
ubykuo
assert(title, “Testing for n00bz”);


To test
or not
to test

Resolver bugs en entornos productivos

- Interrumpe la experiencia del usuario.
- Los desarrolladores tienen que hacer context switch entre la tarea actual y atender el bug.
- Interrumpe el ciclo de desarrollo y no siempre es diagnosticado por el mismo desarrollador que lo implementó.
- **Si se repite periódicamente, deteriora la relación de confianza que se construye con el cliente.**

Beneficios de testing automatizado

- Confianza del equipo de desarrollo al desarrollar, re-diseñar y refactorizar.
- Reducción de bugs que llegan a producción.
- Detección temprana de los bugs.
- Más confianza del cliente hacia el equipo para realizar grandes tareas.


Testing,
un mundo
aparte

Test de Unidad

Test de Integración

Test de end-to-end

Smoke testing

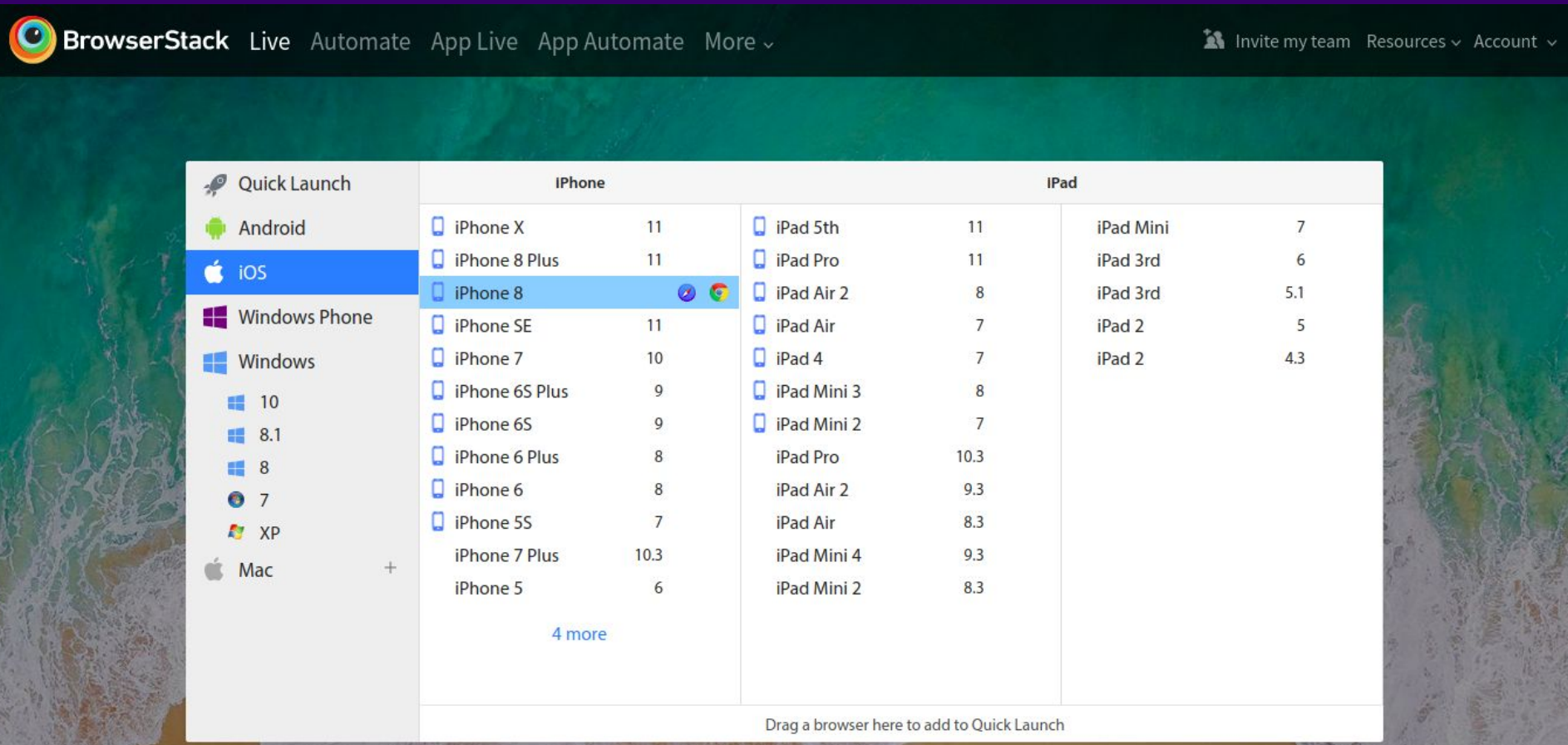
Test de carga

Tests de Unidad

"Identify the lines of code that caused the failure"

- El objetivo es verificar el correcto funcionamiento de un componente de manera aislada (sin dependencias).
- Nos ayuda a identificar cual es la interfaz y el uso que se va a hacer del componente de manera externa.
- Ayuda a identificar responsabilidades en los componentes.
- Nos da feedback más específico sobre el error (mensaje, línea, etc).
- Tardan poco en ejecutarse
- Tiene poca dependencia con el contexto/entorno en el que se ejecutan (hardware, conectividad, etc)

Test de unidad - BrowserStack




Demo time

Tests de integración

Verificar que varios componentes funcionan bien en conjunto.

Ejemplo: testear un endpoint de la API, y verificar que la respuesta sea la correcta, y que los efectos secundarios fueron los esperados, logging, notificaciones, etc.

Ejemplo - Test de integración

```
describe('GET all catalog ', () => {
  it('it should get all the items from the catalog', (done) => {
    chai.request(server)
      .get('/api/catalog/')
      .set('authorization', 'Bearer ' + users.paniolero.token)
      .end((err, res) => {
        if (err) done(err)
        res.should.have.status(200)
        res.body.should.have.property('success').eql(true)
        res.body.should.have.property('catalog').length(catalogData.length)
        done()
      })
  })
})
```

Test end-to-end

“Validar funcionalmente simulando el uso que le daría un usuario real a la plataforma”

Ejemplo - Test e2e

```
describe('Login page', function () {
  describe('reset password form', function () {
    it('should show an error message if the user doesn\'t exist', function () {
      element(by.id('forgot-password-link')).click();

      element(by.id('inputForgetEmail')).sendKeys('wrongemail@example.org');
      element(by.id('resetPassword')).click();

      var helpBlock = element(by.id('reset-help-block'));

      expect(helpers.hasClass(helpBlock, 'visible')).toBe(true);
      expect(helpBlock.getText()).toContain('No user registered');
    });
  });
```

Smoke testing

Tests simples que apuntan a verificar que las funcionalidades críticas del sistema están en funcionamiento.

Por ejemplo: correr un tests e2e que loguee un usuario de prueba en producción y verifique que el login proceda sin problemas, en caso de fallo, notifica al equipo.


Demo time, again

Test de carga (*load*) y estrés (*stress*)

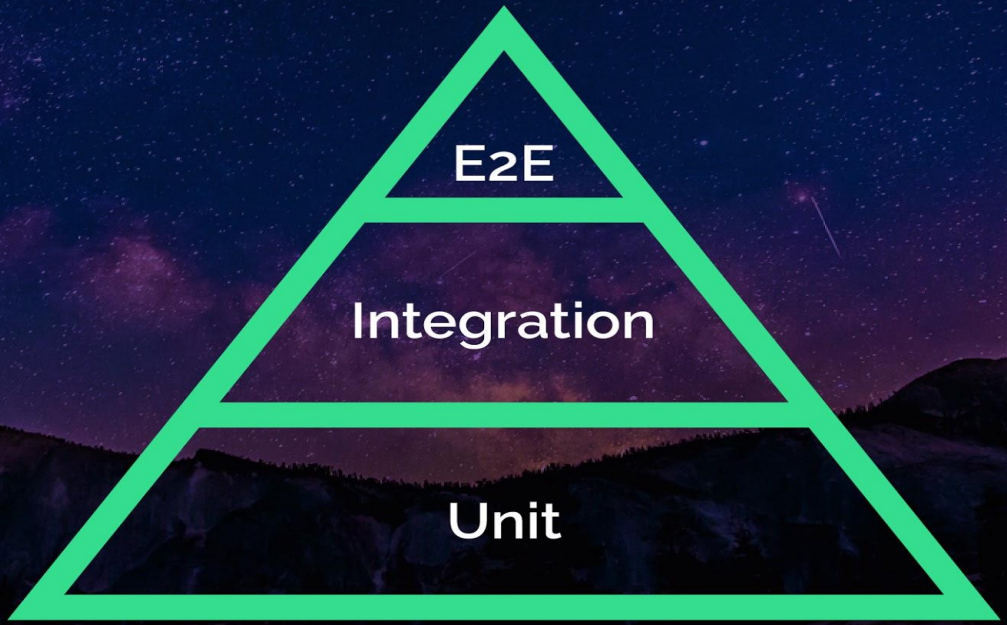
Los **tests de carga** miden el rendimiento del sistema frente a cargas de trabajo por encima del promedio.

Se suele utilizar para determinar el número máximo de usuarios que puede manejar un sistema.

Los **tests de estrés** buscan encontrar los límites del sistema en términos de carga, cuánta puede soportar antes de dejar de responder.

Un test de estrés podría intentar testear si un sistema ecommerce puede tolerar la carga de un CyberMonday.

La Receta



"Write lots of small and fast unit tests. Write some more coarse-grained tests and very few high-level tests that test your application from end to end."

Integración continua ¿Cuando se ejecuta cada test?

Desarrollo (local): test de unidad activos durante el desarrollo.

Staging (preproducción): test de unidad, integración y e2e.

Producción: smoke tests y tests de carga.

Referencias

https://codeahoy.com/2016/07/05/unit-integration-and-end-to-end-tests-finding-the-right-balance/

https://testing.googleblog.com/2015/04/just-say-no-to-more-end-to-end-tests.html

https://martinfowler.com/articles/practical-test-pyramid.html

https://martinfowler.com/articles/qa-in-production.html

https://martinfowler.com/bliki/IntegrationTest.html

https://www.sitepoint.com/javascript-testing-unit-functional-integration/

http://softwaretestingfundamentals.com/smoke-testing/

https://www.blazemeter.com/blog/performance-testing-vs-load-testing-vs-stress-testing

“Do or do not, there is no try… but

before, write a test you must”