MAYO 2022

TESTING EXPLORATORIO

SPRINT I

INTEGRANTES

Botinelli, Rocio
Martinelli, Guillermo
Paz, Federico
Peñuela, Oscar Armando
Vanetta, Agustin
Vuoso, Lucas

DIGITALHOUSE

Testing Exploratorio - DIGITAL booking

Sprint I

Grupo 5

Introducción:

El siguiente informe tiene el objetivo de documentar pruebas exploratorias que se realizaran al proyecto “Digital Booking”, cabe destacar que este informe se complementa con el diseño de pruebas basadas en las historias de usuario y con los test realizados en Postman a la API desarrollada en Java.

Si bien el equipo durante este primer sprint desarrolló casos de prueba para testear la funcionalidad y calidad de la aplicación que está en desarrollo, los casos de prueba que se realizaran a partir de este informe tienen como principal funcionalidad alejarse un poco de las historias de usuario, ya que estas sirvieron como base para armar los demás casos de prueba (ver planilla Excel “Spring 1 - Testing”), y realizar pruebas en base a la experiencia y “intuición” por parte de quien las realice, en este caso quien ocupa el rol de tester en este primer sprint.

La ventaja que nos da este tipo de pruebas es “despegarse” un poco de los requerimientos del cliente/usuario y realizar distintas pruebas al software para asegurar la calidad y funcionalidad, ya que, si nos atamos estrictamente a las historias de usuario, podríamos dejar muchas funcionalidades de lado, y de esta manera perdiéndonos la posibilidad de encontrar distintos tipos de bugs/errores en la aplicación, que terminan repercutiendo en el producto final.

Por último, ya sea a través de testing exploratorio, utilización de casos de prueba basados en las historias de usuario, o el mismo testing a través de Postman, tienen como principal funcionalidad asegurar un producto de calidad, es decir, un producto funcional, que cumpla los requerimientos del cliente y que resuelva una situación/problema especifico que tenga el usuario, y por lo cual utiliza nuestro producto.

Ambientes de prueba

Los distintos escenarios a testear serán en el ambiente del front y back end. En lo que respecta al back se realizaran pruebas sobre el funcionamiento integral de la aplicación, la cual fue estructurada bajo el patrón MVC y vinculada con una base de datos relacional, en este caso MySQL.

Lo que respecta al Front se realizaran pruebas de integración para verificar el funcionamiento del software desarrollado con todos sus componentes, ya que en los test desarrollados anteriormente (en base a las historias de usuario) los test se realizaron principalmente en los componentes, mayormente, de manera aislada de los demás componentes, debido a que aun estaban en desarrollo o por desarrollar.

Tanto para el ambiente de Back como para el ambiente de Front se deberán tener en cuenta las limitaciones que establece la etapa temprana del proyecto, es decir, no se podrá ir más allá de lo que los requerimientos establecieron para este sprint.

Desarrollo:

Backend:

Como requisito para ejecutar localmente la aplicación se requiere Java 17 ya que con diferentes versiones (ej. 16). A continuación, se detalla cómo solucionar este inconveniente: Se debe Cambiar el 'Nivel de idioma' (Configuración del proyecto / Proyecto en la Configuración del módulo disponible desde la ventana de contexto - clic derecho - en la ventana del proyecto) para que coincida con el SDK actual (ignorando posibles conflictos de idioma con versiones más recientes)

Descargue un JDK de Java 17 para realizar el análisis. Eso está en Configuración de plataforma/SDK en el mismo menú contextual, seleccione '+', seleccione 'Descargar JDK', luego descargue e instale una versión de Java 17.

Para este proyecto se descargo corretto-17 ***/ Amazon Corretto versión 17.0.3.*** Project Settings > Project > Proyect SDK & Project lenguaje Nivel: 17 - ().

Planilla de casos de prueba:

#	Titulo	Descripción	Resultado esperado	Resultado obtenido	PASA
1	Borrar categoría existente	Se borra una categoría existente para validar el mensaje en consola	"Categoría eliminada"	"Categoría eliminada"	PASA
2	Borrar categoría inexistente	Se borra una categoría inexistente para validar el mensaje en consola	Error / "Internal Server Error",	"No class Digitalbooking.accommodations.entities.Categoria entity with id 9 exists!"	PASA
3	Ingresar una categoría funcional	Se crea una categoría sin ID, con CategoriaDTO	Un mensaje que confirme la creación del mismo	Se crea el objeto, status 200. No hay mensaje personalizado.	PASA (Se puede mejorar response)
4	Ingresar una categoría con datos incorrectos	Se ingresa una categoría con datos incorrectos en el json.	No debería permitir ingresar una categoría vacía	Crea una categoría vacía, con datos null bajo el id 14.	NO PASA
5	Actualizar una categoría con vacíos	Se actualiza una categoría con datos incorrectos en el json.	No debería permitir actualizar una categoría vacía	Actualiza una categoría vacía, con datos null bajo el id 12.	NO PASA
6	Listar todos	Se utiliza controller: categorías/list (GET)	Traer todos los objetos creados	Status 200 con objetos activos	PASA
7	Eliminar categoría con datos null	Bajo método delate, borramos la categoría con datos null	Status 200; Categoría eliminada	Status 200; Categoría eliminada	PASA

Conclusión backend:

Se realizaron 7 casos de prueba, de los cuales 2 arrojaron un resultado no esperado. Como resultado podemos ver que se detectó la falta de algunas validaciones para evitar que se pueda cometer un error involuntario, como crear una categoría sin datos o actualizar alguna categoría sin datos.

Front end:

Se ejecuta npm-start para correr el proyecto en IDE, el mismo debe traer todas las dependencias necesarias para correr el proyecto. Se utiliza el modelo previsto en figma por PO al cual se puede acceder desde el siguiente link: https://www.figma.com/file/8ky8q2HZ8k2yvrWr3ANDCH/PI---grupo-5?node-id=0%3A1 . Los distintos componentes deben renderizarse de manera legible en todos los dispositivos: Desktop, Tablet y mobile. Todos los componentes deben tener presentes iconos, estilos, fuentes, colores, logos, etc. Que se muestran en el modelo de referencia, ya que aunque no tengan funcionalidad, podrían tenerla en algún otro spring, más allá de esto, según los requerimientos que se pactaron Ana Clara Garcia (Scrum Master) en la daily del dia 25 de mayo 2022, se indico que debe seguirse el modelo “al pie de la letra” y lograr el resultado visual esperado por el PO.

Planilla de casos de prueba:

#	Titulo	Descripción	Resultado esperado	Resultado obtenido	PASA
1	Recorrido general en mobile	Se realiza visualizacion de login, registro e homepage en mobile. 360x740	Modelo responsive, legible	En mobile el 50% de la página lo ocupa el header y el bloque de busqueda. Por lo que el espacio para visualizar hoteles no es el ideal. Bloque categorias queda por debajo del bloque listado. Error en el menu hamburg.	NO PASA
2	Recorrido general en tablet	Se realiza visualizacion de login, registro e homepage en tablet. IpadMini	Modelo responsive, legible	Modelo responsive, legible	PASA
3	Recorrido general en desktop	Se realiza visualización de login, registro e homepage en tablet. Desktop	Modelo responsive, legible	Modelo responsive, legible	PASA (Se puede mejorar response)
4	Iconos del footer	Los mismos no deben tener funcionalidad	Los mismos no deben tener funcionalidad	Redireccionan a http://localhost:3000/*, la cual no tiene contenido.	NO PASA
5	Bloque buscador - Calendar	Se seleccionan varias fechas, para ver cómo reacciona el calendario	El calendario debe moverse entre las fechas seleccionadas	Muestra las fechas que se seleccionan sin importar mes, año o día.	PASA
6	Bloque recomendaciones	Se compara el archivo con figma	Debe coincidir con el modelo de referencia	En gran parte coincide con el modelo de referencia, pero no tiene desarrollado algunos iconos, que para este sprint no tienen funcionalidad, pero al estar maquetados, debería haberse tenido en cuenta. Ej.: Estrellas, distancia del centro, mapa, wifi, pileta.	NO PASA

7	Click en cerrar sesión	Con sesión iniciada se cierra la misma para verificar renderización del header	Deben aparecer botones para iniciar sesión	Renderiza ok	PASA

Conclusión frontend:

Se realizaron 7 casos de prueba, de los cuales 3 arrojaron un resultado no esperado. Los errores tienen que ver con cuestiones visuales, que pueden afectar la experiencia del usuario o no brindarle datos que podrían ser relevantes a la hora de utilizar nuestra aplicación (como el dato de si el hotel tiene o no wifi o pileta). En cuanto a la funcionalidad los links o rutas para ir a las distintas páginas de manera correcta, al igual que el calendar y la lista de ciudades/destinos.