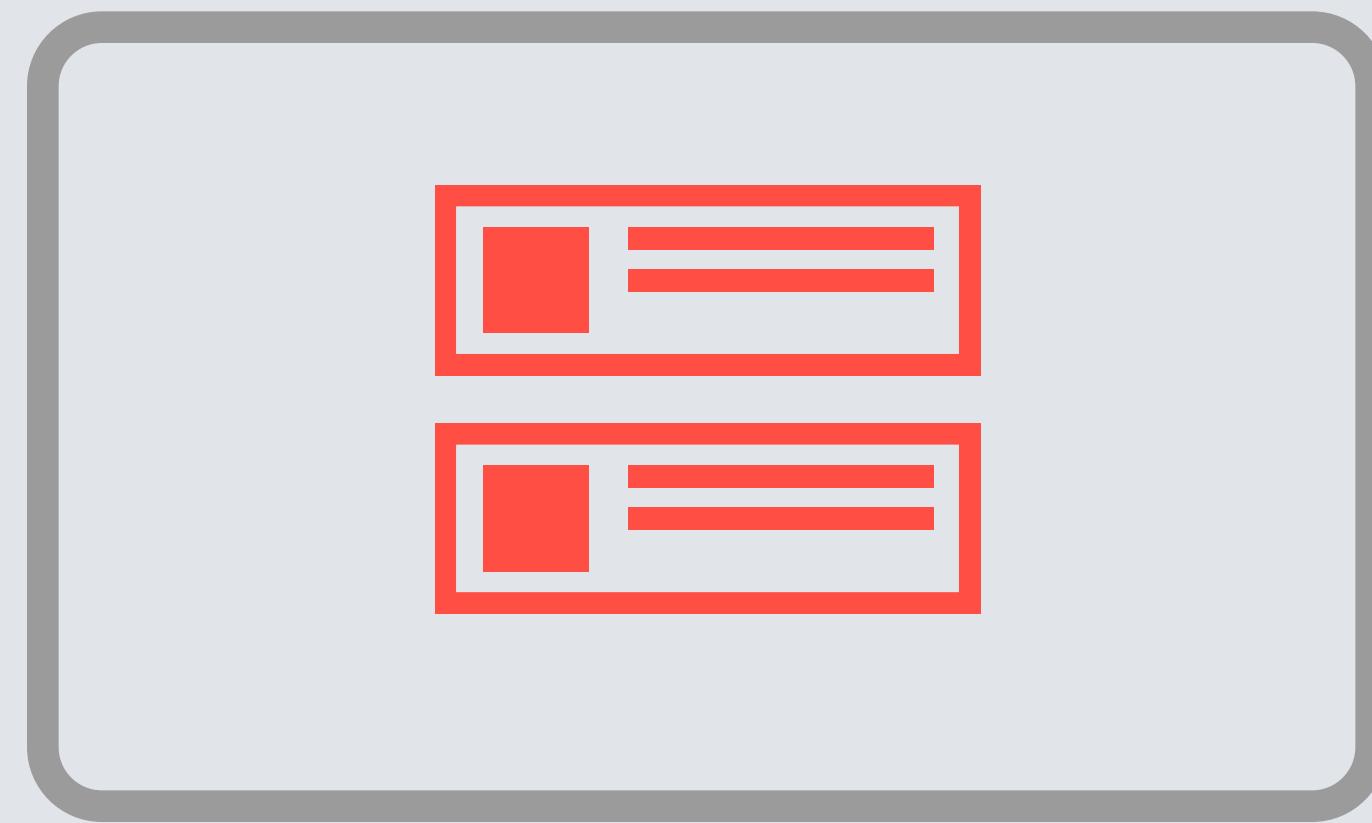
Moja rada: Wzoruj się na istniejącym projekcie

Będzie szybciej - copy / pase -
i gotowe!



Adrian

Moja rada: Wzoruj się na istniejącym projekcie

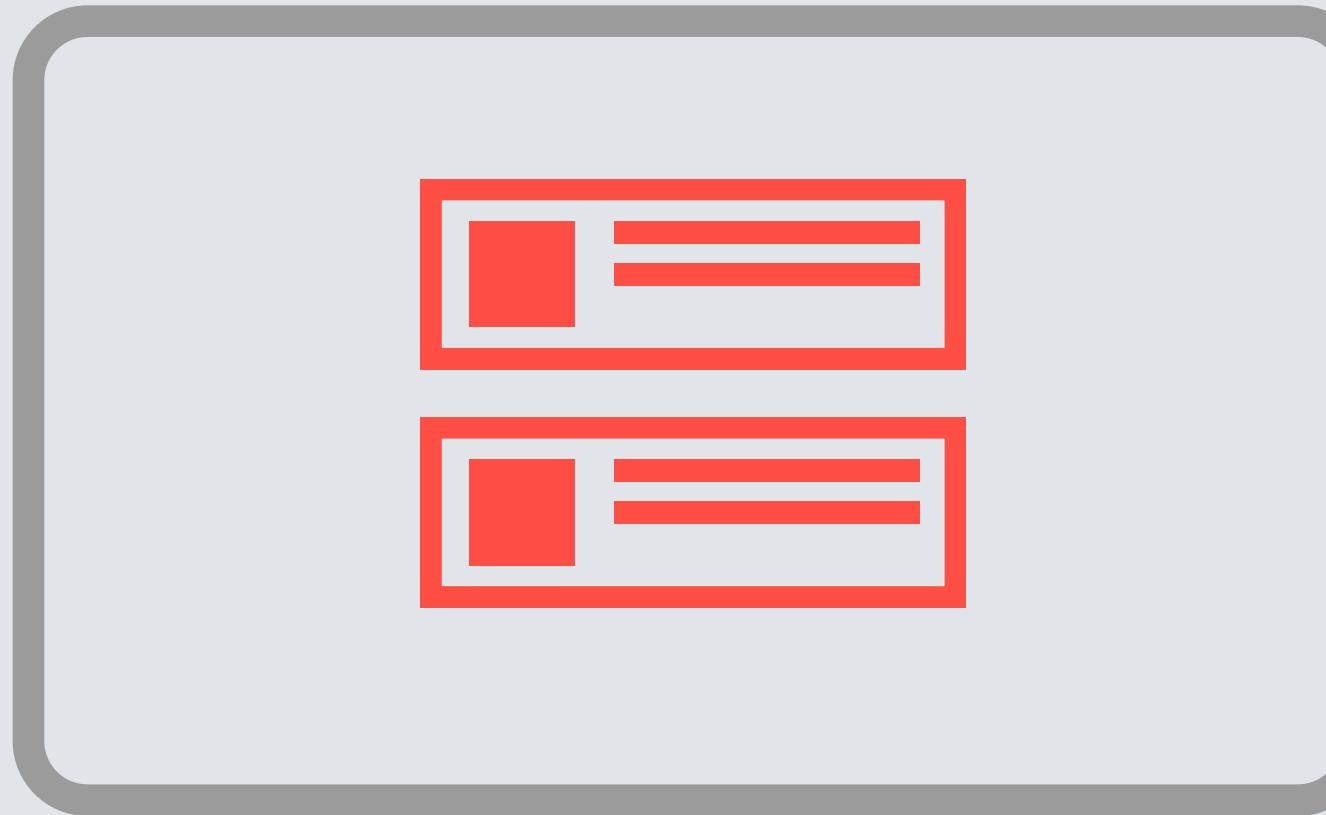


Wysyłka push

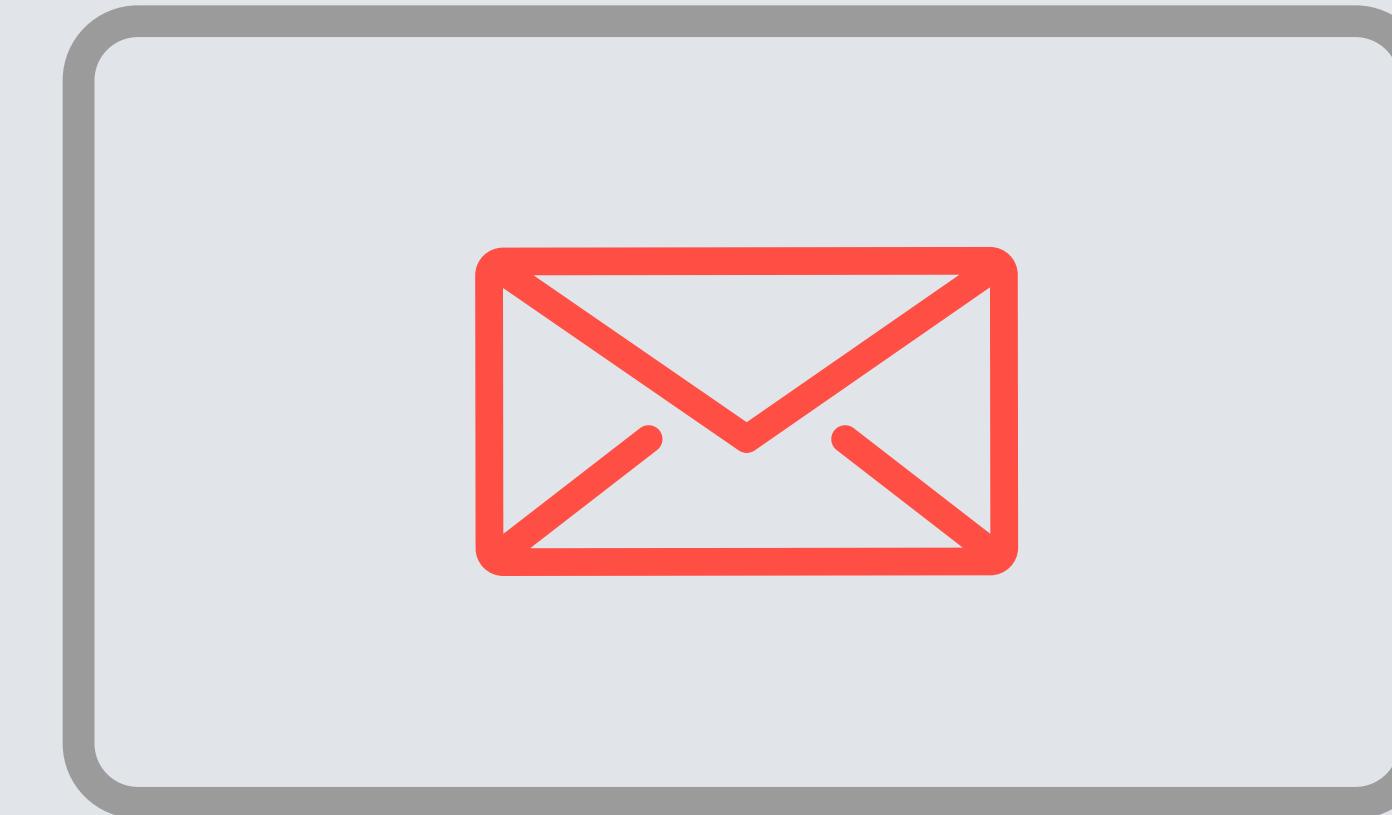


Adrian

Moja rada: Wzoruj się na istniejącym projekcie



Wysyłka push



Wysyłka e-mail



Adrian

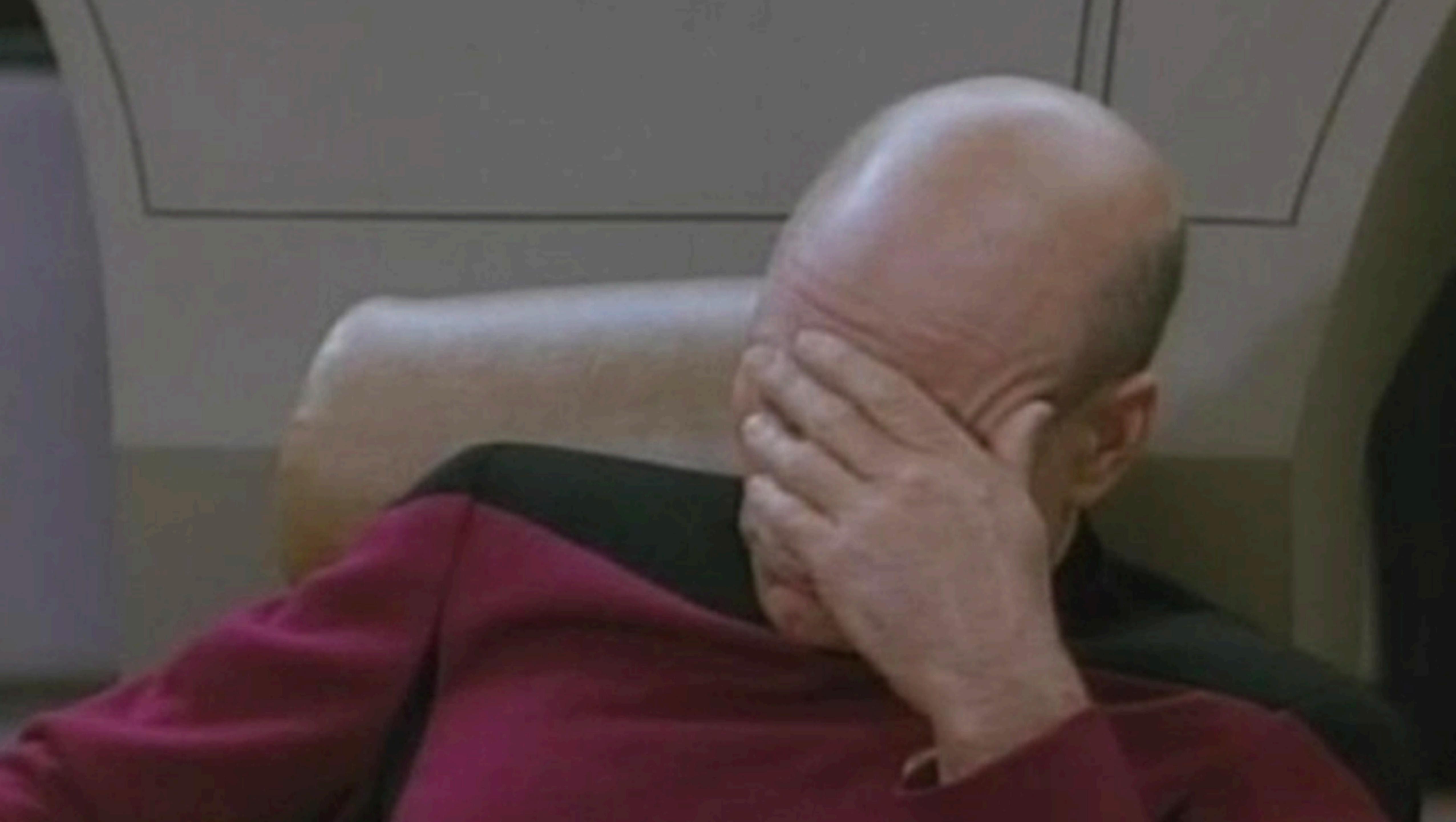
Moja rada: Wzoruj się na istniejącym projekcie



Lecimy na produkcję



Adrian



Moja rada: Wzoruj się na istniejącym projekcie

Aplikacja działa wolno



Adrian

Moja rada: Wzoruj się na istniejącym projekcie

Aplikacja działa bardzo
wolno



Adrian

Moja rada: Wzoruj się na istniejącym projekcie

NGNIX



Adrian

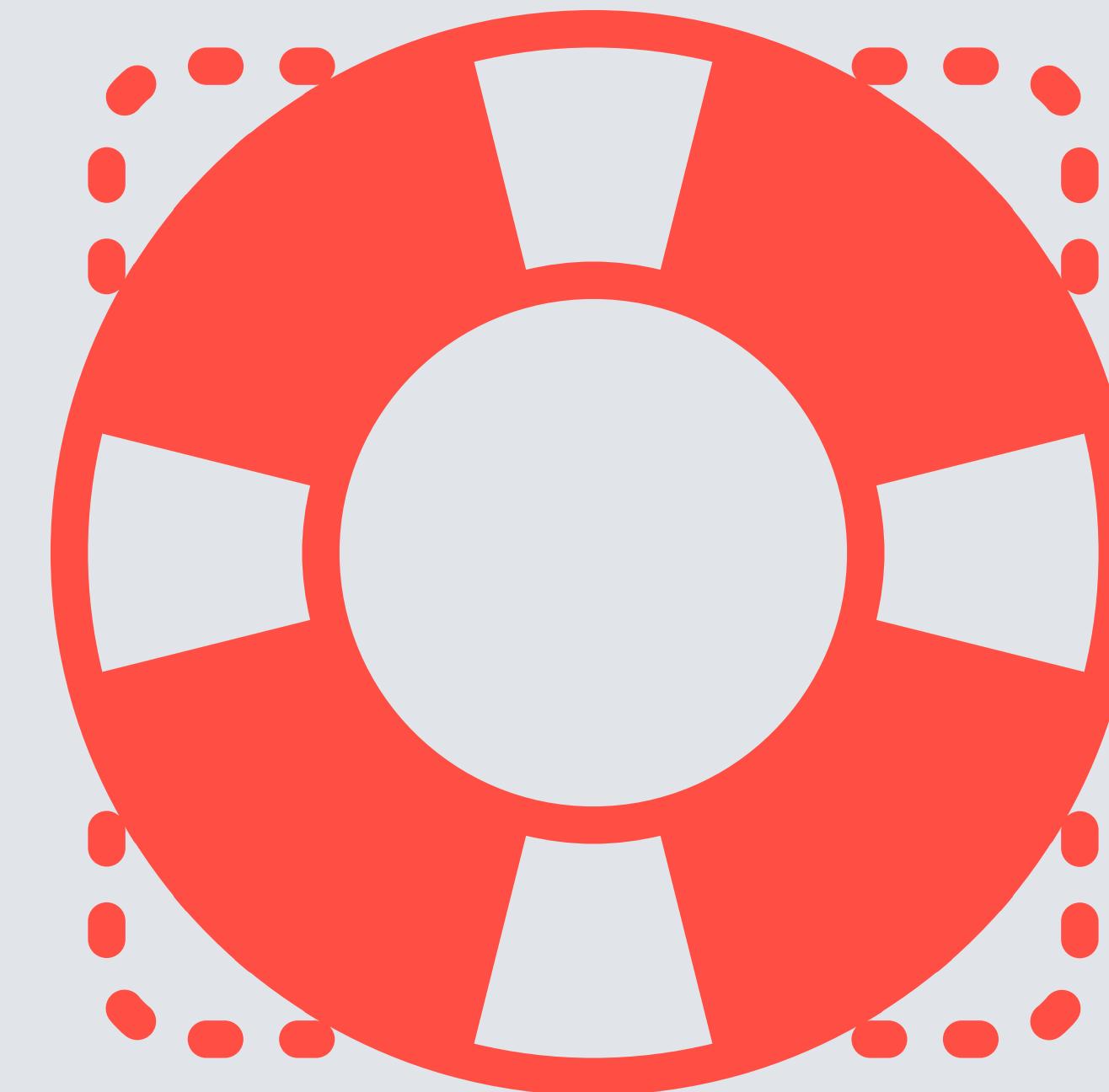
Moja rada: Wzoruj się na istniejącym projekcie

Restart,
działa ok,
działa wolno,
działa bardzo wolno ...



Adrian

Moja rada: Wzoruj się na istniejącym projekcie



Adrian

Moja rada: Wzoruj się na istniejącym projekcie

Dołożmy RAM i zobaczymy,
co się stało



Adrian

Moja rada: Wzoruj się na istniejącym projekcie

Rails



Adrian

Moja rada: Wzoruj się na istniejącym projekcie

Rails

Sidekiq



Adrian

Moja rada: Wzoruj się na istniejącym projekcie

Rails Sidekiq 2 workery



Adrian

Moja rada: Wzoruj się na istniejącym projekcie

Rails (okrajamy do API-mode)

Sidekiq
2 workery



Adrian

Moja rada: Wzoruj się na istniejącym projekcie

Rails (okrajamy do API-mode)

Sidekiq (okrajamy wątki)

2 workery



Adrian

Moja rada: Wzoruj się na istniejącym projekcie

~~Rails~~ (okrajamy do API-mode)

~~Sidekiq~~ (okrajamy wątki)

~~2 workers~~ (wywalamy)



Adrian

Moja lekcja:

Zacznij od czegoś małego z minimalną ilością usług, które są potrzebne. Rozbuduj, gdy pojawią się nowe.



Adrian

2





Sebastian Bobrowski

Ruby Developer



- biega, jeździ na rowerze, pływa
- zdrowo się odżywia
- ojciec dwóch córek

Moja rada:



Sebastian

Moja rada:

Nie trzymaj stałych
elementów w configach.



Sebastian

Moja rada: Nie trzymaj stałych elementów w configach

Bo łatwiej jest trzymać
bezpośrednio w kodzie aplikacji



Sebastian

Moja rada: Nie trzymaj stałych elementów w configach



Oferta



Sebastian

Moja rada: Nie trzymaj stałych elementów w configach



Faktura
Proforma

Oferta



Faktura
VAT

PIT / VAT / Bank



Sebastian

Moja rada: Nie trzymaj stałych elementów w configach

WIELKIE PROMO!



Sebastian



Moja rada: Nie trzymaj stałych elementów w configach



Sebastian



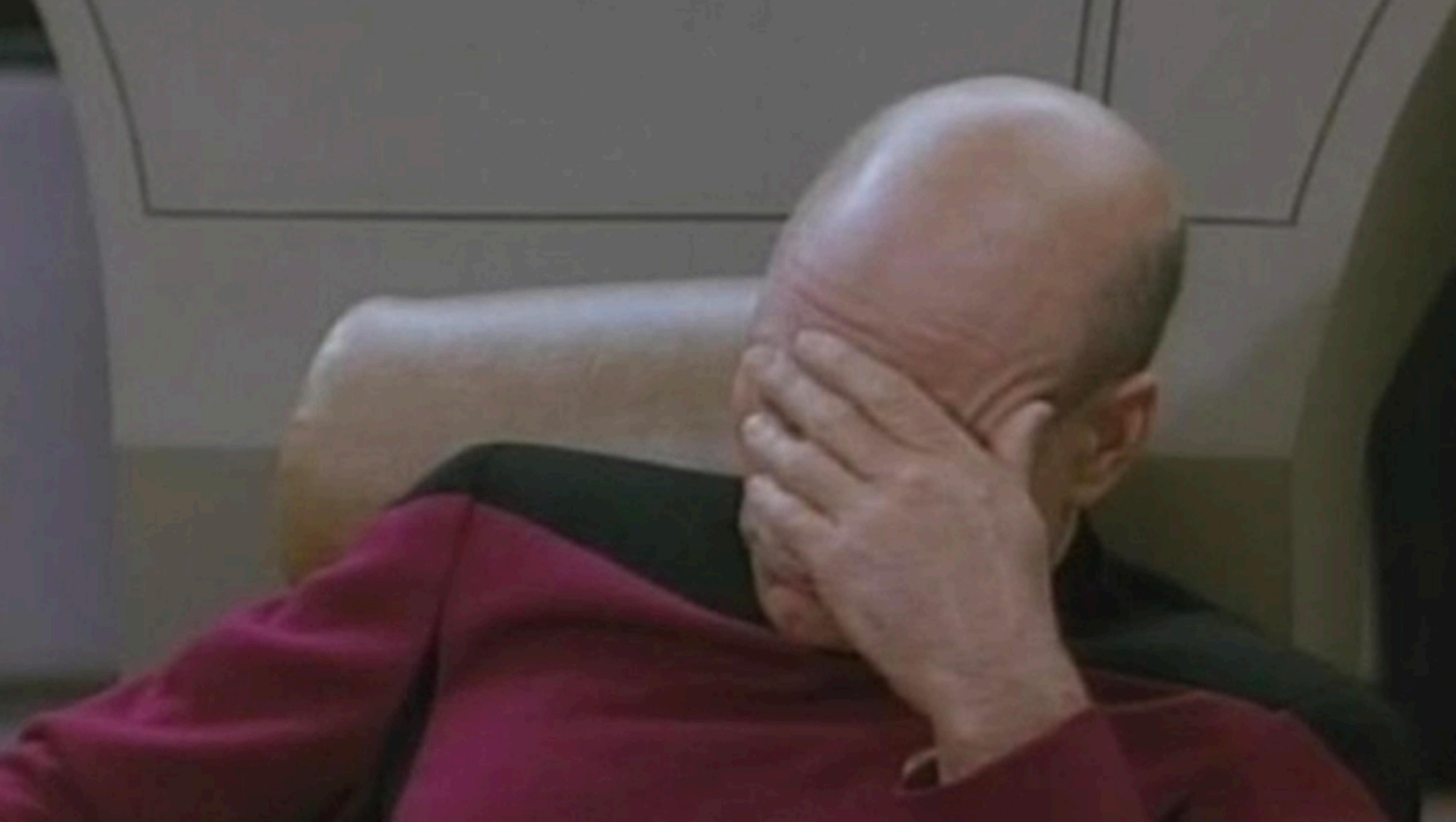
Moja rada: Nie trzymaj stałych elementów w configach



Lecimy na produkcję



Sebastian



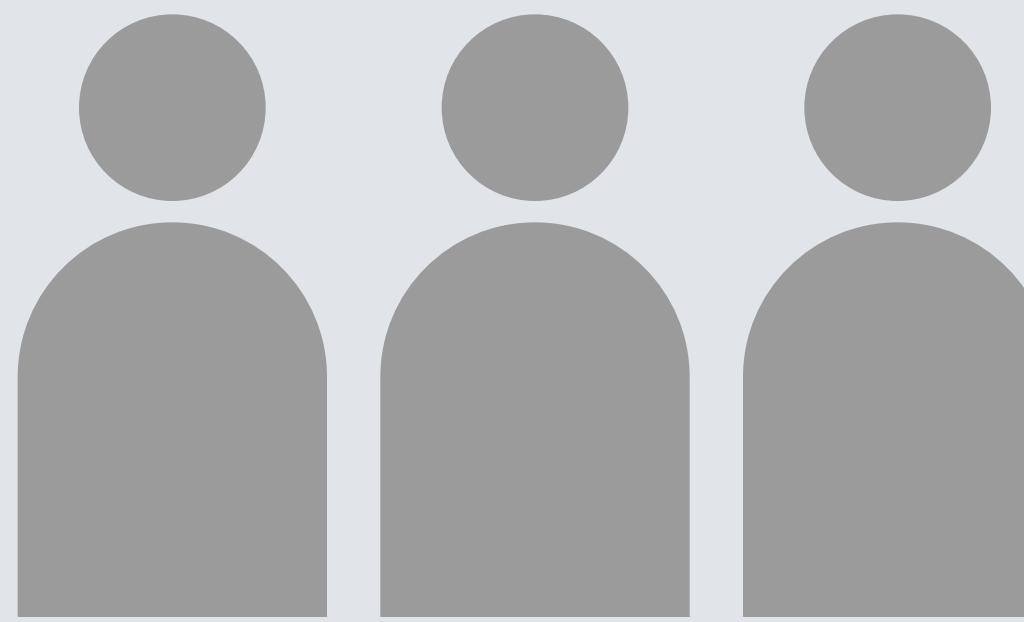
Moja rada: Nie trzymaj stałych elementów w configach

Kilka tysięcy klientów
otrzymało Fakturę VAT
zamiast Proformę.



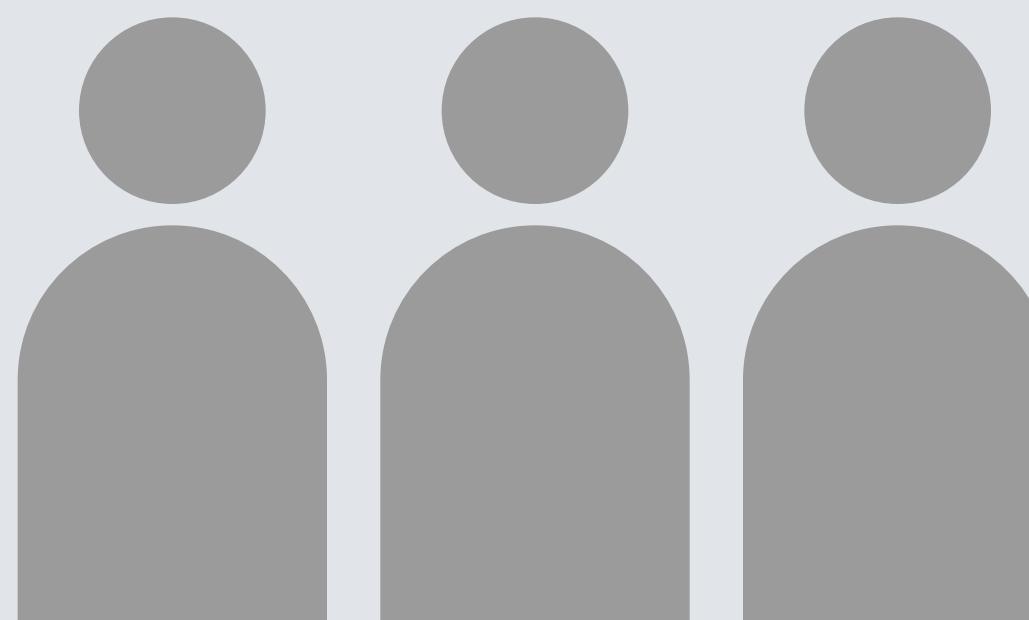
Sebastian

Moja rada: Nie trzymaj stałych elementów w configach



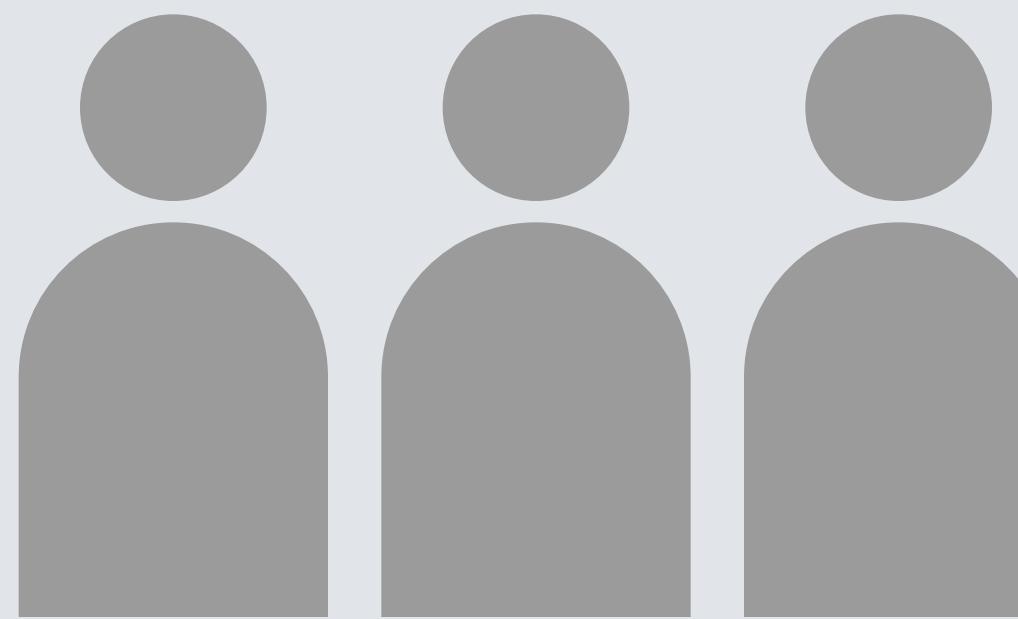
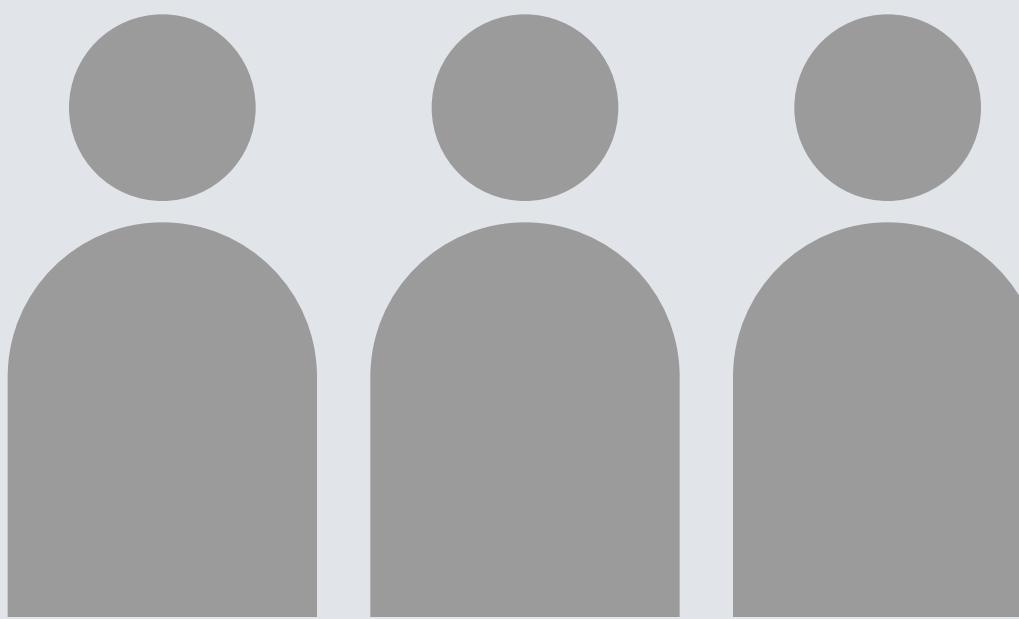
Sebastian

Moja rada: Nie trzymaj stałych elementów w configach



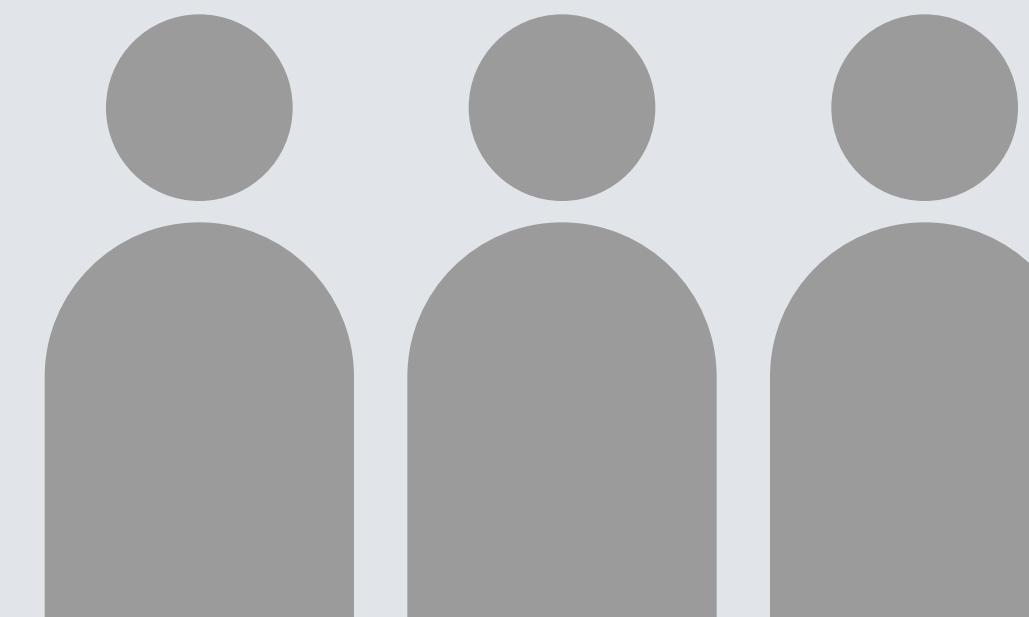
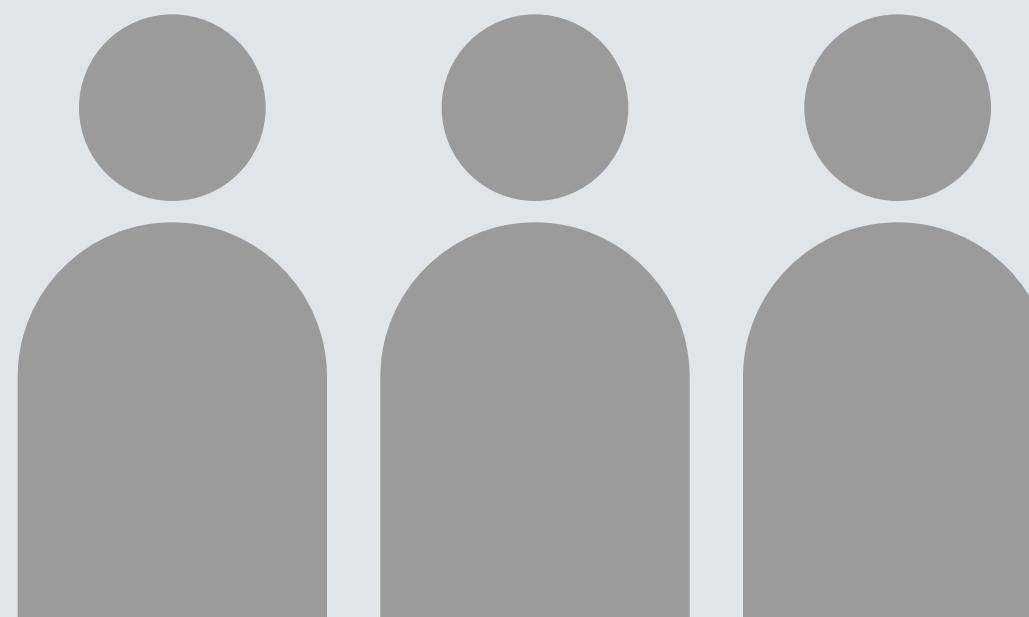
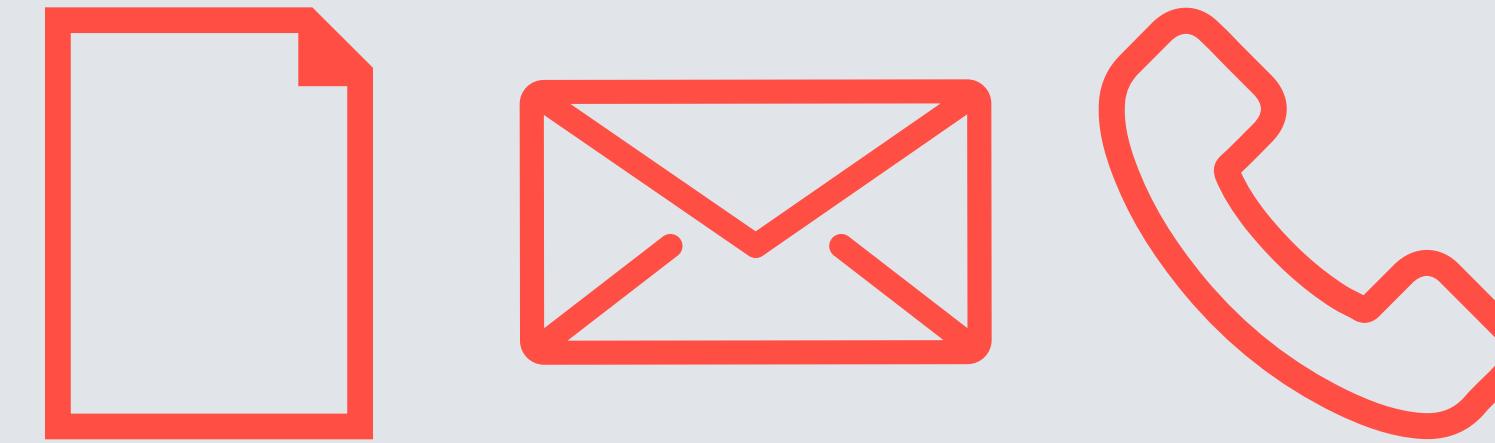
Sebastian

Moja rada: Nie trzymaj stałych elementów w configach



Sebastian

Moja rada: Nie trzymaj stałych elementów w configach



Sebastian

Moja lekcja:

Zawsze stałe, ważne parametry przesuwaj do configa, tak aby kluczowy parametr był zmieniany w jednym, a nie wielu miejscach



Sebastian

3





Alicja Cyganiewicz

Ruby Developer

- jeździ rowerem
- mówi po francusku
- ma śmiesznego psa

Moja rada:



Alicja

Moja rada:

Po prostu wepnij gem,
on to ogarnie!



Alicja

Moja rada: Po prostu wepnij gem, on to ogarnie!

Ktoś już rozwiązał jakiś problem,
zjadł na tym zęby - nie ma sensu,
żebyś to robił drugi raz.



Alicja

Moja rada: Weź ten gem i się nie przejmuj!

Szybkie płatności



Alicja

Moja rada: Weź ten gem i się nie przejmuj!

Szybkie płatności

Rails



Alicja

Moja rada: Weź ten gem i się nie przejmuj!

Szybkie płatności



Rails



Alicja

Moja rada: Weź ten gem i się nie przejmuj!

Szybkie płatności

Roda



Rails



Alicja

Moja rada: Weź ten gem i się nie przejmuj!

Szybkie płatności

XML

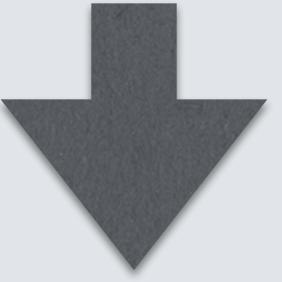


Alicja

Moja rada: Weź ten gem i się nie przejmuj!

Szybkie płatności

XML



Hash



Alicja

Moja rada: Weź ten gem i się nie przejmuj!

Szybkie płatności



XML to Hash Ruby



Alicja

Moja rada: Weź ten gem i się nie przejmuj!

Szybkie płatności



XML to Hash Ruby pure



Alicja

Moja rada: Weź ten gem i się nie przejmuj!

Szybkie płatności



XML to Hash Ruby pure no f*cking Rails



Alicja

Moja rada: Weź ten gem i się nie przejmuj!

Szybkie płatności



XML to Hash Ruby pure no f*cking Rails



Alicja

gem 'activesupport'

Hash.from_xml(...).deep_symbolize_keys

Moja rada: Weź ten gem i się nie przejmuj!

Szybkie płatności

Roda

Active Support



Rails



Alicja

Moja rada: Weź ten gem i się nie przejmuj!

Szybkie płatności

Roda



Rails



Alicja

Active Support

Moja rada: Weź ten gem i się nie przejmuj!

Szybkie płatności

Roda

inny gem (*Nori*)



Rails



Active Support



Alicja

Moja rada: Weź ten gem i się nie przejmuj!

Szybkie płatności

Roda

inny gem (*Nori*)

Shoulda Matchers



Rails



Active Support



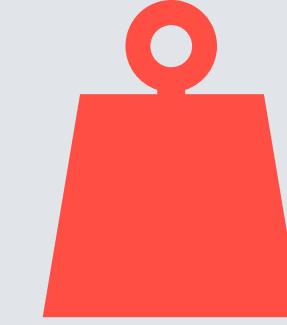
Alicja

Moja rada: Weź ten gem i się nie przejmuj!

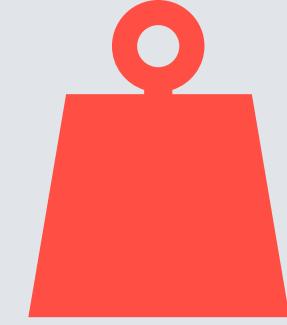
Szybkie płatności

Roda

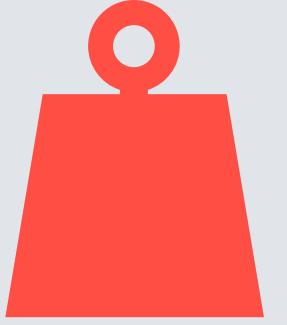
inny gem (Nori)



Rails



Active Support



Shoulda Matchers



Alicja

Moja rada: Weź ten gem i się nie przejmuj!

Szybkie płatności

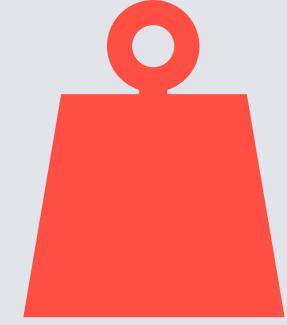
Roda

inny gem (Nori)

-



Rails



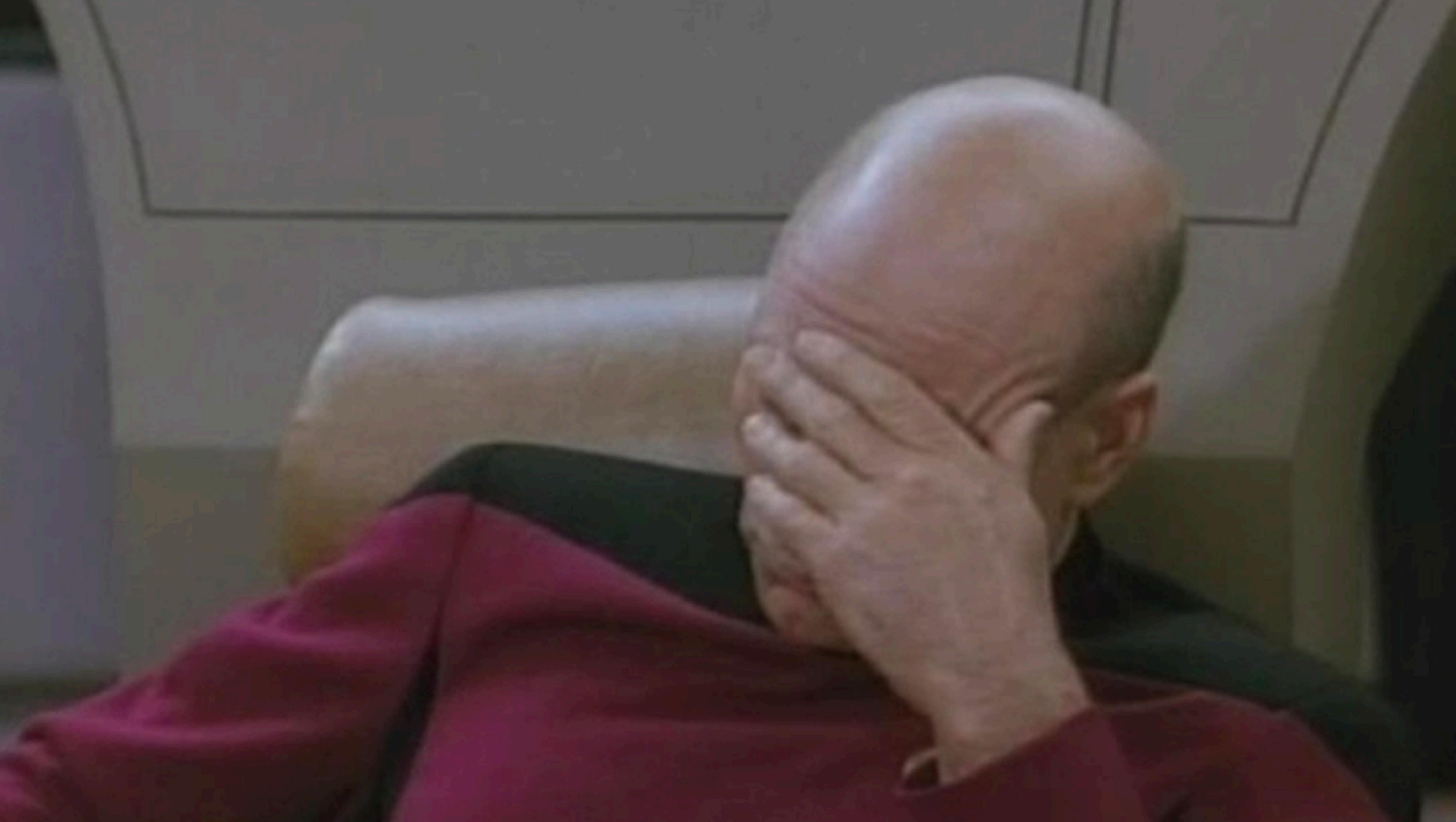
Active Support



Shoulda Matchers



Alicja



Moja rada: Weź ten gem i się nie przejmuj!

Apka zaczęła padać bo Shoulda
Matchers miał *dependency* do Active
Support...



Alicja

Moja rada: Weź ten gem i się nie przejmuj!

Apka zaczęła padać bo Shoulda
Matchers miał *dependecy* do Active
Support...

i chociaż w Gemfile'u nie było go widać - tak
naprawdę był wpiepty i w gruncie rzeczy... nadal z
niego korzystaliśmy



Alicja

Moja rada: Weź ten gem i się nie przejmuj!



Alicja

Moja lekcja:

Zajrzyj gemowi pod maskę!



Alicja

Moja lekcja:

Zajrzyj gemowi pod maskę!

Jak jest ciężki, jakie ma dependency



Alicja

Moja lekcja:

Zajrzyj gemowi pod maskę!

Jak jest ciężki, jakie ma dependency

Jak gem jest napisany, bo może się okazać, że wygoda użytkowania może nieść za sobą ogromny spadek wydajności



Alicja

Moja lekcja:

Zajrzyj gemowi pod maskę!

Jak jest ciężki, jakie ma dependency

Jak gem jest napisany, bo może się okazać, że wygoda użytkowania może nieść za sobą ogromny spadek wydajności



Alicja

Responsible gem collector -
Adama Niedzielski

4





Radek Rochmalski

Ruby Developer

- potrafi gotować
- miłośnik motoryzacji
- 29 lat doświadczenia

Moja rada:



Radek

Moja rada:

Używaj kodu Ruby
w widokach bez
ograniczeń!



Radek

Moja rada: Używaj kodu Ruby w widokach bez ograniczeń!

Jest to wygodny i szybki sposób developmentu.

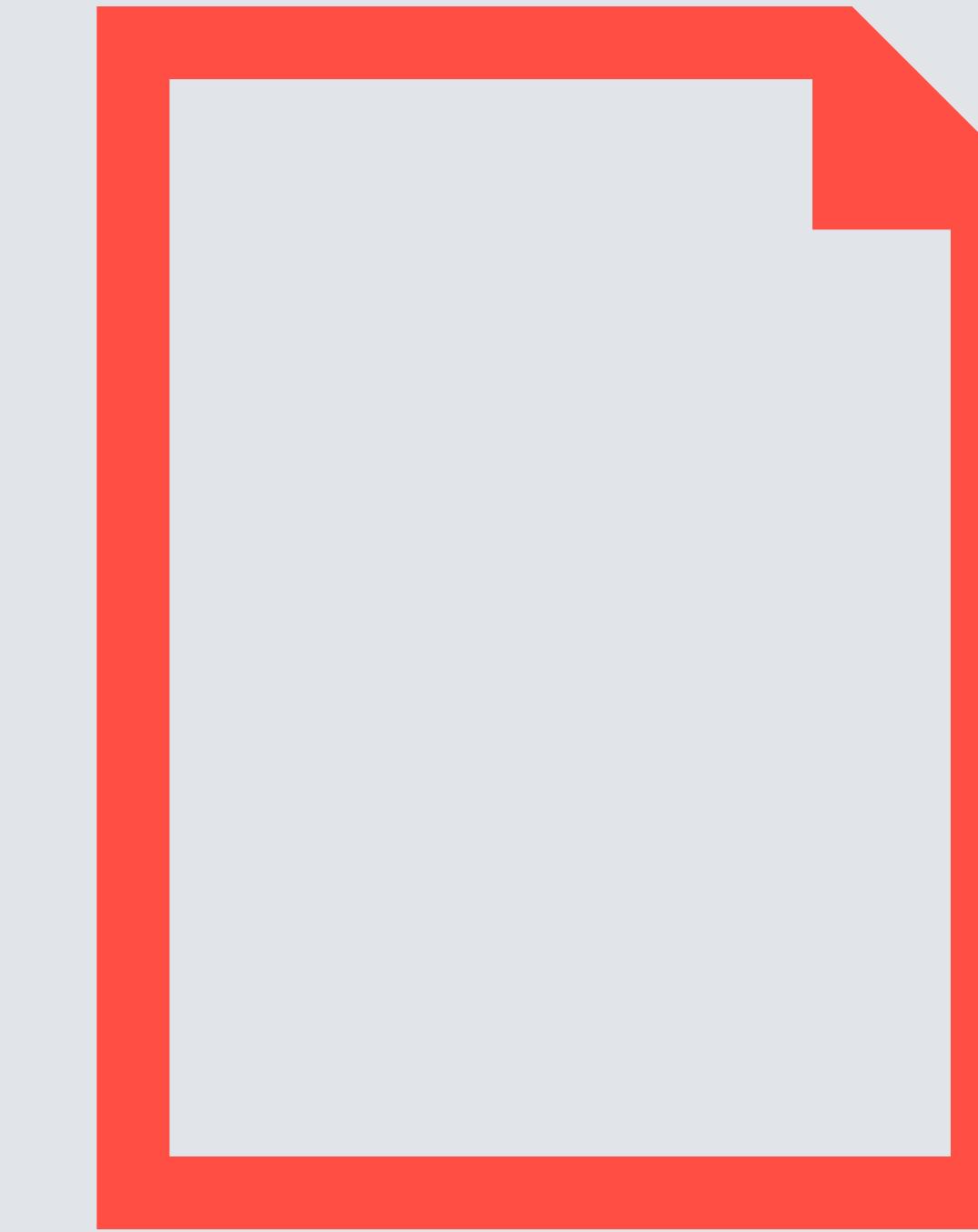


Radek

Moja rada: Używaj kodu Ruby w widokach bez ograniczeń!



Radek



Podgląd faktury

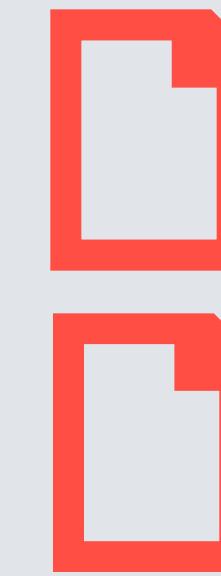
Moja rada: Używaj kodu Ruby w widokach bez ograniczeń!



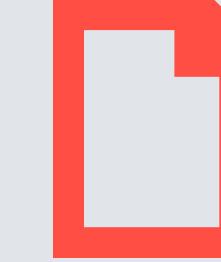
Radek



Podgląd faktury



Faktura VAT



Proforma

Moja rada: Używaj kodu Ruby w widokach bez ograniczeń!

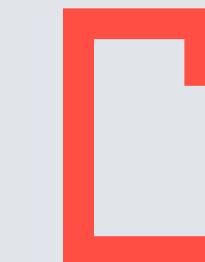


Radek

Podgląd faktury



Faktura VAT



Proforma



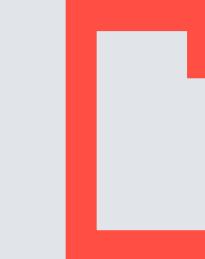
Korygująca



Zaliczkowa

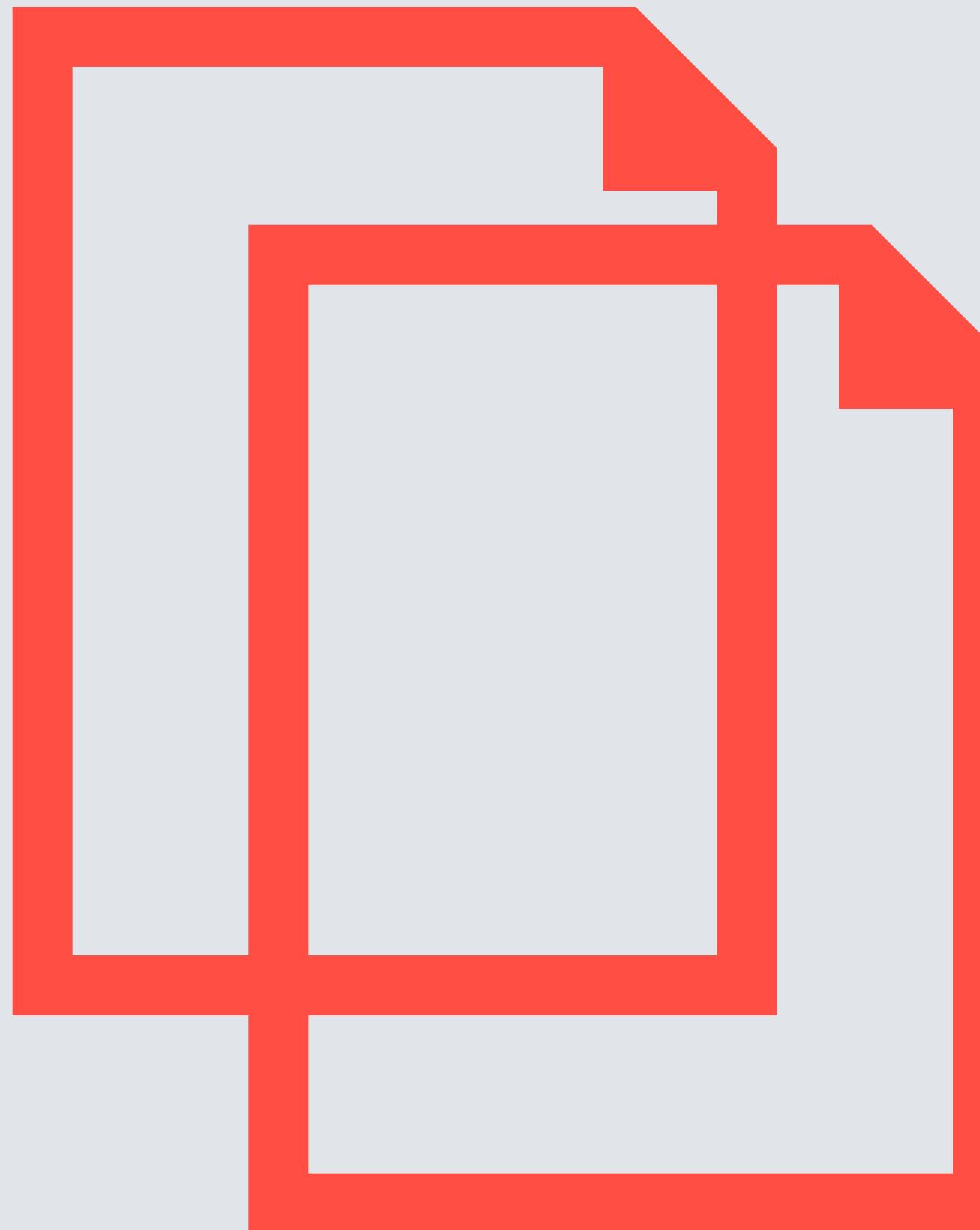


Końcowa



MOSS

Moja rada: Używaj kodu Ruby w widokach bez ograniczeń!

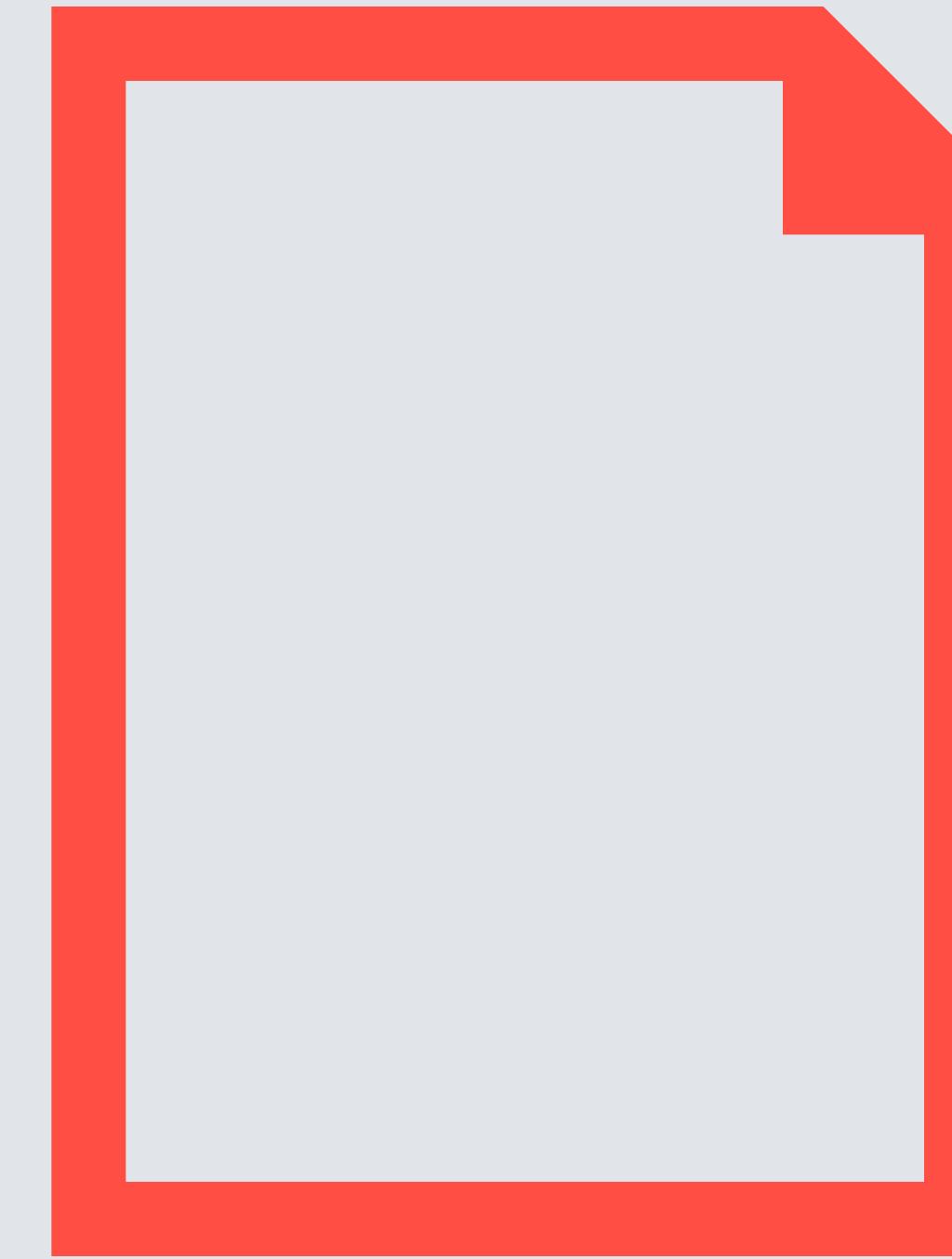


Copy!



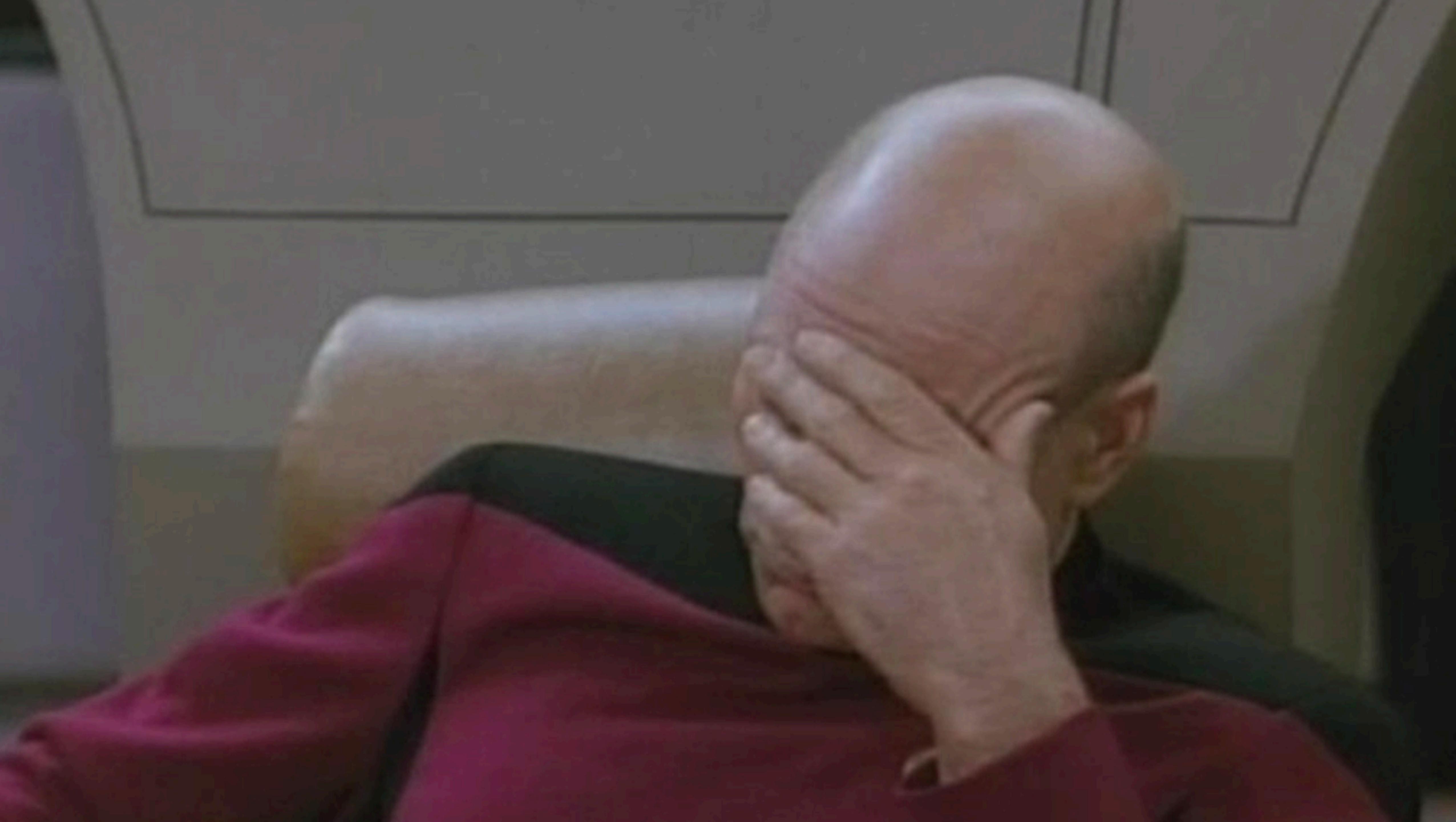
Radek

Moja rada: Używaj kodu Ruby w widokach bez ograniczeń!

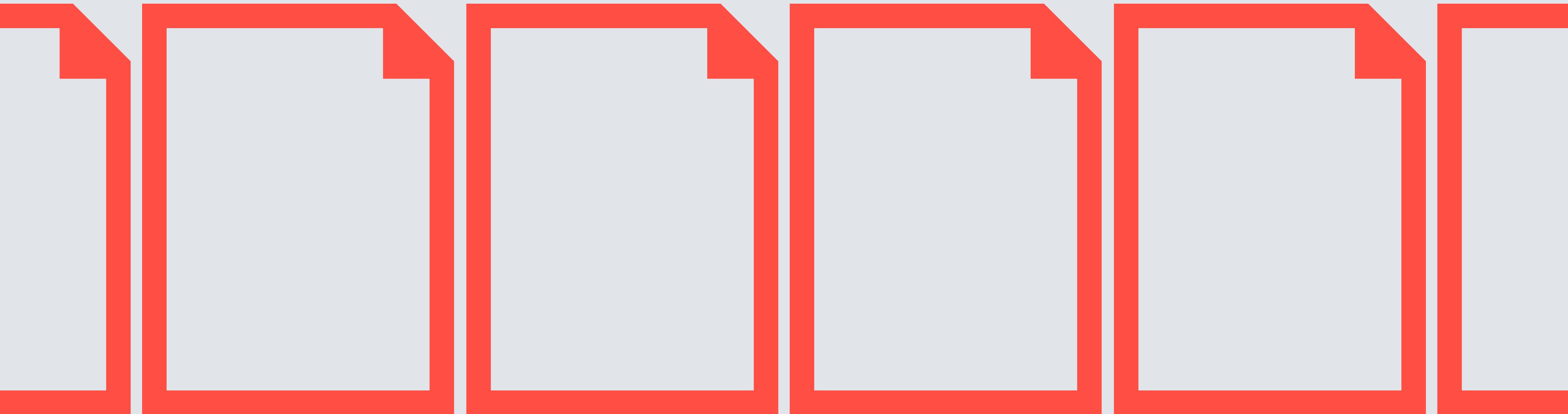


Radek

Ruby > HTML



Moja rada: Używaj kodu Ruby w widokach bez ograniczeń!



Radek

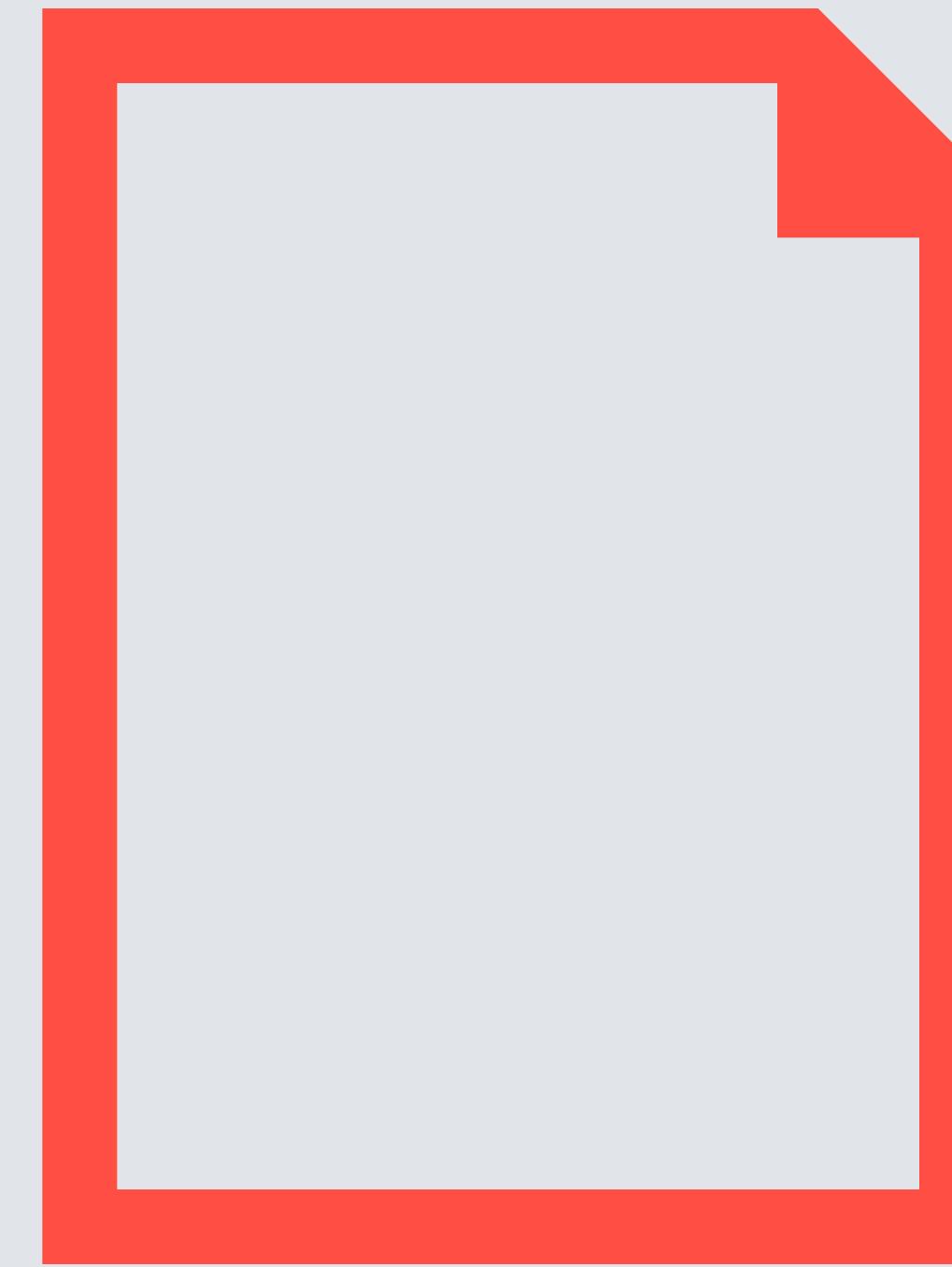
6x

Moja rada: Używaj kodu Ruby w widokach bez ograniczeń!

Redesign



Radek



Widok

Moja rada: Używaj kodu Ruby w widokach bez ograniczeń!

Redesign



Radek



Widok

Dekoratory

Moja rada: Używaj kodu Ruby w widokach bez ograniczeń!

Redesign kosztował nas ok.
2 tygodnie więcej czasu



Radek

Moja lekcja:

Korzystajmy z warstw abstrakcji,
takich jak Decorator i Presenter,
aby w nich przygotowywać dane,
które chcesz wyświetlić.



Radek

5





Piotrek Bryniarski

Ruby Developer

- podróżnik
- miłośnik fotografii

Moja rada:



Piotrek

Moja rada:

Trzymaj logikę
w kontrolerach



Piotrek

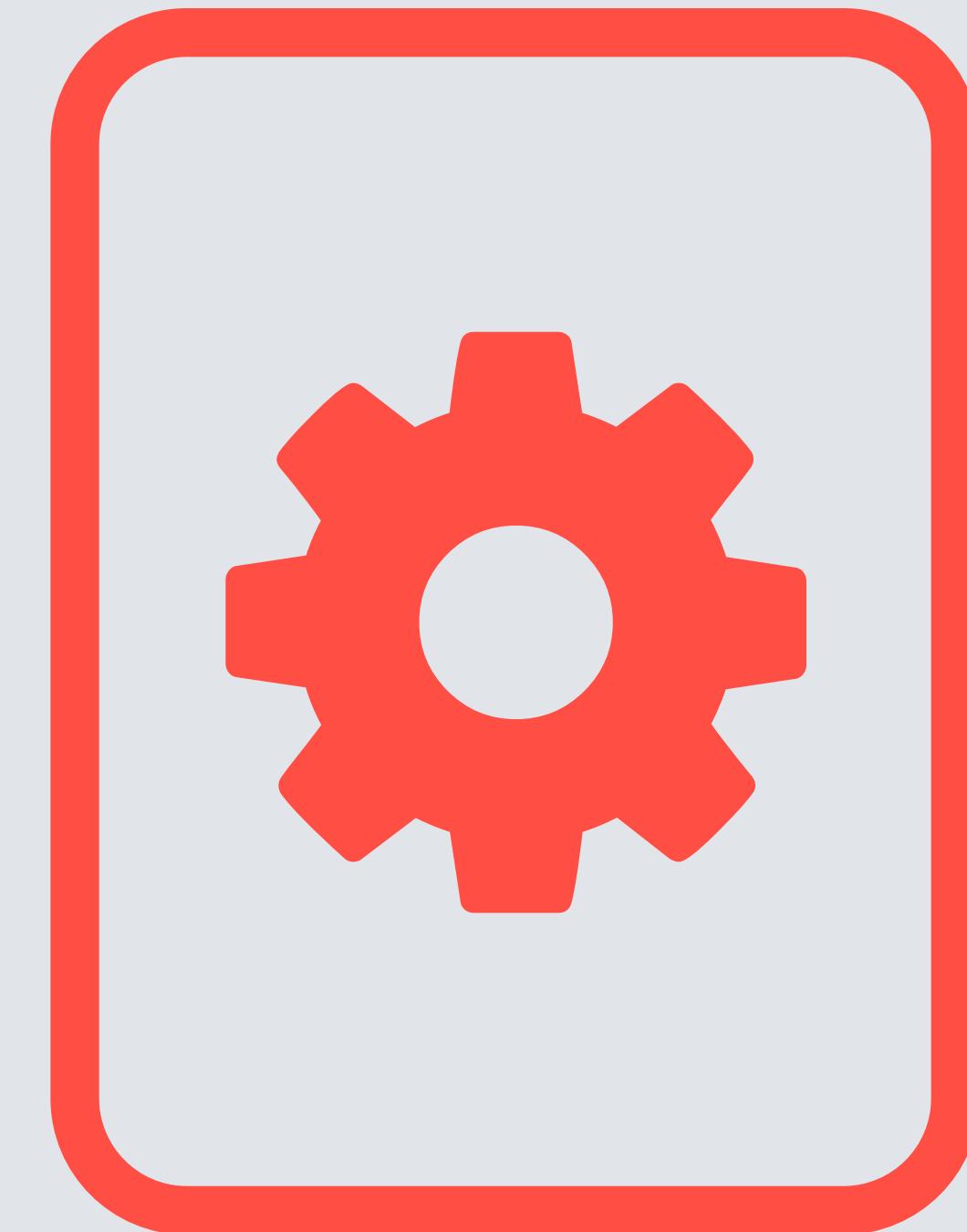
Moja rada: Trzymaj logikę w kontrolerach

We wzorcu MVC to kontroler odpowiada za logikę biznesową



Piotrek

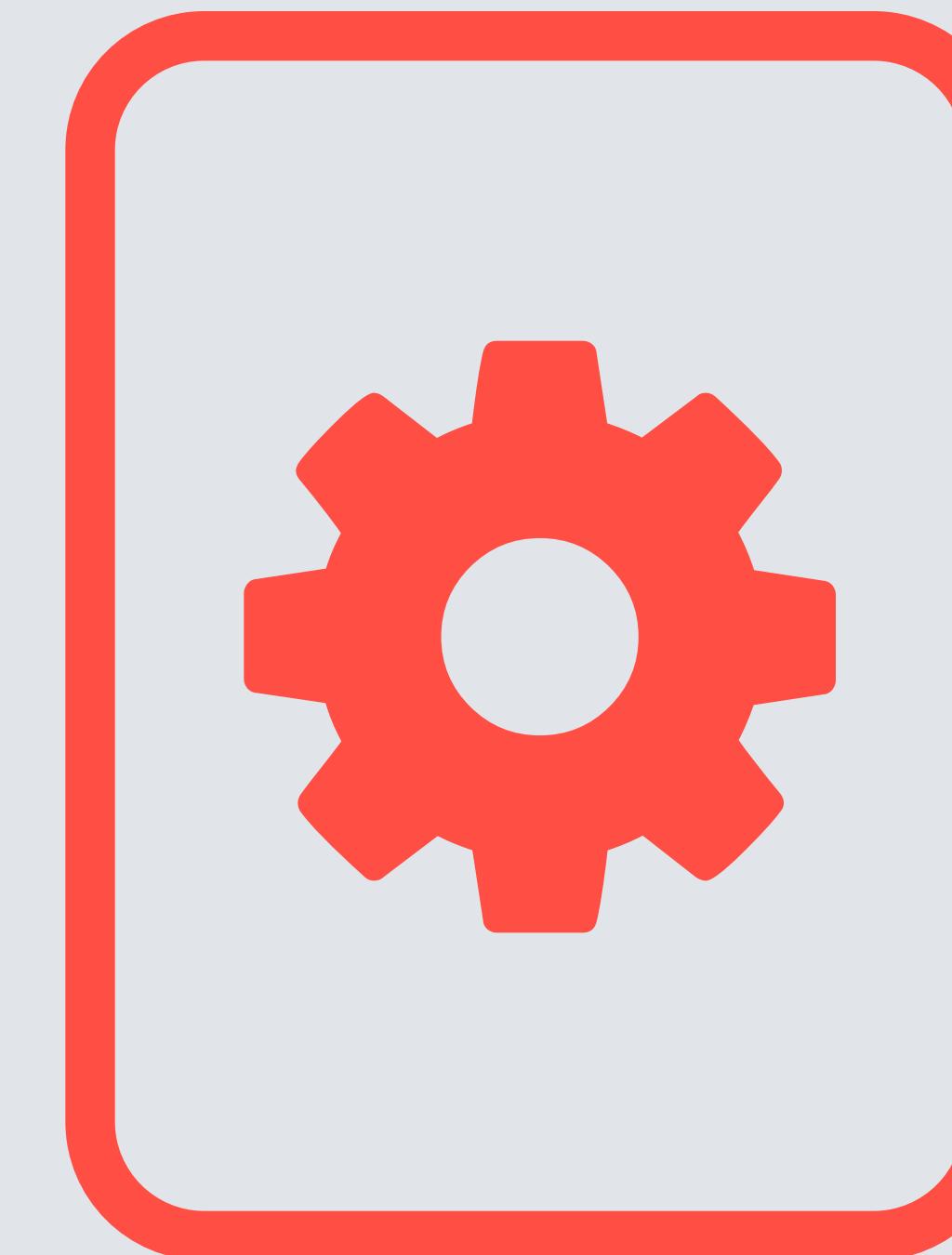
Moja rada: Trzymaj logikę w kontrolerach



Piotrek

Kontroler faktur

Moja rada: Trzymaj logikę w kontrolerach



RODZAJE

STAWKI

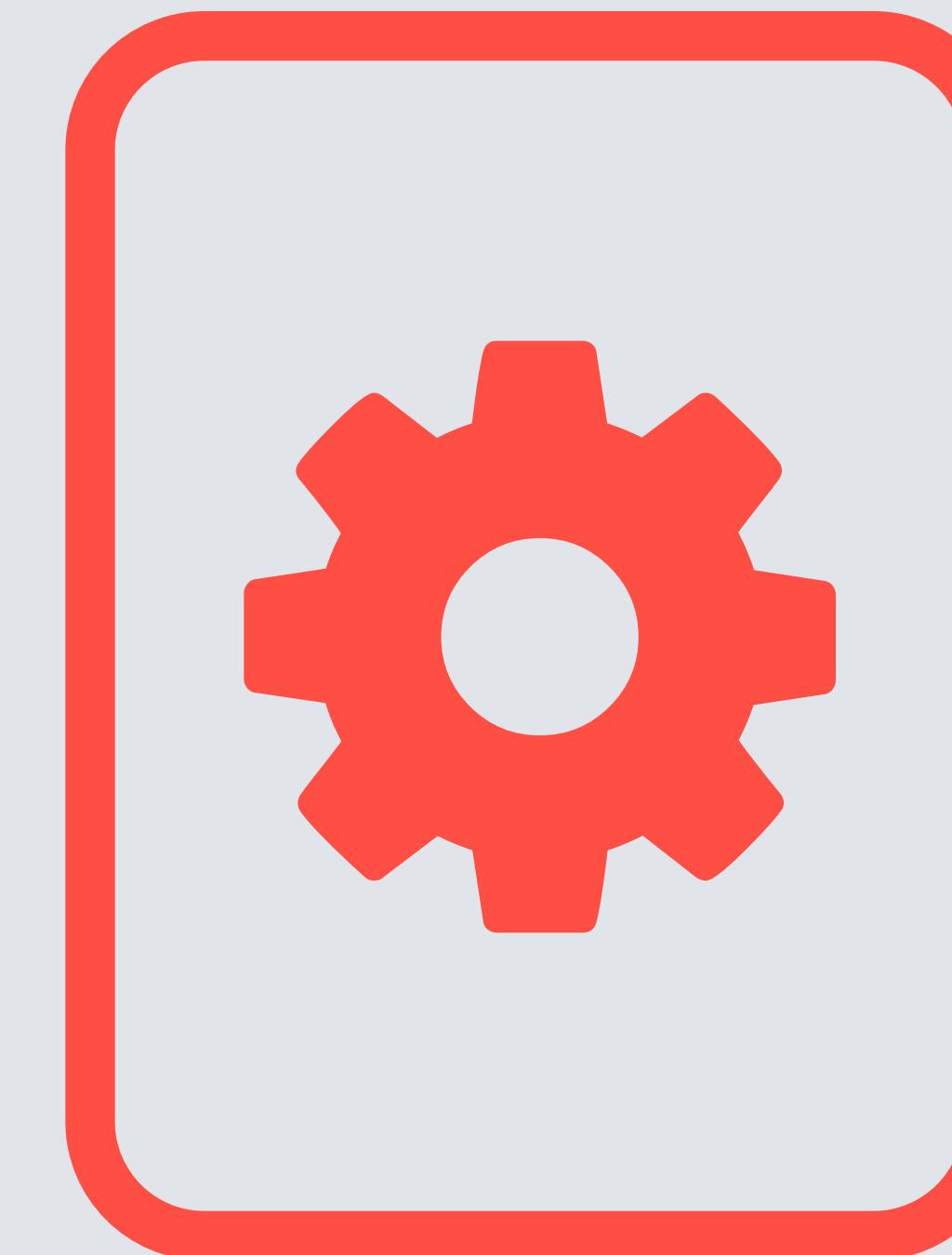
STAN



Piotrek

Kontroler faktur

Moja rada: Trzymaj logikę w kontrolerach



RODZAJE

STAWKI

STAN



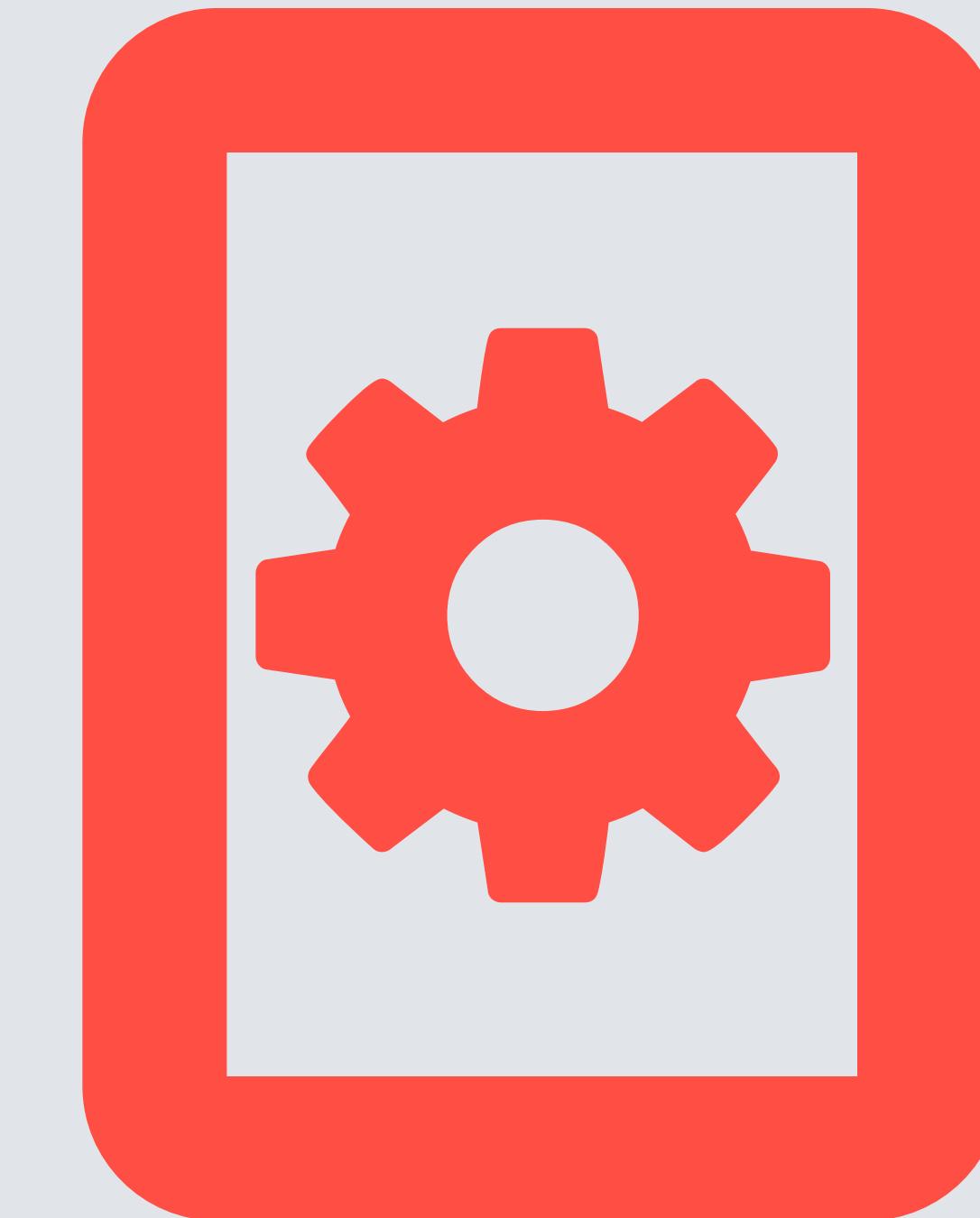
Piotrek

Kontroler faktur

Moja rada: Trzymaj logikę w kontrolerach



Piotrek



RODZAJE

STAWKI

STAN

Kontroler faktur
~ kilka tysięcy linii

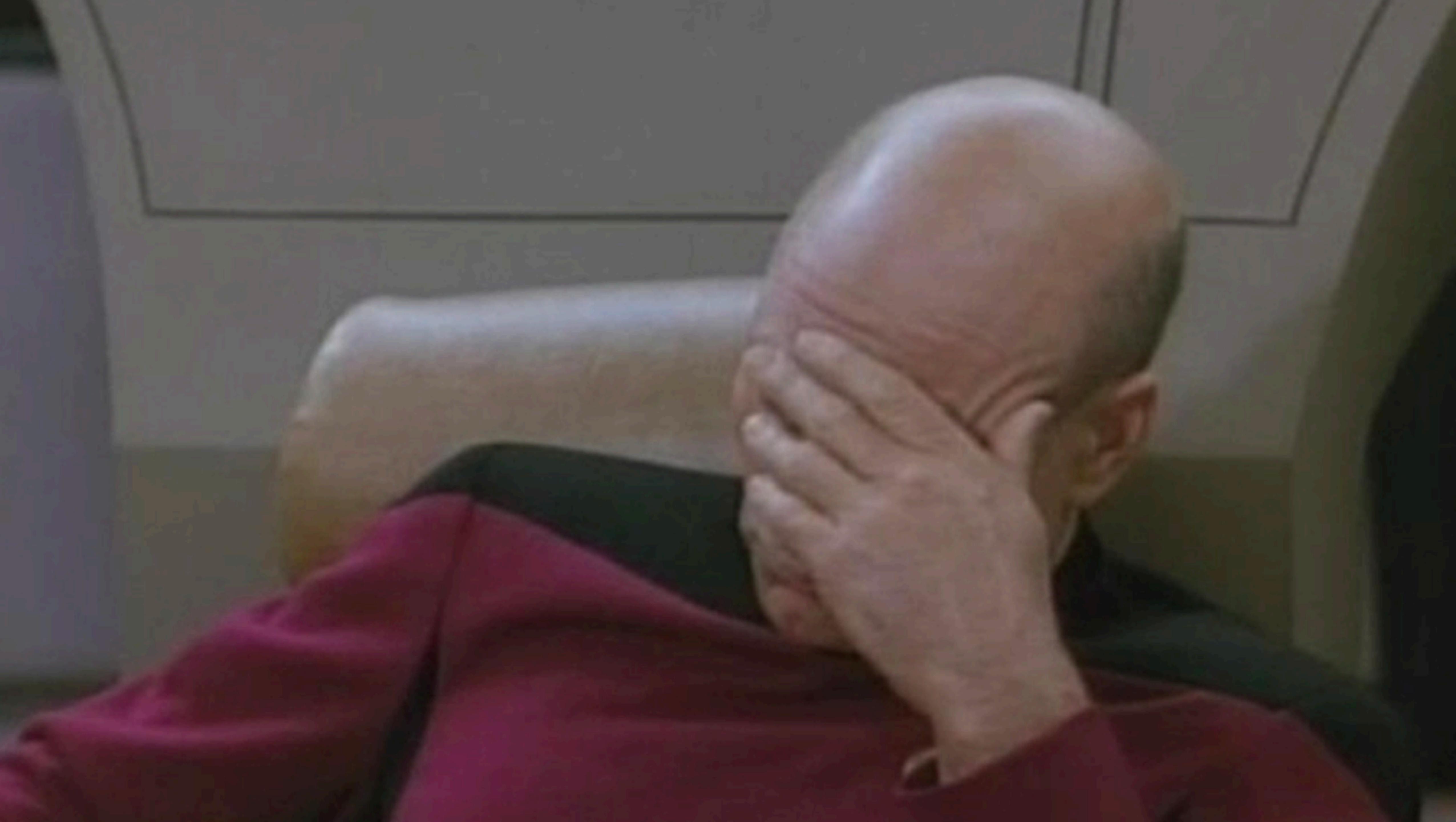
Moja rada: Trzymaj logikę w kontrolerach

API



Piotrek

API dla faktur



Moja rada: Trzymaj logikę w kontrolerach

Przy wystawianiu faktury
przez API, pojawia się błąd,
który już dawno
rozwiązałiśmy.



Piotrek

Moja rada: Trzymaj logikę w kontrolerach

Historia commitów - OK



Piotrek

Moja rada: Trzymaj logikę w kontrolerach

Historia commitów - OK

Debugging - OK



Piotrek

Moja rada: Trzymaj logikę w kontrolerach

Historia commitów - OK

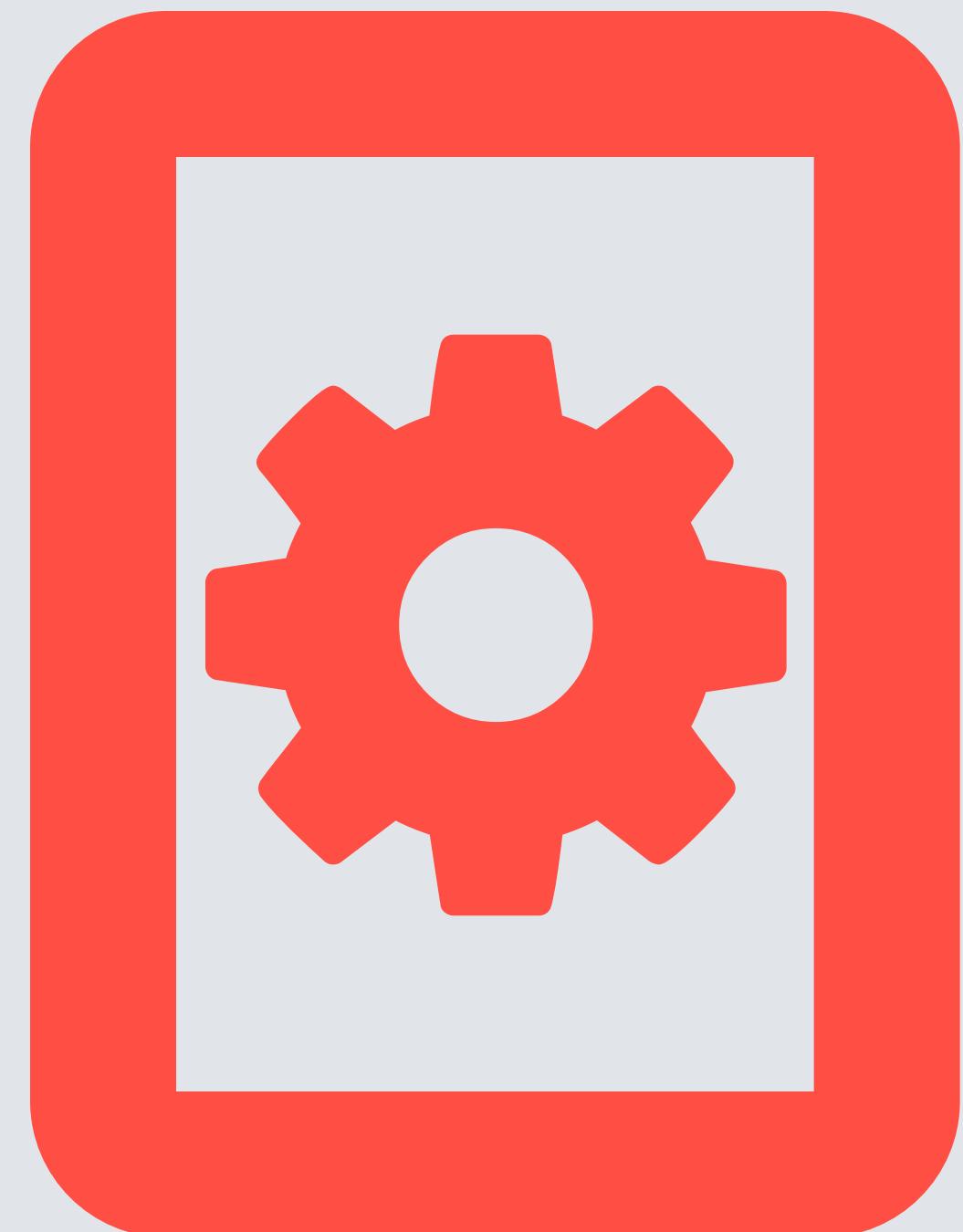
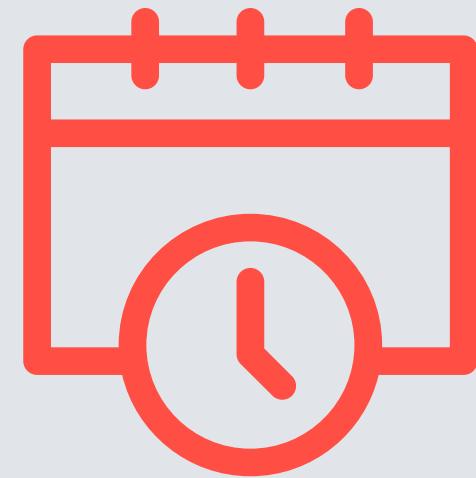
Debugging - OK

...

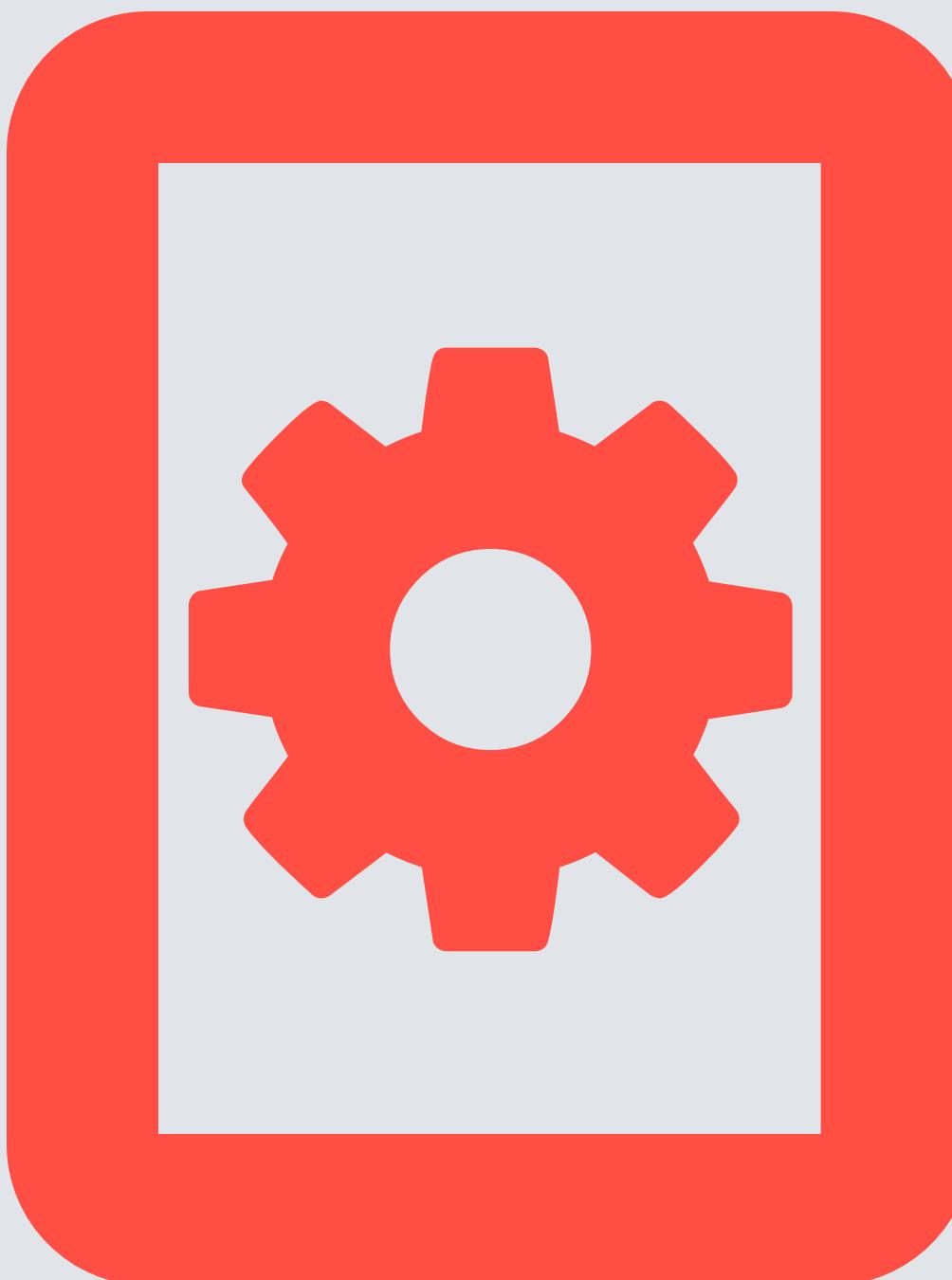


Piotrek

Moja rada: Trzymaj logikę w kontrolerach



VS



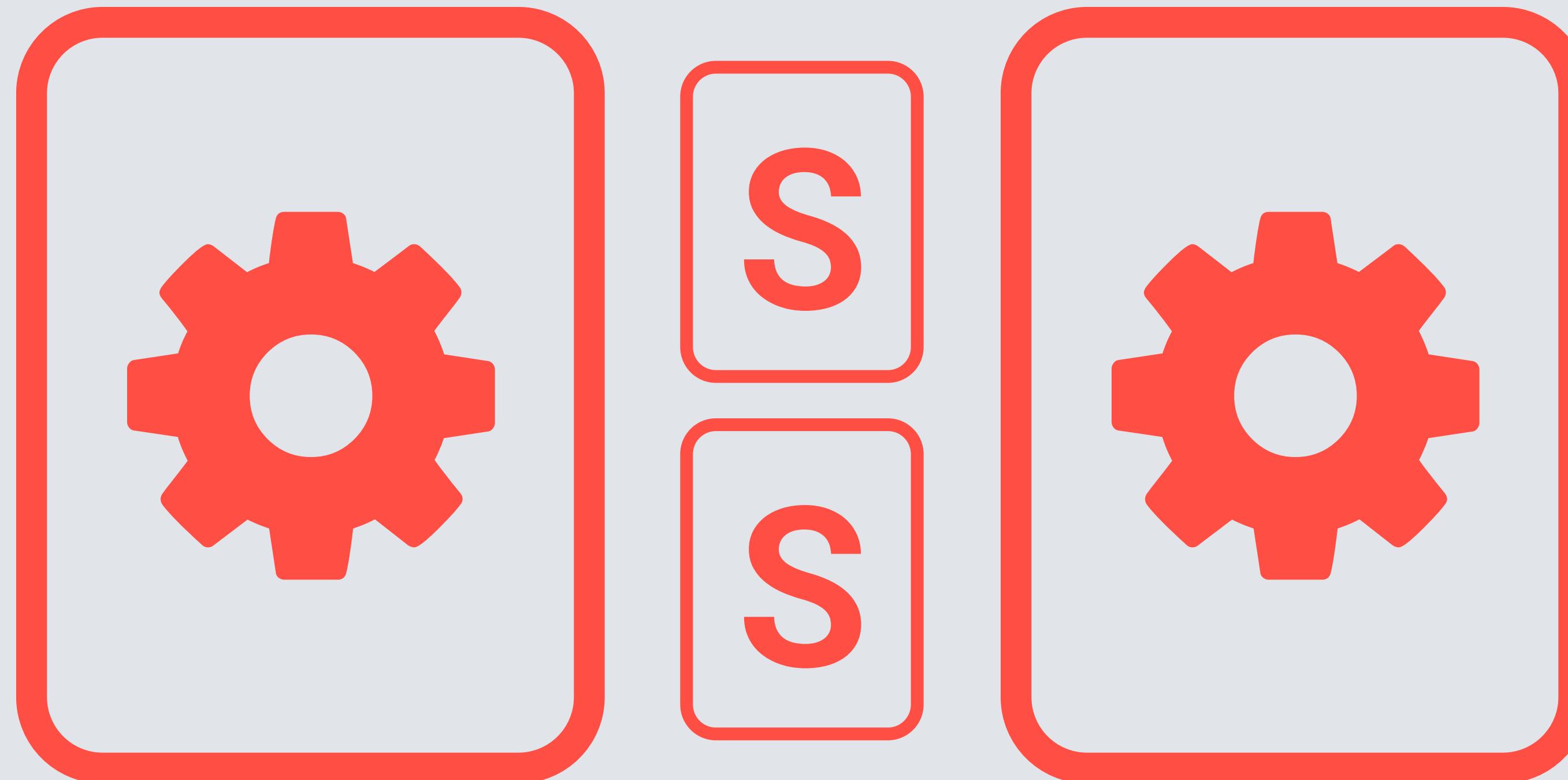
API

APP



Piotrek

Moja rada: Trzymaj logikę w kontrolerach



Piotrek

Moja lekcja:

Wydzielaj logikę biznesową od razu do serwisów nawet, jeśli wywoływane są w jednym kontrolerze.



Piotrek

6





Kazik Szołtysik

Ruby Developer

- DJ
- Sprzedawca najwytrawniejszych sucharów

Moja rada:



Kazik

Moja rada:

Stubuj metody
prywatne
testowanej klasy



Kazik

Moja rada: Stubuj metody prywatne testowanej klasy

Napiszesz prostszy i szybszy
test, bez zależności, dobrze
odizolowany.



Kazik

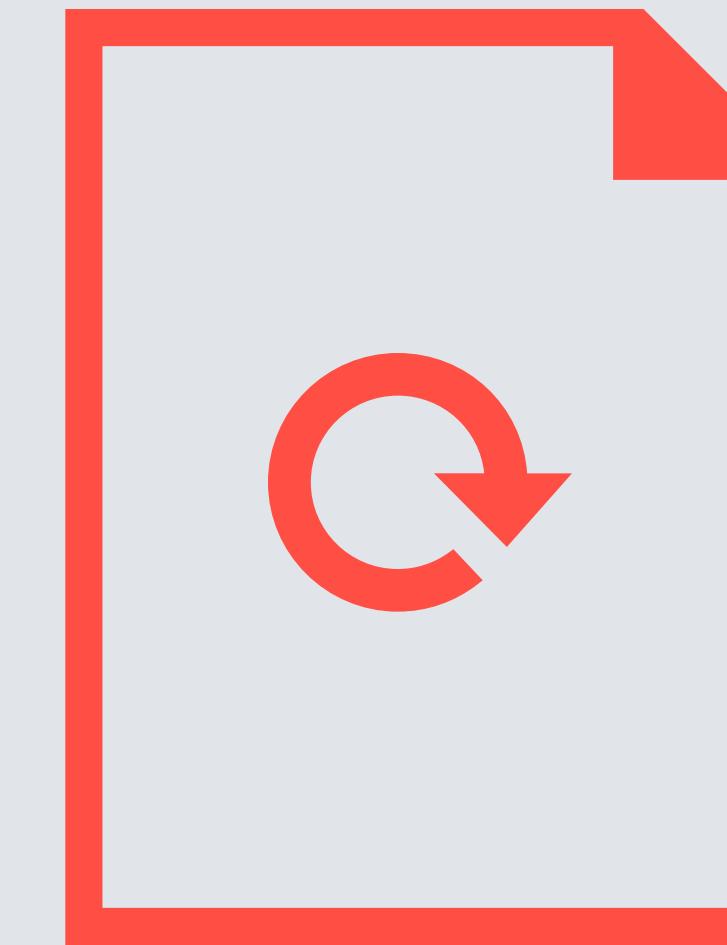
Moja rada: Stubuj metody prywatne testowanej klasy

Stub



Kazik

Moja rada: Stubuj metody prywatne testowanej klasy

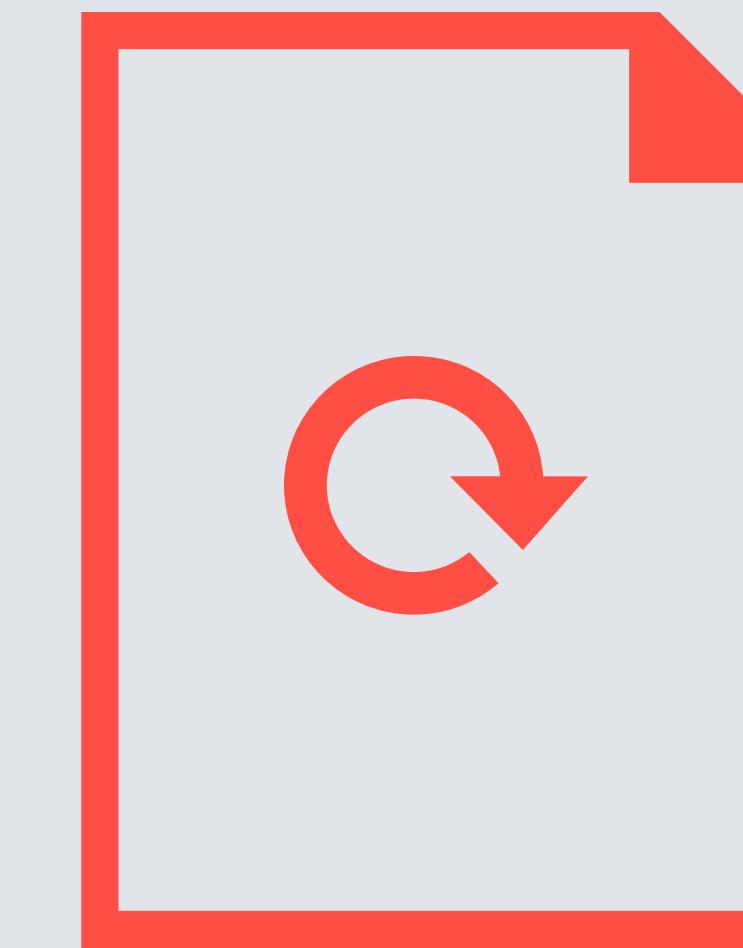


Abonament

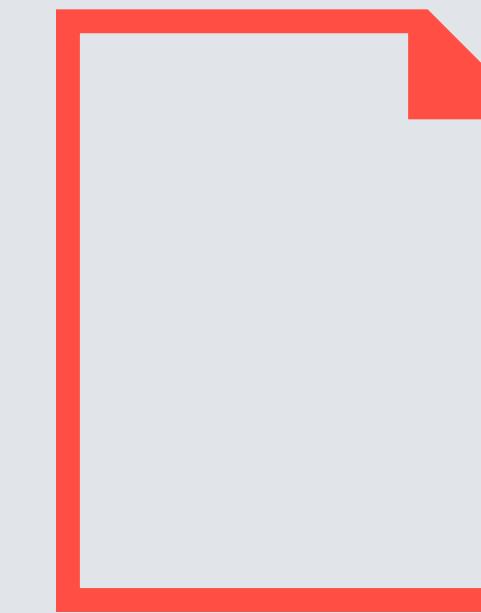


Kazik

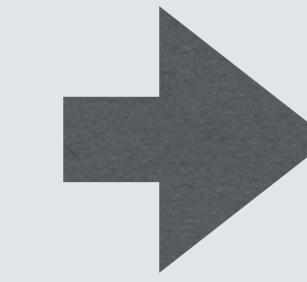
Moja rada: Stubuj metody prywatne testowanej klasy



Abonament



Umowa



User



Kazik

Moja rada: Stubuj metody prywatne testowanej klasy



Kazik

Klasa

Przyjąć obiekt usera
i odzaj umowy

Wysłać userowi maila
z nową umową

Trzymać błąd w
zmiennej instancji

Moja rada: Stubuj metody prywatne testowanej klasy



Kazik

```
1 module Payments
2   class ChangeKaiPlan
3
4     def initialize(user, params)
5       @user, @params = user, params
6     end
7
8     def error_symbol
9       @error_symbol ||==
10      case
11        when not_kai?             then :not_kai
12        when agreement_not_exist? then :agreement_not_exist
13        when agreement_already_exist? then :agreement_already_exist
14      end
15    end
16
17    def perform
18      return false if error_symbol.present?
19      ::SystemMailer.change_kai_agreement(user.id, agreement_kind.description).deliver_later
20      true
21    rescue StandardError
22      @error_symbol = :error
23      false
24    end
25
26    private
27    attr_reader :user, :params
28
29    def not_kai?
30      !user.kai?
31    end
32
33    def agreement_not_exist?
34      agreement_kind.nil?
35    end
36
37    def agreement_already_exist?
38      agreement_kind == current_agreement_kind
39    end
40
41    def agreement_kind
42      Kai::Agreement::Kind.find_by(:symbol, params[:agreement_kind])
43    end
44
45    def current_agreement_kind
46      user.kai.current_agreement.kind
47    end
48  end
49 end
```

Moja rada: Stubuj metody prywatne testowanej klasy



Kazik

```
3 describe Payments::ChangeKaiPlan, payments_process: true do
4   let(:user) { build(:empty_user) }
5   let(:surcharge) { double }
6   subject { described_class.new(user, {agreement_kind: 'no_gratis_scan'}) }
7
8   describe "#perform" do
9     before {
10       subject.stubs(:error_symbol).returns(false)
11     }
12
13   context "when true" do
14     its(:perform) { should eql(true) }
15   end
16 end
17
18 describe "#error_symbol" do
19   before do
20     subject.stubs(:not_kai?).returns(not_kai)
21     subject.stubs(:agreement_not_exist?).returns(agreement_not_exist)
22     subject.stubs(:agreement_already_exist?).returns(agreement_already_exist)
23   end
24
25   context "when not_kai" do
26     let(:not_kai) { true }
27     let(:agreement_not_exist) { false }
28     let(:agreement_already_exist) { false }
29     its(:error_symbol) { should == :not_kai }
30   end
31
32   context "when agreement_not_exist" do
33     let(:not_kai) { false }
34     let(:agreement_not_exist) { true }
35     let(:agreement_already_exist) { false }
36     its(:error_symbol) { should == :agreement_not_exist }
37   end
38
39   context "when agreement_already_exist" do
40     let(:not_kai) { false }
41     let(:agreement_not_exist) { false }
42     let(:agreement_already_exist) { true }
43     its(:error_symbol) { should == :agreement_already_exist }
44   end
45 end
46 end
```

Moja rada: Stubuj metody prywatne testowanej klasy

```
Payments::ChangeKaiPlan
  #perform
  when true
    perform
      should eql true
  #error_symbol
  when agreement_already_exist
    error_symbol
      should == :agreement_already_exist
  when agreement_not_exist
    error_symbol
      should == :agreement_not_exist
  when not_kai
    error_symbol
      should == :not_kai
```



Kazik

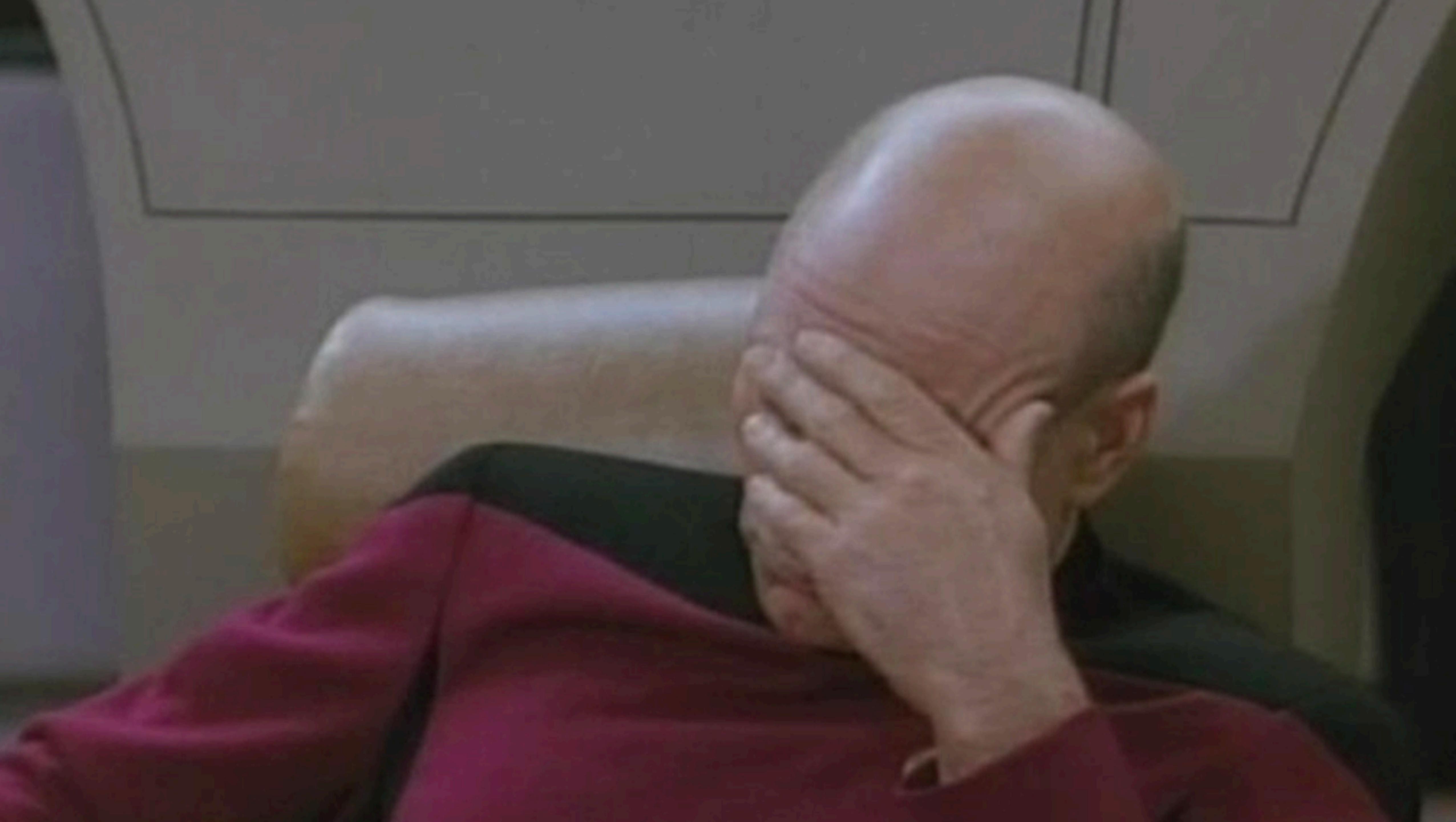
Moja rada: Stubuj metody prywatne testowanej klasy



Lecimy na produkcję



Kazik



Moja rada: Stubuj metody prywatne testowanej klasy

O błędzie dowiedzieliśmy się z naszego Działu Obsługi Klienta w momencie, gdy ktoś z naszych userów nie otrzymał maila z nową umową.



Kazik

Moja rada: Stubuj metody prywatne testowanej klasy

```
def not_kai?  
    !user.kai?  
end
```



Kazik

Moja rada: Stubuj metody prywatne testowanej klasy

```
def not_kai?  
    !user.kai?  
end
```

```
def not_kai?  
    # !user.kai?  
end
```



Kazik

Moja rada: Stubuj metody prywatne testowanej klasy

```
def not_kai?  
  !user.kai?  
end
```

```
def not_kai?  
  # !user.kai?  
end
```

```
when not_kai  
  error_symbol  
    should == :not_kai
```



Kazik

Moja rada: Stubuj metody prywatne testowanej klasy

Rozwiązanie



Kazik

Moja rada: Stubuj metody prywatne testowanej klasy

Rozwiążanie Przerobienie wadliwego testu



Kazik

Moja lekcja:

Stubuj, ale nie bezpośrednio na prywatnych metodach testowanej klasy bo w nich siedzi cała logika.



Kazik

Moja lekcja:

Stubuj, ale nie bezpośrednio na prywatnych metodach testowanej klasy bo w nich siedzi cała logika.

Stubując je, pozbawiamy nasz test tego, co najważniejsze, czyli przetestowania naszej logiki biznesowej i zabezpieczenia jej przed zmianami.



Kazik

7





Radek

Moja rada:



Radek

Moja rada:

Używaj rescue nil



Radek

Moja rada: Używaj rescue nil

Po co się martwić poszczególnymi rodzajów błędów? Złapmy wszystkie, przecież błąd to błąd.



Radek

Moja rada: Używaj rescue nil



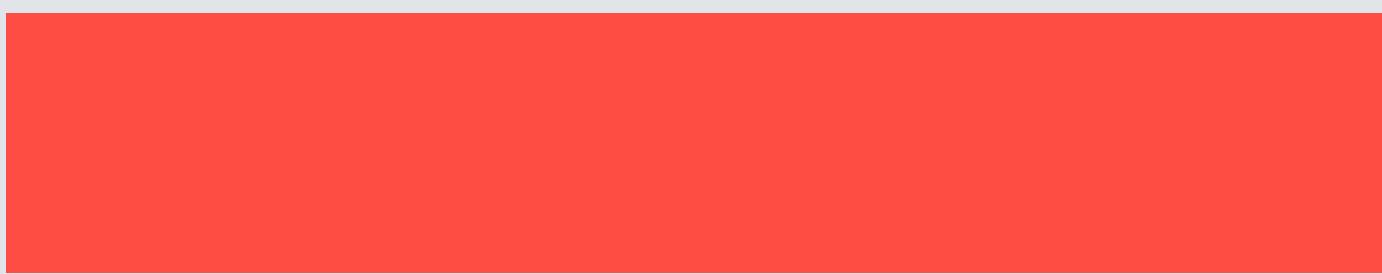
Radek

Moja rada: Używaj rescue nil



Radek

■ Begin



Rescue nil

■ End

Moja rada: Używaj rescue nil



Radek

■ Begin



■ Rescue

nil

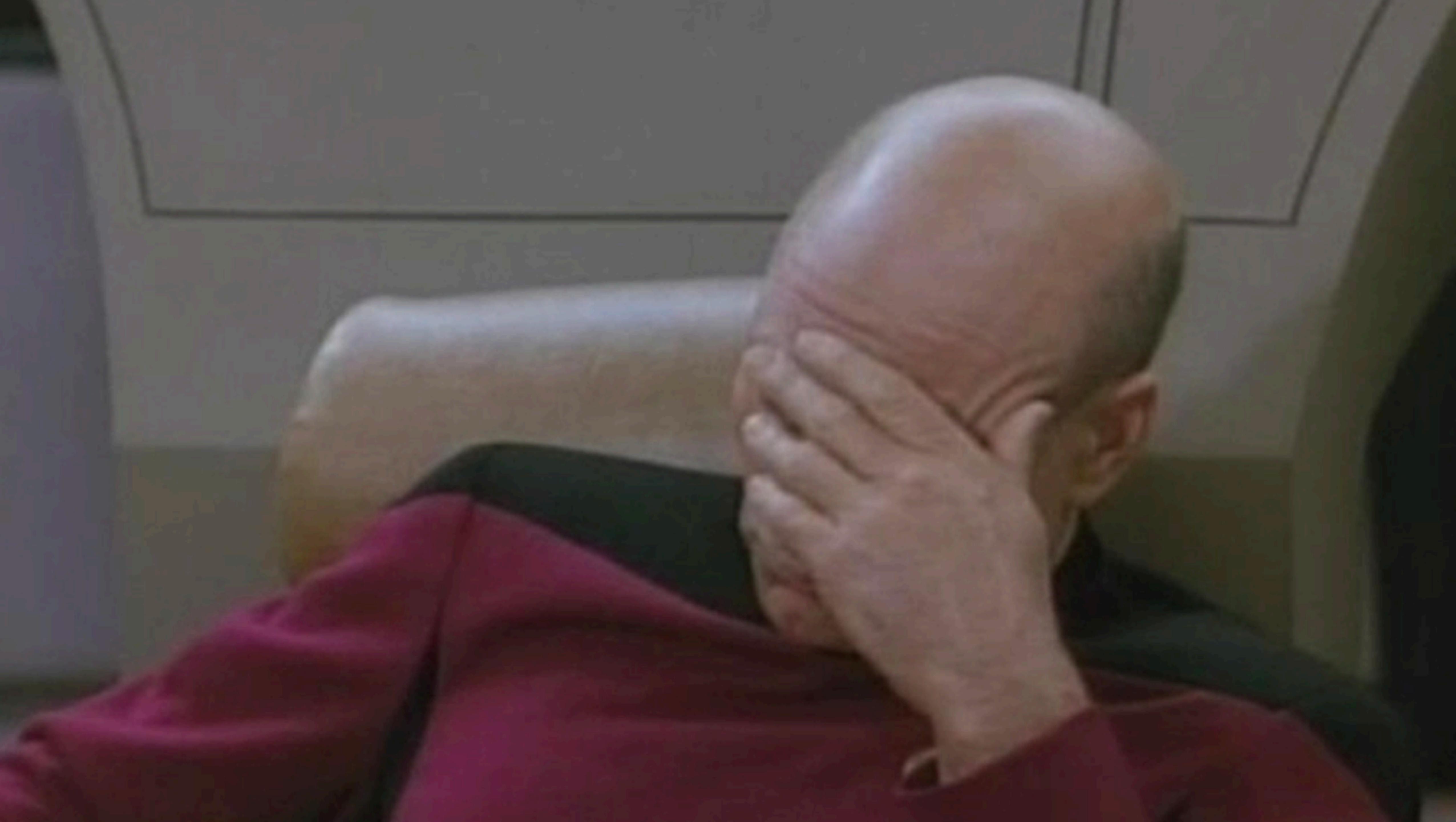
■ End

Moja rada: Używaj rescue nil

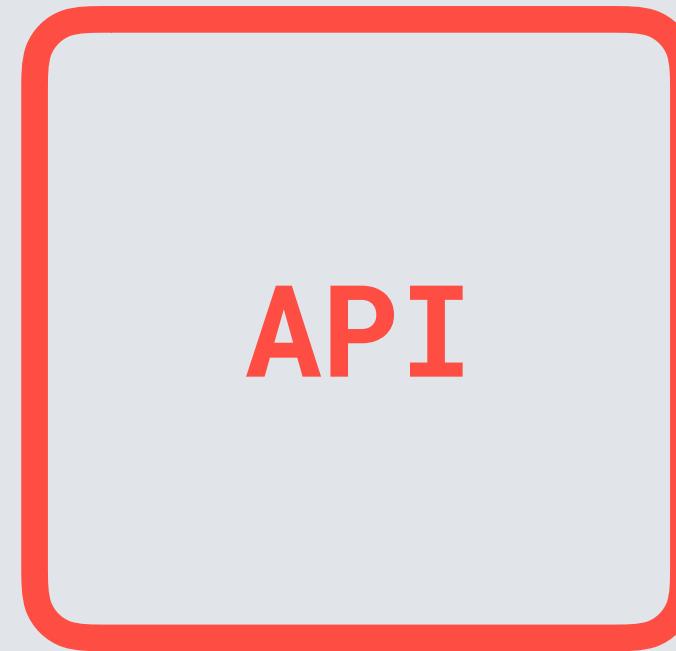
Aplikacja działała rok, nie wykazując błędów.



Radek



Moja rada: Używaj rescue nil



Usługa OCR



Radek

Moja rada: Używaj rescue nil

System informowania o błędach milczał, ponieważ wszystkie były obsłużone.



Radek

Moja rada: Używaj rescue nil

Ratujemy



Radek

Moja rada: Używaj rescue nil

Ratujemy

- Naprawiamy zadania



Radek

Moja rada: Używaj rescue nil

Ratujemy

- Naprawiamy zadania
- Definiujemy rodzaje błędów



Radek

Moja rada: Używaj rescue nil

Ratujemy

- Naprawiamy zadania
- Definiujemy rodzaje błędów
- Przebudowujemy obsługę błędów



Radek

Moja lekcja:

Nie używaj rescue bez określenia rodzaju błędu, który chcesz obsłużyć. Rescue nil działa tak, jakby nic się nie stało.



Radek

8





Sebastian

Moja rada:



Sebastian

Moja rada:

Używaj Callbacków
Active Record do
logiki biznesowej



Sebastian

Moja rada: Używaj Callbacków AR do logiki biznesowej

Nie musisz pamiętać o akcji,
która ma się wykonać w ramach
danej transakcji.

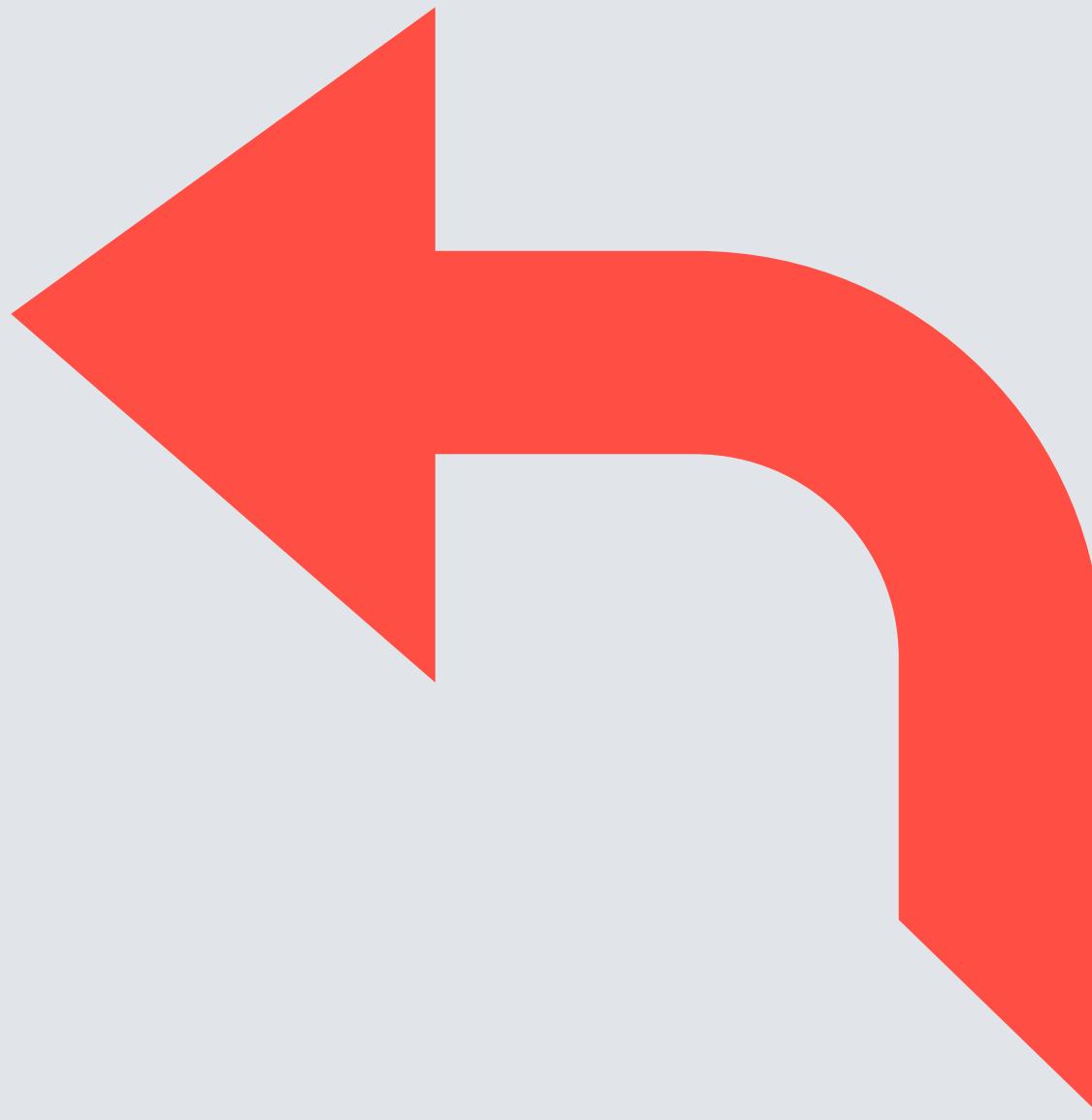


Sebastian

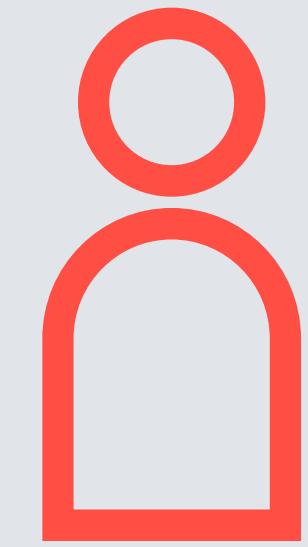
Moja rada: Używaj Callbacków AR do logiki biznesowej



Sebastian



Moja rada: Używaj Callbacków AR do logiki biznesowej

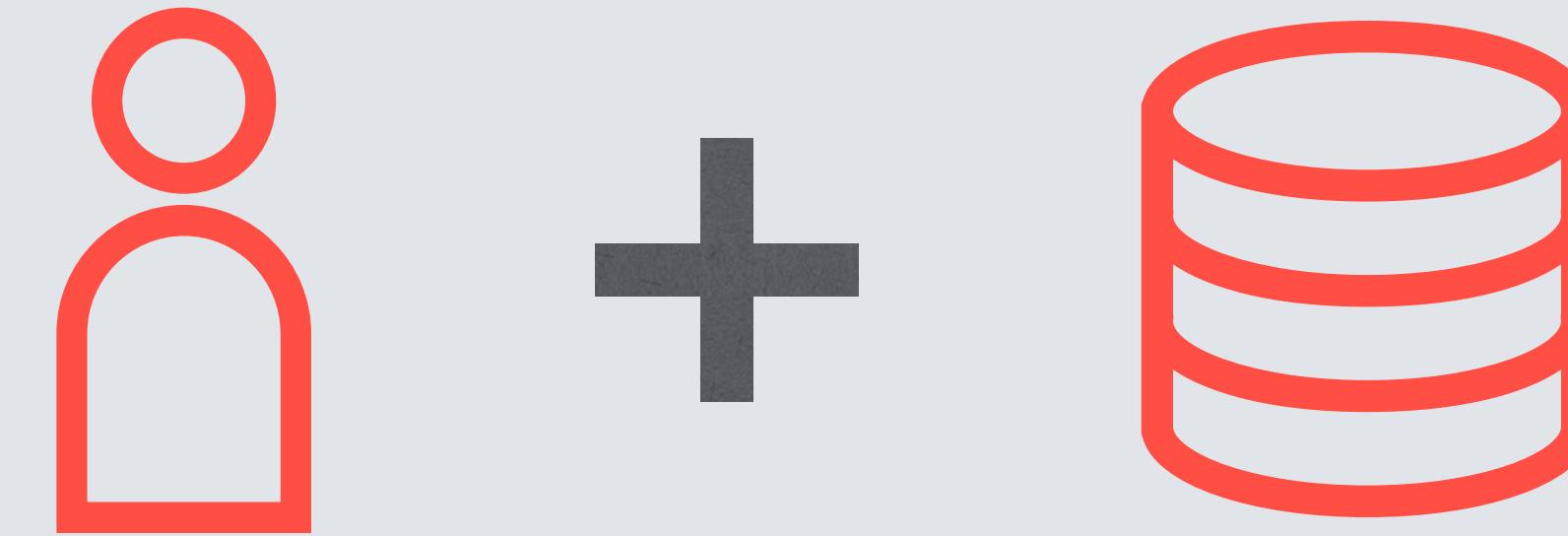


Nowy user



Sebastian

Moja rada: Używaj Callbacków AR do logiki biznesowej



Nowy user

Baza danych



Sebastian

Moja rada: Używaj Callbacków AR do logiki biznesowej



Nowy user

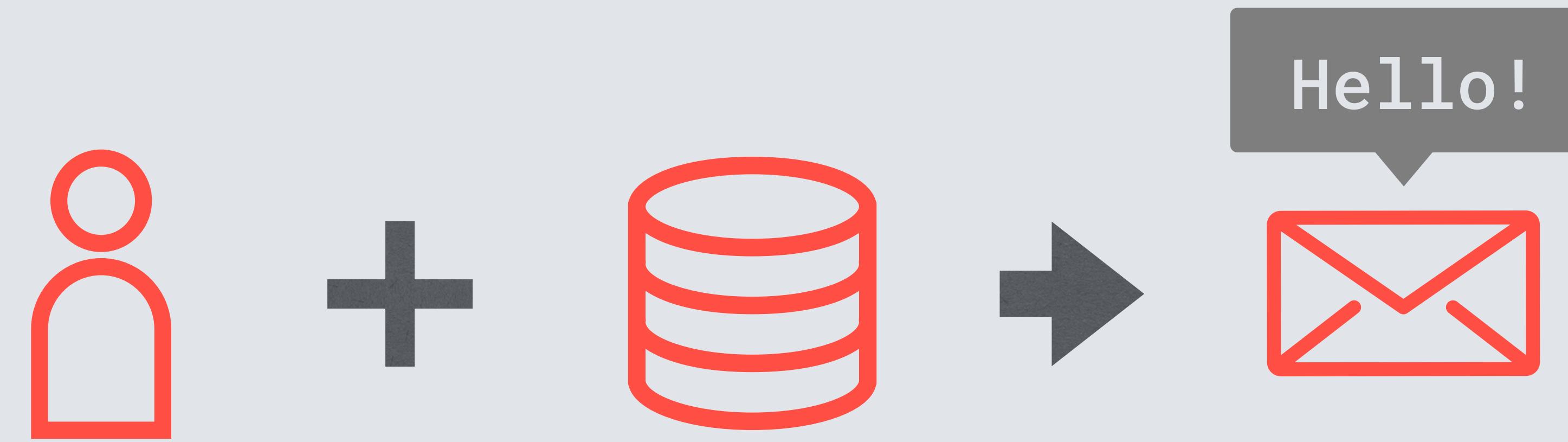
Baza danych

Mail
powitalny



Sebastian

Moja rada: Używaj Callbacków AR do logiki biznesowej



Sebastian

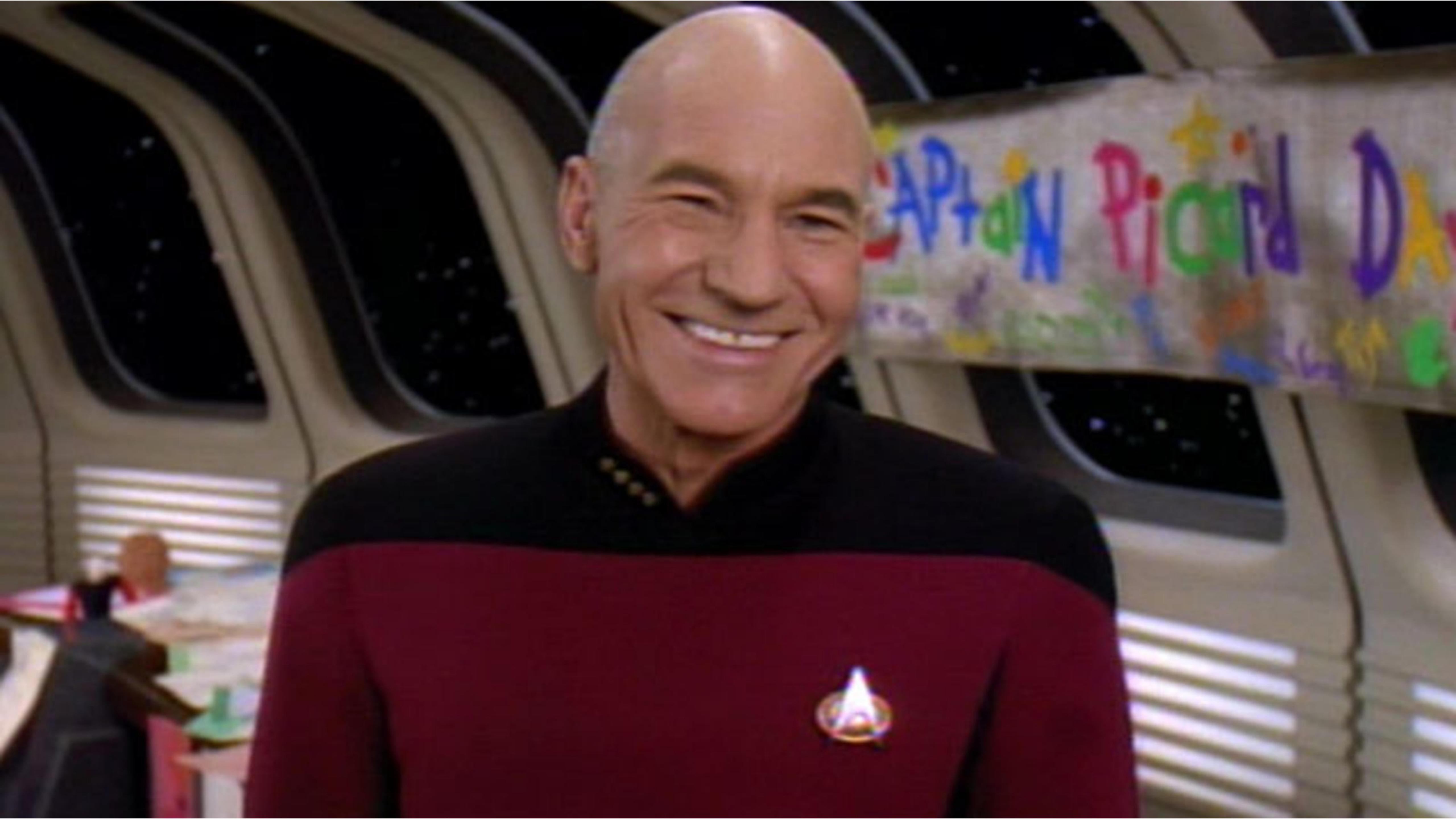
Moja rada: Używaj Callbacków AR do logiki biznesowej



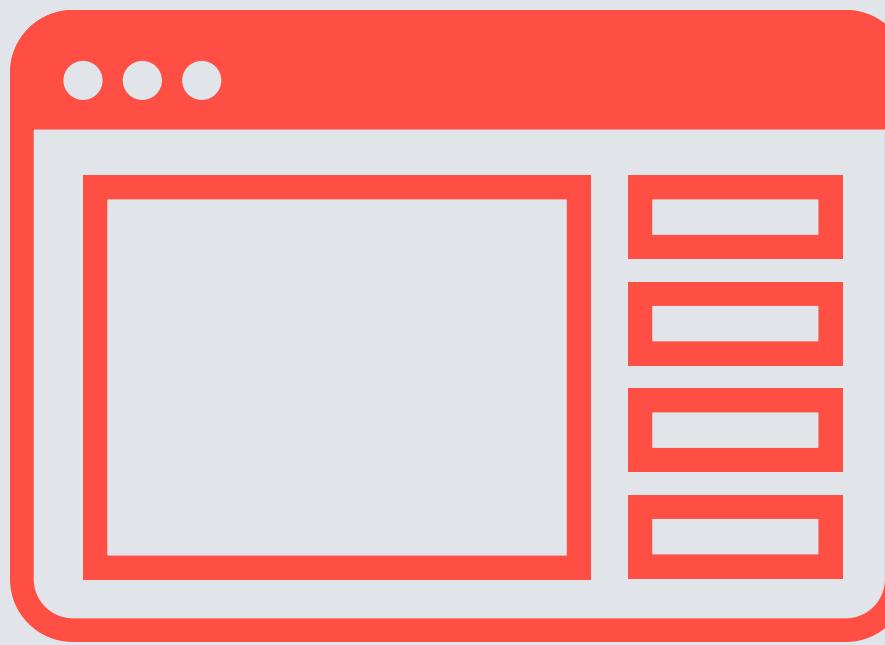
Lecimy na produkcję



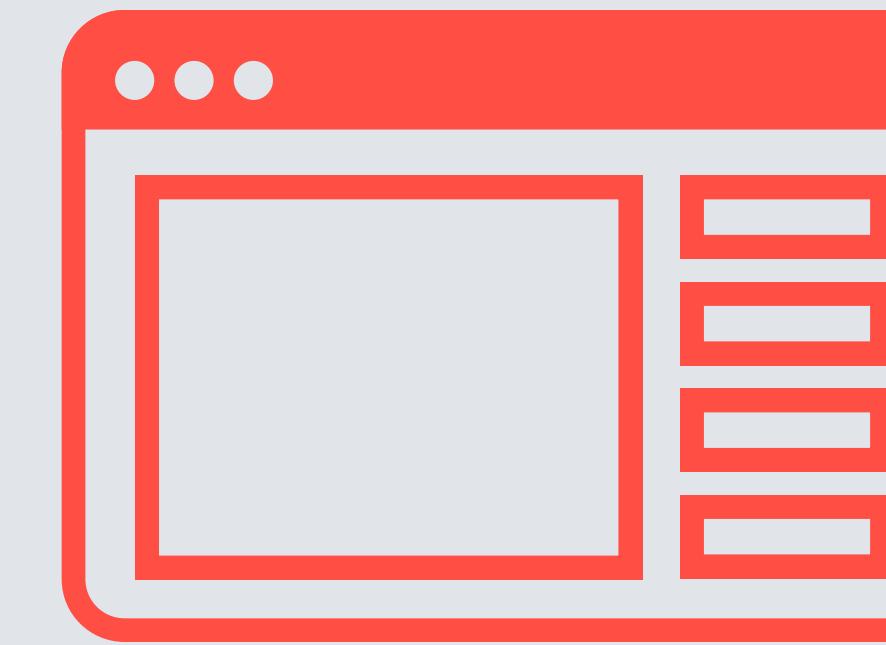
Sebastian



Moja rada: Używaj Callbacków AR do logiki biznesowej



Porachunek



inFakt

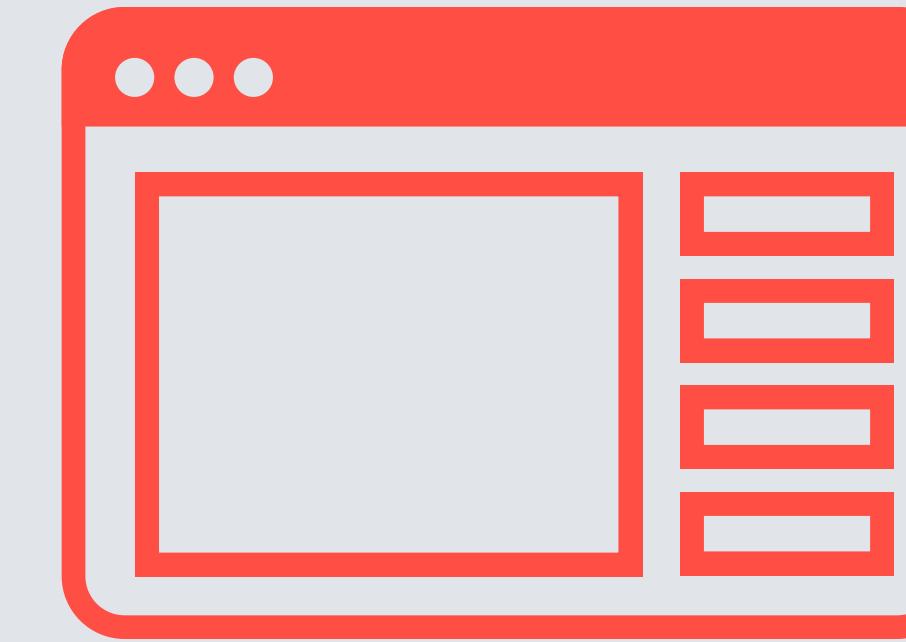


Sebastian

Moja rada: Używaj Callbacków AR do logiki biznesowej



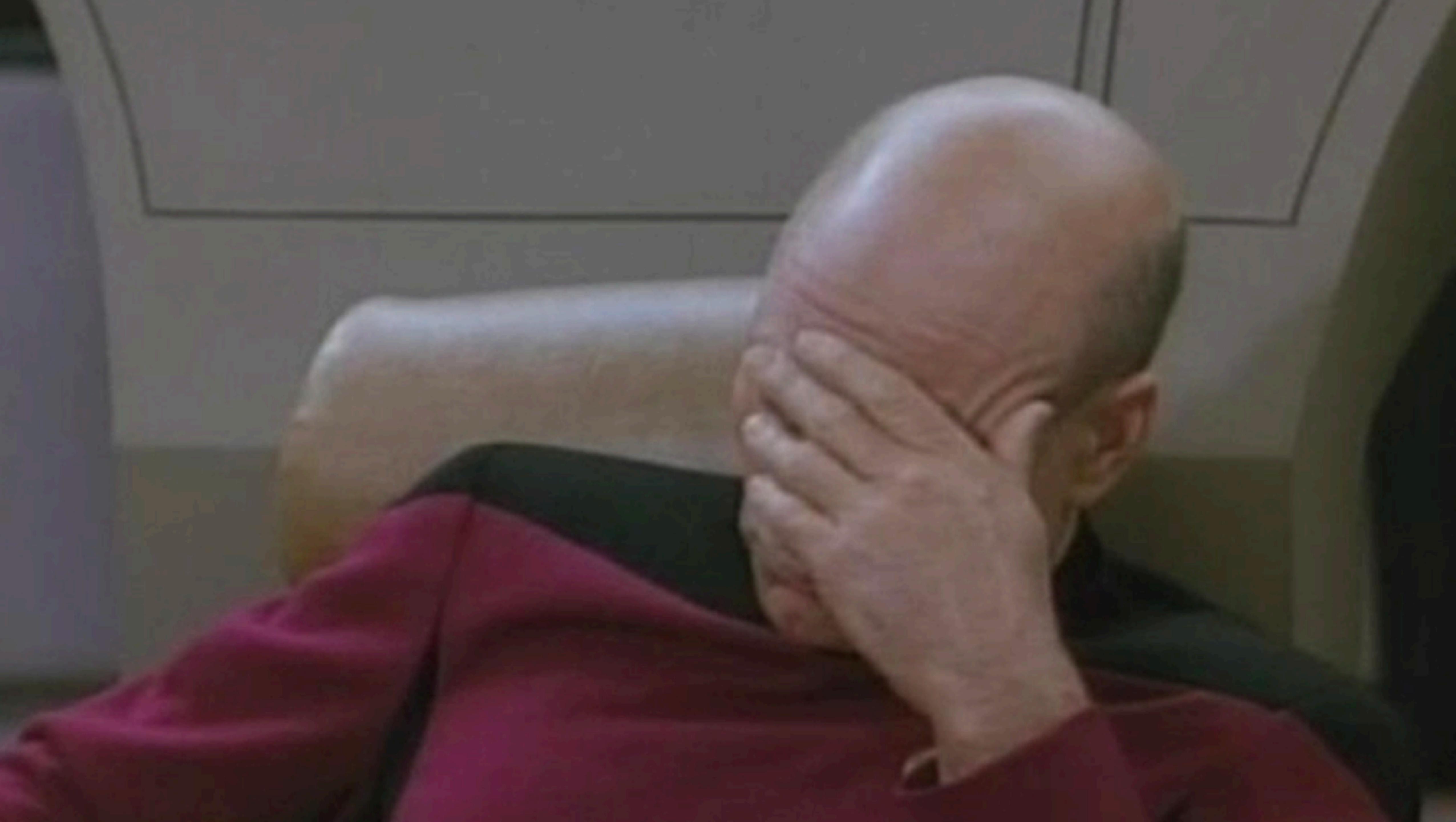
Porachunek



inFakt



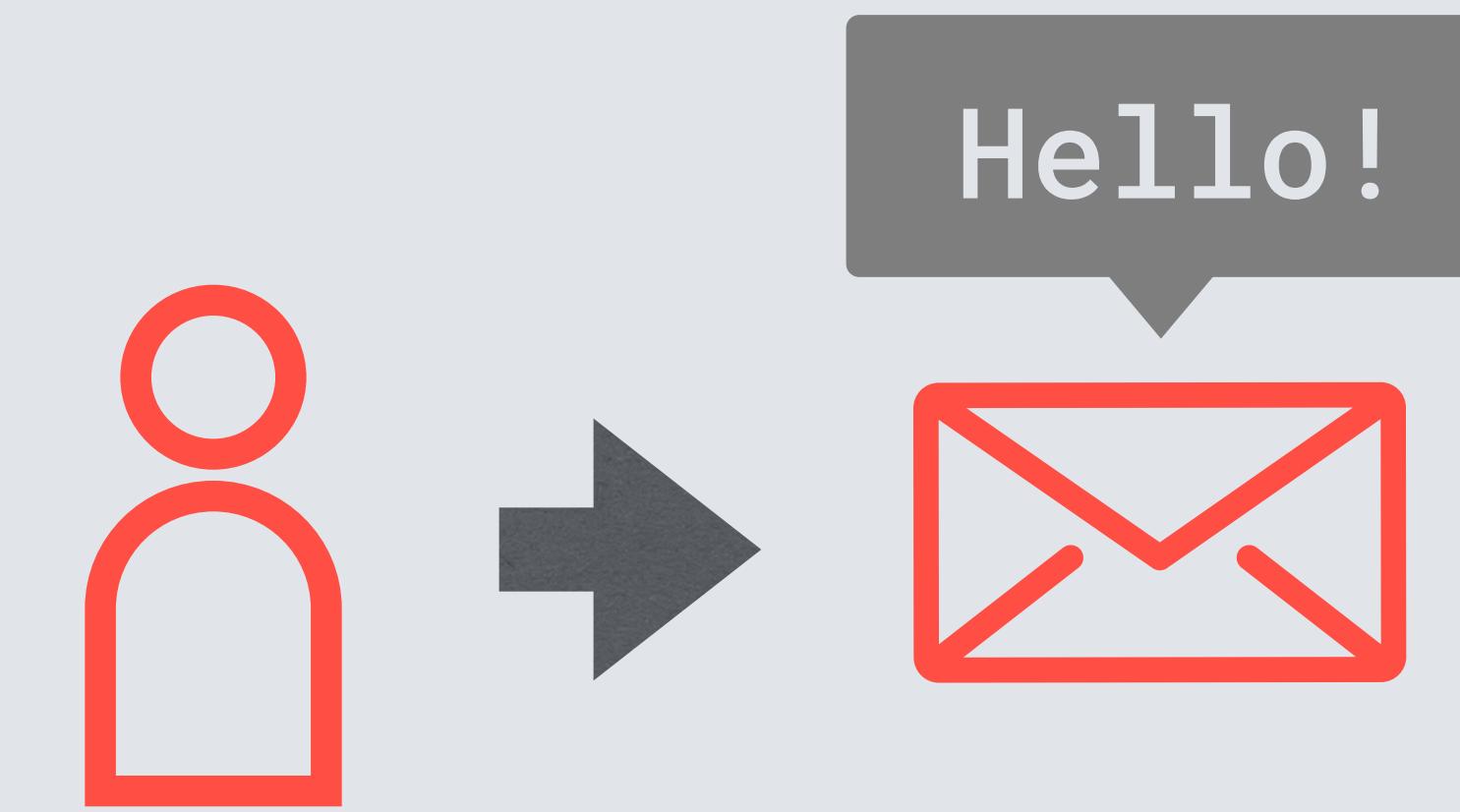
Sebastian



Moja rada: Używaj Callbacków AR do logiki biznesowej



Sebastian



Moja rada: Używaj Callbacków AR do logiki biznesowej

Całą reakcję Klientów musiał
przyjąć na siebie Dział
Obsługi Klienta, który się
tłumaczył się z naszego błędu



Sebastian

Moja lekcja:

Callbacki potrafią być nieprzewidywalne
i potrafią zaskoczyć w najmniej oczekiwany
momencie.



Sebastian

Moja lekcja:

Callbacki potrafią być nieprzewidywalne i potrafią zaskoczyć w najmniej oczekiwany momencie.

Zamiast callbacków - jawnie i świadomie wywołuj metody



Sebastian

9





Alicja

Moja rada:



Alicja

Moja rada:

Używaj dużo
metaprogrammingu



Alicja

Moja rada: Używaj dużo metaprogrammingu

Klasę z dużą ilością podobnych metod możesz zapisać w kilku linijkach.

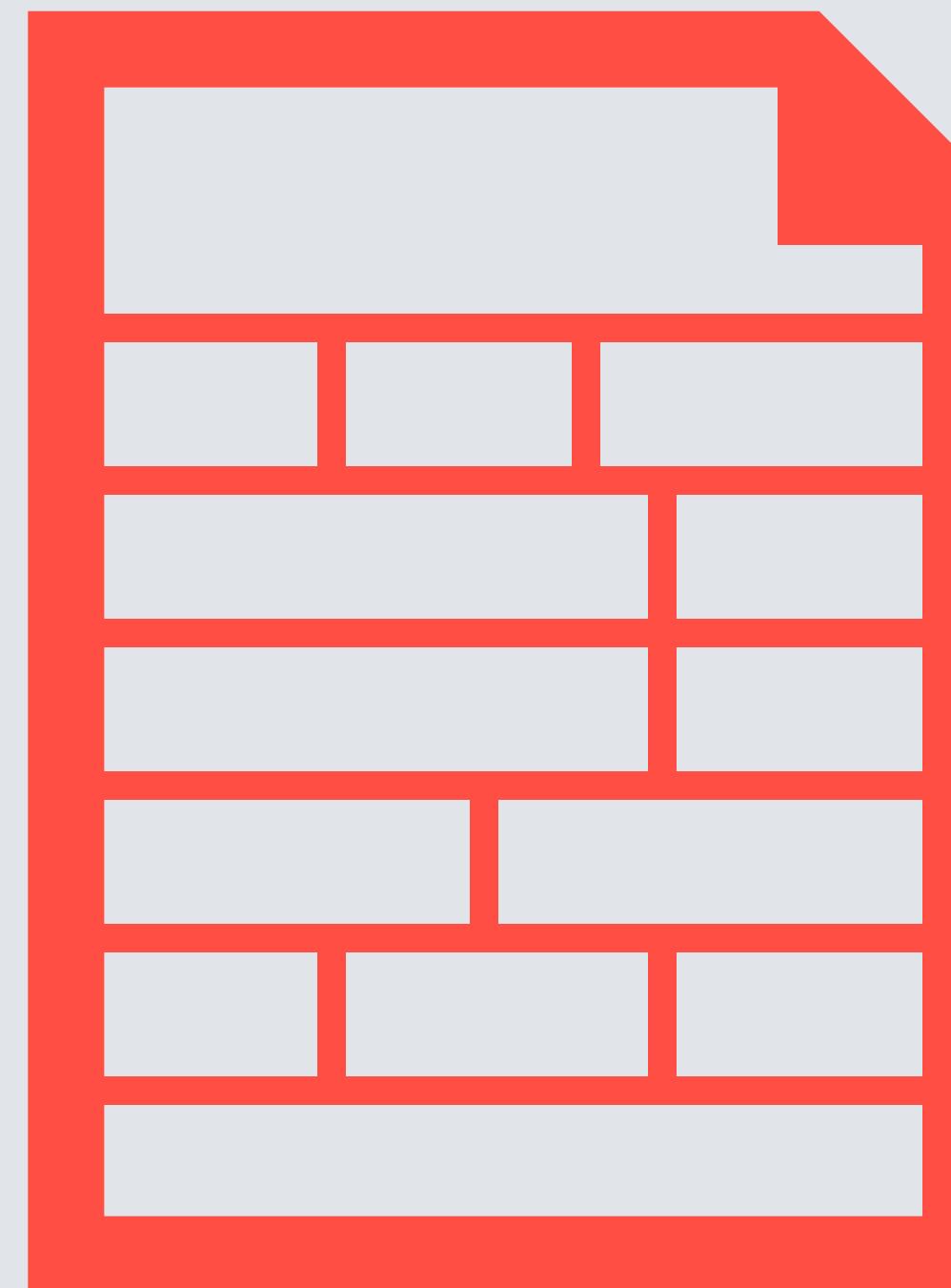


Alicja

Moja rada: Używaj dużo metaprogrammingu



Alicja

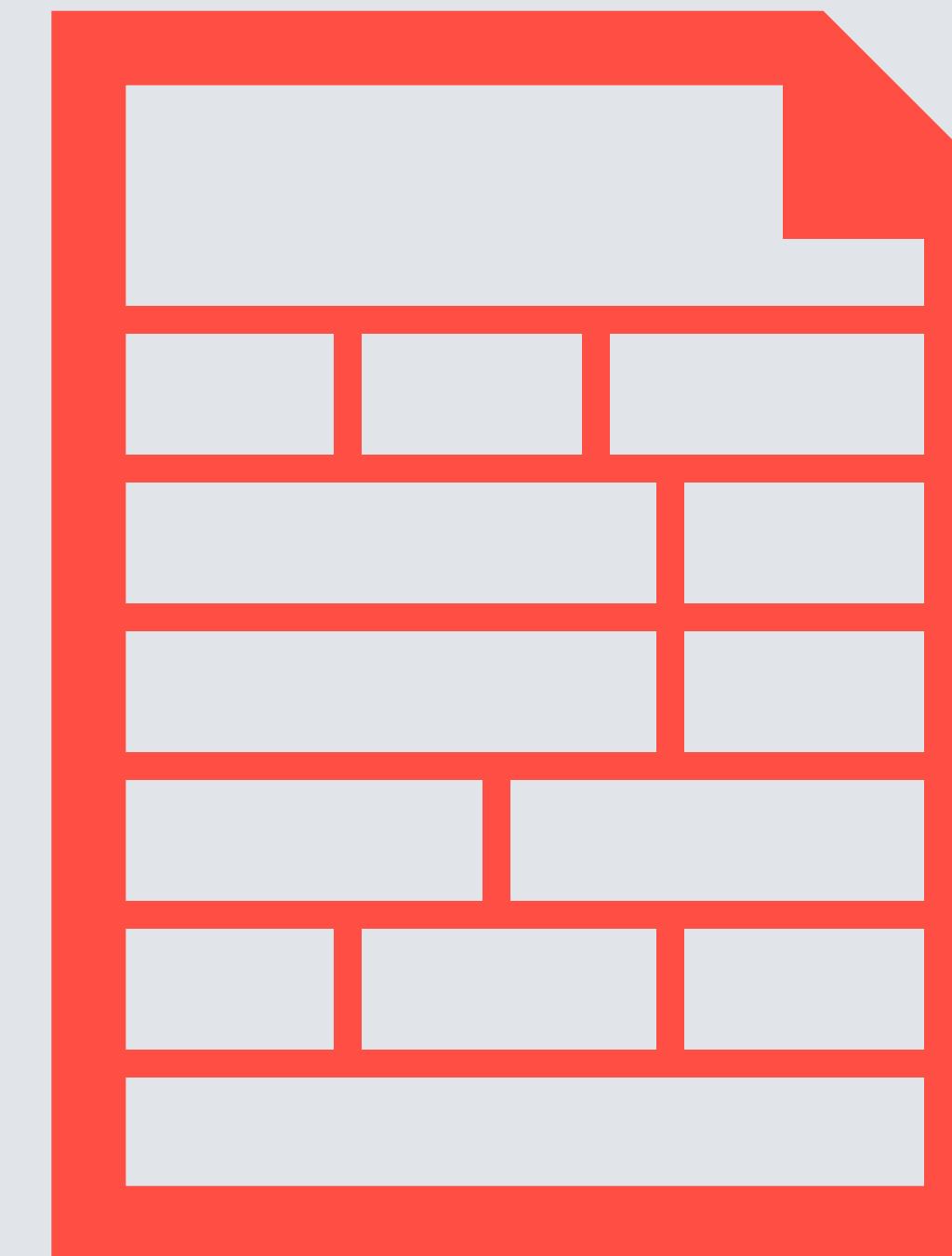


Kreator PIT

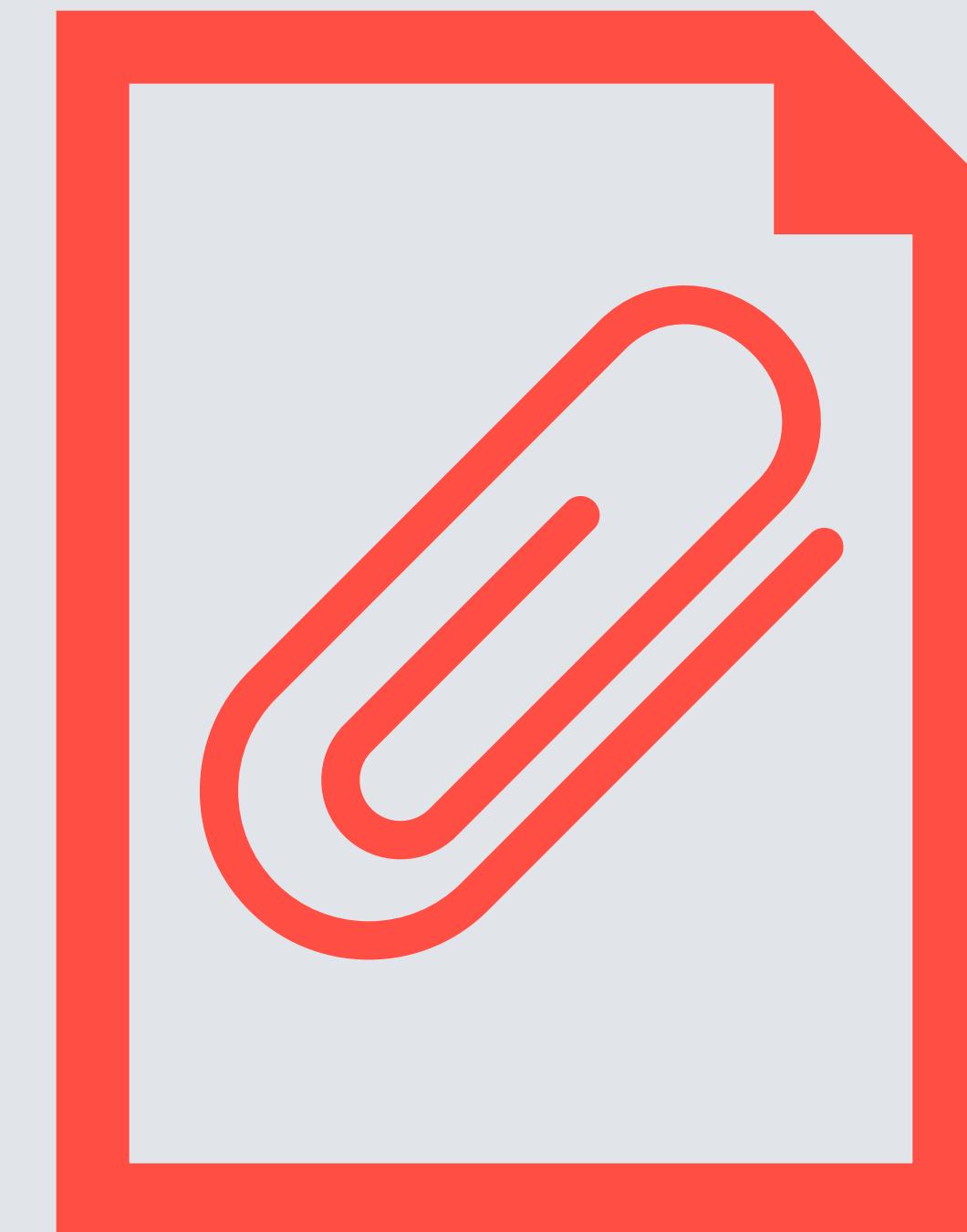
Moja rada: Używaj dużo metaprogrammingu



Alicja

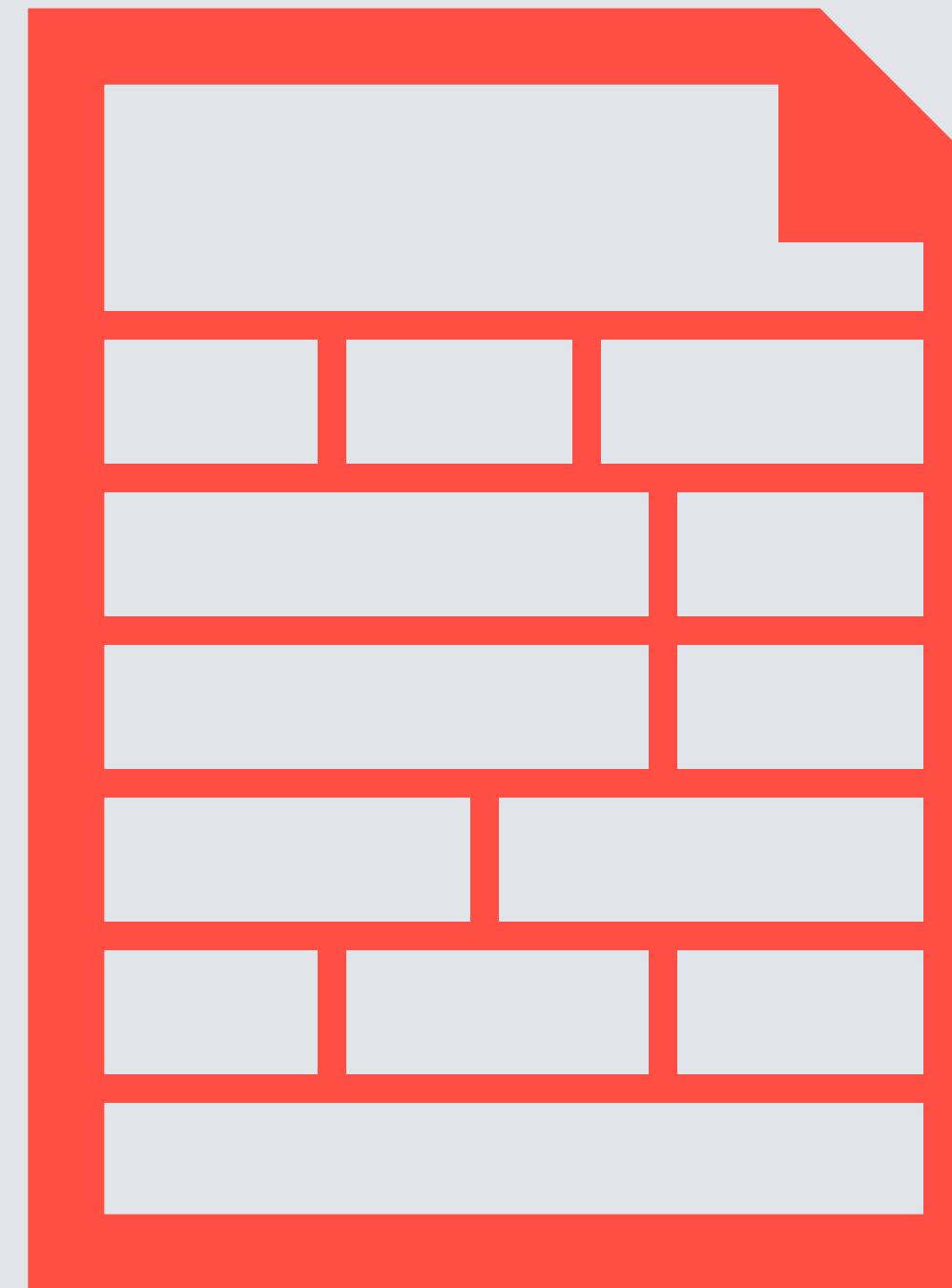
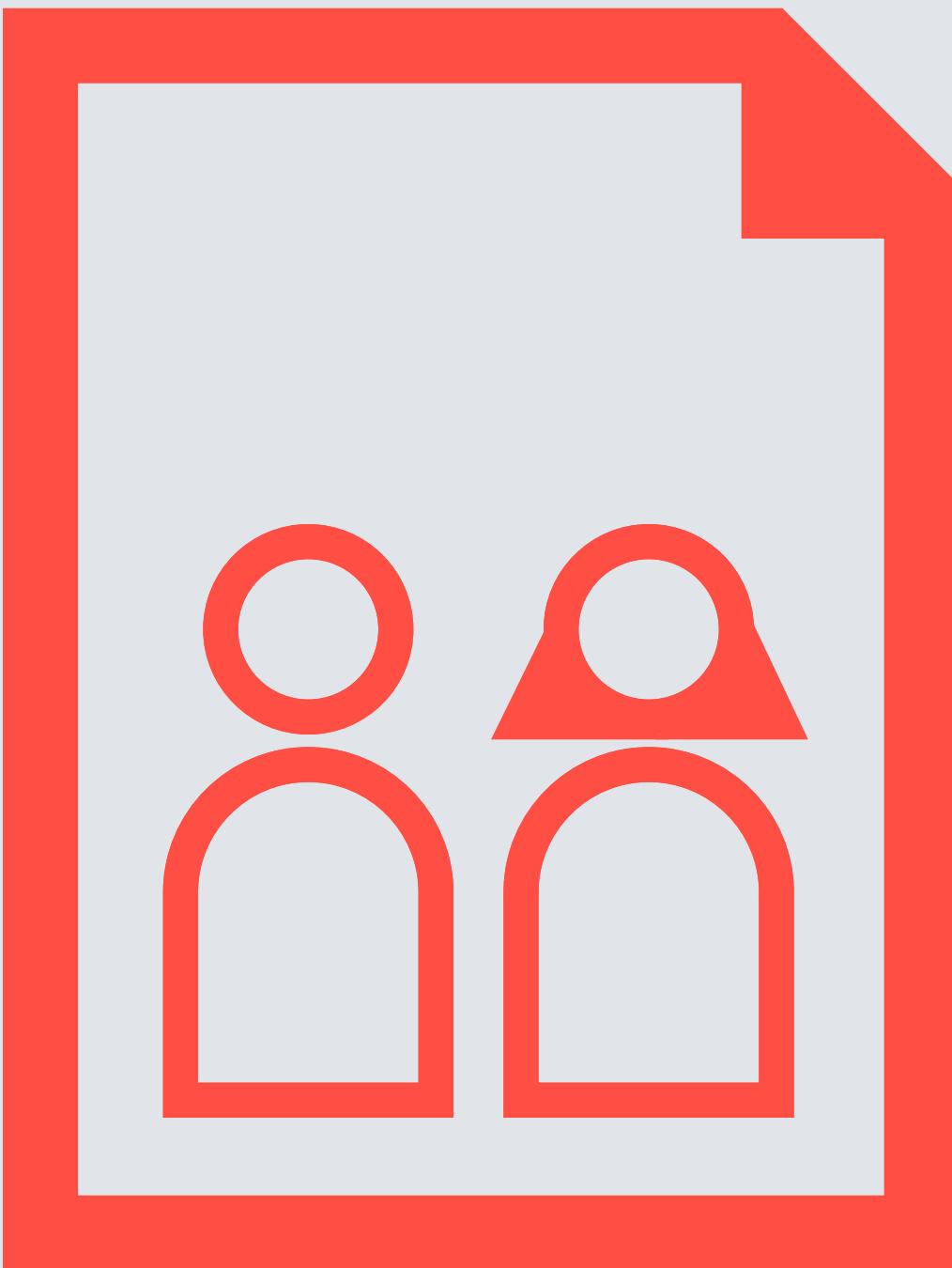


Kreator PIT



Załączniki

Moja rada: Używaj dużo metaprogrammingu



Warianty
Alicja

Kreator PIT

Załączniki

Moja rada: Używaj dużo metaprogrammingu

Po co pisać blisko kilkaset podobnych metod?



Alicja

Moja rada: Używaj dużo metaprogramingu

```
1 def tax_advance_month1_sum
2   #...
3 end
4
5 def tax_advance_month1_sum_spouse
6   #...
7 end
8
9 def tax_advance_month2_sum
10 #...
11 end
12
13 def tax_advance_month2_sum_spouse
14 #...
15 end
16
17
```



Moja rada: Używaj dużo metaprogrammingu

Po co pisać blisko kilkaset podobnych metod?

define_method



Alicja

Moja rada: Używaj dużo metaprogramingu

```
1 [ "", "_spouse"].each do |sufix|
2   %w(1 2 3 4 5 6 7 8 9 10 11 12).each do |i|
3     define_method("tax_advance_month#{i}_sum#{sufix}") do
4       sum = send(:"tax_advance_activity_month#{i}#{sufix}")
5       if send(:"lease#{sufix}?" )
6         sum += send(:"lease_advance_month#{i}#{sufix}" )
7       end
8       return sum.to_money if send(:"other_tax_advance_total_summary#{sufix}?" )
9       if send(:"second_activity_present#{sufix}?" )
10         sum += send(:"other_tax_advance_activity_month#{i}#{sufix}" )
11       end
12       if send(:"first_company_present#{sufix}?" )
13         sum += send(:"first_company_month#{i}#{sufix}" )
14       end
15       if send(:"second_company_present#{sufix}?" )
16         sum += send(:"second_company_month#{i}#{sufix}" )
17       end
18       if send(:"third_company_present#{sufix}?" )
19         sum += send(:"third_company_month#{i}#{sufix}" )
20       end
21       sum.to_money
22     end
23   end
24 end
```



Moja rada: Używaj dużo metaprogrammingu

```
1 [ "", "_spouse"].each do |sufix|
2   %w(1 2 3 4 5 6 7 8 9 10 11 12).each do |i|
3     define_method("tax_advance_month#{i}_sum#{sufix}") do
4       sum = send(:"tax_advance_activity_month#{i}#{sufix}")
5       if send(:"lease#{sufix}?")
6         sum += send(:"lease_advance_month#{i}#{sufix}")
7       end
8       return sum.to_money if send(:"other_tax_advance_total_summary#{sufix}?")
9       if send(:"second_activity_present#{sufix}?")
10         sum += send(:"other_tax_advance_activity_month#{i}#{sufix}")
11       end
12       if send(:"first_company_present#{sufix}?")
13         sum += send(:"first_company_month#{i}#{sufix}")
14       end
15       if send(:"second_company_present#{sufix}?")
16         sum += send(:"second_company_month#{i}#{sufix}")
17       end
18       if send(:"third_company_present#{sufix}?")
19         sum += send(:"third_company_month#{i}#{sufix}")
20       end
21       sum.to_money
22     end
23   end
24 
```



Moja rada: Używaj dużo metaprogramingu

```
1 [ "", "_spouse" ].each do |sufix|
2   %w(1 2 3 4 5 6 7 8 9 10 11 12).each do |i|
3     define_method("tax_advance_month#{i}_sum#{sufix}") do
4       sum = send(:"tax_advance_activity_month#{i}#{sufix}")
5       if send(:"lease#{sufix}?" )
6         sum += send(:"lease_advance_month#{i}#{sufix}" )
7       end
8       return sum.to_money if send(:"other_tax_advance_total_summary#{sufix}?" )
9       if send(:"second_activity_present#{sufix}?" )
10        sum += send(:"other_tax_advance_activity_month#{i}#{sufix}" )
11      end
12      if send(:"first_company_present#{sufix}?" )
13        sum += send(:"first_company_month#{i}#{sufix}" )
14      end
15      if send(:"second_company_present#{sufix}?" )
16        sum += send(:"second_company_month#{i}#{sufix}" )
17      end
18      if send(:"third_company_present#{sufix}?" )
19        sum += send(:"third_company_month#{i}#{sufix}" )
20      end
21      sum.to_money
22    end
23  end
```



Moja rada: Używaj dużo metaprogramingu

```
1 [ "", "_spouse" ].each do |sufix|
2   %w(1 2 3 4 5 6 7 8 9 10 11 12).each do |i|
3     define_method("tax_advance_month#{i}_sum#{sufix}") do
4       sum = send(:"tax_advance_activity_month#{i}#{sufix}")
5       if send(:"lease#{sufix}?")
6         sum += send(:"lease_advance_month#{i}#{sufix}")
7       end
8       return sum.to_money if send(:"other_tax_advance_total_summary#{sufix}?"')
9       if send(:"second_activity_present#{sufix}?"')
10         sum += send(:"other_tax_advance_activity_month#{i}#{sufix}")
11       end
12       if send(:"first_company_present#{sufix}?"')
13         sum += send(:"first_company_month#{i}#{sufix}")
14       end
15       if send(:"second_company_present#{sufix}?"')
16         sum += send(:"second_company_month#{i}#{sufix}")
17       end
18       if send(:"third_company_present#{sufix}?"')
19         sum += send(:"third_company_month#{i}#{sufix}")
20       end
21       sum.to_money
22     end
23   end
24 end
```



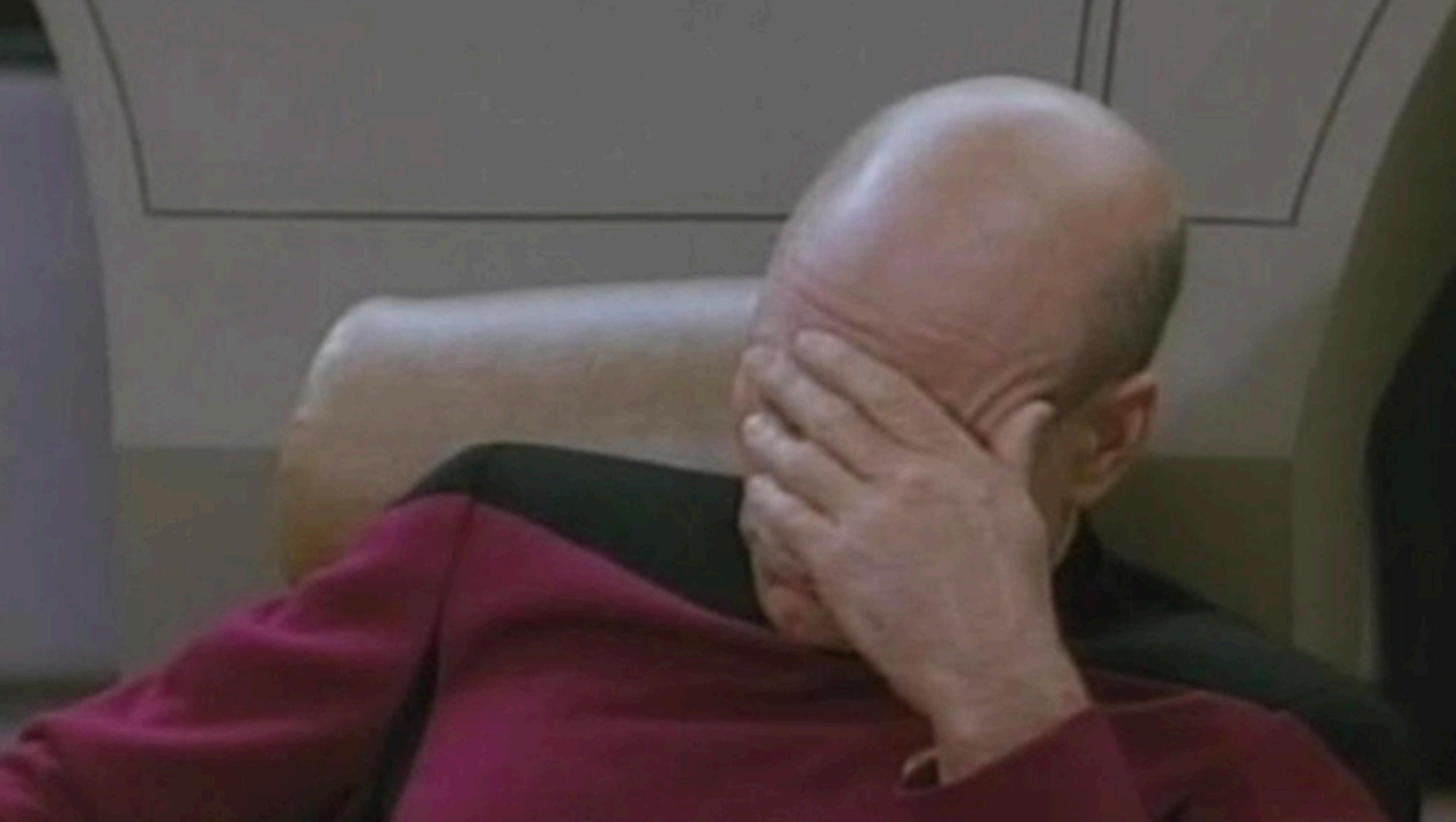
Moja rada: Używaj dużo metaprogrammingu



Lecimy na produkcję



Alicja



Moja rada: Używaj dużo metaprogrammingu

Wartość w polu „160 dla współmałżonka” liczy się niepoprawnie.



Alicja

Moja rada: Używaj dużo metaprogrammingu

Jak znaleźć tę metodę?



Alicja

Moja rada: Używaj dużo metaprogrammingu

Jak znaleźć tę metodę?

grep tax_advance_p160_sum_spouse



Alicja

Moja rada: Używaj dużo metaprogrammingu

Jak znaleźć tę metodę?

grep tax_advance_p160_sum_spouse

define_method(„tax_advance_p#{i}sum#{suffix}”)



Alicja

good
luck.

Moja rada: Używaj dużo metaprogrammingu

Debugujemy...



Alicja

Moja rada: Używaj dużo metaprogrammingu

Debugujemy...

Poprawka: 1 min.



Alicja

Moja rada: Używaj dużo metaprogrammingu

Debugujemy...

Poprawka: 1 min.

Znalezienie metody: !&%^ min.



Alicja

Moja lekcja:

Ostrożnie używajmy
metaprogramowania



Alicja

Moja lekcja:

Ostrożnie używajmy
metaprogramowania

Pozostawiajmy komentarz z pełną
nazwą metody, żeby później
łatwiej ją znaleźć



Alicja

18



Jaka jest Twoja najgorsza rada?





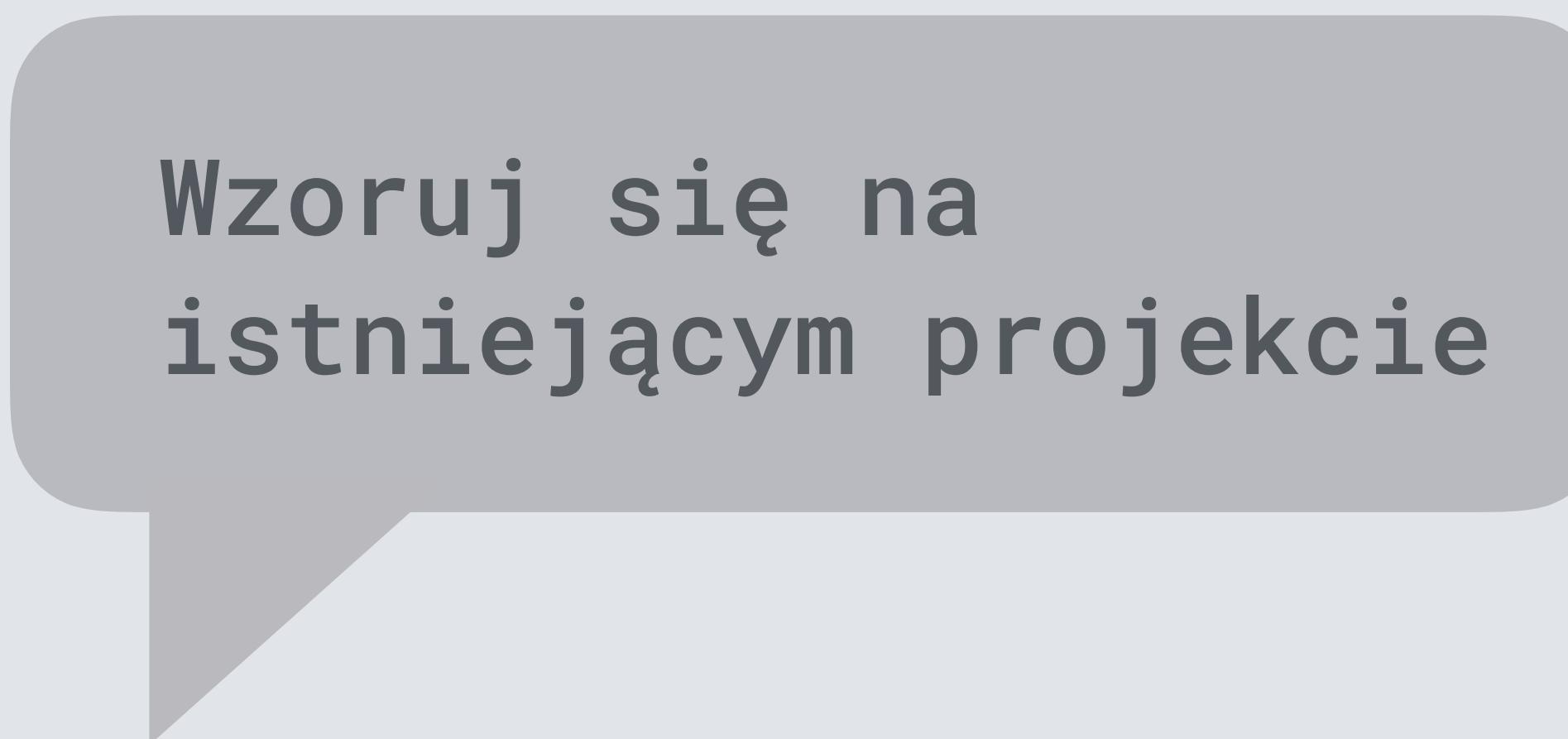
One more thing...

Najgorsze rady dla developera Ruby / Rails

Nasze dzisiejsze „rady”

Najgorsze rady dla developera Ruby / Rails

Nasze dzisiejsze „rady”



Wzoruj się na
istniejącym projekcie

Najgorsze rady dla developera Ruby / Rails

Nasze dzisiejsze „rady”

Wzoruj się na
istniejącym projekcie

Nie trzymaj stałych
elementów w
configach.

Najgorsze rady dla developera Ruby / Rails

Nasze dzisiejsze „rady”

Wzoruj się na
istniejącym projekcie

Po prostu wepnij
gem, on to ogarnie!

Nie trzymaj stałych
elementów w
configach.

Najgorsze rady dla developera Ruby / Rails

Nasze dzisiejsze „rady”

Wzoruj się na
istniejącym projekcie

Nie trzymaj stałych
elementów w
configach.

Po prostu wepnij
gem, on to ogarnie!

Używaj kodu Ruby
w widokach bez
ograniczeń!

Najgorsze rady dla developera Ruby / Rails

Nasze dzisiejsze „rady”

Wzoruj się na
istniejącym projekcie

Nie trzymaj stałych
elementów w
configach.

Po prostu wepnij
gem, on to ogarnie!

Używaj kodu Ruby
w widokach bez
ograniczeń!

Najgorsze rady dla developera Ruby / Rails

Nasze dzisiejsze „rady”

Wzoruj się na
istniejącym projekcie

Nie trzymaj stałych
elementów w
configach.

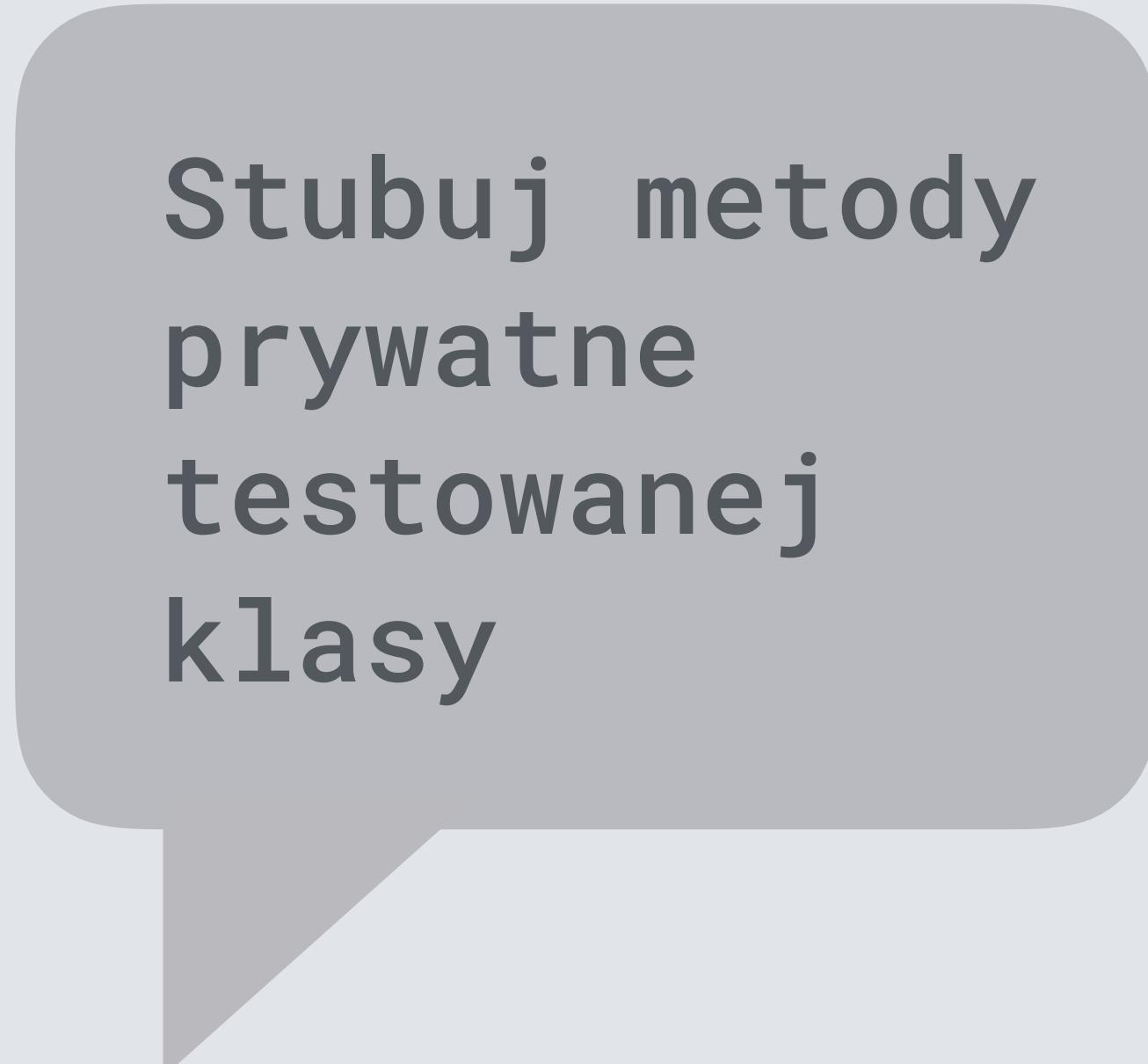
Po prostu wepnij
gem, on to ogarnie!

Używaj kodu Ruby
w widokach bez
ograniczeń!

Trzymaj logikę
w kontrolerach

Najgorsze rady dla developera Ruby / Rails

Nasze dzisiejsze „rady”



Stubuj metody
prywatne
testowanej
klasy

Najgorsze rady dla developera Ruby / Rails

Nasze dzisiejsze „rady”

Stubuj metody
prywatne
testowanej
klasy

Używaj rescue nil

Najgorsze rady dla developera Ruby / Rails

Nasze dzisiejsze „rady”

Stubuj metody
prywatne
testowanej
klasy

Używaj rescue nil

Używaj Callbacków Active
Record do logiki
biznesowej

Najgorsze rady dla developera Ruby / Rails

Nasze dzisiejsze „rady”

Stubuj metody prywatne testowanej klasy

Używaj rescue nil

Używaj Callbacków Active Record do logiki biznesowej

Używaj dużo metaprogramingu

Najgorsze rady dla developera Ruby / Rails

Mielimy więcej pomysłów ;)

Najgorsze rady dla developera Ruby / Rails

Mielimy więcej pomysłów ;)

- Używaj HAML i CoffeeScript.

Najgorsze rady dla developera Ruby / Rails

Mielimy więcej pomysłów ;)

- Używaj HAML i CoffeeScript.
- Nie przejmuj się, że aplikacja wolno działa.
Ruby jest wolny.

Najgorsze rady dla developera Ruby / Rails

Mielimy więcej pomysłów ;)

- Używaj HAML i CoffeeScript.
- Nie przejmuj się, że aplikacja wolno działa.
Ruby jest wolny.
- Używaj default_scope.

Najgorsze rady dla developera Ruby / Rails

Mielimy więcej pomysłów ;)

- Używaj HAML i CoffeeScript.
- Nie przejmuj się, że aplikacja wolno działa.
Ruby jest wolny.
- Używaj default_scope.
- Zrób z tego gema.

Najgorsze rady dla developera Ruby / Rails

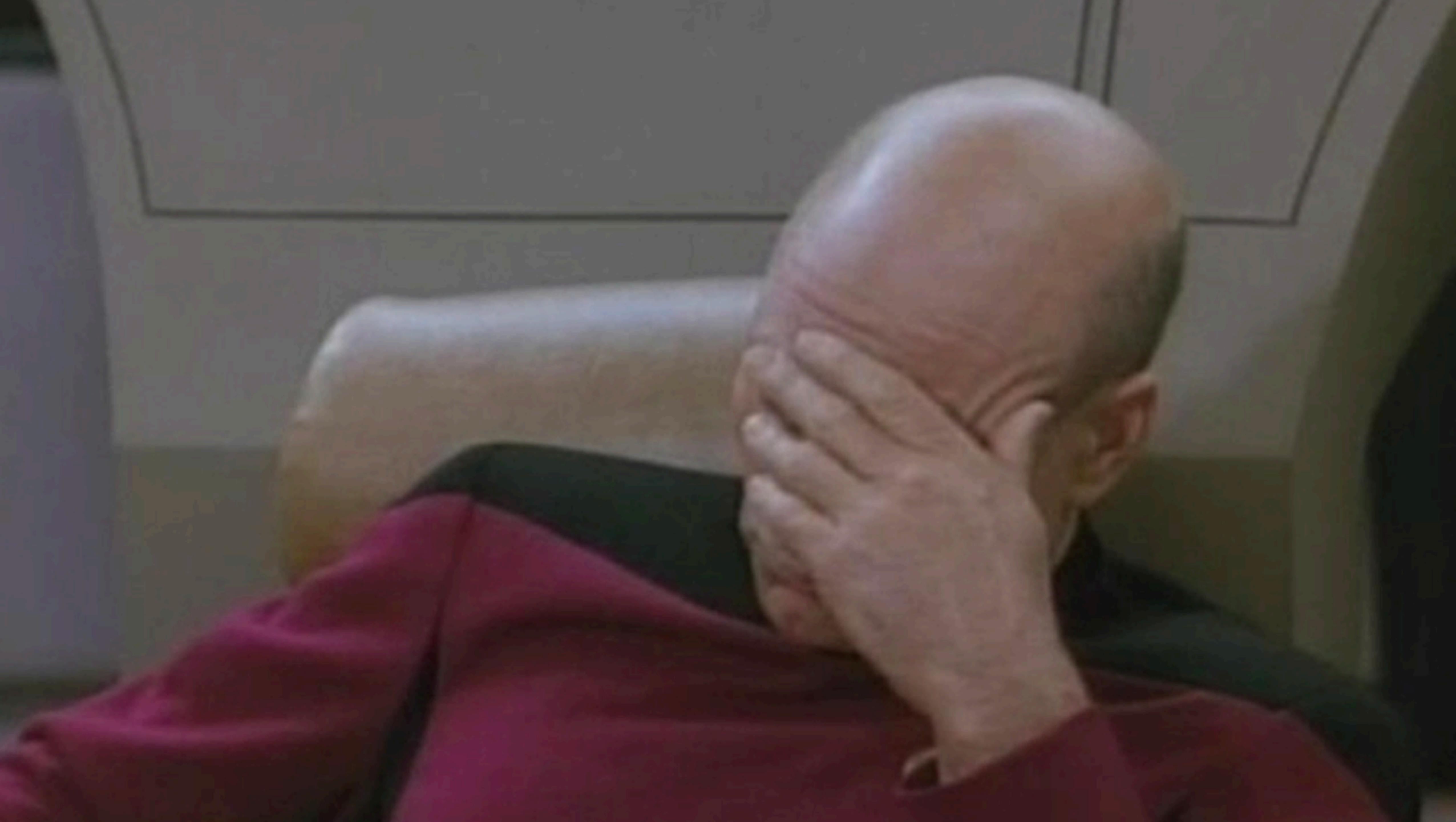
Mielimy więcej pomysłów ;)

- Używaj HAML i CoffeeScript.
- Nie przejmuj się, że aplikacja wolno działa.
Ruby jest wolny.
- Używaj default_scope.
- Zrób z tego gema.
- Używaj Railsów do wszystkiego.

Najgorsze rady dla developera Ruby / Rails

Mielimy więcej pomysłów ;)

- Używaj HAML i CoffeeScript.
- Nie przejmuj się, że aplikacja wolno działa.
Ruby jest wolny.
- Używaj default_scope.
- Zrób z tego gema.
- Używaj Railsów do wszystkiego.
- ...



Dziękujemy!