

Stefan Daschek / @noniq@chaos.social

Using live request data for testing while upgrading a Rails app

About me

- ▶ Stefan Daschek (aka noniq)

About me

- ▶ Stefan Daschek (aka noniq)
- ▶ studied Computer Sciences (TU Wien)

About me

- ▶ Stefan Daschek (
- ▶ studied Computer

4 Framework

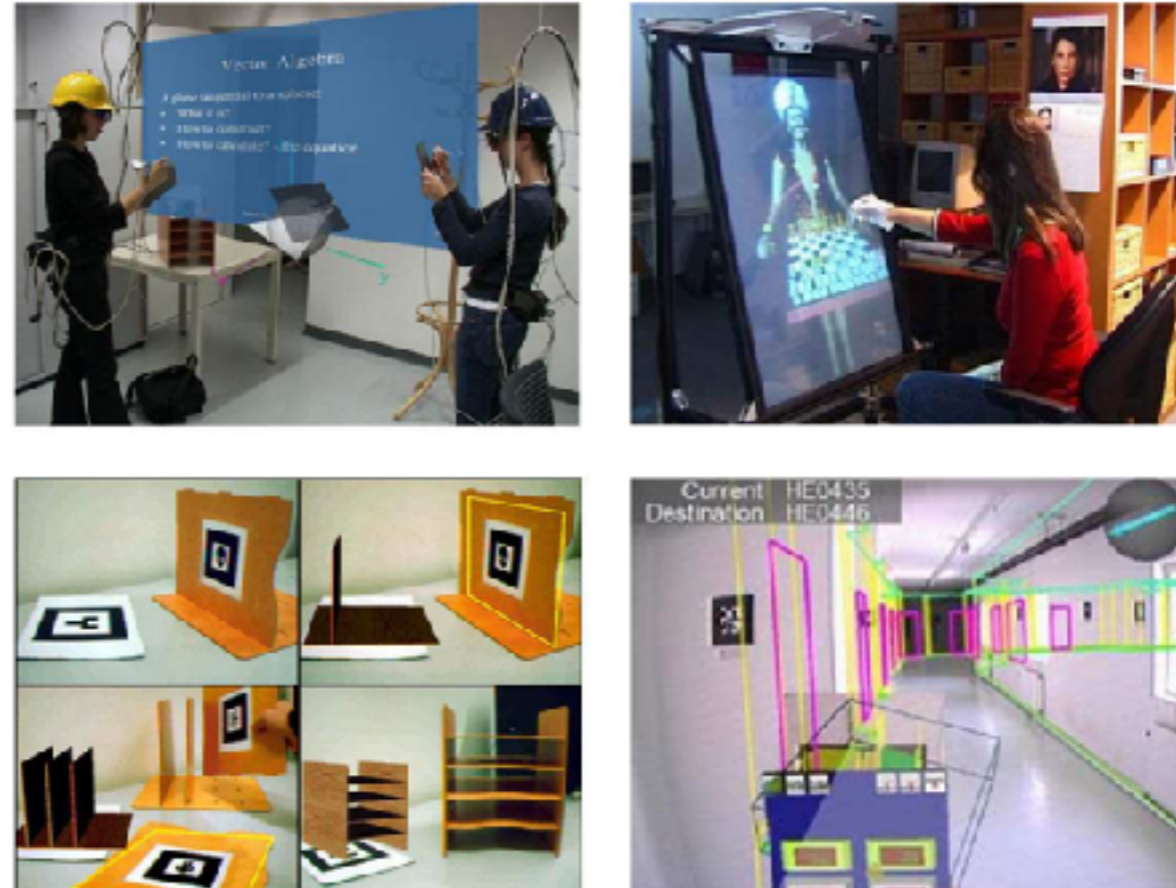


Abbildung 4.6: Studierstube-Applikationen. Von links oben im Uhrzeigersinn: Construct3D, Der Türkische Schachspieler, SignPost, AEKI

Reality-Animationen¹¹.

- *SignPost* leitet – mit einem mobilen Augmented-Reality-Setup ausgerüstete – Personen durch ihnen unbekannte Gebäude¹².

Abbildung 4.6 zeigt Beispiele der genannten Applikationen.

4.2 Open Tracker

*Open Tracker*¹³ ist ein unter der LGPL¹⁴ verfügbares flexibles Tracking-Framework. Es erlaubt die Definition eines beliebigen *Datenfluss-Netzwerks* (Abbildung 4.7), repräsentiert durch einen gerichteten Graph aus Knoten dreier Typen:

About me

- ▶ Stefan Daschek (
- ▶ studied Computer

4 Framework

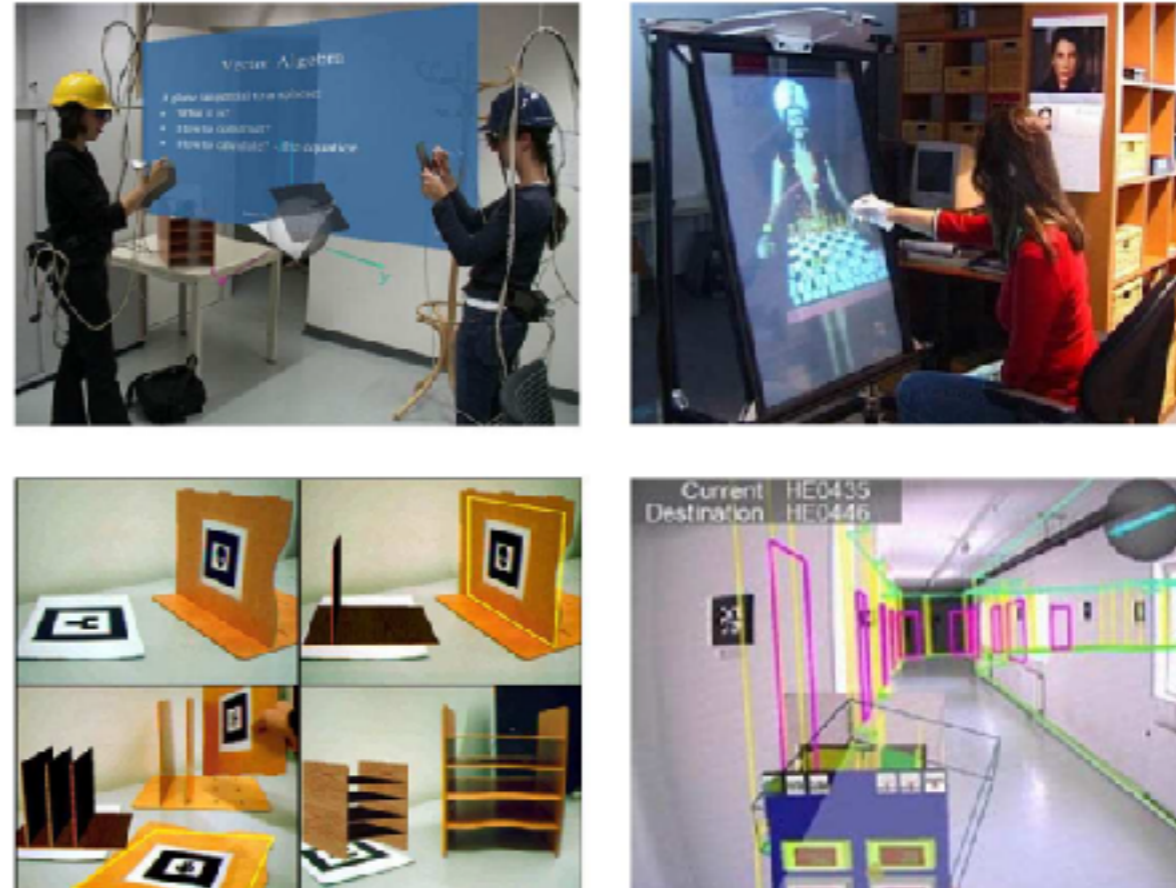


Abbildung 4.6: Studierstube-Applikationen. Von links oben im Uhrzeigersinn: Construct3D, Der Türkische Schachspieler, SignFinder, EKI

Reality-Animationen¹¹.

- SignFinder – mit einem mobilen Augmented-Reality-Setup ausgerüstete – Personen bekannte Gebäude¹².

Abbildung 4.6 zeigt Beispiele der genannten Applikationen.

4.2 Open Tracker

Open Tracker¹³ ist ein unter der LGPL¹⁴ verfügbares flexibles Tracking-Framework. Es erlaubt die Definition eines beliebigen Datenfluss-Netzwerks (Abbildung 4.7), repräsentiert durch einen gerichteten Graph aus Knoten dreier Typen:

(The state of Augmented Reality in 2005)

About me

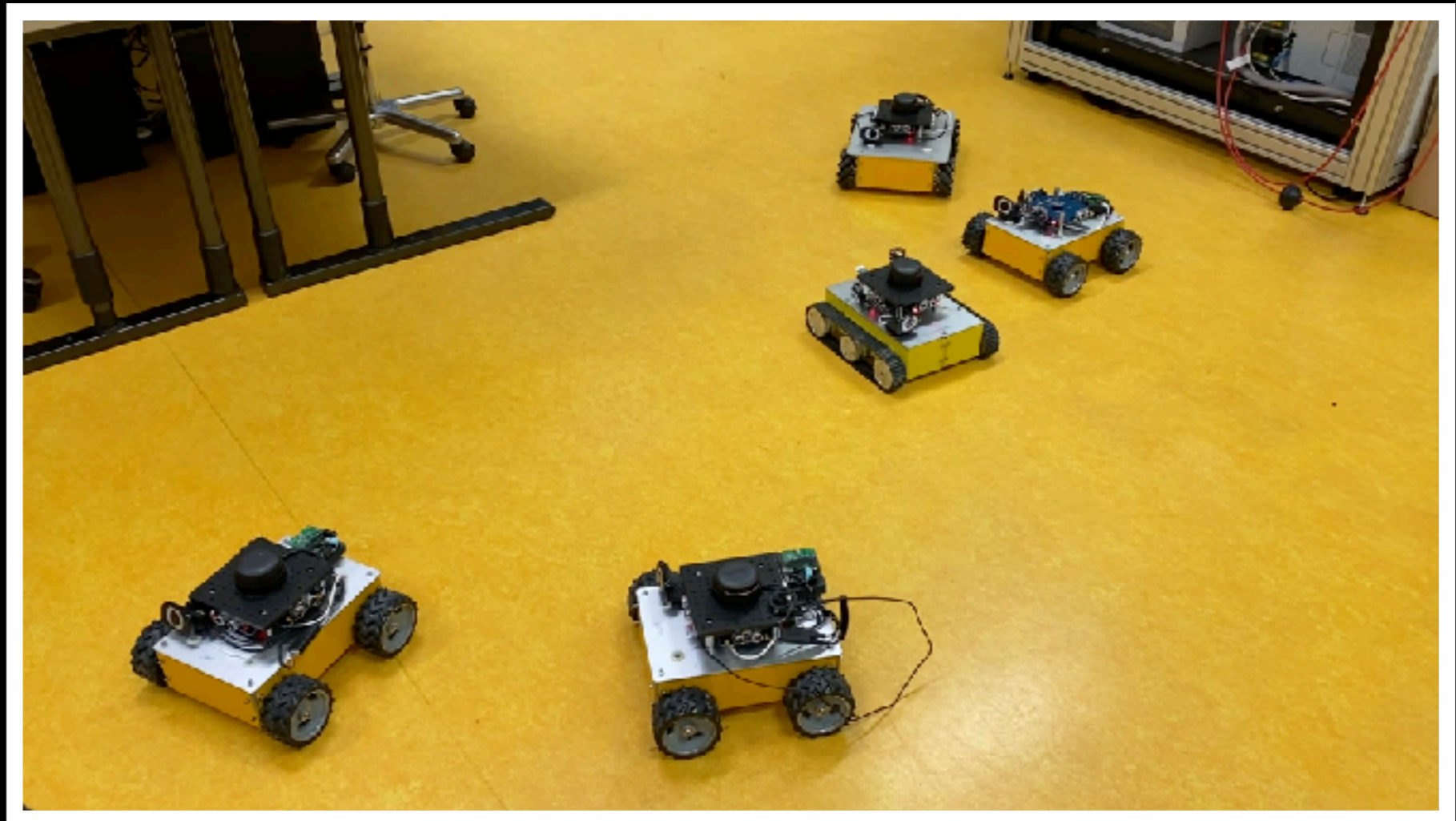
- ▶ Stefan Daschek (aka noniq)
- ▶ studied Computer Sciences (TU Wien)

About me

- ▶ Stefan Daschek (aka noniq)
- ▶ studied Computer Sciences (@ TU Wien)
- ▶ teaching Mobile Robotics (@ FH Wr. Neustadt)

About me

- ▶ Stefan Daschek (aka noniq)
- ▶ studied Computer Sciences (@ TU Wien)
- ▶ teaching Mobile Robotics (@ FH Wr. Neustadt)



About me

- ▶ Stefan Daschek (aka noniq)
- ▶ studied Computer Sciences (@ TU Wien)
- ▶ teaching Mobile Robotics (@ FH Wr. Neustadt)

About me

- ▶ Stefan Daschek (aka noniq)
- ▶ studied Computer Sciences (@ TU Wien)
- ▶ teaching Mobile Robotics (@ FH Wr. Neustadt)

About me



About me

- ▶ Stefan Daschek (aka noniq)
- ▶ studied Computer Sciences (@ TU Wien)
- ▶ teaching Mobile Robotics (@ FH Wr. Neustadt)

About me

- ▶ Stefan Daschek (aka noniq)
- ▶ studied Computer Sciences (@ TU Wien)
- ▶ teaching Mobile Robotics (@ FH Wr. Neustadt)
- ▶ owning one half of a two-person-company

About me

- ▶ Stefan Daschek (aka noniq)
- ▶ studied Computer Sciences (@ TU Wien)
- ▶ teaching Mobile Robotics (@ FH Wr. Neudorf)
- ▶ owning one half of a two-person-company


A THE ENTWURF

099 123456789 | 099 123456789 | 099 123456789

Was wir machen funktioniert.

Webapplikationen · Security-Testing · Elektronik

Wir sind ...




Stefan Daschek

Has a B.Sc. in Computer Science (TU Wien) and a M.Sc. in Information Systems (FH Wr. Neudorf). He is currently working as a Senior Software Engineer at a company in Vienna. He is also a member of the Austrian National Cyber Security Centre (ANSS) and a frequent speaker at conferences. He is also a member of the Austrian National Cyber Security Centre (ANSS) and a frequent speaker at conferences.

LINKS: [www.stefan-daschek.com](#)

🐙 🐙 🐙



Moritz Kobrna

Has a B.Sc. in Computer Science (TU Wien) and a M.Sc. in Information Systems (FH Wr. Neudorf). He is currently working as a Senior Software Engineer at a company in Vienna. He is also a member of the Austrian National Cyber Security Centre (ANSS) and a frequent speaker at conferences. He is also a member of the Austrian National Cyber Security Centre (ANSS) and a frequent speaker at conferences.

LINKS: [www.moritz-kobrna.com](#)

🐙 🐙 🐙

About me

- ▶ Stefan Daschek (aka noniq)
- ▶ studied Computer Sciences (@ TU Wien)
- ▶ teaching Mobile Robotics (@ FH Wr. Neusiedl)
- ▶ owning one half of a two-person-company
- ❖ we build strange things

The screenshot shows a website with a green header. At the top left is the logo 'A' with 'THE' and 'ENTWURF' below it. At the top right is a navigation menu with 'Blog', 'Techlog', and 'Kontakt'. The main content area has a green background with the text 'Was wir machen funktioniert.' and 'Webapplikationen · Security-Testing · Elektronik'. Below this is a white section titled 'Wir sind ...' featuring two circular profile pictures of men, Stefan Daschek and Moritz Kobrna, with their names and short biographies below them. The page also includes a navigation menu at the top right with links for 'Blog', 'Techlog', and 'Kontakt'.

About me

▶ Stefan Daschek (aka noniq)

▶ st

▶ te

▶ ov

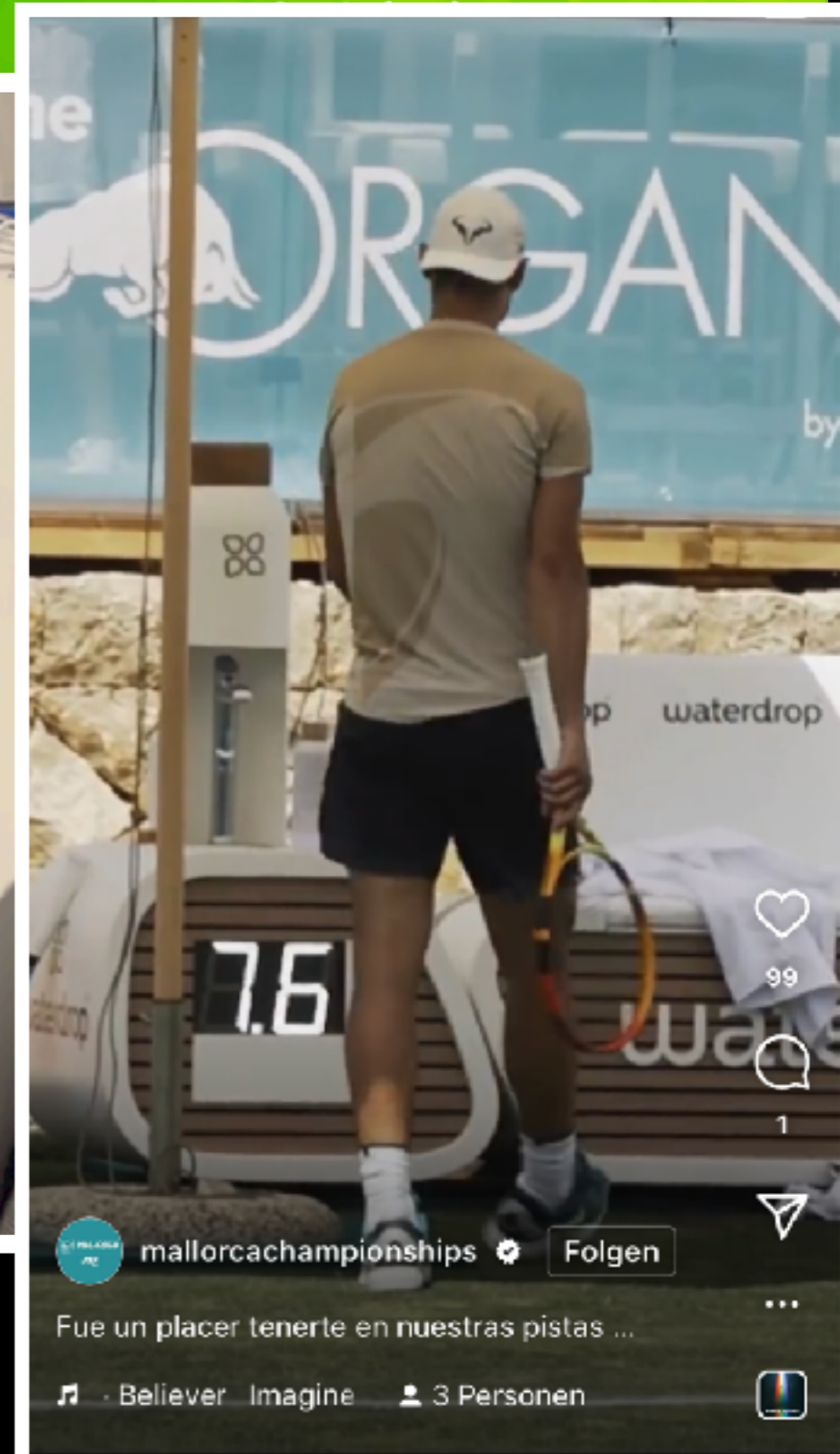
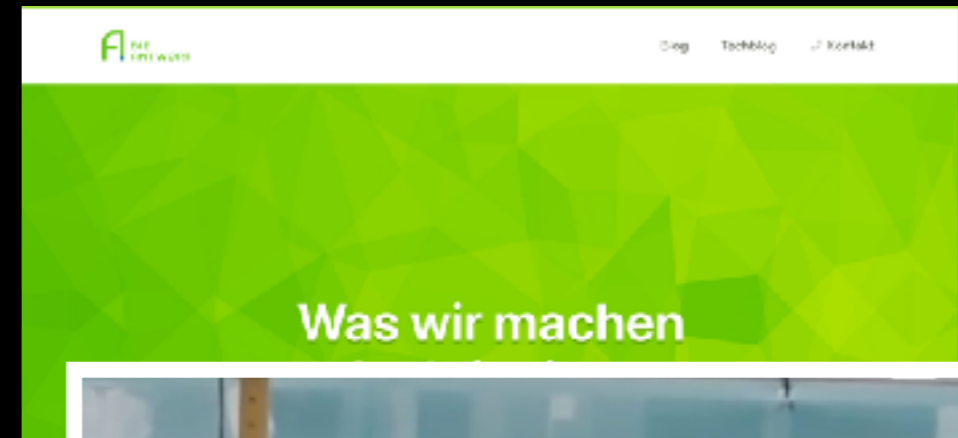
co



About me

▶ Stefan Daschek (aka noniq)

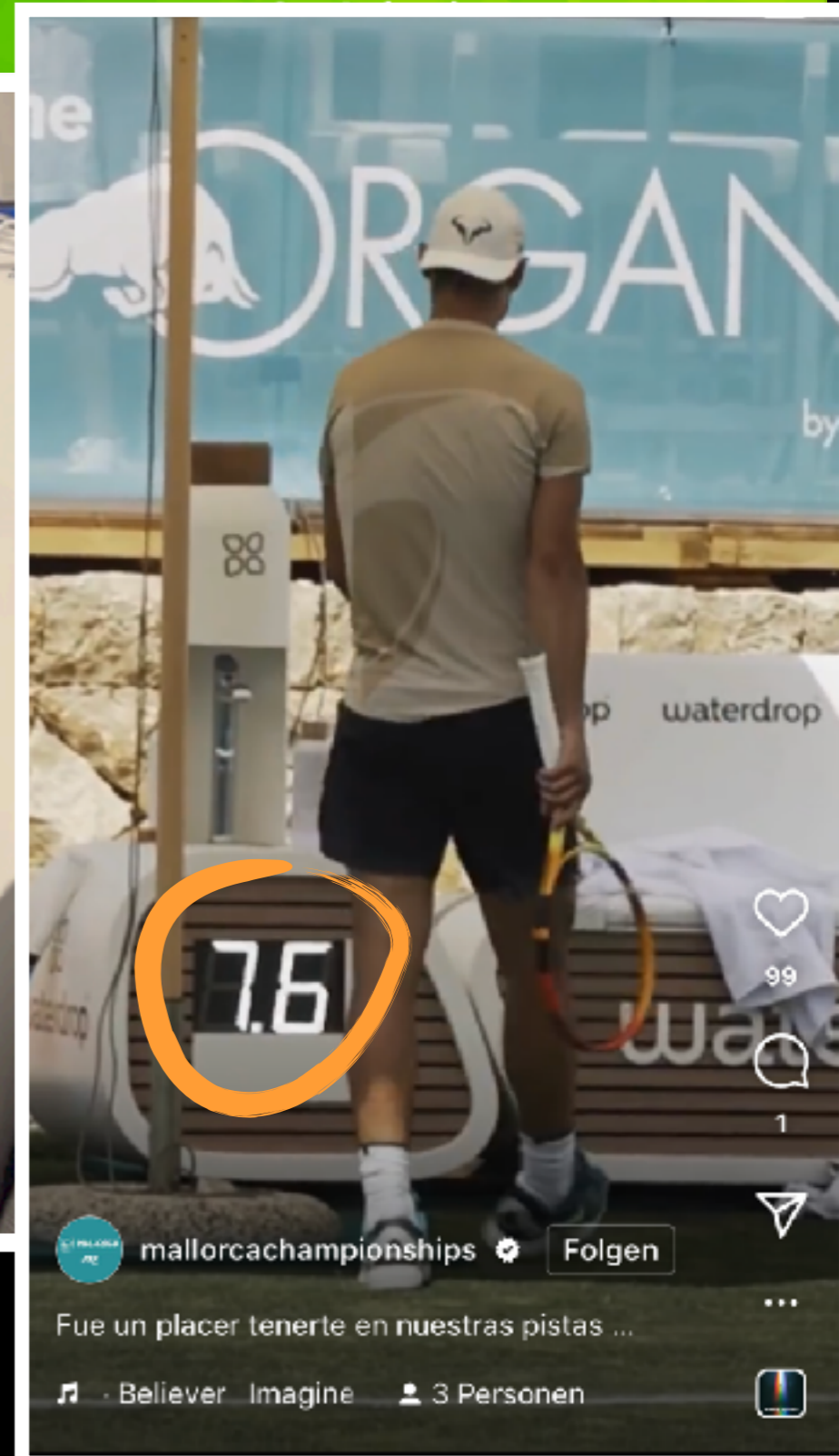
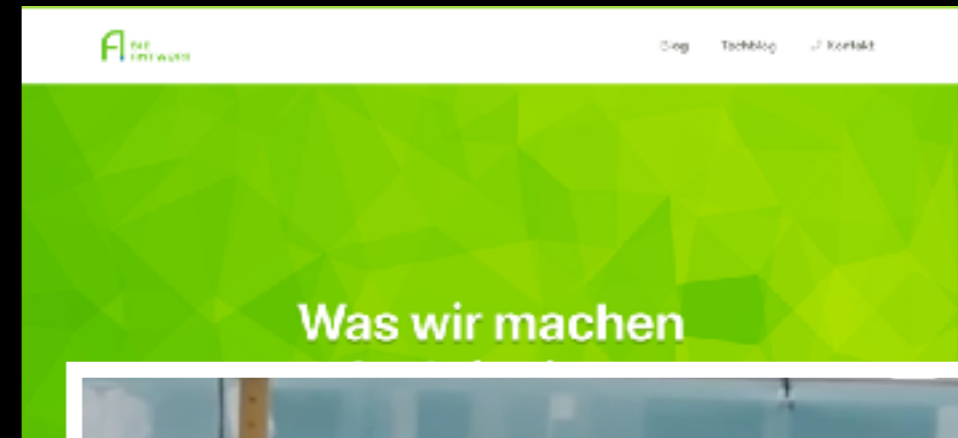
- ▶ st
 - ▶ te
 - ▶ ov
 - ▶ co
- ❖



About me

▶ Stefan Daschek (aka noniq)

- ▶ st
 - ▶ te
 - ▶ ov
 - ▶ co
- ❖



About me

- ▶ Stefan Daschek (aka noniq)
- ▶ studied Computer Sciences (@ TU Wien)
- ▶ teaching Mobile Robotics (@ FH Wr. Neudorf)
- ▶ owning one half of a two-person-company
- ❖ we build strange things


A THE ENTWURF

Blog TechBlog Kontakt

Was wir machen funktioniert.

Webapplikationen · Security-Testing · Elektronik


Wir sind ...



Stefan Daschek

hat sich in der Vergangenheit (1995) mit dem Web und anderen nicht mehr aufgedeckt zu programmieren. Kann seinen Perl/Python/Kursen ziemlich ganz gut im Zirkelbau. Hat die AN 1998/99 nach während seines Informatik Studiums 2004 gegründet. Bleibt sich als Hacker im ursprünglichen Sinn und engagiert sich in kleiner Projekten z.B. bei STASS/MACT/SCHEM UND [@stefan](#)

🐙 📄 📧



Moritz Kobrna

hat Erfahrung mit dem Web (aber hat seinen als Kind privat Telefonieren im Nachhinein verlegt. Konnt auf niedrige eine coole Serverprojekte produzieren. Ist als Server-Admin über mehrere Jahre für den Betrieb von Servern tätig. Gründet von Bedding-Store.

🐙 📄 📧

About me

- ▶ Stefan Daschek (aka noniq)
- ▶ studied Computer Sciences (@ TU Wien)
- ▶ teaching Mobile Robotics (@ FH Wr. Neusiedl)
- ▶ owning one half of a two-person-company
 - ❖ we build strange things
 - ❖ we also make Rails apps


A THE ENTWURF

Blog TechBlog Kontakt

Was wir machen funktioniert.

Webapplikationen · Security-Testing · Elektronik

Wir sind ...




Stefan Daschek

hat sich in der Vergangenheit (2002) mit dem Web und dem Web 2.0 auseinandergesetzt und dabei seine Leidenschaft für Programmieren, seine eigenen Perl/Python/JavaScript-Projekte und seine Fähigkeiten in der Webentwicklung (HTML, CSS, JavaScript) nach und nach in die Informatik-Studienszene 2004 eingebracht. Er ist sich als Hacker im ursprünglichen Sinne und engagiert sich in der Open-Source-Bewegung (z.B. bei Drupal, Drupal.org, Drupal.de).

UPD: [@stefan.niq](#)

🐙 🐙 🐙 🐙



Moritz Kobrna

hat sich in der Vergangenheit (2002) mit dem Web und dem Web 2.0 auseinandergesetzt und dabei seine Leidenschaft für Programmieren, seine eigenen Perl/Python/JavaScript-Projekte und seine Fähigkeiten in der Webentwicklung (HTML, CSS, JavaScript) nach und nach in die Informatik-Studienszene 2004 eingebracht. Er ist sich als Hacker im ursprünglichen Sinne und engagiert sich in der Open-Source-Bewegung (z.B. bei Drupal, Drupal.org, Drupal.de).

UPD: [@moritz.kobrna](#)

🐙 🐙 🐙 🐙

About me

- ▶ Stefan Daschek (aka noniq)
- ▶ studied Computer Sciences (@ TU Wien)
- ▶ teaching Mobile Robotics (@ FH Wr. Neufeld)
- ▶ owning one half of a two-person-company
 - ❖ we build strange things
 - ❖ we also make Rails apps
 - ➔ we started our first Rails project in January 2006


A THE ENTWURF

Blog TechBlog Kontakt

Was wir machen funktioniert.

Webapplikationen · Security-Testing · Elektronik


Wir sind ...



Stefan Daschek

hat sich in der Vergangenheit (1995) mit dem Web beschäftigt und seitdem sich mehr auf die Entwicklung von Programmen konzentriert. Kann seinen Perl- und Python-Kenntnissen ganz gut im Web anwenden. Hat 1998-2001 an der TU Wien studiert und ist seitdem in der IT-Branche tätig. Seit 2004 ist er als Berater tätig und engagiert sich in der Open-Source-Bewegung. [LinkedIn](#)

🐙 🐙 🐙



Moritz Kobrna

hat sich in der Vergangenheit (1995) mit dem Web beschäftigt und seitdem sich mehr auf die Entwicklung von Programmen konzentriert. Kann seinen Perl- und Python-Kenntnissen ganz gut im Web anwenden. Hat 1998-2001 an der TU Wien studiert und ist seitdem in der IT-Branche tätig. Seit 2004 ist er als Berater tätig und engagiert sich in der Open-Source-Bewegung. [LinkedIn](#)

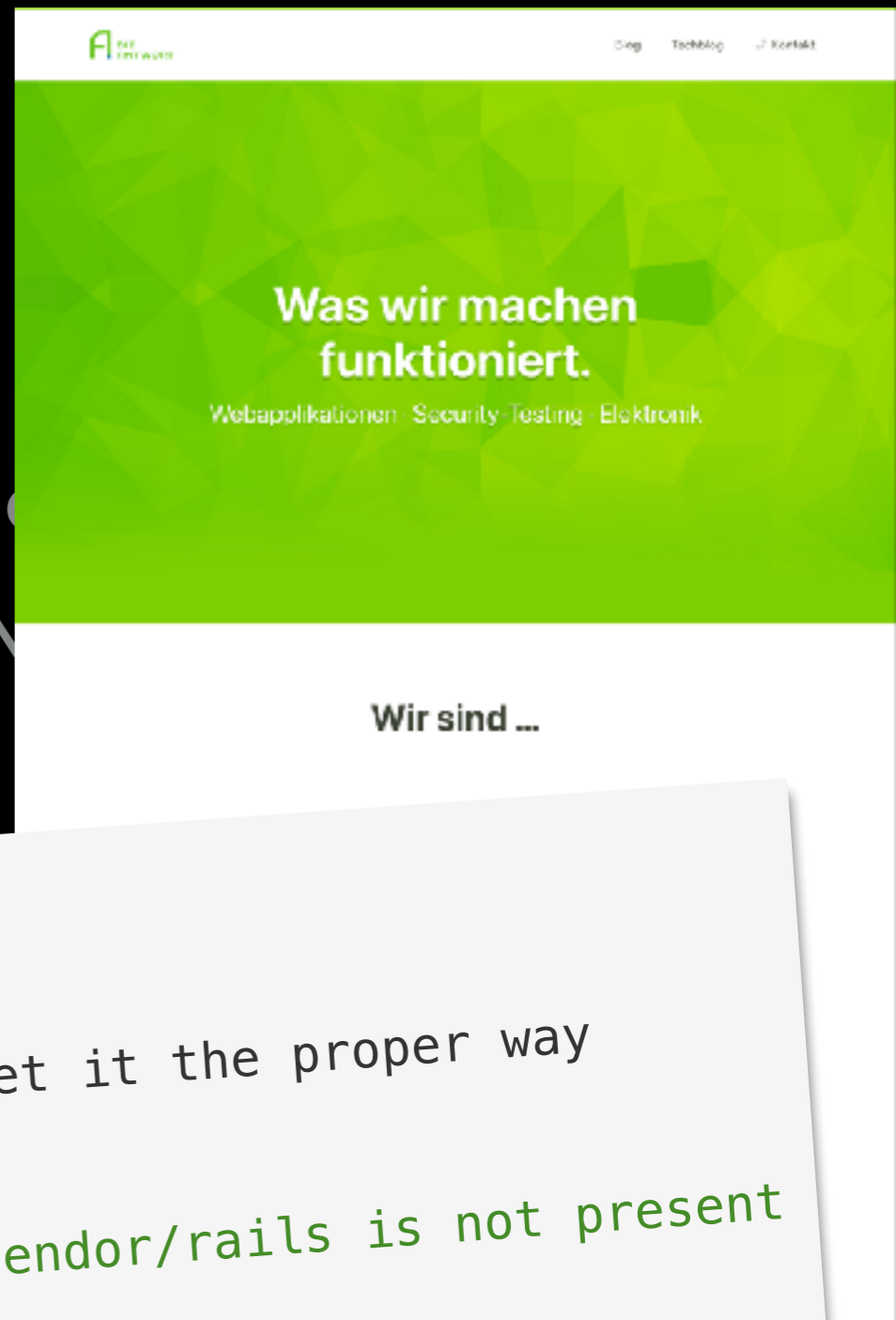
🐙 🐙 🐙

About me

- ▶ Stefan Daschek (aka noniq)
- ▶ studied Computer Sciences (@ TU Wien)
- ▶ teaching Mobile Robotics (@ FH Wr. Neufeld)
- ▶ owning one half of a two-person-company

```
--- config/environment.rb (revision 195)
+++ config/environment.rb (revision 196)
@@ -4,6 +4,9 @@
 # you don't control web/app server and can't set it the proper way
 # ENV['RAILS_ENV'] ||= 'production'

+# Specifies gem version of Rails to use when vendor/rails is not present
+RAILS_GEM_VERSION = '1.1.4'
+
# Bootstrap the Rails environment, frameworks, and default configuration
require File.join(File.dirname(__FILE__), 'boot')
```



About me

- ▶ Stefan Daschek (aka noniq)
- ▶ studied Computer Sciences (@ TU W
- ▶ teaching Mobile Robotics (@ FH Wr. N
- ▶ owning one half of a two-person-company

Remember svn / subversion?

```
--- config/environment.rb (revision 195)
+++ config/environment.rb (revision 196)
@@ -4,6 +4,9 @@
 # you don't control web/app server and can't set it the proper way
 # ENV['RAILS_ENV'] ||= 'production'

+# Specifies gem version of Rails to use when vendor/rails is not present
+RAILS_GEM_VERSION = '1.1.4'
+
# Bootstrap the Rails environment, frameworks, and default configuration
require File.join(File.dirname(__FILE__), 'boot')
```

About me

- ▶ Stefan Daschek (aka noniq)
- ▶ studied Computer Sciences (@ TU W
- ▶ teaching Mobile Robotics (@ FH Wr. N
- ▶ owning one half of a two-person-company

Remember svn / subversion?

```
--- config/environment.rb (revision 195)
+++ config/environment.rb (revision 196)
@@ -4,6 +4,9 @@
 # you don't control web/app server and can't set it the proper way
 # ENV['RAILS_ENV'] ||= 'production'
+# Specifies gem version of Rails to use when vendor/rails is not present
+RAILS_GEM_VERSION = '1.1.4'
# Bootstrap the Rails environment, frameworks, and default configuration
require File.join(File.dirname(__FILE__), 'boot')
```


About me

- ▶ Stefan Daschek (aka noniq)
- ▶ studied Computer Sciences (@ TU Wien)
- ▶ teaching Mobile Robotics (@ FH Wr. Neustadt)
- ▶ owning one half of a two-person-company
 - ❖ we build strange things
 - ❖ we also make Rails apps
 - ➔ we started our first Rails project in January 2006

Back on track:

**Using live request data for testing
while upgrading a Rails app**

Starting point:

A legacy Rails app

Legacy Rails app:

Legacy Rails app:

- ▶ Initial commit in 2012

Legacy Rails app:

- ▶ Initial commit in 2012
- ▶ ~50 models and controllers each (~18k LoC)

Legacy Rails app:

- ▶ Initial commit in 2012
- ▶ ~50 models and controllers each (~18k LoC)
- ▶ Rails 3.2 / Ruby 2.3

Legacy Rails app:

- ▶ Initial commit in 2012
- ▶ ~50 models and controllers each (~18k LoC)
- ▶ Rails 3.2 / Ruby 2.3
- ▶ Few tests

Legacy Rails app:

- ▶ Initial commit in 2012
- ▶ ~50 models and controllers each (~18k LoC)
- ▶ Rails 3.2 / Ruby 2.3
- ▶ Few tests
- ▶ Complex database logic

Legacy Rails app:

- ▶ Initial commit in 2012
- ▶ ~50 models and controllers each (~18k LoC)
- ▶ Rails 3.2 / Ruby 2.3
- ▶ Few tests
- ▶ Complex database logic

```
class User < ActiveRecord::Base
  attr_accessible :password, :password_confirmation, :email, :first_name, :last_name, :phone, :address_line_1, :address_line_2, :city, :state, :zip, :country

  validates :password, :password_confirmation, :email, :first_name, :last_name, :phone, :address_line_1, :address_line_2, :city, :state, :zip, :country

  def password_required?
    !password.blank?
  end

  def password_confirmation_required?
    !password_confirmation.blank?
  end

  def email_required?
    !email.blank?
  end

  def first_name_required?
    !first_name.blank?
  end

  def last_name_required?
    !last_name.blank?
  end

  def phone_required?
    !phone.blank?
  end

  def address_line_1_required?
    !address_line_1.blank?
  end

  def address_line_2_required?
    !address_line_2.blank?
  end

  def city_required?
    !city.blank?
  end

  def state_required?
    !state.blank?
  end

  def zip_required?
    !zip.blank?
  end

  def country_required?
    !country.blank?
  end

  def password_length?
    password.length.between?(8, 16)
  end

  def password_confirmation_length?
    password_confirmation.length.between?(8, 16)
  end

  def email_length?
    email.length.between?(6, 32)
  end

  def first_name_length?
    first_name.length.between?(2, 32)
  end

  def last_name_length?
    last_name.length.between?(2, 32)
  end

  def phone_length?
    phone.length.between?(7, 15)
  end

  def address_line_1_length?
    address_line_1.length.between?(5, 255)
  end

  def address_line_2_length?
    address_line_2.length.between?(5, 255)
  end

  def city_length?
    city.length.between?(3, 64)
  end

  def state_length?
    state.length.between?(2, 3)
  end

  def zip_length?
    zip.length.between?(5, 10)
  end

  def country_length?
    country.length.between?(2, 3)
  end

  def password_complexity?
    password =~ /\d/ && password =~ /[a-z]/ && password =~ /[A-Z]/ && password =~ /[\W_]/
  end

  def password_confirmation_complexity?
    password_confirmation =~ /\d/ && password_confirmation =~ /[a-z]/ && password_confirmation =~ /[A-Z]/ && password_confirmation =~ /[\W_]/
  end

  def email_format?
    email =~ /\A([\w+\-.]*(?@\w+)?\.\w+)\z/
  end

  def first_name_format?
    first_name =~ /\A[a-zA-Z ]*\z
  end

  def last_name_format?
    last_name =~ /\A[a-zA-Z ]*\z
  end

  def phone_format?
    phone =~ /\A(\d{3} )?\d{3}-\d{4}\z
  end

  def address_line_1_format?
    address_line_1 =~ /\A[a-zA-Z0-9 ]*\z
  end

  def address_line_2_format?
    address_line_2 =~ /\A[a-zA-Z0-9 ]*\z
  end

  def city_format?
    city =~ /\A[a-zA-Z ]*\z
  end

  def state_format?
    state =~ /\A[a-zA-Z ]*\z
  end

  def zip_format?
    zip =~ /\A\d{5}(-\d{4})?\z
  end

  def country_format?
    country =~ /\A[a-zA-Z ]*\z
  end

  def password_complexity_message?
    "Password must be 8-16 characters long, contain at least one digit, one lowercase letter, one uppercase letter, and one special character."
  end

  def password_confirmation_complexity_message?
    "Password confirmation must be 8-16 characters long, contain at least one digit, one lowercase letter, one uppercase letter, and one special character."
  end

  def email_format_message?
    "Email address is invalid."
  end

  def first_name_format_message?
    "First name is invalid."
  end

  def last_name_format_message?
    "Last name is invalid."
  end

  def phone_format_message?
    "Phone number is invalid."
  end

  def address_line_1_format_message?
    "Address line 1 is invalid."
  end

  def address_line_2_format_message?
    "Address line 2 is invalid."
  end

  def city_format_message?
    "City is invalid."
  end

  def state_format_message?
    "State is invalid."
  end

  def zip_format_message?
    "Zip code is invalid."
  end

  def country_format_message?
    "Country is invalid."
  end

  def password_complexity_message?
    "Password must be 8-16 characters long, contain at least one digit, one lowercase letter, one uppercase letter, and one special character."
  end

  def password_confirmation_complexity_message?
    "Password confirmation must be 8-16 characters long, contain at least one digit, one lowercase letter, one uppercase letter, and one special character."
  end

  def email_format_message?
    "Email address is invalid."
  end

  def first_name_format_message?
    "First name is invalid."
  end

  def last_name_format_message?
    "Last name is invalid."
  end

  def phone_format_message?
    "Phone number is invalid."
  end

  def address_line_1_format_message?
    "Address line 1 is invalid."
  end

  def address_line_2_format_message?
    "Address line 2 is invalid."
  end

  def city_format_message?
    "City is invalid."
  end

  def state_format_message?
    "State is invalid."
  end

  def zip_format_message?
    "Zip code is invalid."
  end

  def country_format_message?
    "Country is invalid."
  end

  def password_complexity_message?
    "Password must be 8-16 characters long, contain at least one digit, one lowercase letter, one uppercase letter, and one special character."
  end

  def password_confirmation_complexity_message?
    "Password confirmation must be 8-16 characters long, contain at least one digit, one lowercase letter, one uppercase letter, and one special character."
  end

  def email_format_message?
    "Email address is invalid."
  end

  def first_name_format_message?
    "First name is invalid."
  end

  def last_name_format_message?
    "Last name is invalid."
  end

  def phone_format_message?
    "Phone number is invalid."
  end

  def address_line_1_format_message?
    "Address line 1 is invalid."
  end

  def address_line_2_format_message?
    "Address line 2 is invalid."
  end

  def city_format_message?
    "City is invalid."
  end

  def state_format_message?
    "State is invalid."
  end

  def zip_format_message?
    "Zip code is invalid."
  end

  def country_format_message?
    "Country is invalid."
  end
end
```

Scopes, scopes, scopes ...

Legacy Rails app:

- ▶ Initial commit in 2012
- ▶ ~50 models and controllers each (~18k LoC)
- ▶ Rails 3.2 / Ruby 2.3
- ▶ Few tests
- ▶ Complex database logic



Scopes, scopes, scopes...

Legacy Rails app:

- ▶ Initial commit in 2012
- ▶ ~50 models and controllers each (~18k LoC)
- ▶ Rails 3.2 / Ruby 2.3
- ▶ Few tests
- ▶ Complex database logic

Challenge accepted!



```
class ApplicationController < ActionController::Base
  # Prevent CSRF attacks by raising an exception if the authenticating user
  # does not have the appropriate CSRF token.
  protect_from_forgery with: :exception

  # See ActionController::RequestForgeryProtection for details
  # http://api.rubyonrails.org/classes/ActionController::RequestForgeryProtection.html

  # Reload the browser's cache by setting the Cache-Control header to 'no-cache'
  def self.cache_sweeper
    respond_to do |format|
      format.html { response.headers[:"Cache-Control"] = "no-cache" }
    end
  end
end

class User < ActiveRecord::Base
  attr_accessible :username, :password, :email

  # Example of a complex database query using raw SQL
  def find_by_username_or_email(username, email)
    sql = "SELECT * FROM users WHERE username = ? OR email = ?"
    self.execute(sql, username, email)
  end

  # Example of a complex database query using ActiveRecord's query API
  def find_by_username_or_email(username, email)
    self.where("username = ? OR email = ?", username, email)
  end
end
```

Scopes, scopes, scopes...

About 160 commits later ...

About 160 commmits later ...

- ▶ Rails 5.2 / Ruby 2.6 🎉

About 160 commmits later ...

- ▶ Rails 5.2 / Ruby 2.6 🎉
- ▶ More tests (and they were passing)

About 160 commits later ...

- ▶ Rails 5.2 / Ruby 2.6 🎉
- ▶ More tests (and they were passing)
- ▶ But: Did we miss anything?



About 160 commits later ...

- ▶ Rails 5.2 / Ruby 2.6 🎉
- ▶ More tests (and they were passing)
- ▶ But: Did we miss anything?



What if we could run the upgraded version in parallel to the production app, feeding it live data?

Production App

Upgraded App



Production App



Upgraded App



Production App



Upgraded App



Production App

Upgraded App



Production App

Upgraded App



But how?

Step 1: Apply some nginx magic

```
location / {  
    mirror /_mirror;  
    # ...  
}  
  
location = /_mirror {  
    internal;  
    proxy_pass http://upgraded_app$request_uri;  
}
```


Step 1: Apply some nginx magic

```
location / {  
    mirror /_mirror;  
    # ...  
}  
  
location = /_mirror {  
    internal;  
    proxy_pass http://upgraded_app$request_uri;  
}
```

The `ngx_http_mirror_module` (1.13.4) implements mirroring of an original request by creating background mirror subrequests. Responses to mirror subrequests are ignored.

Step 2: Disable all the modern goodies

```
# For example (non-exhaustive):  
config.action_controller.per_form_csrf_tokens = false  
config.action_controller.forgery_protection_origin_check = false  
config.action_dispatch.cookies_serializer = :marshal  
config.action_dispatch.use_authenticated_cookie_encryption = false
```

Step 3: Make sure cookies are compatible

- ▶ Make sure `config.secret_token` is identical
- ▶ Do not use `config.secret_key_base` in the upgraded app (yet)

Step 3: Make sure cookies are compatible

- ▶ Make sure `config.secret_token` is identical
- ▶ Do not use `config.secret_key_base` in the upgraded app (yet)

Please note that you should wait to set `secret_key_base` until you have 100% of your userbase on Rails 4.x and are reasonably sure you will not need to rollback to Rails 3.x. This is because cookies signed based on the new `secret_key_base` in Rails 4.x are not backwards compatible with Rails 3.x.

But even then . . .

. . . the data will diverge over time

Exhibit 1: Speed differences

Production App

Upgraded App

Exhibit 1: Speed differences

Production App

100 docs

Upgraded App

100 docs

Exhibit 1: Speed differences

Production App

Upgraded App

Request 1

→ creates a lot of new documents in the database

100 docs



200 docs

100 docs



200 docs

Exhibit 1: Speed differences

Production App

Upgraded App

100 docs

100 docs

Request 1

→ creates a lot of new documents in the database

+100

+100

200 docs

200 docs

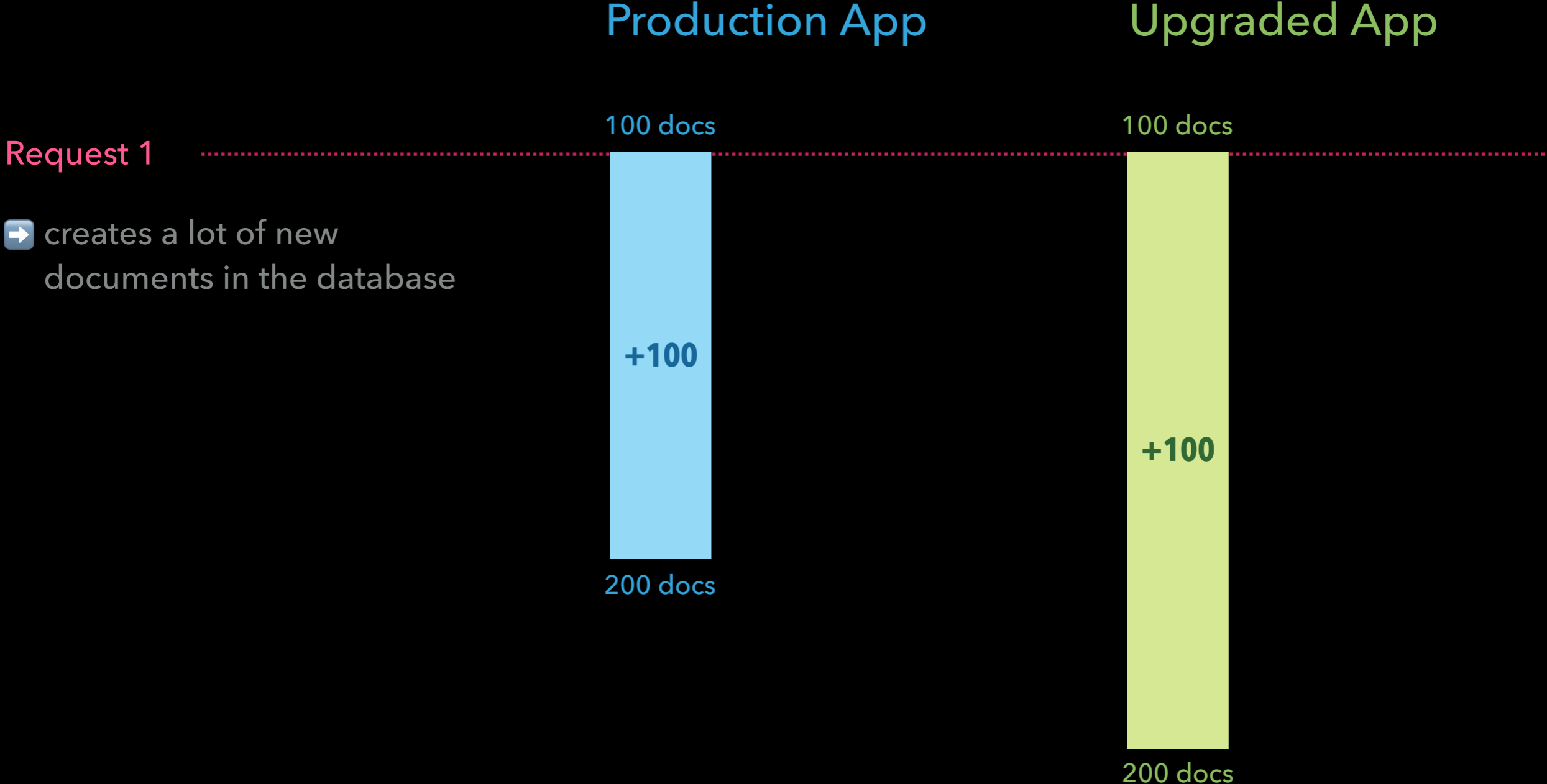


Exhibit 1: Speed differences

Production App

Upgraded App

Request 1

→ creates a lot of new documents in the database

100 docs

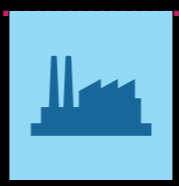
100 docs



Request 2

→ processes all existing documents

200 docs



200 docs

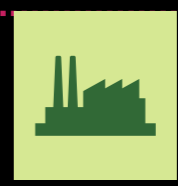


Exhibit 1: Speed differences

Production App

Upgraded App

Request 1

→ creates a lot of new documents in the database

100 docs

100 docs

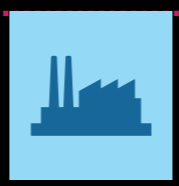


Request 2

→ processes all existing documents

200 docs

??? docs



200 docs

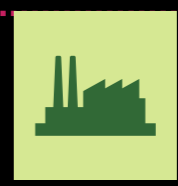


Exhibit 2: Password Reset

Production App

Upgraded App

Exhibit 2: Password Reset

Production App

Upgraded App

GET /password/new

→ renders password reset form

→ renders password reset form

Exhibit 2: Password Reset

Production App

Upgraded App

GET /password/new

→ renders password reset form

→ renders password reset form

POST /password

→ creates random password reset token ("foo")

→ creates random password reset token ("bar")

→ sends password reset link (to user)

→ sends password reset link (to sandbox)

Exhibit 2: Password Reset

Production App

Upgraded App

GET /password/new

→ renders password reset form

→ renders password reset form

POST /password

→ creates random password reset token ("foo")

→ creates random password reset token ("bar")

→ sends password reset link (to user)

→ sends password reset link (to sandbox)

GET /password/edit?token=foo

→ valid token, continue to change password

→ invalid token, password remains unchanged

→ new password "456"

→ password still "123"

Exhibit 2: Password Reset

Production App

Upgraded App

GET /password/new

→ renders password reset form

→ renders password reset form

POST /password

→ creates random password reset token ("foo")

→ creates random password reset token ("bar")

→ sends password reset link (to user)

→ sends password reset link (to sandbox)

GET /password/edit?token=foo

→ valid token, continue to change password

→ invalid token, password remains unchanged

→ new password "456"

→ password still "123"

Surprisingly though . . .

it doesn't matter!

Sign in with password

Production App

Upgraded App

Sign in with password

Production App

Upgraded App

GET /sign_in

→ renders sign in form

→ renders sign in form

Sign in with password

Production App

Upgraded App

GET /sign_in

→ renders sign in form

→ renders sign in form

POST /sign_in?password=456

→ correct password

→ incorrect password

→ creates session cookie

→ responds with 401

Sign in with password

Production App

Upgraded App

GET /sign_in

→ renders sign in form

→ renders sign in form

POST /sign_in?password=456

→ correct password

→ incorrect password

→ creates session cookie

→ responds with 401

GET /my_profile

→ receives valid
session cookie

→ receives valid
session cookie

→ allows access

→ allows access

Sign in with password

Production App

Upgraded App

GET /sign_in

→ renders sign in form

→ renders sign in form

POST /sign_in?password=456

→ correct password

→ incorrect password

→ creates session cookie

→ responds with 401

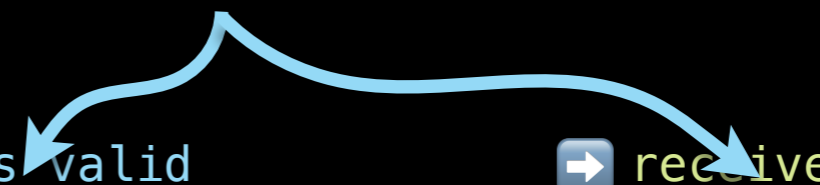
GET /my_profile

→ receives valid session cookie

→ receives valid session cookie

→ allows access

→ allows access



Some workarounds needed (of course ...)

```
class User < ApplicationRecord
  # ...

  # Fix backward compatibility for Devise (Authenticatable):
  # The older version used in the production app serializes 3 arguments into
  # the session, but the newer version in the upgraded app expects only 2
  # arguments.
  #
  # For now our code needs to support both: old-style sessions for requests
  # mirrored from the production app, and new-style sessions for requests
  # when browsing the upgraded app directly.
  def self.serialize_from_session(*args)
    key = args.size == 2 ? args[0] : args[1]
    to_adapter.get(key)
  end
end
```


Was it worth the effort?

What did we get out of it?

What did we get out of it?

- ▶ noticed (and fixed) a handful of upgrade-related bugs that our tests missed

What did we get out of it?

- ▶ noticed (and fixed) a handful of upgrade-related bugs that our tests missed
- ▶ found no (relevant) differences in data even after several weeks of running the upgraded app in parallel

Was it worth the effort?

Was it worth the effort?

Absolutely! 😎

Thanks!

Stefan Daschek / @noniq@chaos.social