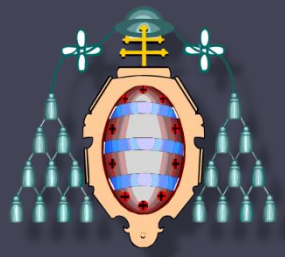
ES
Español



Universidad de Oviedo

Escuela de Ingeniería Informática

ARQUITECTURA DEL SOFTWARE

Arquitectura del Software

Lab. 09

- BDD
- Pruebas de Aceptación

2020-21

José Emilio Labra Gayo
Pablo González
Irene Cid
Paulino Álvarez

Pruebas de aceptación y BDD

Pruebas que pueden ejecutarse con el cliente
- Si pasan, el producto es aceptado

Behaviour-Driven-Development (BDD)
- Variante de TDD (test driven development)
  - Pruebas basadas en comportamiento
  - Relacionado con ATDD (Acceptance-Test Driven Development)
- Comportamiento = Historias de usuario
- También conocido como: Especificación por ejemplos
- Objetivo: Especificaciones ejecutables
- Algunas herramientas:
  - cucumber, jBehave, concordion

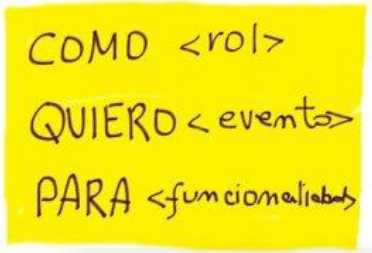
BDD - Historias de usuario

- Breves (en teoría se escriben en tarjetas)
- Deben ser legibles (y aprobadas) por expertos de dominio (negocio)
- Otros atributos deseables:
  - Independientes (sin interrelaciones fuertes)
  - Negociables (sin detalles concretos)
  - Con valor para el usuario
  - Estimables (para incluirlas en Sprints)
  - Pequeñas (en otro caso considerar dividirlas)
  - Se pueden validar (automatizar)

BDD - Historias de usuario

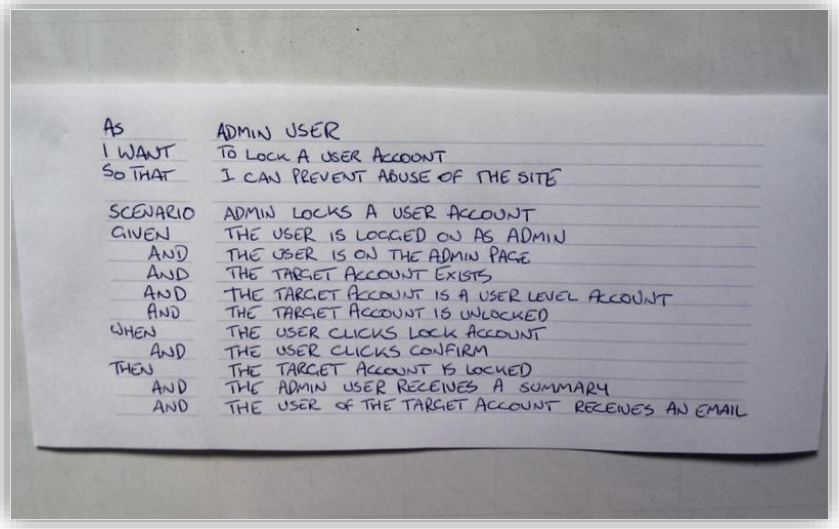
Característica: *Título (descripción de historia)*

Como [rol]
Quiero [característica]
Para [beneficio]

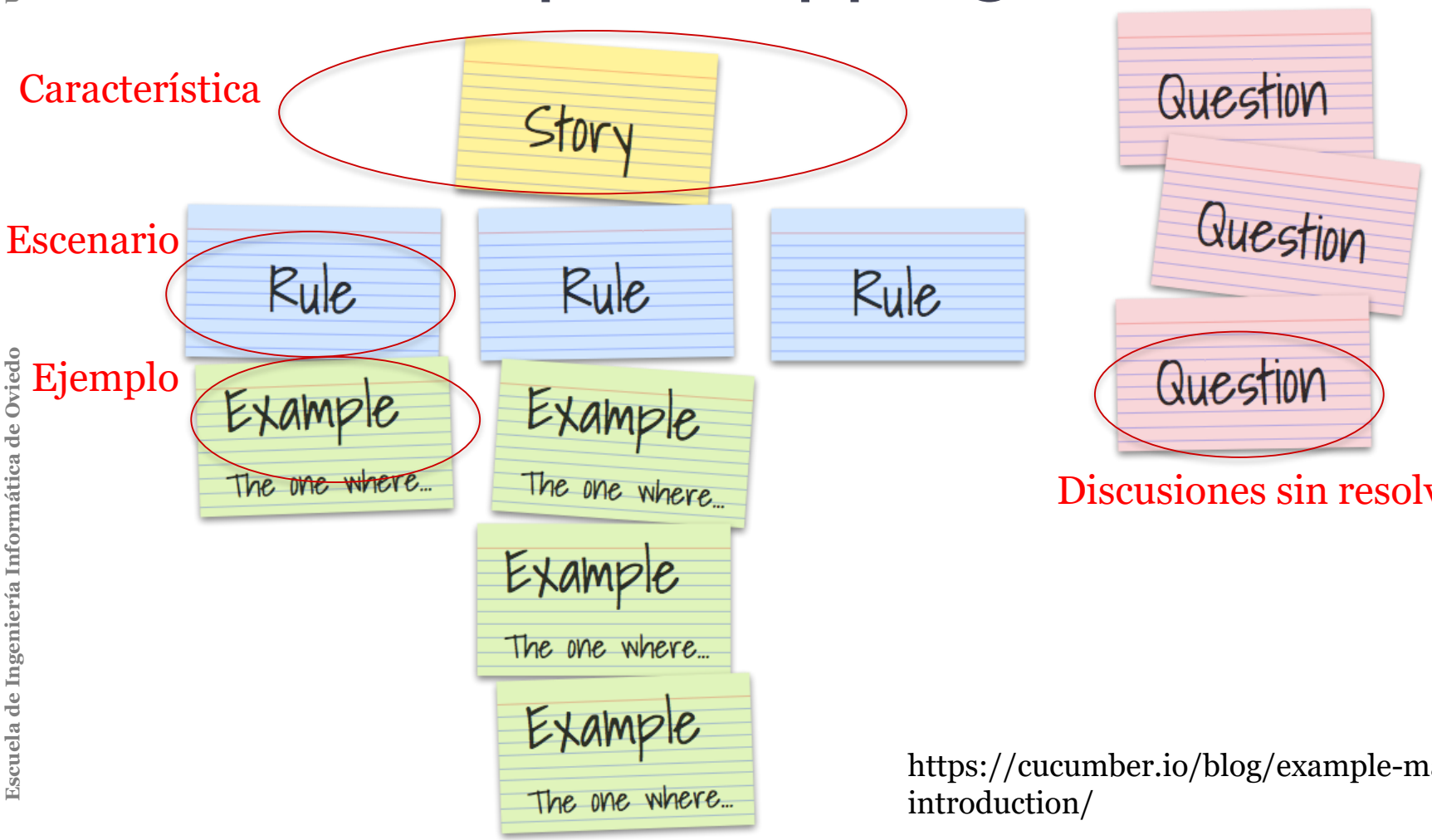


Varios escenarios

Dado [Contexto]
Y [un poco más de contexto]
Cuando [Suceso]
Entonces [Resultado]
Y [otro resultado]



BDD – Example Mapping



https://cucumber.io/blog/example-mapping-introduction/

BDD con Cucumber

- Cucumber = desarrollado en Ruby (2008)
  - Rspec (Ruby), jbehave (Java)
- Se basa en Lenguaje Gherkin (lenguaje interno pare definir las historias)
  https://cucumber.io/docs/gherkin/
  Puede usarse en diferentes idiomas (asturiano, …)
- Historias de usuarios enlazadas con definiciones de los pasos
  - Definiciones de pasos se ejecutan para validar las historias de usuario

BDD con Cucumber

Característica: Describe una funcionalidad

- Una funcionalidad puede tener varios escenarios
- Escenario:
  - Comportamiento del sistema en un contexto determinado
    - *Given*: Contexto
    - *When*: Interacción con el sistema
    - *Then*: Comprueba el resultado
    - *Examples*: Datos concretos

BDD con Cucumber

Referencias:

- Java: cucumber-jvm
  - https://github.com/cucumber/cucumber-jvm
- Eclipse support:
  - http://cucumber.github.io/cucumber-eclipse/
- Visual Studio Code: Cucumber (Gherkin) Full Support
  - https://marketplace.visualstudio.com/items?itemName=alexkrechik.cucumberautocomplete

Ejemplo BDD

- Crear una historia de usuario (paso a paso)
  - Instalar Cucumber
  - Escribir el primer escenario en Gherkin
  - Escribir las definiciones de los pasos en el lenguaje de programación escogido
  - Ejecutar Cucumber

Ejemplo BDD con Cucumber

- Depende del lenguaje de programación o entorno
  - Java/Javascript/Python/…
  - Instalación en: https://cucumber.io/
- React ejemplo: `https://github.com/Arquisoft/radarin_0`

jest-cucumber: Módulo que permite crear historias con Gherkin y convertirlas en test para ejecutar con Jest.

```
$ npm install --save-dev puppeteer jest-cucumber
```

jest-puppeteer: Módulo que nos permitirá ejecutar los test en un navegador por defecto. También se puede configurar para ser usado con Selenium.

```
$ npm install --save-dev puppeteer jest-puppeteer
```

Ejemplo BDD

- Ejemplo de historia de usuario en Node.js.

```
Feature: Registering a new user

Scenario: The user is not registered in the site
  Given An unregistered user
  When I fill the data in the form and press submit
  Then A welcome message should be shown in the screen

Scenario: The user is already registered in the site
  Given An already registered user
  When I fill the data in the form and press submit
  Then An error message should be shown in the screen
```

Característica

Escenario

Escenario

.../webapp/e2e/features/register-form.feature

Ejemplo BDD

.../webapp/e2e/steps/register-form-steps.js

```
test('The user is not registered in the site', ({given,when,then}) => {

  let email;
  let username;

  given('An unregistered user', () => {
    email = "newuser@test.com"
    username = "newuser"
  });

  when('I fill the data in the form and press submit', async () => {
    await expect(page).toMatch('Hi, ASW students')
    await expect(page).toFillForm('form[name="register"]', {
      username: username,
      email: email,
    })
    await expect(page).toClick('button', { text: 'Submit' })
  });

  then('A welcome message should be shown in the screen', async () => {
    await expect(page).toMatch('Welcome to ASW')
  });
});
```

Ejemplo BDD [Configuración]

- jest-config.js
  - Este archivo vincula todo junto
  - Dice a **jest** dónde están los archivos de prueba de pasos
  - Configuración para iniciar y desmontar las pruebas

```
module.exports = {
    testEnvironment: './custom-environment.js',
    testMatch: ["**/steps/*.js"],
    testTimeout: 30000,
    globalSetup: './global-setup.js',
    globalTeardown: './global-teardown.js',
    setupFilesAfterEnv: ["expect-puppeteer"]
}
```

Ejemplo BDD [Configuración]

- custom-environment.js
  - Configura cómo iniciar el navegador para realizar las pruebas
  - Usamos puppeteer para esta tarea
  - También se puede configurar con otros navegadores.
  - Usamos **headless=true** para ejecutar las pruebas en el sistema CI, pero podemos cambiarlo a falso para ejecutarlas localmente
  - El parámetro **slowMo** es útil para ralentizar las pruebas y ver qué está sucediendo.

```
var NodeEnvironemnt = require('jest-environment-node')
var puppeteer = require('puppeteer')
class CustomEnvironment extends NodeEnvironemnt {
    constructor(config, context){
        super(config, context)
    }
    async setup(){
        await super.setup()
        this.global.browser = await puppeteer.launch({
            headless: true,
            //slowMo: 20
        })
        this.global.page = await this.global.browser.newPage()
    }
    async teardown(){
        await this.global.browser.close()
        await super.teardown()
    }
}
module.exports = CustomEnvironment
```

Ejemplo BDD [Configuración]

- global-setup.js
  - Configura cómo lanzar el sistema
  - Para probar esta aplicación necesitamos: la base de datos, restapi y webapp
  - La base de datos y el restapi se inician utilizando dos scripts adicionales
  - **BROWSER=none** indica que no queremos iniciar el navegador con npm start (ya configuramos cómo iniciar el navegador)

```
const { setup: setupDevServer } = require("jest-dev-server")
module.exports = async () => {
    await setupDevServer([
    {
        command: 'node start-db.js',
        launchTimeout: 100000,
        debug:true,
        port: 27017,
    },
    {
        command: 'node start-restapi.js',
        launchTimeout: 60000,
        debug:true,
        port: 5000,
    },
    {
        command: 'BROWSER=none npm start',
        launchTimeout: 60000,
        debug: true,
        port: 3000
    }])
}
```

Ejemplo BDD [Configuración]

- start-db.js
  - Crear en memoria una instancia de mongodb
  - Reutiliza la función startdb (). También se utiliza para lanzar una base de datos en memoria para las pruebas unitarias de restapi

```
const server = require('../../restapi/tests/server-for-tests')
server.startdb()
```

- start-restapi.js
  - Crear el restapi server y se conecta a la base de datos en memoria de mongo
  - Reutiliza la función startserver() del restapi (también se usa en las pruebas unitarias del restapi)

```
const server = require('../../restapi/tests/server-for-tests')
server.startserver()
```

Ejemplo BDD [Configuración]

- Configuración package.json en apartado scripts:

```
"test:e2e": "cd e2e && jest",
```

- Ejecutando las pruebas:
  - **npm run test:e2e**

Ejemplo BDD

- Resultado

```
 PASS  feature/step-definition/register-form-steps.js (7.515s)
  Registering a new user
    ✓ The user is not registered in the site (5146ms)
    ✓ The user is already registered in the site (523ms)

Test Suites: 1 passed, 1 total
Tests:       2 passed, 2 total
Snapshots:   0 total
Time:        7.919s, estimated 11s
Ran all test suites.
```

Ejemplo cucumber + selenium + Java (Spring-boot) de años anteriores

https://github.com/arquisoft/votingSystem0

Pruebas para navegadores

- Automatización con Navegadores
  - https://cucumber.io/docs/guides/browser-automation/
  - Otros herramientas
    - Selenium WebDriver - http://docs.seleniumhq.org/
    - Capybara - http://teamcapybara.github.io/capybara/
    - Watir - https://watir.com/
    - Serenity - http://serenity-bdd.info

Selenium

- Selenium IDE: permite registrar acciones
  - Firefox y Chrome plugins
- Genera código para ejecutar esas acciones.
- Configuración de Travis

  https://lkrnac.net/blog/2016/01/run-selenium-tests-on-travisci/

Bibliografía y enlaces de interés

- User Story Mapping by Jeff Patton
  - **User Story Mapping: Discover the Whole Story, Build the Right Product, 1st Edition**
    https://www.amazon.com/User-Story-Mapping-Discover-Product/dp/1491904909
- Historias de Usuario
  - Scrum. Historias de Usuario (Fernando Llopis, Universidad de Alicante)
    https://fernandollopis.dlsi.ua.es/?p=39
  - User stories with Gherkin and Cucumber (Michael Williams)
    https://medium.com/@mvwi/story-writing-with-gherkin-and-cucumber-1878124c284c
  - Cucumber 10 minutes tutorial (JS)
    https://docs.cucumber.io/guides/10-minute-tutorial/
- Pruebas basadas en navegador
  - Automated UI Testing with Selenium and JavaScript
    https://itnext.io/automated-ui-testing-with-selenium-and-javascript-90bbe7ca13a3

Primer Release (próxima semana)

- Mandar **contribuciones** (issue) por cada miembro del equipo el día antes
- Presentación de 15 mínutos
- Cobertura de pruebas unitarias al 30% optimo
- Documentación actualizada
- Despliegue en **heroku**
- No necesario implementar pruebas de aceptación
- Rama especifica **release** o etiqueta en el repositorio