



PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE
ESCUELA DE INGENIERÍA
DEPARTAMENTO DE CIENCIA DE LA COMPUTACIÓN

IIC3745 — Testing
2023 - 2

Tarea 4

Fecha de entrega: Viernes 24 de Noviembre del 2023 a las 23:59

Información General

El objetivo de esta tarea es aplicar el tercer nivel de testing descrito en el modelo V a una aplicación web: System Testing. Además se resaltara la importancia de mantener el set de pruebas actualizado a medida de que se avanza en el desarrollo. Los estudiantes en esta tarea podrán aplicar de manera practica los conceptos vistos en cátedra.

Objetivos

- Profundizar conocimientos del framework Ruby on Rails para desarrollar aplicaciones web aplicando técnicas de testing.
- Profundizar conocimientos sobre herramientas y buenas prácticas de desarrollo de software.
- Aplicar conocimientos adquiridos durante el curso para asegurar calidad de un software.

Contexto

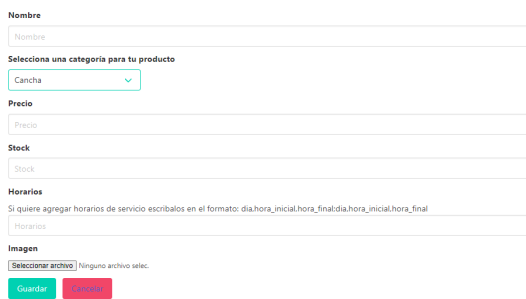
La empresa DCCompany se contacta con ustedes, les comentan que quedaron satisfechos con los tests que crearon y que hayan podido detectar defectos en la aplicación. La empresa ahora les pide dar un paso más y crear tests de sistema para probar toda la aplicación simulando las acciones de un usuario. Además señalan que han estado teniendo problemas con las reservas de canchas ya que los usuarios están reservando horas que no corresponden a las promocionadas. Se les pide que en este ciclo realicen los cambios necesarios para asegurarse que las reservas se hagan solamente en los horarios señalados.

Tareas a realizar

- Mejorar las reservas de canchas. En específico se les pide realizar las modificaciones necesarias de tal manera de asegurarse de que los clientes realicen las reservas dentro del horario publicado. Actualmente se dispone de un campo para los clientes donde uno puede elegir una fecha y hora que no necesariamente corresponde a la promocionada.

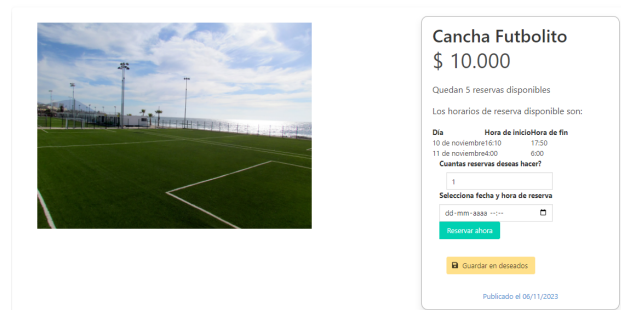
Para solucionar este problema tienen completa libertad de modificar la aplicación a su gusto lo cual podría incluir agregar métodos de validación más estrictos, modificar la forma en que se publica la cancha y/o la forma en que los clientes realizan la reserva. Independiente de lo que hagan deberán explicar en su readme los cambios realizados en la aplicación detallando nuevos modelos, cambios en modelos existentes o nuevos flujos en la aplicación que deberán seguir los clientes. Importante los cambios realizados no deben eliminar otras funcionalidades como por ejemplo la venta de implementos deportivos.

Crear Producto



Formulario de creación de producto. Campos: Nombre (input), Selección de categoría (dropdown con 'Cancha' seleccionada), Precio (input), Stock (input), Horarios (input con placeholder 'Si quiere agregar horarios de servicio escribalos en el formato: dia.hora_inicial.hora_final:dia.hora_inicial.hora_final'), Imagen (botón 'Seleccionar archivo' y 'Guardar').

(a) Formulario creación cancha



Formulario de reservas de cancha. Incluye una imagen de la cancha y un panel de reservas. El panel muestra: 'Cancha Futbolito \$ 10.000', 'Quedan 5 reservas disponibles', 'Los horarios de reserva disponible son:', una tabla de horarios (10 de noviembre 10:00-17:30, 11 de noviembre 08:00-16:00), 'Cuentas reservas desees hacer?' (input con '1'), 'Selecciona fecha y hora de reserva' (calendar icon), 'Reservar ahora' (botón verde), 'Guardar en favoritos' (botón amarillo), y 'Publicado el 06/11/2023'.

(b) Formulario reservas

Figura 1: Sistema de reserva de canchas que se busca mejorar

- Según los cambios que realicen para el punto anterior deberán actualizar el set de pruebas de la entrega anterior. La creación de nuevos tests dependerá completamente de los cambios que realicen y deberán cumplir los mismos mínimos que la tarea 3. Se recomienda señalar en el readme brevemente los cambios realizados en los tests de tal manera que el corrector los pueda identificar fácilmente.
- Mantener un 100% de coverage en los modelos y controladores de la aplicación que se pidieron en la entrega anterior.
- Crear 3 tests de sistema probando la navegación de la pagina. Para cada tests deberán definir una vista inicial por ejemplo el landing page y una vista final a la cual deberán llegar mediante su test interactuando con la pagina por ejemplo apretando botones o links. Deberán incluir en el README

claramente el punto inicial, el trayecto a seguir y el punto final de cada test. Se recomienda por temas de orden incluir además el mismo detalle como comentario en el código de cada test.

- Crear tests de sistema probando el formulario de creación de productos y otro a libre elección que no sea el de login. Para el formulario de creación de productos deberán probar al menos un caso happy path y 3 caminos alternativos. Para el formulario que elijan deberán probar un happy path y al menos 2 caminos alternativos.

Importante no es necesario ejecutar los tests de sistema en su integración continua. Para los que utilicen rspec pueden utilizar el parámetro `--exclude-pattern` para excluir los tests de sistema. Si usan MiniTest, ejecutar `rails tests` no ejecuta los tests de sistema ya que estos se ejecutan con `rails test:system`.

Recuerden que son libres de realizar cualquier cambio en la aplicación mientras no eliminen funcionalidades de la página.

Tests de sistema

Para esta entrega deberán crear tests de sistema utilizando la gema Capibara o utilizando Cypress mediante la gema Cypress-on-rails. Si ocupan capibara pueden trabajar con ella utilizando MiniTest o Rspec pero si deciden ocupar Cypress todo lo relacionado a sus tests de ubicara dentro de la carpeta cypress funcionando independientemente a los tests de la entrega anterior. En las próximas ayudantías se introducirá brevemente el uso de Capibara y Cypress mostrándoles ejemplos de como usar ambas herramientas para lo que se pide en esta tarea sin embargo puede que necesiten profundizar más para esto pueden revisar las documentaciones de ambas herramientas¹ o ejemplos en internet.

Para el correcto funcionamiento de sus tests deberán elegir algún driver el cual es un controlador que permitirá a sus tests conectarse y enviar instrucciones a su navegador. Esta configuración si usan Capibara se define en `test_helper` o `rails_helper` en la carpeta de sus tests y para Cypress se elije manualmente al iniciarlo.

Se recomienda trabajar con algún driver de Selenium (hay varias versiones dependiendo del navegador la más común es la de google chrome) ya que es un driver que permite la ejecución de javascript. Sin embargo en caso de no poder instalar Selenium y haber intentado las soluciones propuestas más adelante pueden utilizar como ultima instancia el driver `:rack_test` el cual no permite ejecutar javascript ya que es un driver más simple que simula un navegador con solo rails. Ojo que hay partes de la aplicación que utilizan javascript por lo que tienen que tener cuidado con lo que van a testear o realizar cambios para que no sea necesario ese javascript si están utilizando el driver `:rack_test`.

Tip: En caso de observar comportamientos extraños por ejemplo que se cargue correctamente la página pero que no encuentre algo que aparece en el html cargado. En ese caso se recomienda incluir algunos segundos de sleep antes de realizar los asserts del test para asegurarse de que la página se termine de cargar correctamente.

Problemas frecuentes con el setup de Selenium

El problema más común que se pueden topar con Selenium es que no se logre comunicar con su navegador esto se ve como que al ejecutar sus tests se abren varias pestañas del navegador las cuales no muestran nada y se cierran después de unos segundos junto con un error en consola. Este problema es porque Selenium no instalo la versión correcta del webdriver para conectarse a su navegador o que el driver que esta en la configuración es para un navegador que no tienen instalado. Para solucionarlo prueben primero actualizar la gema selenium-webdriver en su gemfile cambiando la línea de la gema por `gem 'selenium-webdriver', ' >= 4.11.0'` y ejecutar `bundler install`.

¹[Documentación Capibara](#) - [Documentación Cypress-on-Rails](#) - [Documentación Cypress](#)

Si lo anterior no funciona para arreglar el error deberán revisar la versión de su navegador y instalar la versión correspondiente del webdriver manualmente. Por ejemplo para google chrome toda esta información se encuentra en [esta pagina de sus ChromeDriver](#). Otra alternativa es llevar la versión de su navegador a la que señala el webdriver que instalo selenium esto implica instalar la versión correspondiente del navegador señalada en consola al ejecutar los tests.

Vocabulario

Como recordatorio de las diferencias entre un happy path y un camino alternativo:

- **Test/Caso happy path:** Es cuando se realiza una prueba en un contexto en donde lo que se busca hacer es exitoso. Por ejemplo se instancia un modelo valido o se manda un request valido. En estos casos generalmente lo que se espera es que la acción se realice correctamente y la pagina reaccione sin lanzar alertas o errores por lo tanto los tests fallan cuando lo anterior no se cumple.
- **Test/Caso de camino alternativo:** Es cuando se realiza una prueba en un contexto en donde lo que se busca hacer no se logra. Por ejemplo tratamos de instanciar un modelo invalido, cometemos un error en el request o nos falta autorización. En estos casos generalmente lo que se espera es que la acción que se quería realizar no se realice y que la pagine genere algún tipo de mensaje o alerta. Por lo tanto los tests fallan solo si la acción se realiza exitosamente y/o no se genera ninguna alerta o error.

Entrega

La entrega de esta tarea se realizara en github para esto cuando estén listo en el mismo repositorioprivado de la tarea 3 deberán crear la rama **T4** desde main la cual no deberá ser modificada posterior a la fecha de entrega. Se controlara el ultimo commit de la rama T4 para definir entregas atarazadas.

Criterio de Corrección

A continuación, se presenta la distribución de puntaje que se usará para evaluar esta tarea.

- Mejorar el sistema de reservas [1 pts].
- Tests de sistema para la navegacion [2 pts].
- Test de sistema para formularios [2 pts].
- Actualizar el set de pruebas dado las modificaciones realizadas [0.5 pts]
- Mantener 100% de coverage [0.5 pts].

Descuentos

Los revisores podrán aplicar descuentos a las notas de sus entregas:

- Hasta 5 décimas a criterio del revisor si realizan algo que dificulte la corrección de su entrega y obligue al revisor a demorarse más tiempo de lo necesario en corregir. Por ejemplo: No especificar algo importante en el **README** sobre su código, modificar las variables de entorno para conectarse a la base de datos o casos similares.
- Descuento de hasta un punto si se descubre que alguna de las funcionalidades originales de la aplicación ya no funciona en una de sus entregas.

Consideraciones Generales

- Ustedes deberán seguir usando el mismo repositorio de la tarea 3.
- Para cada entrega deberán actualizar el README que se encuentra dentro del proyecto. En este archivo tendrán que señalar lo que lograron hacer, quien hizo cada cosa e incluir cualquier información de su aplicación que pueda ser útil o necesaria para el corrector por ejemplo de que forma invocar los tests o cual de las herramientas usaron.

Reportar problemas en el equipo

En el caso de que algún integrante no aportara como fue esperado en la tarea, podrán reportarlo enviando un correo con asunto **Problema Equipo {NumeroGrupo} Testing** a juanandresarriagada@uc.cl explicando en detalle lo ocurrido. Posterior a eso se revisara el caso en detalle con los involucrados y se analizara si corresponde aplicar algún descuento. Instamos a todas las parejas que mantengan una buena comunicación y sean responsables con el resto de su equipo para evitar problemas de este estilo.

Advertencia: Si algún integrante del grupo no aporta en las tareas puede implicar recibir nota mínima en esa entrega.

Restricciones y alcances

- En caso de dudas con respecto al enunciado, código base o conceptual deberán realizarlas en alguno de los foros disponibles en canvas.
- El uso de gemas adicionales deberá ser consultado previamente en el foro a excepción de la gema FactoryBot que si puede ser incluida.

Entrega

Código: Todo el código de su proyecto al terminar la entrega deberá encontrarse en una rama en su repositorio llamada **T4**. Dicha rama se creará de su rama principal. **En caso de querer entregar con atraso deberán crear la rama correspondiente a la entrega solamente cuando estén listos ya que en caso contrario se corregirá el ultimo commit que se vea al momento de realizar la corrección.**

Atraso:

Los grupos que todavía mantengan su cupón de atraso para las tareas pueden utilizarlo y este entregara 2 días adicionales para entregar sin descuento. Para hacer uso del cupón lo que deben hacer es cuando estén listos crear la rama T4 posterior a la fecha de entrega original.

Integridad académica

Este curso se adscribe al Código de Honor establecido por la Escuela de Ingeniería. Todo trabajo evaluado en este curso debe ser hecho **individualmente** o en **los grupos asignados** según sea definido en la evaluación y **sin apoyo de terceros**. Se espera que los alumnos mantengan altos estándares de honestidad académica, acorde al Código de Honor de la Universidad. Cualquier acto deshonesto o fraude académico está prohibido; los alumnos que incurran en este tipo de acciones se exponen a un Procedimiento Sumario. Es responsabilidad de cada alumno conocer y respetar el documento sobre Integridad Académica publicado por la Dirección de Pregrado de la Escuela de Ingeniería.