Una caricatura de una persona

Descripción generada automáticamente con confianza media

**Universidad Nacional Mayor de San Marcos**

*(Universidad del Perú, DECANA DE AMÉRICA)*

**Facultad de Ingeniería de Sistemas e Informática**

Escuela Profesional de Ingeniería de Software

**Reporte Pruebas Unitarias**

**2022**

**Asignatura**: Gestión de Configuración y Mantenimiento

**Docente**: Lenis Rossi Wong Portillo

**Ciclo**: 2022-1

|  |  |
| --- | --- |
| **GRUPO 2** | |
| Integrantes | Código |
| Elton John Rodriguez Callata | 17200298 |
| Cesar Augusto Pino Benavente | 17200116 |
| Jhonatan David Vásquez Enrique | 17200306 |
| Jorge Luis Rupay Huamaní | 17200302 |
| Frank Pizarro Sánchez | 17200120 |
| Rafael Alonso Laos Carrasco | 20200143 |

**Método:**

El presente documento es un reporte de las pruebas unitarias realizadas durante el desarrollo del sistema SGAPC, el cual busca describir la secuencia de pasos, datos de entrada, salida esperada, resultados y observaciones obtenidas en el momento que se evaluaron los casos de prueba en cada prueba unitaria para cada módulo evaluado del proyecto para la implementación del sistema SGAPC desarrollado para CompuPeruSAC por la consultora de software G2-Consultores.

Módulos probados:

1. Autenticación (AuthController)
2. Usuario (UserController)
3. Producto (ProductController)

Se sigue la siguiente nomenclatura para cada caso de prueba:

**PU-abb**

**a: Identificador numérico incremental por cada módulo probado**

**bb: Identificador numérico incremental por cada prueba unitaria**

**Reporte de casos de prueba:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Prueba unitaria:** | testAutenticacionTest | **Módulo probado:** | AuthController |
| **Responsable:** | Rafael Laos Carrasco | **Fecha:** | 29/06/2022 |
| **Propósito:** | Probar módulo de autenticación para el caso en que halla errores en algún campo al intentar ingresar al sistema. | | |
| **Precondiciones:** | Ninguna | **Caso de prueba:** | PU101 |
| **Secuencia del test** | * Se realiza llamada al test cuando se acceda a la ruta asignada a la función AuthController. * Evalúa si existen condiciones de error en la entrada de datos. * Verifica la existencia y veracidad de las credenciales ingresadas. * Retorna una respuesta y el estado de la solicitud. * Verifica que el retorno obtenido sea igual al esperado. | | |
| **Descripción de pasos realizados por usuario** | * Ingresa a la dirección de la plataforma * Visualiza interfaz de login * Ingresa datos en campo usuario * Ingresa datos en campo contraseña * Da clic en botón “Login” para acceder | | |
| **Entrada:** | {  loginDTO:{  usernameoremail: admin  pasword: 12345  }  result: (result.hasErrors()=True)  } | | |
| **Salida Esperada:** | ResponseEntity{  response: (“El campo”+campo+” '”+mensaje\_error)  HttpStatus: HttpStatus.BAD\_REQUEST  } | | |
| **Resultado de la prueba:** | Exitosa | | |
| **Errores obtenidos:** |  | | |
| **Retroalimentación y observaciones:** | El módulo probado funciona adecuadamente para el caso descrito. | | |
| **Prueba unitaria:** | testAutenticacionTest | **Módulo probado:** | AuthController |
| **Responsable:** | Rafael Laos Carrasco | **Fecha:** | 29/06/2022 |
| **Propósito:** | Probar módulo de autenticación para el caso en que no halla errores en algún campo al intentar ingresar al sistema y las credenciales ingresadas sean válidas. | | |
| **Precondiciones:** | Ninguna | **Caso de prueba:** | PU102 |
| **Secuencia del test** | * Se realiza llamada al test cuando se acceda a la ruta asignada a la función AuthController. * Evalúa si existen condiciones de error en la entrada de datos. * Verifica la existencia y veracidad de las credenciales ingresadas. * Retorna una respuesta y el estado de la solicitud. * Verifica que el retorno obtenido sea igual al esperado. | | |
| **Descripción de pasos realizados por usuario** | * Ingresa a la dirección de la plataforma * Visualiza interfaz de login * Ingresa datos en campo usuario * Ingresa datos en campo contraseña * Da clic en botón “Login” para acceder | | |
| **Entrada:** | {  loginDTO:{  usernameoremail: admin  pasword: 12345  }  result: (result.hasErrors()=False)  } | | |
| **Salida Esperada:** | ResponseEntity{  response: (“Ha iniciado sesion con exito”, ”admin”,  authentication.getAuthorities(), new  JwtAuthResponseDTO(token))  HttpStatus: HttpStatus.OK  } | | |
| **Resultado de la prueba:** | Exitosa | | |
| **Errores obtenidos:** |  | | |
| **Retroalimentación y observaciones:** | El módulo probado funciona adecuadamente para el caso descrito. | | |
| **Prueba unitaria:** | testAutenticacionTest | **Módulo probado:** | AuthController |
| **Responsable:** | Rafael Laos Carrasco | **Fecha:** | 29/06/2022 |
| **Propósito:** | Probar módulo de autenticación para el caso en que no halla errores en algún campo al intentar ingresar al sistema y las credenciales ingresadas no sean válidas. | | |
| **Precondiciones:** | Ninguna | **Caso de prueba:** | PU103 |
| **Secuencia del test** | * Se realiza llamada al test cuando se acceda a la ruta asignada a la función AuthController. * Evalúa si existen condiciones de error en la entrada de datos. * Verifica la existencia y veracidad de las credenciales ingresadas. * Retorna una respuesta y el estado de la solicitud. * Verifica que el retorno obtenido sea igual al esperado. | | |
| **Descripción de pasos realizados por usuario** | * Ingresa a la dirección de la plataforma * Visualiza interfaz de login * Ingresa datos en campo usuario * Ingresa datos en campo contraseña * Da clic en botón “Login” para acceder | | |
| **Entrada:** | {  loginDTO:{  usernameoremail: admin  pasword: 123456  }  result: (result.hasErrors()=False)  } | | |
| **Salida Esperada:** | ResponseEntity{  response: (“Error en las credenciales de acceso”,  e.getLocalizedMessage)  HttpStatus: HttpStatus.UNAUTHORIZED  } | | |
| **Resultado de la prueba:** | Exitosa | | |
| **Errores obtenidos:** |  | | |
| **Retroalimentación y observaciones:** | El módulo probado funciona adecuadamente para el caso descrito. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Prueba unitaria:** | testGetAllUserTest | **Módulo probado:** | UserController |
| **Responsable:** | Rafael Laos Carrasco | **Fecha:** | 18/07/2022 |
| **Propósito:** | Probar la funcionalidad de realizar una consulta completa a la base de datos dentro del módulo de usuario para el caso en que no hallan errores internos del servidor al realizar esta consulta. | | |
| **Precondiciones:** | Admin Access | **Caso de prueba:** | PU201 |
| **Secuencia del test** | * Se realiza llamada al test cuando se acceda a la ruta asignada a la función getAll dentro de UserController. * Trata de consultar los parámetros entrantes en la base de datos. * Verifica el estado de la consulta. * Retorna una respuesta y el estado de la solicitud. * Verifica que el retorno obtenido sea igual al esperado. | | |
| **Descripción de pasos realizados por usuario** | Funcionalidad aún no implementada. | | |
| **Entrada:** | {  RequestParam:{  int numberOfPage: 0  int sizeOfPage: 10  String sortPeer: “id”  String sortDir: “asc”  }  Exception e: False  } | | |
| **Salida Esperada:** | ResponseEntity{  response{  “data”: userService.getAll(0,10,”id”,”asc”)  }  HttpStatus: HttpStatus.OK  } | | |
| **Resultado de la prueba:** | Exitosa | | |
| **Errores obtenidos:** |  | | |
| **Retroalimentación y observaciones:** | El módulo probado funciona adecuadamente para el caso descrito. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Prueba unitaria:** | testGetAllUserTest | **Módulo probado:** | UserController |
| **Responsable:** | Rafael Laos Carrasco | **Fecha:** | 18/07/2022 |
| **Propósito:** | Probar la funcionalidad de realizar una consulta completa a la base de datos dentro del módulo de usuario para el caso en que se presenten errores internos del servidor al intentar realizar esta consulta. | | |
| **Precondiciones:** | Admin Access | **Caso de prueba:** | PU202 |
| **Secuencia del test** | * Se realiza llamada al test cuando se acceda a la ruta asignada a la función getAll dentro de UserController. * Trata de consultar los parámetros entrantes en la base de datos. * Verifica el estado de la consulta. * Retorna una respuesta y el estado de la solicitud. * Verifica que el retorno obtenido sea igual al esperado. | | |
| **Descripción de pasos realizados por usuario** | Funcionalidad aún no implementada. | | |
| **Entrada:** | {  RequestParam:{  int numberOfPage: 0  int sizeOfPage: 10  String sortPeer: “id”  String sortDir: “asc”  }  Exception e: True  } | | |
| **Salida Esperada:** | ResponseEntity{  response{  “error”: “Error al realizar la consulta a la base de  datos”  }  HttpStatus: HttpStatus.INTERNAL\_SERVER\_ERROR  } | | |
| **Resultado de la prueba:** | Exitosa | | |
| **Errores obtenidos:** |  | | |
| **Retroalimentación y observaciones:** | El módulo probado funciona adecuadamente para el caso descrito. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Prueba unitaria:** | testShowUserTest | **Módulo probado:** | UserController |
| **Responsable:** | Rafael Laos Carrasco | **Fecha:** | 18/07/2022 |
| **Propósito:** | Probar la funcionalidad de mostrar los datos de un usuario registrado buscándolo por su id en la base de datos dentro del módulo de usuario para el caso en que se presenten errores internos del servidor al intentar realizar esta consulta. | | |
| **Precondiciones:** | Admin Access | **Caso de prueba:** | PU211 |
| **Secuencia del test** | * Se realiza llamada al test cuando se acceda a la ruta asignada a la función show dentro de UserController. * Trata de consultar el usuario buscado por su id en la base de datos. * Verifica el estado de la consulta. * Verifica la existencia del id ingresado en la base de datos. * Retorna una respuesta y el estado de la solicitud. * Verifica que el retorno obtenido sea igual al esperado. | | |
| **Descripción de pasos realizados por usuario** | Funcionalidad aún no implementada. | | |
| **Entrada:** | {  RequestParam:{  Long id: 77777722  }  Exception e: True  } | | |
| **Salida Esperada:** | ResponseEntity{  response{  “message”: “Error al realizar la consulta a la base de  datos”  “error”: e.getMessage()+e.getMostEspecificCause()  }  HttpStatus: HttpStatus.INTERNAL\_SERVER\_ERROR  } | | |
| **Resultado de la prueba:** | Exitosa | | |
| **Errores obtenidos:** |  | | |
| **Retroalimentación y observaciones:** | El módulo probado funciona adecuadamente para el caso descrito. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Prueba unitaria:** | testShowUserTest | **Módulo probado:** | UserController |
| **Responsable:** | Rafael Laos Carrasco | **Fecha:** | 18/07/2022 |
| **Propósito:** | Probar la funcionalidad de mostrar los datos de un usuario registrado buscándolo por su id en la base de datos dentro del módulo de usuario para el caso en que no se presenten errores internos del servidor al intentar realizar esta consulta, pero no se encuentre el id buscado. | | |
| **Precondiciones:** | Admin Access | **Caso de prueba:** | PU212 |
| **Secuencia del test** | * Se realiza llamada al test cuando se acceda a la ruta asignada a la función show dentro de UserController. * Trata de consultar el usuario buscado por su id en la base de datos. * Verifica el estado de la consulta. * Verifica la existencia del id ingresado en la base de datos. * Retorna una respuesta y el estado de la solicitud. * Verifica que el retorno obtenido sea igual al esperado. | | |
| **Descripción de pasos realizados por usuario** | Funcionalidad aún no implementada. | | |
| **Entrada:** | {  RequestParam:{  Long id: 77777723  }  Exception e: False  } | | |
| **Salida Esperada:** | ResponseEntity{  response{  “message”: “El usuario con ID: 77777723 no existe  en la base de datos”  }  HttpStatus: HttpStatus.NOT\_FOUND  } | | |
| **Resultado de la prueba:** | Exitosa | | |
| **Errores obtenidos:** |  | | |
| **Retroalimentación y observaciones:** | El módulo probado funciona adecuadamente para el caso descrito. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Prueba unitaria:** | testShowUserTest | **Módulo probado:** | UserController |
| **Responsable:** | Rafael Laos Carrasco | **Fecha:** | 18/07/2022 |
| **Propósito:** | Probar la funcionalidad de mostrar los datos de un usuario registrado buscándolo por su id en la base de datos dentro del módulo de usuario para el caso en que no se presenten errores internos del servidor al intentar realizar esta consulta y se encuentre el id buscado. | | |
| **Precondiciones:** | Admin Access | **Caso de prueba:** | PU213 |
| **Secuencia del test** | * Se realiza llamada al test cuando se acceda a la ruta asignada a la función show dentro de UserController. * Trata de consultar el usuario buscado por su id en la base de datos. * Verifica el estado de la consulta. * Verifica la existencia del id ingresado en la base de datos. * Retorna una respuesta y el estado de la solicitud. * Verifica que el retorno obtenido sea igual al esperado. | | |
| **Descripción de pasos realizados por usuario** | Funcionalidad aún no implementada. | | |
| **Entrada:** | {  RequestParam:{  Long id: 77777722  }  Exception e: False  } | | |
| **Salida Esperada:** | ResponseEntity{  response{  “data”: userService.findById(77777722)  }  HttpStatus: HttpStatus.OK  } | | |
| **Resultado de la prueba:** | Exitosa | | |
| **Errores obtenidos:** |  | | |
| **Retroalimentación y observaciones:** | El módulo probado funciona adecuadamente para el caso descrito. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Prueba unitaria:** | testCreateUserTest | **Módulo probado:** | UserController |
| **Responsable:** | Rafael Laos Carrasco | **Fecha:** | 18/07/2022 |
| **Propósito:** | Probar la funcionalidad de crear un nuevo usuario dentro del módulo de usuario para el caso en que algún campo de registro haya sido ingresado erróneamente. | | |
| **Precondiciones:** | Admin Access | **Caso de prueba:** | PU221 |
| **Secuencia del test** | * Se realiza llamada al test cuando se acceda a la ruta asignada a la función create dentro de UserController. * Verifica si la entrada ingresada tiene errores. * Trata de consultar los parámetros entrantes en la base de datos. * Verifica el estado de la consulta. * Crea el usuario requerido y lo asigna a la respuesta. * Retorna una respuesta y el estado de la solicitud. * Verifica que el retorno obtenido sea igual al esperado. | | |
| **Descripción de pasos realizados por usuario** | * Ingresa al sistema con acceso de administrador mediante el módulo y la interfaz integrada de login. * Selecciona menú lateral “Personal”. * Selecciona submenú lateral “Agregar Personal”. * Llena datos de registro solicitados. * Da click al botón de “CREAR USUARIO” | | |
| **Entrada:** | {  RequestBody:{  UserDTO: userDTO  Binding result: result  }  result.hasErrors(): True  Exception e: False  } | | |
| **Salida Esperada:** | ResponseEntity{  response{  “errors”: result.getFieldErrors()  }  HttpStatus: HttpStatus.BAD\_REQUEST  } | | |
| **Resultado de la prueba:** | Exitosa | | |
| **Errores obtenidos:** |  | | |
| **Retroalimentación y observaciones:** | El módulo probado funciona adecuadamente para el caso descrito. | | |
| **Prueba unitaria:** | testCreateUserTest | **Módulo probado:** | UserController |
| **Responsable:** | Rafael Laos Carrasco | **Fecha:** | 18/07/2022 |
| **Propósito:** | Probar la funcionalidad de crear un nuevo usuario dentro del módulo de usuario para el caso en que ningún campo de registro haya sido ingresado erróneamente, pero se presenten errores internos del servidor al intentar realizar esta consulta. | | |
| **Precondiciones:** | Admin Access | **Caso de prueba:** | PU222 |
| **Secuencia del test** | * Se realiza llamada al test cuando se acceda a la ruta asignada a la función create dentro de UserController. * Verifica si la entrada ingresada tiene errores. * Trata de consultar los parámetros entrantes en la base de datos. * Verifica el estado de la consulta. * Crea el usuario requerido y lo asigna a la respuesta. * Retorna una respuesta y el estado de la solicitud. * Verifica que el retorno obtenido sea igual al esperado. | | |
| **Descripción de pasos realizados por usuario** | * Ingresa al sistema con acceso de administrador mediante el módulo y la interfaz integrada de login. * Selecciona menú lateral “Personal”. * Selecciona submenú lateral “Agregar Personal”. * Llena datos de registro solicitados. * Da click al botón de “CREAR USUARIO” | | |
| **Entrada:** | {  RequestBody:{  UserDTO: userDTO  Binding result: result  }  result.hasErrors(): False  Exception e: True  } | | |
| **Salida Esperada:** | ResponseEntity{  response{  “message”: “Error al realizar la consulta a la base de datos”  “error”: e.getMessage()+e.getMostEspecificCause() }  HttpStatus: HttpStatus.INTERNAL\_SERVER\_ERROR } | | |
| **Resultado de la prueba:** | Exitosa | | |
| **Errores obtenidos:** |  | | |
| **Retroalimentación y observaciones:** | El módulo probado funciona adecuadamente para el caso descrito. | | |
| **Prueba unitaria:** | testCreateUserTest | **Módulo probado:** | UserController |
| **Responsable:** | Rafael Laos Carrasco | **Fecha:** | 18/07/2022 |
| **Propósito:** | Probar la funcionalidad de crear un nuevo usuario dentro del módulo de usuario para el caso en que ningún campo de registro haya sido ingresado erróneamente y no se presenten errores internos del servidor al intentar realizar esta consulta. | | |
| **Precondiciones:** | Admin Access | **Caso de prueba:** | PU223 |
| **Secuencia del test** | * Se realiza llamada al test cuando se acceda a la ruta asignada a la función create dentro de UserController. * Verifica si la entrada ingresada tiene errores. * Trata de consultar los parámetros entrantes en la base de datos. * Verifica el estado de la consulta. * Crea el usuario requerido y lo asigna a la respuesta. * Retorna una respuesta y el estado de la solicitud. * Verifica que el retorno obtenido sea igual al esperado. | | |
| **Descripción de pasos realizados por usuario** | * Ingresa al sistema con acceso de administrador mediante el módulo y la interfaz integrada de login. * Selecciona menú lateral “Personal”. * Selecciona submenú lateral “Agregar Personal”. * Llena datos de registro solicitados. * Da click al botón de “CREAR USUARIO” | | |
| **Entrada:** | {  RequestBody:{  UserDTO: userDTO  Binding result: result  }  result.hasErrors(): False  Exception e: False  } | | |
| **Salida Esperada:** | ResponseEntity{  response{  “message”: “El usuario ha sido creado con exito”  “data”: userService.save(userDTO) }  HttpStatus: HttpStatus.CREATED } | | |
| **Resultado de la prueba:** | Exitosa | | |
| **Errores obtenidos:** |  | | |
| **Retroalimentación y observaciones:** | El módulo probado funciona adecuadamente para el caso descrito. | | |
| **Prueba unitaria:** | testAddRoleToUserTest | **Módulo probado:** | UserController |
| **Responsable:** | Rafael Laos Carrasco | **Fecha:** | 18/07/2022 |
| **Propósito:** | Probar la funcionalidad de asignar un rol a un usuario creado dentro del módulo de usuario para el caso en que se presenten errores internos del servidor al intentar realizar esta consulta. | | |
| **Precondiciones:** | Admin Access | **Caso de prueba:** | PU231 |
| **Secuencia del test** | * Se realiza llamada al test cuando se acceda a la ruta asignada a la función addRoleToUser dentro de UserController. * Trata de consultar en la base de datos parámetros ingresados. * Verifica el estado de la consulta. * Asigna el rol al usuario según ids de ambos. * Retorna una respuesta y el estado de la solicitud. * Verifica que el retorno obtenido sea igual al esperado. | | |
| **Descripción de pasos realizados por usuario** | Funcionalidad aún no implementada. | | |
| **Entrada:** | {  RequestParam:{  Long userId: userId  Lonr roleId: rolId  }  Exception e: True  } | | |
| **Salida Esperada:** | ResponseEntity{  response{  “message”: “Error al realizar la consulta a la base  de datos”  “error”: e.getMessage()+e.getMostEspecificCause()  }  HttpStatus: HttpStatus.INTERNAL\_SERVER\_ERROR  } | | |
| **Resultado de la prueba:** | Exitosa | | |
| **Errores obtenidos:** |  | | |
| **Retroalimentación y observaciones:** | El módulo probado funciona adecuadamente para el caso descrito. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Prueba unitaria:** | testAddRoleToUserTest | **Módulo probado:** | UserController |
| **Responsable:** | Rafael Laos Carrasco | **Fecha:** | 18/07/2022 |
| **Propósito:** | Probar la funcionalidad de asignar un rol a un usuario creado dentro del módulo de usuario para el caso en que no se presenten errores internos del servidor al intentar realizar esta consulta. | | |
| **Precondiciones:** | Admin Access | **Caso de prueba:** | PU232 |
| **Secuencia del test** | * Se realiza llamada al test cuando se acceda a la ruta asignada a la función addRoleToUser dentro de UserController. * Trata de consultar en la base de datos parámetros ingresados. * Verifica el estado de la consulta. * Asigna el rol al usuario según ids de ambos. * Retorna una respuesta y el estado de la solicitud. * Verifica que el retorno obtenido sea igual al esperado. | | |
| **Descripción de pasos realizados por usuario** | Funcionalidad aún no implementada. | | |
| **Entrada:** | {  RequestParam:{  Long userId: userId  Lonr roleId: rolId  }  Exception e: False  } | | |
| **Salida Esperada:** | ResponseEntity{  response{  “message”: “El rol ha sido asignado con exito”  “data”: userService.addRoleToUser(userID, roleId)  }  HttpStatus: HttpStatus.CREATED  } | | |
| **Resultado de la prueba:** | Exitosa | | |
| **Errores obtenidos:** |  | | |
| **Retroalimentación y observaciones:** | El módulo probado funciona adecuadamente para el caso descrito. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Prueba unitaria:** | testUpdateUserTest | **Módulo probado:** | UserController |
| **Responsable:** | Rafael Laos Carrasco | **Fecha:** | 18/07/2022 |
| **Propósito:** | Probar la funcionalidad de actualizar los datos de un usuario creado dentro del módulo de usuario para el caso en que algún campo de actualización haya sido ingresado erróneamente. | | |
| **Precondiciones:** | Admin Access | **Caso de prueba:** | PU241 |
| **Secuencia del test** | * Se realiza llamada al test cuando se acceda a la ruta asignada a la función update dentro de UserController. * Verifica si existen errores en los campos de entrada. * Verifica si el id del usuario buscado existe en la base de datos. * Trata de consultar en la base de datos parámetros ingresados. * Verifica el estado de la consulta. * Edita los datos del usuario ingresados. * Retorna una respuesta y el estado de la solicitud. * Verifica que el retorno obtenido sea igual al esperado. | | |
| **Descripción de pasos realizados por usuario** | Funcionalidad aún no implementada. | | |
| **Entrada:** | {  {  UserDTO: userDTO  BindingResult: result  Long id: id  }  Exception e: False  result.hasErrors(): True  userService.update(userDTO,id): != null  } | | |
| **Salida Esperada:** | ResponseEntity{  response{  “error”: result.getFieldErrors()  }  HttpStatus: HttpStatus.BAD\_REQUEST  } | | |
| **Resultado de la prueba:** | Exitosa | | |
| **Errores obtenidos:** |  | | |
| **Retroalimentación y observaciones:** | El módulo probado funciona adecuadamente para el caso descrito. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Prueba unitaria:** | testUpdateUserTest | **Módulo probado:** | UserController |
| **Responsable:** | Rafael Laos Carrasco | **Fecha:** | 18/07/2022 |
| **Propósito:** | Probar la funcionalidad de actualizar los datos de un usuario creado dentro del módulo de usuario para el caso en que ningún campo de actualización haya sido ingresado erróneamente, pero no se encuentre el id del usuario buscado. | | |
| **Precondiciones:** | Admin Access | **Caso de prueba:** | PU242 |
| **Secuencia del test** | * Se realiza llamada al test cuando se acceda a la ruta asignada a la función update dentro de UserController. * Verifica si existen errores en los campos de entrada. * Verifica si el id del usuario buscado existe en la base de datos. * Trata de consultar en la base de datos parámetros ingresados. * Verifica el estado de la consulta. * Edita los datos del usuario ingresados. * Retorna una respuesta y el estado de la solicitud. * Verifica que el retorno obtenido sea igual al esperado. | | |
| **Descripción de pasos realizados por usuario** | Funcionalidad aún no implementada. | | |
| **Entrada:** | {  {  UserDTO: userDTO  BindingResult: result  Long id: id  }  Exception e: False  result.hasErrors(): False  userService.update(userDTO,id): == null } | | |
| **Salida Esperada:** | ResponseEntity{  response{  “message”: "Error no se pudo editar el usuario con  ID: id no existe en la base de datos”  }  HttpStatus: HttpStatus.NOT\_FOUND  } | | |
| **Resultado de la prueba:** | Exitosa | | |
| **Errores obtenidos:** |  | | |
| **Retroalimentación y observaciones:** | El módulo probado funciona adecuadamente para el caso descrito. | | |
| **Prueba unitaria:** | testUpdateUserTest | **Módulo probado:** | UserController |
| **Responsable:** | Rafael Laos Carrasco | **Fecha:** | 18/07/2022 |
| **Propósito:** | Probar la funcionalidad de actualizar los datos de un usuario creado dentro del módulo de usuario para el caso en que ningún campo de actualización haya sido ingresado erróneamente, se encuentre el id del usuario buscado y no se presenten errores internos del servidor al intentar realizar esta consulta. | | |
| **Precondiciones:** | Admin Access | **Caso de prueba:** | PU243 |
| **Secuencia del test** | * Se realiza llamada al test cuando se acceda a la ruta asignada a la función update dentro de UserController. * Verifica si existen errores en los campos de entrada. * Verifica si el id del usuario buscado existe en la base de datos. * Trata de consultar en la base de datos parámetros ingresados. * Verifica el estado de la consulta. * Edita los datos del usuario ingresados. * Retorna una respuesta y el estado de la solicitud. * Verifica que el retorno obtenido sea igual al esperado. | | |
| **Descripción de pasos realizados por usuario** | Funcionalidad aún no implementada. | | |
| **Entrada:** | {  {  UserDTO: userDTO  BindingResult: result  Long id: id  }  Exception e: False  result.hasErrors(): False  userService.update(userDTO,id): != null } | | |
| **Salida Esperada:** | ResponseEntity{  response{  “message”: "El usuario ha sido actualizado con éxito!”  “data”: userService.update(userDTO,id)  }  HttpStatus: HttpStatus.OK } | | |
| **Resultado de la prueba:** | Exitosa | | |
| **Errores obtenidos:** |  | | |
| **Retroalimentación y observaciones:** | El módulo probado funciona adecuadamente para el caso descrito. | | |
| **Prueba unitaria:** | testUpdateUserTest | **Módulo probado:** | UserController |
| **Responsable:** | Rafael Laos Carrasco | **Fecha:** | 18/07/2022 |
| **Propósito:** | Probar la funcionalidad de actualizar los datos de un usuario creado dentro del módulo de usuario para el caso en que ningún campo de actualización haya sido ingresado erróneamente, se encuentre el id del usuario buscado, pero se presenten errores internos del servidor al intentar realizar esta consulta. | | |
| **Precondiciones:** | Admin Access | **Caso de prueba:** | PU244 |
| **Secuencia del test** | * Se realiza llamada al test cuando se acceda a la ruta asignada a la función update dentro de UserController. * Verifica si existen errores en los campos de entrada. * Verifica si el id del usuario buscado existe en la base de datos. * Trata de consultar en la base de datos parámetros ingresados. * Verifica el estado de la consulta. * Edita los datos del usuario ingresados. * Retorna una respuesta y el estado de la solicitud. * Verifica que el retorno obtenido sea igual al esperado. | | |
| **Descripción de pasos realizados por usuario** | Funcionalidad aún no implementada. | | |
| **Entrada:** | {  {  UserDTO: userDTO  BindingResult: result  Long id: id }  Exception e: True  result.hasErrors(): False  userService.update(userDTO,id): != null } | | |
| **Salida Esperada:** | ResponseEntity{  response{  “message”: “Error al realizar la consulta a la base  de datos”  “error”: e.getMessage()+e.getMostEspecificCause()  }  HttpStatus: HttpStatus.INTERNAL\_SERVER\_ERROR } | | |
| **Resultado de la prueba:** | Exitosa | | |
| **Errores obtenidos:** |  | | |
| **Retroalimentación y observaciones:** | El módulo probado funciona adecuadamente para el caso descrito. | | |
| **Prueba unitaria:** | testDeleteUserTest | **Módulo probado:** | UserController |
| **Responsable:** | Rafael Laos Carrasco | **Fecha:** | 18/07/2022 |
| **Propósito:** | Probar la funcionalidad de eliminar un usuario creado dentro del módulo de usuario para el caso en que no se presenten errores internos del servidor al intentar realizar esta consulta. | | |
| **Precondiciones:** | Admin Access | **Caso de prueba:** | PU251 |
| **Secuencia del test** | * Se realiza llamada al test cuando se acceda a la ruta asignada a la función delete dentro de UserController. * Trata de consultar en la base de datos parámetros ingresados. * Verifica el estado de la consulta. * Elimina el usuario correspondiente al id ingresado de la base de datos * Retorna una respuesta y el estado de la solicitud. * Verifica que el retorno obtenido sea igual al esperado. | | |
| **Descripción de pasos realizados por usuario** | Funcionalidad aún no implementada. | | |
| **Entrada:** | {  RequestParam:{  Long id: id  }  Exception e: False  } | | |
| **Salida Esperada:** | ResponseEntity{  response{  “message”: “El usuario ha sido eliminado don exito!”  “data”: userService.findById(id)  }  HttpStatus: HttpStatus.OK  } | | |
| **Resultado de la prueba:** | Exitosa | | |
| **Errores obtenidos:** |  | | |
| **Retroalimentación y observaciones:** | El módulo probado funciona adecuadamente para el caso descrito. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Prueba unitaria:** | testDeleteUserTest | **Módulo probado:** | UserController |
| **Responsable:** | Rafael Laos Carrasco | **Fecha:** | 18/07/2022 |
| **Propósito:** | Probar la funcionalidad de eliminar un usuario creado dentro del módulo de usuario para el caso en que se presenten errores internos del servidor al intentar realizar esta consulta. | | |
| **Precondiciones:** | Admin Access | **Caso de prueba:** | PU252 |
| **Secuencia del test** | * Se realiza llamada al test cuando se acceda a la ruta asignada a la función delete dentro de UserController. * Trata de consultar en la base de datos parámetros ingresados. * Verifica el estado de la consulta. * Elimina el usuario correspondiente al id ingresado de la base de datos * Retorna una respuesta y el estado de la solicitud. * Verifica que el retorno obtenido sea igual al esperado. | | |
| **Descripción de pasos realizados por usuario** | Funcionalidad aún no implementada. | | |
| **Entrada:** | {  RequestParam:{  Long id: id  }  Exception e: True  } | | |
| **Salida Esperada:** | ResponseEntity{  response{  “message”: “Error al eliminar el usuario de la base de datos”  “error”: e.getMessage()+e.getLocalizadMessage()  }  HttpStatus: HttpStatus.INTERNAL\_SERVER\_ERROR  } | | |
| **Resultado de la prueba:** | Exitosa | | |
| **Errores obtenidos:** |  | | |
| **Retroalimentación y observaciones:** | El módulo probado funciona adecuadamente para el caso descrito. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Prueba unitaria:** | testGetAllProductTest | **Módulo probado:** | ProductController |
| **Responsable:** | Rafael Laos Carrasco | **Fecha:** | 05/08/2022 |
| **Propósito:** | Probar la funcionalidad de realizar una consulta completa a la base de datos dentro del módulo de producto para el caso en que no hallan errores internos del servidor al intentar realizar esta consulta. | | |
| **Precondiciones:** | Admin, Grocer Access | **Caso de prueba:** | PU301 |
| **Secuencia del test** | * Se realiza llamada al test cuando se acceda a la ruta asignada a la función getAll dentro de ProductController. * Trata de consultar los parámetros entrantes en la base de datos. * Verifica el estado de la consulta. * Retorna una respuesta y el estado de la solicitud. * Verifica que el retorno obtenido sea igual al esperado. | | |
| **Descripción de pasos realizados por usuario** |  | | |
| **Entrada:** | {  RequestParam:{  int numberOfPage: 0  int sizeOfPage: 10  String sortPeer: “id”  String sortDir: “asc”  }  Exception e: False  } | | |
| **Salida Esperada:** | ResponseEntity{  response{  “data”: productService.getAll(0,10,”id”,”asc”)  }  HttpStatus: HttpStatus.OK  } | | |
| **Resultado de la prueba:** | Exitosa | | |
| **Errores obtenidos:** |  | | |
| **Retroalimentación y observaciones:** | El módulo probado funciona adecuadamente para el caso descrito. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Prueba unitaria:** | testGetAllProductTest | **Módulo probado:** | ProductController |
| **Responsable:** | Rafael Laos Carrasco | **Fecha:** | 05/08/2022 |
| **Propósito:** | Probar la funcionalidad de realizar una consulta completa a la base de datos dentro del módulo de producto para el caso en que se presenten errores internos del servidor al intentar realizar esta consulta. | | |
| **Precondiciones:** | Admin, Grocer Access | **Caso de prueba:** | PU302 |
| **Secuencia del test** | * Se realiza llamada al test cuando se acceda a la ruta asignada a la función getAll dentro de ProductController. * Trata de consultar los parámetros entrantes en la base de datos. * Verifica el estado de la consulta. * Retorna una respuesta y el estado de la solicitud. * Verifica que el retorno obtenido sea igual al esperado. | | |
| **Descripción de pasos realizados por usuario** |  | | |
| **Entrada:** | {  RequestParam:{  int numberOfPage: 0  int sizeOfPage: 10  String sortPeer: “id”  String sortDir: “asc”  }  Exception e: True  } | | |
| **Salida Esperada:** | ResponseEntity{  response{  “error”: “Error al realizar la consulta a la base de  datos”  }  HttpStatus: HttpStatus.INTERNAL\_SERVER\_ERROR  } | | |
| **Resultado de la prueba:** | Exitosa | | |
| **Errores obtenidos:** |  | | |
| **Retroalimentación y observaciones:** | El módulo probado funciona adecuadamente para el caso descrito. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Prueba unitaria:** | testShowProductTest | **Módulo probado:** | ProductController |
| **Responsable:** | Rafael Laos Carrasco | **Fecha:** | 05/08/2022 |
| **Propósito:** | Probar la funcionalidad de mostrar los datos de un producto registrado buscándolo por su id en la base de datos dentro del módulo de producto para el caso en que se presenten errores internos del servidor al intentar realizar esta consulta. | | |
| **Precondiciones:** | Admin, Grocer Access | **Caso de prueba:** | PU311 |
| **Secuencia del test** | * Se realiza llamada al test cuando se acceda a la ruta asignada a la función show dentro de ProductController. * Trata de consultar el producto buscado por su id en la base de datos. * Verifica el estado de la consulta. * Retorna una respuesta y el estado de la solicitud. * Verifica que el retorno obtenido sea igual al esperado. | | |
| **Descripción de pasos realizados por usuario** |  | | |
| **Entrada:** | {  RequestParam:{  Long id: 123  }  Exception e: True  } | | |
| **Salida Esperada:** | ResponseEntity{  response{  “message”: “Error al realizar la consulta a la base de  datos”  “error”: e.getMessage()+e.getMostEspecificCause()  }  HttpStatus: HttpStatus.INTERNAL\_SERVER\_ERROR  } | | |
| **Resultado de la prueba:** | Exitosa | | |
| **Errores obtenidos:** |  | | |
| **Retroalimentación y observaciones:** | El módulo probado funciona adecuadamente para el caso descrito. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Prueba unitaria:** | testShowProductTest | **Módulo probado:** | ProductController |
| **Responsable:** | Rafael Laos Carrasco | **Fecha:** | 05/08/2022 |
| **Propósito:** | Probar la funcionalidad de mostrar los datos de un producto registrado buscándolo por su id en la base de datos dentro del módulo de producto para el caso en que no hallan errores internos del servidor al intentar realizar esta consulta. | | |
| **Precondiciones:** | Admin, Grocer Access | **Caso de prueba:** | PU312 |
| **Secuencia del test** | * Se realiza llamada al test cuando se acceda a la ruta asignada a la función show dentro de ProductController. * Trata de consultar el producto buscado por su id en la base de datos. * Verifica el estado de la consulta. * Retorna una respuesta y el estado de la solicitud. * Verifica que el retorno obtenido sea igual al esperado. | | |
| **Descripción de pasos realizados por usuario** |  | | |
| **Entrada:** | {  RequestParam:{  Long id: 123  }  Exception e: False  } | | |
| **Salida Esperada:** | ResponseEntity{  response{  “data”: productService.findById(123)  }  HttpStatus: HttpStatus.OK  } | | |
| **Resultado de la prueba:** | Exitosa | | |
| **Errores obtenidos:** |  | | |
| **Retroalimentación y observaciones:** | El módulo probado funciona adecuadamente para el caso descrito. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Prueba unitaria:** | testCreateProductTest | **Módulo probado:** | ProductController |
| **Responsable:** | Rafael Laos Carrasco | **Fecha:** | 05/08/2022 |
| **Propósito:** | Probar la funcionalidad de crear un nuevo producto dentro del módulo de producto para el caso en que algún campo del producto haya sido ingresado erróneamente. | | |
| **Precondiciones:** | Admin, Grocer Access | **Caso de prueba:** | PU321 |
| **Secuencia del test** | * Se realiza llamada al test cuando se acceda a la ruta asignada a la función create dentro de PrductController. * Verifica si la entrada ingresada tiene errores. * Trata de consultar los parámetros entrantes en la base de datos. * Verifica el estado de la consulta. * Crea el producto requerido y lo asigna a la respuesta. * Retorna una respuesta y el estado de la solicitud. * Verifica que el retorno obtenido sea igual al esperado. | | |
| **Descripción de pasos realizados por usuario** |  | | |
| **Entrada:** | {  RequestBody:{  ProductDTO: productDTO  Binding result: result  }  result.hasErrors(): True  Exception e: False  } | | |
| **Salida Esperada:** | ResponseEntity{  response{  “errors”: result.getFieldErrors()  }  HttpStatus: HttpStatus.BAD\_REQUEST  } | | |
| **Resultado de la prueba:** | Exitosa | | |
| **Errores obtenidos:** |  | | |
| **Retroalimentación y observaciones:** | El módulo probado funciona adecuadamente para el caso descrito. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Prueba unitaria:** | testCreateProductTest | **Módulo probado:** | ProductController |
| **Responsable:** | Rafael Laos Carrasco | **Fecha:** | 05/08/2022 |
| **Propósito:** | Probar la funcionalidad de crear un nuevo producto dentro del módulo de producto para el caso en que ningún campo del producto haya sido ingresado erróneamente, pero se presenten errores internos del servidor al intentar realizar esta consulta. | | |
| **Precondiciones:** | Admin, Grocer Access | **Caso de prueba:** | PU322 |
| **Secuencia del test** | * Se realiza llamada al test cuando se acceda a la ruta asignada a la función create dentro de PrductController. * Verifica si la entrada ingresada tiene errores. * Trata de consultar los parámetros entrantes en la base de datos. * Verifica el estado de la consulta. * Crea el producto requerido y lo asigna a la respuesta. * Retorna una respuesta y el estado de la solicitud. * Verifica que el retorno obtenido sea igual al esperado. | | |
| **Descripción de pasos realizados por usuario** |  | | |
| **Entrada:** | {  RequestBody:{  ProductDTO: productDTO  Binding result: result  }  result.hasErrors(): False  Exception e: True  } | | |
| **Salida Esperada:** | ResponseEntity{  response{  “message”: “Error al realizar la consulta a la base de datos”  “error”: e.getMessage()+e.getMostEspecificCause()  }  HttpStatus: HttpStatus.INTERNAL\_SERVER\_ERROR  } | | |
| **Resultado de la prueba:** | Exitosa | | |
| **Errores obtenidos:** |  | | |
| **Retroalimentación y observaciones:** | El módulo probado funciona adecuadamente para el caso descrito. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Prueba unitaria:** | testCreateProductTest | **Módulo probado:** | ProductController |
| **Responsable:** | Rafael Laos Carrasco | **Fecha:** | 05/08/2022 |
| **Propósito:** | Probar la funcionalidad de crear un nuevo producto dentro del módulo de producto para el caso en que ningún campo de registro haya sido ingresado erróneamente y no se presenten errores internos del servidor al intentar realizar esta consulta. | | |
| **Precondiciones:** | Admin, Grocer Access | **Caso de prueba:** | PU323 |
| **Secuencia del test** | * Se realiza llamada al test cuando se acceda a la ruta asignada a la función create dentro de PrductController. * Verifica si la entrada ingresada tiene errores. * Trata de consultar los parámetros entrantes en la base de datos. * Verifica el estado de la consulta. * Crea el producto requerido y lo asigna a la respuesta. * Retorna una respuesta y el estado de la solicitud. * Verifica que el retorno obtenido sea igual al esperado. | | |
| **Descripción de pasos realizados por usuario** |  | | |
| **Entrada:** | {  RequestBody:{  ProductDTO: productDTO  Binding result: result  }  result.hasErrors(): False  Exception e: False  } | | |
| **Salida Esperada:** | ResponseEntity{  response{  “message”: “El usuario ha sido creado con exito!”  “data”: productService.save(productDTO)  }  HttpStatus: HttpStatus.CREATED  } | | |
| **Resultado de la prueba:** | Exitosa | | |
| **Errores obtenidos:** |  | | |
| **Retroalimentación y observaciones:** | El módulo probado funciona adecuadamente para el caso descrito. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Prueba unitaria:** | testUpdateProductTest | **Módulo probado:** | ProductController |
| **Responsable:** | Rafael Laos Carrasco | **Fecha:** | 05/08/2022 |
| **Propósito:** | Probar la funcionalidad de actualizar los datos de un producto creado dentro del módulo de producto para el caso en que algún campo de actualización haya sido ingresado erróneamente. | | |
| **Precondiciones:** | Admin, Grocer Access | **Caso de prueba:** | PU341 |
| **Secuencia del test** | * Se realiza llamada al test cuando se acceda a la ruta asignada a la función update dentro de ProductController. * Verifica si existen errores en los campos de entrada. * Trata de consultar en la base de datos parámetros ingresados. * Verifica el estado de la consulta. * Edita los datos del producto ingresados. * Retorna una respuesta y el estado de la solicitud. * Verifica que el retorno obtenido sea igual al esperado. | | |
| **Descripción de pasos realizados por usuario** |  | | |
| **Entrada:** | {  {  ProductDTO: productDTO  BindingResult: result  Long id: id  }  Exception e: False  result.hasErrors(): True  } | | |
| **Salida Esperada:** | ResponseEntity{  response{  “error”: result.getFieldErrors()  }  HttpStatus: HttpStatus.BAD\_REQUEST  } | | |
| **Resultado de la prueba:** | Exitosa | | |
| **Errores obtenidos:** |  | | |
| **Retroalimentación y observaciones:** | El módulo probado funciona adecuadamente para el caso descrito. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Prueba unitaria:** | testUpdateProductTest | **Módulo probado:** | ProductController |
| **Responsable:** | Rafael Laos Carrasco | **Fecha:** | 05/08/2022 |
| **Propósito:** | Probar la funcionalidad de actualizar los datos de un producto creado dentro del módulo de producto para el caso en que ningún campo de actualización haya sido ingresado erróneamente y no se hallen errores internos del servidor al intentar realizar esta consulta. | | |
| **Precondiciones:** | Admin, Grocer Access | **Caso de prueba:** | PU342 |
| **Secuencia del test** | * Se realiza llamada al test cuando se acceda a la ruta asignada a la función update dentro de ProductController. * Verifica si existen errores en los campos de entrada. * Trata de consultar en la base de datos parámetros ingresados. * Verifica el estado de la consulta. * Edita los datos del producto ingresados. * Retorna una respuesta y el estado de la solicitud. * Verifica que el retorno obtenido sea igual al esperado. | | |
| **Descripción de pasos realizados por usuario** |  | | |
| **Entrada:** | {  {  ProductDTO: productDTO  BindingResult: result  Long id: id  }  Exception e: False  result.hasErrors(): False  } | | |
| **Salida Esperada:** | ResponseEntity{  response{  “message”: "El producto ha sido actualizado con  éxito!”  “data”: productService.update(productDTO,id)  }  HttpStatus: HttpStatus.OK  } | | |
| **Resultado de la prueba:** | Exitosa | | |
| **Errores obtenidos:** |  | | |
| **Retroalimentación y observaciones:** | El módulo probado funciona adecuadamente para el caso descrito. | | |
| **Prueba unitaria:** | testUpdateProductTest | **Módulo probado:** | ProductController |
| **Responsable:** | Rafael Laos Carrasco | **Fecha:** | 05/08/2022 |
| **Propósito:** | Probar la funcionalidad de actualizar los datos de un producto creado dentro del módulo de producto para el caso en que ningún campo de actualización haya sido ingresado erróneamente, pero se presenten errores internos del servidor al intentar realizar esta consulta. | | |
| **Precondiciones:** | Admin, Grocer Access | **Caso de prueba:** | PU343 |
| **Secuencia del test** | * Se realiza llamada al test cuando se acceda a la ruta asignada a la función update dentro de ProductController. * Verifica si existen errores en los campos de entrada. * Trata de consultar en la base de datos parámetros ingresados. * Verifica el estado de la consulta. * Edita los datos del producto ingresados. * Retorna una respuesta y el estado de la solicitud. * Verifica que el retorno obtenido sea igual al esperado. | | |
| **Descripción de pasos realizados por usuario** |  | | |
| **Entrada:** | {  {  ProductDTO: productDTO  BindingResult: result  Long id: id  }  Exception e: True  result.hasErrors(): False  } | | |
| **Salida Esperada:** | ResponseEntity{  response{  “message”: “Error al realizar la consulta a la base  de datos”  “error”: e.getMessage()+e.getMostEspecificCause()  }  HttpStatus: HttpStatus.INTERNAL\_SERVER\_ERROR } | | |
| **Resultado de la prueba:** | Exitosa | | |
| **Errores obtenidos:** |  | | |
| **Retroalimentación y observaciones:** | El módulo probado funciona adecuadamente para el caso descrito. | | |
| **Prueba unitaria:** | testDeleteProductTest | **Módulo probado:** | ProductController |
| **Responsable:** | Rafael Laos Carrasco | **Fecha:** | 05/08/2022 |
| **Propósito:** | Probar la funcionalidad de eliminar un producto creado dentro del módulo de producto para el caso en que no se presenten errores internos del servidor al intentar realizar esta consulta. | | |
| **Precondiciones:** | Admin, Grocer Access | **Caso de prueba:** | PU351 |
| **Secuencia del test** | * Se realiza llamada al test cuando se acceda a la ruta asignada a la función delete dentro de ProductController. * Trata de consultar en la base de datos parámetros ingresados. * Verifica el estado de la consulta. * Elimina el usuario correspondiente al id ingresado de la base de datos * Retorna una respuesta y el estado de la solicitud. * Verifica que el retorno obtenido sea igual al esperado. | | |
| **Descripción de pasos realizados por usuario** |  | | |
| **Entrada:** | {  RequestParam:{  Long id: id  }  Exception e: False  } | | |
| **Salida Esperada:** | ResponseEntity{  response{  “message”: “El producto ha sido eliminado con  exito!”  “data”: productService.findById(id)  }  HttpStatus: HttpStatus.OK  } | | |
| **Resultado de la prueba:** | Exitosa | | |
| **Errores obtenidos:** |  | | |
| **Retroalimentación y observaciones:** | El módulo probado funciona adecuadamente para el caso descrito. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Prueba unitaria:** | testDeleteProductTest | **Módulo probado:** | ProductController |
| **Responsable:** | Rafael Laos Carrasco | **Fecha:** | 05/08/2022 |
| **Propósito:** | Probar la funcionalidad de eliminar un producto creado dentro del módulo de producto para el caso en que se hallen errores internos del servidor al intentar realizar esta consulta. | | |
| **Precondiciones:** | Admin, Grocer Access | **Caso de prueba:** | PU352 |
| **Secuencia del test** | * Se realiza llamada al test cuando se acceda a la ruta asignada a la función delete dentro de ProductController. * Trata de consultar en la base de datos parámetros ingresados. * Verifica el estado de la consulta. * Elimina el usuario correspondiente al id ingresado de la base de datos * Retorna una respuesta y el estado de la solicitud. * Verifica que el retorno obtenido sea igual al esperado. | | |
| **Descripción de pasos realizados por usuario** |  | | |
| **Entrada:** | {  RequestParam:{  Long id: id  }  Exception e: True  } | | |
| **Salida Esperada:** | ResponseEntity{  response{  “message”: “Error al eliminar el producto de la base de datos”  “error”: e.getMessage()+e.getLocalizadMessage()  }  HttpStatus: HttpStatus.INTERNAL\_SERVER\_ERROR  } | | |
| **Resultado de la prueba:** | Exitosa | | |
| **Errores obtenidos:** |  | | |
| **Retroalimentación y observaciones:** | El módulo probado funciona adecuadamente para el caso descrito. | | |

**Resultados obtenidos y conclusiones:**

Para el hito final, se realizaron pruebas unitarias a las funcionalidades desarrolladas e integradas incluidas en los módulos backend de autenticación de login (AuthController.java), usuario (UserController.java) y producto (ProductController.java), se probaron todos los casos lógicamente posibles y se obtuvieron los resultados esperados sin errores. Los tres módulos funcionales descritos son aprobados para la realización de pruebas de integración y pruebas no funcionales correspondientes para el despliegue final del sistema.