Anexo

Documentación de Testing

Primer sprint:

Comprobación en Base de Datos.

Una de las primeras labores realizadas por parte del grupo fue la creación de la Base de Datos, por ello, también fue una de las primeras cosas para probar. Para realizar dicha actividad se analizó la base de datos realizando diversas consultas para chequear datos: consultas a tablas para ver la información y por otro lado, modificaciones e inserciones. Las pruebas resultaron satisfactorias, salvo por el pequeño error en el campo que tenían en común las tablas de “comidas”, donde la columna de “tiempo de preparación” estaba mal expresada: el tiempo se estaba brindando solamente en segundos, cuando debía ser en minutos. Esto fue rápidamente corregido por el DBA.

Luego de estas pruebas y consultas contra la base de datos, se procedió a testear, de manera más visual, que las diversas URL a las que se podía acceder desde el proyecto trajeran correctamente la información hacia el browser: por ejemplo, si uno accedía a la URL **http://localhost/Proyecto_Touch/public/menu/comidas/pollos**

Debía ver los datos traídos desde la BD respecto a la tabla “pollos”, con sus columnas de ‘descripción’ y ‘precio’, por ejemplo. Con ello se probó visualmente que esto se realizaba correctamente, y se procedió a insertar algunos nuevos elementos para corroborar que posteriormente se iban a poder añadir datos.

Pruebas Unitarias (PHPUnit).

Para las siguientes pruebas se procedió a realizar pruebas unitarias, de la mano de PHPUnit. Dicho framework ya viene instalado por default con Laravel.

Las pruebas consistieron en corroborar que las URL configuradas por el archivo “routes.php”, llamaran correctamente a sus correspondientes controladores, y estos procedieran a llamar correctamente los datos y la vista. Por ejemplo, al acceder a la URL **http://localhost/Proyecto_Touch/public/menu/categorias** se debe mostrar cierta información que puede ser chequeada visualmente, para ello se creó el siguiente test, el cuál fue creado por “Consola”, mediante el comando php artisan make:test MenuTest. En esta clase Test, se ubican los diversos TestCase para

chequear los diferentes controladores, URLs y vistas que debe mostrar nuestra aplicación. El test de categorías es el siguiente:

:

TestTest.php | Find Results | .env | TestController.php

```php
<?php

use Illuminate\Foundation\Testing\WithoutMiddleware;
use Illuminate\Foundation\Testing\DatabaseMigrations;
use Illuminate\Foundation\Testing\DatabaseTransactions;

class MenuTest extends TestCase
{
    /**
     * Test para probar el contenido de los menúes al ser traídos de la BD.
     *
     * @return void
     */

    public function testMenuCategorias()
    {
        $this->visit('/menu/categorias')
             ->see('Comidas');
    }

```

Se comprueba que al acceder a la URL /menú/categorías, se deba mostrar el mensaje de “Comidas” en alguna parte de la página. Este mecanismo se utilizó para corroborar las demás pantallas, chequeando que se traiga correctamente alguna información característica de esa página, sabiendo así que la consulta y la pantalla muestran información correcta.

A continuación, se muestran más de los TestCase hechos:

```php
public function testMenuPollos()
{
    $this->visit('/menu/comidas/pollos')
         ->see('pollo con fritas');
}

public function testMenuCarnes()
{
    $this->visit('/menu/comidas/carnes')
         ->see('asado de tira');
}

public function testMenuCerdos()
{
    $this->visit('/menu/comidas/cerdos')
         ->see('lechon asado para 2');
}
```

En los TestCase se sigue la misma lógica, entrar a la URL y corroborar que se muestra información correcta:

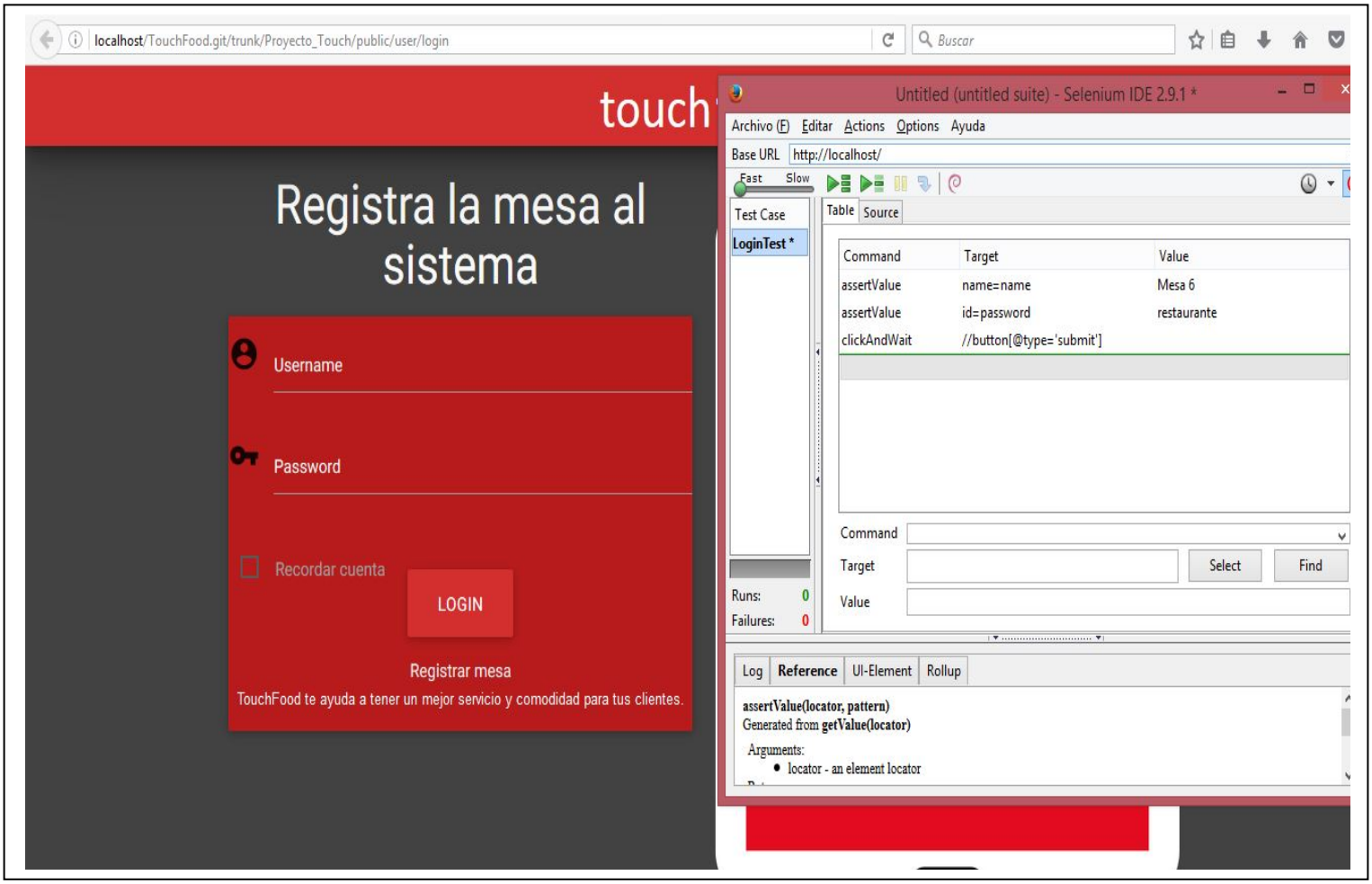
```php
public function testMenuPicadas()
{
    $this->visit('/menu/comidas/picadas')
         ->see('tablita caliente');
}

public function testMenuMinutas()
{
    $this->visit('/menu/comidas/minutas')
         ->see('hamburguesa');
}

public function testMenuSopas()
{
    $this->visit('/menu/comidas/sopas')
         ->see('sopa clasica');
}
```

Resultado de ejecución de algunos test:

Segundo sprint:

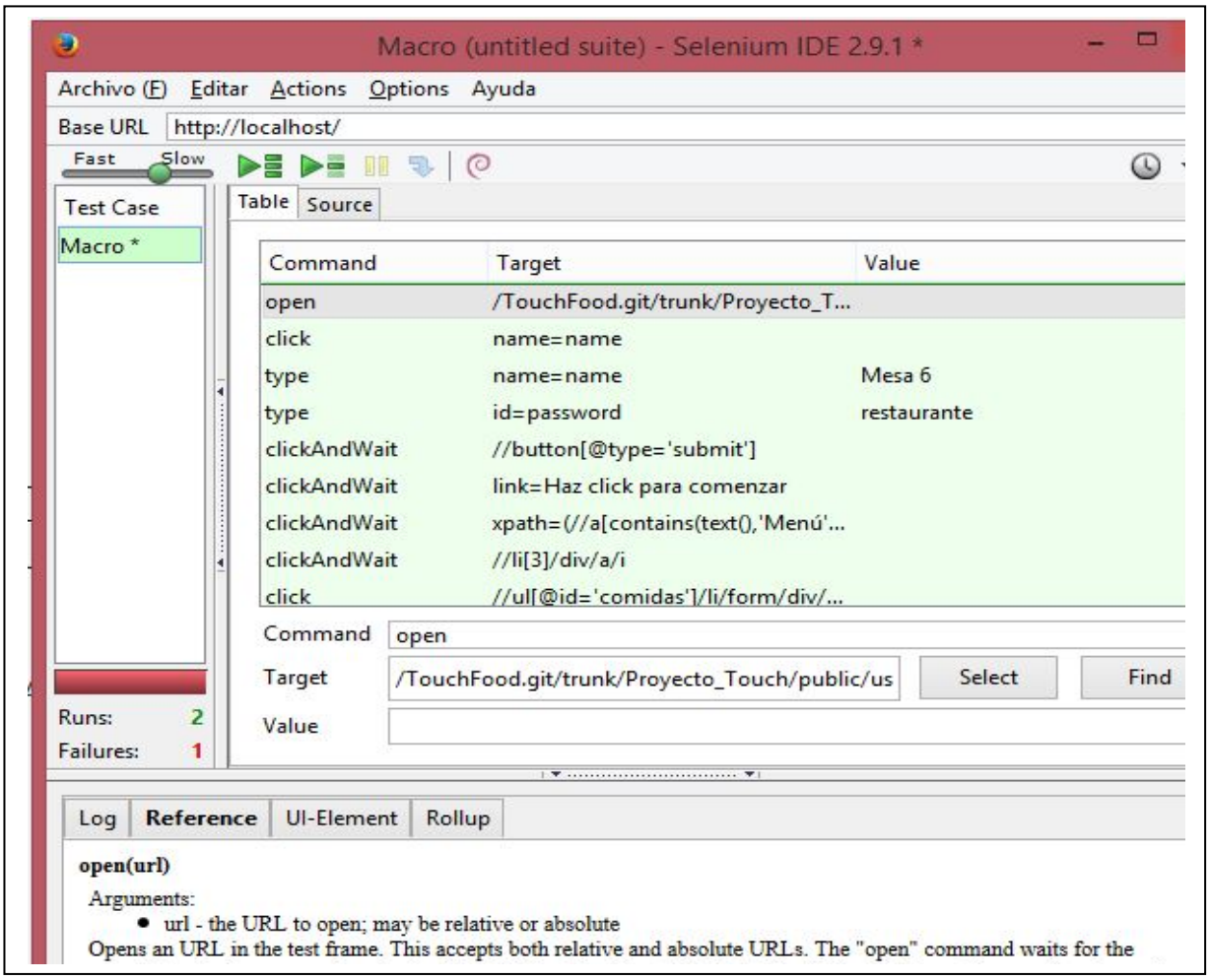
PHPUnit:
Se continúo utilizando PHPUnit para hacer las pruebas unitarias sobre las páginas que no habían sido analizadas y las nuevas que fueron agregadas:
se comprobó, buscando un texto específico, las páginas de Login para las mesas, el Admin, el listado de comidas que realiza uno al hacer el pedido (listado de pedidos) y pago, entre otras.

Selenium:
Por otro lado, se pasó a usar una tecnología nueva en este sprint para realizar test: Selenium IDE.
Esta herramienta es un plugin del browser Firefox (existen para otros navegadores). Selenium nos permite realizar pruebas “en caliente”, podemos, por ejemplo, seleccionar diferentes campos inputs y realizar un test mediante el cual se compruebe que estos valores sean los correctos, y con ello ver si la página reacciona como debe hacerlo:



En el ejemplo de la imagen anterior, se comprueba que los campos de Username y Password sean los que el sistema acepta: si se coloca otro valor y ejecuta el test, este fallará. Esta herramienta nos brinda facilidades para realizar pruebas y de manera rápida.

Por otro lado, también se usó Selenium para realizar un macro, el cual simula la operación real que podría tener un usuario: elección de comidas/bebidas/postres y realizar el pago. Si la funcionalidad de la aplicación no varía, solamente si se hizo, por ejemplo refactor, este macro debería funcionar antes y luego de ello.



En el ejemplo anterior, se simula el login de una mesa y la operación que tendrá un usuario, pidiendo diversas comidas/ bebidas, y luego seleccionando la opción de pagar.

Tercer sprint:

Durante el tercer sprint, se procedió a hacer una evaluación general de la aplicación, probando todas las funcionalidades. Se utilizó un browser para ir chequeando que cada funcionalidad realicé lo que debía hacer (se usó Google Chrome y Mozilla Firefox). Por otro lado, también se volvió a aprovechar la utilidad de Selenium IDE (plugin para Mozilla), para realizar pequeñas pruebas sobre el navegador y algunos macros para realizar pruebas sobre toda la aplicación, como si un usuario real la manipulara.

Por otro lado, se realizaron dos pruebas más, a cargo de terceros ajenos a la aplicación:
- Prueba con usuario técnico: se otorgó la URL de la aplicación a un tercero para que realice la tarea de “beta tester”, probando el software y dando un feedback respecto a lo que se encontró. El resultado fue muy positivo, ya que tuvo una crítica muy buena, reconociendo lo vistoso del front- end y que la aplicación realizaba de manera correcta sus funcionalidades.

- Prueba con usuario no técnico: mediante el localhost, se otorgó el manejo de la aplicación a un tercero no técnico para que simule una operación real con el software. La prueba arrojó comentarios alentadores por parte del usuario, ya que destacó su buen manejo y comodidad para operar, sin complicaciones ni prestación a confusiones.

La aplicación superó las pruebas planteadas por el equipo en general, lo cual nos brinda una experiencia ampliamente satisfactoria, a nivel cursada de la materia como a nivel profesional, ya que nos ha ayudado a tener un fuerte contacto con lo que sería una situación real de trabajo.

Control de versiones		
Número	**Fecha**	**Motivo**
1	ago-16	Se crea el documento
2	ago-16	Se agregan las funcionalidad correspondientes al primer sprint
3	sep-16	Se edita el formato del documento
4	sep-16	Se editan cantidad de horas trabajadas por cada integrante
5	oct-16	Se aprueba por el grupo
6	oct-16	Se agregan las funcionalidades correspondientes al segundo sprint
7	oct-16	Se actualiza el formato del documento
8	nov-16	Se editan la cantidad de horas trabajadas por cada integrante
9	nov-16	Se aprueba por el grupo
10	nov-16	Agregado de funcionalidades correspondientes al tercer sprint
11	nov-16	Se aprueba por el grupo