**Universidad Nacional de Ingeniería**

[Facultad de Electrotecnia y Computación](http://www.fec.uni.edu.ni/) Ingeniería de Software II

### **Carrera:**

Ingeniería en Computación

### **Titulo Laboratorio:**

Laboratorio III– Diseño de casos de prueba caja blanca y caja negra

### **Grupo:**

5T1-CO

### **Integrantes:**

2015 - 0012U Castillo Hurtado Yader David.

2015 - 0578U Rivera Moreira Oscar Danilo.

2015 - 1107U Sánchez Narváez Cristopher David.

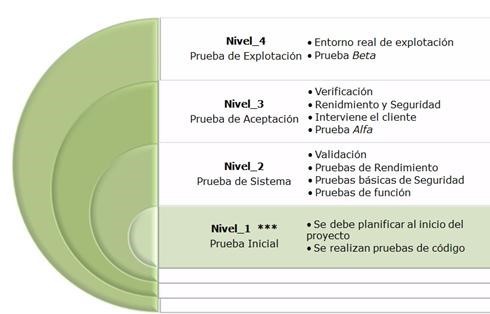
**I - Objetivos de realizar pruebas de sistema**

Las pruebas del sistema tienen como objetivo ejercitar profundamente el sistema comprobando la integración del sistema de información globalmente, verificando el funcionamiento correcto de las interfaces entre los distintos subsistemas que lo componen y con el resto de sistemas de información con los que se comunica.

Son pruebas de integración del sistema de información completo, y permiten probar el sistema en su conjunto y con otros sistemas con los que se relaciona para verificar que las especificaciones funcionales y técnicas se cumplen. Dan una visión muy similar a su comportamiento en el entorno de producción.

**II - Tipos de pruebas que se realizan a los sistemas de información**

* **Pruebas funcionales**. Dirigidas a asegurar que el sistema de información realiza correctamente todas las funciones que se han detallado en las especificaciones dadas por el usuario del sistema.
* **Pruebas de comunicaciones**. Determinan que las interfaces entre los componentes del sistema funcionan adecuadamente, tanto a través de dispositivos remotos, como locales. Asimismo, se han de probar las interfaces hombre/máquina.
* **Pruebas de rendimiento**. Consisten en determinar que los tiempos de respuesta están dentro de los intervalos establecidos en las especificaciones del sistema.
* **Pruebas de volumen**. Consisten en examinar el funcionamiento del sistema cuando está trabajando con grandes volúmenes de datos, simulando las cargas de trabajo esperadas.
* **Pruebas de sobrecarga**. Consisten en comprobar el funcionamiento del sistema en el umbral límite de los recursos, sometiéndole a cargas masivas. El objetivo es establecer los puntos extremos en los cuales el sistema empieza a operar por debajo de los requisitos establecidos.
* **Pruebas de disponibilidad de datos**. Consisten en demostrar que el sistema puede recuperarse ante fallos, tanto de equipo físico como lógico, sin comprometer la integridad de los datos.
* **Pruebas de facilidad de uso**. Consisten en comprobar la adaptabilidad del sistema a las necesidades de los usuarios, tanto para asegurar que se acomoda a su modo habitual de trabajo, como para determinar las facilidades que aporta al introducir datos en el sistema y obtener los resultados.
* **Pruebas de operación**. Consisten en comprobar la correcta implementación de los procedimientos de operación, incluyendo la planificación y control de trabajos, arranque y rearranque del sistema, etc.
* **Pruebas de entorno**. Consisten en verificar las interacciones del sistema con otros sistemas dentro del mismo entorno.
* **Pruebas de seguridad**. Consisten en verificar los mecanismos de control de acceso al sistema para evitar alteraciones indebidas en los datos.

**III - Ciclo de prueba en un sistema de información**

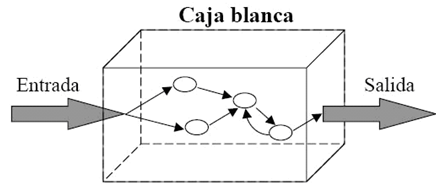
**Fig. 1: Ciclo de prueba de un S.I.**

**IV - En qué consisten las pruebas de caja blanca.**

La prueba de caja blanca se basa en el diseño de casos de prueba que usa la estructura de control del diseño procedimental para derivarlos. Mediante la prueba de la caja blanca el ingeniero del software puede obtener casos de prueba que:

1. Garanticen que se ejerciten por lo menos una vez todos los caminos independientes de cada módulo, programa o método.
2. Ejerciten todas las decisiones lógicas en las vertientes verdadera y falsa.
3. Ejecuten todos los bucles en sus límites operacionales.
4. Ejerciten las estructuras internas de datos para asegurar su validez.

Es por ello que se considera a la prueba de Caja Blanca como uno de los tipos de pruebas más importantes que se le aplican a los software, logrando como resultado que disminuya en un gran porcentaje el número de errores existentes en los sistemas y por ende una mayor calidad y confiabilidad.

**Fig. 2: Representación de caja blanca.**

Algunas pruebas de caja blanca son las siguientes:

**1. Camino Básico**

Esta técnica permite obtener una medida de la complejidad lógica de un diseño y usar esta medida como guía para la definición de un conjunto básico.

La idea es derivar casos de prueba a partir de un conjunto dado de caminos independientes por los cuales puede circular el flujo de control. Para obtener dicho conjunto de caminos independientes se construye el Grafo de Flujo asociado y se calcula su complejidad ciclomática. Los pasos que se siguen para aplicar esta técnica son:

1. A partir del diseño o del código fuente, se dibuja el grafo de flujo asociado.
2. Se calcula la complejidad ciclomática del grafo.
3. Se determina un conjunto básico de caminos independientes.
4. Se preparan los casos de prueba que obliguen a la ejecución de cada camino del conjunto básico.

Los casos de prueba derivados del conjunto básico garantizan que durante la prueba se ejecuta por lo menos una vez cada sentencia del programa.

**2. Notación de grafo de flujo**

Un Grafo de Flujo está formado por 3 componentes fundamentales que ayudan a su elaboración, comprensión y nos brinda información para confirmar que el trabajo se está haciendo adecuadamente.

Los componentes son:

### Nodo

Cada círculo representado se denomina nodo del Grafo de Flujo, el cual representa una o más secuencias procedimentales. Un solo nodo puede corresponder a una secuencia de procesos o a una sentencia de decisión. Puede ser también que haya nodos que no se asocien, se utilizan principalmente al inicio y final del grafo.

### Aristas

Las flechas del grafo se denominan aristas y representan el flujo de control, son análogas a las representadas en un diagrama de flujo. Una arista debe terminar en un nodo, incluso aunque el nodo no represente ninguna sentencia procedimental.

### Regiones

Las regiones son las áreas delimitadas por las aristas y nodos. También se incluye el área exterior del grafo, contando como una región más. Las regiones se enumeran. La cantidad de regiones es equivalente a la cantidad de caminos independientes del conjunto básico de un programa.

**3. Complejidad Ciclomática**

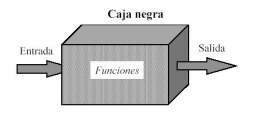
La complejidad ciclomática es una métrica de software extremadamente útil pues proporciona una medición cuantitativa de la complejidad lógica de un programa. El valor calculado como complejidad ciclomática define el número de caminos independientes del conjunto básico de un programa y nos da un límite superior para el número de pruebas que se deben realizar para asegurar que se ejecute cada sentencia al menos una vez. Un camino independiente es cualquier camino del programa que introduce por lo menos un nuevo conjunto de sentencias de procesamiento o una nueva condición. El camino independiente se debe mover por lo menos por una arista que no haya sido recorrida anteriormente.

**V - En qué consisten las pruebas de caja negra.**

Estas pruebas permiten obtener un conjunto de condiciones de entrada que ejerciten completamente todos los requisitos funcionales de un programa. En ellas se ignora la estructura de control, concentrándose en los requisitos funcionales del sistema y ejercitándolos.

La prueba de Caja Negra no es una alternativa a las técnicas de prueba de la Caja Blanca, sino un enfoque complementario que intenta descubrir diferentes tipos de errores a los encontrados en los métodos de la Caja Blanca. Muchos autores consideran que estas pruebas permiten encontrar:

1. Funciones incorrectas o ausentes.
2. Errores de interfaz.
3. Errores en estructuras de datos o en accesos a las Bases de Datos externas.
4. Errores de rendimiento.
5. Errores de inicialización y terminación.



**Fig. 3: Representación de una caja negra**

Algunas de las pruebas de caja negra son las siguientes:

**1. Particion equivalente**

Una partición equivalente es una técnica de prueba de Caja Negra que divide el dominio de entrada de un programa en clases de datos de los que se pueden derivar casos de prueba. El diseño de estos casos de prueba para la partición equivalente se basa en la evaluación de las clases de equivalencia.

El diseño de casos de prueba para la partición equivalente se basa en una evaluación de las clases de equivalencia para una condición de entrada. Una clase de equivalencia representa un conjunto de estados válidos o inválidos para condiciones de entrada.

Regularmente, una condición de entrada es un valor numérico específico, un rango de valores, un conjunto de valores relacionados o una condición lógica.

**2. Procedimientos de prueba**

Un procedimiento de prueba especifica cómo realizar uno o varios casos de prueba o parte de estos. Por ejemplo, un procedimiento de prueba puede ser una instrucción para un individuo sobre cómo ha de realizar un caso de prueba manualmente, o puede ser una especificaron de cómo interaccionar manualmente con una herramienta de automatización de pruebas para crear componentes ejecutables de pruebas.

El cómo llevar a cabo un caso de prueba puede ser especificado por un procedimiento de prueba, pero es a menudo útil reutilizar un procedimiento de prueba para varios casos de prueba y reutilizar varios procedimientos de prueba para varios casos de prueba.

**3. Componentes de prueba**

Un componente de prueba automatiza uno o varios procedimientos de prueba o parte de ellos.

Los componentes de pruebas pueden ser desarrollados utilizando lenguaje de guiones o un lenguaje de programación o pueden ser grabados con una herramienta de automatización de pruebas.

Los componentes de pruebas se utilizan para probar los componentes en el modelo de implementación proporcionando entradas de prueba, controlando y monitoreando la ejecución de los componentes a probar y, posiblemente informando de los resultados de las pruebas.

**4. Plan de prueba**

El propósito del plan de pruebas es dejar de forma explícita el alcance, el enfoque, los recursos requeridos, el calendario, los responsables y el manejo de riesgos de un proceso de pruebas.

Está constituido por un conjunto de pruebas. Cada prueba debe:

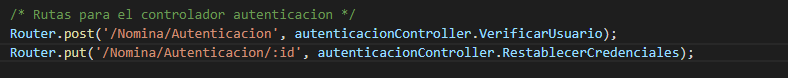
* Dejar claro qué tipo de propiedades se quieren probar (corrección, robustez, fiabilidad, amigabilidad...).
* Dejar claro cómo se mide el resultado.
* Especificar en qué consiste la prueba (hasta el último detalle de cómo se ejecuta).
* Definir cuál es el resultado que se espera (identificación, tolerancia...). ¿Cómo se decide que el resultado es acorde con lo esperado?

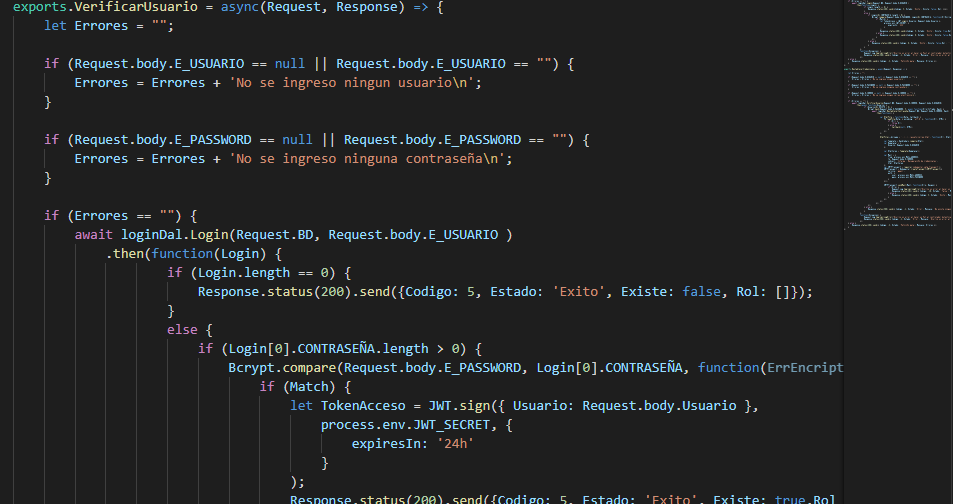
Las pruebas carecen de utilidad, tanto, si no se sabe exactamente lo que se quiere probar, sí no se está claro cómo se prueba, o si el análisis del resultado se hace a simple vista. Estas mismas ideas se suelen agrupar diciendo que un caso de prueba consta de 3 bloques de información:

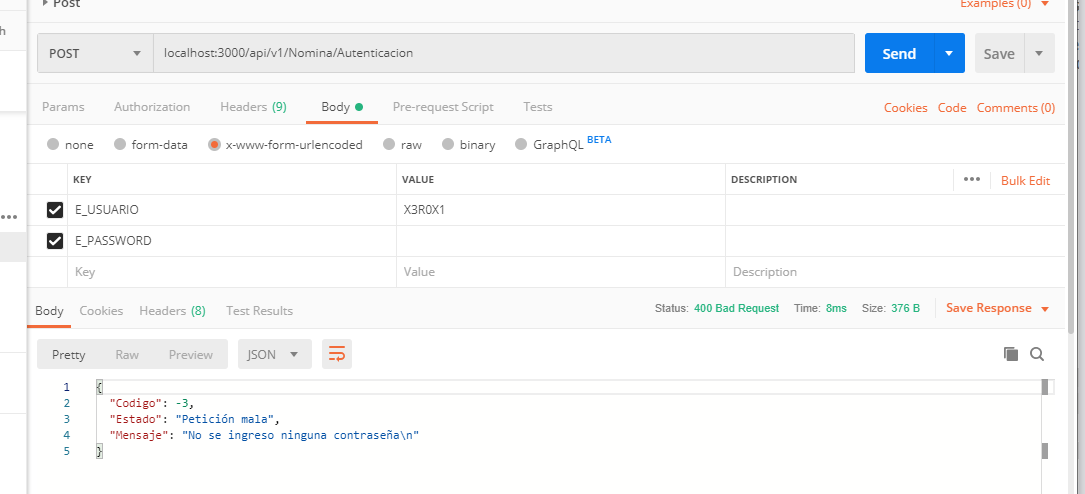
1. El propósito de la prueba.
2. Los pasos de ejecución de la prueba.
3. El resultado que se espera.

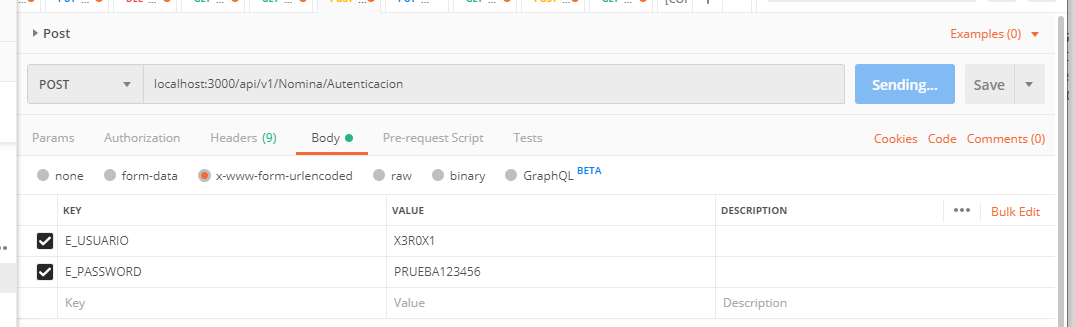
**VI - Ejemplos de Caja Blanca Aplicados al Sistema de Información del Proyecto de Curso.**

**1 – Autenticación**

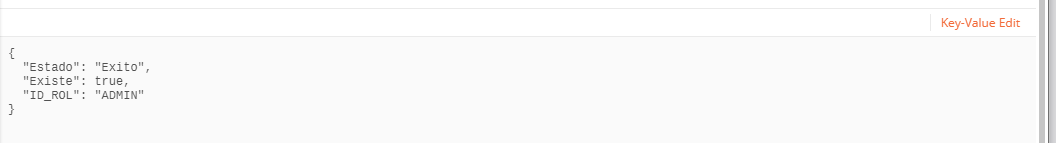
**Fig. 5: Ruta del controlador de autentificación**

**Fig. 6: Validación del proceso de inicio de sesión.**

**Fig. 7: Petición de insertar en la API usando POST MAN**

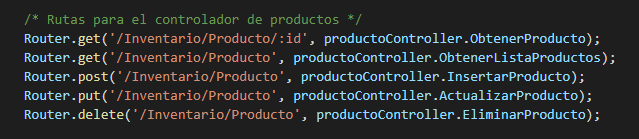


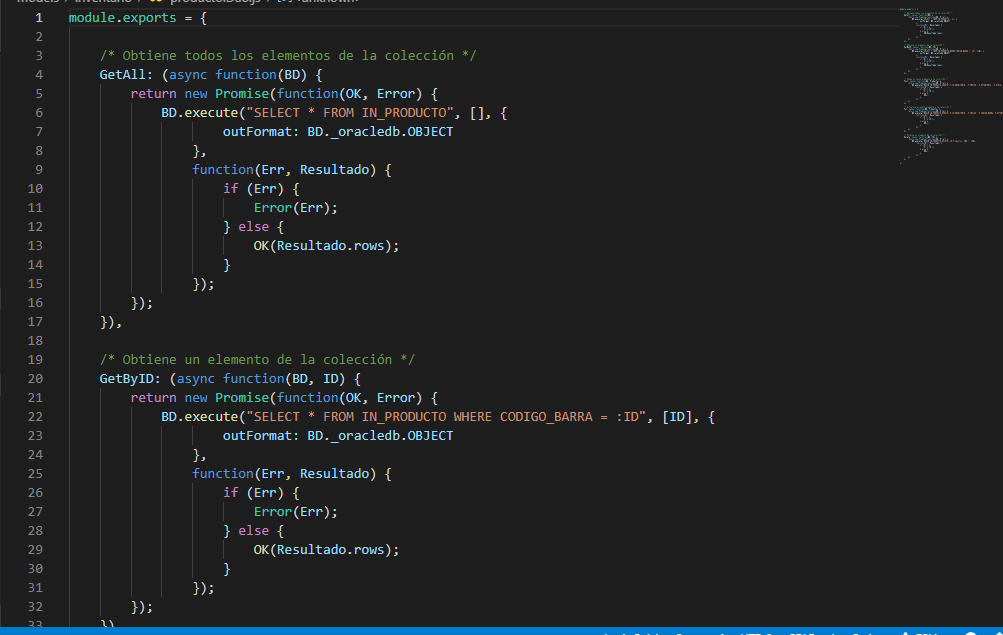
**Fig. 8: Valores existente en la BD**

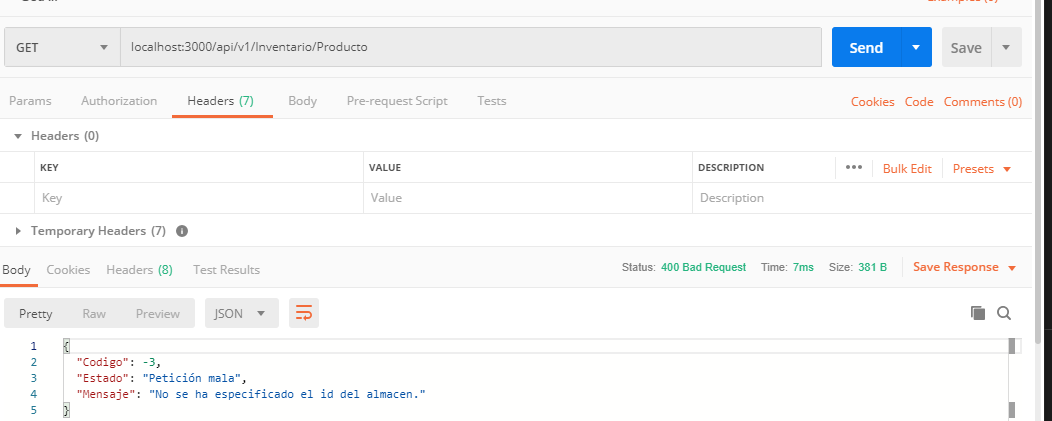


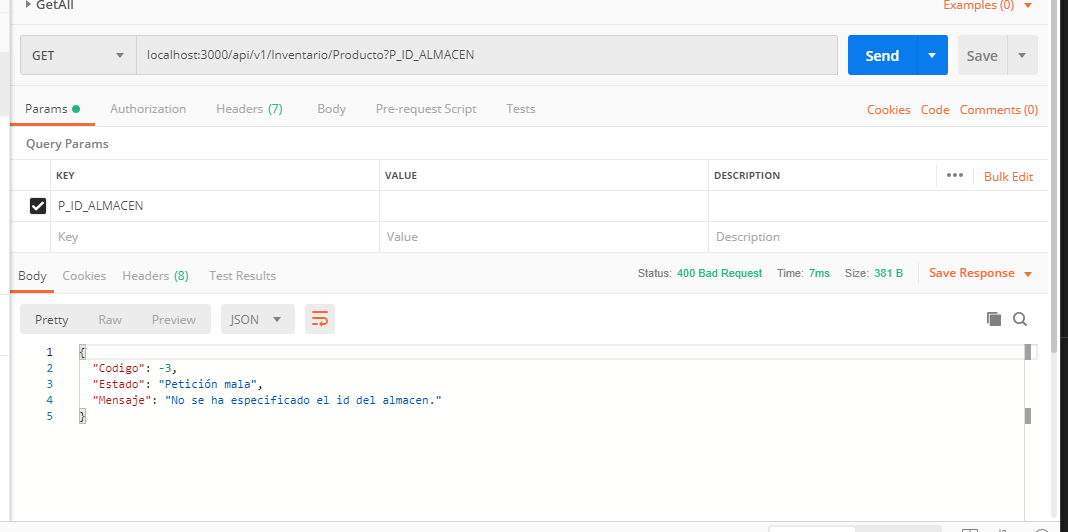
**Fig. 9: Retorno de valores insertados.**

**2 – Tabla Producto**

**Fig. 10: Ruta del controlador de Producto**

**Fig. 11: Validación del proceso de inicio de sesión.**

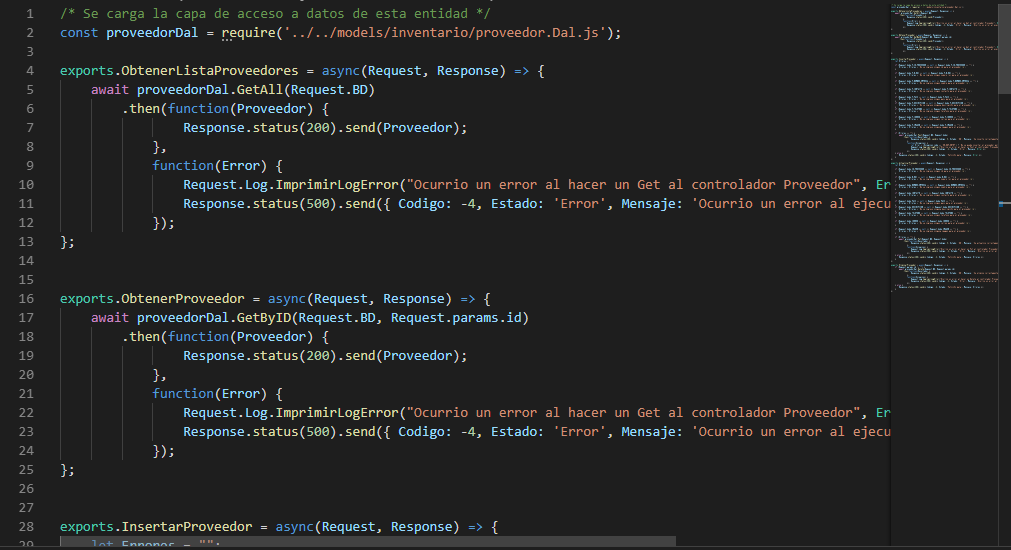
**Fig. 12: Petición de valores a la API de productos.**

**Fig. 13: Petición de valores a la API de productos del Almacén.**

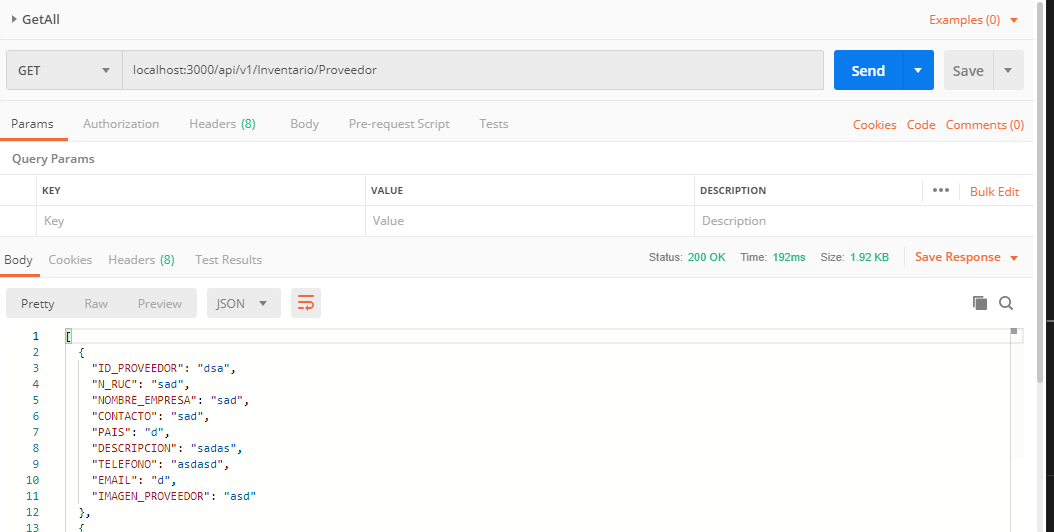
**Fig. 13: Retorno de la petición con valores del producto.**

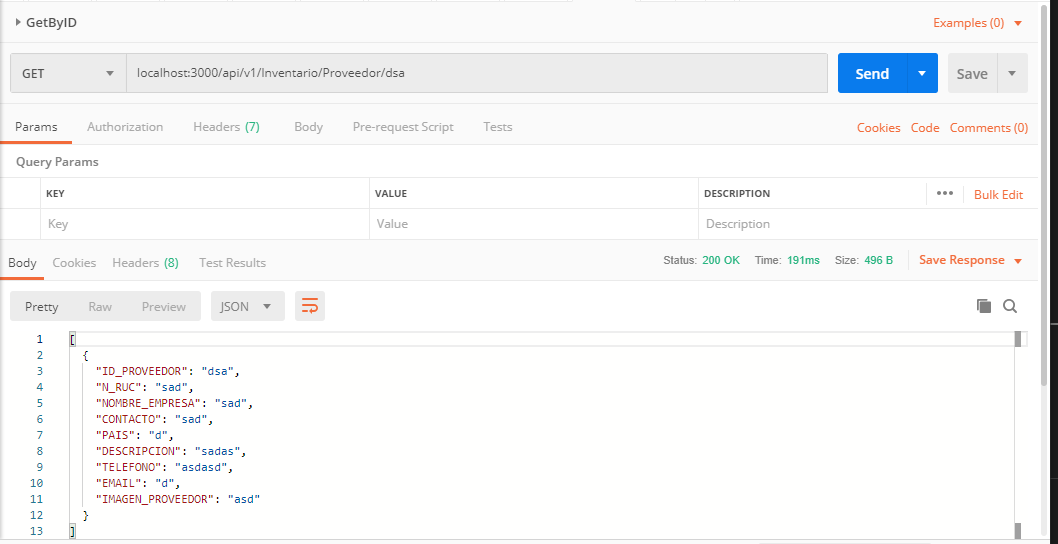
**3 – Tabla Proveedor**

**Fig. 14: Ruta del controlador de Proveedor**

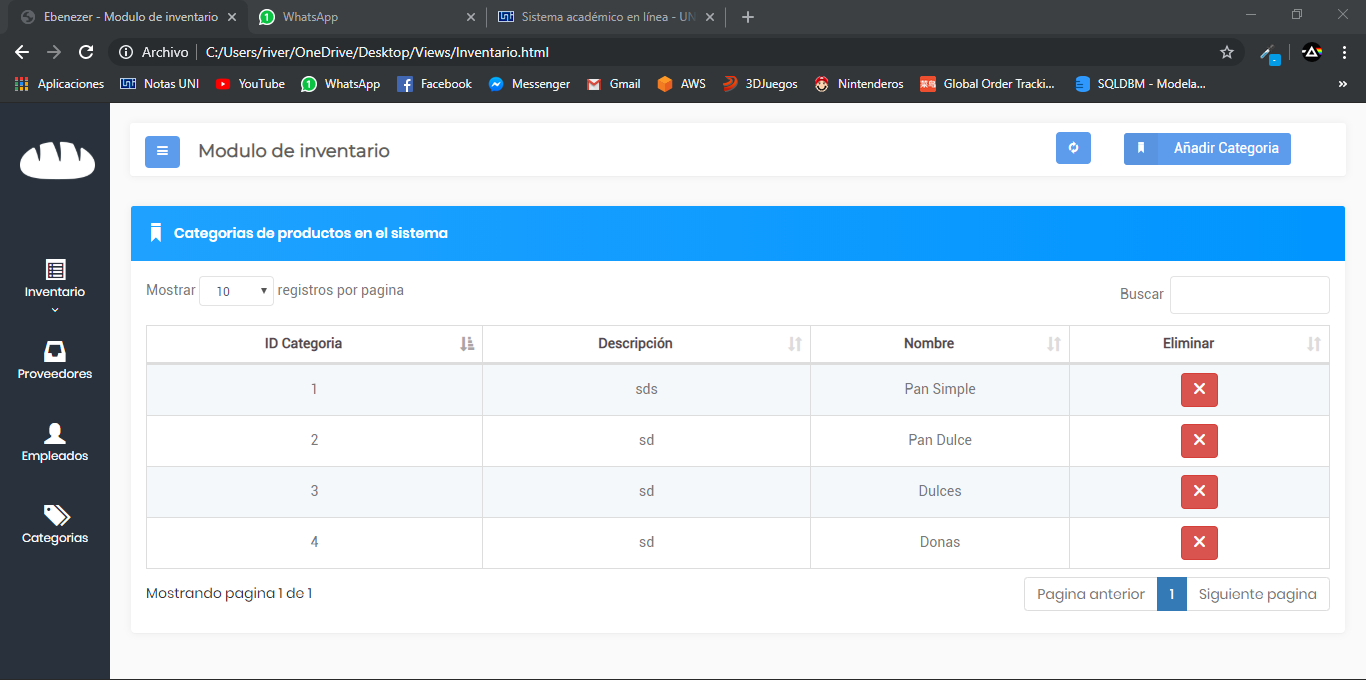
**Fig. 15: Validación del proceso de lista de proveedores.**

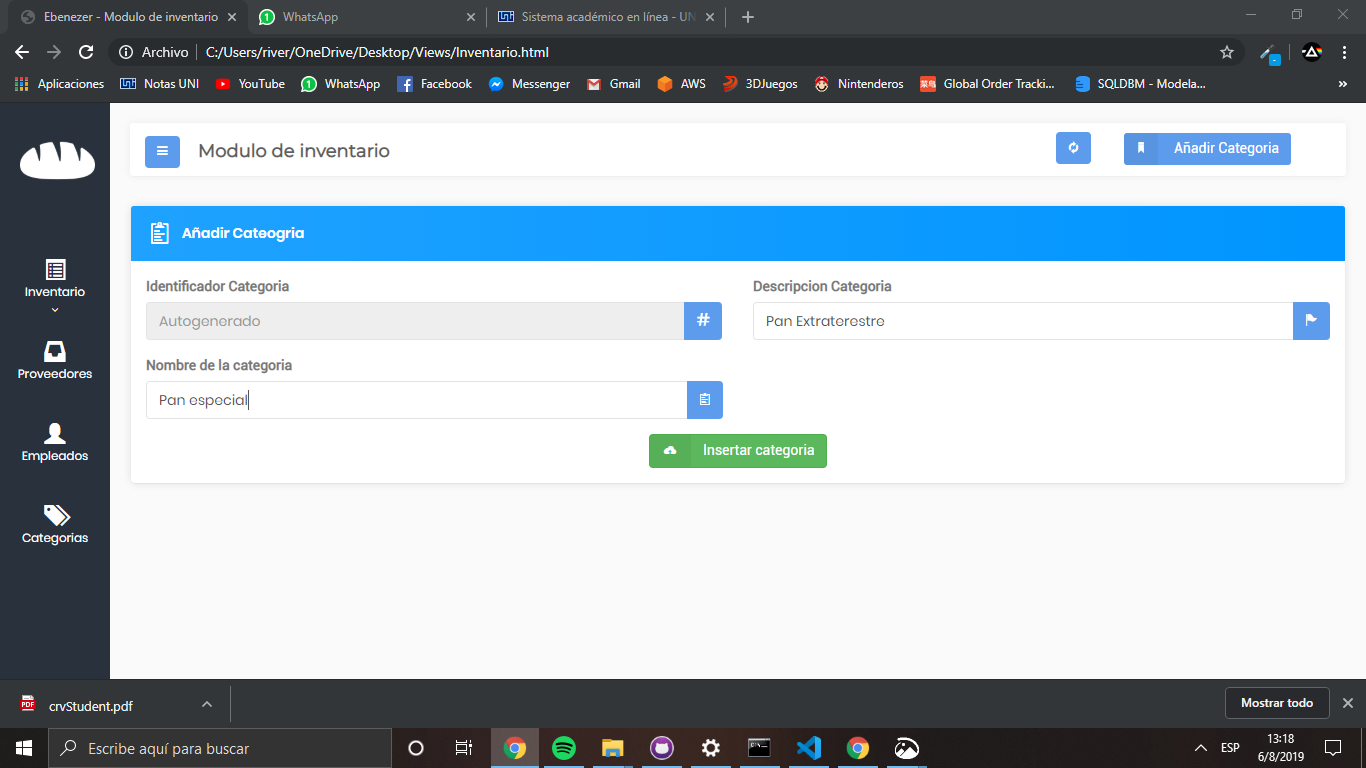
**Fig. 16: Validación para obtener ID de Proveedores.**

**Fig. 17: Petición a la API para información de Proveedores.**

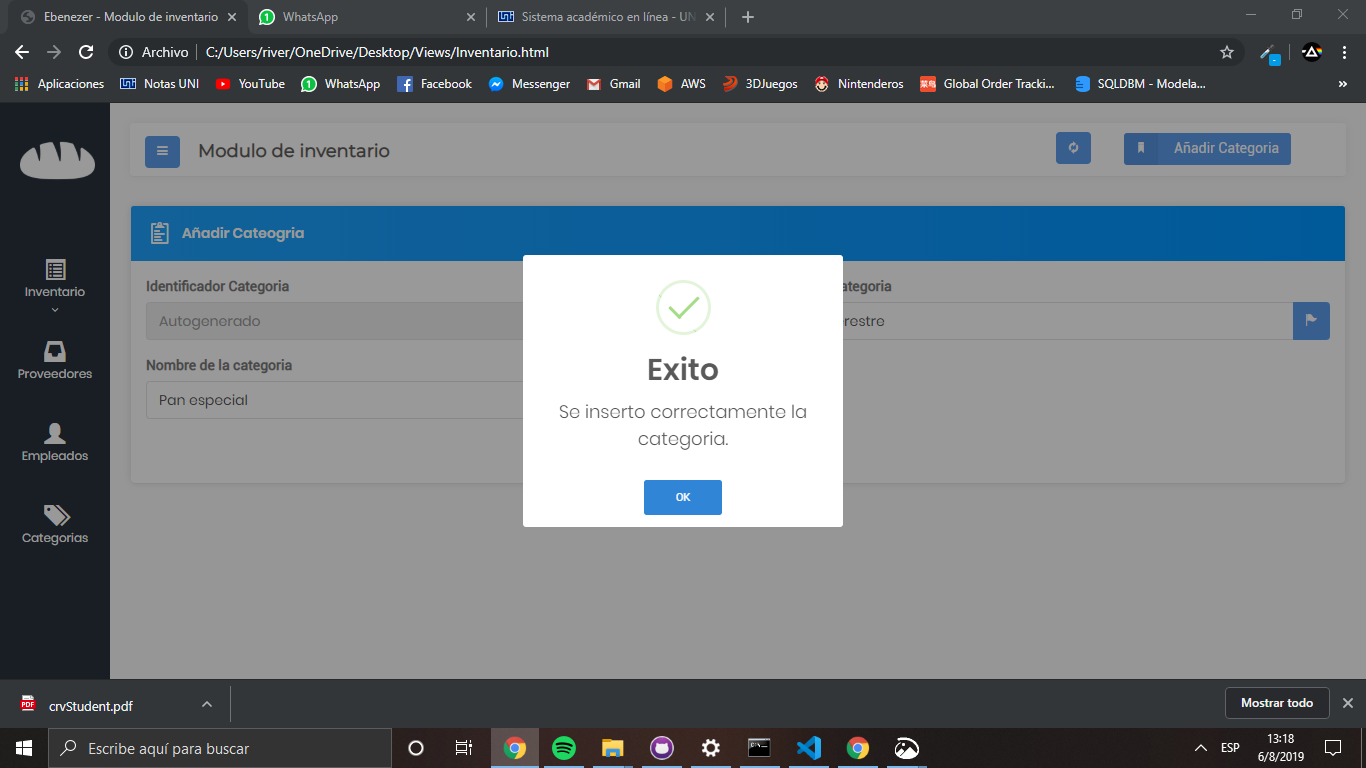
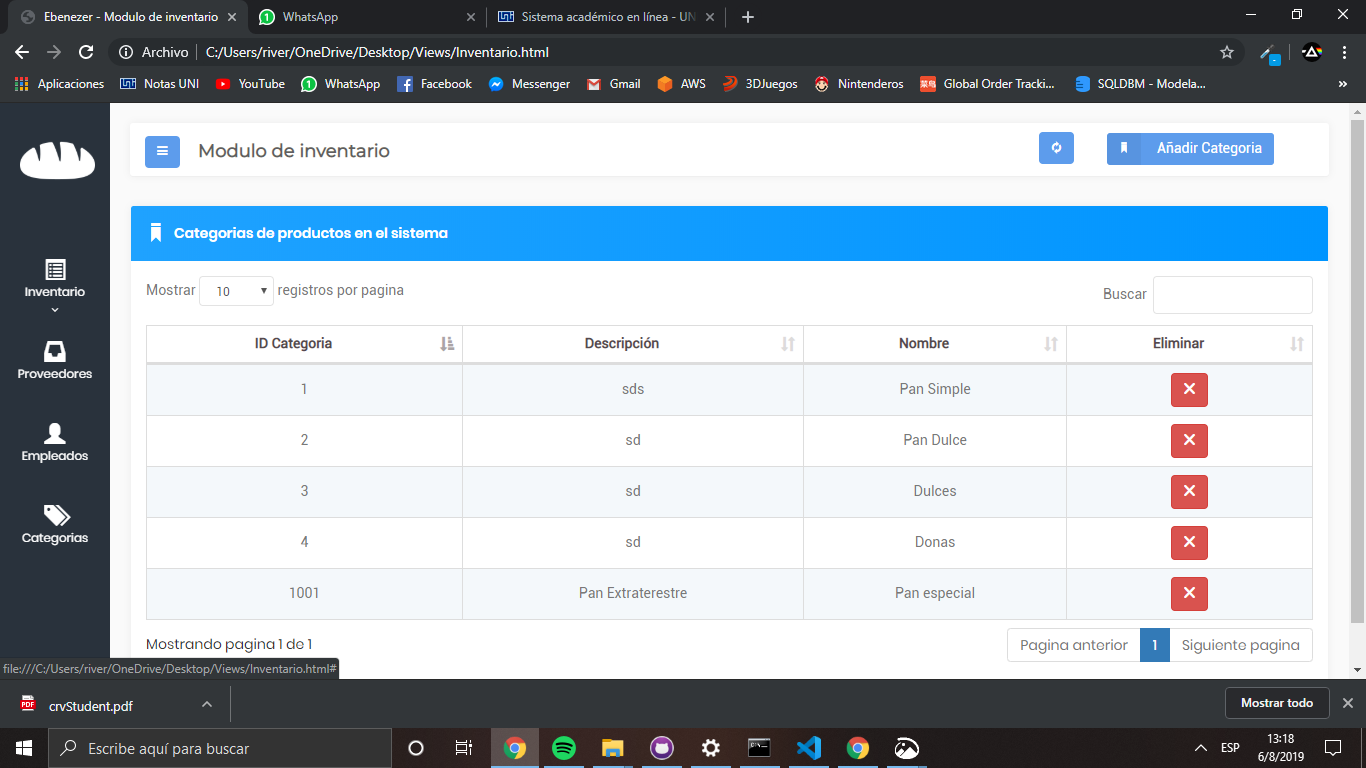
**Fig. 18: Retorno de la petición con valores del proveedor.**

**VII - Ejemplos de Caja Negra Aplicados al Sistema de Información del Proyecto de Curso**

1. **Categoría**

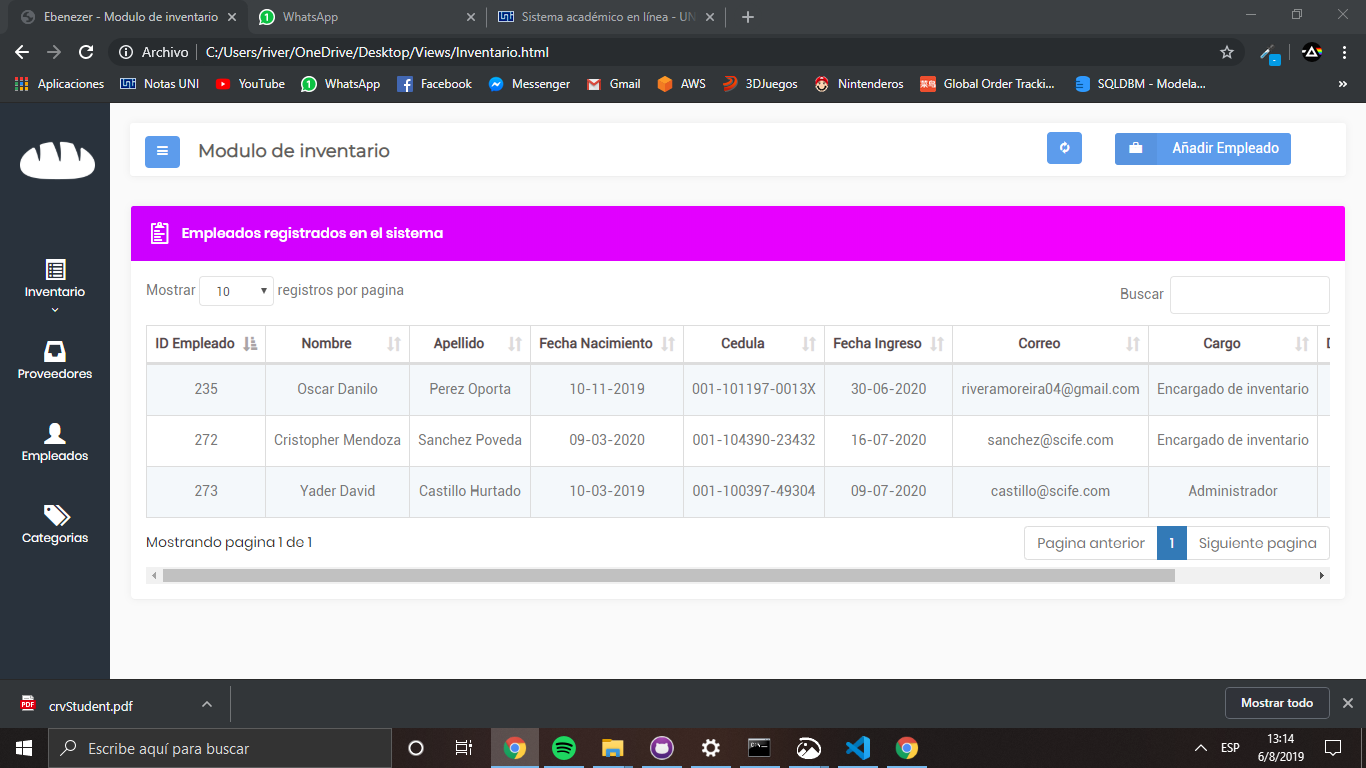
**Fig. 19: Interfaz con listado de productos por categoría.**

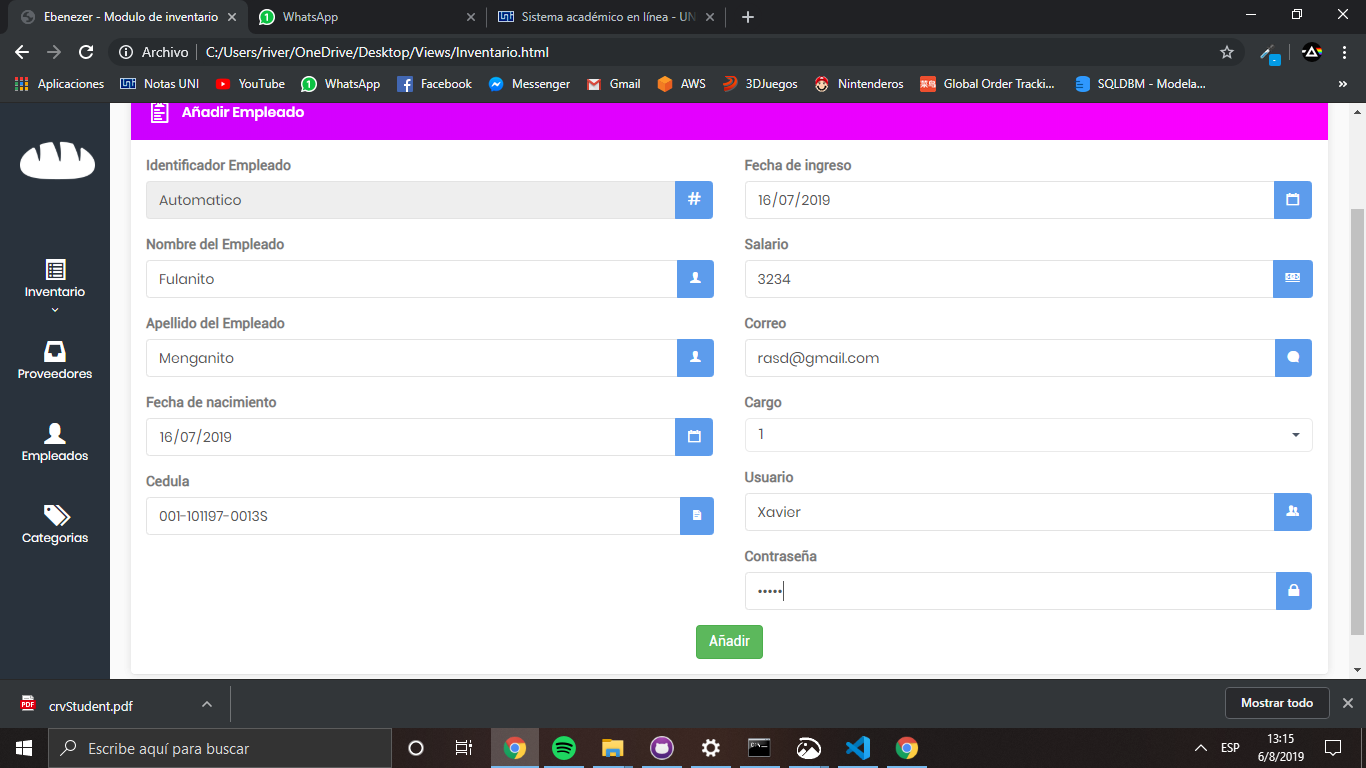
**Fig. 20: Añadir categoría de producto.**

 **Fig. 21: Categoría añadida exitosamente.**

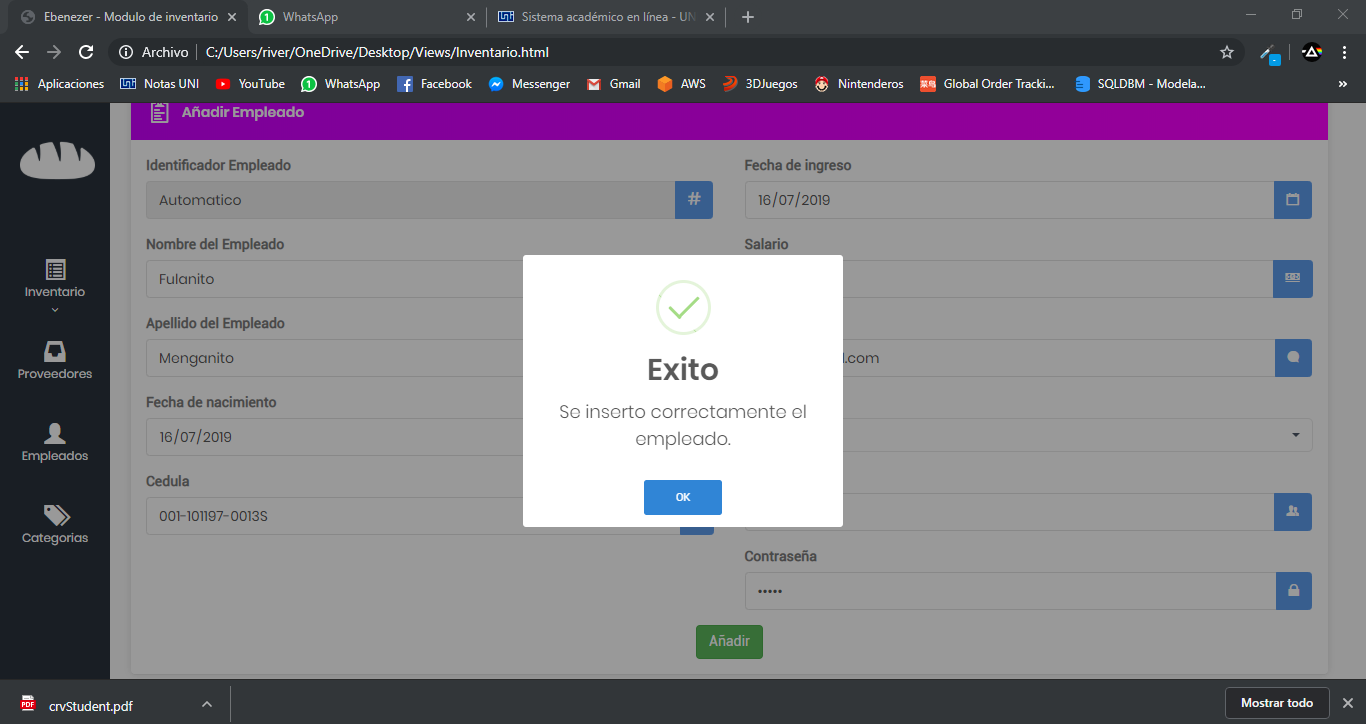
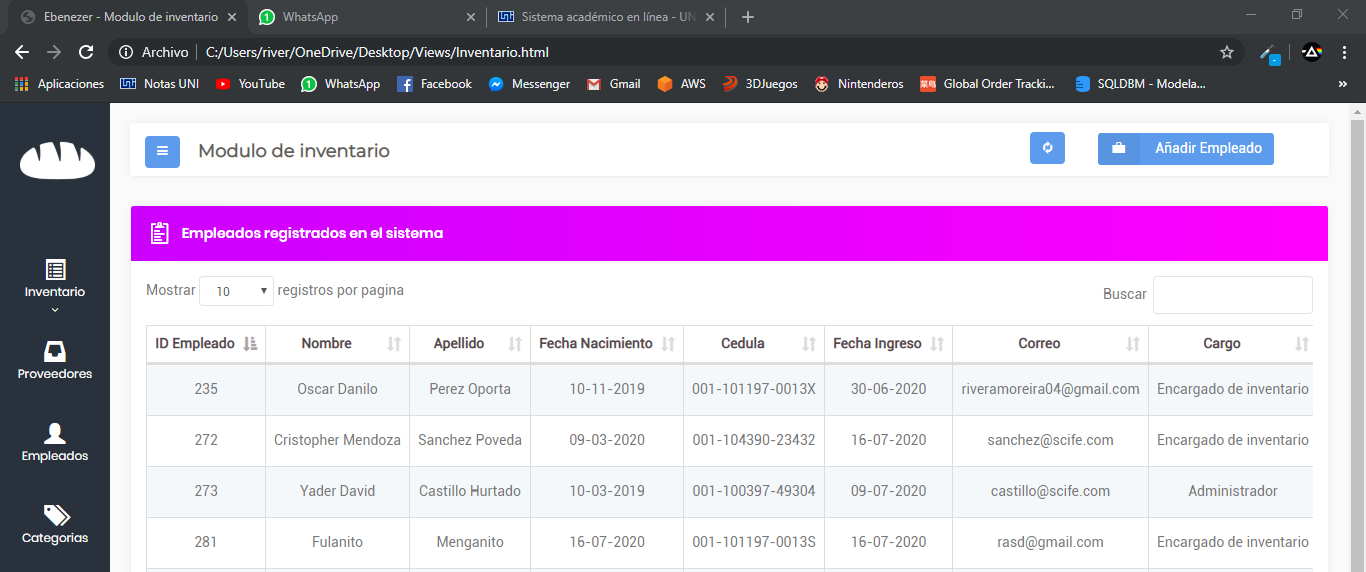
**Fig. 22: Listado de categoría con la nueva recién añadida.**

1. **Tabla Empleados**

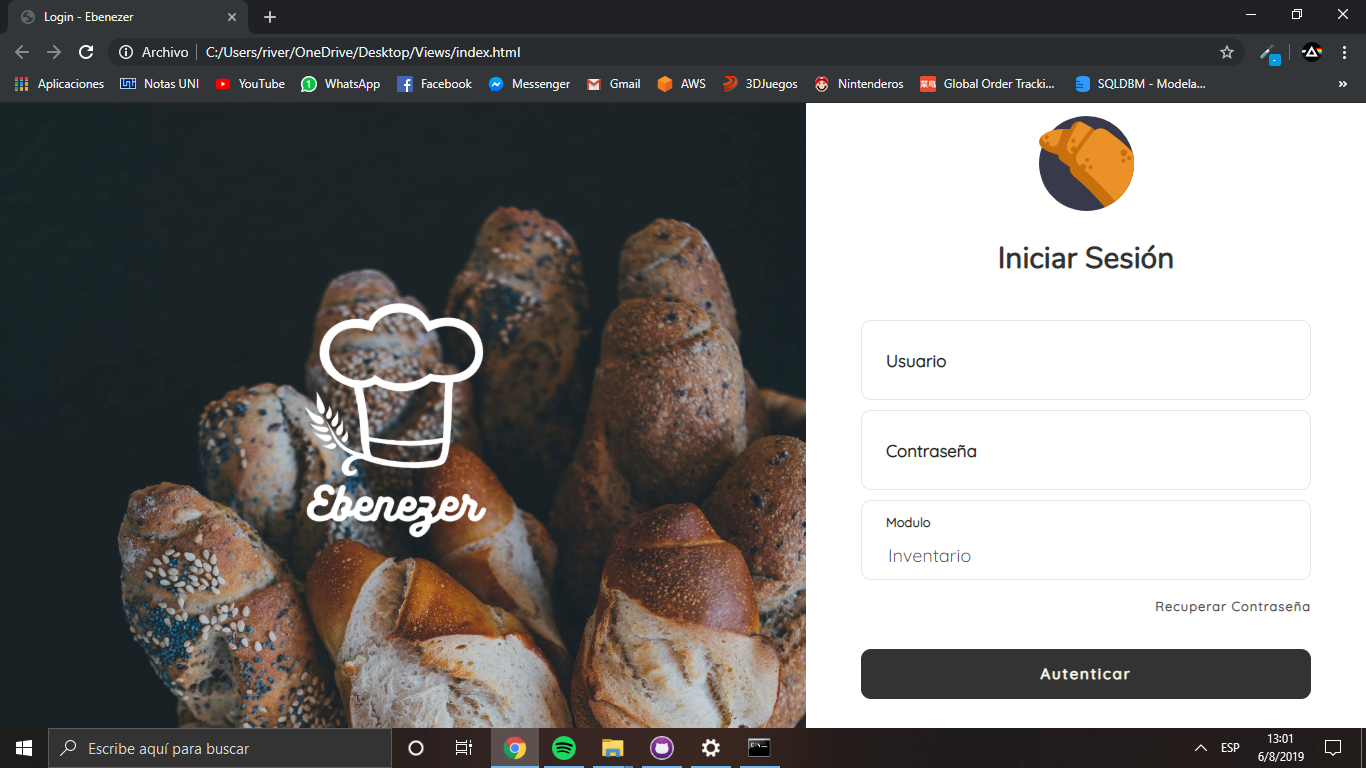


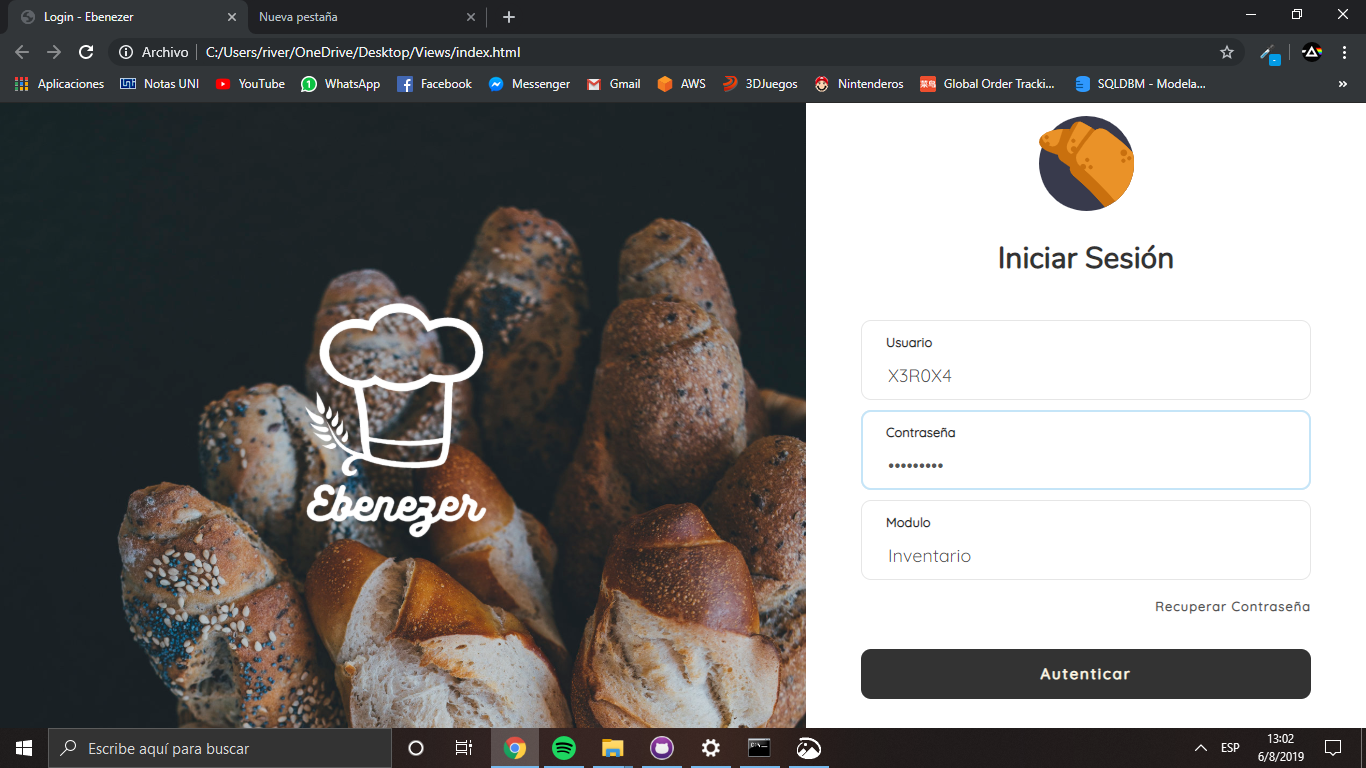
**Fig. 23: Listado de los empleados en la empresa.**

**Fig. 24: Añadiendo un empleado nuevo.**

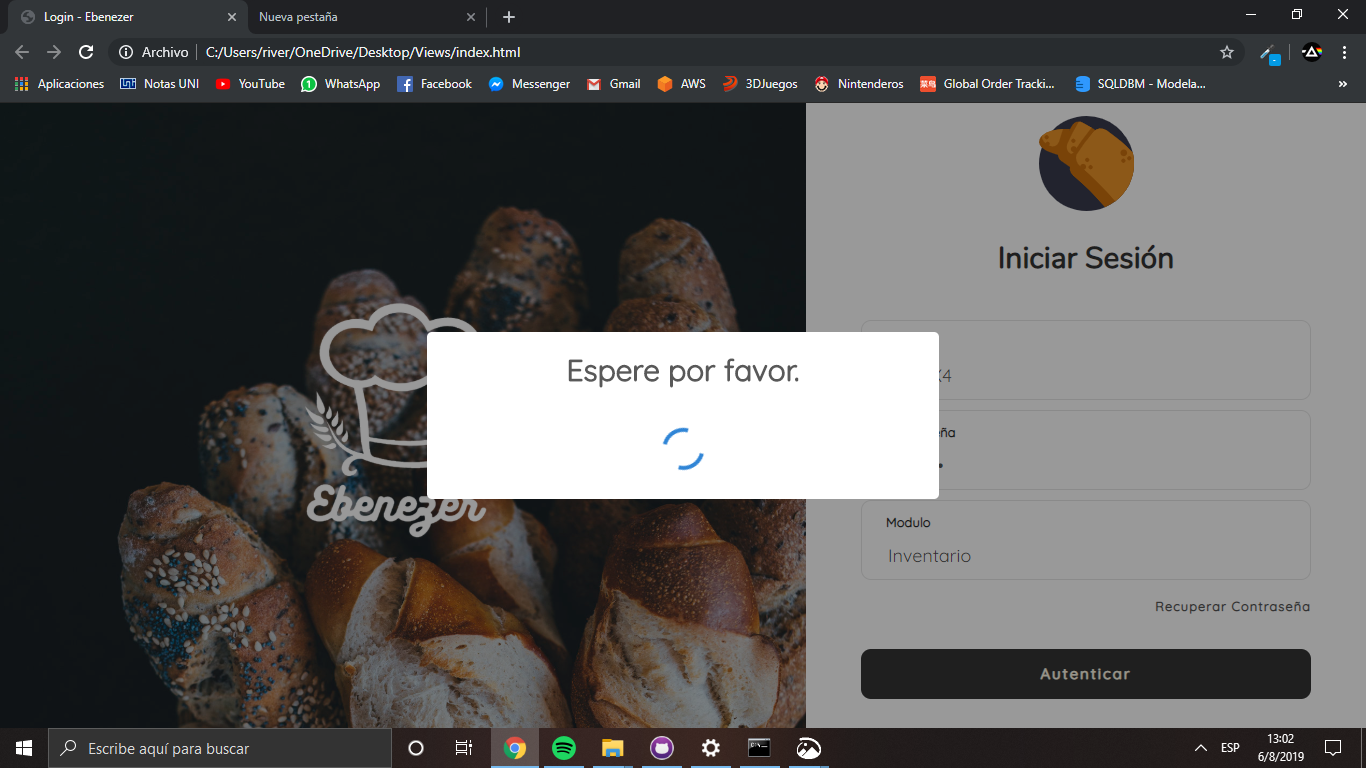
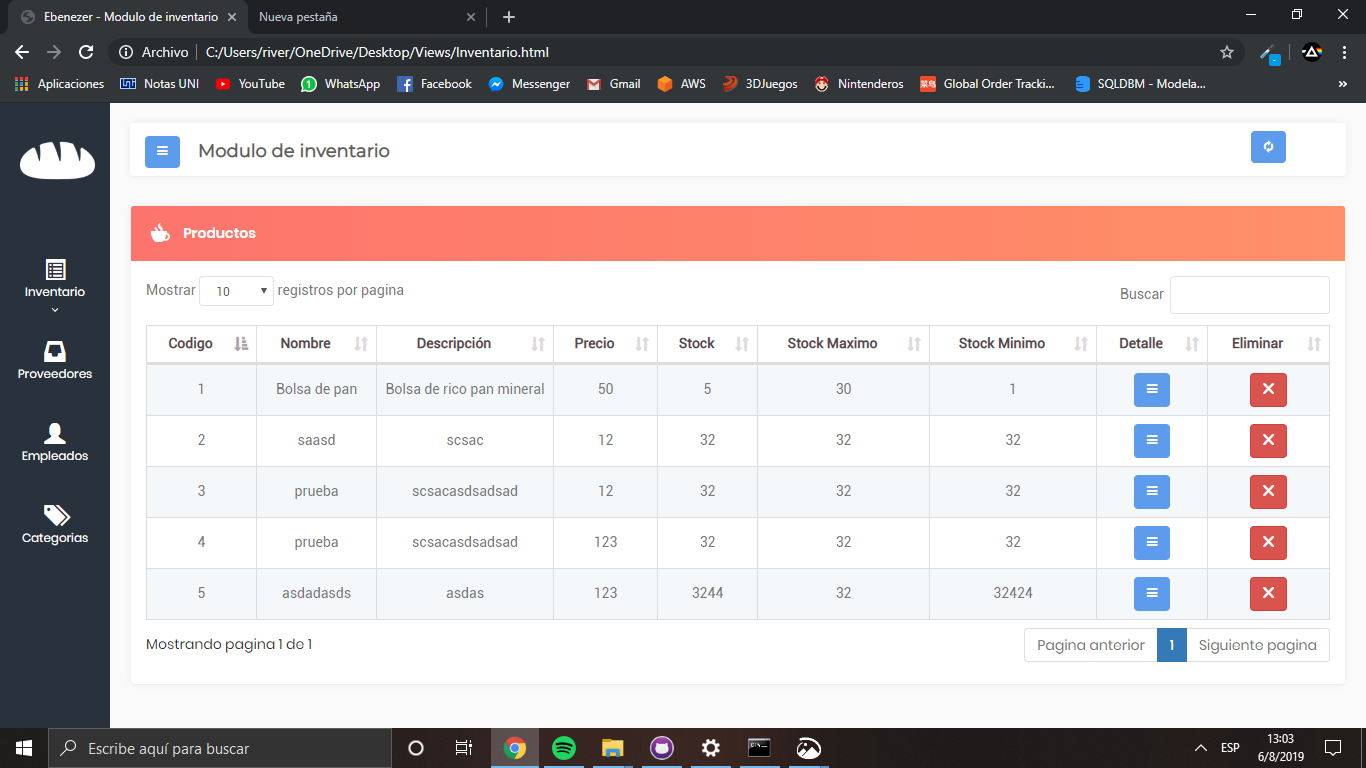
**Fig. 25: Empleado añadido exitosamente.**

**Fig. 26: Listado de empleados actualizada.**

1. **Acceso al Sistema**

**Fig. 27: Página principal de inicio de sesión.**

**Fig. 28: Campos para inserción de credenciales.**

**Fig. 29: Validación de credenciales de inicio de sesión.**

**Fig. 30: Página principal cargada exitosamente.**

**IX - Bibliografía**

<https://www.buenastareas.com/ensayos/Pruebas-De-Caja-Negra/1477804.html>

<https://www.scribd.com/document/49666181/Pruebas-Con-Ejemplos>

<https://html.rincondelvago.com/metodos-de-prueba-caja-de-pandora.html?url=metodos-de-prueba-caja-de-pandora>

<https://www.ecured.cu/Pruebas_de_caja_blanca>

<http://scielo.sld.cu/scielo.php?script=sci_arttext&pid=S1815-59362014000200003>

<https://www.yunbitsoftware.com/blog/2016/09/02/importancia-de-las-pruebas-en-desarrollo-de-software/>