Développement d'une application web



Ecriture des tests de non-régression

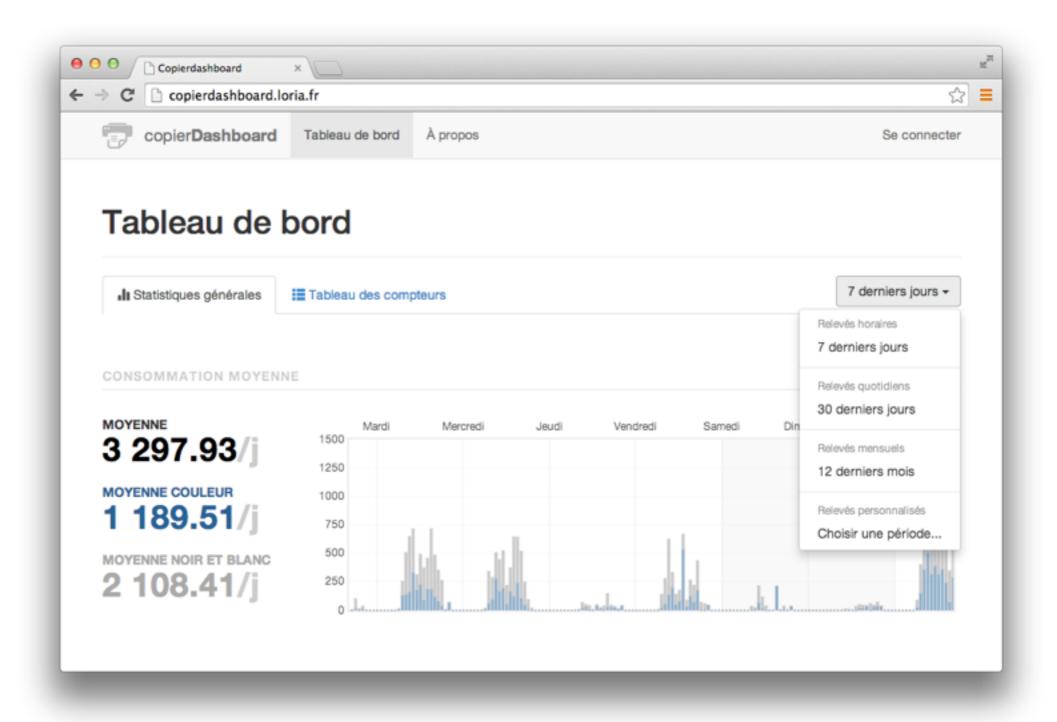
Finalité

- Garantir le fonctionnement intégral d'une application web en production
- Eviter les régressions lors de l'évolution de l'application

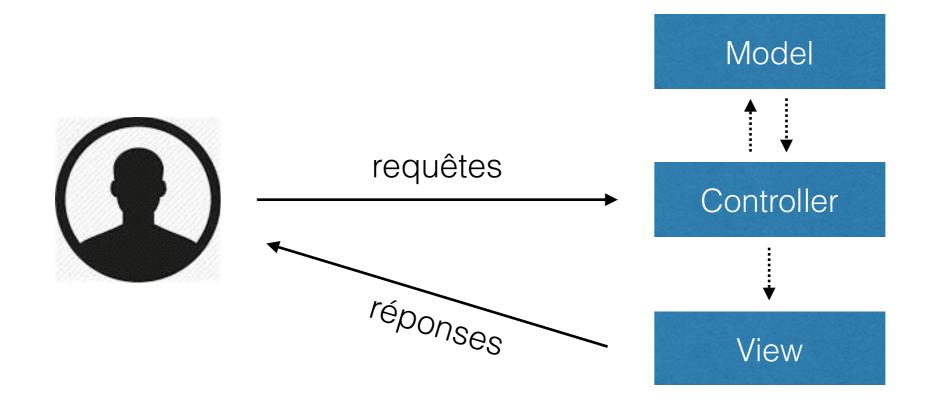


Anatomie d'une web app

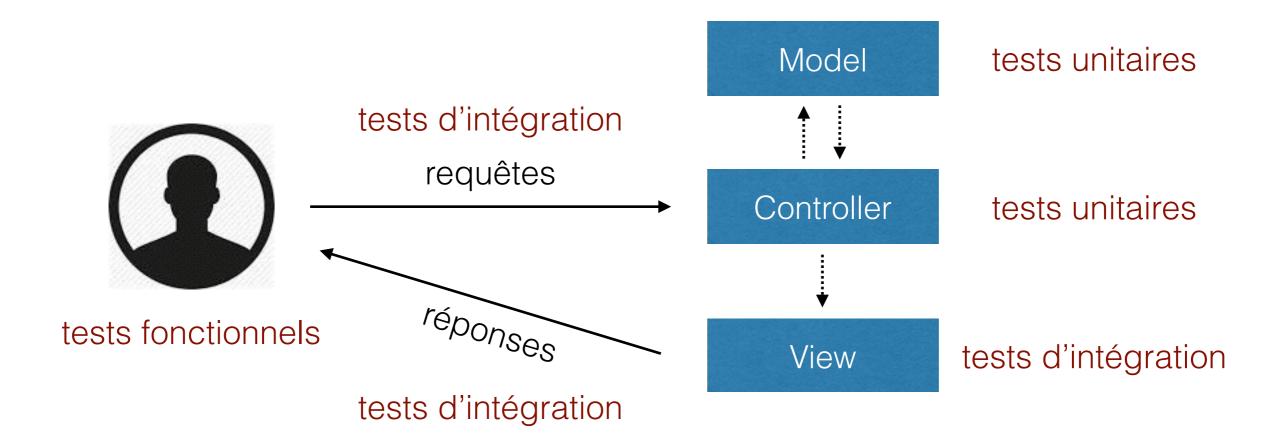
Fonctionnement d'une web app



Architecture d'une web app



Architecture d'une web app



La quête de la couverture totale

On peut absolument TOUT tester

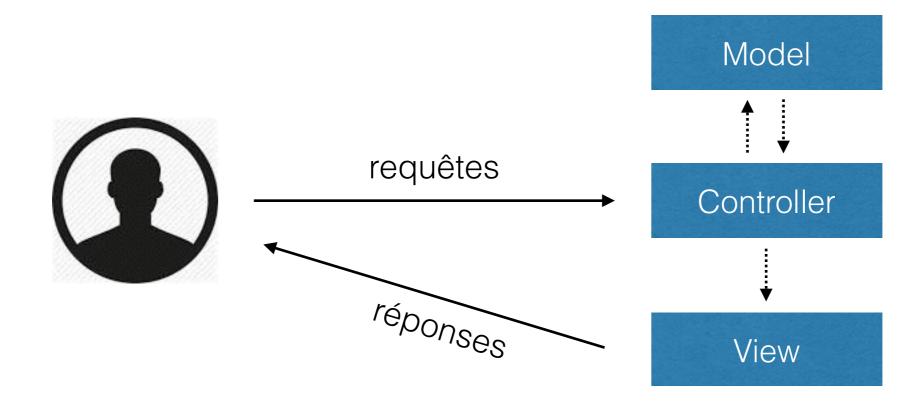
(et cela cache un piège sournois)

Différents types de test

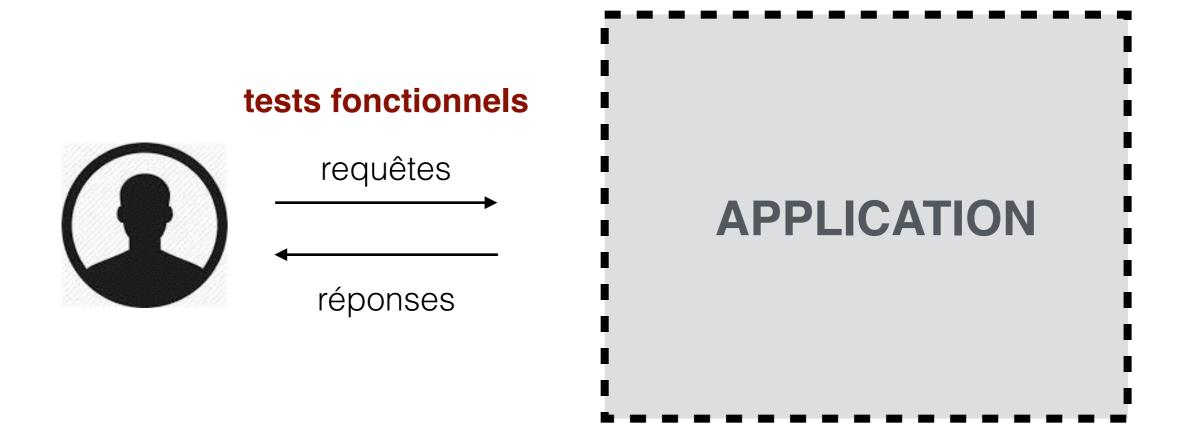
	Type	Cible	Domaine
-	Tests fonctionnels	Utilisateurs	Fonctionnalités
	Test d'intégration	Développeurs	Inter-connexion des composants
	Tests unitaires	Développeurs	Architecture de l'application
		• • •	

Test des spécifications fonctionnelles

Vue complète



Vue en boîte noire



Test fonctionnels

- · Ce que l'utilisateur fait et voit
- Test l'ensemble des couches de l'application

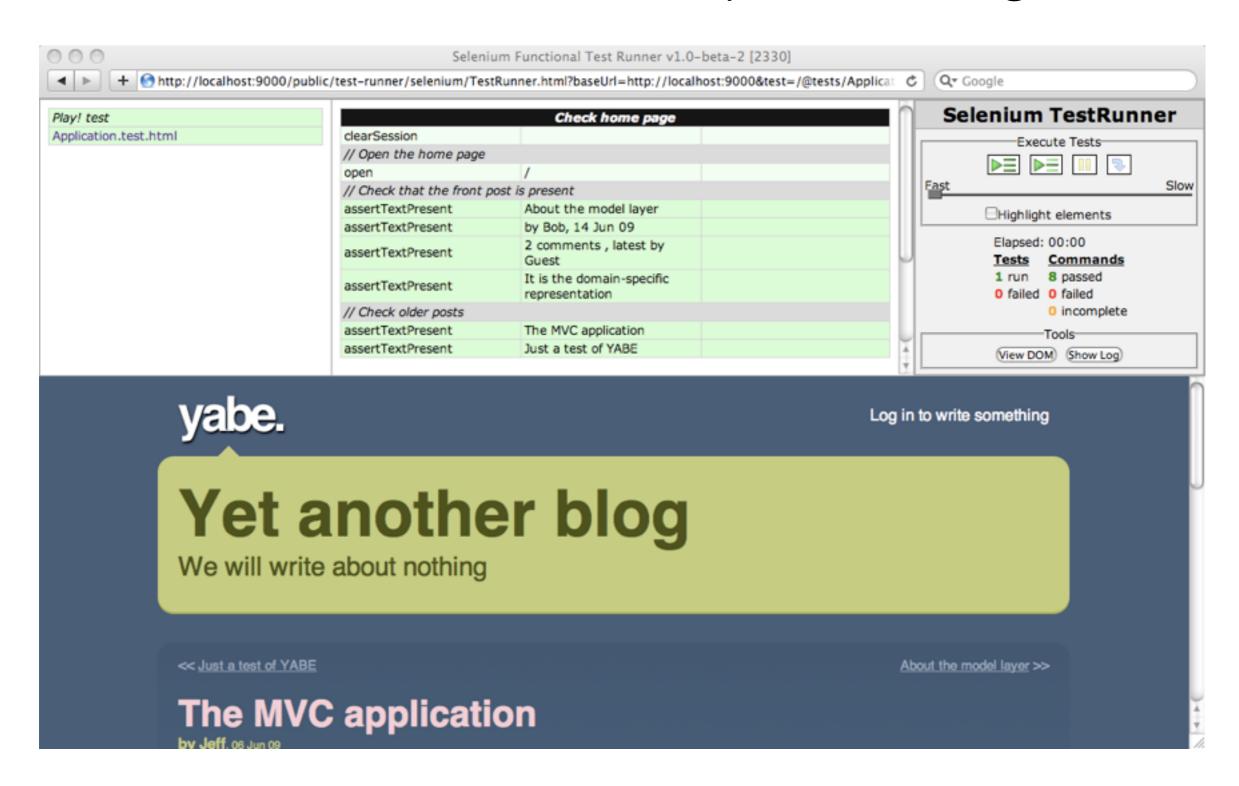
Exemple de scénario

Création d'un enregistrement par utilisateur lambda

- 1. Un utilisateur lambda s'authentifie sur l'application
- 2. Il accède à la page du formulaire
- 3. Il rempli 2 champs et valide le formulaire
 - Il doit voir un message de succès
 - Il doit voir son enregistrement

Simulation des scénarios par le **navigateur**

Simulation des scénarios par le navigateur



Simulation des scénarios par le navigateur

- Lourd à utiliser surtout à plusieurs et en production
- Attention au mode « enregistrement live »
 privilégier la sélection des composant
 par attributs HTML (classe, ID ou autre attribut)

Moteurs de rendu « headless » Webkit, PhantomJS, etc.

- Simulations complètes sans navigateur « lourd »
- En ligne de commande
- Multi-environnement

Ecriture des scénarios dans un langage dédié (DSL)

Objectifs

- Ecrire les tests dans un format compréhensible par les acteurs concernés
 - Inclure les tests dans le SCM
 - Déployer en qualification (pas en production -> préserver intégrité)

Ecriture des scénarios dans un DSL

(Cucumber, Gherkin, etc.)

```
language: fr
Fonctionnalité: Consulter ses facture passées
 Afin de permettre aux utilisateurs de voir l'évolution de leur communication
 En tant qu'utilisateur connecté
 Je veux être capable de consulter l'ensemble des montants de mes factures
 Contexte:
   Soit un utilisateur connecté en tant que "vco" avec comme mot de passe "monkey"
   Et que ces factures existent au nom de "vco" :
                         | Montant |
       Année | Mois
      | 2010 | Janvier | 40
     | 2009 | Décembre | 35
     | 2009 | Novembre | 35
 Scénario: Consulter une facture en particulier
   Soit je suis sur la page de consultation de mes factures
   Lorsque je clique sur le lien "Facture de Novembre"
   Alors je devrais voir "Facture du mois de Novembre 2009"
   Et je devrais voir "Montant de votre facture: 35 euros"
```

Ecriture des scénarios dans un DSL (Cucumber, Gherkin, etc.)

```
Etantdonné /^que je suis sur la page de connexion du wiki$/ do
@browser.goto('http://example.com/start?do=login')
end

Quand /^je m'identifie en tant que "([^"]*)" \
avec le mot de passe "([^"]*)"$/ do |id, password|
@browser.text_field(:name, 'u').set(id)
@browser.text_field(:name, 'p').set(password)
@browser.button(:value, 'Connexion').click
end

Alors /^je devrais voir "([^"]*)"$/ do |text|
@browser.text.include?(text).should == true
end
```

Ecriture des scénarios dans un DSL

- On écrit :
 - les scénarios (langage non technique)
 - les tests associés (code)
 - les jeux de données (code)
- L'outil interprète et exécute l'ensemble des tests

Ecriture des scénarios dans un DSL

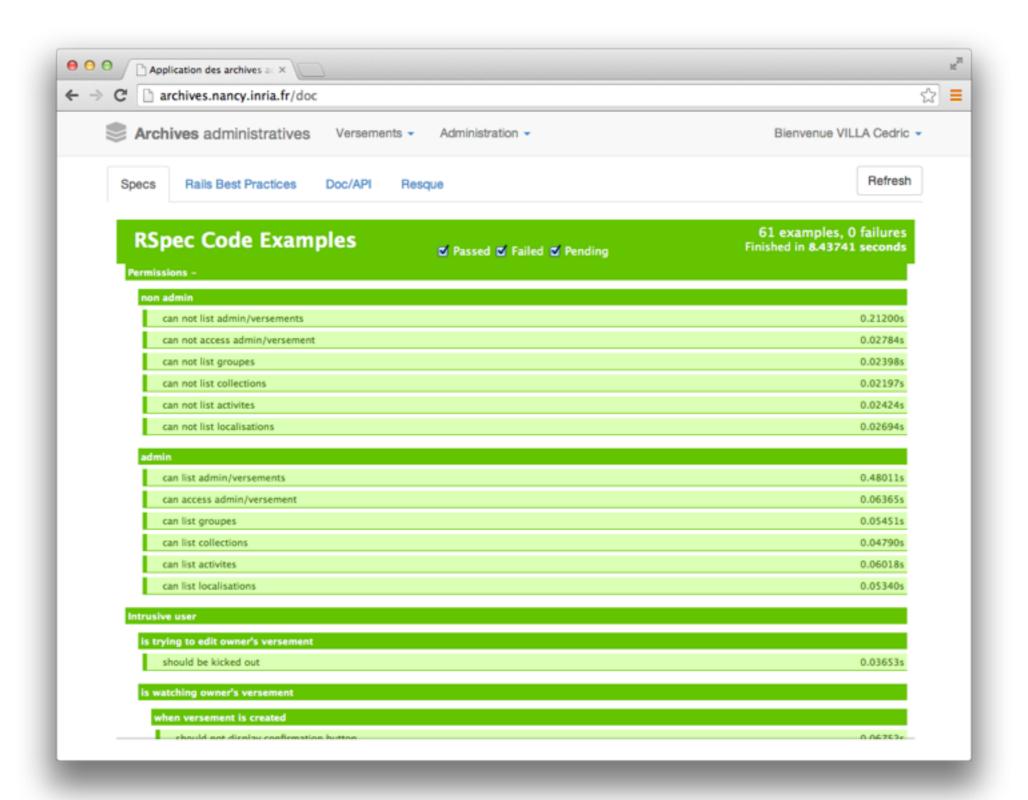
(RSpec / JUnit / Jasmine / unittest / NUnit)

```
feature 'Versements creation' do
  scenario 'when lambda user creates a versement' do
    let(:groupe) { create(:groupe, :nom => 'SRH') }
let(:activite) { create(:activite, :nom => 'Communication') }
    before(:each) do
      CASClient::Frameworks::Rails::Filter.fake('villa')
      visit new_versement_path
      select 'SRH', from: 'versement[id_groupe]'
      select 'Communication', from: 'versement[id_activite]'
      click_button 'Créer un versement'
    end
    it 'show successful flash notice' do
      expect(page).to have_content 'Le versement a bien été créé.'
    end
    it 'is displayed with YEAR-XXX format' do
      expect(page).to have_content "#{DateTime.now.year}-XXX"
    end
```

Execution des tests en ligne de commande

```
\Theta \Theta \Theta
                            archives-admin — bash — 80×24
archives-admin[master*] $ bundle exec rspec --format documentation
BoitesHelper
  collection
    should return a valid title if set
    should return 'aucune' if not set
  destruction label
    should return destruction label for destroyed versement
  admin_localisation_label
    should return 'Bureau(C123) for boite in bureau
    should return 'Bureau(non défini) for boite in bureau not located
    should return 'à définir' if localisation & cote is not set
    should return 'C008(1-2-3) for boite in salle archive
  numero_boite on new versement
    should return a (YEAR)-XXX-001 on creation
  numero_boite on existing versement
    should return a (YEAR)-001-001 on confirm
Confirmed versement
  should show validate button
  should show reject button
 when validating
    should be validated
    when email is sent
```

Bonus : générer le cahier de recettes



Intégration dans IDE

```
\Theta \Theta \Theta
                                                            csv export.rb - archives-admin
                                                                                                                                      UNREGISTERED w
OPEN FILES
                                                csv_export.rb
FOLDERS
                                            1 # encoding: utf-8

    ▼ archives-admin

                                               # CSV data export
  ▶ .bundle
                                                # - export all app data in CSV format
  ₩ app
                                               # Author:: Cedric Villa (https://github.com/driket)
   ▶ assets
                                               # Copyright:: Check licence file
   ▶ controllers
                                               # License:: Check license file
   ▶ helpers
                                                class CsvExport
    ▶ mailers
                                                  # Setup default values

    models

                                                 # Params::

    ▼ services

                                          11
                                                  # +semicolon_format:: use semicolon instead of comma
        app_migration.rb
                                                  # +line_break_filter:: remove line breaks
                                           12
        app_version.rb
                                           13
                                                  def initialize(params = {})
                                                    # setup default values
                                          14
        settings.rb
                                           15
                                                    params[:semicolon_format] ||= true
params[:line_break_filter] ||= true
       .gitkeep
                                          16
       activite.rb
                                                    # transform params into instance variables
                                          17
       boite.rb
                                           18
                                                    params.each { |name, value| instance_variable_set("@#{name}", value) }
                                           19
       collection.rb
                                           20
       groupe.rb
                                           21
                                                  # Process CSV
       ldap_user.rb
                                          22
                                                  # - request all boites with params
       localisation.rb
                                          23
                                                  # - generate and return csv file
       versement.rb
                                           24
                                                  def process
   ▶ views
                                           25
                                                    # request all boites + versements
  ▶ config
                                           26
                                                    boites = boites_with_versements
  b db
                                           27
                                                    # csv format
                                           28
                                                             = custom_csv_export(boites)
  29
                                                    # apply active filters
     changelog.rdoc
                                           30
                                                    linebreak_filter(csv) if @line_break_filter
     data_migration.rdoc
                                           31
                                                    comma_filter(csv) if @semicolon_format
     export_archives.sql
     README_FOR_APP
13:38:31 - INFO - Running all specs
   Finished in 10.04 seconds
61 examples, 0 failures
On master in archives-admin (UNLICENSED), 4 errors, Line 1, Column 1
                                                                                                                                            Ruby
```

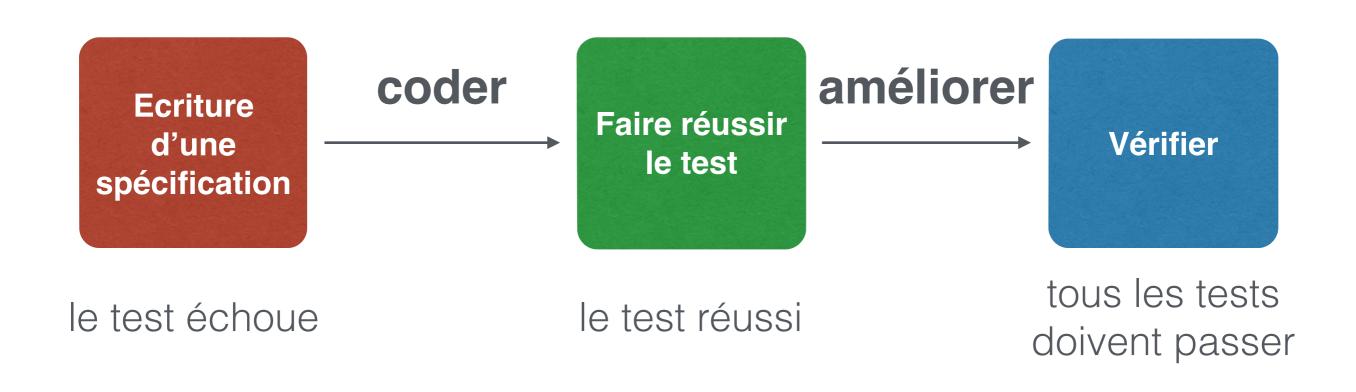
Échouez tôt et échouez souvent.

Échouez tôt, échouez souvent

- Une défaillance est plus facilement réparable juste après son apparition
- Il vaut mieux être alerté par un indicateur, que par un utilisateur furieux.

Faites-en un rituel

Implémentation en 3 étapes



Bilan

- Beaucoup d'aprioris
- Finalement plutôt passionnant
 - Amélioration significative de la qualité globale (code + fonctionnement)
 - Plus aisé de partager (confiance en son code)

Références

- http://betterspecs.org/fr/ (bonnes pratiques tests RSPEC)
- http://phantomjs.org/ (headless web rendering)
- http://shcatula.wordpress.com/2013/02/21/java-rspecalternative/ (alternative à RSPEC pour Java)
- http://blog.octo.com/jai-limpression-decrire-mes-tests-en-double/ (gestion doublons dans les tests)