Développement d'une **application web**

Ecriture des **tests** de **non-régression**

cedric.villa@loria.fr

Finalité

- Garantir le **fonctionnement intégral** d'une application web en production

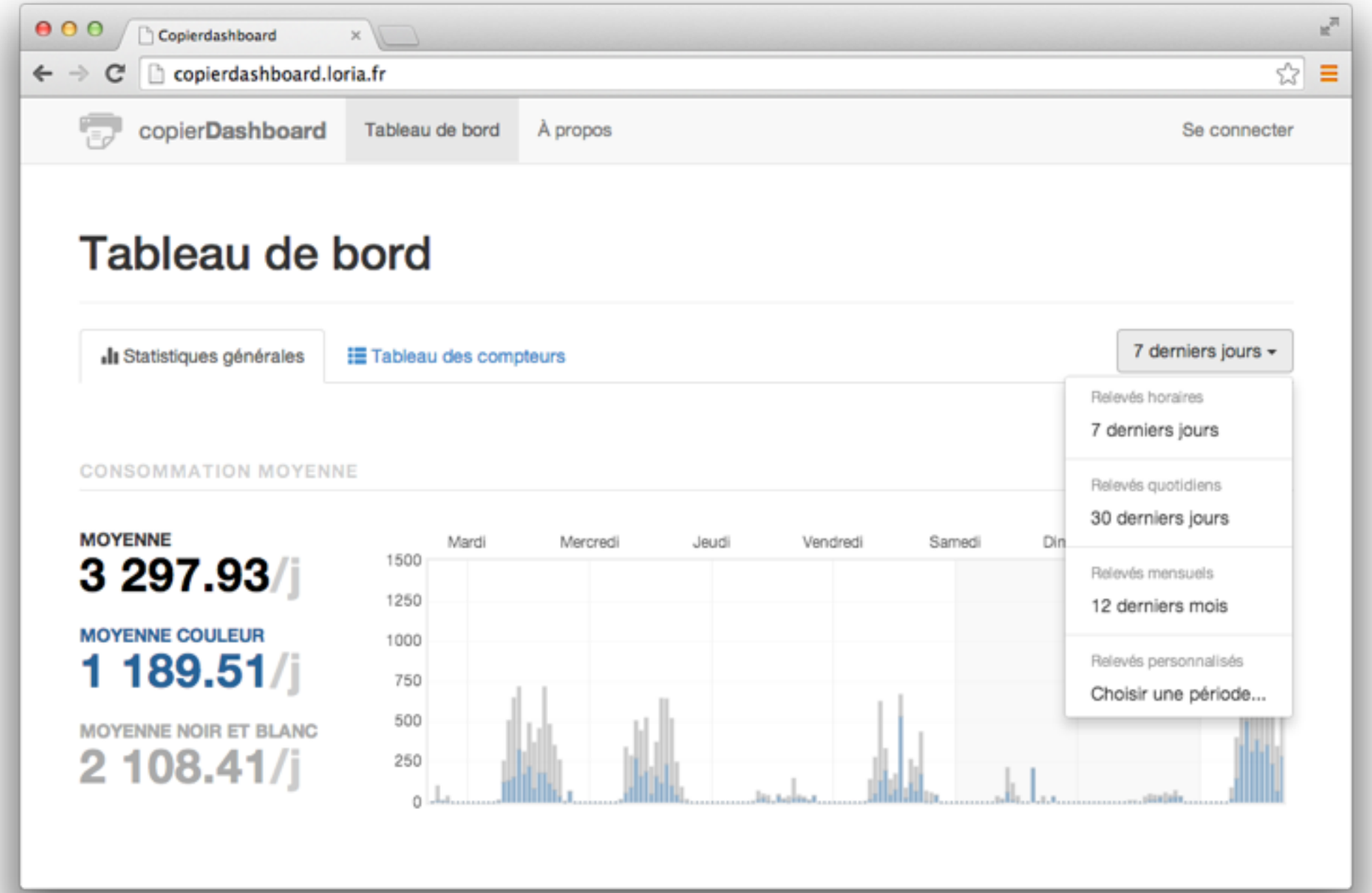
- Eviter les **régressions** lors de **l'évolution** de l'application

cedric.villa@loria.fr

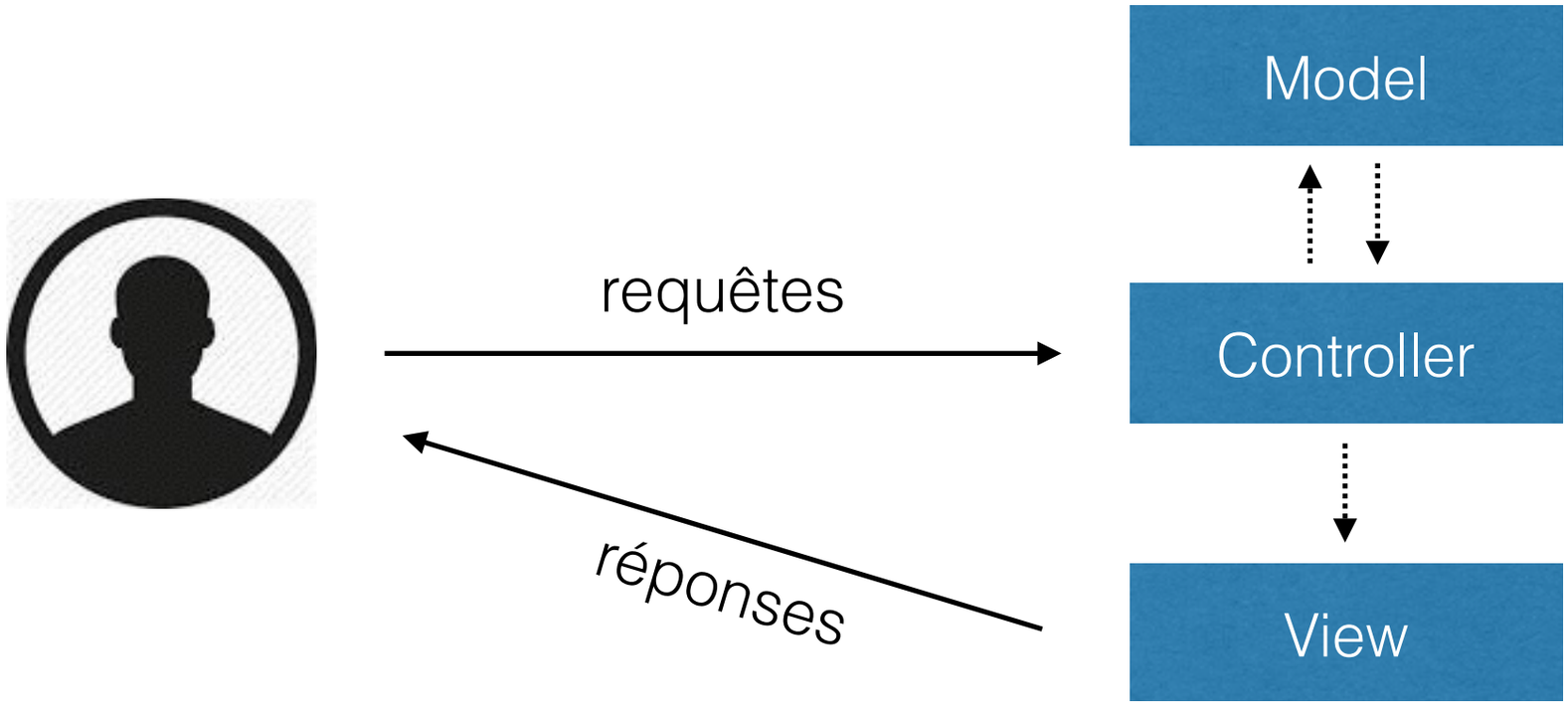
Anatomie d’une **web app**

cedric.villa@loria.fr

Fonctionnement d'une web app



Architecture d’une web app



cedric.villa@loria.fr

Architecture d'une web app

tests fonctionnels

tests d'intégration

requêtes

réponses

tests d'intégration

Model

tests unitaires

Controller

tests unitaires

View

tests d'intégration

cedric.villa@loria.fr

La quête de la couverture totale

On peut absolument TOUT tester

(et cela cache un piège sournois)

cedric.villa@loria.fr

Différents types de test

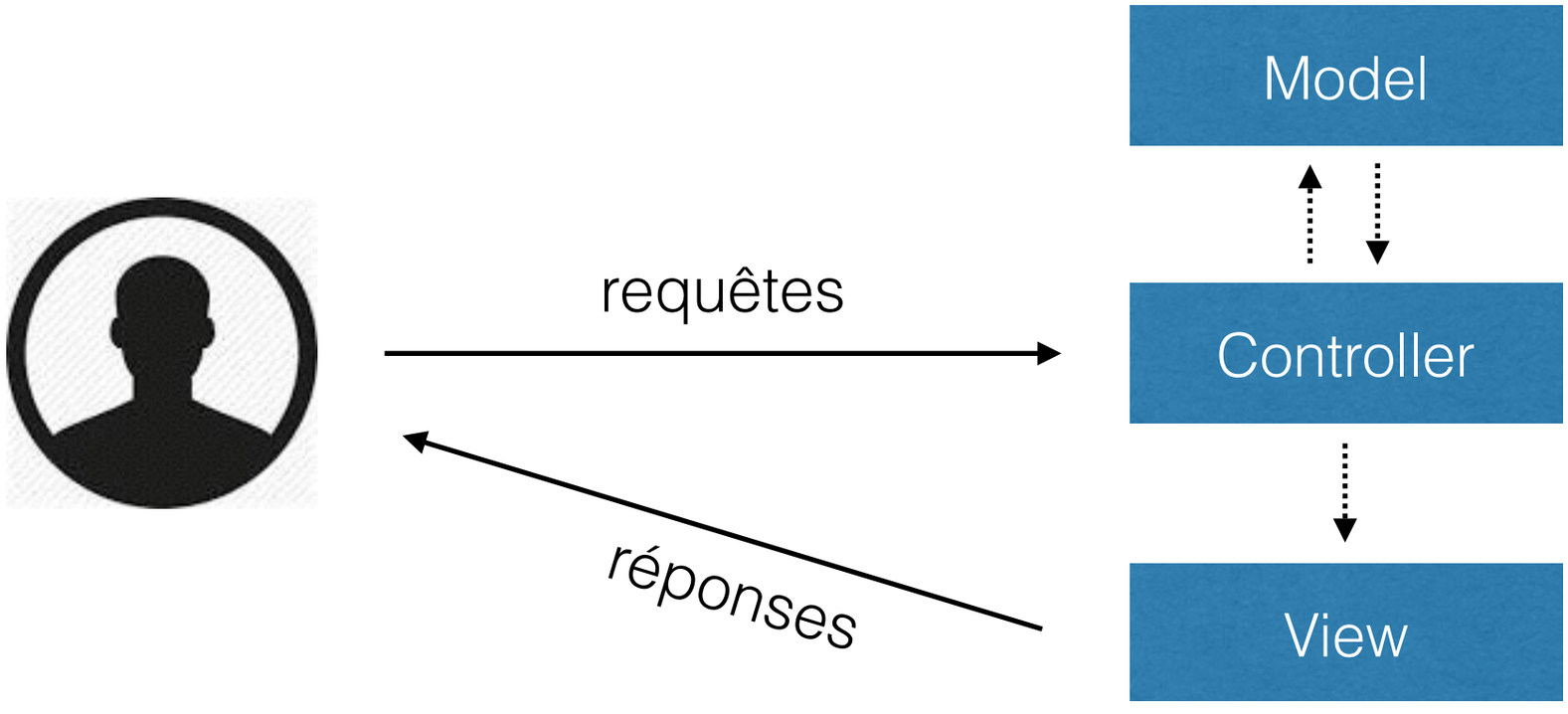
Type	Cible	Domaine
Tests fonctionnels	Utilisateurs	Fonctionnalités
Test d'intégration	Développeurs	Inter-connexion des composants
Tests unitaires	Développeurs	Architecture de l'application
...	...	...

cedric.villa@loria.fr

Test des **spécifications fonctionnelles**

cedric.villa@loria.fr

Vue complète



Vue en boîte noire

tests fonctionnels

requêtes

réponses

APPLICATION

cedric.villa@loria.fr

Test fonctionnels

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

Exemple de scénario

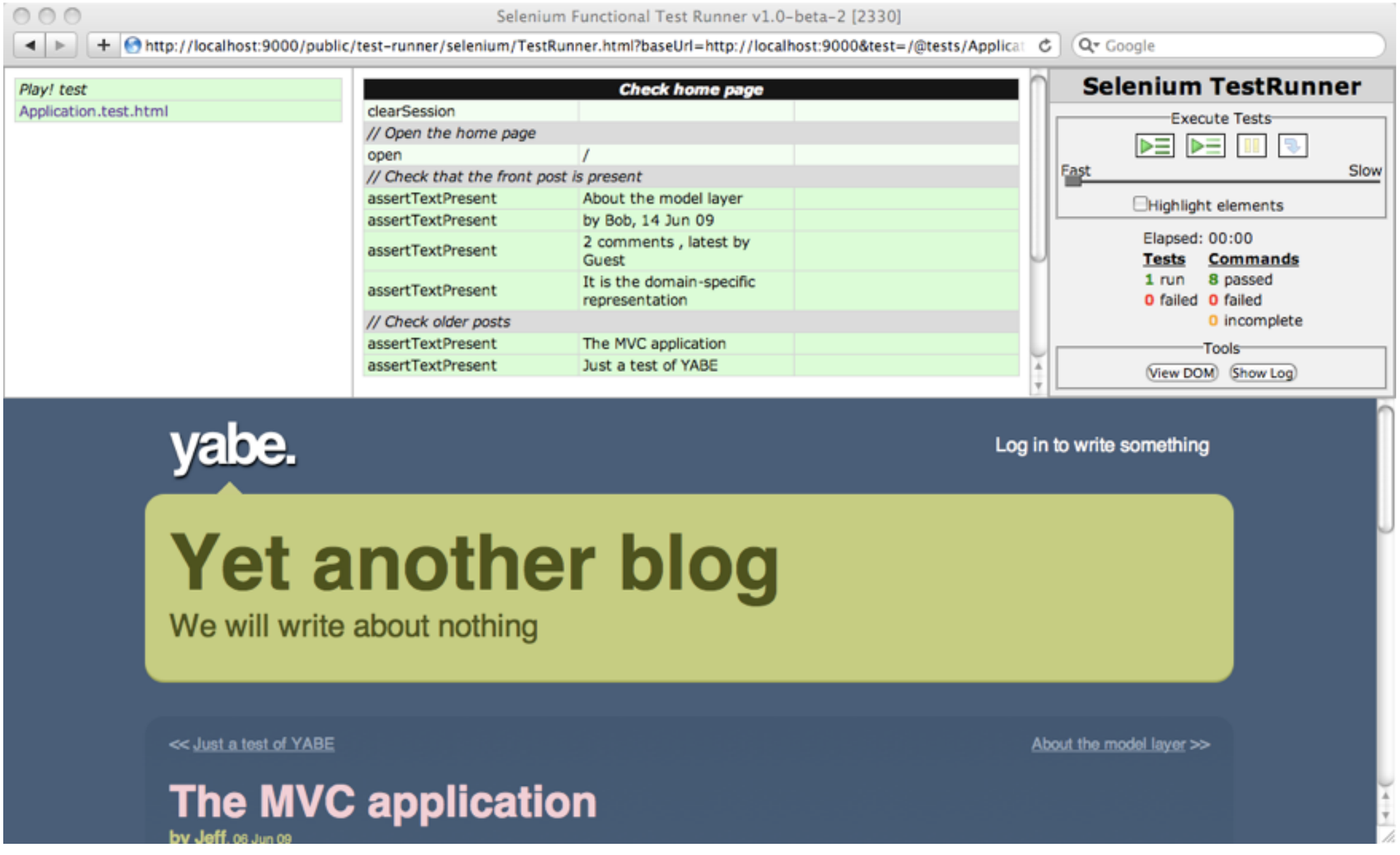
Création d'un enregistrement par utilisateur lambda

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**

cedric.villa@loria.fr

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)

cedric.villa@loria.fr

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" :
      | Année | Mois     | Montant |
      | 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
```

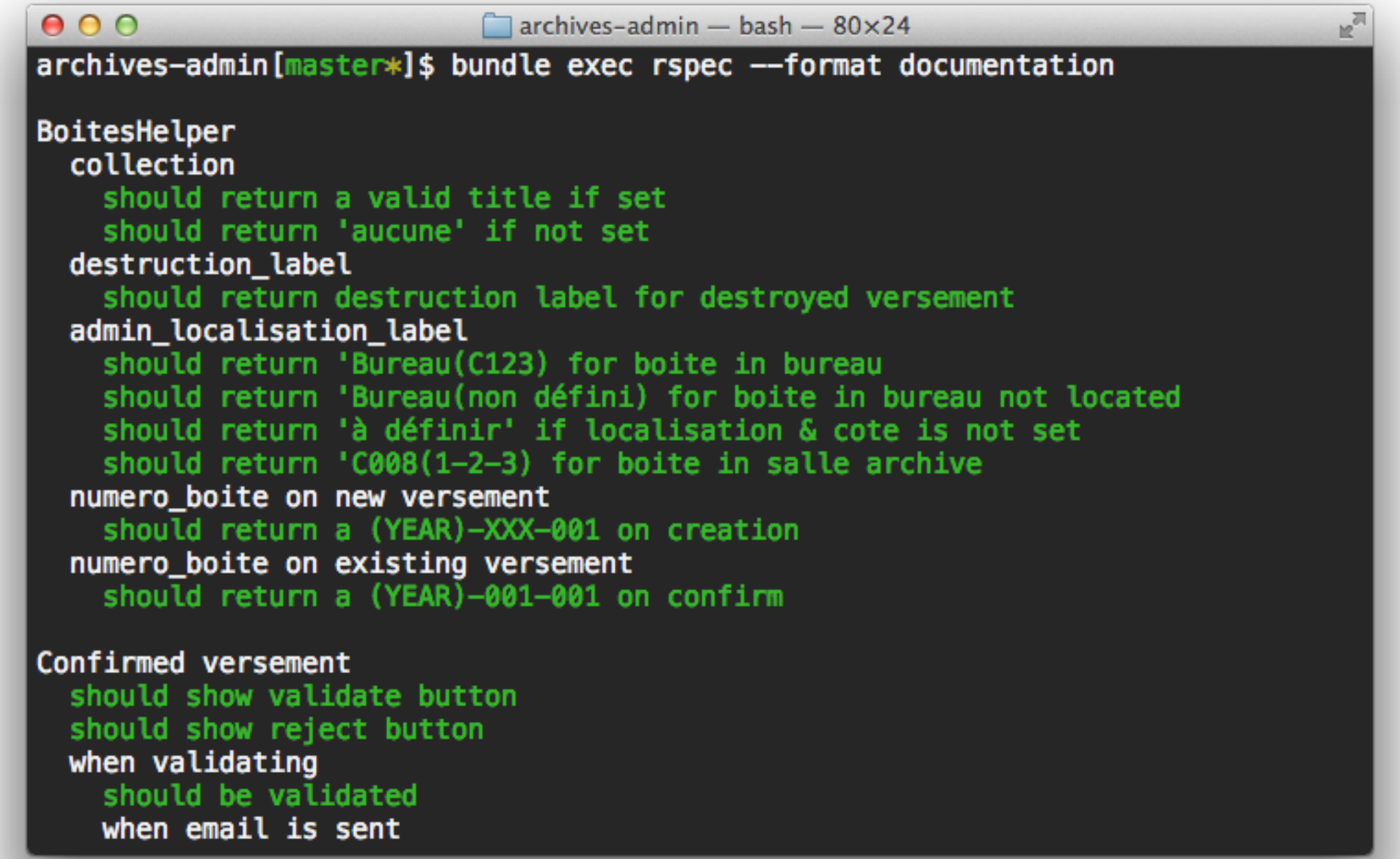
Ecriture des scénarios dans un DSL

- 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

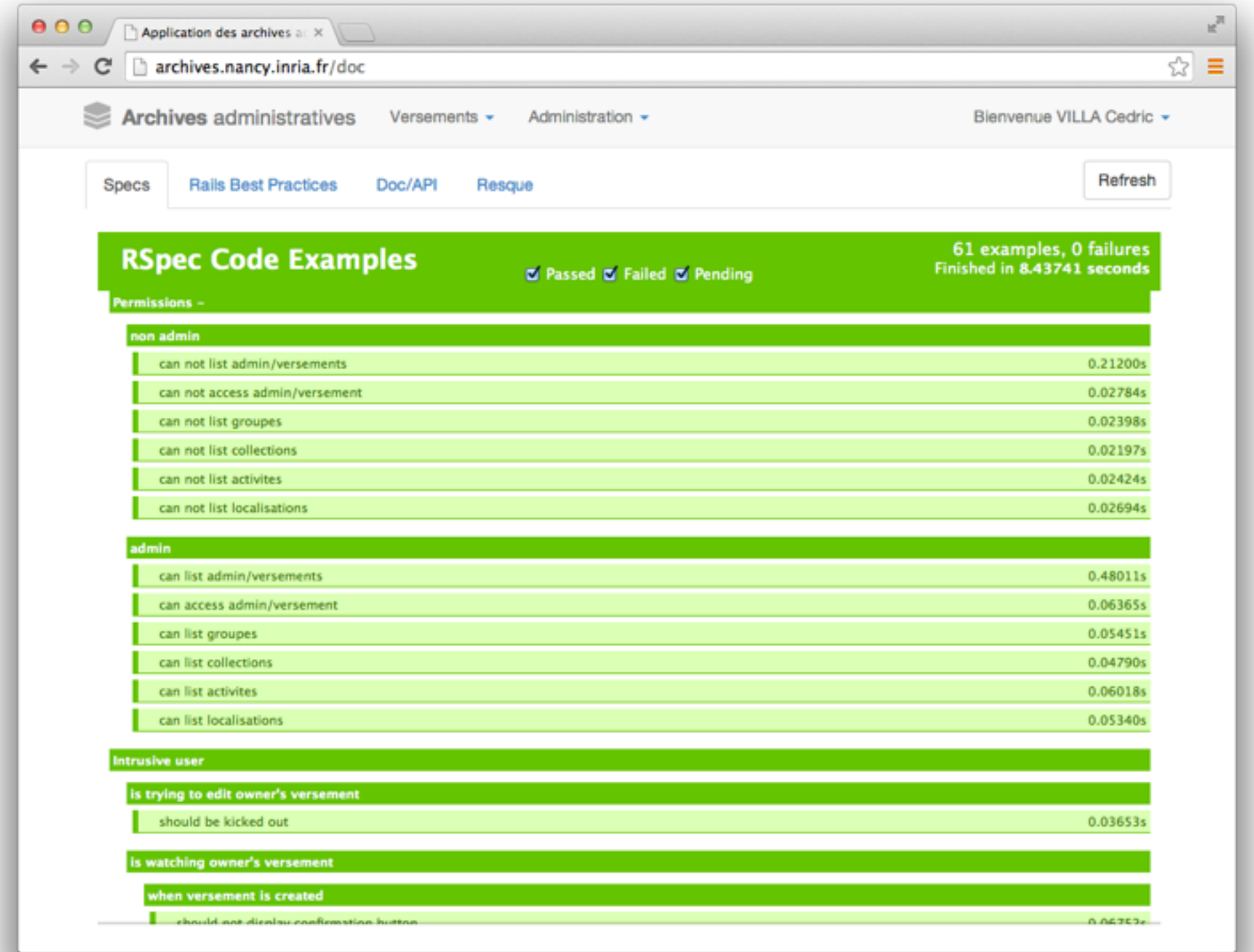
Ecriture des scénarios dans un DSL
(RSpec / JUnit / Jasmine / unittest / NUnit)

```
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
```

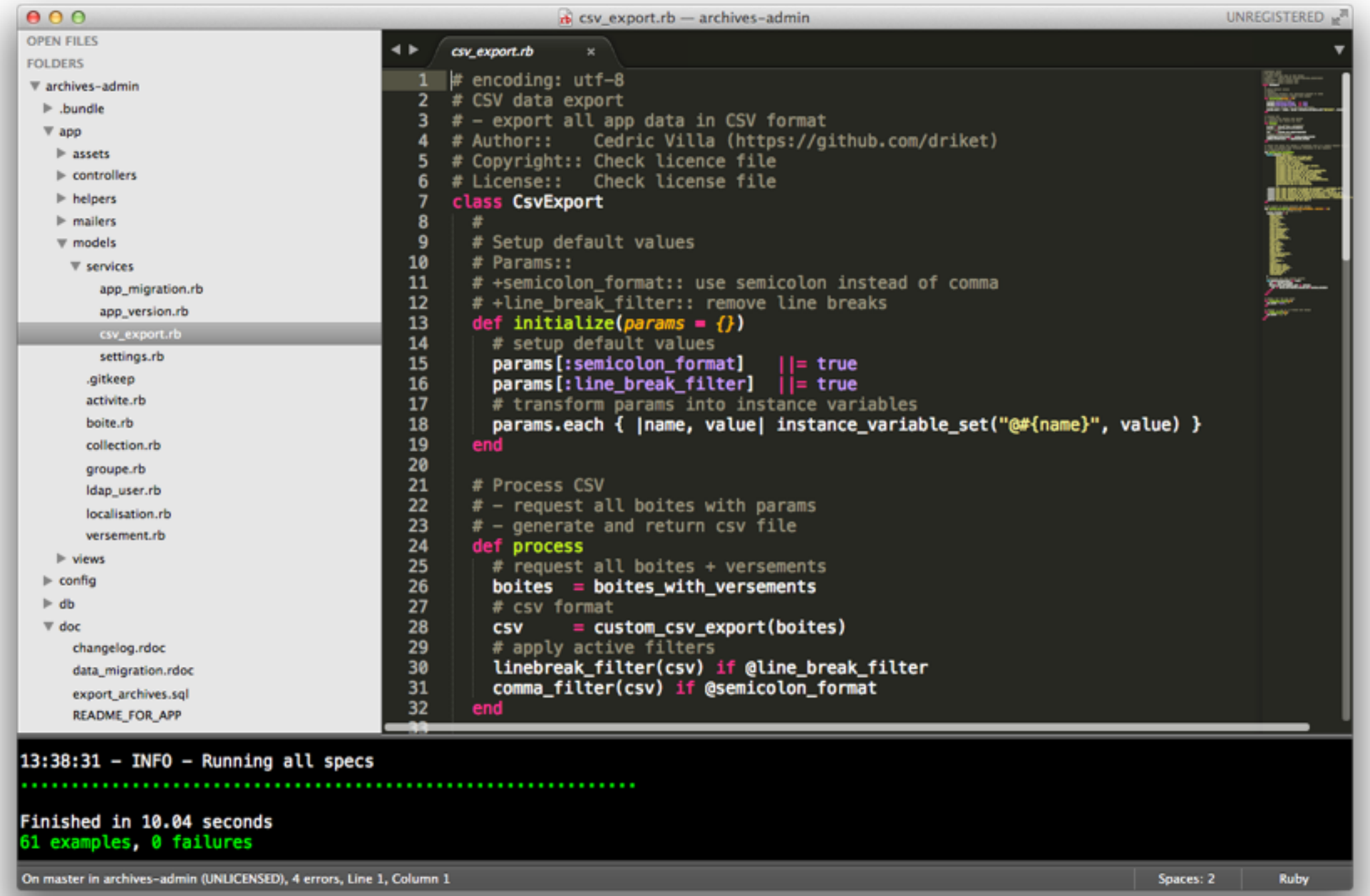
Execution des tests en ligne de commande



Bonus : générer le cahier de recettes



Intégration dans IDE



Échouez **tôt** et
échouez **souvent**.

cedric.villa@loria.fr

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**

cedric.villa@loria.fr

Implémentation en 3 étapes

le test échoue

le test réussi

tous les tests doivent passer

cedric.villa@loria.fr

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-rspec-alternative/
  (alternative à RSPEC pour Java)

- http://blog.octo.com/jai-limpression-decrire-mes-tests-en-double/
  (gestion doublons dans les tests)