Temario

Unidad 5

a) Testing

Temas: Introducción testing.Testing y cobertura. Historias de usuarios y relación con testing. Entorno de testing.

b) Test unitarios en Minitest

Temas: Estructura de un test. Métodos para validar. Fixtures y YAML.

c) Test funcionales con Minitest

Temas: Devise y testing. TDD.

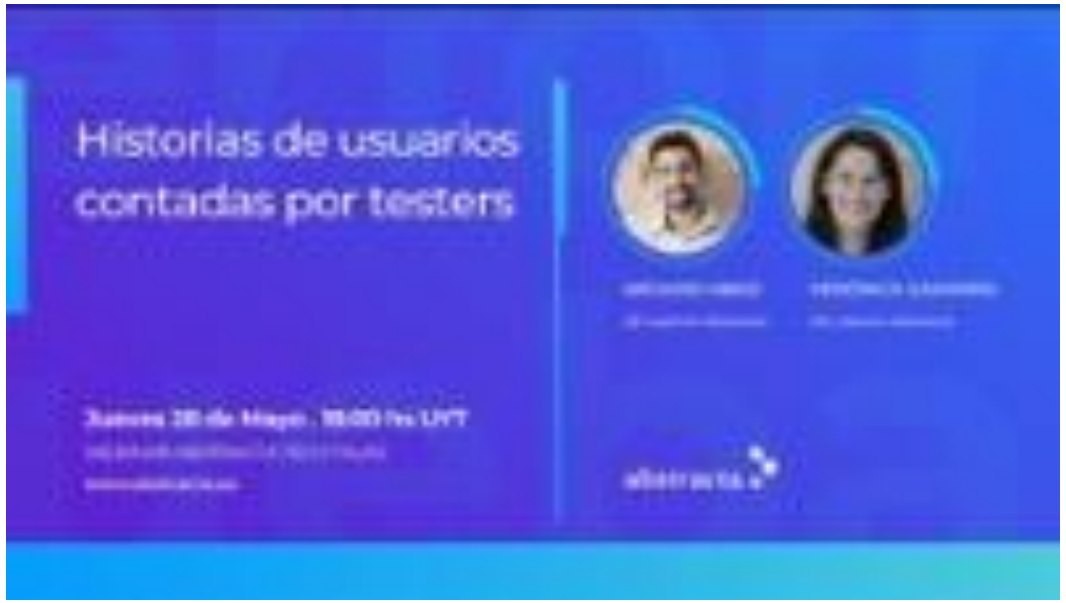
Unidad 5

Testing

Entorno testing y estructura

•El archivo de configuración de **test** es: config/environments/test.rb.
•En el Gemfile, las gemas que estén en el grupo **test** solo están disponibles en ese ambiente.
•En el archivo config/database.yml se encuentra la configuración de la base de datos de pruebas bajo la llave **test**.

Historias de usuarios y relación con testing



Unidad 5

Test unitarios con Minitest

Prueba, fixtures y asserts

Prueba de Modelos:
Al crear un modelo con rails g model, se crean también:
•Un archivo de prueba en la carpeta test/models (p.e. article_test.rb).
•Un archivo para definir los datos iniciales en test/fixtures (p.e. articles.yml).
Lo principal que se debe probar en los modelos es:
•Validaciones.
•Scopes.
Métodos con lógica

En general, las pruebas se pueden dividir en dos grandes categorías: unitarias y de integración.
Las **pruebas unitarias** se escriben para probar la lógica de métodos específicos de la aplicación, aislando sistemas externos como bases de datos, otros servidores, etc.
Por ejemplo, una prueba para un método que calcula el número de palabras de un texto sería una prueba unitaria.

Unidad 5

Test funcionales con Minitest

Fixtures con YAML

Para el propósito de hacer pruebas unitarias de un mailer, los fixtures se utilizan para proporcionar un ejemplo de cómo el resultado debe lucir. Debido a que estos son ejemplos de mensajes de correo electrónico, y no datos de Active Record como los otros fixtures, que se mantienen en su propio subdirectorio aparte de los otros fixtures. El nombre del directorio dentro de test/fixtures corresponde directamente al nombre del mailer. Por lo tanto, para un mailer llamado UserMailer, los fixtures deben residir en el directorio test/fixtures/user_mailer.

Pruebas funcionales unitarias

Las pruebas funcionales para mailers envuelve más que sólo comprobar que el cuerpo del correo electrónico, los destinatarios y así sucesivamente están correctas. En las pruebas de correo funcionales.

```
27  require 'test_helper'
28
29  class UserControllerTest < ActionDispatch::IntegrationTest
30    test "invite friend" do
31      assert_difference 'ActionMailer::Base.deliveries.size', +1 do
32        post invite_friend_url, params: { email: 'friend@example.com' }
33      end
34      invite_email = ActionMailer::Base.deliveries.last
35
36      assert_equal "You have been invited by me@example.com", invite_email.subject
37      assert_equal 'friend@example.com', invite_email.to[0]
38      assert_match(/Hi friend@example.com/, invite_email.body.to_s)
39    end
```

Definición TDD (BDD)

TDD: es un proceso para escribir pruebas automatizadas en el que primero se escribe una prueba fallida, después se implementa el código para que funcione y, por último, se mejora el código verificando que la prueba siga pasando.

También existe BDD: es muy similar pero el lenguaje cambia un poco para que sea entendible tanto por programadores como por personas que tienen mayor conocimiento del negocio.

Material complementario de la unidad

Link a video relacionado

1. https://www.youtube.com/watch?v=IvphiOxVIQg

Link a lectura complementaria

1. https://daniel-morales.gitbook.io/guias-de-rails-espanol/xvii-testing-de-aplicaciones-rails/2-introduccion-a-las-pruebas

Link a investigación relacionada

1. https://daniel-morales.gitbook.io/guias-de-rails-espanol/