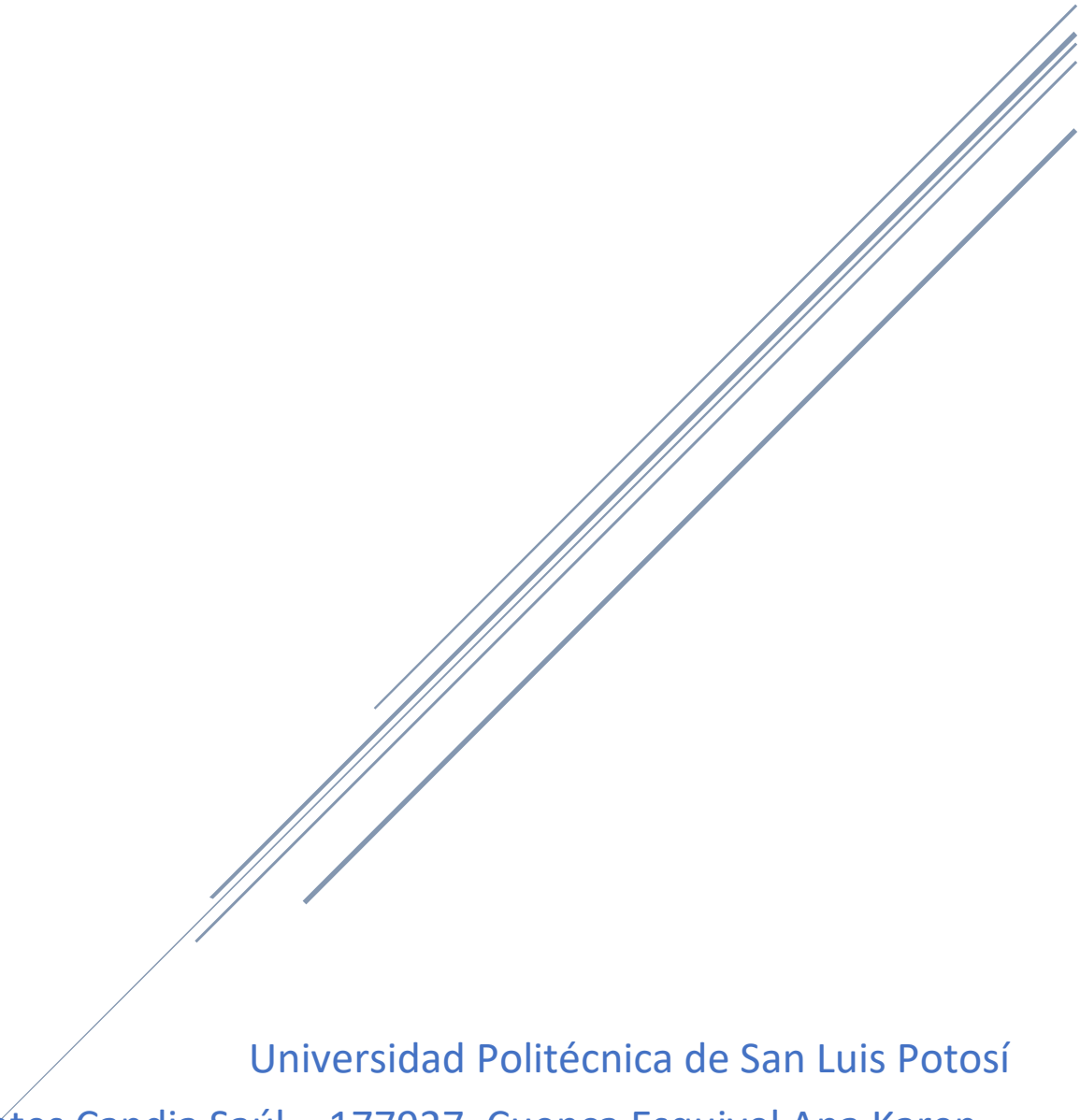
CASOS DE PRUEBA. TÉCNICA DE CAJA NEGRA.

Ingeniería de Software 1



Universidad Politécnica de San Luis Potosí

Cervantes Candia Saúl – 177927, Cuenca Esquivel Ana Karen – 177932, Méndez Pérez Aldo Giovanni - 172122

TÉCNICA DE CAJA NEGRA

La técnica de caja negra es una metodología de prueba que se centra en probar la funcionalidad de un sistema sin conocer su estructura interna. En lugar de examinar el código fuente o la lógica interna, esta técnica se enfoca en las entradas y salidas del sistema, tratándolo como una "caja negra" donde se observa su comportamiento externo sin tener en cuenta cómo funciona internamente.

Dentro de este documento se analizarán algunos casos de usos, puesto que ya se habían analizado en cuestión de código, excepto por estos métodos. El primer caso fue en el alta de pedidos porque no se estaban guardando correctamente los datos, el segundo es en la relación que se tiene en usuario y rol por poco conocimiento del uso de la API sequelize, mientras que los casos presentes en los métodos son variables plausibles a errores de entrada del usuario.

MÉTODOS

VALORES AL LÍMITE

La siguiente tabla presenta ejemplos de valores que se encuentran en los límites de las restricciones establecidas para las variables de contraseña y nombre en el sistema. Estos ejemplos muestran las clases válidas e inválidas, junto con representantes que ilustran los extremos de las restricciones especificadas.

Variables	Clases validas	Clases invalidas	Representantes
Contraseña	8 a 32 caracteres	<8 >32	squad301 – 8 caracteres LorenaLaPro777HernandezGutierrez – 32 caracteres Ratas3* - 7 caracteres ElmarcianoAncestralInmortal134340 – 33 caracteres
Nombre	Letras	Números y caracteres especiales	1aura Sebastián Iñaki Urie1

PARTICIÓN EQUIVALENTE.

La siguiente tabla muestra la segmentación de condiciones de entrada para las variables de contraseña y nombre del cliente, identificando las clases de equivalencia válidas e inválidas. Estas clases de equivalencia son fundamentales para el análisis de casos de prueba, ya que representan diferentes conjuntos de datos que deberían comportarse de manera similar en el sistema.

Condición de entrada	Clase de equivalencia valida	Clase de equivalencia invalida
Contraseña	Caracteres >8 y caracteres <32	Caracteres <8 y caracteres >32

Nombre Cliente	Letras	Números y caracteres especiales

TABLAS

Las siguientes tablas muestran las pruebas de clase negra ejecutadas dentro de la ejecución del sistema realizado hasta ahora para el proyecto. Mostrando detalles y bloques de código para su mayor comprensión.

Id	Acción	Requerimiento Funcional	Entrada	Proceso	Salida	Descripción
PF-01	Capturar pedidos	Otro	Cumplir el formato de la fecha y hora	Crear Pedido	2024-04-19 14:30:00.000Z	Al registrar un pedido, no se guarda correctamente los datos por el formato que se generaba de la fecha
PF-02	Ingresar usuario	Otro	Nombre, correo, contraseña y rol	Crear usuario	5 Juan JuanPerez05@gmail.com TeamoMamitañ/05 2 Cajero	Al guardar al usuario, el IdRol y el rol no se guardaban correctamente en la base de datos

	Prueba
Número	1
Prioridad	Alta Pedido
Descripción	No se guardaban los datos correctamente ya que la fecha no estaba de forma correcta
Archivo	BreadcrumbsProyecto_Waffles_React/node/controllers /PedidoController.js
Solución	Se modifico la asignación de la fecha

Código errado:

```
const fechaRegex = /^\d{4}-\d{2}-\d{2}$/; // Expresión regular para el formato YYYY-MM-DD
if (!fechaRegex.test(req.body.fecha)) {
  return res.status(400).json({ message: 'El formato de la fecha es incorrecto. Debe ser YYYY-MM-DD.' });
}
```

```
// Obtiene la fecha y hora actuales del servidor en formato ISO 8601
const fechaHoraActual = new Date().toISOString(); // "2024-04-12T14:30:00.000Z"
// Divide la cadena de fecha y hora en sus partes correspondientes
const [fechaActual, horaActual] = fechaHoraActual.split('T'); // ["2024-04-12", "14:30:00.000Z"]
```

Código Correcto:

```
// Obtener la fecha y hora actual en la zona horaria de Ciudad de México
const fechaHoraActual = new Date().toLocaleString('es-MX', { timeZone: 'America/Mexico_City' });
// Divide la cadena de fecha y hora en sus partes correspondientes
const [fechaActual, horaActual] = fechaHoraActual.split(' ');
```

	Prueba
Número	2
Prioridad	Crear Usuario
Descripción	Al ingresar un usuario, no se guardaba correctamente los datos en la base de datos ocasionando problemas por falta de existencia
Archivo	Proyecto_Waffles_React/node/models /UsuarioModel.js
Solución	Se modificó la instrucción “belongsTo”

Código errado:

```
UsuarioModel.belongsToMany(RolModel, {
  as: 'roles', // Alias for the association
  foreignKey: 'idRol', // Foreign key in the Usuario table
  onDelete: 'CASCADE', // Opcional: elimina automáticamente los roles asociados cuando se elimina un usuario
  through: 'usuario_rol' // Join table for the many-to-many relationship
});
```

Código Correcto:

```
UsuarioModel.belongsTo(RolModel, {
    foreignKey: 'idRol', // Nombre de la columna en UsuarioModel que actúa como clave foránea
    as: 'rol', // Alias para la asociación, opcional
});
```