**INFORME DE PRUEBAS**

**Versión 1.0**

**Lima, abril 2023**

**Historial de revisiones**

| **Fecha** | **Revisión** | **Descripción** | **Autor** |
| --- | --- | --- | --- |
| 23/05/2023 | 1.0 | Primera versión del informe de pruebas 2 | Consultora HomeSkill S.A. |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**ÍNDICE**

[**1. Introducción 4**](#_u1x02jbedr9q)

[1.1. Propósito del documento 4](#_87oc0xu4el2)

[1.2. Alcance del informe 4](#_n7to5wnr1ioo)

[**2. Plan de pruebas 4**](#_av1xjd5zffq7)

[2.2. Recursos necesarios 7](#_apnpj32s3nbj)

[2.3. Estrategias de pruebas 8](#_rmhlonjetqe0)

[**3. Casos de prueba 8**](#_4afv3rviibo2)

[3.1. Escenario de prueba 8](#_b9hzxj9ph1rm)

[3.1.1. Prueba de requerimiento 04 8](#_9dkp42jfru23)

[3.1.2. Prueba de requerimiento 05 9](#_msoe6z5ua9o9)

[**4. Resultados de prueba 10**](#_ypg2dcjl42bm)

[**5. Conclusiones 10**](#_y7m49kbuc754)

# **1. Introducción**

## **1.1. Propósito del documento**

El propósito principal es proporcionar una descripción detallada de cómo se probó el software, qué casos de prueba se ejecutaron, qué resultados se obtuvieron, qué incidencias se encontraron, y cómo se corrigieron, las áreas que necesitan mejorar y las recomendaciones para futuras pruebas.

## 

## **1.2. Alcance del informe**

El informe de pruebas proporciona una evaluación completa y detallada del software del proyecto GoShop en términos de su calidad y capacidad para satisfacer los requisitos del usuario. El informe incluye información sobre los requisitos funcionales que se han evaluado, y cualquier problema que se haya encontrado durante las pruebas y cómo se han abordado.

# 

# **2. Plan de pruebas**

**2.1. Tipos de pruebas**

**Tabla 1**

*Prueba de Requerimiento 4*

| **Vista de perfil de usuario** | |
| --- | --- |
| **Resumen:** Una vista de perfil de usuario que permita a los usuarios ver y editar su información personal y configuraciones de cuenta | |
| **Pasos:**   1. Se crea una instancia del objeto Client para simular solicitudes HTTP. 2. Se crea un usuario de prueba utilizando el modelo User de Django, con un nombre de usuario, contraseña y correo electrónico. 3. Se crea un perfil de usuario asociado al usuario de prueba utilizando el modelo UserProfile. 4. En el método test\_user\_profile\_view(), se realiza el inicio de sesión del usuario de prueba utilizando el método login() del cliente. 5. Se realiza una solicitud GET a la vista "profile" utilizando el método get() del cliente y la función reverse() para obtener la URL correspondiente a la vista. 6. Se verifica que el código de estado de la respuesta sea 200, lo que indica que la vista se ha cargado correctamente. 7. En el método test\_user\_config\_view(), se realiza nuevamente el inicio de sesión del usuario de prueba. 8. Se realiza una solicitud GET a la vista "config" utilizando el método get() del cliente y la función reverse() para obtener la URL correspondiente a la vista. 9. Se verifica que el código de estado de la respuesta sea 200, lo que indica que la vista se ha cargado correctamente. | |
| **Resultados Esperados:**  **Método ‘test\_user\_profile\_view’**   * La vista "profile" se cargue correctamente y que la respuesta HTTP tenga un código de estado 200.   **Método ‘test\_user\_config\_view’**   * El resultado esperado de la prueba sería que el código de estado de la respuesta (response.status\_code) sea igual a 200. | |

*Nota*. Elaboración Propia.

**Tabla 2**

*Prueba de Requerimiento 5*

| **Método de pago** | |
| --- | --- |
| **Resumen:** Visualizar sus tarjetas registradas, y por otro, agregar más tarjetas para realizar sus compras en la tienda virtual. | |
| **Pasos:**   1. Se crea una instancia del objeto Client para simular solicitudes HTTP. 2. Se crea un usuario de prueba utilizando el modelo User de Django, con un nombre de usuario, contraseña y correo electrónico. 3. Se crea un perfil de usuario asociado al usuario de prueba utilizando el modelo UserProfile. 4. En el método test\_user\_payment\_view(), se realiza el inicio de sesión del usuario de prueba utilizando el método login() del cliente. 5. Se realiza una solicitud GET a la vista "payment" utilizando el método get() del cliente y la función reverse() para obtener la URL correspondiente a la vista. 6. Se verifica que el código de estado de la respuesta sea 200, lo que indica que la vista se ha cargado correctamente. 7. En el método test\_add\_payment\_method(), se realiza nuevamente el inicio de sesión del usuario de prueba. 8. Se realiza una solicitud POST a la vista "add\_payment\_method" utilizando el método post() del cliente y la función reverse() para obtener la DATA correspondiente a la tarjeta. 9. Se verifica que el código de estado de la respuesta sea 200, lo que indica que la vista se ha cargado correctamente. 10. Se verifica que la data de la tarjeta exista usando la función FILTER al objeto owner\_id, number; y la función EXISTS para retornar si existe la tarjeta. 11. En el método test\_remember\_payment\_method(), se realiza nuevamente el inicio de sesión del usuario de prueba. 12. Se realiza una solicitud POST al "remember\_payment\_method" y "'remember-card" utilizando el método post() del cliente y la función reverse() para obtener la info de la tarjeta. 13. Se verifica que el código de estado de la respuesta sea 200, lo que indica que la vista se ha cargado correctamente. 14. Se actualizan los valores del objeto CARD desde la base de datos. 15. Se verifica si la tarjeta se guarda según la respuesta del usuario. | |
| **Resultados Esperados:**  **Método ‘test\_user\_payment\_view’**   * La vista "payment" se cargue correctamente y que la respuesta HTTP tenga un código de estado 200.   **Método ‘test\_add\_payment\_method’**   * El método "payment" se cargue correctamente y que la respuesta HTTP tenga un código de estado 200. * Se verifica que exista la tarjeta según los datos ingresados y se agrega a las lista de métodos de pago.   **Método ‘test\_remember\_payment\_method’**   * El método "payment" se cargue correctamente y que la respuesta HTTP tenga un código de estado 200. * Se actualizan los valores de la tarjeta desde la base de datos. * Según la respuesta del cliente se guarda o no la credenciales de la tarjeta. | |

*Nota*. Elaboración Propia.

## **2.2. Recursos necesarios**

* **Hardware necesario**

Se necesitará un equipo con suficiente capacidad de procesamiento y memoria RAM para ejecutar el servidor de desarrollo de Django, el navegador web y cualquier otra aplicación necesaria para realizar las pruebas.

* **Software necesario**

Se necesitará lo siguiente:

* Python 3.8 y las dependencias necesarias para ejecutar el proyecto de Django.
* Un entorno virtual para el proyecto de Django (Visual Studio).
* Un servidor de desarrollo de Django para ejecutar el proyecto y las pruebas.
* Un navegador web para realizar las pruebas.
* **Base de datos**

Se necesitará una base de datos para el proyecto de Django. Si se usa SQLite para el proyecto, no se necesitará configurar nada adicional, pero si se usa otro motor de base de datos, se necesitará configurarlo y asegurarse de que esté en funcionamiento.

* **Recursos adicionales**: Para las pruebas específicas del proyecto de Django mencionadas anteriormente, se necesitará lo siguiente:
* Vista de descripción de productos: se necesitará al menos un objeto de categoría de producto y un objeto de producto para realizar las pruebas. Se pueden crear estos objetos en la función setUp del TestCase.
* Vista de registro de usuario: no se necesitan recursos adicionales, ya que se puede utilizar la función de registro de Django para crear nuevos usuarios.
* Vista de búsqueda y visualización de productos: se necesitará al menos un objeto de categoría de producto y varios objetos de producto para realizar las pruebas. Se pueden crear estos objetos en la función setUp del TestCase.

## **2.3. Estrategias de pruebas**

Las estratégias de prueba que hemos usado en estos dos test, cuáles son el registro de usuario, visualización de productos y descripción de productos, son de tipo unitaria y de integración, las prueba unitarias son pruebas a nivel código para verificar su comportamiento y validarlo, mientras que las de integración son para evaluar la interacción entre diferentes requisitos y cómo funcionan en conjunto.

* Estrategia de prueba de requerimiento 04

Las pruebas de unidad se centrará en verificar los campos de registro, como el nombre de usuario, la contraseña y el correo electrónico. Se utilizarán aserciones y métodos de comparación para asegurarse de que los datos ingresados por el usuario se almacenarán correctamente en la base de datos. Además de las solicitudes HTTP utilizando el cliente de prueba de Django y verificará que, después del registro exitoso, el usuario sea redirigido correctamente a la página esperada.

* Estrategia de prueba de requerimiento 05

# **3. Casos de prueba**

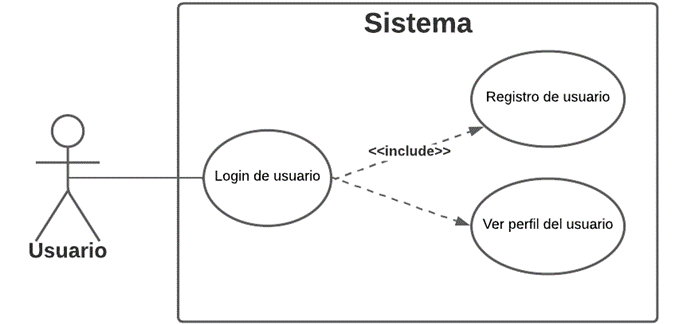
## **3.1. Escenario de prueba**

### ***3.1.1. Prueba de requerimiento 04***

El propósito del caso de prueba es verificar el correcto funcionamiento de la vista de registro de usuario en una aplicación web, cuando se reciben solicitudes GET y POST, se redirecciona al usuario a la página correcta y se crea el usuario en la base de datos sin problemas. Para lograrlo, se configuran los elementos necesarios, se realizan solicitudes GET y POST, se comprueba la respuesta HTTP de la vista del perfil de usuario, se verifica que la redirección se efectúe adecuadamente y se confirma que el usuario se haya creado sin errores en la base de datos mediante la ejecución de diversas afirmaciones y validaciones.

**Figura 1**

*Diagrama de casos de uso 04.*



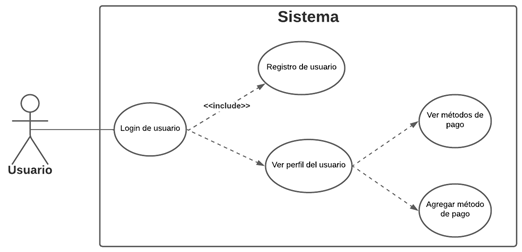
*Nota*. Elaboración propia

### ***3.1.2. Prueba de requerimiento 05***

El propósito del caso de prueba es verificar el correcto funcionamiento de la vista de pago de usuario en una aplicación web, cuando se reciben solicitudes GET y POST, se redirecciona al usuario a la página correcta y se crea el usuario en la base de datos sin problemas. Para lograrlo, se configuran los elementos necesarios, se realizan solicitudes GET y POST, se comprueba la respuesta HTTP de la vista del perfil de usuario, se verifica que la redirección se efectúe adecuadamente y se confirma que el usuario se haya creado sin errores en la base de datos mediante la ejecución de diversas afirmaciones y validaciones. Además, se realizan solicitudes REVERSE y EXISTS, para retornar la lista o base de datos a la cual se llama y comprobar que existe la data en la base de datos.

**Figura 2**

*Diagrama de casos de uso 05.*



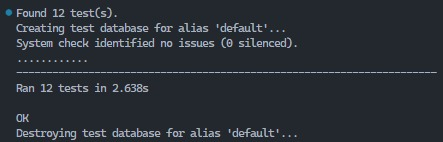
# **4. Resultados de prueba**

Se efectuaron 6 pruebas utilizando el comando "python3 manage.py test -v 1", en donde con ayuda del TestCase de python se identificaron las secciones a testear, en donde el requerimiento 4 contenía 3 pruebas con respecto a la vista de perfil de usuario y en el requerimiento 5 se encuentra 3 pruebas relacionadas al método de pago.

Los resultados obtenidos indican que todas las pruebas fueron exitosas, lo que sugiere que las funcionalidades Vista de perfil de usuario y Método de pago funcionan correctamente. Se creó una base de datos de prueba para ejecutar las pruebas, y no se identificaron problemas en el sistema durante la ejecución de las mismas, por último señalar que el tiempo total de ejecución de las pruebas fue de 2.638s.

**Figura 3**

*Resultado de prueba*



Tener en cuenta que los tests se realizaron en conjunto con las pruebas anteriores, ya que cada vez que se incremente el programa se deberá ir realizando más pruebas para poder verificar si todo funciona correctamente.

# **5. Conclusiones**

* La vista de perfil de usuario es fundamental para proporcionar a los usuarios una forma intuitiva y atractiva de ver y gestionar su información personal y configuraciones de cuenta. Esta vista debe mostrar información básica del usuario, permitir ediciones, ofrecer acceso a configuraciones de cuenta, mostrar un historial de actividades. Se puede extraer de estas pruebas es que se está verificando el correcto funcionamiento de las vistas de perfil y configuración. Al obtener una respuesta con un código de estado 200 (OK), se comprueba que las vistas se cargan correctamente y no se producen errores importantes durante el proceso.
* La vista de pagos de los clientes es fundamental para proporcionar a los usuarios de manera interactiva e intuitiva, ver sus métodos de pago ya sean agregar métodos o recordar métodos de pago. Esta vista te permite agregar el método de pago del clientes tanto un recordatorio si desea guardar el método de pago para mayor flexibilidad. Se verifica la existencia de la tarjeta en la base de datos y su respectiva actualización. Al obtener una respuesta con un código de estado 200 (OK), se comprueba que las vistas se cargan correctamente y no se producen errores importantes durante el proceso.