

**APLICACIÓN WEB DE ESCRITORIO DE GESTION DE INVENTARIO A TRAVES DE
LA HERRAMIENTA ELECTRONJS PARA LA FABRICA PREFABRICADOS MARLON
STEFAN**

**Autor:
LUIS DAVID CELIS CAICEDO**



Directora: Dra. Rocío De Belén Contreras Manrique

Universidad de Pamplona

Facultad de Ingenierías y Arquitectura

Departamento de Ingenierías Electrónica, Eléctrica, Sistemas y Telecomunicaciones, EEST

Ingeniería de Sistemas

Villa del Rosario, Colombia.

2022

DEDICATORIA

Quiero dedicar el siguiente trabajo por encima de todo a Dios, a mis padres Luis Felipe Celis Andrade y Gloria Caicedo Contreras que gracias a su esfuerzo diario me brindaron la oportunidad de tener una educación de alta calidad, por el apoyo incondicional que me brindaron y siempre han estado cuando los he necesitado, y por siempre creer en mí.

A mis abuelos, Darío Gustavo Celis y María Luzmila Andrade, los cuales me apoyaron para iniciar esta gran etapa de mi vida que estoy terminando y lamentablemente ya no están conmigo.

A mis hermanas, Daniela Liseth Celis Caicedo y Luz Mary Celis Caicedo por su apoyo incondicional y siempre poder contar con ellas cuando lo he necesitado.

AGRADECIMIENTOS

Agradezco a Dios, que ha estado presente durante esta gran etapa de mi vida y permitirme terminarla con éxito, a mis abuelos, mis padres y hermanas.

Agradezco a mis compañeros y amigos con los cuales compartí tanto momentos juntos, nos apoyamos mutuamente en los momentos más difíciles y gracias sus consejos y apoyo estoy culminando esta etapa de mi vida.

Agradezco en especial a Dimar David Galvan Bayona, el cual pude conocer cuando comencé mi vida universitaria, desde entonces se convirtió en uno de mis mejores amigos, nos apoyamos mutuamente en todas las materias que veíamos juntos, compartimos victorias como derrotas académicas de las cuales tengo muy buenos recuerdos, le agradezco su apoyo incondicional hacia a mí, por apoyarme en los momentos más difíciles tanto en la vida académica como en lo personal, y sobre todo por su gran calidez como persona.

Agradezco en especial a Jorge Sneyder Rodriguez Rubio, con quien desarrolle una gran amistad desde el momento en que nos conocimos, desde un principio se mostró como la gran persona que es, el cual en momentos de necesidad siempre ha estado para apoyarme sin necesidad de pedir su ayuda, y cuando lo he hecho nunca he obtenido un no por respuesta, agradezco sus grandes valores y cualidades como persona.

Agradezco en especial a Jose Camilo Guerrero Marquez, el cual conozco desde hace más de 8 años, con quien tengo los mejores recuerdos de mi etapa escolar los cuales conservo con mucha alegría y nostalgia, a pesar de que no pudimos estudiar juntos en la misma

universidad nunca perdimos contacto, le doy gracias porque siempre ha estado para escuchar mis problemas, por todos los días que nos reuníamos en discord para jugar y olvidarnos de los problemas del mundo real, por esas charlas por horas que me servían como terapia, le doy gracias por ser el hermano que nunca tuve.

RESUMEN

En el presente trabajo de grado se realizó una aplicación web de escritorio, la cual tiene como propósito la mejora del proceso de manejo de inventario de la fábrica Prefabricados Marlon – Stefan. Por lo tanto, el desarrollo comenzó debido a una necesidad por parte de la persona encargada de llevar el control del inventario, la cual manifestó que tenía inconvenientes para llevar dicho control, en particular la gestión de los productos y materiales que ingresaban o egresaban del lugar de almacenamiento. Para lograr el desarrollo de la aplicación, primeramente se identificaron los requerimientos funcionales, no funcionales y pseudo requerimientos, con los cuales se realizaron los distintos diagramas que conforman la arquitectura del software, luego se realizaron los diseños de la aplicación los cuales sirvieron como base para la codificación del frontend en Angular, el cual fue configurado con electronjs para poder tener la aplicación web en forma de aplicación de escritorio, el backend de la aplicación fue realizado en Nodejs en el cual se aplicó la arquitectura hexagonal. Este trabajo de grado fue llevado a cabo con la metodología ágil de desarrollo Scrum la cual permitió gestionar el desarrollo de la aplicación. Por último, se realizaron pruebas unitarias en el backend de la aplicación para la comprobación del correcto funcionamiento del mismo, permitiendo así finalizar el desarrollo de la aplicación web de escritorio.

Palabras claves: Gestión de inventario, Electronjs, Aplicación de escritorio, Aplicación web, Scrum.

ABSTRACT

In the present degree work, a desktop web application was made, which has the purpose of improving the inventory management process of the company Prefabricados Marlon - Stefan. Therefore, the development began due to a need on the part of the person in charge of keeping control of the inventory, who stated that he had problems to keep said control, in particular the management of the products and materials that entered or left the place. storage. To achieve the development of the application, firstly the functional, non-functional and pseudo requirements were identified, with which the different diagrams that make up the software architecture were made, then the application designs were made, which served as the basis for the coding of the frontend in Angular, which was configured with electronjs to be able to have the web application in the form of a desktop application, the backend of the application was made in Nodejs in which the hexagonal architecture was applied. This degree work was carried out with the agile Scrum development methodology which allowed managing the development of the application. Finally, unit tests were carried out on the backend of the application to verify its correct operation, thus allowing the development of the desktop web application to be completed.

Keywords: Inventory Management, Electronjs, Desktop Application, Web Application, Scrum.

TABLA DE CONTENIDO

Contenido

CAPÍTULO 1. PLANTEAMIENTO DEL PROBLEMA	23
1.1. PLANTEAMIENTO DEL PROBLEMA	23
1.2. FORMULACIÓN DEL PROBLEMA	24
1.3. JUSTIFICACIÓN	25
1.4. OBJETIVOS	26
1.4.1. Objetivo General	26
1.4.2. Objetivos Específicos	26
1.4.3. Acotaciones	26
CAPÍTULO 2. MARCO TEÓRICO	27
2.1. ANTECEDENTES	27
2.1.1. Antecedentes Internacional	27
2.1.2. Antecedentes Nacional	28
2.1.3. Antecedentes Regional	29
2.2. MARCO TEÓRICO	31
2.2.1. FRAMEWORK	31
2.2.1.1. Ventajas de los Frameworks	31
2.2.1.2. Ejemplos de Frameworks	32

2.2.2. APLICACIÓN WEB	32
2.2.3. ELECTRONJS	33
2.2.4. ANGULAR	34
2.2.5. NODEJS	35
2.2.6. MODELO-VISTA-CONTROLADOR (MVC)	35
2.2.7. MVVM	37
2.2.8. ARQUITECTURA HEXAGONAL	38
2.2.9. METODOLOGÍA ÁGIL DE DESARROLLO SCRUM	39
2.2.9.1. Roles Scrum	39
2.2.9.2. Artefactos de Scrum	40
2.2.10. API (Application Programming Interface)	40
2.2.11. POSTGRESQL	41
2.2.12. PRUEBAS UNITARIAS	42
2.2.12.1. Motivos para realizar un test unitario	43
2.2.12.2. Las 3 A's del unit testing	43
2.3. MARCO LEGAL	44
2.3.1. Ley Estatutaria 1581 de 2012	44
2.3.2. Ley 1341 TIC en Colombia	44
CAPITULO 3. DESARROLLO DEL SISTEMA	46
3.1. Identificar los requerimientos funcionales y no funcionales suministrados por el cliente para el desarrollo de la aplicación web de escritorio.	46

3.1.1. Requerimientos Funcionales -----	46
3.1.2. Requerimientos no funcionales -----	50
3.1.3. Pseudo Requerimientos-----	51
3.1.4. Diagrama de Casos de Uso-----	53
3.1.5. Diagrama de Actividades -----	54
3.2. Analizar las diferentes metodologías de desarrollo a través de un cuadro comparativo para la aplicación web de escritorio de gestión de inventario. -----	65
3.2.1. SCRUM-----	65
3.2.1.1. Artefactos de scrum.....	66
3.2.2. XP (Extreme Programming) -----	67
3.2.2.1. Valores, principios y prácticas	67
3.2.3. Comparación entre SCRUM y XP -----	69
3.3. Diseñar la aplicación web de escritorio mediante la plataforma Figma para la gestión de inventario de la fábrica. -----	71
3.3.1. Modelo relacional base de datos -----	71
3.3.2. Diccionario base de datos -----	73
3.3.3. Wireframes -----	77
CAPITULO 4. METODOLOGÍA PARA DESARROLLO AGIL -----	89
4.1. Roles -----	89
4.2. Product Backlog-----	89
4.3. Sprint Backlog-----	90

CAPITULO 5. DESARROLLO DE LA APLICACIÓN WEB DE ESCRITORIO ----- 95

5.1. Implementar la aplicación web de escritorio para la gestión de inventario de la fábrica. ----- 95

5.1.1. Arquitectura hexagonal para el backend 95

5.1.2. Desarrollo del backend..... 97

5.1.3. Desarrollo del frontend 99

5.1.3.1. Creación del proyecto..... 99

5.1.3.2. Creación y configuración de electronjs 100

5.1.4. Interfaces de usuario 101

5.1.4.1. Inicio de sesión 102

5.1.4.2. Menú principal (Administrador) 102

5.1.4.3. Menú principal (Operador) 103

5.1.4.4. Inventario 104

5.1.4.5. Operadores 107

5.1.4.6. Clientes 108

5.1.4.7. Operaciones..... 111

5.1.4.7.1. Ingresos..... 112

5.1.4.7.2. Egresos 115

5.1.5. Despliegue de la base datos 118

5.1.6. Creación de la aplicación de escritorio ejecutable con electron 120

5.2. Realizar pruebas a la aplicación web de escritorio para la gestión de inventario.-122

5.2.1. Instalación y configuración de las dependencias 122

5.2.2. Como se realizan las pruebas 123

5.2.3. Realización de pruebas..... 124

5.2.3.1. Pruebas del servicio usuario	125
5.2.3.2. Pruebas del servicio producto	127
5.2.3.3. Pruebas del servicio material.....	128
5.2.3.4. Pruebas del servicio cliente	129
5.2.3.5. Pruebas del servicio empresa.....	129
5.2.3.6. Pruebas del servicio ingresos productos	130
5.2.3.7. Pruebas del servicio ingresos materiales	131
5.2.3.8. Pruebas del servicio egresos productos	132
5.2.3.9. Pruebas del servicio egresos materiales.....	133
6. CONCLUSIONES -----	135
7. REFERENCIAS BIBLIOGRÁFICAS -----	136
ANEXOS-----	140
ANEXO A. MANUAL DE USUARIO -----	140
ANEXO B. ENCUESTA VIABILIDAD-----	160
ANEXO C. CARTA DE APROBACION POR PARTE DE LA FABRICA PREFABRICADOS MARLON STEFAN -----	164

Lista de Ilustraciones

Ilustración 1. Scrum sprint - Tomado de https://www.atlassian.com/es/agile/scrum/sprints ...	39
Ilustración 2. Diagrama de casos de uso.....	53
Ilustración 3. Diagrama de actividad (Inicio de sesión)	55
Ilustración 4. Diagrama de actividad (Registro operador)	56
Ilustración 5. Diagrama de actividad (Registro cliente)	57
Ilustración 6. Diagrama de actividad (Registro empresa)	58
Ilustración 7. Diagrama de actividad (Registro producto)	59
Ilustración 8. Diagrama de actividad (Registro material).....	60
Ilustración 9. Diagrama de actividad (Registro ingreso de producto).....	61
Ilustración 10. Diagrama de actividad (Registro egreso de producto)	62
Ilustración 11. Diagrama de actividad (Registro ingreso de material)	63
Ilustración 12. Diagrama de actividad (Registro egreso de material)	64
Ilustración 13. Ciclo de sprint Tomado de https://www.atlassian.com/es/agile/scrum	66
Ilustración 14. Modelo relacional.....	72
Ilustración 15. Vista inicio de sesión.....	78
Ilustración 16. Vista menú principal (Administrador).....	78
Ilustración 17. Vista inventario (productos)	79
Ilustración 18. Vista inventario (materiales)	79
Ilustración 19. Vista operadores (visualización y registro)	80
Ilustración 20. Vista de clientes (Visualización y registro).....	80
Ilustración 21. Vista de empresas (Visualización y registro)	81
Ilustración 22. Vista menú de operaciones.....	81

Ilustración 23. Vista de ingresos productos (Visualización)	82
Ilustración 24. Vista de ingreso productos (Registro)	83
Ilustración 25. Vista de ingreso materiales (Visualización)	84
Ilustración 26. Visualización de ingreso materiales (Registro)	84
Ilustración 27. Vista de egreso productos (Visualización)	85
Ilustración 28. Vista de egreso productos (Registro)	86
Ilustración 29. Vista de egreso materiales (Visualización)	87
Ilustración 30. Vista de egreso materiales (Registro)	88
Ilustración 31. Arquitectura hexagonal - Tomado de https://medium.com/@edusalguero/arquitectura-hexagonal-59834bb44b7f	95
Ilustración 32. Flujo de comunicación del backend – Elaboración propia	96
Ilustración 33. Estructura del backend - Elaboración propia	97
Ilustración 34. Importaciones del controlador - Elaboración propia	98
Ilustración 35. Endpoints del controlador - Elaboración propia	98
Ilustración 36. Importaciones del servicio usuario - Elaboración propia	98
Ilustración 37. Método para obtener los usuarios de tipo operador - Elaboración propia	99
Ilustración 38. Comando para crear un proyecto de Angular - Elaboración propia	99
Ilustración 39. Comando para instalar la dependencia de electronjs - Elaboración propia	100
Ilustración 40. Main.js que ejecuta la aplicación de escritorio - Elaboración propia	100
Ilustración 41. Script para la ejecución de electrón - Elaboración propia	101
Ilustración 42. Index.html del proyecto en Angular - Elaboración propia	101
Ilustración 43. Inicio de sesión - Elaboración propia	102
Ilustración 44. Menú principal (Administrador)	103
Ilustración 45. Menú principal (Operador)	104

Ilustración 46. Inventario (Productos) - Elaboración propia	105
Ilustración 47. Modal registro de producto - Elaboración propia.....	105
Ilustración 48. Lista para ver productos o materiales - Elaboración propia	106
Ilustración 49. Inventario (materiales) – Elaboración propia	106
Ilustración 50. Modal registro de material - Elaboración propia.....	107
Ilustración 51. Visualizar operadores - Elaboración propia	108
Ilustración 52. Modal registro de operador - Elaboración propia.....	108
Ilustración 53. Lista para ver clientes o empresas - Elaboración propia	109
Ilustración 54. Visualizar clientes - Elaboración propia.....	109
Ilustración 55. Modal registro cliente - Elaboración propia.....	110
Ilustración 56. Visualizar empresas - Elaboración propia	110
Ilustración 57. Modal registro empresa - Elaboración propia	111
Ilustración 58. Interfaz operaciones - Elaboración propia.....	112
Ilustración 59. Visualización ingresos productos - Elaboración propia	113
Ilustración 60. Interfaz registro ingreso de productos - Elaboración propia	114
Ilustración 61. Lista desplegable ingresos de productos y materiales - Elaboración propia ..	114
Ilustración 62. Visualización ingresos materiales - Elaboración propia	115
Ilustración 63. Interfaz registro ingreso de materiales - Elaboración propia.....	115
Ilustración 64. Visualización egresos productos - Elaboración propia	116
Ilustración 65. Interfaz registro egreso de productos - Elaboración propia.....	117
Ilustración 66. . Lista desplegable egresos de productos y materiales - Elaboración propia .	117
Ilustración 67. Visualización egresos materiales - Elaboración propia.....	118
Ilustración 68. Interfaz registro egreso de materiales - Elaboración propia.....	118
Ilustración 69. Instancia de base de datos en ElephantSQL	119

Ilustración 70. Interfaz para insertar código SQL en el hosting	119
Ilustración 71. Tablas de la base de datos en ElephantSQL	119
Ilustración 72. Prueba endpoint obtener usuarios	125
Ilustración 73. Prueba endpoint obtener usuario por cedula	125
Ilustración 74. Prueba endpoint crear usuario	125
Ilustración 75. Prueba endpoint eliminar usuario	126
Ilustración 76. Prueba endpoint iniciar sesión	126
Ilustración 77. Resultado de las pruebas del servicio usuario	126
Ilustración 78. Prueba endpoint obtener productos	127
Ilustración 79. Prueba endpoint obtener producto por id	127
Ilustración 80. Prueba endpoint obtener productos filtrados por calzado	127
Ilustración 81. Resultados de las pruebas del servicio producto	127
Ilustración 82. Prueba endpoint obtener materiales	128
Ilustración 83. Prueba endpoint obtener material por código	128
Ilustración 84. Prueba endpoint obtener materiales filtrados por color	128
Ilustración 85. Resultados de las pruebas del servicio material	128
Ilustración 86. Prueba endpoint obtener clientes	129
Ilustración 87. Prueba endpoint obtener cliente por cedula	129
Ilustración 88. Resultados de las pruebas del servicio cliente	129
Ilustración 89. Prueba endpoint obtener empresas	129
Ilustración 90. Prueba endpoint obtener empresa por identificación	130
Ilustración 91. Resultados de las pruebas del servicio empresa	130
Ilustración 92. Prueba endpoint obtener ingresos de productos	130
Ilustración 93. Prueba endpoint obtener ingreso por su id	130

Ilustración 94. Prueba endpoint obtener ingreso por usuario	131
Ilustración 95. Resultados de las pruebas del servicio ingresos productos	131
Ilustración 96. Prueba endpoint obtener ingresos de materiales	131
Ilustración 97. Prueba endpoint obtener ingreso material por id.....	131
Ilustración 98. Prueba endpoint obtener ingreso material por usuario	132
Ilustración 99. Resultados de las pruebas del servicio ingresos materiales.....	132
Ilustración 100. Prueba endpoint obtener egresos de productos	132
Ilustración 101. Prueba endpoint obtener egresos de productos por id	132
Ilustración 102. Prueba endpoint obtener egresos de productos por usuario	133
Ilustración 103. Resultados de las pruebas del servicio egresos productos.....	133
Ilustración 104. Prueba endpoint obtener egresos de materiales.....	133
Ilustración 105. Prueba endpoint obtener egreso de material por su id.....	133
Ilustración 106. Prueba endpoint obtener egresos de material por usuario	134
Ilustración 107. Resultados de las pruebas del servicio egresos materiales	134
Ilustración 108. Resultado pregunta 1	160
Ilustración 109. Resultado pregunta 2	161
Ilustración 110. Resultado pregunta 3	162
Ilustración 111. Resultado pregunta 4	162
Ilustración 112. Resultado pregunta 5	163

Lista de Tablas

Tabla 1. Requerimiento funcional 1	46
Tabla 2. Requerimiento funcional 2	46
Tabla 3. Requerimiento funcional 3	47
Tabla 4. Requerimiento funcional 4	47
Tabla 5. Requerimiento funcional 5	47
Tabla 6. Requerimiento funcional 6	48
Tabla 7. Requerimiento funcional 7	48
Tabla 8. Requerimiento funcional 8	48
Tabla 9. Requerimiento funcional 9	49
Tabla 10. Requerimiento funcional 10	49
Tabla 11. Requerimiento funcional 11	49
Tabla 12. Requerimiento funcional 12	49
Tabla 13. Requerimiento funcional 13	50
Tabla 14. Requerimiento no funcional 1	50
Tabla 15. Requerimiento no funcional 2	50
Tabla 16. Requerimiento no funcional 3	51
Tabla 17. Pseudo requerimiento 1	51
Tabla 18. Pseudo requerimiento 2	51
Tabla 19. Pseudo requerimiento 3	52
Tabla 20. Comparación Scrum y XP	69
Tabla 21. Tabla usuario	73
Tabla 22. Tabla empresa.....	73

Tabla 23. Tabla cliente	74
Tabla 24. Tabla producto.....	74
Tabla 25. Tabla ingreso de producto	75
Tabla 26. Tabla egreso de producto.....	75
Tabla 27. Tabla material.....	76
Tabla 28. Tabla ingreso de material	76
Tabla 29. Tabla egreso de material.....	77
Tabla 30. Roles scrum	89
Tabla 31. Producto backlog.....	89
Tabla 32. Sprint 1	90
Tabla 33. Sprint 2	91
Tabla 34. Sprint 3	91
Tabla 35. Sprint 4	92
Tabla 36. Sprint 5	92
Tabla 37. Sprint 6	93
Tabla 38. Sprint 7	93

INTRODUCCIÓN

El trabajo de grado que se presenta a continuación tiene como finalidad y objetivo el desarrollo de una aplicación web de escritorio para la gestión del inventario de la fábrica prefabricados Marlon Estefan, la cual se enfoca en la elaboración de los materiales correspondientes para la producción de calzado (suela, corte, accesorios). Mediante esta aplicación se espera tener el control del inventario, de los clientes, así como de los ingresos y de los egresos de los productos fabricados por el negocio, al igual como el ingreso de materiales, se optó por enfocarse en que la aplicación sea de escritorio para facilitar el uso en el equipo de cómputo del negocio y de esta manera mantener seguro el tráfico de la aplicación al no estar desplegada en algún hosting.

Generalmente las aplicaciones de escritorio se han desarrollado con herramientas tecnológicas tradicionales y aunque están resultan útiles para la elaboración de un producto final de software casi siempre los diseños de las interfaces graficas se ven rezagados por las nuevas herramientas que han surgido en los últimos años, primordialmente las tecnologías enfocadas a la web, las herramientas de desarrollo web tienen una flexibilidad mayor en comparación con las herramientas tradicionales y permiten tener un resultado mucho más estético y un amplio abanico de posibilidades en cuanto se refiere a capa de presentación de la aplicación, es principalmente por este motivo que se ha decidido hacer uso de la herramienta de desarrollo electronjs ya que esta nos permite ejecutar en una aplicación de escritorio un proyecto web realizado con frameworks como lo puede ser Angular, el cual fue usado para el presente trabajo, para así finalmente obtener un producto con un diseño de presentación como se puede encontrar en aplicaciones web, pero con un producto final que se ejecutara en forma de escritorio sin acceder a este mediante una página en internet.

El desarrollo de esta aplicación a partido desde los requerimientos por parte del cliente y en base a ellos se ha planteado los diferentes diagramas que permiten el diseño de la arquitectura del software. Además de esto se ha basado el desarrollo en el marco de trabajo o metodología ágil de desarrollo Scrum, en lo que se refiere a la codificación se hizo bajo la arquitectura hexagonal en la cual se debe aislar el dominio (lógica de negocio) de la capa de presentación, tal cual como se trabajan actualmente en los desarrollos web, así que para esto se realizó el backend bajo un diseño orientado al dominio el cual fue desarrollado con Nodejs y Expressjs en este se exponen los servicios los cuales se encargan de comunicarse con la base de datos de PostgreSQL la cual se desplego en un hosting llamado ElephantSQL dichos servicios serán consumidos por el frontend de la aplicación mediante peticiones HTTP con los módulos provistos por Angular, dicho frontend hace referencia a la capa de presentación de la aplicación, como se mencionó anteriormente este se desarrolló con Angular el cual es un framework para el desarrollo de aplicaciones de una sola página (SPA). Además, se configuro el frontend desarrollado en Angular con el framework Electronjs para adaptar la aplicación web a una aplicación de escritorio. Por último, se tienen las pruebas unitarias realizadas en el backend para verificar su correcto funcionamiento.

CAPÍTULO 1. PLANTEAMIENTO DEL PROBLEMA

1.1. PLANTEAMIENTO DEL PROBLEMA

Actualmente la mayoría de las empresas optan por usar sistemas de información para dar soporte a los procesos que se llevan a cabo en las mismas, permitiendo tener un mayor control y facilitar el acceso de la información. Por lo tanto, se tiene como objetivo realizar una aplicación web de escritorio para la fábrica “Prefabricados Marlon Stefan”, con el fin de que la fábrica pueda tener un mejor control sobre su inventario, así como una mayor accesibilidad a la información sobre los materiales que se tienen en el inventario actualmente.

Por consiguiente, la fábrica elabora los materiales correspondientes para la producción de calzado (suela, corte y accesorios). Además, no cuenta con ningún sistema de información el cual de soporte a este proceso del inventario, dicho proceso es llevado a cabo mediante el uso de herramientas como Excel y también se hacen registros a mano en hojas, resultando que al momento de querer obtener información como por ejemplo, la cantidad de suelas disponibles de un tipo en específico requiera de un mayor tiempo de lo necesario, debido a que se debe buscar manualmente en los registros que se tengan, hasta encontrar el tipo de suela requerido, al igual también representa peligros en el almacenamiento de la información, debido a que se pueden extraviar las hojas donde tengan una parte del inventario o pueda surgir algún daño en el equipo donde manejen las hojas de Excel, resultando así en la pérdida de la información. Además, el uso de Excel y registros a mano es ineficiente a la hora de querer acceder rápidamente a la información del inventario.

1.2. FORMULACIÓN DEL PROBLEMA

¿Cómo el desarrollo de una aplicación web de escritorio utilizando electronjs puede beneficiar al proceso de gestión de inventario de la fábrica “Prefabricados Marlon Stefan”?

1.3. JUSTIFICACIÓN

Los avances de la tecnología han permitido que muchas tareas que antes eran llevado a cabo manualmente por personas, hoy en día el crecimiento del uso de sistemas de información ha dado paso a que muchas empresas tengan un mejor control de los datos, al no manejarlos físicamente en papel, como se suele hacer en algunas empresas. Por consiguiente, las aplicaciones de escritorio suelen desarrollarse mediante el uso de tecnologías como Java en NetBeans, lo cual es funcional pero los componentes que nos provee Java no son atractivos visualmente para el cliente, por lo tanto, se utilizara Electronjs el cual es un framework que nos permite crear aplicaciones de escritorio con JavaScript, HTML, y CSS, logrando así tener la parte visual de una aplicación web funcionando como una aplicación de escritorio.

Por consiguiente, se tendrá una aplicación web de escritorio la cual permitirá que la fábrica pueda prescindir del uso del papel para llevar el control del inventario, así como también reducir una parte de los costos al no necesitar de la compra de este suministro. Esto nos conlleva, a que el manejo del inventario sea más práctico y sencillo, y a su vez facilitara el acceso y seguridad de la información. En esto se basa este proyecto, en el desarrollo de una aplicación web de escritorio para la fábrica “Prefabricados Marlon Stefan” la cual, de soporte al proceso de la gestión del inventario, mejorando así el control que se lleva hoy en día.

1.4. OBJETIVOS

1.4.1. Objetivo General

- Desarrollar una aplicación web de escritorio de gestión de inventario a través de la herramienta electronjs para la fábrica “Prefabricados Marlon Stefan”

1.4.2. Objetivos Específicos

- Identificar los requerimientos funcionales y no funcionales suministrados por el cliente para el desarrollo de la aplicación web de escritorio.
- Analizar las diferentes metodologías de desarrollo a través de un cuadro comparativo para la aplicación web de escritorio de gestión de inventario.
- Diseñar la aplicación web de escritorio mediante la plataforma Figma para la gestión de inventario de la fábrica.
- Implementar la aplicación web de escritorio para la gestión de inventario de la fábrica.
- Realizar pruebas a la aplicación web de escritorio para la gestión de inventario.

1.4.3. Acotaciones

- Se utilizará la plataforma Figma para realizar los diseños del sistema de información.
- El frontend de la aplicación web de escritorio se desarrollará en Angular.
- El frontend de la aplicación web seguirá el patrón modelo–vista–modelo de vista (abreviado MVVM).
- El backend de la aplicación web de escritorio se desarrollará con NodeJS y Express.
- El backend de la aplicación se realizará bajo la arquitectura hexagonal.
- Creación de la base de datos con PostgreSQL para el almacenamiento de la información.

CAPÍTULO 2. MARCO TEÓRICO

2.1. ANTECEDENTES

Se presenta los trabajos de investigación, los artículos que son importantes referentes para la propuesta.

2.1.1. Antecedentes Internacional

En la tesis de (Marquez Solis, 2020) titulada “Desarrollo de una aplicación web para la gestión de inventarios de la empresa 1Solutions S.A.C. – Huancayo”, el autor tenía como objetivo mejorar la gestión de inventario durante el año 2020 en la empresa 1Solutions S.A.C. Por lo tanto, planteo como solución implementar una aplicación web para la gestión de inventarios que permita agregar herramientas y productos y gestionar servicios a clientes y compras realizadas, para que, de esa manera, la empresa reduzca pérdidas en existencias y mejore el tiempo de pedidos de productos.

En la tesis de (Diaz Arrieta & Ortiz Vinueza, 2018) titulada “Desarrollo de una aplicación web y móvil para la gestión de inventario y pedidos utilizando servicios RESTful”, los autores tenían como objetivo el desarrollo de una aplicación web y aplicación móvil para la gestión de inventarios (GICO) para la empresa Comercial Ortiz. El sistema fue realizado utilizando la metodología de desarrollo ágil Scrum, mediante la cual se pudo crear un trabajo cooperativo entre el cliente y el equipo de desarrollo, para la toma de requerimientos se utilizó el estándar IEEE 830, con la cual se pudieron analizar las necesidades del cliente para poder

realizar el desarrollo del sistema. Se utilizó herramientas tecnológicas como: JAVA, JavaScript, Json, EdGrid 2, PostgreSQL 9.5, iReport, NetBeans IDE, Android Studio, pgAdmin, Payara server 4.2, Servicios RESTful, para el consumo de servicios desde la aplicación móvil se usó las librerías, Retrofit, Volley. El producto final se evaluó según el estándar ISO 9126 que en su apartado se establecen métricas externas, se evaluaron las características de mantenibilidad, portabilidad y eficiencia de las cuales se seleccionaron subcaracterísticas como: capacidad para ser analizado, capacidad para ser cambiado, fácil instalación, disponibilidad de coexistencia y tiempos de respuesta.

2.1.2. Antecedentes Nacional

En la tesis de (Hernandez Castro & Herrera Briceño, 2021) titulada “Aplicación web para controlar el inventario de la empresa First medical de Colombia”, los autores tenían como objetivo diseñar e implementar una aplicación web de manejo de inventario para el software “QUIRON”, con el fin de tener una trazabilidad en tiempo real del inventario de First Medical de Colombia. Debido a que dentro de la institución es necesario llevar a cabo las buenas prácticas documentales en cuanto al manejo de formularios y protocolos de documentación, las cuales cumplan con los requerimientos sistema de gestión de calidad institucional. Sin embargo, existen diferentes variables que pueden retrasar esta actividad, como las falencias en el control y trazabilidad de los insumos y repuestos que afectan directamente el inventario, debido a la premura, no es viable la actualización en el momento en que se retira un repuesto o insumo del almacén.

En la tesis de (Romero, 2019) titulada “Diseño e implementación de sistema de inventarios para el almacén de pinturas y ferretería Ferrecolor”, el autor tenía como objetivo

realizar un sistema de información el cual pudiera gestionar de manera eficiente el inventario haciendo uso de la tecnología Java EE debido a que la ferretería Ferrecolor carecía de una herramienta informática que le permitiera ejercer mayor control sobre los productos que comercializa. Para esto realizo un análisis de requerimiento el cual mediante el uso de la metodología ágil SCRUM le permitió dar solución al problema creando una aplicación web con el estándar de Java EE, el framework JSF (Java Server Faces) y el motor de base de datos MySQL.

En la tesis de (Jaramillo Bolívar, 2019) titulada “Desarrollo de una plataforma de gestión de eventos para el área de gestión humana y administrativa de Ceiba Software”, el autor realizo el desarrollo de una plataforma para la gestión de eventos del area de gestión humana y administrativa, en la cual implementó las buenas prácticas en ambos frentes, desde lo visual, hasta el core y el núcleo de la aplicación, a través de un conjunto de patrones y reglas o pautas que hicieron del código una estructura natural y mantenible: la arquitectura hexagonal, el diseño dirigido por el dominio, y en general, un código fácil de entender e interpretar.

2.1.3. Antecedentes Regional

En la tesis de (Betancur Rosas, 2018) titulada “Aplicación web para la gestión de inventario de reactivos del laboratorio de química de la universidad de Pamplona sede Villa del Rosario” el autor desarrolla e implementa una aplicación web para la gestión de inventario de reactivos del laboratorio de química de la Universidad de Pamplona sede Villa del Rosario, debido a que el control de inventario y solicitud de reactivos se lleva a cabo en formatos manuales. Para realizar este desarrollo el autor determinó el diseño metodológico, estableciendo

la investigación descriptiva, que permitió conocer la realidad de la población objeto de estudio conformada por los auxiliares, docentes y coordinador del laboratorio, a los cuales se aplicó el instrumento de recolección, es decir, la entrevista y la observación directa, en donde la metodología de analizarla cuantitativa y cualitativamente mediante tablas y gráficas que permitieron conocer el nivel de aceptación de los Docentes y Administrativos, en poder tener una herramienta que facilite la gestión de solicitud de reactivos que se encuentran disponible en la página web de la Universidad de Pamplona y así poder diligenciar el formato en forma virtual fla-04 v 02 solicitud de reactivos y medios.

En la tesis de (Picon Rincon, 2014) titulada “Desarrollo de un software para hacer zoometría en ganado blanco orejinegro para la universidad Francisco de Paula Santander Ocaña” el autor lo resume como una propuesta tecnológica, a la solución del manejo deficiente que se estaba llevando a cabo de la información relacionada con las actividades de zoometría aplicadas al ganado blanco orejinegro de la universidad Francisco de Paula Santander Ocaña; desarrollando un software de escritorio, el cual se generó haciendo uso de la metodología ágil XP, brindando así, una herramienta que ayude y facilite a la toma de decisiones.

2.2. MARCO TEÓRICO

2.2.1. FRAMEWORK

Un framework es un marco o esquema de trabajo generalmente utilizado por programadores para realizar el desarrollo de software. Utilizar un framework permite agilizar los procesos de desarrollo ya que evita tener que escribir código de forma repetitiva, asegura unas buenas prácticas y la consistencia del código. Por consiguiente, un framework es por tanto un conjunto de herramientas y módulos que pueden ser reutilizados para varios proyectos.



Figura 1. Imagen de frameworks Tomada de <https://www.suratica.es/que-es-un-framework/>

2.2.1.1. Ventajas de los Frameworks

Entre las ventajas de utilizar un framework para el desarrollo de software distinguimos:

(a) El programador ahorra tiempo ya que dispone ya del esqueleto sobre el que desarrollar una aplicación. (b) Facilita los desarrollos colaborativos, al dejar definidos unos estándares de programación. (c) Al estar ampliamente extendido, es más fácil encontrar herramientas, módulos e información para utilizarlo. (d) Proporciona mayor seguridad, al tener gran parte de las potenciales vulnerabilidades resueltas. (e) Normalmente existe una comunidad detrás, un conjunto de desarrolladores que pueden ayudar a responder consultas.

2.2.1.2. Ejemplos de Frameworks

Estos son algunos de los frameworks más conocidos: (a) .Net: es Framework de Microsoft y uno de los más utilizados. (b) Symphony: proyecto PHP de software libre. (c) Zend Framework: framework de código abierto para desarrollar aplicaciones web y con servicios web PHP. (d) Laravel: uno de los frameworks de código abierto más fáciles de asimilar para PHP. (e) Django: framework de desarrollo web de código abierto escrito en Python. (f) Ruby on Rails: framework de aplicaciones web de código abierto escrito en el lenguaje de programación Ruby. (g) Angular: framework de código abierto desarrollado en TypeScript y mantenido por Google.

2.2.2. APLICACIÓN WEB

Una aplicación web es un tipo de software cuyo lenguaje está codificado para ser un navegador web y su ejecución se lleva a través de un navegador de Internet o una intranet. A pesar de la similitud de los términos, no hay que confundir aplicación web y sitio web, pues los propósitos de ambas plataformas digitales dan respuesta a necesidades muy diferentes: (a) Mientras que un sitio web aporta información, una aplicación web está enfocada en la realización de acciones. (b) Una aplicación web puede pertenecer al proyecto de un sitio web, pero al revés no. (c) Una aplicación web está compuesta por un mayor número de tareas y funciones y son mucho más complejas.



Figura 2. Aplicación Web Tomado de <https://mrhouston.net/blog/que-son-las-aplicaciones-web/>

2.2.3. ELECTRONJS

Electron.js es una plataforma para desarrollar aplicaciones de escritorio usando tecnologías web (HTML, CSS y JavaScript) creada y mantenida por GitHub. Electron.js funciona creando dos tipos de procesos, el proceso main y el proceso renderer. El primero es un proceso de Node.js, este es el proceso principal, viene a ser la aplicación en sí misma, este proceso tiene acceso a varias API de Electron.js solo para este proceso que ayudan a la comunicación con el SO y realizar distintas acciones o efectos. Por lo tanto, el segundo (renderer) es un proceso de Chromium, con una diferencia, este Chromium tiene un Node.js incorporado y acceso a todos sus módulos y los que se instalan con npm (esto permitiría usar React.js, Angular.js, Polymer, etc. Para desarrollar la UI o cualquier otra librería básicamente), por lo que desde el renderer se puede usar módulos como fs para leer y escribir en el disco, o hacer peticiones a una base de datos directamente.

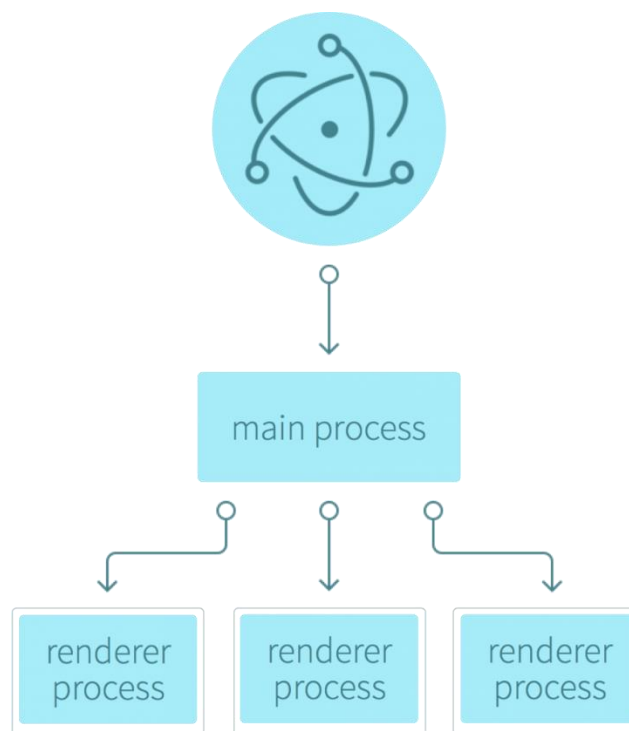


Figura 3. Electronjs Tomado de <https://platzi.com/blog/aplicaciones-escritorio-electron-js/>

2.2.4. ANGULAR

Es uno de los framework de JavaScript más populares, que los desarrolladores actualmente utilizan para crear sitios web dinámicos. Es un framework de ingeniería de software de código abierto que se emplea para crear aplicaciones web de una sola página. Los desarrolladores también lo usan para crear menús animados para páginas web HTML. El framework es una creación de los ingenieros de Google, Misko Hervey y Adam Abrons, donde Google lanzó oficialmente la primera versión, AngularJS, en el año 2012, y la ha mantenido desde entonces. AngularJS usa el patrón de diseño Modelo-Vista-Controlador (MVC), que se usa en el desarrollo de aplicaciones web. En el contexto de AngularJS, el modelo es el framework, mientras la vista es HTML y el control es JavaScript, el código entre ellos es sincronizado y facilita el trabajo de los desarrolladores porque reduce la cantidad de código necesario para escribir. Las ventajas de AngularJS es por poseer una arquitectura bidireccional de datos porque enlaza JavaScript y HTML y ahorra mucho tiempo a los desarrolladores, posee directivas para un marco amplio en la funcionalidad de los archivos HTML en la estructura de código en las plantillas son más organizadas y permite la modificación o reparación de las aplicaciones, se pueden realizar pruebas unitarias y de integración, y compatibilidad en los navegadores web en ordenadores de escritorio y dispositivos móviles.



Figura 4. ¿Qué es Angular? Tomado de <https://www.hostinger.es/tutoriales/que-es-angular>

2.2.5. NODEJS

Es una plataforma que se usa principalmente para desarrollar aplicaciones web implementando JavaScript en el servidor, la idea principal de NodeJS es permanecer ligero y eficiente en la superficie del uso intensivo de datos en tiempo real de las aplicaciones que se ejecutan en dispositivos distribuidos. Es un entorno de desarrollo para ejecutar JavaScript en el servidor, fue creado con el enfoque de ser útil en la creación de programas de red altamente escalables. Las principales ventajas de NodeJS es por ser rápido, está construido para funcionar de manera asíncrona, es una excelente opción para aplicaciones en tiempo real, funciona muy bien con bases de datos no relacionales (NoSQL) y siempre existen cambios activos en la plataforma.



Figura 5. ¿Qué es NodeJS? Tomado de <https://niixer.com/index.php/2020/11/16/node-js-en-el-desarrollo-web-su-importancia-y-uso-actualmente/>

2.2.6. MODELO-VISTA-CONTROLADOR (MVC)

Es un patrón para el desarrollo de aplicaciones de software y una guía para el diseño de arquitectura de aplicaciones que ofrece una alta interactividad con los usuarios. La finalidad de este patrón es ordenar y organizar la aplicación en tres estructuras en forma independiente e interconectadas, la primera estructura es un modelo que representa los datos de la aplicación de

software y sus reglas de negocio (Modelo), el segundo es un conjunto de interfaces que representan los formularios y la parte del lado del cliente que es la entrada y salida de los datos por parte del usuario (Vista) y el tercero es un conjunto de componentes llamados controladores que son los responsables en llevar a cabo las peticiones de los usuarios y el control del flujo en la ejecución por parte de la aplicación del software.

En otros conceptos importantes nos encontramos: La vista es el componente que se encarga de mostrarle al usuario final las pantallas, ventanas, páginas y formularios; el resultado de una solicitud, el modelo es donde se implementa la base de datos aquí es donde se realizan las consultas, búsquedas, filtros y actualizaciones, y el controlador es el componente que se encarga de gestionar las instrucciones que se reciben, atenderlas y procesarlas. Por medio de él se comunican el modelo y la vista de la aplicación.

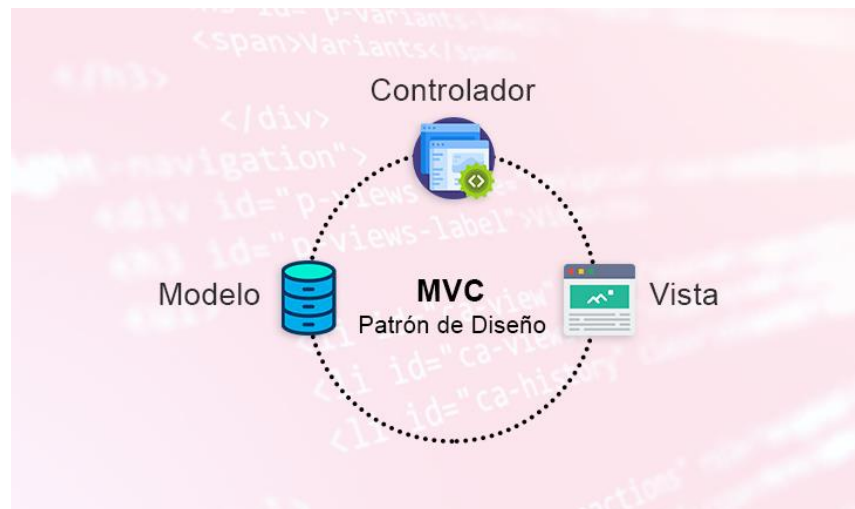


Figura 6. Patrón de diseño MVC Tomado de <https://www.easyappcode.com/patron-de-diseno-mvc-que-es-y-como-puedo-utilizarlo>

2.2.7. MVVM

MVVM es un refinamiento del diseño de MVC en el que ViewModel en MVVM facilita la separación del desarrollo de la interfaz gráfica de usuario (UI), es decir, la capa de presentación. ViewModel (VM) es responsable de exponer (convertir) los objetos de datos del modelo de tal manera que los objetos se manejen y presenten más fácilmente. En este sentido, ViewModel es más un modelo que una vista y maneja la mayoría, si no toda, la lógica de visualización de la vista. MVVM es una variación del patrón de diseño del modelo de presentación de Martin Fowler en el sentido de que un modelo de presentación abstrae la vista de una manera que no depende de la plataforma de interfaz de usuario específica. Por lo tanto, ni el Presentador ni la Vista son conscientes el uno del otro.

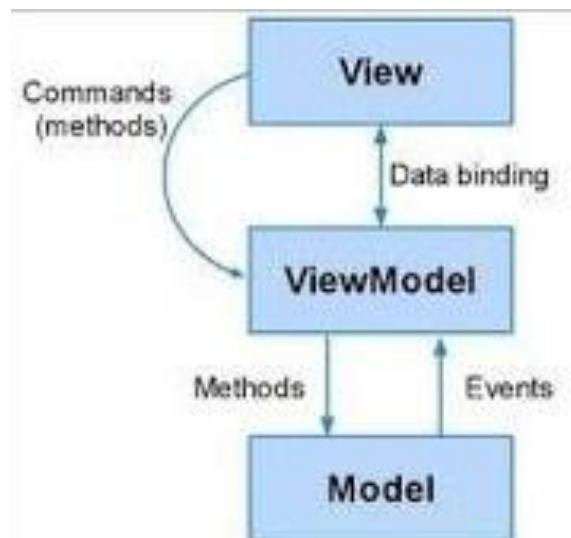


Figura 7. MVVM Tomado de <https://www.code-sample.com/2019/11/is-angularjs-mvvm-or-mvc-mvvm-vs-mvc.html>

2.2.8. ARQUITECTURA HEXAGONAL

La arquitectura hexagonal es un estilo de arquitectura de software que mueve el foco de un programador desde un plano más conceptual hacia la distinción entre el interior y el exterior del software. La parte interior son los casos prácticos y el modelo domain está construido sobre ello. La parte exterior es UI, base de datos, mensajería, etc. La conexión entre el interior y el exterior es mediante puertos, y su implementación equivalente se conocen como adaptadores. Por esta razón, este estilo de arquitectura hexagonal se conoce habitualmente como Puertos y Adaptadores.

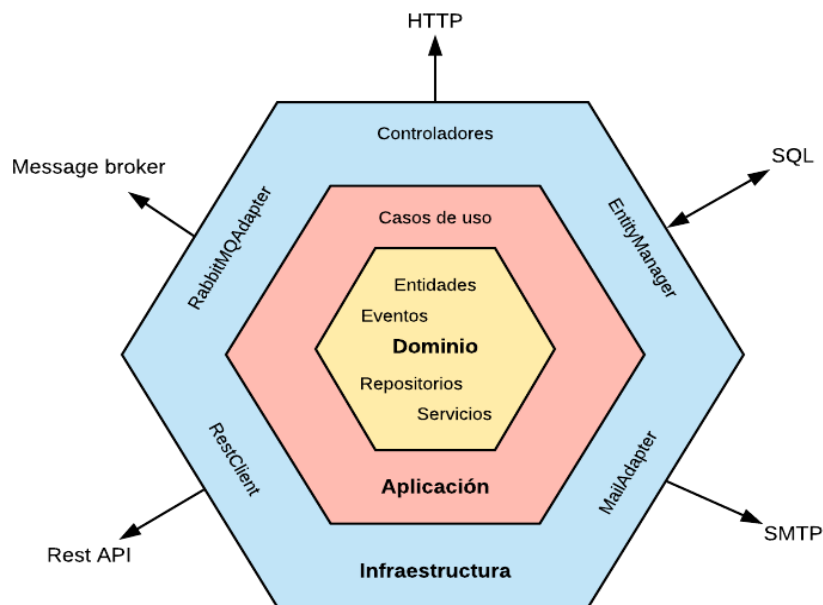


Figura 8. Arquitectura Hexagonal Tomado de <https://medium.com/@edusalguero/arquitectura-hexagonal-59834bb44b7f>

2.2.9. METODOLOGÍA ÁGIL DE DESARROLLO SCRUM

Scrum es un marco que permite el trabajo colaborativo entre equipos. Al igual que un equipo de rugby (de donde proviene su nombre) cuando entrena para un gran partido, scrum anima a los equipos a aprender a través de las experiencias, a autoorganizarse mientras aborda un problema y a reflexionar sobre sus victorias y derrotas para mejorar continuamente.

Aunque son los equipos de desarrollo de software los que utilizan con mayor frecuencia este tipo de scrum, sus principios y lecciones se pueden aplicar a todo tipo de trabajo en equipo. Esta es una de las razones por las que es tan popular. Aunque se considera a menudo un marco de gestión de proyectos ágil, scrum incluye un conjunto de reuniones, herramientas y funciones que, de forma coordinada, ayudan a los equipos a estructurar y gestionar su trabajo. (Atlassian, 2018)

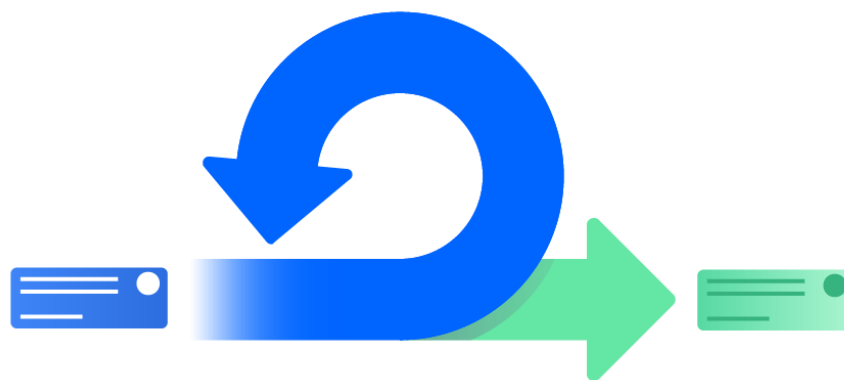


Ilustración 1. Scrum sprint - Tomado de <https://www.atlassian.com/es/agile/scrum/sprints>

2.2.9.1. Roles Scrum

Scrum presenta tres roles: el propietario del producto, el experto en scrum y los miembros del equipo de desarrollo. Los tres roles de scrum describen las responsabilidades clave de los miembros del equipo de scrum. No son cargos, lo que significa que cualquier cargo,

incluso los que tienes en la actualidad, puede desempeñar uno de los roles. Como la esencia del scrum es el empirismo, la autoorganización y la mejora continua, los tres roles dan una definición mínima de las responsabilidades y obligaciones para permitir a los equipos realizar el trabajo de manera eficaz; así se consigue que asuman la tarea de organizarse y seguir mejorando. (Atlassian, s.f.)

2.2.9.2. Artefactos de Scrum

Cuando hablamos de "artefactos", nos referimos a algo que fabricamos, por ejemplo, una herramienta para solucionar un problema. En scrum, estos tres artefactos son un backlog del producto, un backlog de sprint y un incremento con tu definición de "hecho". Estas son las tres constantes en un equipo de scrum que seguimos revisando y en las que seguimos invirtiendo horas y horas.

Los artefactos son: (a) Backlog del producto: es la lista principal del trabajo que debe realizar el propietario del producto o el gestor de productos. (b) Backlog de sprint: se trata de la lista de elementos, historias de usuario o correcciones de errores, seleccionadas por el equipo de desarrollo, para su implementación en el ciclo actual de sprint. (c) Incremento: es el producto final utilizable de un sprint. (Atlassian, 2018)

2.2.10. API (Application Programming Interface)

Es un conjunto de procedimientos y funciones creados para permitir el acceso al lado del servidor de aplicaciones por parte de los desarrolladores con la prioridad de reutilizar servicios creados para una finalidad o necesidad. Puede emplear para comunicarse con una base de datos, sistema operativo o con protocolos de comunicaciones. Es un conjunto de instrucciones, protocolos de comunicación y herramientas para desarrollar e interconectar diferentes aplicaciones de software, en pocas palabras es un conjunto de métodos,

perfectamente definidos y acotados, que permite la comunicación entre dos o varias aplicaciones y sistemas, sin la necesidad de modificar estas. Las API pueden ser a nivel de aplicaciones web, sistemas operativos, bases de datos, librerías de software e incluso a nivel de hardware de un sistema.



Figura 9. ¿Qué es una API? Tomado de <https://www.wrike.com/es/blog/que-es-una-api-necesitas-saber/>

2.2.11. POSTGRESQL

PostgreSQL, es un sistema de código abierto de administración de bases de datos del tipo relacional, aunque también es posible ejecutar consultas que sean no relaciones. En este sistema, las consultas relacionales se basan en SQL, mientras que las no relacionales hacen uso de JSON. Se trata de un sistema de código abierto y además gratuito, y su desarrollo es llevado adelante por una gran comunidad de colaboradores de todo el mundo que día a día ponen su granito de arena para hacer de este sistema una de las opciones más sólidas a nivel de bases de datos. Dos detalles a destacar de PostgreSQL es que posee data types (tipos de datos) avanzados y permite ejecutar optimizaciones de rendimiento avanzadas, que son características que por lo

general solo se ven en sistemas de bases de datos comerciales, como por ejemplo SQL Server de Microsoft u Oracle de la compañía homónima.



Figura 10. ¿Qué es PostgreSQL? Tomado de <https://blog.infranetworking.com/servidor-postgresql/>

2.2.12. PRUEBAS UNITARIAS

Las pruebas unitarias consisten en aislar una parte del código y comprobar que funciona a la perfección. Son pequeños tests que validan el comportamiento de un objeto y la lógica. El unit testing suele realizarse durante la fase de desarrollo de aplicaciones de software o móviles. Normalmente las llevan a cabo los desarrolladores, aunque en la práctica, también pueden realizarlas los responsables de QA.

Hay una especie de mito respecto a las pruebas unitarias. Algunos desarrolladores están convencidos de que son una pérdida de tiempo y las evitan buscando ahorrar tiempo. Nada más alejado de la realidad. Con ellas se detectan antes errores que, sin las pruebas unitarias, no se podrían detectar hasta fases más avanzadas como las pruebas de sistema, de integración e incluso en la beta. Realizar pruebas unitarias con regularidad supone, al final, un ahorro de tiempo y dinero.

2.2.12.1. Motivos para realizar un test unitario

Estos son los motivos para realizar un test unitario: (a) Las pruebas unitarias demuestran que la lógica del código está en buen estado y que funcionará en todos los casos. (b) Aumentan la legibilidad del código y ayudan a los desarrolladores a entender el código base, lo que facilita hacer cambios más rápidamente. (c) Los test unitarios bien realizados sirven como documentación del proyecto. (d) Se realizan en pocos milisegundos, por lo que podrás realizar cientos de ellas en muy poco tiempo.

2.2.12.2. Las 3 A's del unit testing

Para llevar a cabo buenas pruebas unitarias, deben estar estructuradas siguiendo las tres A's del Unit Testing. Se trata de un concepto fundamental respecto a este tipo de pruebas, que describe un proceso compuesto de tres pasos: (a) Arrange (organizar). Es el primer paso de las pruebas unitarias. En esta parte se definen los requisitos que debe cumplir el código. (b) Act (actuar). Es el paso intermedio de las pruebas, el momento de ejecutar el test que dará lugar a los resultados que deberás analizar. (c) Assert (afirmar). En el último paso, es el momento de comprobar si los resultados obtenidos son los que se esperaban. Si es así, se valida y se sigue adelante. Si no, se corrige el error hasta que desaparezca.

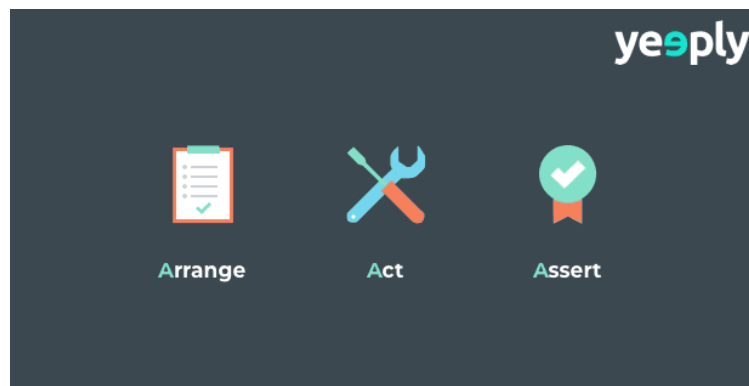


Figura 11. ¿Qué son pruebas unitarias? Tomado de <https://www.yeeply.com/blog/que-son-pruebas-unitarias/>

2.3. MARCO LEGAL

A continuación, se presenta un listado de leyes que el gobierno de colombiano dispone para la elaboración de proyectos.

2.3.1. Ley Estatutaria 1581 de 2012

Por la cual se dictan disposiciones generales para la protección de datos personales.

Según ARTÍCULO 1°. OBJETO. La presente ley tiene por objeto desarrollar el derecho constitucional que tienen todas las personas a conocer, actualizar y rectificar las informaciones que se hayan recogido sobre ellas en bases de datos o archivos, y los demás derechos, libertades y garantías constitucionales a que se refiere el artículo 15 de la Constitución Política; así como el derecho a la información consagrado en el artículo 20 de la misma.

2.3.2. Ley 1341 TIC en Colombia

Por la cual se definen principios y conceptos sobre la sociedad de la información y la organización de las Tecnologías de la Información y las Comunicaciones –TIC–, se crea la Agencia Nacional de Espectro y se dictan otras disposiciones. Por lo tanto, **ARTÍCULO 1°. OBJETO.** La presente ley determina el marco general para la formulación de las políticas públicas que regirán el sector de las Tecnologías de la Información y las Comunicaciones, su ordenamiento general, el régimen de competencia, la protección al usuario, así como lo concerniente a la cobertura, la calidad del servicio, la promoción de la inversión en el sector y el desarrollo de estas tecnologías, el uso eficiente de las redes y del espectro radioeléctrico, así como las potestades del Estado en relación con la planeación, la gestión, la administración adecuada y eficiente de los recursos, regulación, control y vigilancia del mismo y facilitando el

libre acceso y sin discriminación de los habitantes del territorio nacional a la Sociedad de la Información.

CAPITULO 3. DESARROLLO DEL SISTEMA

En este capítulo desarrollaremos una aplicación web de escritorio de gestión de inventario a través de la herramienta electronjs para la fábrica “Prefabricados Marlon Stefan”

3.1. Identificar los requerimientos funcionales y no funcionales suministrados por el cliente para el desarrollo de la aplicación web de escritorio.

3.1.1. Requerimientos Funcionales

Tabla 1. Requerimiento funcional 1

Identificación requerimiento:	del	RF01
Nombre Requerimiento:	del	Inicio de sesión
Características:		Se necesita iniciar sesión para acceder al menú del sistema
Descripción requerimiento:	del	El sistema debe solicitar un usuario y contraseña para realizar el inicio de sesión.
Prioridad del requerimiento: Media		

Tabla 2. Requerimiento funcional 2

Identificación requerimiento:	del	RF02
Nombre Requerimiento:	del	Registrar empresa
Características:		Almacenar los datos de las empresas
Descripción requerimiento:	del	El sistema deberá permitir el registro de empresas mediante su identificación, nombre, dirección y teléfono.
Prioridad del requerimiento: Alta		

Tabla 3. Requerimiento funcional 3

Identificación requerimiento:	del	RF03
Nombre Requerimiento:	del	Registrar cliente
Características:		Almacenar los datos de los clientes
Descripción requerimiento:	del	El sistema deberá permitir el registro de clientes mediante su cedula, nombres, apellidos, celular, correo y la empresa a la que pertenece.
Prioridad del requerimiento: Alta		

Tabla 4. Requerimiento funcional 4

Identificación requerimiento:	del	RF04
Nombre Requerimiento:	del	Registrar operador
Características:		Almacenar los datos de los operadores
Descripción requerimiento:	del	El sistema deberá permitir el registro de operadores mediante su cedula, nombre, correo, usuario y contraseña.
Prioridad del requerimiento: Media		

Tabla 5. Requerimiento funcional 5

Identificación requerimiento:	del	RF05
Nombre Requerimiento:	del	Ver inventario
Características:		Visualizar el inventario
Descripción requerimiento:	del	El sistema deberá permitir la visualización tanto de los productos como los materiales que están almacenados.
Prioridad del requerimiento: Media		

Tabla 6. Requerimiento funcional 6

Identificación requerimiento:	del	RF06
Nombre Requerimiento:	del	Filtro de productos
Características:		Visualizar los productos mediante un filtro de búsqueda.
Descripción requerimiento:	del	El sistema deberá permitir al usuario visualizar los productos mediante un filtro según su nombre, tipo, calzado y género.
Prioridad del requerimiento: Baja		

Tabla 7. Requerimiento funcional 7

Identificación requerimiento:	del	RF07
Nombre Requerimiento:	del	Filtro de materiales
Características:		Visualizar los materiales mediante un filtro de búsqueda.
Descripción requerimiento:	del	El sistema deberá permitir al usuario visualizar los materiales mediante un filtro según su código, nombre, color y unidad.
Prioridad del requerimiento: Baja		

Tabla 8. Requerimiento funcional 8

Identificación requerimiento:	del	RF08
Nombre Requerimiento:	del	Registrar producto
Características:		Almacenar los datos de los productos
Descripción requerimiento:	del	El sistema deberá permitir el registro de productos mediante su nombre, tipo, material, calzado, genero, talla y cantidad.
Prioridad del requerimiento: Alta		

Tabla 9. Requerimiento funcional 9

Identificación requerimiento:	del	RF09
Nombre Requerimiento:	del	Registrar material
Características:		Almacenar los datos de los materiales
Descripción requerimiento:	del	El sistema deberá permitir el registro de materiales mediante su código, nombre, color, unidad y cantidad.
Prioridad del requerimiento: Alta		

Tabla 10. Requerimiento funcional 10

Identificación requerimiento:	del	RF10
Nombre Requerimiento:	del	Ver historial de ingresos
Características:		Visualizar los ingresos de productos y materiales
Descripción requerimiento:	del	El sistema deberá permitir la visualización de los ingresos realizados tanto de los productos como de los materiales.
Prioridad del requerimiento: Media		

Tabla 11. Requerimiento funcional 11

Identificación requerimiento:	del	RF11
Nombre Requerimiento:	del	Registrar ingresos
Características:		Almacenar los datos de ingresos de productos y materiales
Descripción requerimiento:	del	El sistema deberá permitir el registro de ingresos de productos y materiales al inventario.
Prioridad del requerimiento: Alta		

Tabla 12. Requerimiento funcional 12

Identificación requerimiento:	del	RF12
--------------------------------------	------------	-------------

Nombre Requerimiento:	del	Ver historial de egresos
Características:		Visualizar los egresos de productos y materiales
Descripción requerimiento:	del	El sistema deberá permitir la visualización de los egresos realizados tanto de los productos como de los materiales.
Prioridad del requerimiento: Media		

Tabla 13. Requerimiento funcional 13

Identificación requerimiento:	del	RF13
Nombre Requerimiento:	del	Registrar egresos
Características:		Almacenar los datos de egresos de productos y materiales
Descripción requerimiento:	del	El sistema deberá permitir la visualización de los egresos realizados tanto de los productos como de los materiales.
Prioridad del requerimiento: Alta		

3.1.2. Requerimientos no funcionales

Tabla 14. Requerimiento no funcional 1

Identificación requerimiento:	del	RNF01
Nombre Requerimiento:	del	Encriptación de contraseña
Características:		Encriptar la contraseña de los usuarios para su almacenamiento.
Descripción requerimiento:	del	El sistema deberá encriptar la contraseña para ser almacenada en la base de datos.
Prioridad del requerimiento: Alta		

Tabla 15. Requerimiento no funcional 2

Identificación requerimiento:	del	RNF02
--	------------	--------------

Nombre del Requerimiento:	Intuitivo
Características:	- Manejo de colores - Menús sencillos - Opciones fáciles de identificar
Descripción del requerimiento:	El sistema debe manejar los mismos colores para todas sus interfaces, los menús deben ser sencillos para que su manejo sea fácil de identificar.
Prioridad del requerimiento: Media	

Tabla 16. Requerimiento no funcional 3

Identificación del requerimiento:	RNF03
Nombre del Requerimiento:	Mensaje en pantalla
Características:	Generar mensajes en pantalla
Descripción del requerimiento:	El sistema deberá mostrar en pantalla un mensaje para informar el resultado de una operación o validación.
Prioridad del requerimiento: Baja	

3.1.3. Pseudo Requerimientos

Tabla 17. Pseudo requerimiento 1

Identificación del pseudo requerimiento:	PR01
Nombre del Pseudo Requerimiento:	Base de datos
Características:	Base de datos relacional
Descripción del pseudo requerimiento:	El sistema deberá almacenar todos los datos en una base de datos relacional PostgreSQL

Tabla 18. Pseudo requerimiento 2

Identificación del requerimiento:	PR02
--	-------------

Nombre Requerimiento:	del	Frontend
Características:		Angular
Descripción requerimiento:	del	El frontend de la aplicación debe realizarse con el framework Angular

Tabla 19. Pseudo requerimiento 3

Identificación requerimiento:	del	PR03
Nombre Requerimiento:	del	Backend
Características:		Nodejs con Expressjs
Descripción requerimiento:	del	El backend de la aplicación debe realizarse con Nodejs y Expressjs

3.1.4. Diagrama de Casos de Uso

A continuación, se presenta el diagrama de casos de uso tomando como base los requerimientos funcionales.

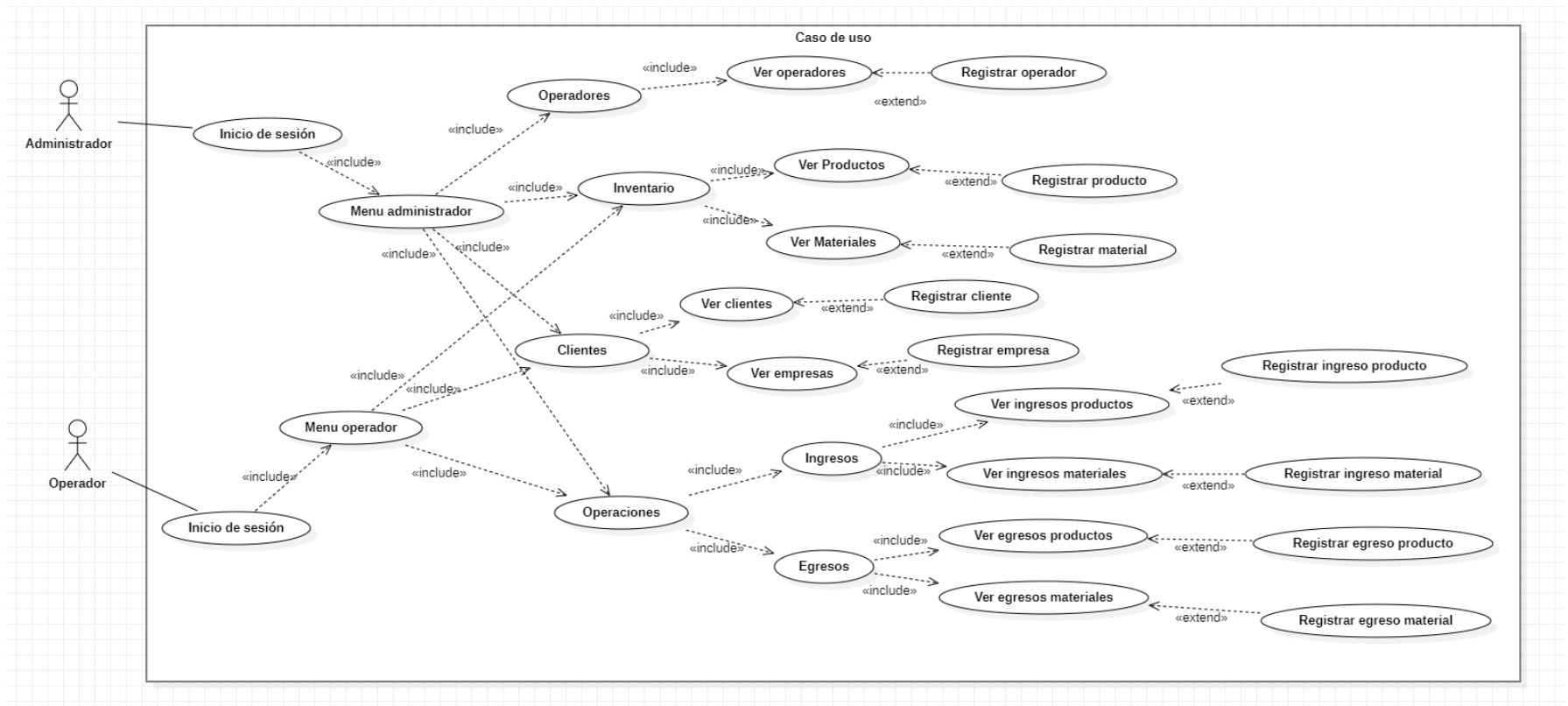


Ilustración 2. Diagrama de casos de uso

En el diagrama de caso de uso tenemos dos actores los cuales son el Administrador y el Operador cada uno puede iniciar sesión en el sistema, al iniciar tienen acceso a su respectivo menú ya sea de administrador u operador. El administrador cuenta con las opciones de inventario en la cual podrá ver los productos y materiales del mismo así como registrar nuevos de estos, clientes en donde poder ver y registrar nuevos clientes así como empresas, operadores en la cual podrá ver y registrar operadores los cuales tienen acceso al sistema, y por último operaciones en la cual podrá controlar el apartado de ingresos y egresos del inventario. Por otra parte se tiene al operador, el cual cuenta con las mismas opciones del administrador exceptuando la opción de poner gestionar los operadores del sistema.

3.1.5. Diagrama de Actividades

A continuación, se presentan los diagramas de actividades los cuales hacen referencia a las funciones de las actividades y el flujo del proceso de negocio, para la elaboración del diagrama se utilizó la herramienta StarUML.

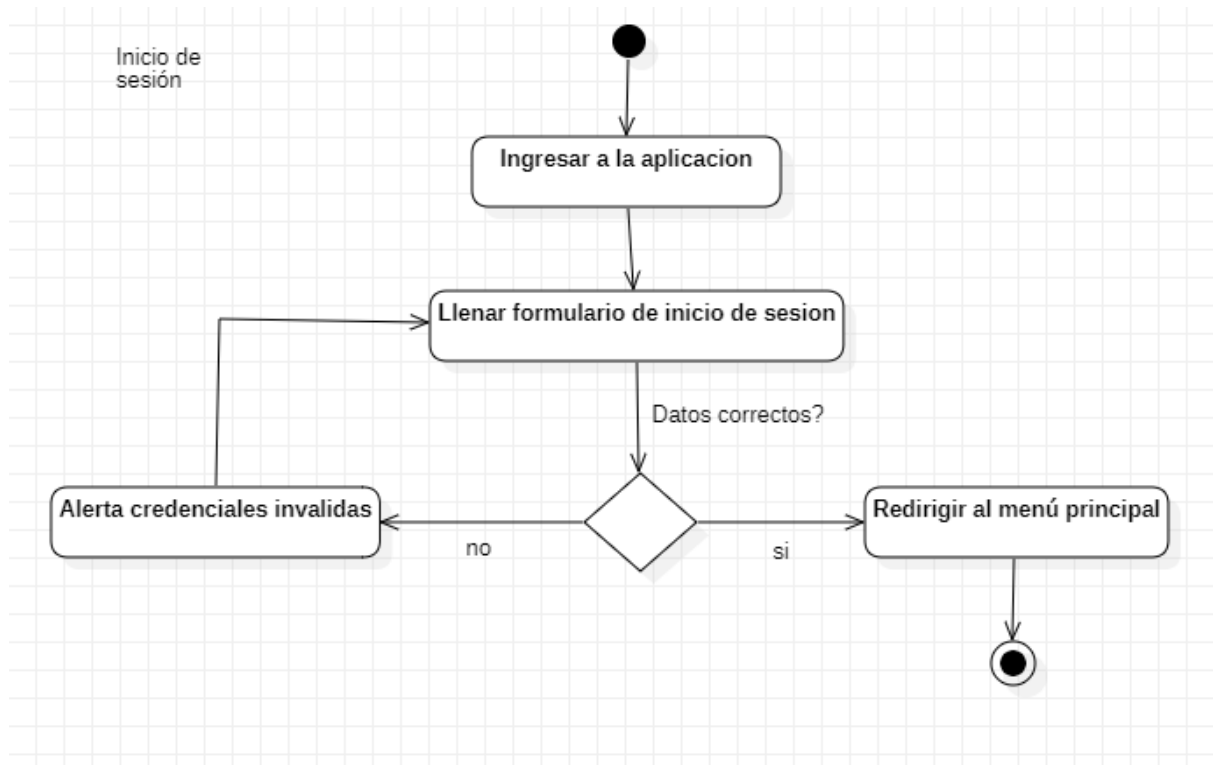


Ilustración 3. Diagrama de actividad (Inicio de sesión)

En este diagrama se describen las actividades que se deben realizar para poder iniciar sesión en el sistema, para esto se debe ingresar a la aplicación, la cual responderá con un formulario para iniciar sesión, el usuario deberá llenar el formulario para iniciar, aquí se comprobarán si las credenciales son válidas si son invalidas se alertará en pantalla, de lo contrario se le enviara al menú principal del rol iniciado.

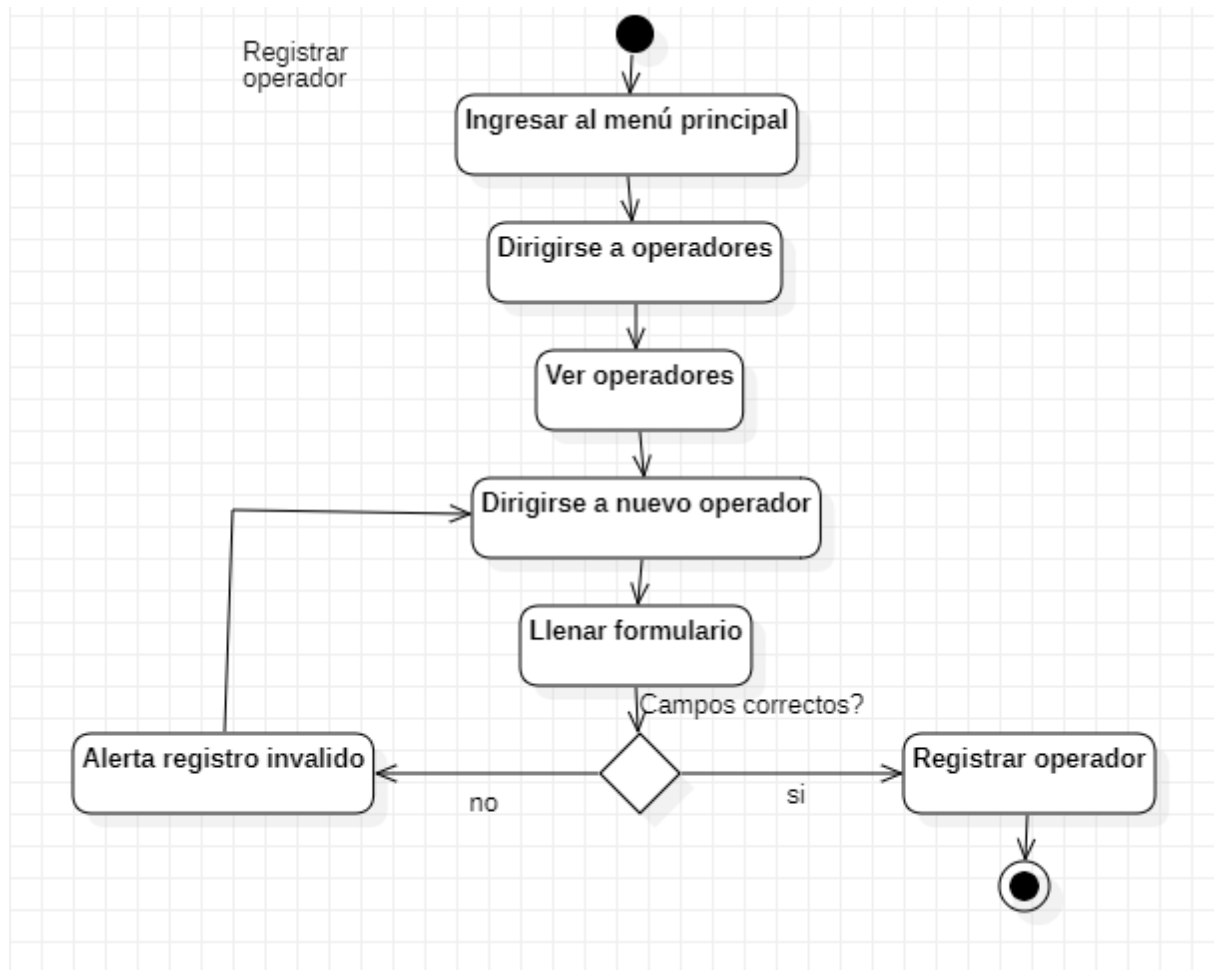


Ilustración 4. Diagrama de actividad (Registro operador)

Este diagrama describe las actividades que se deben realizar para poder registrar un operador, para esto se debe ingresar al menú principal, dirigirse a operadores, y una vez dentro se podrá dirigir a nuevo operador en el cual se le mostrara un formulario para registrarlo, aquí se comprobaran si lo datos son correctos para después confirmar mediante una alerta el registro o informar algún error.

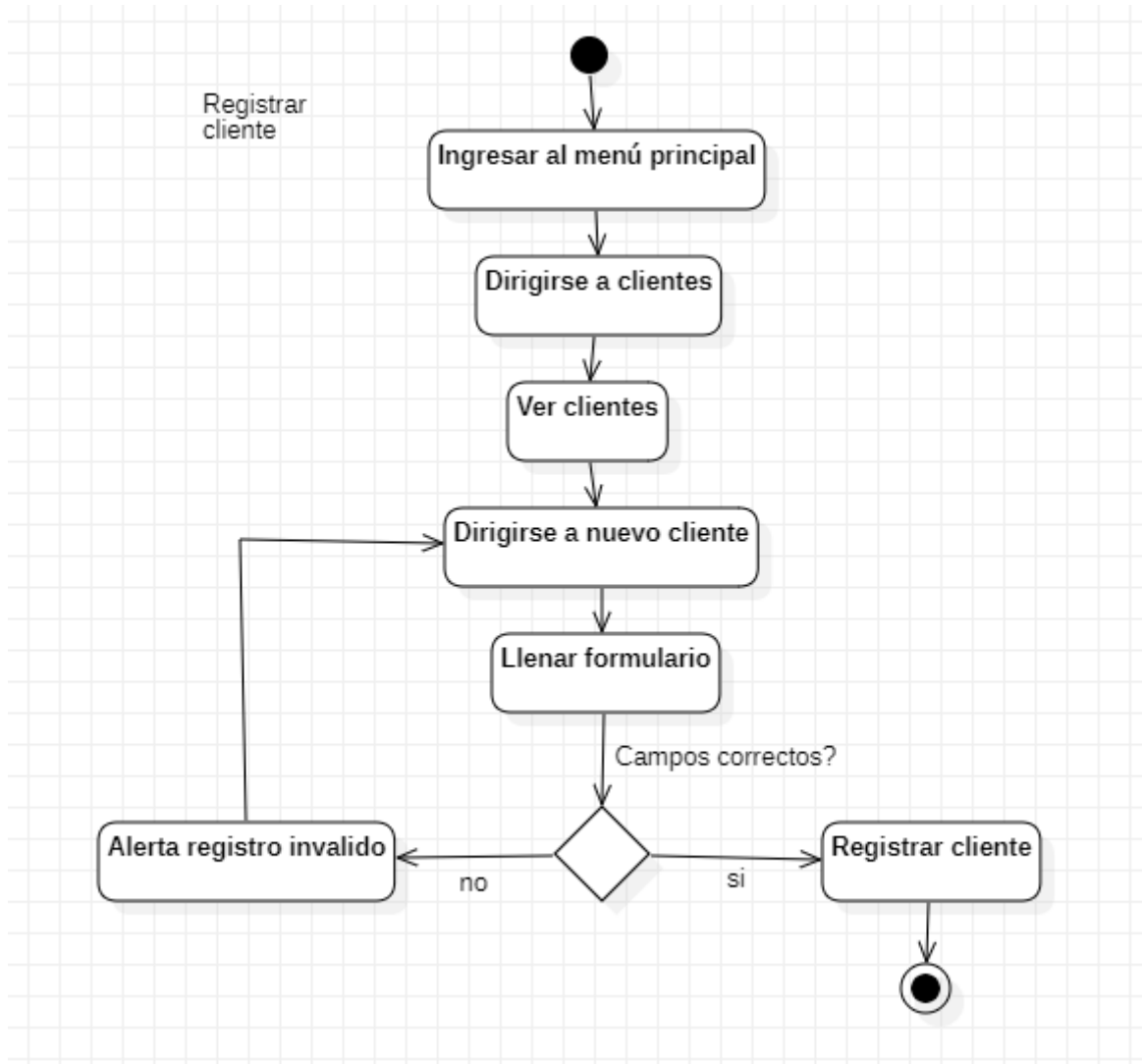


Ilustración 5. Diagrama de actividad (Registro cliente)

En este diagrama se describen las actividades para el registro de un cliente, para esto se debe ingresar al menú principal, dirigirse a clientes, y una vez dentro se podrá dirigir a nuevo cliente en el cual se le mostrara un formulario para registrarlo, aquí se comprobarán si los datos son correctos para después confirmar mediante una alerta el registro o informar algún error.

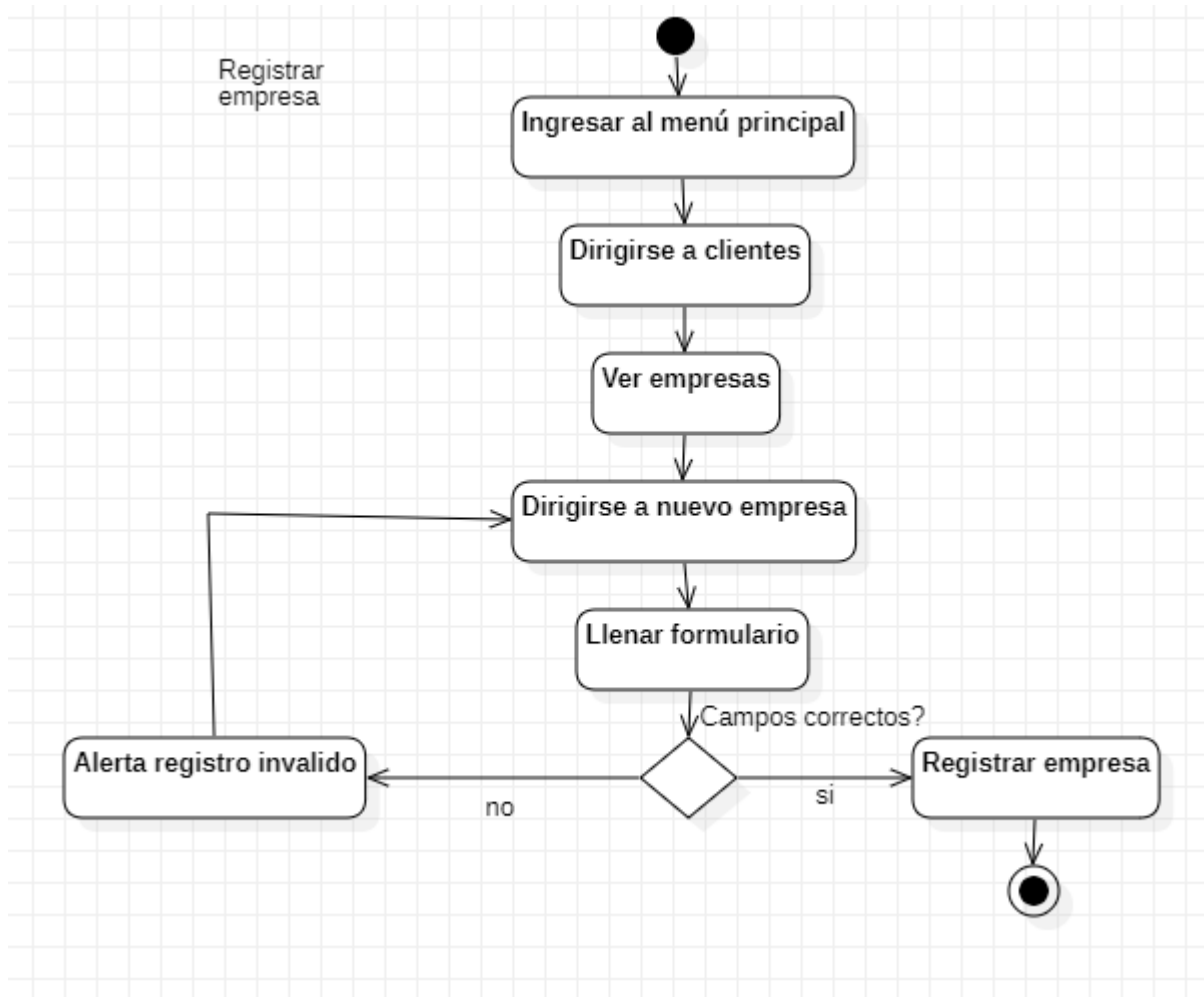


Ilustración 6. Diagrama de actividad (Registro empresa)

En este diagrama se describen las actividades para el registro de una empresa, para esto se debe ingresar al menú principal, dirigirse a clientes, y una vez dentro debe seleccionar ver empresas, donde podrá dirigirse a nueva empresa en la cual se le mostrara un formulario para registrarla, aquí se comprobarán si los datos son correctos para después confirmar mediante una alerta el registro o informar algún error.

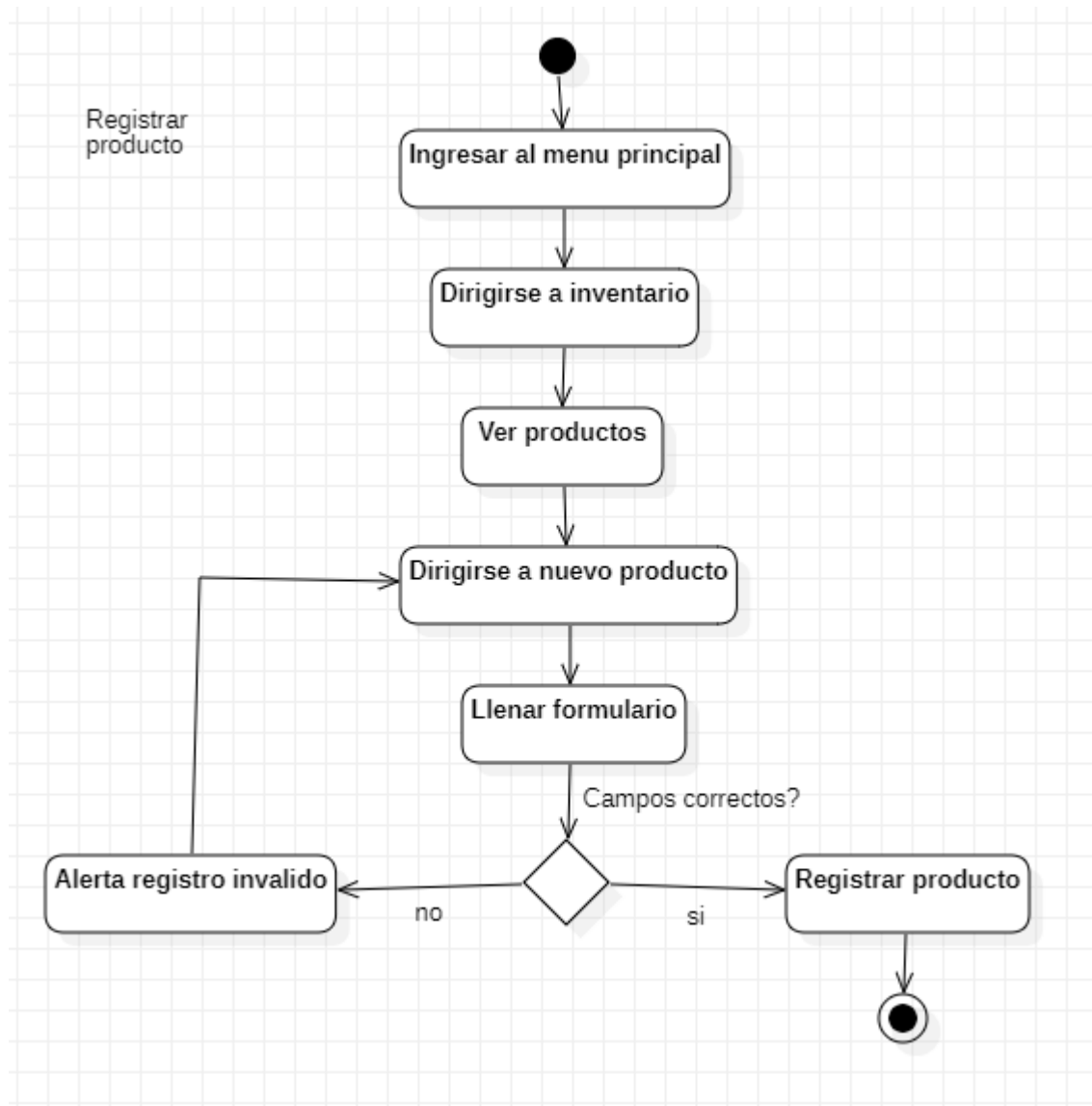


Ilustración 7. Diagrama de actividad (Registro producto)

En este diagrama se describen las actividades para el registro de un producto, para esto se debe ingresar al menú principal, dirigirse a inventario, y una vez dentro se podrá dirigir a nuevo producto en el cual se le mostrara un formulario para registrarlo, aquí se comprobarán si lo datos son correctos para después confirmar mediante una alerta el registro o informar algún error.

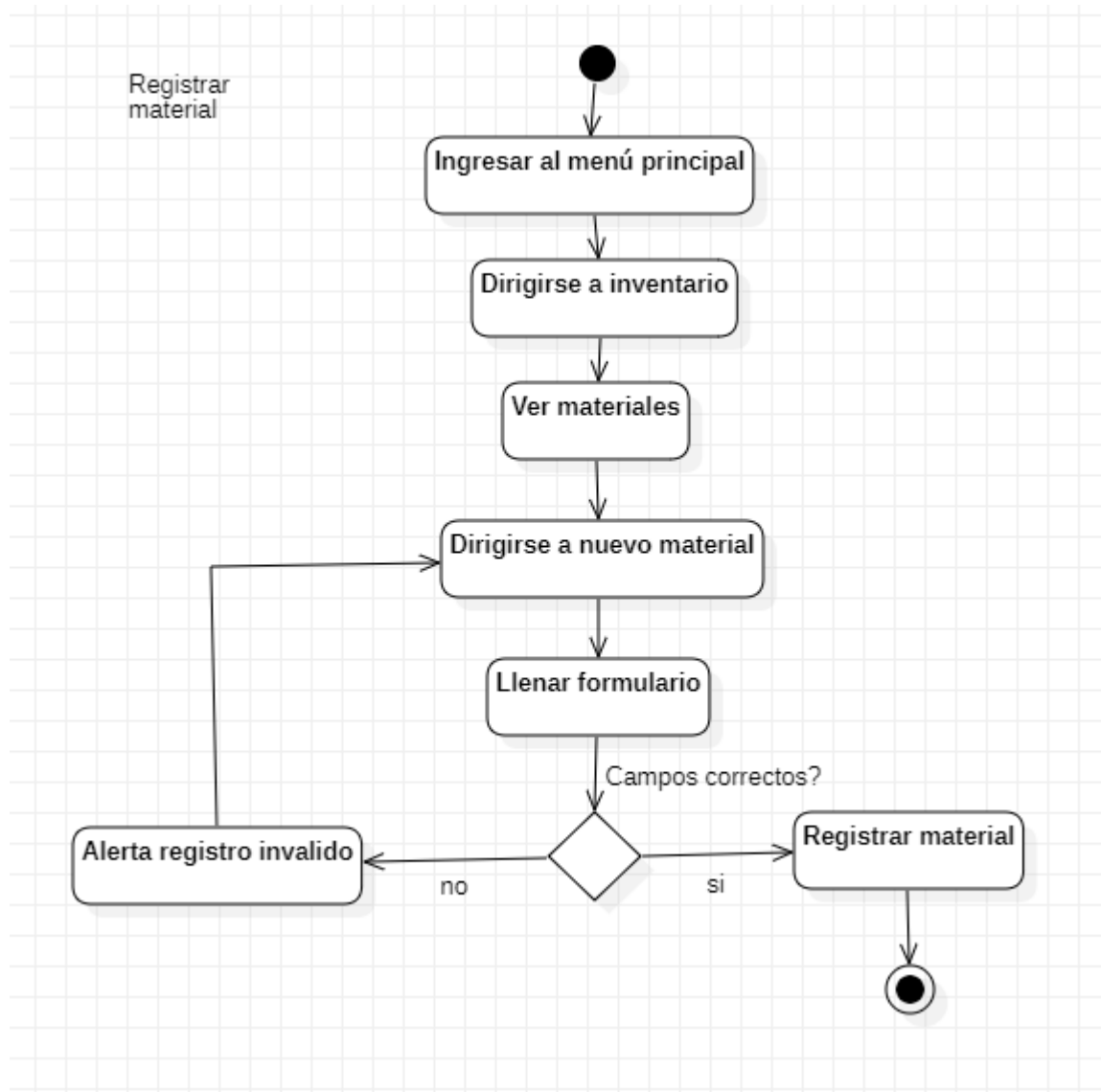


Ilustración 8. Diagrama de actividad (Registro material)

En este diagrama se describen las actividades para el registro de un material, para esto se debe ingresar al menú principal, dirigirse a inventario, y una vez dentro debe seleccionar ver materiales, donde podrá dirigirse a nuevo material en el cual se le mostrara un formulario para registrarlo, aquí se comprobarán si los datos son correctos para después confirmar mediante una alerta el registro o informar algún error.

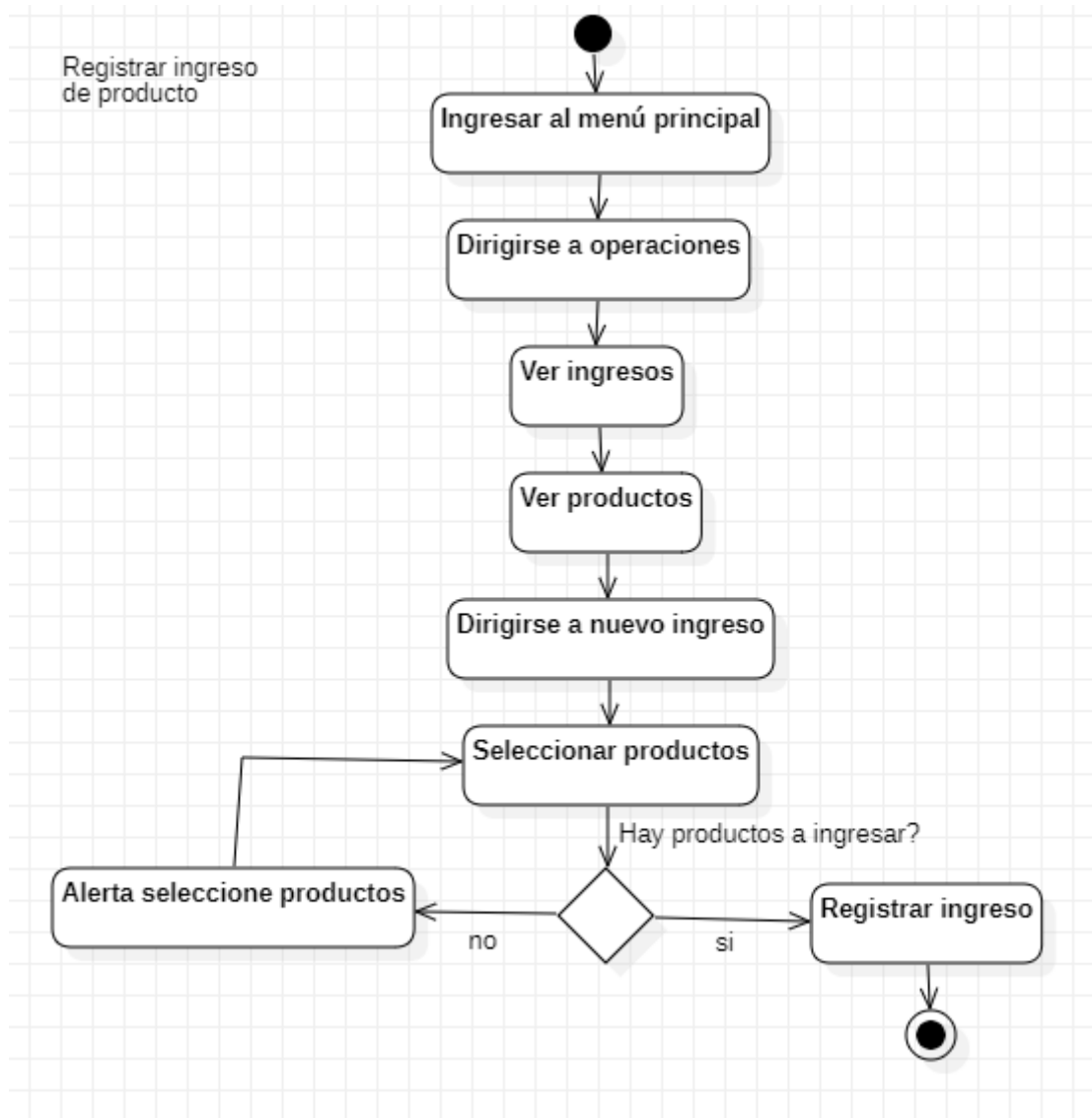


Ilustración 9. Diagrama de actividad (Registro ingreso de producto)

En este diagrama se describen las actividades para el registro de ingresos de productos, para esto se debe ingresar al menú principal, dirigirse a operaciones, luego debe entrar a ver ingresos, y una vez dentro debe seleccionar ver productos, donde podrá dirigirse a nuevo ingreso en el cual se le mostrara una ventana en la cual podrá seleccionar los productos a ingresar junto a sus cantidades, luego de dar a realizar el ingreso se comprobarán si los datos son correctos para después confirmar mediante una alerta el registro o informar algún error.

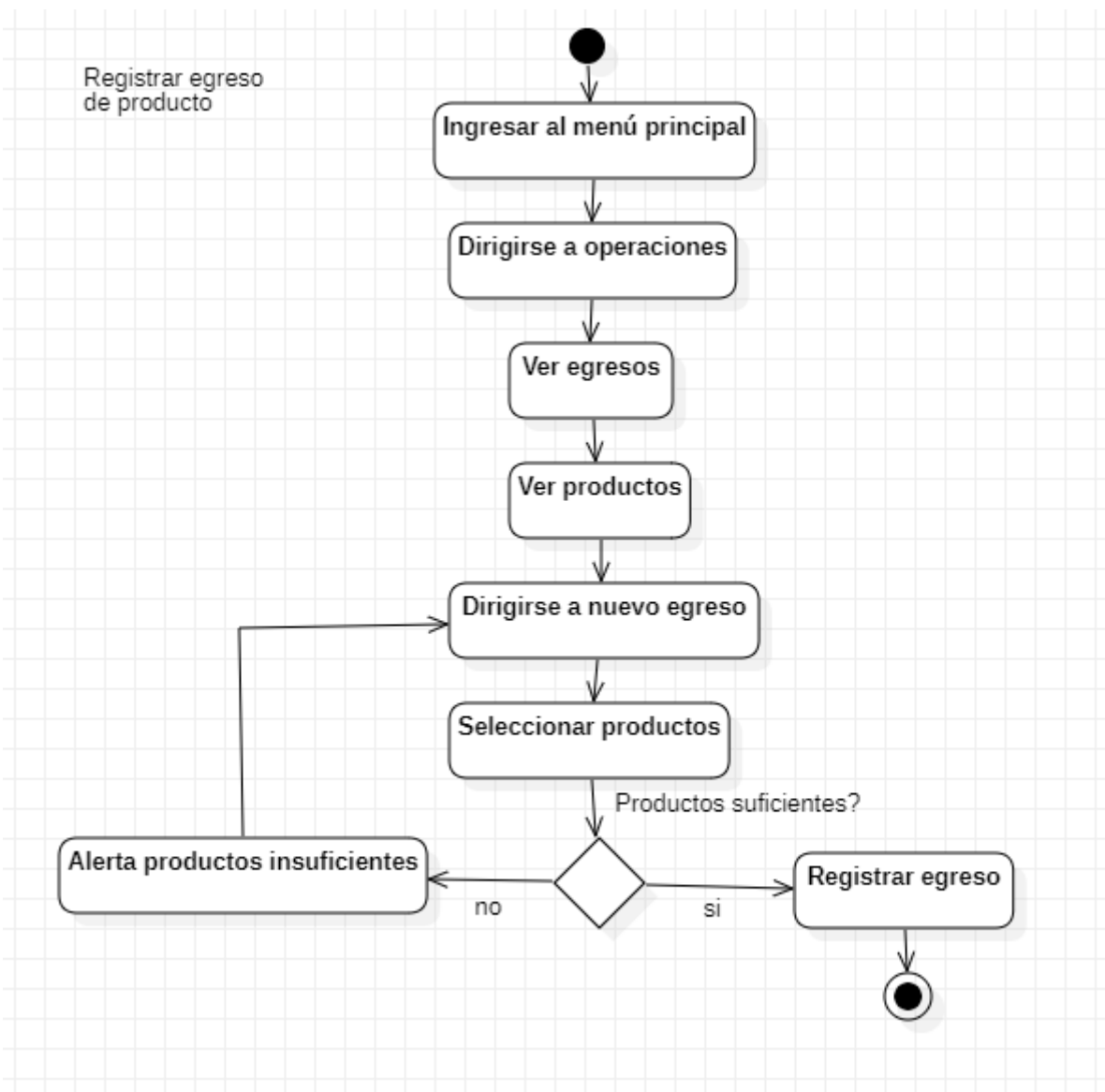


Ilustración 10. Diagrama de actividad (Registro egreso de producto)

En este diagrama se describen las actividades para el registro de egresos de productos, para esto se debe ingresar al menú principal, dirigirse a operaciones, luego debe entrar a ver egresos, y una vez dentro debe seleccionar ver productos, donde podrá dirigirse a nuevo egreso en el cual se le mostrara una ventana en la cual podrá seleccionar los productos a egresar junto a sus cantidades, luego de dar a realizar el egreso se comprobarán si los datos son correctos para después confirmar mediante una alerta el registro o informar algún error.

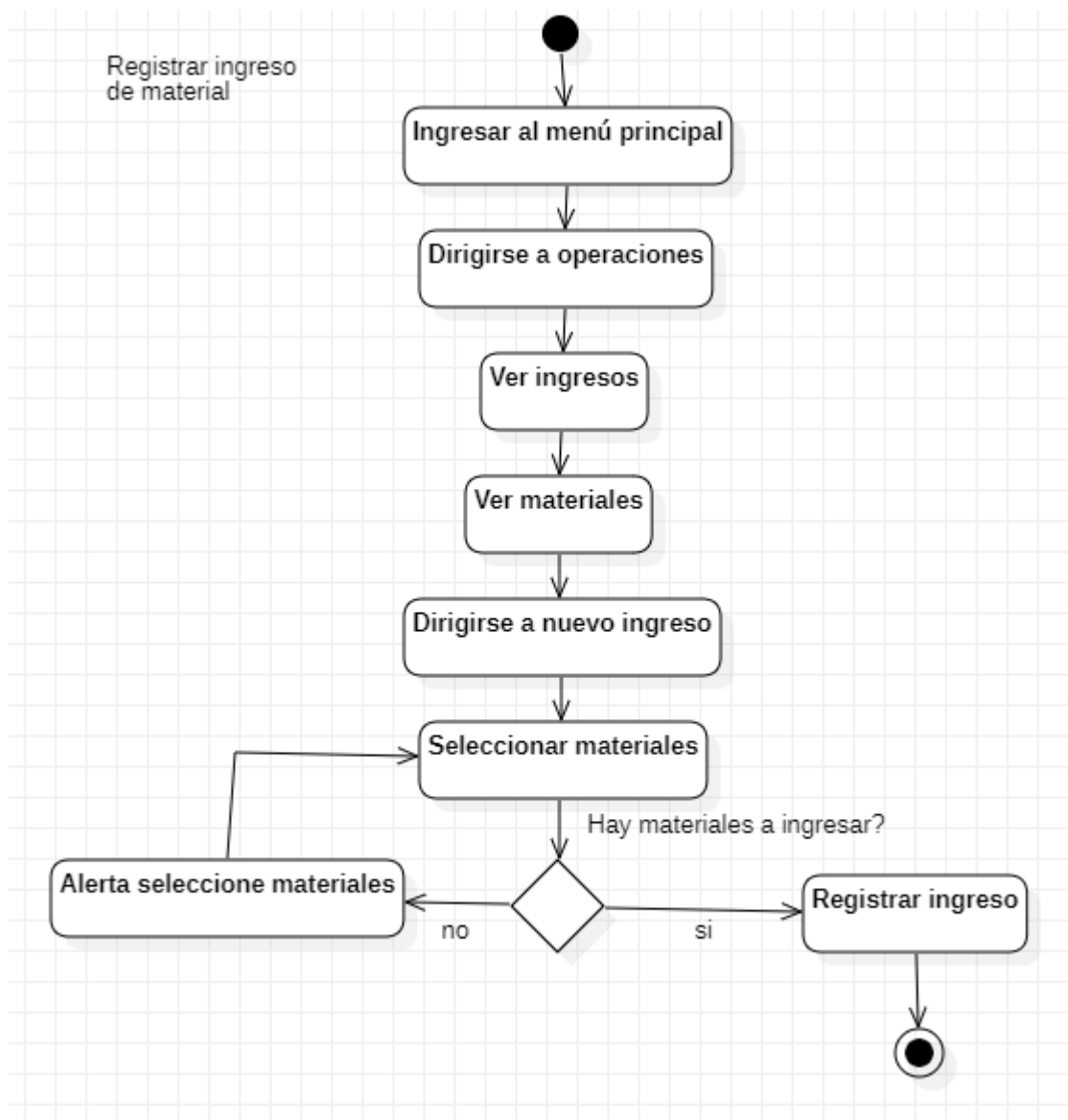


Ilustración 11. Diagrama de actividad (Registro ingreso de material)

En este diagrama se describen las actividades para el registro de ingresos de materiales, para esto se debe ingresar al menú principal, dirigirse a operaciones, luego debe entrar a ver ingresos, y una vez dentro debe seleccionar ver materiales, donde podrá dirigirse a nuevo ingreso en el cual se le mostrara una ventana en la cual podrá seleccionar los materiales a ingresar junto a sus cantidades, luego de dar a realizar el ingreso se comprobarán si los datos son correctos para después confirmar mediante una alerta el registro o informar algún error.

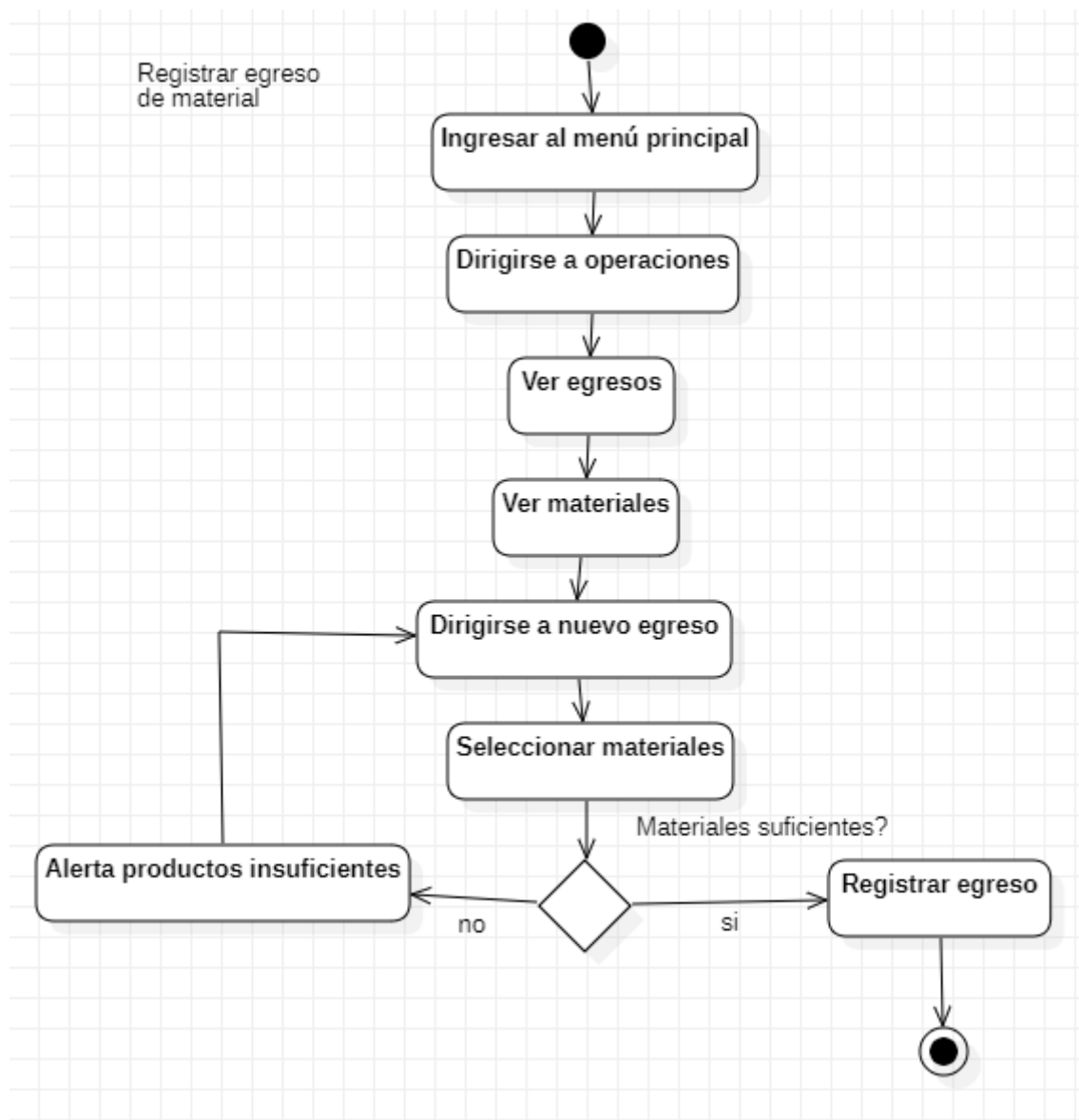


Ilustración 12. Diagrama de actividad (Registro egreso de material)

En este diagrama se describen las actividades para el registro de egresos de materiales, para esto se debe ingresar al menú principal, dirigirse a operaciones, luego debe entrar a ver egresos, y una vez dentro debe seleccionar ver materiales, donde podrá dirigirse a nuevo egreso en el cual se le mostrara una ventana en la cual podrá seleccionar los materiales a egresar junto a sus cantidades, luego de dar a realizar el egreso se comprobarán si los datos son correctos para después confirmar mediante una alerta el registro o informar algún error.

3.2. Analizar las diferentes metodologías de desarrollo a través de un cuadro comparativo para la aplicación web de escritorio de gestión de inventario.

A continuación, se realizará una comparación entre dos de las metodologías ágiles de desarrollo más conocidas, las cuales son SCRUM y XP (Extreme Programming), esta comparativa se hace con el fin de seleccionar cuál de las dos será aplicada en el presente trabajo.

Primeramente, se realizará una descripción sobre en qué consiste cada metodología, luego se mostrará en un cuadro donde se comparará algunas características de cada una. Posteriormente, se hará una conclusión sobre cual metodología se escogió para el desarrollo del proyecto.

3.2.1. SCRUM

Scrum es un marco que permite el trabajo colaborativo entre equipos. Normalmente son los equipos de desarrollo de software los que utilizan con mayor frecuencia este tipo de scrum, pero debido a sus principios y lecciones se pueden aplicar a todo tipo de trabajo en equipo. Aunque se considera a menudo un marco de gestión de proyectos ágil, scrum incluye un conjunto de reuniones, herramientas y funciones que, de forma coordinada, ayudan a los equipos a estructurar y gestionar su trabajo. (Atlassian, 2018)

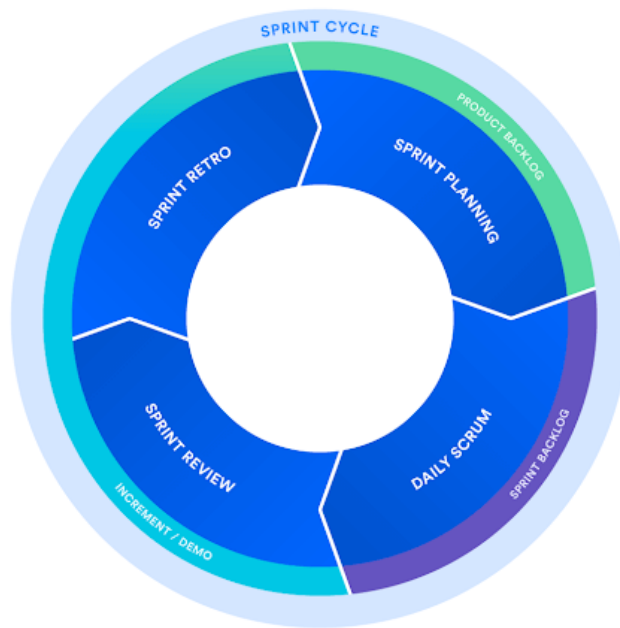


Ilustración 13. Ciclo de sprint Tomado de <https://www.atlassian.com/es/agile/scrum>

En esta imagen se puede observar el funcionamiento del ciclo de un sprint de scrum, en donde se comienza con el sprint planning en el cual se planifica el trabajo que se va a realizar en dicho sprint, luego se comienza a trabajar diariamente en las tareas asignadas para lo cual se realiza un scrum diario o daily scrum el objetivo de esta reunión es que se adecue el objetivo y se disponga de un plan para el trabajo de ese día. Al finalizar el sprint se realiza una revisión o sprint review en la cual se revisa el incremento realizado y se expone a las partes interesadas, posterior a esto se realiza un sprint retro en donde se documentó y analiza que ha funcionado y que no ha funcionado en el sprint.

3.2.1.1. Artefactos de scrum

Scrum tiene tres artefactos los cuales son, (a) Product Backlog: este contiene una lista con las tareas principales del trabajo, la cual se compone de funciones, requisitos, mejoras y correcciones. El propietario del producto está constantemente revisando, cambiando las

prioridades y realizando el mantenimiento del backlog del producto, ya que, a medida que sabemos más o que cambia el mercado, es posible que los elementos ya no sean relevantes o que los problemas se solucionen de otras maneras. (b) Sprint Backlog: este contiene una lista de elementos llamadas historias de usuario las cuales son seleccionadas para implementarse en el ciclo actual de un sprint. Como se mencionó anteriormente en el sprint planning se escogen los elementos que se trabajaran en el sprint backlog del product backlog. (c) Increment: este es el producto final utilizable de un sprint el cual es la demostración de la finalización de un sprint. (Atlassian, 2018)

3.2.2. XP (Extreme Programming)

Se considera como una metodología de desarrollo de software ágil, creada específicamente para promover la aplicación de prácticas de ingeniería apropiadas para la creación de software. Tiene como objetivo principal que un equipo de desarrollo pueda producir software de mejor calidad de forma constante y a su vez busca promover una buena calidad de vida para el equipo. (OpenWebinars, 2020)

3.2.2.1. Valores, principios y prácticas

En la metodología XP es necesario entender los valores, principios y prácticas que lo componen para poder aplicarla correctamente, estos son:

Valores

Existen 5 valores de XP los cuales son, (a) Comunicación: este valor hace referencia a la importancia de una comunicación apropiada, donde se pueda discutir y expresar ideas mediante herramientas prácticas. (b) Simplicidad: este valor se basa en realizar únicamente lo

necesario, poder hacer el trabajo de la manera más simple para que de esta forma sea más fácil de mantener. (c) Retroalimentación: la retroalimentación continua es fundamental en un trabajo en equipo debido a que permite descubrir posibles fallos o realizar mejoras en el trabajo a futuro. (d) Coraje: este hace referencia al valor que se necesita valor para plantear soluciones a los problemas que reducen la eficacia de su equipo, logrando que se dejen de hacer cosas que no funcionan. (e) Respeto: el respeto es quizás el valor más importante debido a que el equipo debe respetarse mutuamente, para de esta manera poder comunicarse entre sí, proporcionar y aceptar comentarios que beneficien su relación. (OpenWebinars, 2020)

Principios

En la metodología XP se encuentran 15 principios los cuales permiten la mejora en la toma de decisiones, pero entre estos resaltan 5 los cuales son llamados principios core, los cuales son: retroalimentación rápida, asumir simplicidad, cambio incremental, abrazar el cambio y trabajo de calidad.

Prácticas

Por último, encontramos las prácticas de la metodología XP, las cuales proponen buenas prácticas de planificación, organización, comunicación y de ingeniería de software, logrando así junto a los valores y principios poder crear una cultura de equipo de excelencia. A continuación, se mencionarán las prácticas de XP más populares.

El juego de planificación

Esta práctica tiene como objetivo compartir las responsabilidades de planificación entre el equipo y el cliente, por lo cual se requiere de una fuerte participación del cliente en el proceso de planificación.

Pequeños releases

Esta práctica hace referencia a que el equipo de desarrollo debe ser capaz de realizar entregas de versiones iterativas del sistema al cliente con frecuencia.

Metáforas

Esta práctica se basa en tener la capacidad de explicar el diseño del sistema a nuevas personas por medio de metáforas en vez de pedirles que lean una gran cantidad de documentación.

Desarrollo guiado por pruebas

Esta práctica es un enfoque el cual combina 2 practicas las cuales son primero escribir la prueba y luego mejorarla a través de la refactorización, permitiendo así crear código de calidad.

3.2.3. Comparación entre SCRUM y XP

Luego de conocer el funcionamiento de cada una de las metodologías, se procederá a realizar un cuadro en el que se describirán algunas características de cada una donde se podrá ver las diferencias que existen entre estas.

Tabla 20. Comparación Scrum y XP

SCRUM		XP
Se centra en	En esta metodología de desarrollo ágil podemos ver que se centra en la administración del proyecto.	En esta metodología de desarrollo ágil podemos ver que se centra en la programación o la creación del producto.

	SCRUM	XP
Forma de trabajar	En esta metodología cada miembro del equipo trabaja de manera individual.	En esta metodología el trabajo debe realizarse en parejas.
Tiempo de entrega	En scrum el tiempo de cada sprint puede tener una duración entre 1 a 4 semanas.	En XP las entregas son realizadas en un plazo de 1 a 3 semanas.
Modificación en las entregas	Al realizar una entrega de las tareas de un sprint y estas hayan sido aprobadas por el product owner, dichas tareas ya no serán modificadas en los siguientes sprints, sino que se apartan.	Al realizar una entrega al finalizar, las tareas que funcionen correctamente y hayan sido aprobadas, puede que después puedan ser modificadas.
Orden de prioridades	En scrum se sigue el orden de prioridades marcadas por el product owner en el sprint backlog, aunque se pueden cambiar las prioridades si presenta una mejora para el desarrollo de las actividades.	En XP se debe seguir estrictamente el orden de prioridades de las tareas definidas por el cliente, sin oportunidad a realizar cambios en las mismas.

Una vez visto el funcionamiento de cada metodología y las diferencia que existen entre estas nos damos cuenta que, aunque se encuentran ciertas diferencias como se puede ver en la

forma de trabajar, ya que en Scrum se realiza un trabajo de manera individual y en XP se debe realizar un trabajo en parejas, las dos también cuentan con ciertas similitudes en su funcionamiento como lo puede ser el control de las entregas debido a que manejan iteraciones para la realización de las mismas, y como el plazo de dichas entregas puede llegar a tener una duración similar. Sin embargo, para la realización de este proyecto se optará por el uso de la metodología Scrum debido a que en la metodología XP es mucho más rígida, debido a la gran cantidad de reglas y pautas estrictas que delimitan la manera en que se puede realizar el desarrollo. Además, de que se puede aplicar Scrum para cualquier proceso en donde se requiera una iteración y aportes por parte del cliente, mientras que XP solo se podría usar en la programación o creación del producto.

3.3. Diseñar la aplicación web de escritorio mediante la plataforma Figma para la gestión de inventario de la fábrica.

3.3.1. Modelo relacional base de datos

El modelo relacional proporciona una manera de poder observar de las entidades que conforman una base de datos, este modelo se basa en el uso de relaciones las cuales se pueden considerar como un conjunto de datos a los que se le suele llamar tuplas. El modelo de datos relacionales proporcionó una forma estándar de representar y consultar datos que cualquier aplicación podría usar. Desde el principio, los desarrolladores reconocieron que la principal ventaja del modelo de base de datos relacional era el uso de tablas. Las tablas son una forma intuitiva, eficiente y flexible de almacenar y acceder a información estructurada. (Oracle, s.f.)

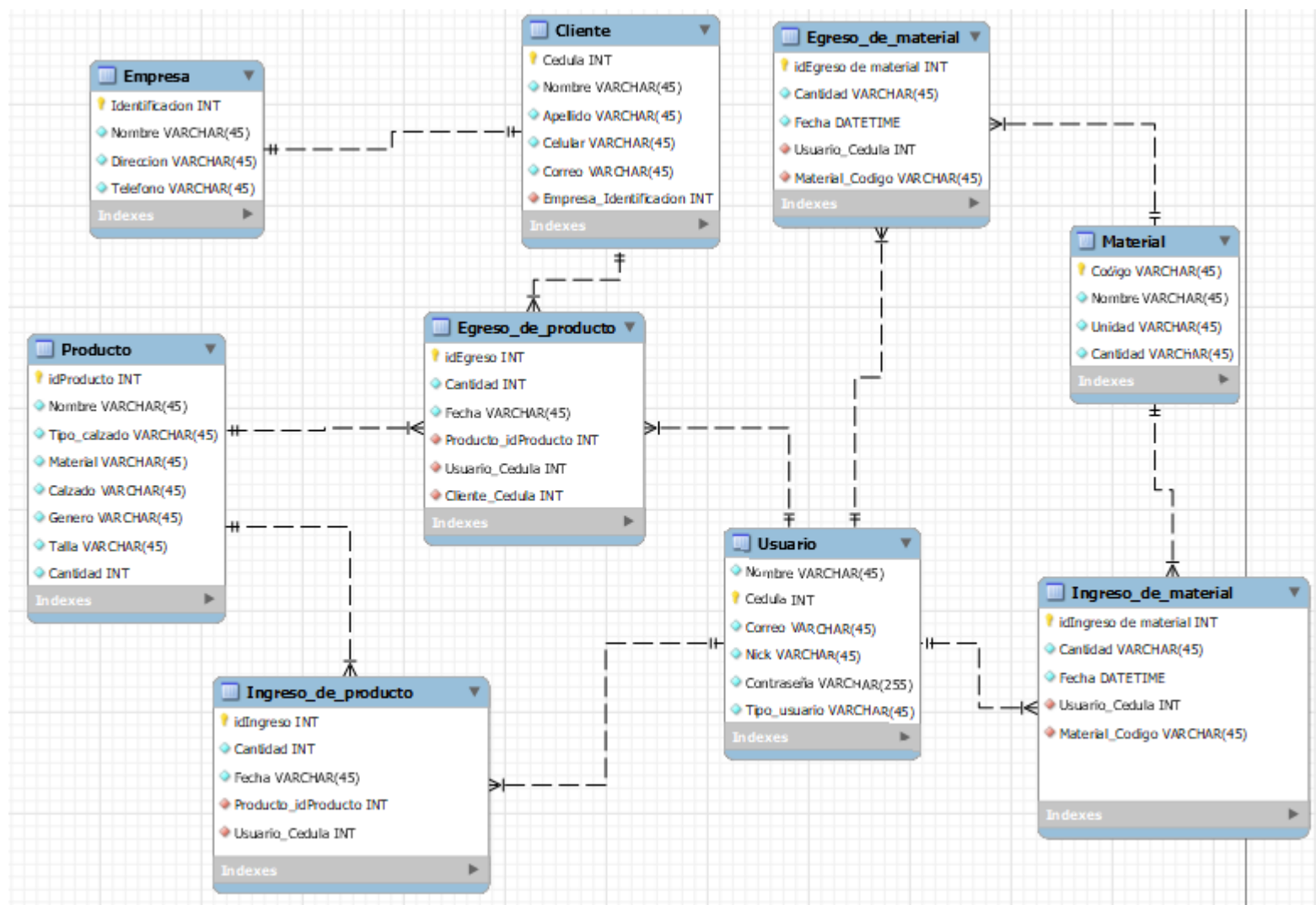


Ilustración 14. Modelo relacional

3.3.2. Diccionario base de datos

Tabla 21. Tabla usuario

Tabla usuario			
Nombre de atributo	Tipo	Restricción	Descripción
Cedula	Int	Primary key	Contiene la cedula que identifica al usuario.
Nombre	Varchar(45)		Contiene el nombre del usuario.
Correo	Varchar(45)		Contiene el correo del usuario.
Nick	Varchar(45)	Unique	Contiene el usuario que le permite iniciar sesión en el sistema.
Contraseña	Varchar(255)		Contiene la contraseña que le permite iniciar sesión en el sistema.
Tipo de usuario	Varchar(45)		Contiene el tipo del usuario ya sea Administrador u operador.

Tabla 22. Tabla empresa

Tabla empresa			
Nombre de atributo	Tipo	Restricción	Descripción
Identificación	Int	Primary key	Contiene un id que identifica a la empresa.
Nombre	Varchar(45)		Contiene el nombre de la empresa.
Dirección	Varchar(45)		Contiene la dirección de la empresa.
Teléfono	Varchar(45)		Contiene el teléfono de la empresa.

Tabla 23. Tabla cliente

Tabla cliente			
Nombre de atributo	Tipo	Restricción	Descripción
Cedula	Int	Primary key	Contiene la cedula que identifica al cliente
Nombre	Varchar(45)		Contiene el nombre del cliente.
Apellido	Varchar(45)		Contiene el apellido del cliente
Celular	Varchar(45)		Contiene el celular del cliente.
Correo	Varchar(45)		Contiene el correo del cliente.
Empresa_identificación	Int	Foreign key	Contiene el id de identificación de la empresa a la que pertenece el cliente.

Tabla 24. Tabla producto

Tabla producto			
Nombre de atributo	Tipo	Restricción	Descripción
Id	Int	Primary key	Contiene el identificador del producto.
Nombre	Varchar(45)		Contiene el nombre del producto.
Tipo calzado	Varchar(45)		Contiene el tipo de producto el cual hace referencia a la parte del calzado (Corte, suela, etc.).
Material	Varchar(45)		Contiene el material del producto.
Calzado	Varchar(45)		Contiene el calzado para cual está hecho el producto.

Genero	Varchar(45)		Contiene el genero del calzado para el cual está hecho el producto.
Talla	Varchar(45)		Contiene la talla del producto.
Cantidad	Varchar(45)		Contiene la cantidad disponible de producto.

Tabla 25. Tabla ingreso de producto

Tabla ingreso de producto			
Nombre de atributo	Tipo	Restricción	Descripción
Id	Int	Primary key	Contiene el identificador del ingreso.
Cantidad	Int		Contiene la cantidad que se ingresa al producto
Fecha	Date		Contiene la fecha de ingreso.
Id producto	Int	Foreign key	Contiene la id del producto del ingreso.
Cedula usuario	Int	Foreign key	Contiene la cedula del usuario que realiza el ingreso.

Tabla 26. Tabla egreso de producto

Tabla egreso de producto			
Nombre de atributo	Tipo	Restricción	Descripción
Id	Int	Primary key	Contiene el identificador del egreso.
Cantidad	Int		Contiene la cantidad que se egresa del producto
Fecha	Date		Contiene la fecha de egreso.
Id producto	Int	Foreign key	Contiene la id del producto del egreso.

Cedula usuario	Int	Foreign key	Contiene la cedula del usuario que realiza el egreso.
Cedula cliente	Int	Foreign key	Contiene la cedula del cliente al que se le realiza el egreso.

Tabla 27. Tabla material

Tabla material			
Nombre de atributo	Tipo	Restricción	Descripción
Código	Varchar(45)	Primary key	Contiene el código que identifica al material.
Nombre	Varchar(45)		Contiene el nombre del material.
Unidad	Varchar(45)		Contiene la unidad que representa la cantidad del material.
Cantidad	Varchar(45)		Contiene la cantidad disponible del material.

Tabla 28. Tabla ingreso de material

Tabla ingreso de material			
Nombre de atributo	Tipo	Restricción	Descripción
Id	Int	Primary key	Contiene el identificador del ingreso.
Cantidad	Int		Contiene la cantidad que se ingresa al material.
Fecha	Date		Contiene la fecha de ingreso.
Cedula usuario	Int	Foreign key	Contiene la cedula del usuario que realiza el ingreso.
Código material	Int	Foreign key	Contiene el código del material que se ingresa.

Tabla 29. Tabla egreso de material

Tabla egreso de material			
Nombre de atributo	Tipo	Restricción	Descripción
Id	Int	Primary key	Contiene el identificador del egreso.
Cantidad	Int		Contiene la cantidad que se egresa del material.
Fecha	Date		Contiene la fecha de egreso.
Cedula usuario	Int	Foreign key	Contiene la cedula del usuario que realiza el egreso.
Código material	Int	Foreign key	Contiene el código del material que se egresa.

3.3.3. Wireframes

A continuación, se mostrarán los wireframes del sistema realizados en la plataforma Figma la cual es una herramienta intuitiva y fácil de usar para la creación de prototipos web, los cuales tienen una apariencia muy similar a un producto final desarrollado. Además, se encuentra alojada en un sitio web a diferencia de otras herramientas similares. Aunque Figma funcione en la nube, también cuenta con una aplicación para ejecutarla desde un equipo de cómputo. En las siguientes ilustraciones se podrán observar los prototipos creados para la aplicación web de escritorio (*ver ilustración 15 a la ilustración 30*)

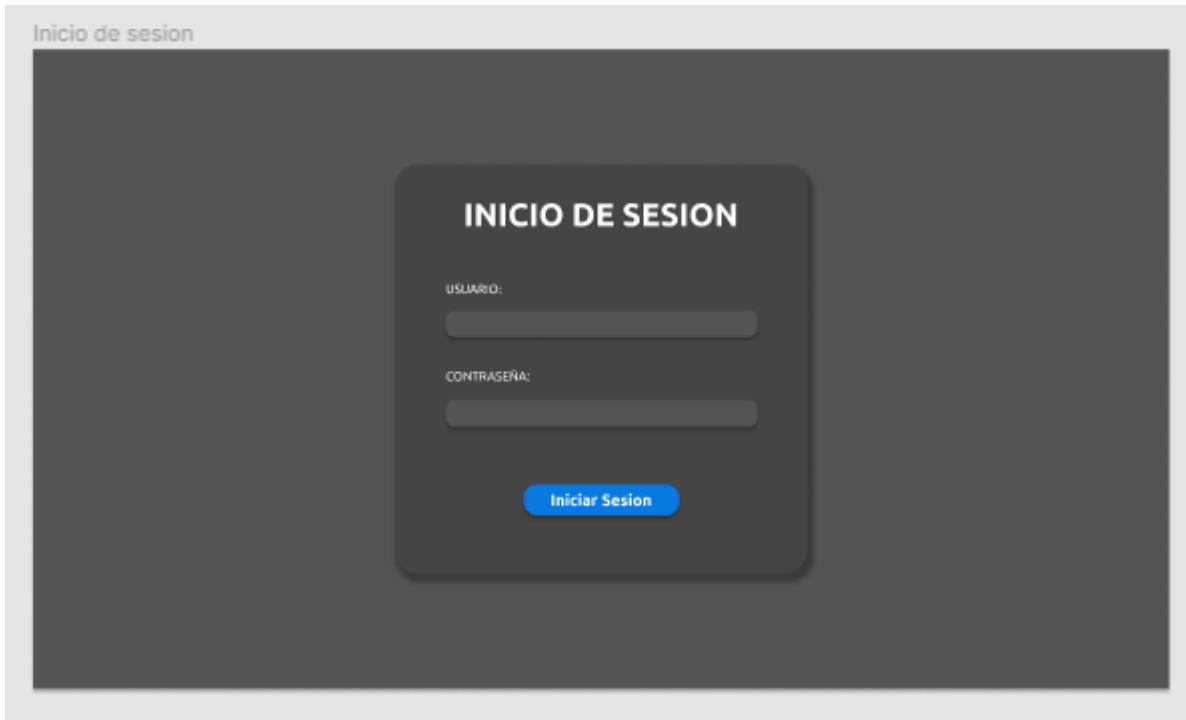


Ilustración 15. Vista inicio de sesión

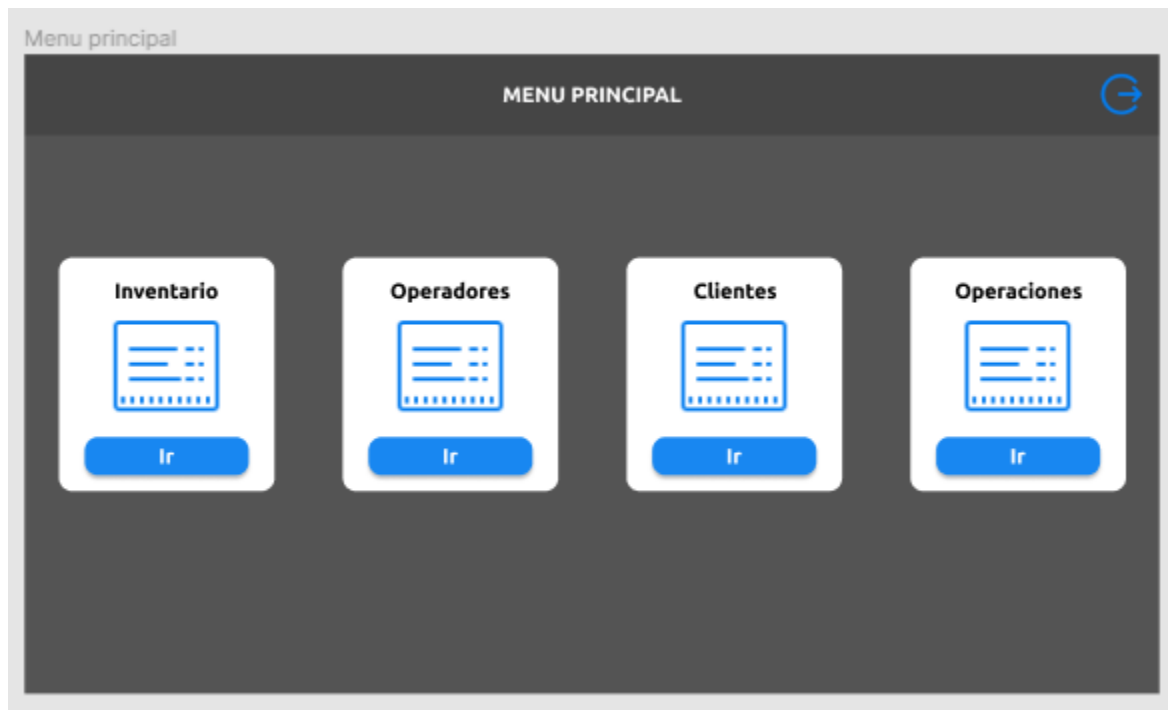


Ilustración 16. Vista menú principal (Administrador)

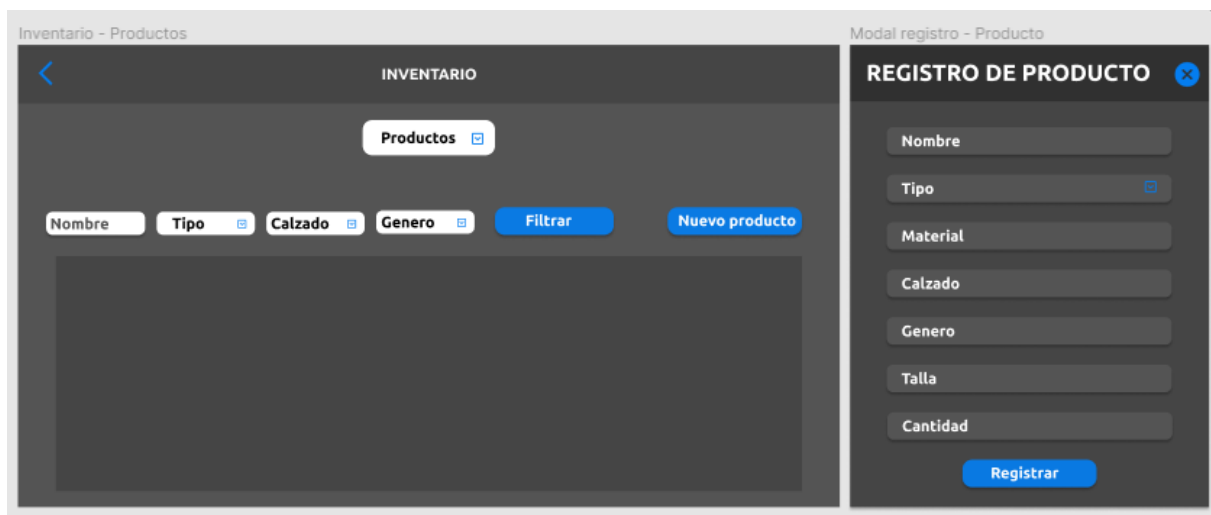


Ilustración 17. Vista inventario (productos)

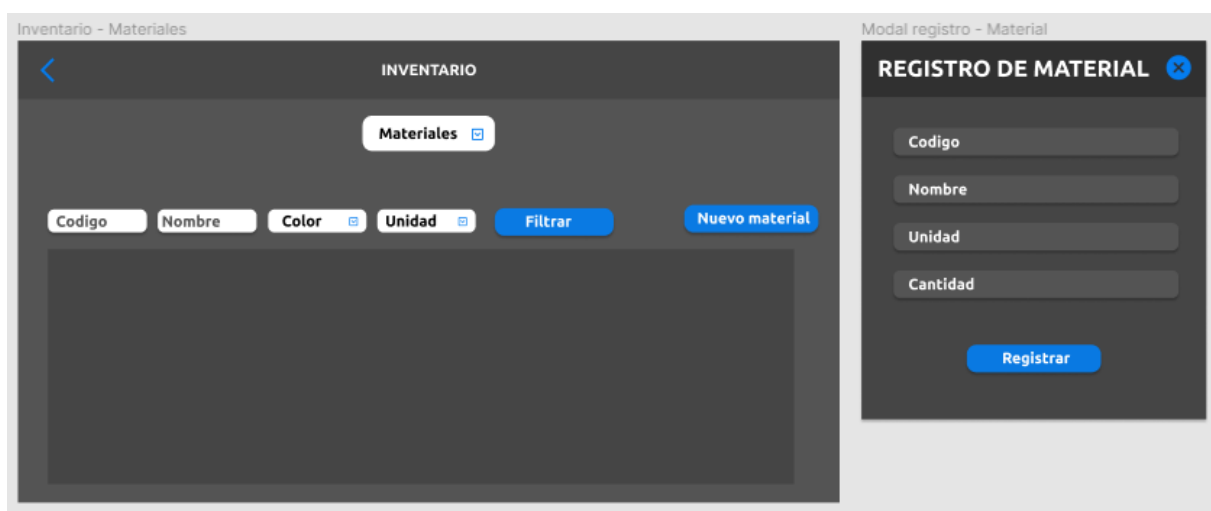


Ilustración 18. Vista inventario (materiales)

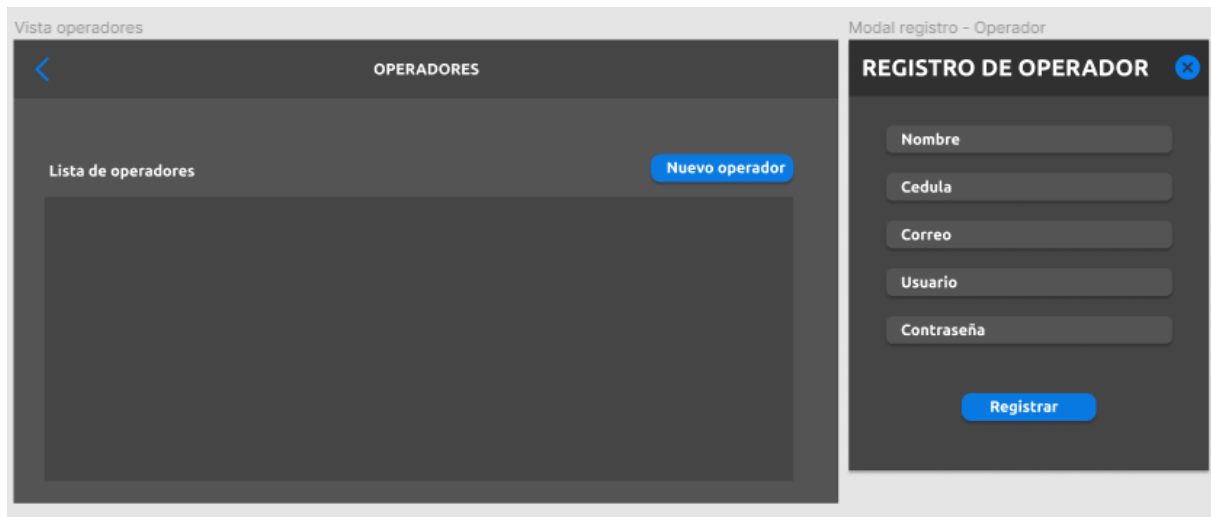


Ilustración 19. Vista operadores (visualización y registro)

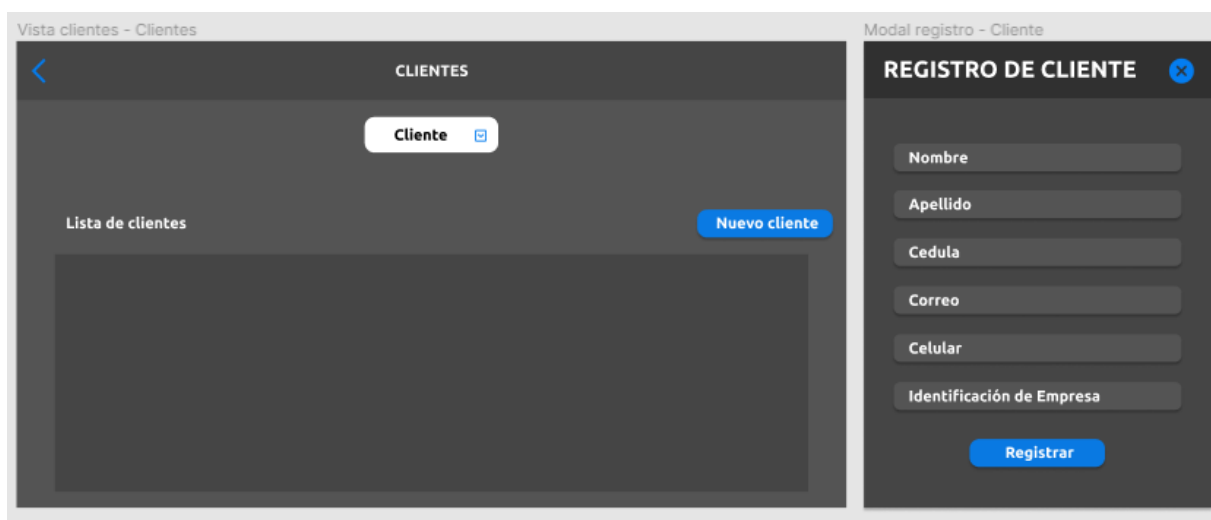


Ilustración 20. Vista de clientes (Visualización y registro)

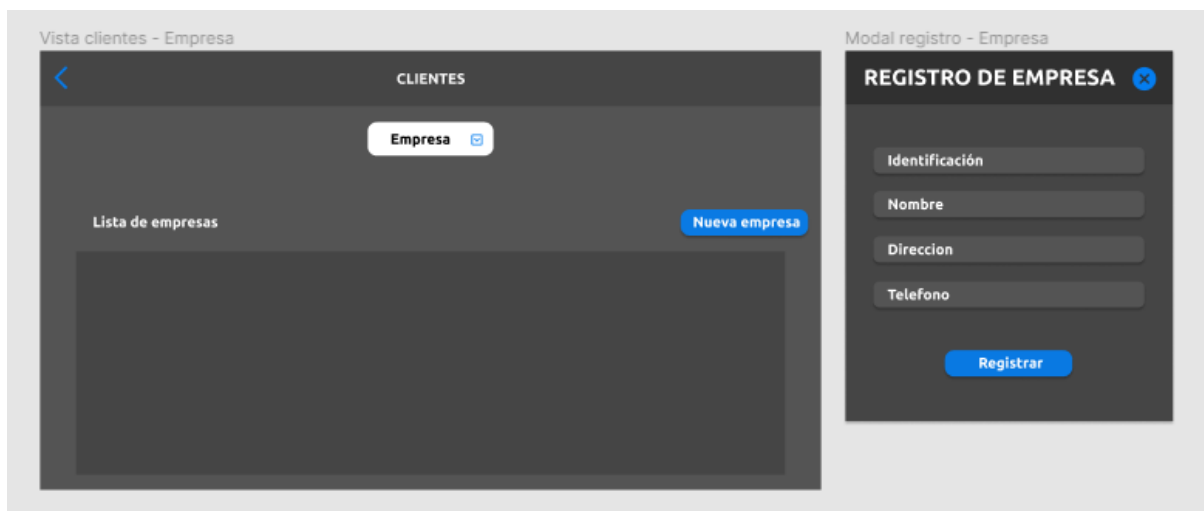


Ilustración 21. Vista de empresas (Visualización y registro)



Ilustración 22. Vista menú de operaciones

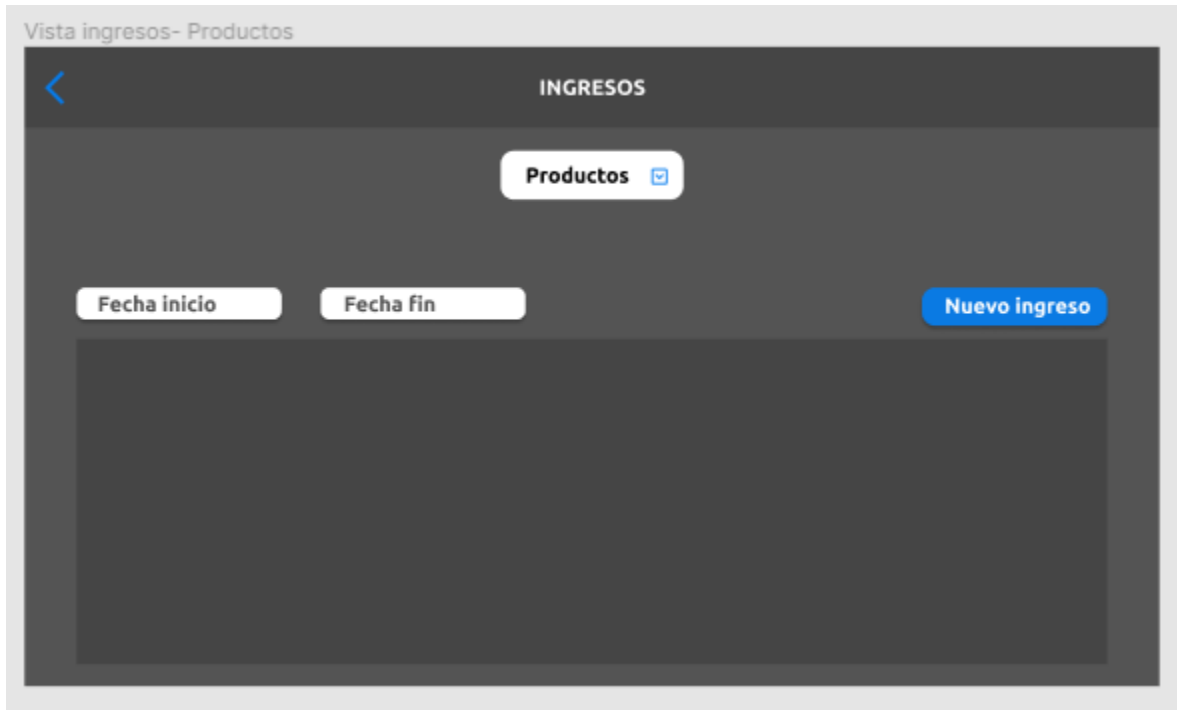


Ilustración 23. Vista de ingresos productos (Visualización)

Vista ingreso producto

<

INGRESO DE PRODUCTOS

Nombre

Tipo 

Calzado 

Genero 

Filtrar

Detalles producto

Registrar ingreso

Ilustración 24. Vista de ingreso productos (Registro)



Ilustración 25. Vista de ingreso materiales (Visualización)



Ilustración 26. Visualización de ingreso materiales (Registro)

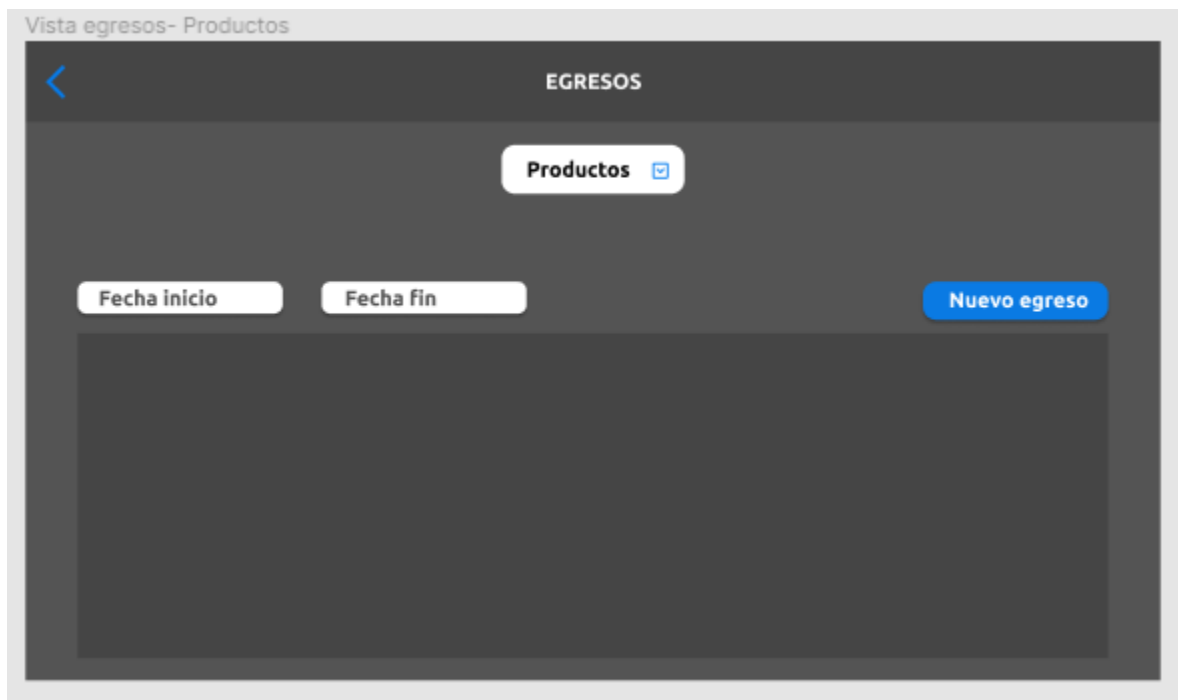


Ilustración 27. Vista de egreso productos (Visualización)

Modal egreso- Producto

EGRESO DE PRODUCTOS

Nombre Tipo Calzado Genero Filtrar

C.C. Cliente

Detalles producto

Registrar egreso

Ilustración 28. Vista de egreso productos (Registro)

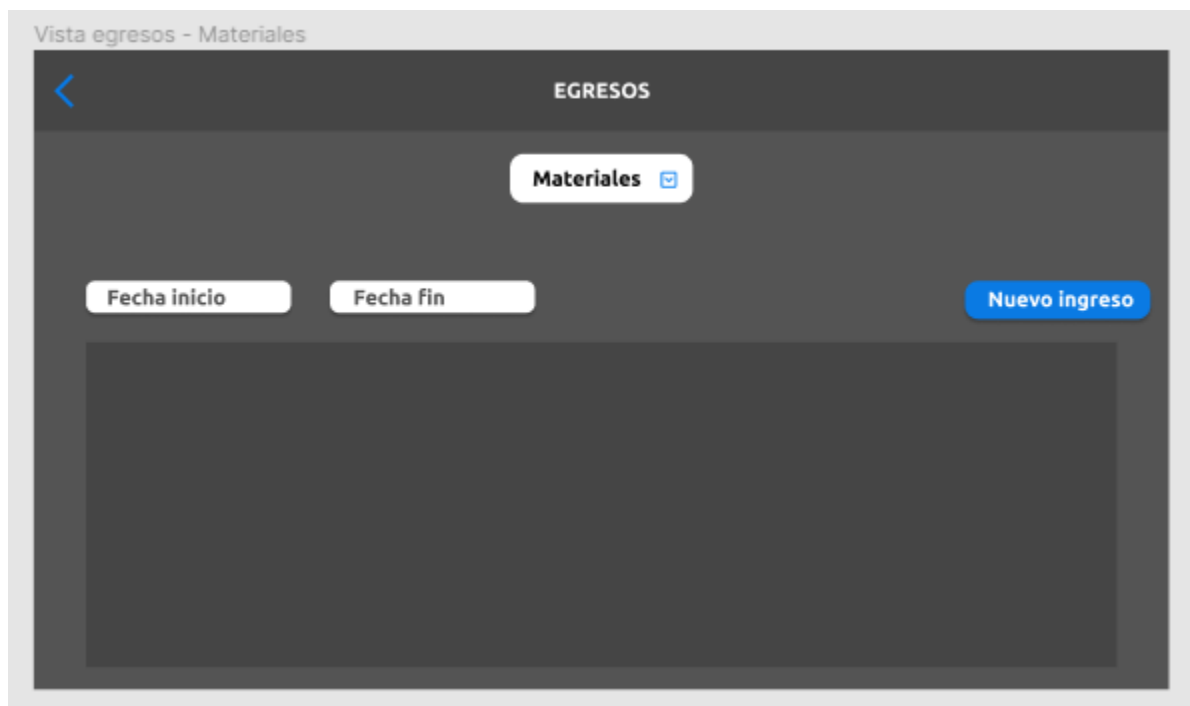


Ilustración 29. Vista de egreso materiales (Visualización)

Modal egreso- Materiales

EGRESO DE MATERIALES

Codigo **Nombre** **Unidad** 

Filtrar

Detalles material

Registrar egreso

Ilustración 30. Vista de egreso materiales (Registro)

CAPITULO 4. METODOLOGÍA PARA DESARROLLO AGIL

A continuación, se presentará la metodología ágil de desarrollo Scrum, donde se incluye el product backlog el cual contiene las etapas de los sprints, así como el sprint backlog en el cual se especifican las tareas correspondiente para posteriormente proceder al desarrollo de la aplicación web de escritorio.

4.1. Roles

En Scrum existen 3 roles que conforman el equipo, los cuales son el Scrum Master, el cual es la persona con mayor conocimiento en aplicar dicha metodología, el Product Owner, el cual es el dueño del producto, y el equipo de desarrollo. A continuación, se mostrará el equipo Scrum del presente trabajo.

Tabla 30. Roles scrum

Rol	Encargado
Scrum Master	Luis David Celis Caicedo
Product Owner	Nancy Yudith Marquez
Developer	Luis David Celis Caicedo

4.2. Product Backlog

A continuación, se mostrarán los ítems planteados en el producto backlog junto con la descripción de cada uno.

Tabla 31. Producto backlog

Ítem	Descripción	Prioridad
1	Diseño de la arquitectura del software Se realizar el diseño de la arquitectura del sistema de información haciendo uso de los diferentes diagramas que son necesarios para el desarrollo del software.	1

2	Desarrollo del frontend de la aplicación	Desarrollo del frontend de la aplicación web de escritorio a través de Angular y Electronjs.	2
3	Base de datos	Creación del modelo relacional de la base de datos y codificación de la base de datos en PostgreSQL.	2
4	Desarrollo del backend de la aplicación	Creación y desarrollo de la API en Node.js la cual contendrá los servicios que se consumirán en el frontend.	2
5	Consumo de los servicios de la API desde Angular	Mediante el módulo HttpClient se consumirán los servicios que han sido creados en la API de esta manera se podrá comunicar el frontend con el backend.	2
6	Realización de pruebas	Por medio de pruebas se verificará que los servicios creados en el backend de la aplicación cumplen con su funcionamiento.	2
7	Manual de usuario	Desarrollo de un manual de usuario de la aplicación web de escritorio.	1

4.3. Sprint Backlog

A continuación, se mostrarán los sprints de scrum donde se especifica el número del sprint, el ítem del product backlog y las tareas desarrolladas para dicho ítem, junto a las fechas de cada tarea.

Tabla 32. Sprint 1

Sprint 1						
Ítem Backlog	Tareas		Fecha Fin	Fecha Fin	Total Días	Completada al finalizar el Sprint
1. Diseño de la arquitectura del software	1	Toma de requerimientos del cliente.	11/07/2022	13/07/2022	3	Si
	2	Establecimiento de requerimientos funcionales, no funcionales y pseudo requerimientos	15/07/2022	22/07/2022	8	Si

	3	Realización de diagrama de casos de uso y diagramas de actividades	23/07/2022	30/07/2022	8	Si
--	---	--	------------	------------	---	----

Tabla 33. Sprint 2

Sprint 2						
Ítem Backlog	Tareas		Fecha Inicio	Fecha Fin	Total Días	Completada al finalizar el Sprint
2. Desarrollo del frontend de la aplicación	1	Diseño de los wireframes en Figma	1/08/2022	10/08/2022	10	Si
	2	Creación del proyecto en Angular y configuración del proyecto con Electronjs	12/08/2022	16/08/2022	5	Si
	3	Creación de los componentes en Angular.	19/08/2022	29/08/2022	11	Si
	4	Creación y configuración de las rutas del proyecto.	30/08/2022	04/09/2022	5	Si

Tabla 34. Sprint 3

Sprint 3						
Ítem Backlog	Tareas		Fecha Inicio	Fecha Fin	Total Días	Completada al finalizar el Sprint
3. Base de datos	1	Diseño del modelo relacional	05/09/2022	08/09/2022	4	Si
	2	Codificación de la base de datos.	09/09/2022	15/09/2022	7	Si
	3	Creación de la base de datos y despliegue en ElephantSQL	16/09/2022	22/09/2022	7	Si

Tabla 35. Sprint 4

Sprint 4						
Ítem Backlog		Tareas	Fecha Inicio	Fecha Fin	Total Días	Completada al finalizar el Sprint
4. Desarrollo del backend de la aplicación	1	Creación y configuración de la API en Node.js	23/09/2022	26/09/2022	4	Si
	2	Creación de las estructura del proyecto e instalación de las dependencias.	27/09/2022	28/09/2022	2	Si
	3	Codificación del repositorio y comprobación de conexión con la base de datos	03/10/2022	05/10/2022	3	Si
	4	Codificación de los ORM de las entidades.	06/10/2022	12/10/2022	7	Si
	5	Codificación de los servicios que se comunican con los ORM.	13/10/2022	17/10/2022	5	Si
	6	Codificación de los controladores de la API y configuración de las rutas.	18/10/2022	23/10/2022	6	Si

Tabla 36. Sprint 5

Sprint 5						
Ítem Backlog		Tareas	Fecha Inicio	Fecha Fin	Total Días	Completada al finalizar el Sprint
5. Consumo de los servicios de	1	Instalación de los módulos correspondient	24/10/2022	24/10/2022	1	Si

la API desde Angular		es para el consumo de servicios.				
	2	Creación y codificación de los servicios en Angular.	25/10/2022	01/11/2022	8	Si
	3	Realización de los consumos de los servicios entre los componentes y los servicios creados.	01/11/2022	08/11/2022	8	Si

Tabla 37. Sprint 6

Sprint 6						
Ítem Backlog		Tareas	Fecha Inicio	Fecha Fin	Total Días	Completada al finalizar el Sprint
6. Realización de pruebas	1	Instalación de la dependencia para el test de los endpoints en el backend.	09/11/2022	09/11/2022	1	Si
	2	Codificación de las pruebas de los endpoints.	10/11/2022	15/11/2022	6	Si
	3	Realización de las pruebas	16/11/2022	19/11/2022	4	Si

Tabla 38. Sprint 7

Sprint 7					
Ítem Backlog	Tareas	Fecha Inicio	Fecha Fin	Total Días	Completada al finalizar el Sprint

7. Manual de usuario	1	Desarrollo del manual de usuario del sistema el cual servirá como guía para el correcto manejo de la aplicación web de escritorio.	21/11/2022	27/11/2022	7	Si
-----------------------------	----------	--	------------	------------	---	----

CAPITULO 5. DESARROLLO DE LA APLICACIÓN WEB DE ESCRITORIO

5.1. Implementar la aplicación web de escritorio para la gestión de inventario de la fábrica.

5.1.1. Arquitectura hexagonal para el backend

Para el desarrollo del backend en Nodejs se hizo uso de la arquitectura hexagonal, para esto se debe tener en cuenta como se presenta el backend en dicha arquitectura.

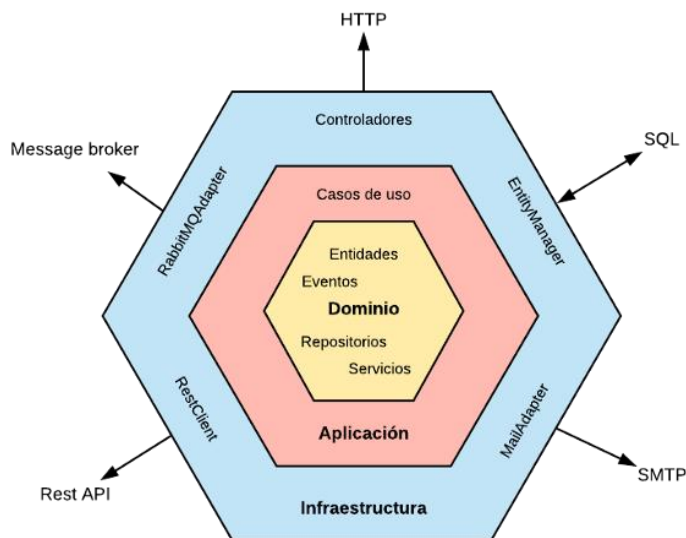


Ilustración 31. Arquitectura hexagonal - Tomado de <https://medium.com/@edusalguero/arquitectura-hexagonal-59834bb44b7f>

Según la ilustración anterior, se puede observar que el núcleo de las capas de la arquitectura hexagonal lo compone el dominio, en el que se encuentra las entidades, eventos, repositorios y servicios. Por lo tanto, se puede deducir que el backend debe tener un diseño orientado al dominio, normalmente al dominio se le define como una esfera de conocimiento, pero en este ámbito del desarrollo hace referencia la esfera entorno a la cual gira la lógica de la aplicación. Dicha lógica de la aplicación es la misma que se puede conocer como la lógica de

negocio, la cual se refiere a las reglas de nivel superior sobre como esta lógica interactúa entre sí para la creación y modificación de datos.

Ahora, una vez que se sabe que se debe orientar el diseño del backend al dominio se realizó un diagrama en el cual se puede ver los componentes del backend y como estos interactúan.

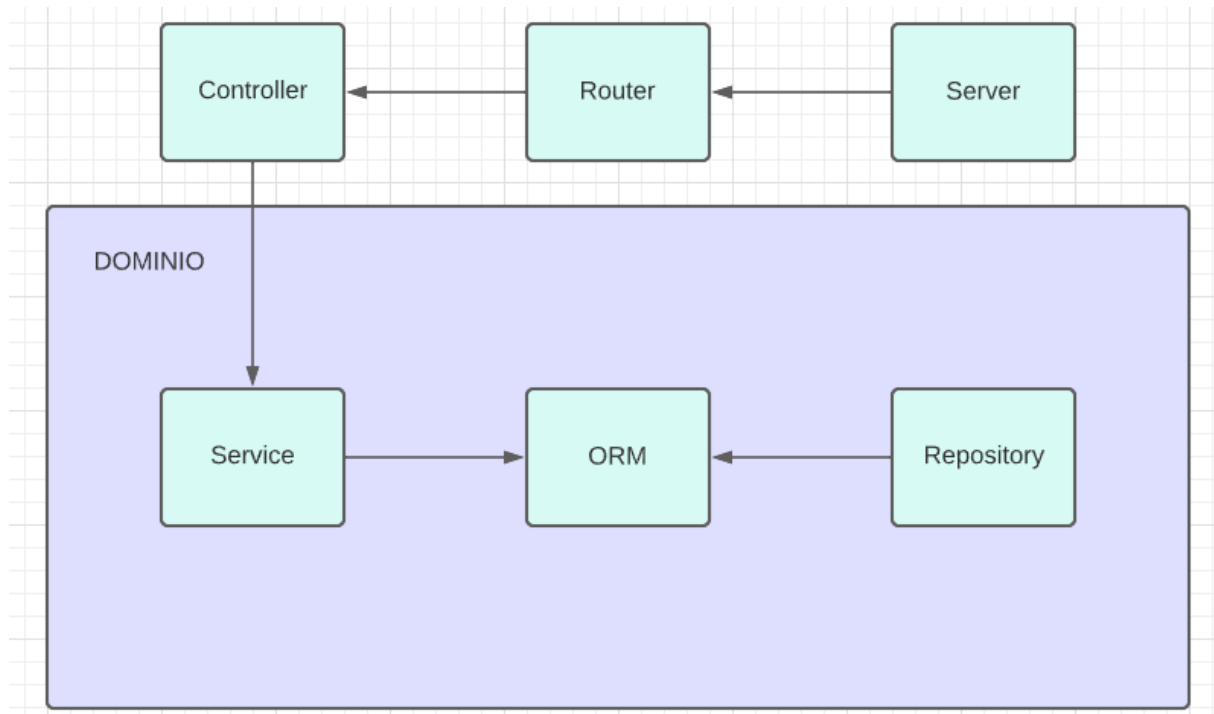


Ilustración 32. Flujo de comunicación del backend – Elaboración propia

Como se puede observar en la ilustración anterior el dominio del backend se compone de servicios, orm's y el repositorio. El flujo funciona de la siguiente manera, se tiene el servidor el cual expone ciertas rutas las cuales funcionan como endpoints para el consumo del backend, estas rutas usan su respectivo controlador el cual contiene los métodos por cuales se pueden consumir las rutas, para esto el controlador importa el servicio el cual contiene los métodos usados en el controlador, dichos métodos hacen uso del orm respectivo a la función que tienen asignada y finalmente los orm se conectan con el repositorio para poder enviar y recibir información a la base de datos.

5.1.2. Desarrollo del backend

Para el desarrollo del backend de la aplicación web de escritorio se hizo uso de Nodejs y Expressjs, además se hizo uso del editor de código Visual Studio. A continuación se mostrará la estructura que tiene el proyecto en el backend así como una explicación de cómo funciona el mismo.

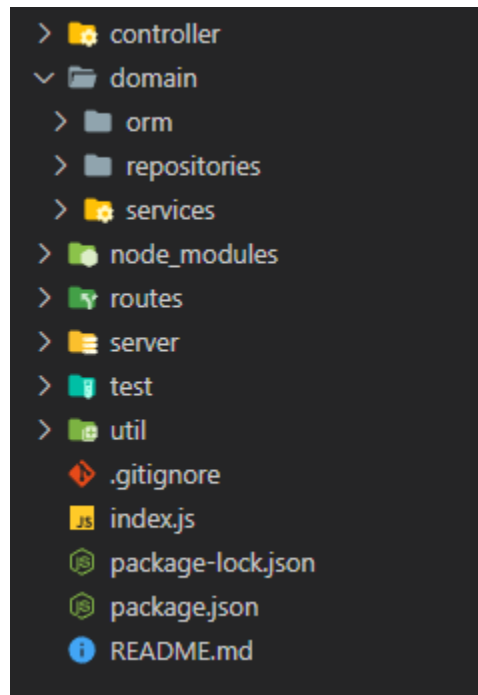


Ilustración 33. Estructura del backend - Elaboración propia

Como se puede observar la estructura del backend cumple con lo definido anteriormente en el diseño orientado al dominio. El archivo principal de un proyecto de Nodejs es el “package.json” en el cual se encuentran definidos los meta datos del proyecto, tales como la descripción, licencia, dependencias y scripts. Luego se tienen las demás carpetas que conforman el diseño de la arquitectura hexagonal. A continuación se realizará la explicación de cómo funciona la comunicación para el consumo de un servicio, para esto vamos a iniciar directamente desde el controlador. En este caso se describirá el funcionamiento del servicio de usuarios.

```
const express = require('express'),
      router = express.Router(),
      magic = require('../util/magic'),
      users = require('../domain/services/service-user');
```

Ilustración 34. Importaciones del controlador - Elaboración propia

Primero tenemos el controlador de usuario, en el cual al inicio se realizan las importaciones que se usaran en dicho archivo, entre las cuales podemos ver la importación del servicio de usuarios.

```
router.get('/users/', users.GetAll);
router.get('/users/:id', users.GetById);
router.post('/users/', users.Store);
router.post('/users/signin', users.Signin);
router.post('/users/verify', users.verifyToken);
router.delete('/users/:id', users.DeleteById);
router.put('/users/', users.UpdateById);
```

Ilustración 35. Endpoints del controlador - Elaboración propia

Aquí se puede observar cómo se declaran los endpoints que se expondrán en la ruta del usuario y en cada endpoint se hace uso de los métodos creados en el servicio. Ahora se mostrará como esta creado el servicio.

```
const magic = require('../../util/magic');
const enum_ = require('../../util/enum');
const ormUser = require('../orm/orm-user');
```

Ilustración 36. Importaciones del servicio usuario - Elaboración propia

Como se explicó anteriormente el servicio importa el orm para poder enviar la información que recibe de la petición al método del orm correspondiente si es necesario, y el orm responderá información obtenida de la base de datos de vuelta al servicio.

```

exports.GetAll = async () =>{
  try{
    const response = await pool.query(`SELECT * FROM usuario WHERE tipo_usu = 'Operador'`);
    return response.rows;
  }catch(err){
    console.log(" err orm-user.GetAll = ", err);
    return await {err:{code: 123, message: err}}
  }
}

```

Ilustración 37. Método para obtener los usuarios de tipo operador - Elaboración propia

Este método podemos ver que no se recibe información del servicio pero retorna la información obtenida de la consulta SQL realizada a la base de datos. Por último, el orm responde con la información correspondiente, esta es recibida por el servicio y posteriormente es enviada como respuesta a la petición realizada al backend.

5.1.3. Desarrollo del frontend

Para el desarrollo del frontend de la aplicación web de escritorio se hizo uso de Angular y Electronjs, además se hizo uso del editor de código Visual Studio Code. A continuación se mostrará como se realizó creación del proyecto, la configuración con electronjs y como se consumieron los servicios del backend.

5.1.3.1. Creación del proyecto

Para crear un proyecto con Angular se debe ejecutar el siguiente comando en una terminal de comandos.

```

$ ng new app-marlon-stefan

```

Ilustración 38. Comando para crear un proyecto de Angular - Elaboración propia

Este comando nos crea un proyecto de angular con su respectiva estructura, una vez creado el proyecto de angular se instaló la dependencia de electronjs con el siguiente comando.

```
$ npm install --save-dev electron@latest
```

Ilustración 39. Comando para instalar la dependencia de electronjs - Elaboración propia

5.1.3.2. Creación y configuración de electronjs

Una vez instalado electronjs se creó un archivo llamado main.js el cual servirá para iniciar el proyecto de Angular como una aplicación de escritorio, dicho archivo contiene el proceso principal que se ejecuta donde se controla el ciclo de vida de la aplicación, así como la interfaz gráfica del usuario y sus elementos.

```
const { app, BrowserWindow } = require("electron");
let appWin;
createWindow = () => {
  appWin = new BrowserWindow({
    title: "Marlon Stefan",
    width: 1280,
    height: 720,
    resizable: false,
    webPreferences: {
      contextIsolation: false,
      nodeIntegration: true
    }
  });
  appWin.loadURL(`file://${__dirname}/dist/index.html`);
  appWin.setMenu(null);
  appWin.on("closed", () => {
    appWin = null;
  });
}
app.on("ready", createWindow);
app.on("window-all-closed", () => {
  if (process.platform !== "darwin") {
    app.quit();
  }
});
```

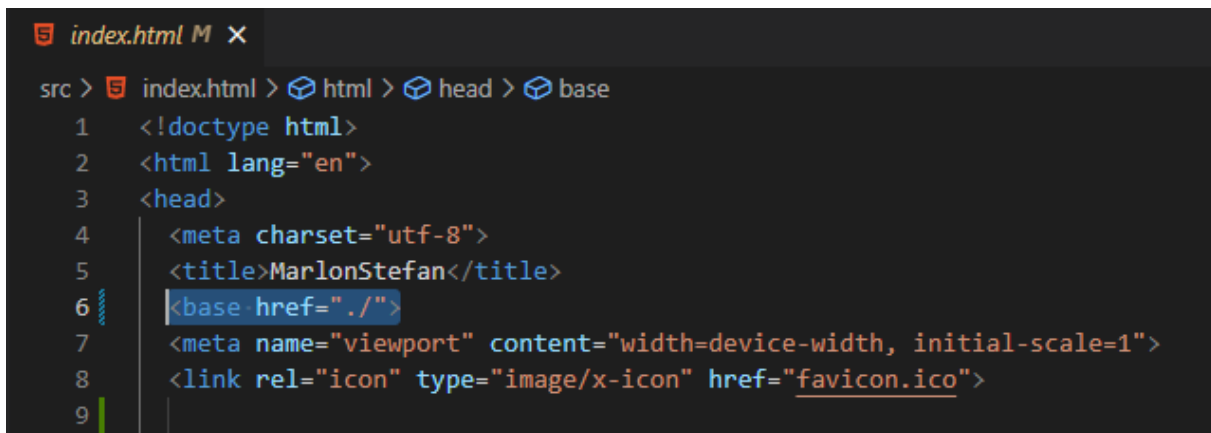
Ilustración 40. Main.js que ejecuta la aplicación de escritorio - Elaboración propia

Una vez creado este archivo, se creó el script correspondiente para realizar la ejecución de electronjs, en el cual primero se realiza un build al proyecto y luego se ejecuta el archivo main.js como se podrá ver a continuación.

```
"build-electron": "ng build && electron .",
```

Ilustración 41. Script para la ejecución de electrón - Elaboración propia

Posteriormente, se configuro el archivo index.html generado por Angular en el cual se cambió el “base href” de “/” a “./” debido a que electronjs funciona con un sistema de archivos y no con rutas de dominio.



```
index.html M X
src > index.html > html > head > base
1 <!doctype html>
2 <html lang="en">
3 <head>
4   <meta charset="utf-8">
5   <title>MarlonStefan</title>
6   <base href="./">
7   <meta name="viewport" content="width=device-width, initial-scale=1">
8   <link rel="icon" type="image/x-icon" href="favicon.ico">
9
```

Ilustración 42. Index.html del proyecto en Angular - Elaboración propia

Una vez realizado lo anterior ya está configurado electronjs con Angular por lo cual al ejecutar el script creado anteriormente, se iniciará una aplicación de escritorio la cual contendrá las interfaces graficas creadas en Angular.

5.1.4. Interfaces de usuario

A continuación se mostrarán las interfaces de usuario creadas en Angular junto a una descripción de su funcionamiento.

5.1.4.1. Inicio de sesión

Al iniciar la aplicación se le mostrara al usuario la siguiente pantalla, la cual tiene como función el poder iniciar sesión en el sistema, para esto debe ingresar su usuario y contraseña en los campos correspondientes.

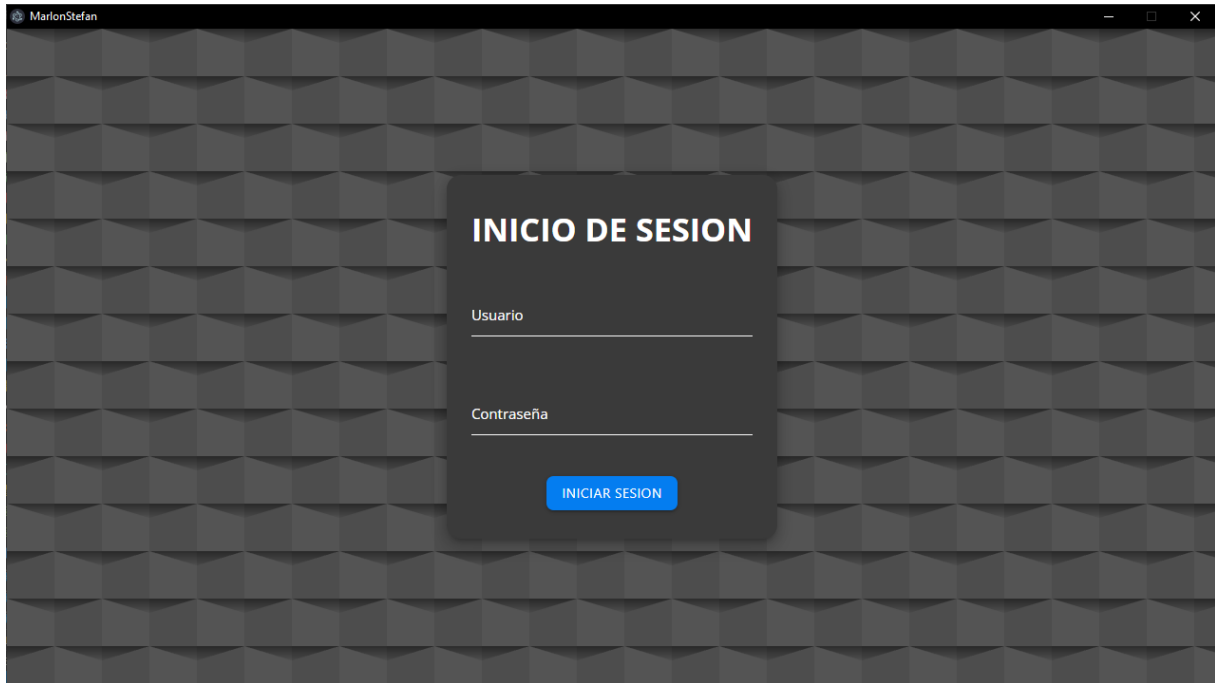


Ilustración 43. Inicio de sesión - Elaboración propia

5.1.4.2. Menú principal (Administrador)

Esta interfaz contiene el menú con las opciones disponibles para un usuario de tipo Administrador, las cuales son visualizar el inventario, los operadores, los clientes, y las operaciones de ingreso y egreso.

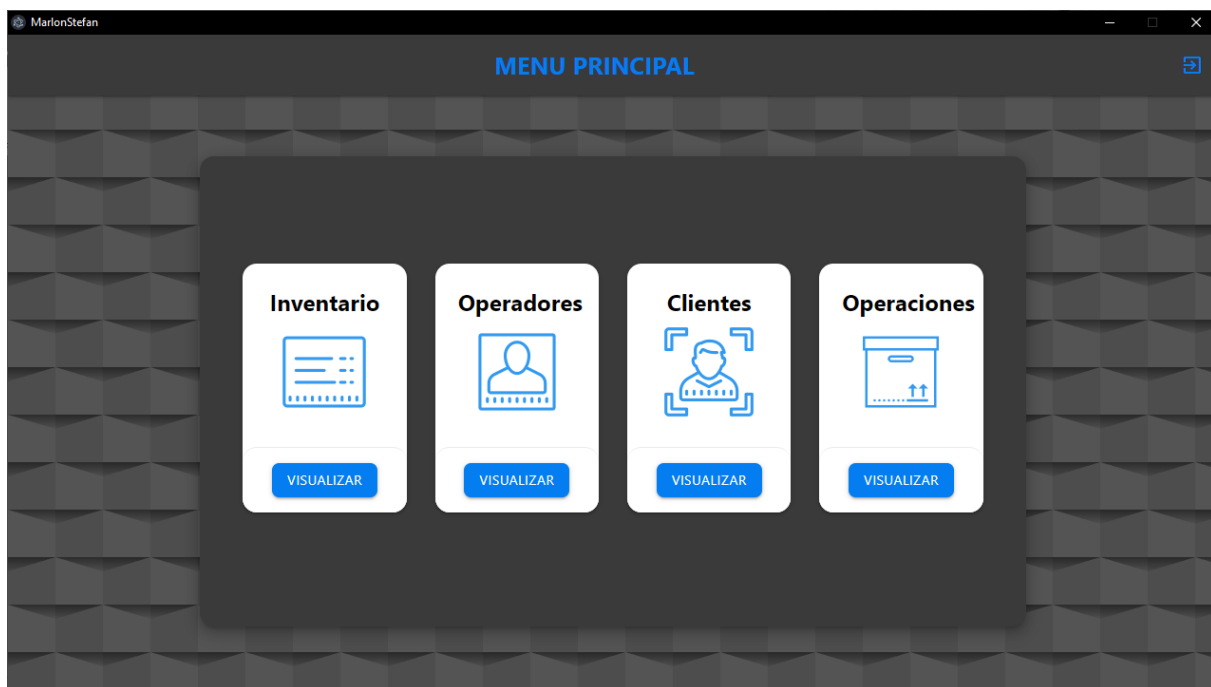


Ilustración 44. Menú principal (Administrador)

5.1.4.3. Menú principal (Operador)

Esta interfaz contiene el menú con las opciones disponibles para un usuario de tipo Operador, las cuales son las mismas del Administrador exceptuando la opción de visualizar los operadores.

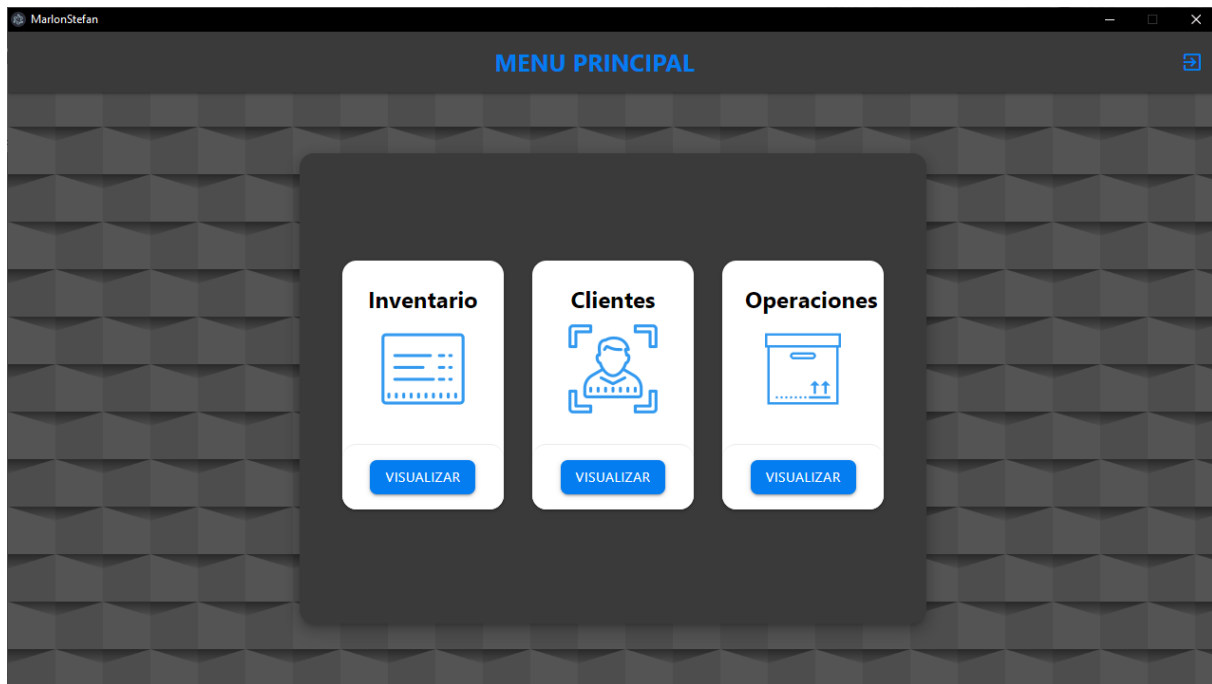


Ilustración 45. Menú principal (Operador)

5.1.4.4. Inventario

Una vez se ingrese en visualizar el inventario se podrán ver primero los productos del que han sido registrados en el inventario, así como también un filtro para poder obtener algún producto en concreto con mayor exactitud. Además, si se da clic al botón “nuevo” se generará un modal el cual permitirá el registro de un nuevo producto como se puede ver a continuación.

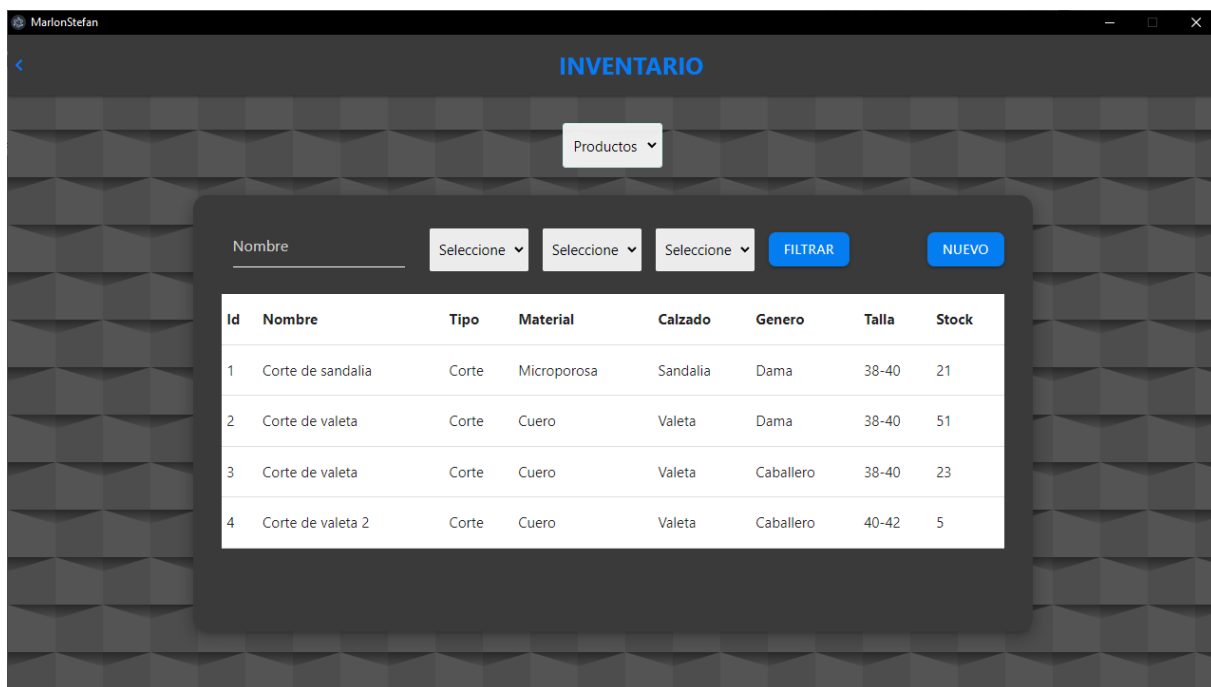


Ilustración 46. Inventario (Productos) - Elaboración propia

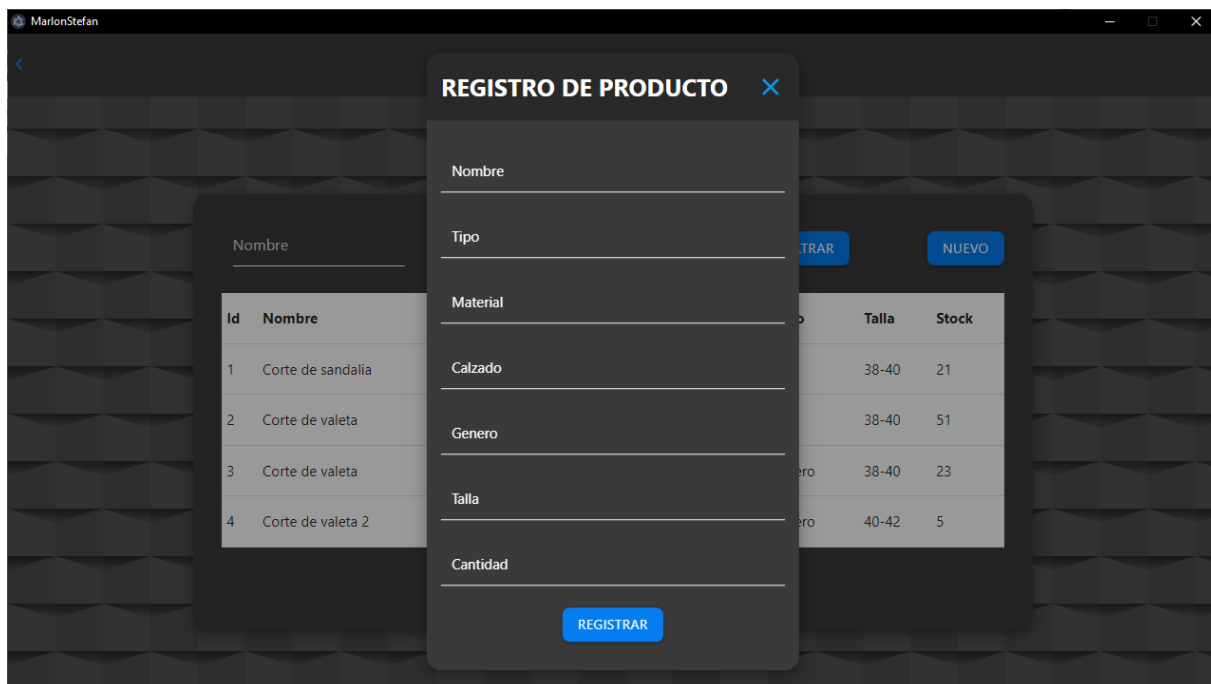


Ilustración 47. Modal registro de producto - Elaboración propia

Para poder visualizar los materiales del inventario se debe abrir la lista desplegable disponible en la parte superior de la interfaz y seleccionar “materiales”.



Ilustración 48. Lista para ver productos o materiales - Elaboración propia

Una vez hecho esto se podrán visualizar los materiales que hayan sido registrados en el inventario del sistema. Al igual que en los productos se cuenta un filtro especial para los materiales así como, un botón que permite el registro de un nuevo material mediante su respectivo moda.

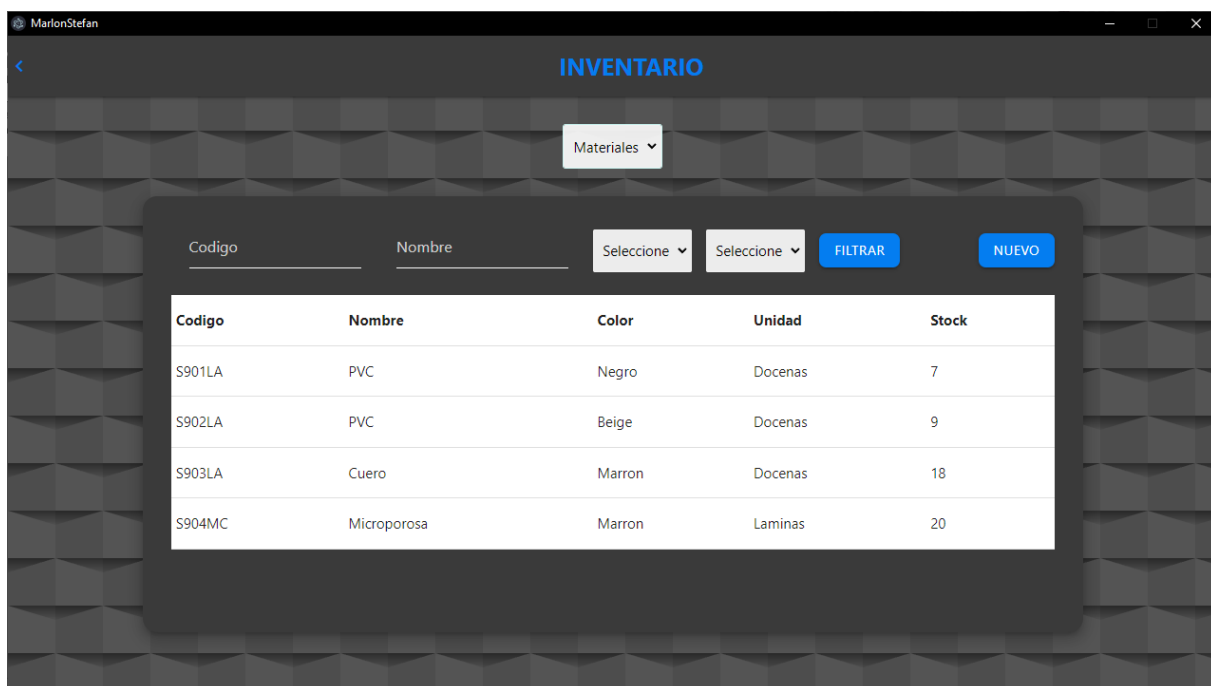


Ilustración 49. Inventario (materiales) – Elaboración propia

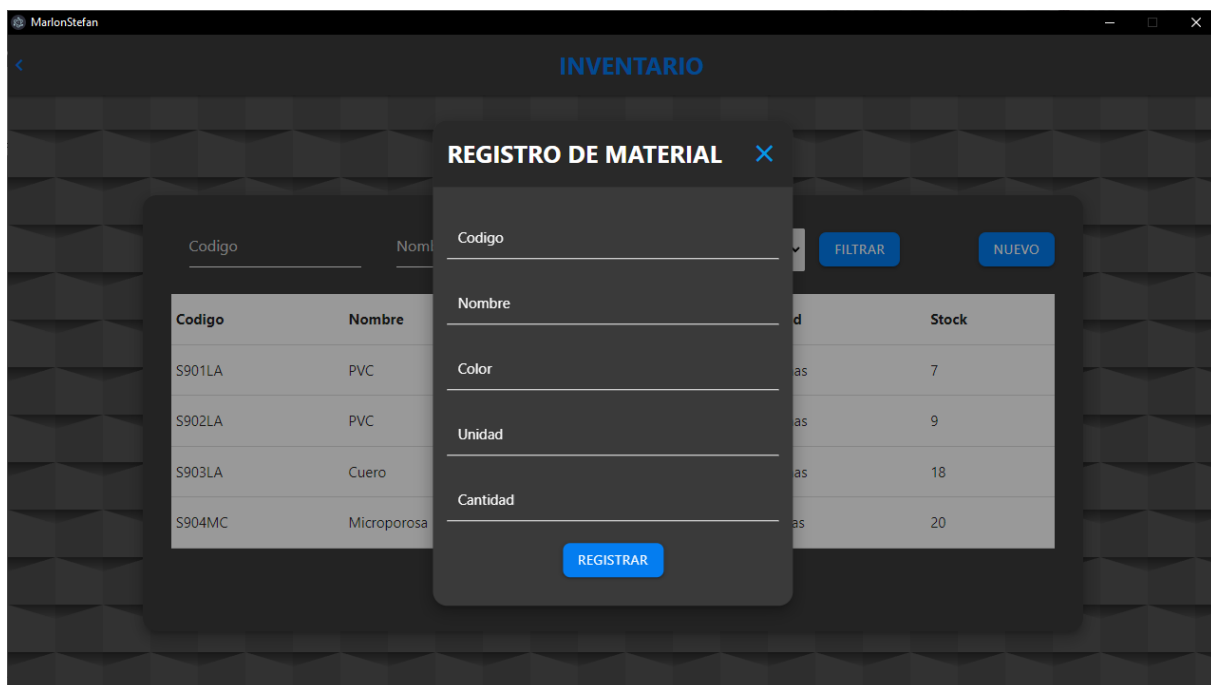


Ilustración 50. Modal registro de material - Elaboración propia

5.1.4.5. Operadores

En esta interfaz se pueden visualizar los operadores, los cuales podrán tener acceso al sistema con el usuario y contraseña que se registren mediante el modal disponible al dar clic en el botón nuevo, disponible en la misma interfaz donde se visualizan los operadores.



Ilustración 51. Visualizar operadores - Elaboración propia

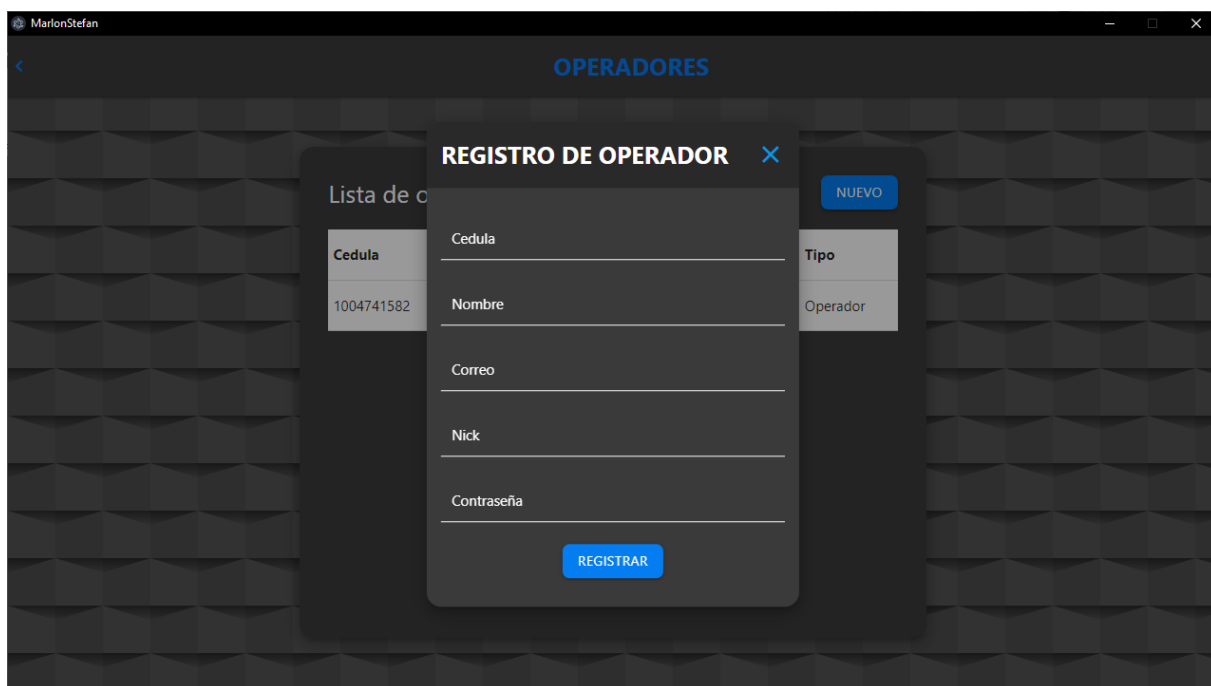


Ilustración 52. Modal registro de operador - Elaboración propia

5.1.4.6. Clientes

En esta interfaz al igual que en la de inventario cuenta con una lista desplegable en la parte superior de la vista, como se puede ver a continuación.

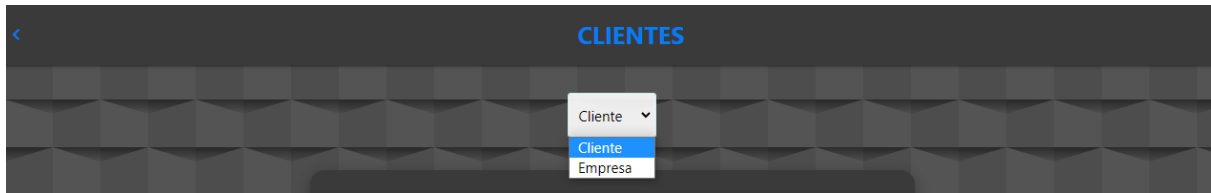


Ilustración 53. Lista para ver clientes o empresas - Elaboración propia

Cuando se seleccione cliente, se podrán visualizar los clientes que estén registrados en el sistema, para poder registrar un nuevo cliente al igual que en las otras interfaces existe un botón “nuevo” el cual abre un modal para realizar dicha funcionalidad.

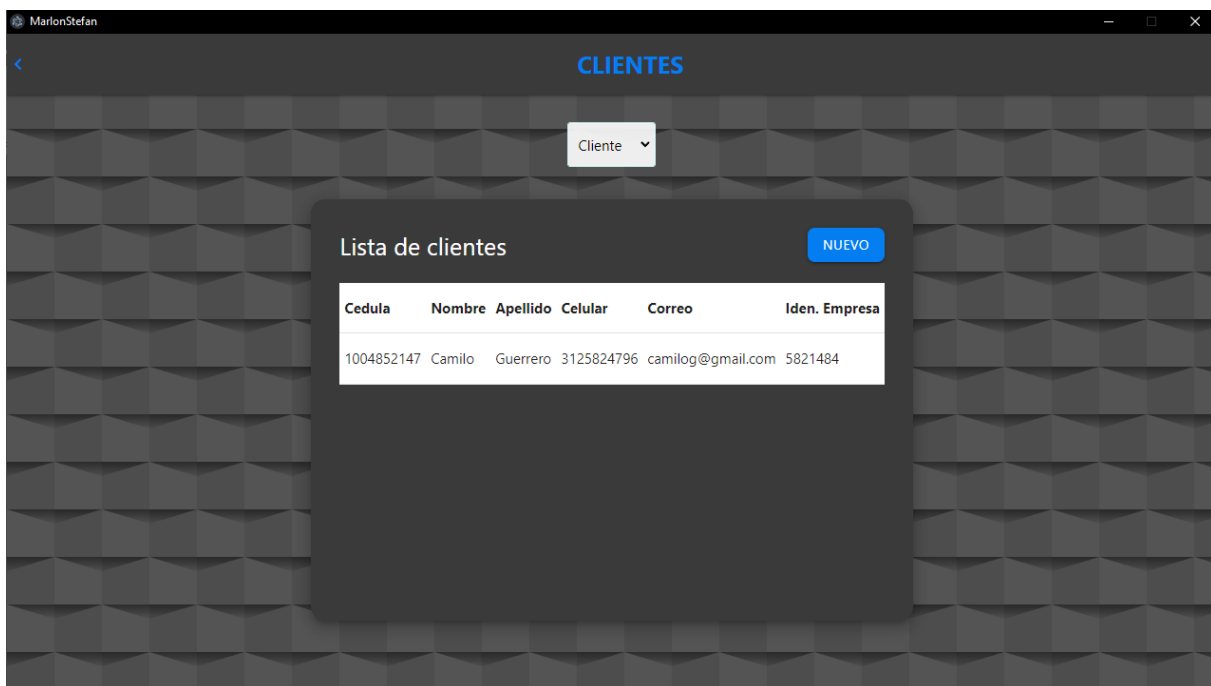


Ilustración 54. Visualizar clientes - Elaboración propia

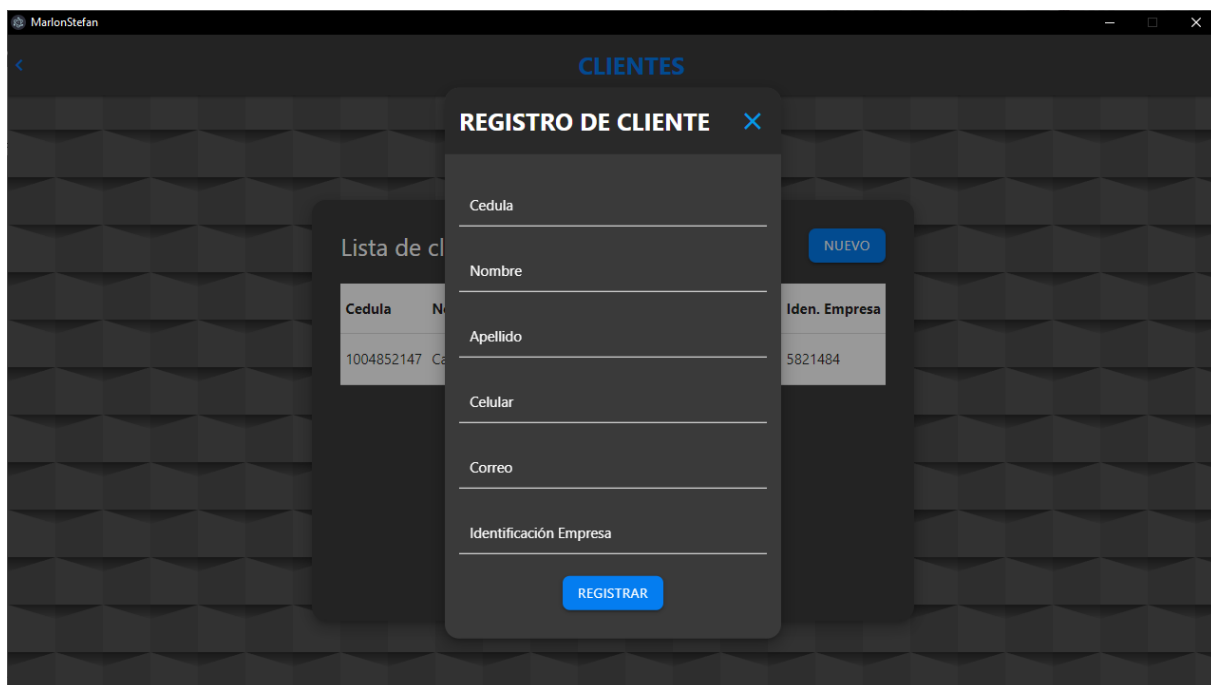


Ilustración 55. Modal registro cliente - Elaboración propia

Si se selecciona empresa en la lista desplegable, se podrán visualizar las empresas que estén registradas en el sistema, para poder registrar una nueva empresa al igual que en interfaz anterior se cuenta con un botón “nuevo” el cual abre un modal para realizar dicha funcionalidad.

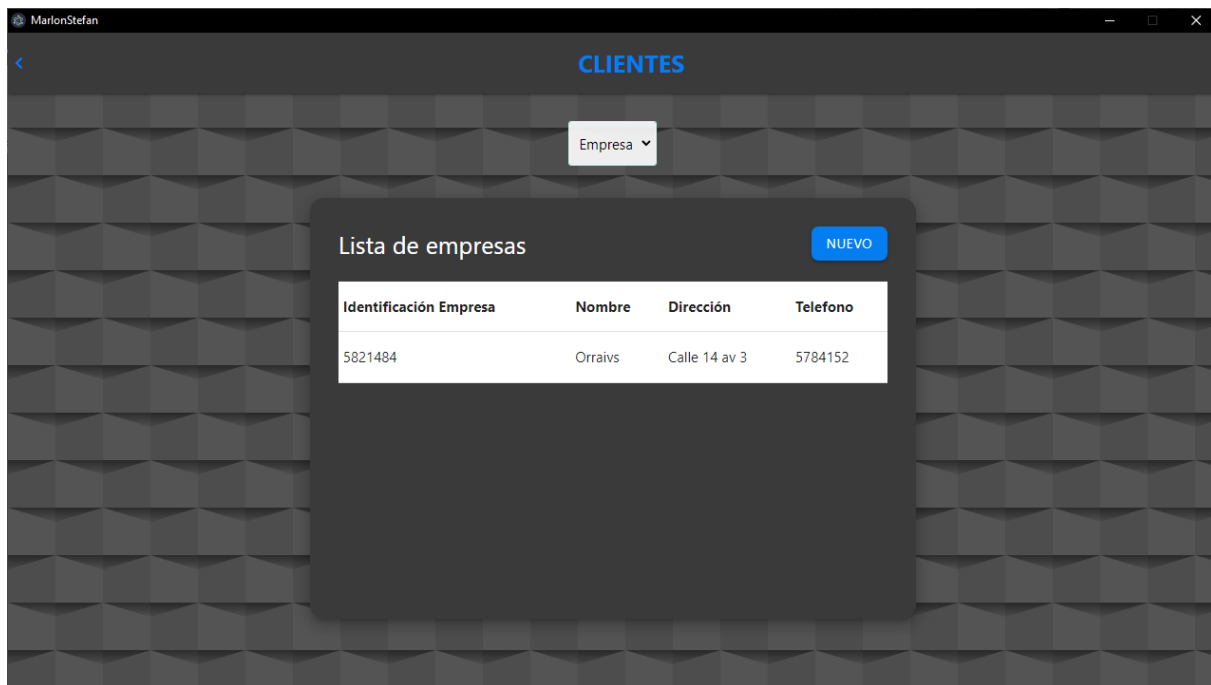


Ilustración 56. Visualizar empresas - Elaboración propia

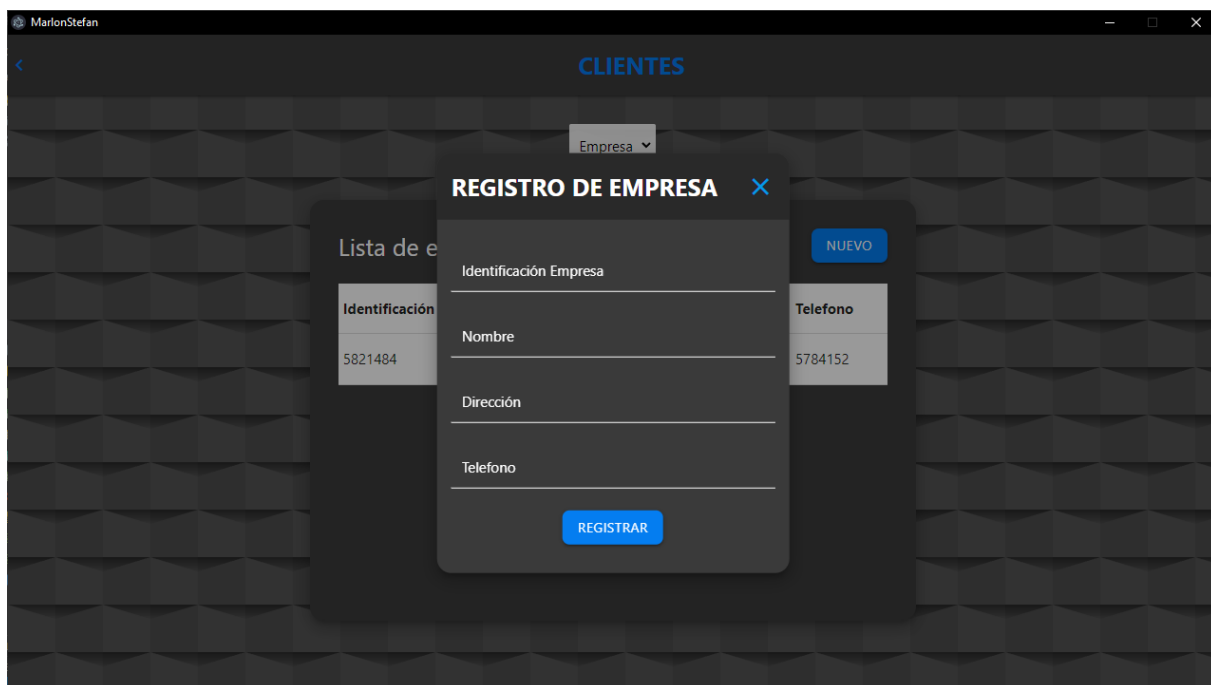


Ilustración 57. Modal registro empresa - Elaboración propia

5.1.4.7. Operaciones

Para realizar las operaciones de ingresos y egresos primero se debe seleccionar visualizar operaciones desde el menú principal, lo cual abrirá la siguiente interfaz con las dos operaciones mencionadas.

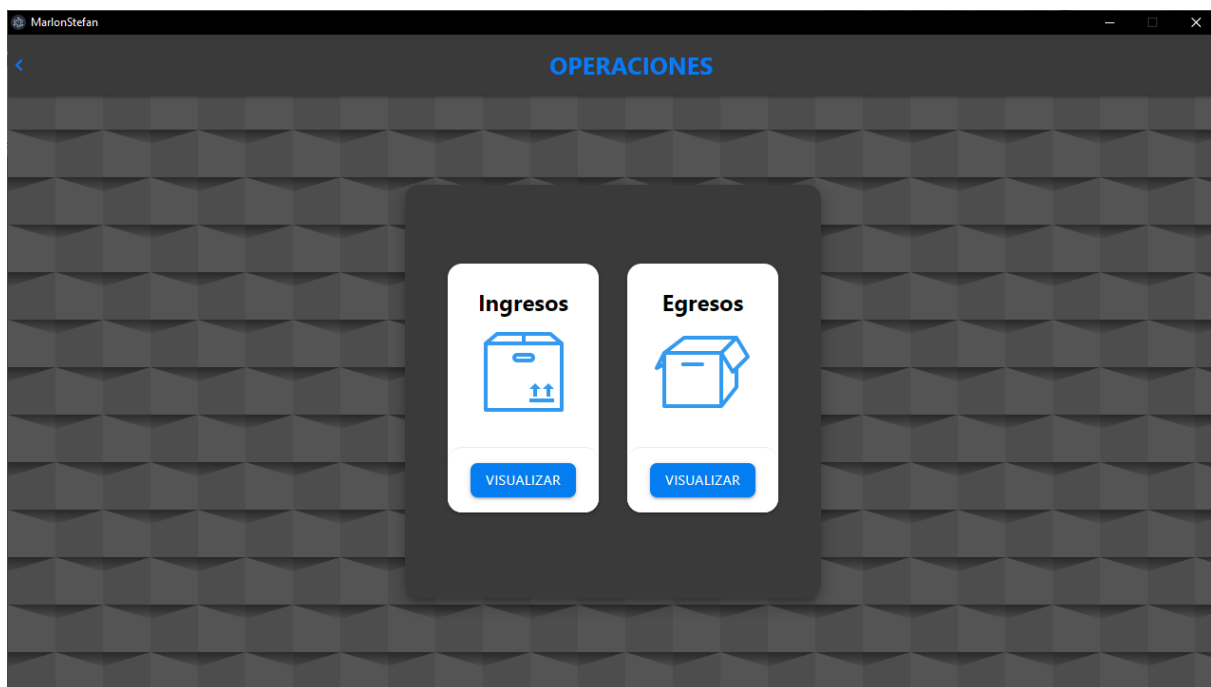


Ilustración 58. Interfaz operaciones - Elaboración propia

5.1.4.7.1. Ingresos

Esta interfaz cuenta con la misma estructura de las interfaces de inventario y clientes, logrando así que el usuario tenga un fácil manejo de las interfaces. Primero se mostrarán los ingresos de los productos, contando de igual manera con un filtro por fechas para poder visualizar de manera rápida y efectiva los ingresos. También, cuenta con un botón “nuevo” para realizar el registro de un nuevo ingreso.

Id	Fecha de ingreso	Id producto	Cantidad ingresada	Operador
1	2022-NOV-17	1	5	1004845348
2	2022-NOV-17	2	5	1004845348
3	2022-NOV-17	3	5	1004845348
4	2022-NOV-17	1	5	1004845348
5	2022-NOV-17	2	5	1004845348

Ilustración 59. Visualización ingresos productos - Elaboración propia

Cuando se seleccione registrar un nuevo ingreso, se mostrará la siguiente interfaz en la cual se podrán seleccionar los productos que se ingresaran al inventario pudiendo especificar la cantidad de nuevos productos que ingresaran, igual que en la visualización del inventario se cuenta con un filtro para los productos.

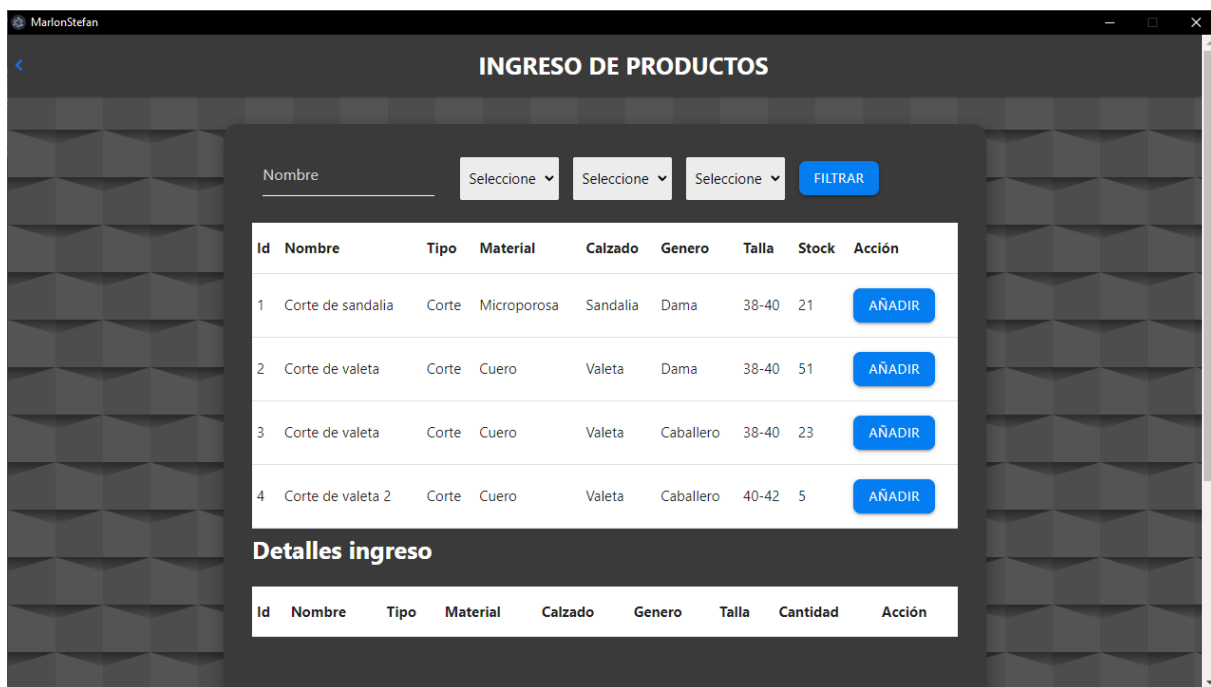


Ilustración 60. Interfaz registro ingreso de productos - Elaboración propia

Como se mencionó anteriormente también se pueden ver los ingresos de los materiales y registrar un nuevo ingreso solo escogiendo materiales en la lista desplegable disponible en la parte superior de la interfaz de ingresos.



Ilustración 61. Lista desplegable ingresos de productos y materiales - Elaboración propia

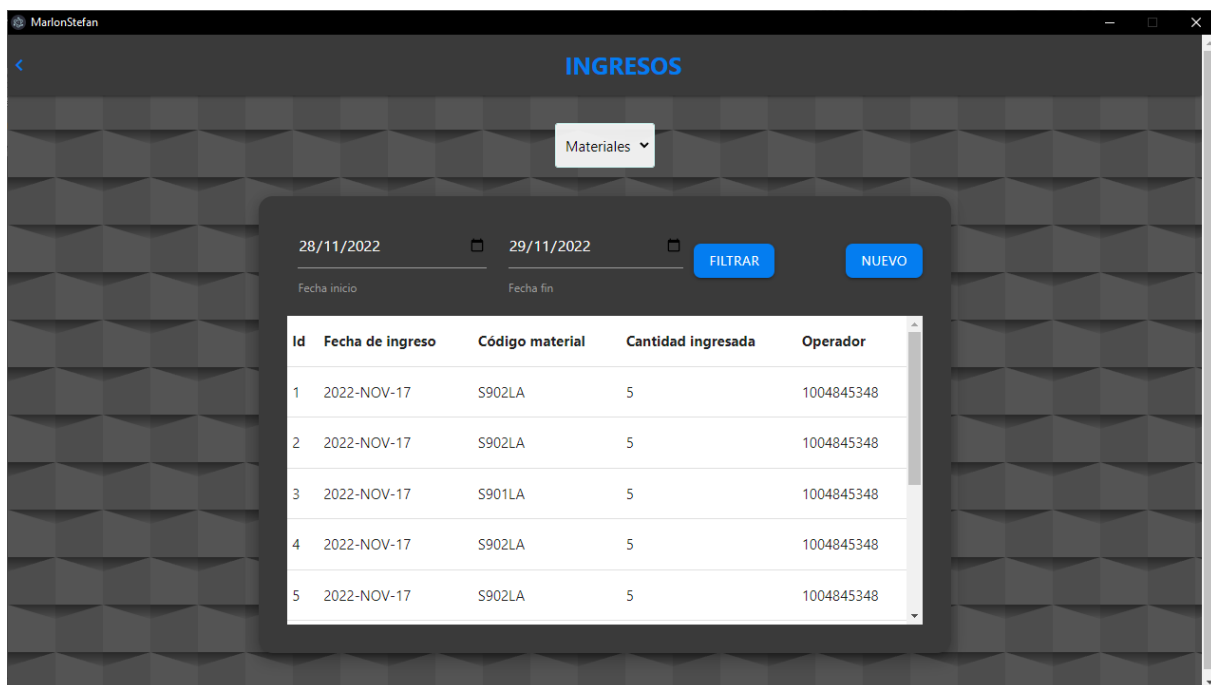


Ilustración 62. Visualización ingresos materiales - Elaboración propia

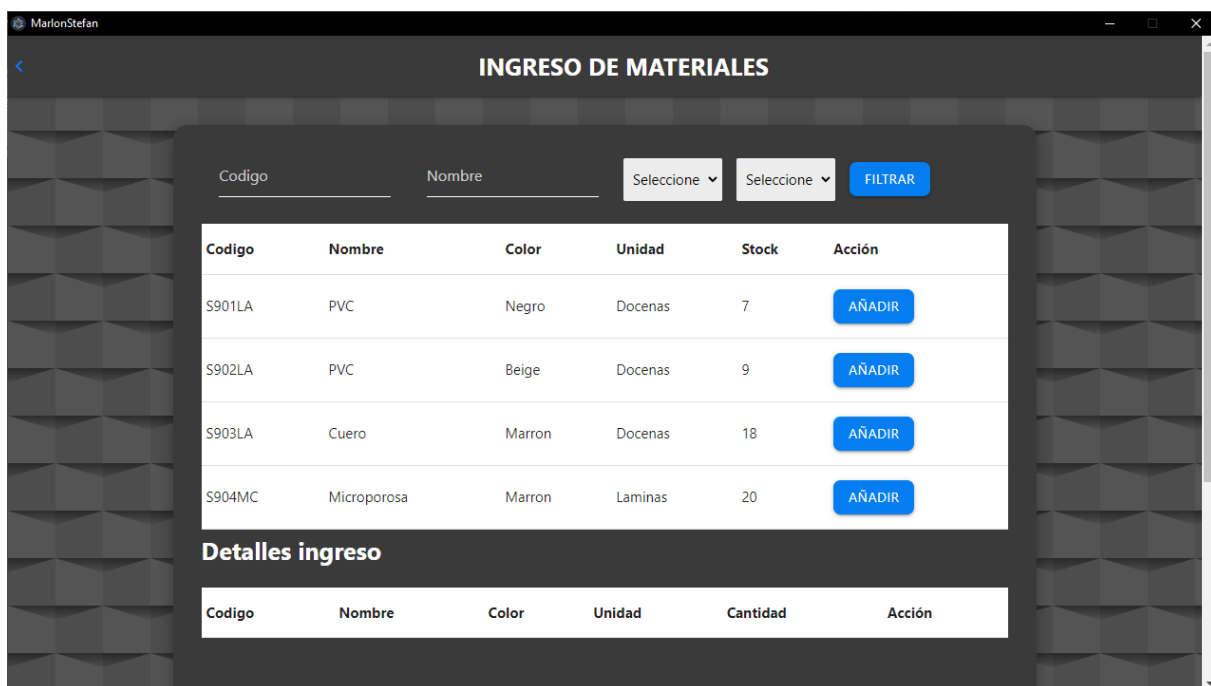
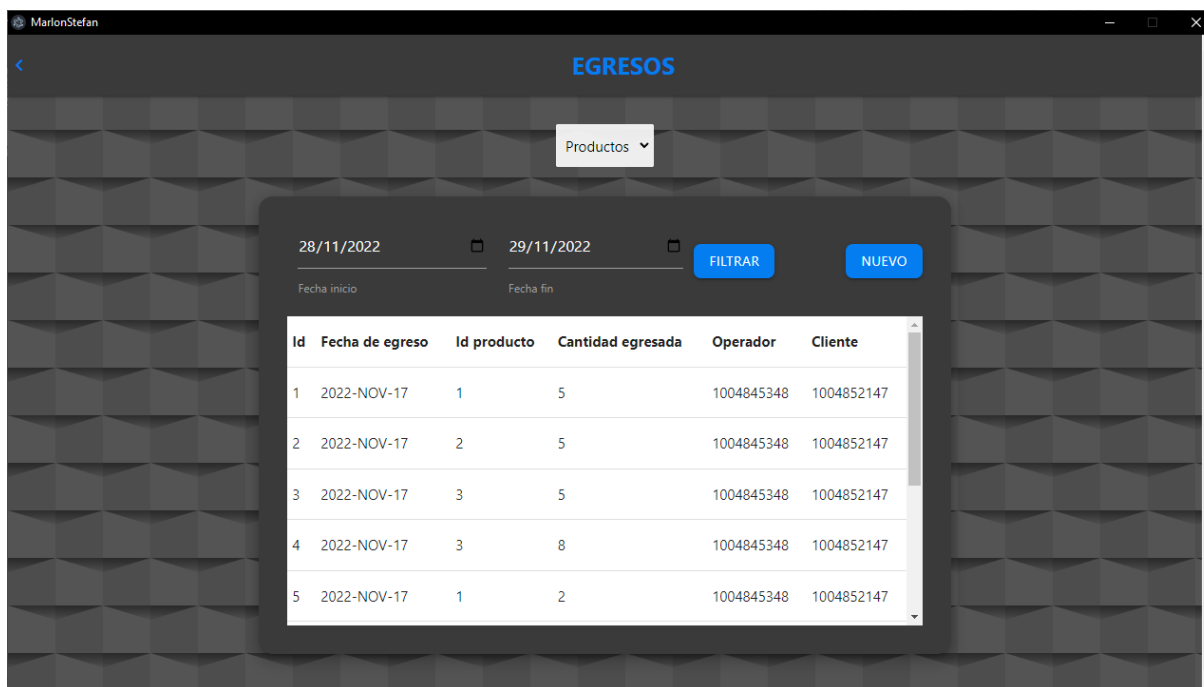


Ilustración 63. Interfaz registro ingreso de materiales - Elaboración propia

5.1.4.7.2. Egresos

Al igual que en la interfaz de ingresos esta interfaz cuenta con la misma estructura de las interfaces de inventario y clientes. Al dirigirse a egresos desde el menú de operaciones

primero se mostrarán los egresos de los productos, contando de igual manera con un filtro por fechas para poder visualizar de manera rápida y efectiva los egresos. También, cuenta con un botón “nuevo” para realizar el registro de un nuevo egreso en el sistema.



The screenshot shows a web application window titled 'MarlonStefan' with a dark theme. The main heading is 'EGRESOS' in blue. Below it is a 'Productos' dropdown menu. A date range filter is set from '28/11/2022' to '29/11/2022', with 'Filtrar' and 'Nuevo' buttons. A table displays the following data:

Id	Fecha de egreso	Id producto	Cantidad egresada	Operador	Cliente
1	2022-NOV-17	1	5	1004845348	1004852147
2	2022-NOV-17	2	5	1004845348	1004852147
3	2022-NOV-17	3	5	1004845348	1004852147
4	2022-NOV-17	3	8	1004845348	1004852147
5	2022-NOV-17	1	2	1004845348	1004852147

Ilustración 64. Visualización egresos productos - Elaboración propia

Cuando se seleccione registrar un nuevo egreso, se mostrará la siguiente interfaz en la cual se podrán seleccionar los productos que se egresaran del inventario pudiendo especificar la cantidad de productos que saldrán del inventario, igual que en la interfaz de ingresos se cuenta con un filtro para los productos. Además, se cuenta con un campo donde se debe digitar la cedula del cliente al cual se le está realizando el egreso.

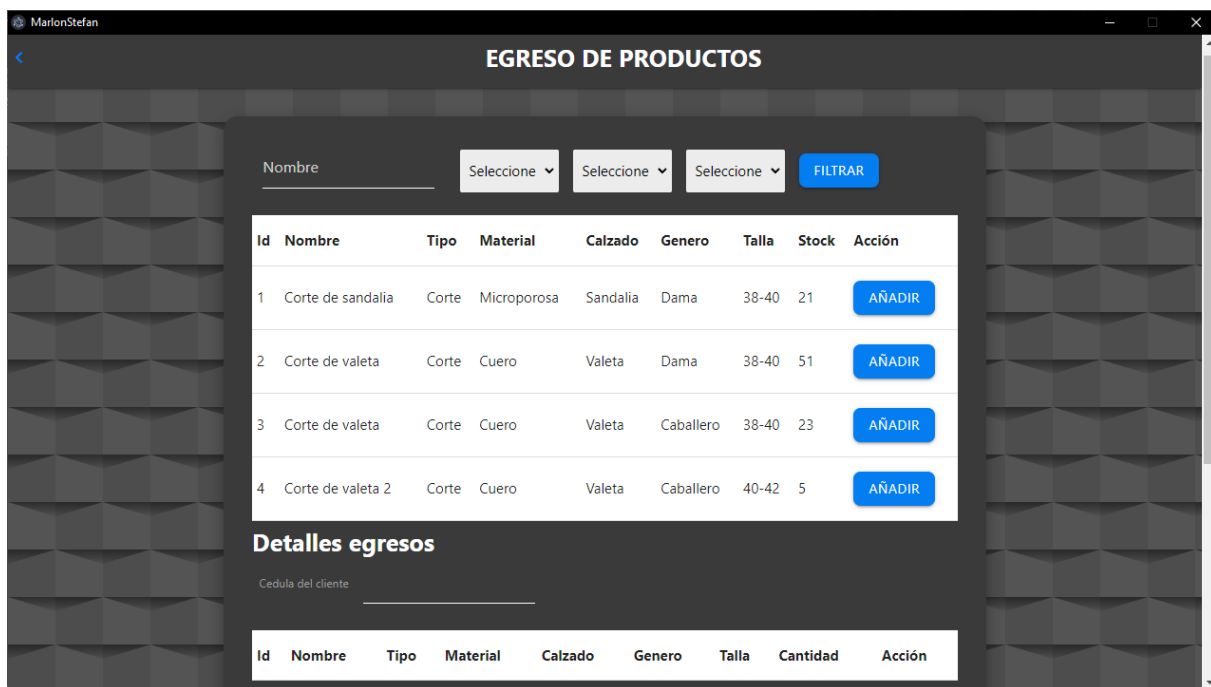


Ilustración 65. Interfaz registro egreso de productos - Elaboración propia

Como se mencionó anteriormente también se pueden ver los egresos de los materiales y registrar un nuevo egreso solo escogiendo materiales en la lista desplegable disponible en la parte superior de la interfaz de egresos.



Ilustración 66. . Lista desplegable egresos de productos y materiales - Elaboración propia

The screenshot shows a web application window titled 'EGRESOS'. At the top, there is a dropdown menu labeled 'Materiales'. Below it, a date range is set from '28/11/2022' to '29/11/2022', with 'Fecha inicio' and 'Fecha fin' labels. There are 'FILTRAR' and 'NUEVO' buttons. The main content is a table with the following data:

Id	Fecha de egreso	Código material	Cantidad egresada	Operador
1	2022-NOV-17	S902LA	5	1004845348
2	2022-NOV-17	S902LA	5	1004845348
3	2022-NOV-17	S901LA	5	1004845348
4	2022-NOV-17	S902LA	1	1004845348
5	2022-NOV-17	S902LA	1	1004845348

Ilustración 67. Visualización egresos materiales - Elaboración propia

The screenshot shows a web application window titled 'EGRESO DE MATERIALES'. At the top, there are search filters for 'Codigo' and 'Nombre', and two 'Seleccione' dropdown menus. There is a 'FILTRAR' button. The main content is a table with the following data:

Codigo	Nombre	Color	Unidad	Stock	Acción
S901LA	PVC	Negro	Docenas	7	<button>AÑADIR</button>
S902LA	PVC	Beige	Docenas	9	<button>AÑADIR</button>
S903LA	Cuero	Marron	Docenas	18	<button>AÑADIR</button>
S904MC	Microporosa	Marron	Laminas	20	<button>AÑADIR</button>

Below the table, there is a section titled 'Detalles egresos' with a table structure:

Codigo	Nombre	Color	Unidad	Cantidad	Acción
--------	--------	-------	--------	----------	--------

Ilustración 68. Interfaz registro egreso de materiales - Elaboración propia

5.1.5. Despliegue de la base datos

La base de datos fue desplegada en ElephantSQL un servicio de hosting para bases de datos de PostgreSQL. Para esto se creó una instancia para almacenar la BD en dicha página.

Name	Host	Plan	Datacenter	Actions
app-marlon	peanut	Tiny Turtle	Amazon Web Services US-East-1 (Northern Virginia)	Edit

Ilustración 69. Instancia de base de datos en ElephantSQL

Luego de haber creado la instancia se procedió a realizar la creación de las tablas de la base de datos por medio de lenguaje SQL, para esto ElephantSQL provee un apartado en el cual se puede insertar directamente el código SQL ya sea para la realización de consultas o en este caso para la creación de las entidades de la BD.

Ilustración 70. Interfaz para insertar código SQL en el hosting

Por último, se puede observar que al insertar y ejecutar el código SQL de la base de datos se pueden observar las tablas que se crearon en la BD.

cliente (public) (0 KiB, ~0 rows)
egreso_material (public) (0 KiB, ~0 rows)
egreso_producto (public) (0 KiB, ~0 rows)
empresa (public) (0 KiB, ~0 rows)
ingreso_material (public) (0 KiB, ~0 rows)
ingreso_producto (public) (0 KiB, ~0 rows)
material (public) (0 KiB, ~0 rows)
producto (public) (0 KiB, ~0 rows)
usuario (public) (0 KiB, ~0 rows)

Ilustración 71. Tablas de la base de datos en ElephantSQL

5.1.6. Creación de la aplicación de escritorio ejecutable con electron

Una vez la aplicación en Angular ha sido desarrollada por completo y configurada con electron, se procedió a ejecutar el comando con el cual se creará una carpeta la cual contendrá los archivos necesarios para la ejecución de la aplicación junto un archivo ejecutable para iniciar la misma.

Para esto se realizó la instalación de la dependencia electron-packager la cual permitirá la creación del proyecto ejecutable, para instalar dicha dependencia se hizo uso del siguiente comando.

```
$ npm install electron-packager --save-dev
```

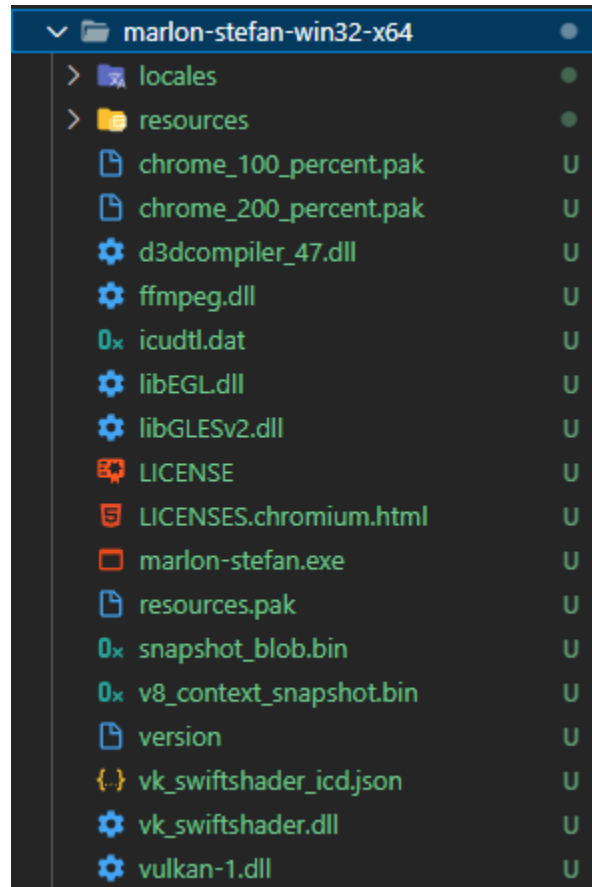
Luego de instalar la dependencia, se podrá ejecutar el comando “electron-packager” el cual recibirá 3 parámetros los cuales son el directorio base de la aplicación, la plataforma del ejecutable y la arquitectura del mismo.

```
$ electron-packager . --platform=win32 --arch=x64
```

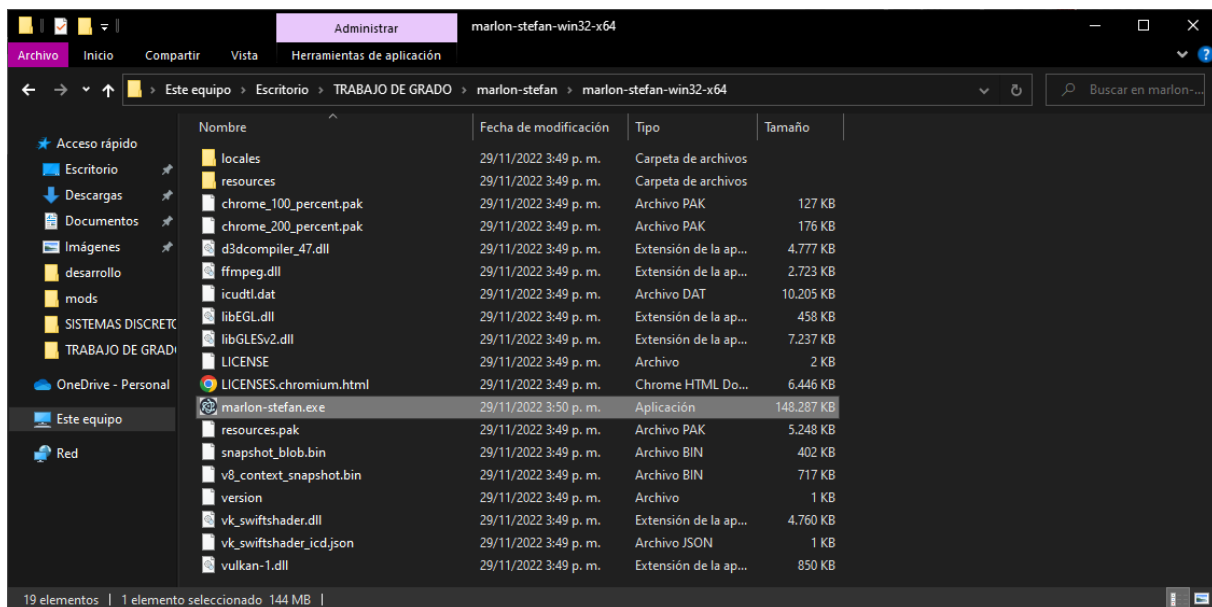
En el primer parámetro se coloca un punto (.) para referenciar la carpeta raíz del proyecto, en el segundo se especifica que la plataforma será Windows y en la arquitectura se especifica que será 64 bits.

```
$ electron-packager . --platform=win32 --arch=x64  
Packaging app for platform win32 x64 using electron v20.1.4
```

Una vez ejecutado el comando se empezará a empacar la aplicación para la plataforma y arquitectura especificada en el comando anterior. Al finalizar se habrá creado una carpeta en la raíz del proyecto como se ve a continuación.



Si abrimos esta carpeta desde el explorador de archivos de Windows se vera de la siguiente manera.



Como se puede observar en la imagen anterior, ya esta creado el ejecutable del proyecto con el cual se puede crear un acceso directo al escritorio e iniciarlo en cualquier momento.

5.2. Realizar pruebas a la aplicación web de escritorio para la gestión de inventario.

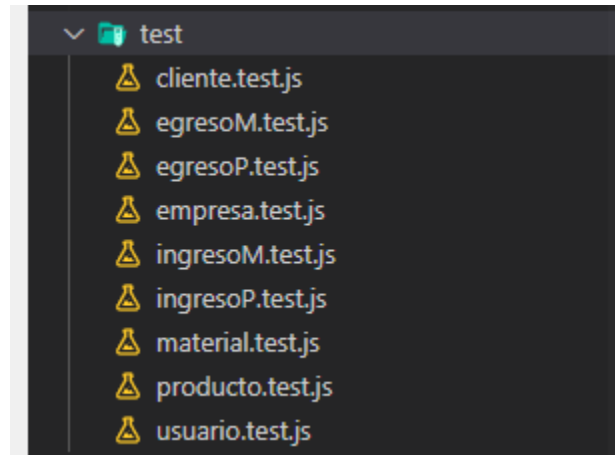
A continuación, se presentará como se realizaron las pruebas unitarias sobre algunos endpoints de los servicios creados en el backend de la aplicación web de escritorio.

5.2.1. Instalación y configuración de las dependencias

Primeramente, se realizó la instalación de las dependencias “jest” el cual es un framework de pruebas que nos permite testear de una manera rápida y fácil nuestro backend, así como también “supertest” el cual nos permite realizar peticiones http a la API desde el mismo backend. Para la instalación se usó el siguiente comando, el cual instaló las dependencias en el apartado de dependencias de desarrollo, esto se hace para que en el momento de desplegar el backend no se instalarán dichas dependencias en el servidor.

```
$ npm install -D jest supertest
```

Una vez instaladas las dependencias se creó una carpeta en el proyecto la cual tendrá el nombre de “test” y dentro de esta se creó un archivo para cada servicio el cual contendrá las pruebas de los endpoints expuestos en cada uno.



Ahora se debe configurar el archivo “package.json” para poder realizar las pruebas creadas en el archivo anteriormente mencionado. Por último, se creó un nuevo script llamado “test” el cual nos permitirá la ejecución de las pruebas mediante la dependencia “jest”.

```
"scripts": {  
  "start": "node .",  
  "start:dev": "nodemon index.js",  
  "test": "jest --forceExit"  
},
```

5.2.2. Como se realizan las pruebas

Para la codificación de las pruebas primero se importó la dependencia “supertest” en cada archivo de pruebas para la realización de las peticiones, así como el archivo “server.js” en el cual se exporta la app del servidor.

```
const request = require('supertest');  
const app = require('../server');
```

Luego, se codificaron las pruebas para cada endpoint de la siguiente manera, primero se hace uso del método describe() el cual recibe dos parámetros el primero es un texto que describe el título de las pruebas, y una función que contiene las pruebas.

```
describe('titulo', () => {  
  //pruebas  
})
```

Una vez definido esto, se realiza la codificación de las pruebas para cada endpoint de la siguiente manera, se usa el método `test()` que al igual que el anterior recibe dos parámetros de los cuales el primero representa un título de la prueba del endpoint y el segundo es una función donde se realiza la prueba, en la cual primero se realiza la petición al endpoint y luego se comprueba que la respuesta recibida sea igual a la que se espera.

```
describe('titulo', () => {  
  //pruebas  
  test('prueba endpoint 1', () => {  
    //peticion  
    //comprobar respuesta  
  })  
  test('prueba endpoint 2', () => {  
    //peticion  
    //comprobar respuesta  
  })  
})
```

5.2.3. Realización de pruebas

Como se mencionó anteriormente se realizaron pruebas para cada servicio expuesto en el backend de la aplicación web de escritorio mediante el uso de las dependencias “jest” y “supertest”, para esto se creó un archivo con la extensión “test.js” que hace referencia a cada prueba de los servicios y dentro de cada prueba de los servicios se testearon los endpoints expuestos para cada ruta del backend. Una vez observado como se realizan dichas pruebas se procederá a mostrar capturas de algunos endpoints de los servicios testeados, y posteriormente los resultados obtenidos en cada prueba (*ver ilustración 72 a la ilustración 107*).

5.2.3.1. Pruebas del servicio usuario

```
const request = require('supertest');
const app = require('../server');

describe('Usuario Endpoints', () => {

  test('deberia obtener todos los usuarios', async () => {
    const res = await request(app)
      .get('/api/users/')
      .send();
    expect(res.statusCode).toEqual(200);
    expect(res.body.status).toEqual(true);
  });
});
```

Ilustración 72. Prueba endpoint obtener usuarios

```
test('deberia obtener usuario por su cedula', async () => {
  const res = await request(app)
    .get('/api/users/'+1004845348)
    .send();
  expect(res.statusCode).toEqual(200);
  expect(res.body.status).toEqual(true);
  expect(res.body.Resp.data[0].nombre_usu).toEqual('David Celis')
});
```

Ilustración 73. Prueba endpoint obtener usuario por cedula

```
test('deberia crear un usuario', async () => {
  const res = await request(app)
    .post('/api/users/')
    .send({
      cedula_usu: 1004845322,
      nombre_usu: "David Celis",
      correo_usu: "david@gmail.com",
      nick_usu: "Schied",
      contra_usu: "12345789A!",
      tipo_usu: "Operador"
    });
  expect(res.statusCode).toEqual(201);
  expect(res.body.status).toEqual(true);
});
```

Ilustración 74. Prueba endpoint crear usuario

```
test('deberia eliminar un usuario', async () => {
  const res = await request(app)
    .delete('/api/users/'+1004845322)
    .send();
  expect(res.statusCode).toEqual(200);
  expect(res.body.status).toEqual(true);
});
```

Ilustración 75. Prueba endpoint eliminar usuario

```
test('deberia iniciar sesion un usuario', async () => {
  const res = await request(app)
    .post('/api/users/signin')
    .send({
      "nick_usu": "Schied",
      "contra_usu": "12345789A!"
    });
  expect(res.statusCode).toEqual(200);
  expect(res.body.status).toEqual(true);
  expect(res.body.Resp.message).toEqual('Valid Credentials');
});
```

Ilustración 76. Prueba endpoint iniciar sesión

```
PASS test/usuario.test.js
  Usuario Endpoints
    ✓ deberia obtener todos los usuarios (804 ms)
    ✓ deberia obtener usuario por su cedula (100 ms)
    ✓ deberia crear un usuario (219 ms)
    ✓ deberia eliminar un usuario (99 ms)
    ✓ deberia iniciar sesion un usuario (186 ms)
```

Ilustración 77. Resultado de las pruebas del servicio usuario

5.2.3.2. Pruebas del servicio producto

```
describe('Producto Endpoints', () => {  
  test('deberia obtener todos los productos', async () => {  
    const res = await request(app)  
      .get('/api/products/')  
      .send();  
    expect(res.statusCode).toEqual(200);  
    expect(res.body.status).toEqual(true);  
  });  
});
```

Ilustración 78. Prueba endpoint obtener productos

```
test('deberia obtener producto por su id', async () => {  
  const res = await request(app)  
    .get('/api/products/'+1)  
    .send();  
  expect(res.statusCode).toEqual(200);  
  expect(res.body.status).toEqual(true);  
  expect(res.body.Resp.data[0].nombre_prod).toEqual('Corte de sandalia')  
});
```

Ilustración 79. Prueba endpoint obtener producto por id

```
test('deberia obtener productos filtrados por calzado', async () => {  
  const res = await request(app)  
    .post('/api/products/filter')  
    .send({  
      nombre_prod: "",  
      tipo_prod: "",  
      calzado_prod: "Sandalia",  
      genero_prod: ""  
    });  
  expect(res.statusCode).toEqual(200);  
  expect(res.body.status).toEqual(true);  
});
```

Ilustración 80. Prueba endpoint obtener productos filtrados por calzado

```
PASS test/producto.test.js  
Producto Endpoints  
  ✓ deberia obtener todos los productos (1310 ms)  
  ✓ deberia obtener producto por su id (121 ms)  
  ✓ deberia obtener productos filtrados por calzado (227 ms)
```

Ilustración 81. Resultados de las pruebas del servicio producto

5.2.3.3. Pruebas del servicio material

```
describe('Material Endpoints', () => {
  test('deberia obtener todos los materiales', async () => {
    const res = await request(app)
      .get('/api/materials/')
      .send();
    expect(res.statusCode).toEqual(200);
    expect(res.body.status).toEqual(true);
  });
});
```

Ilustración 82. Prueba endpoint obtener materiales

```
test('deberia obtener material por su codigo', async () => {
  const res = await request(app)
    .get('/api/materials/'+ 'S902LA')
    .send();
  expect(res.statusCode).toEqual(200);
  expect(res.body.status).toEqual(true);
});
```

Ilustración 83. Prueba endpoint obtener material por código

```
test('deberia obtener materiales filtrados por color', async () => {
  const res = await request(app)
    .post('/api/materials/filter')
    .send({
      codigo_mat: "",
      nombre_mat: "",
      color_mat: "Negro",
      unidad_mat: ""
    });
  expect(res.statusCode).toEqual(200);
  expect(res.body.status).toEqual(true);
});
```

Ilustración 84. Prueba endpoint obtener materiales filtrados por color

```
PASS test/material.test.js
  Material Endpoints
    ✓ deberia obtener todos los materiales (892 ms)
    ✓ deberia obtener material por su codigo (102 ms)
    ✓ deberia obtener materiales filtrados por color (135 ms)
```

Ilustración 85. Resultados de las pruebas del servicio material

5.2.3.4. Pruebas del servicio cliente

```
describe('Cliente Endpoints', () => {
  test('deberia obtener todos los clientes', async () => {
    const res = await request(app)
      .get('/api/clientes/')
      .send();
    expect(res.statusCode).toEqual(200);
    expect(res.body.status).toEqual(true);
  });
});
```

Ilustración 86. Prueba endpoint obtener clientes

```
test('deberia obtener cliente por su cedula', async () => {
  const res = await request(app)
    .get('/api/clientes/'+1004852147)
    .send();
  expect(res.statusCode).toEqual(200);
  expect(res.body.status).toEqual(true);
});
```

Ilustración 87. Prueba endpoint obtener cliente por cedula

```
PASS test/cliente.test.js
  Cliente Endpoints
    ✓ deberia obtener todos los clientes (1162 ms)
    ✓ deberia obtener cliente por su cedula (199 ms)
```

Ilustración 88. Resultados de las pruebas del servicio cliente

5.2.3.5. Pruebas del servicio empresa

```
describe('Empresa Endpoints', () => {
  test('deberia obtener todas las empresas', async () => {
    const res = await request(app)
      .get('/api/empresas/')
      .send();
    expect(res.statusCode).toEqual(200);
    expect(res.body.status).toEqual(true);
  });
});
```

Ilustración 89. Prueba endpoint obtener empresas

```
test('deberia obtener empresa por su identificacion', async () => {
  const res = await request(app)
    .get('/api/empresas/'+5821484)
    .send();
  expect(res.statusCode).toEqual(200);
  expect(res.body.status).toEqual(true);
});
```

Ilustración 90. Prueba endpoint obtener empresa por identificación

```
PASS test/empresa.test.js
  Cliente Endpoints
    ✓ deberia obtener todos las empresas (997 ms)
    ✓ deberia obtener empresa por su identificacion (112 ms)
```

Ilustración 91. Resultados de las pruebas del servicio empresa

5.2.3.6. Pruebas del servicio ingresos productos

```
describe('Ingreso producto Endpoints', () => {
  test('deberia obtener todos los ingresos de productos', async () => {
    const res = await request(app)
      .get('/api/ingresosP/')
      .send();
    expect(res.statusCode).toEqual(200);
    expect(res.body.status).toEqual(true);
  });
});
```

Ilustración 92. Prueba endpoint obtener ingresos de productos

```
test('deberia obtener ingreso producto por su id', async () => {
  const res = await request(app)
    .get('/api/ingresosP/id/'+1)
    .send();
  expect(res.statusCode).toEqual(200);
  expect(res.body.status).toEqual(true);
});
```

Ilustración 93. Prueba endpoint obtener ingreso por su id

```
test('deberia obtener ingreso producto por usuario', async () => {
  const res = await request(app)
    .get('/api/ingresosP/usu/'+1004845348)
    .send();
  expect(res.statusCode).toEqual(200);
  expect(res.body.status).toEqual(true);
});
```

Ilustración 94. Prueba endpoint obtener ingreso por usuario

```
PASS test/ingresoP.test.js
  Cliente Endpoints
    ✓ deberia obtener todos los ingresos de productos (1008 ms)
    ✓ deberia obtener ingreso producto por su id (115 ms)
    ✓ deberia obtener ingreso producto por usuario (159 ms)
```

Ilustración 95. Resultados de las pruebas del servicio ingresos productos

5.2.3.7. Pruebas del servicio ingresos materiales

```
describe('Ingreso material Endpoints', () => {
  test('deberia obtener todos los ingresos de materiales', async () => {
    const res = await request(app)
      .get('/api/ingresosM/')
      .send();
    expect(res.statusCode).toEqual(200);
    expect(res.body.status).toEqual(true);
  });
});
```

Ilustración 96. Prueba endpoint obtener ingresos de materiales

```
test('deberia obtener ingreso material por su id', async () => {
  const res = await request(app)
    .get('/api/ingresosM/id/'+1)
    .send();
  expect(res.statusCode).toEqual(200);
  expect(res.body.status).toEqual(true);
});
```

Ilustración 97. Prueba endpoint obtener ingreso material por id

```
test('deberia obtener ingreso material por usuario', async () => {
  const res = await request(app)
    .get('/api/ingresosM/usu/'+1004845348)
    .send();
  expect(res.statusCode).toEqual(200);
  expect(res.body.status).toEqual(true);
});
```

Ilustración 98. Prueba endpoint obtener ingreso material por usuario

```
PASS test/ingresoM.test.js
Ingreso material Endpoints
  ✓ deberia obtener todos los ingresos de materiales (1094 ms)
  ✓ deberia obtener ingreso material por su id (194 ms)
  ✓ deberia obtener ingreso material por usuario (185 ms)
```

Ilustración 99. Resultados de las pruebas del servicio ingresos materiales

5.2.3.8. Pruebas del servicio egresos productos

```
describe('Egreso producto Endpoints', () => {
  test('deberia obtener todos los egresos de productos', async () => {
    const res = await request(app)
      .get('/api/egresosP/')
      .send();
    expect(res.statusCode).toEqual(200);
    expect(res.body.status).toEqual(true);
  });
});
```

Ilustración 100. Prueba endpoint obtener egresos de productos

```
test('deberia obtener egreso producto por su id', async () => {
  const res = await request(app)
    .get('/api/egresosP/id/'+1)
    .send();
  expect(res.statusCode).toEqual(200);
  expect(res.body.status).toEqual(true);
});
```

Ilustración 101. Prueba endpoint obtener egresos de productos por id

```
test('deberia obtener egreso producto por usuario', async () => {
  const res = await request(app)
    .get('/api/egresosP/usu/'+1004845348)
    .send();
  expect(res.statusCode).toEqual(200);
  expect(res.body.status).toEqual(true);
});
```

Ilustración 102. Prueba endpoint obtener egresos de productos por usuario

```
PASS test/egresoP.test.js
Egreso producto Endpoints
  ✓ deberia obtener todos los egresos de productos (1034 ms)
  ✓ deberia obtener egreso producto por su id (119 ms)
  ✓ deberia obtener egreso producto por usuario (163 ms)
```

Ilustración 103. Resultados de las pruebas del servicio egresos productos

5.2.3.9. Pruebas del servicio egresos materiales

```
describe('Egreso material Endpoints', () => {
  test('deberia obtener todos los egresos de materiales', async () => {
    const res = await request(app)
      .get('/api/egresosM/')
      .send();
    expect(res.statusCode).toEqual(200);
    expect(res.body.status).toEqual(true);
  });
});
```

Ilustración 104. Prueba endpoint obtener egresos de materiales

```
test('deberia obtener egreso material por su id', async () => {
  const res = await request(app)
    .get('/api/egresosM/id/'+1)
    .send();
  expect(res.statusCode).toEqual(200);
  expect(res.body.status).toEqual(true);
});
```

Ilustración 105. Prueba endpoint obtener egreso de material por su id

```
test('deberia obtener egreso material por usuario', async () => {  
  const res = await request(app)  
    .get('/api/egresosM/usu/'+1004845348)  
    .send();  
  expect(res.statusCode).toEqual(200);  
  expect(res.body.status).toEqual(true);  
});
```

Ilustración 106. Prueba endpoint obtener egresos de material por usuario

```
PASS test/egresoM.test.js  
Egreso material Endpoints  
  ✓ deberia obtener todos los egresos de materiales (857 ms)  
  ✓ deberia obtener egreso material por su id (107 ms)  
  ✓ deberia obtener egreso material por usuario (100 ms)
```

Ilustración 107. Resultados de las pruebas del servicio egresos materiales

6. CONCLUSIONES

El desarrollo de la aplicación web de escritorio se llevó a cabo con tecnologías las cuales no había tenido la oportunidad de trabajar a profundidad, como lo fue el framework electronjs el cual es una base importante para este trabajo de grado ya que me permitió el poder usar el potencial de una aplicación web dentro de una aplicación de escritorio logrando así poder tener una interfaz estética y agradable para el usuario, permitiéndome adquirir nuevos conocimientos así como experiencia al trabajar de igual forma con la arquitectura hexagonal la cual determino la forma en que se desarrolló el backend de la aplicación.

Con el presente trabajo de grado se logró la construcción de una aplicación web de escritorio la cual va a facilitar la gestión del inventario de la fábrica Prefabricados Marlon – Stefan, en la cual el usuario va a poder visualizar y controlar el inventario de la fábrica, poder registrar y visualizar los clientes que tienen, así como también se tendrá un control sobre el registro de la salida y entrada de los productos para el calzado fabricados la cual fue una de las necesidades por la cual surgió este proyecto.

A la aplicación se le realizaron pruebas unitarias para la comprobación del funcionamiento de la misma, a partir de dichas pruebas se pudo verificar que todos los servicios que ofrece la aplicación funcionan correctamente.

7. REFERENCIAS BIBLIOGRÁFICAS

- Apiumhub. (20 de Abril de 2020). *¿Qué Es La Arquitectura Hexagonal? Definición Y Ejemplos*. Obtenido de <https://apiumhub.com/es/tech-blog-barcelona/arquitectura-hexagonal/>
- ARIMETRICS. (14 de Abril de 2021). *Qué es Framework*. Obtenido de <https://www.arimetrics.com/glosario-digital/framework>
- Asana. (16 de Mayo de 2022). *Programación extrema (XP)*. Obtenido de <https://asana.com/es/resources/extreme-programming-xp>
- Atlassian. (s.f.). *¿Cuales son los roles de scrum?* Obtenido de <https://www.atlassian.com/es/agile/scrum/roles>
- Atlassian. (Noviembre de 2018). *Atlassian SCRUM*. Obtenido de Atlassian SCRUM: <https://www.atlassian.com/es/agile/scrum>
- Betancur Rosas, V. A. (2018). *APLICACIÓN WEB PARA LA GESTIÓN DE INVENTARIO DE REACTIVOS DEL LABORATORIO DE QUÍMICA DE LA UNIVERSIDAD DE PAMPLONA SEDE VILLA DEL ROSARIO*. Villa del Rosario: UNIVERSIDAD DE PAMPLONA.
- Diaz Arrieta, R. H., & Ortiz Vinueza, J. L. (2018). *Desarrollo de una aplicación web y móvil para la gestión de inventario y pedidos utilizando servicios RESTful*. Riobamba: Escuela Superior Politécnica de Chimborazo. Obtenido de <http://dspace.esPOCH.edu.ec/handle/123456789/9128>
- EasyAPP Code. (02 de Septiembre de 2020). *Patrón de diseño MVC. ¿Qué es y cómo puedo utilizarlo?* Obtenido de <https://www.easyappcode.com/patron-de-diseno-mvc-que-es-y-como-puedo-utilizarlo>

Hernandez Castro, J. F., & Herrera Briceño, K. V. (2021). *Aplicación web para controlar el inventario de la empresa first medical de colombia*. Bogota: Universidad del rosario. Obtenido de <https://repositorio.escuelaing.edu.co/handle/001/1951>

Hostinger. (29 de Julio de 2021). *¿Qué es Angular? – Una guía para principiantes*. Obtenido de <https://www.hostinger.es/tutoriales/que-es-angular>

HTMLGOODIES. (03 de Octubre de 2018). *El patrón Model-View-ViewModel y el desarrollo angular*. Obtenido de <https://www.htmlgoodies.com/javascript/the-model-view-viewmodel-pattern-and-angular-development/>

Infranetworking. (19 de Noviembre de 2019). *Servidor PostgreSQL*. Obtenido de <https://blog.infranetworking.com/servidor-postgresql/>

IslaVisual. (13 de 11 de 2012). *Diferencias entre scrum y xp*. Obtenido de https://islavisual.com/articulos/desarrollo_web/diferencias-entre-scrum-y-xp.php

IslaVisual. (2018). *Scrum versus XP: similitudes y diferencias*. Tecnología Investigación y Academia. Obtenido de <https://revistas.udistrital.edu.co/index.php/tia/article/view/10496>

Jaramillo Bolívar, D. (2019). *Desarrollo de una plataforma de gestión de eventos para el área de gestión humana y administrativa de Ceiba Software*. Medellín: Universidad de Antioquia. Obtenido de <http://hdl.handle.net/10495/15518>

Marquez Solis, P. Y. (2020). *Desarrollo de una aplicación web para la gestión de inventarios de la empresa 1Solutions S.A.C. - Huancayo*. Huancayo: Universidad Continental. Obtenido de <https://hdl.handle.net/20.500.12394/9136>

mrHouston. (18 de Enero de 2018). *Qué son las Aplicaciones Web*. Obtenido de <https://mrhouston.net/blog/que-son-las-aplicaciones-web/>

NIIXER. (16 de Noviembre de 2020). *¿Qué es node js?* Obtenido de <https://niixer.com/index.php/2020/11/16/node-js-en-el-desarrollo-web-su-importancia-y-uso-actualmente/>

OpenWebinars. (18 de Septiembre de 2020). *Extreme Programming: Qué es y cómo aplicarlo*. Obtenido de <https://openwebinars.net/blog/extreme-programming-que-es-y-como-aplicarlo/>

Oracle. (s.f.). *¿Qué es una base de datos relacional (sistema de gestión de bases de datos relacionales)?* Obtenido de <https://www.oracle.com/mx/database/what-is-a-relational-database/>

Parada Gutiérrez, O. (2009). Un enfoque multicriterio para la toma de decisiones en la gestión de inventarios. *Cuadernos de Administración*, 169-187. Obtenido de http://www.scielo.org.co/scielo.php?script=sci_arttext&pid=S0120-35922009000100009&lng=en&tlng=es.

Picon Rincon, J. A. (2014). DESARROLLO DE UN SOFTWARE PARA HACER ZOOMETRIA EN GANADO BLANCO OREJINEGRO PARA LA UNIVERSIDAD FRANCISCO DE PAULA SANTANDER OCAÑA. *Universidad Francisco de Paula Santander Ocaña*, 127. Obtenido de <http://repositorio.ufpso.edu.co/jspui/handle/123456789/1010>

Platzi. (10 de Noviembre de 2016). *Electron.js: conoce el framework JS para aplicaciones de escritorio*. Obtenido de <https://platzi.com/blog/aplicaciones-escritorio-electron-js/>

Romero, E. M. (2019). *Diseño e implementación de sistema de inventarios para el almacén de pinturas y ferretería Ferrecolor*. Villavicencio: Universidad Cooperativa de Colombia. Obtenido de

https://repository.ucc.edu.co/bitstream/20.500.12494/8557/3/2019_Diseño_implementación_sistema.pdf

SURÁTICA Software. (28 de Marzo de 2022). *¿Qué es un framework?* Obtenido de <https://www.suratica.es/que-es-un-framework/>

Wrike. (08 de Julio de 2021). *Qué es una API: todo lo que necesitas saber.* Obtenido de <https://www.wrike.com/es/blog/que-es-una-api-necesitas-saber/>

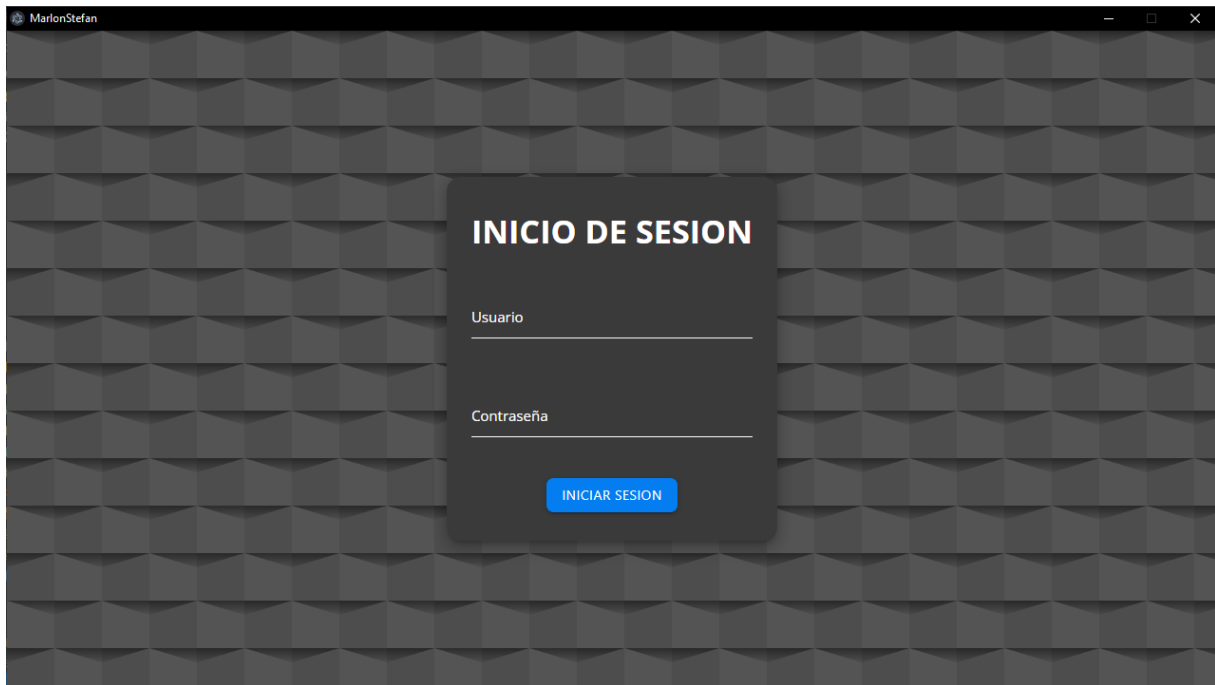
YeePLY. (22 de Agosto de 2019). *¿Qué son las pruebas unitarias?* Obtenido de <https://www.yeeply.com/blog/que-son-pruebas-unitarias/>

ANEXOS

ANEXO A. MANUAL DE USUARIO

Una vez se ejecuta la aplicación, la primera pantalla que se visualizara por primera vez es la de inicio de sesión como se ve a continuación.

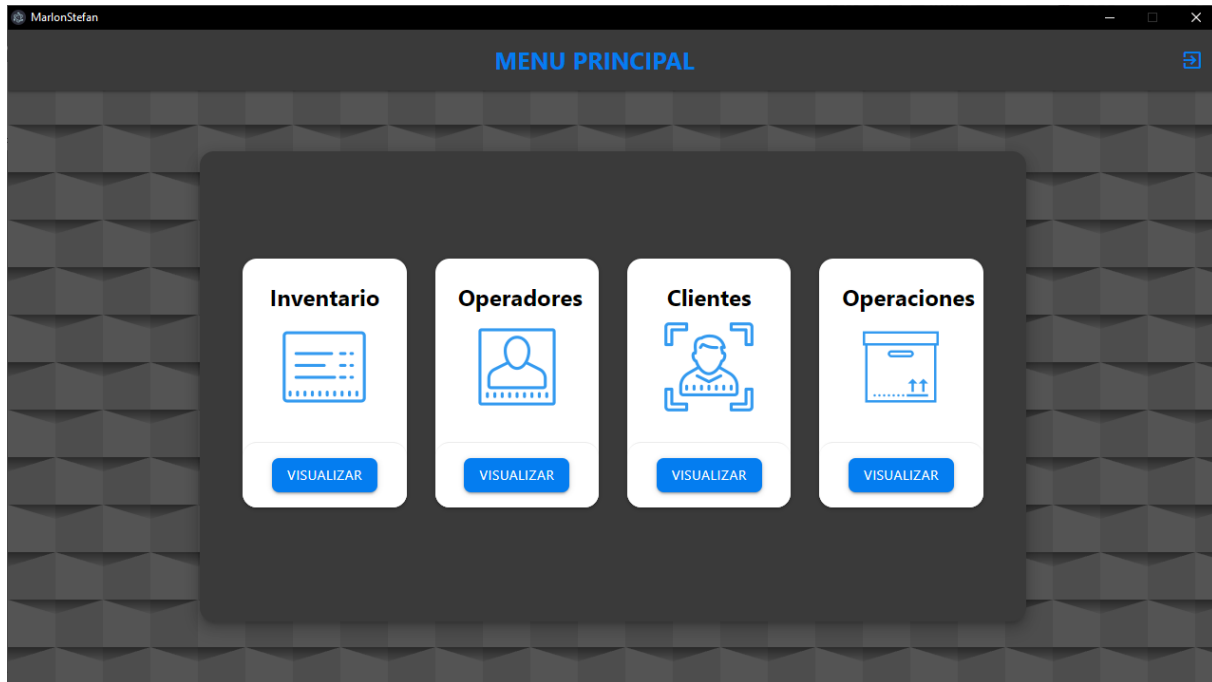
1. Inicio de sesión



En esta interfaz el usuario deberá ingresar sus credenciales, las cuales son un usuario y contraseña para acceder al menú del tipo de usuario al que pertenezca. En el caso de los usuarios de tipo Operador, dichas credenciales serán suministradas por el Administrador ya que este es el único que cuenta con una interfaz para el registro de los operadores del sistema.

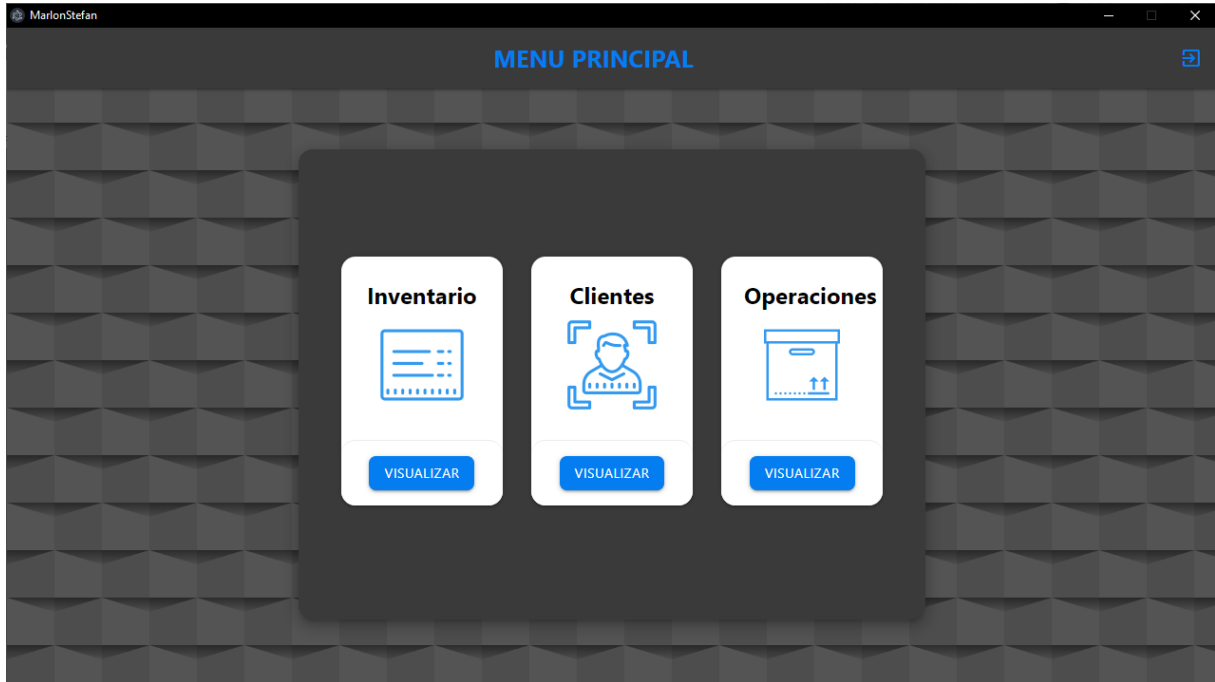
2. Menú principal (Administrador)

Esta interfaz solo se podrá visualizar cuando un usuario de tipo Administrador inicie sesión en el sistema, en este menú se tiene acceso a todas las opciones del sistema de las cuales primero se muestran las principales las cuales son: Inventario, Operadores, Clientes y Operaciones.



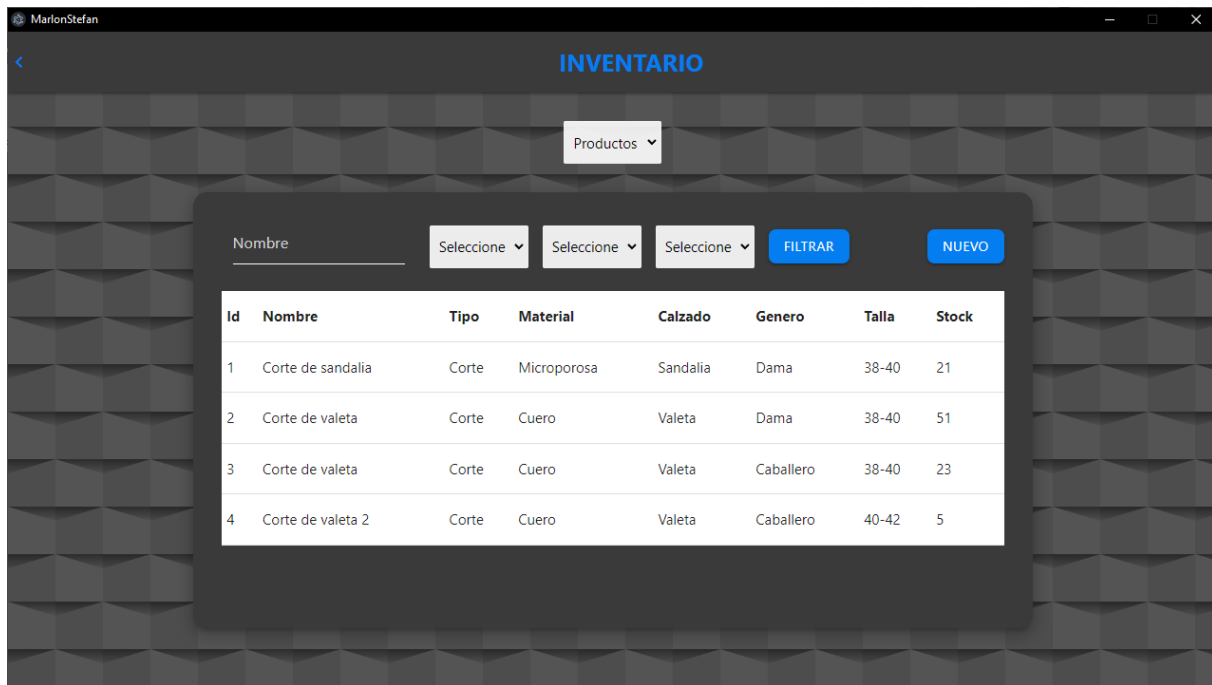
3. Menú principal (Operador)

Esta interfaz solo se podrá visualizar cuando un usuario de tipo Operador inicie sesión en el sistema, en este menú se tiene acceso a algunas de las opciones del sistema de las cuales primero se muestran las principales las cuales son: Inventario, Clientes y Operaciones.

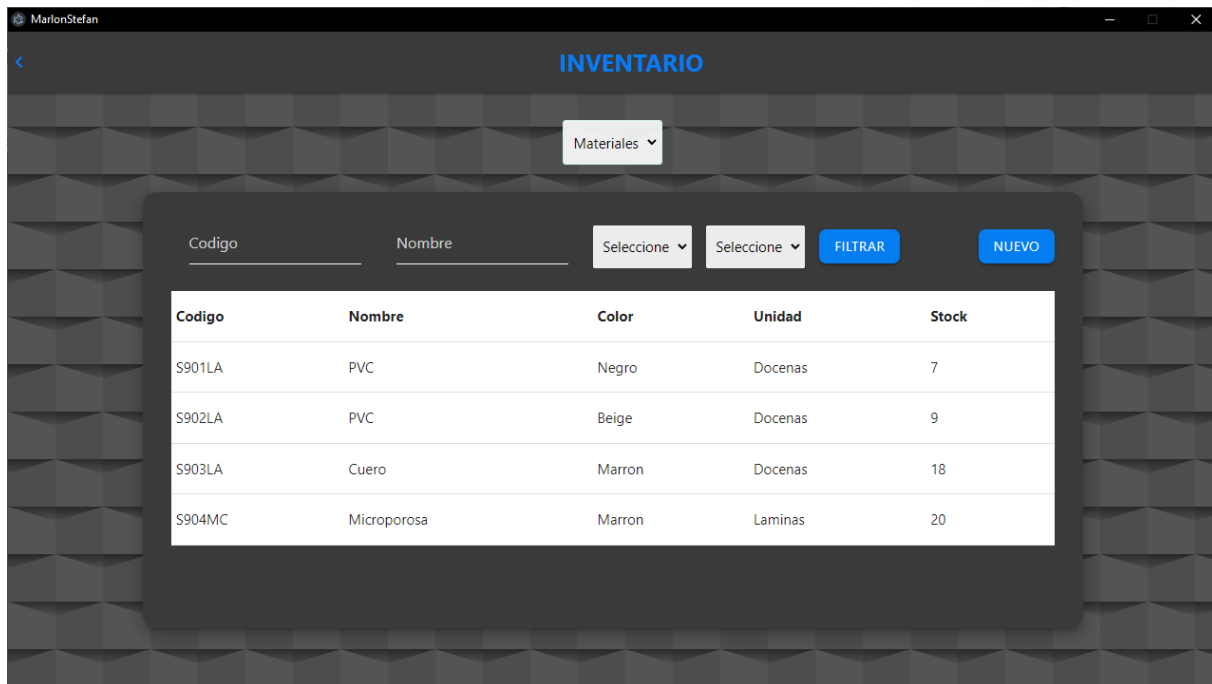


4. Inventario

Esta interfaz se mostrará al seleccionar “Visualizar” inventario desde el menú principal.

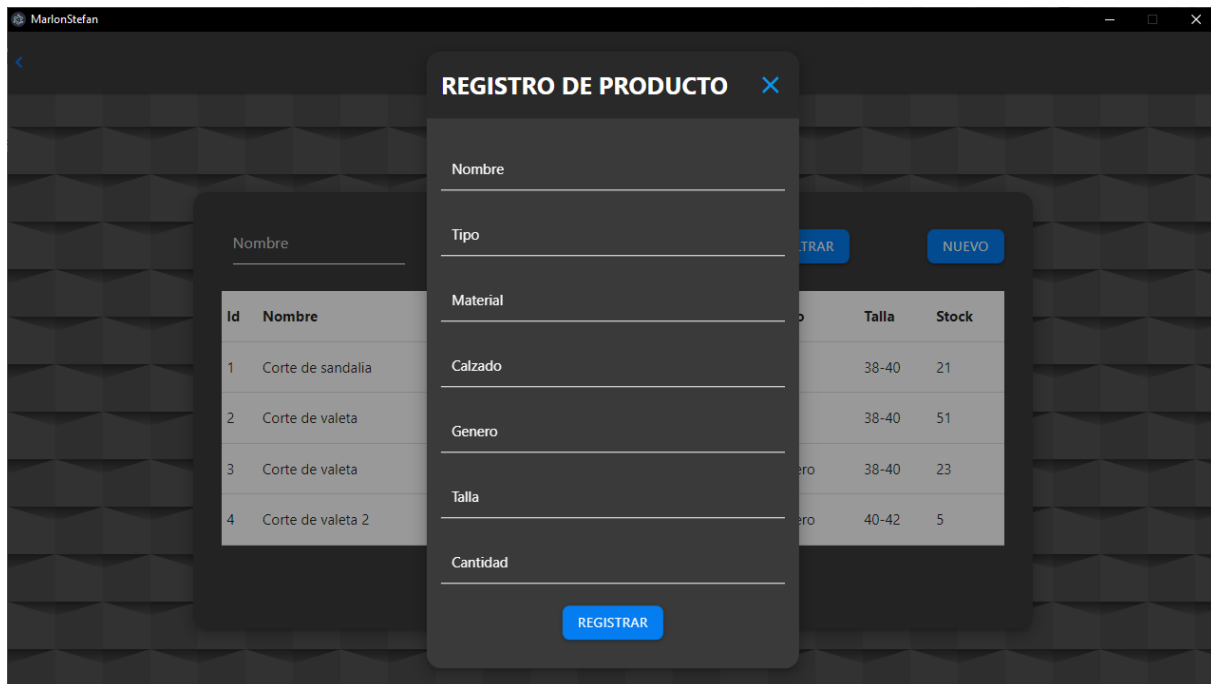


Al ingresar se podrá observar que en la parte superior de la interfaz se encuentra una lista desplegable en la cual se encuentran dos opciones que son: Productos y Materiales. Esta lista permite cambiar entre gestionar los productos o materiales del inventario.



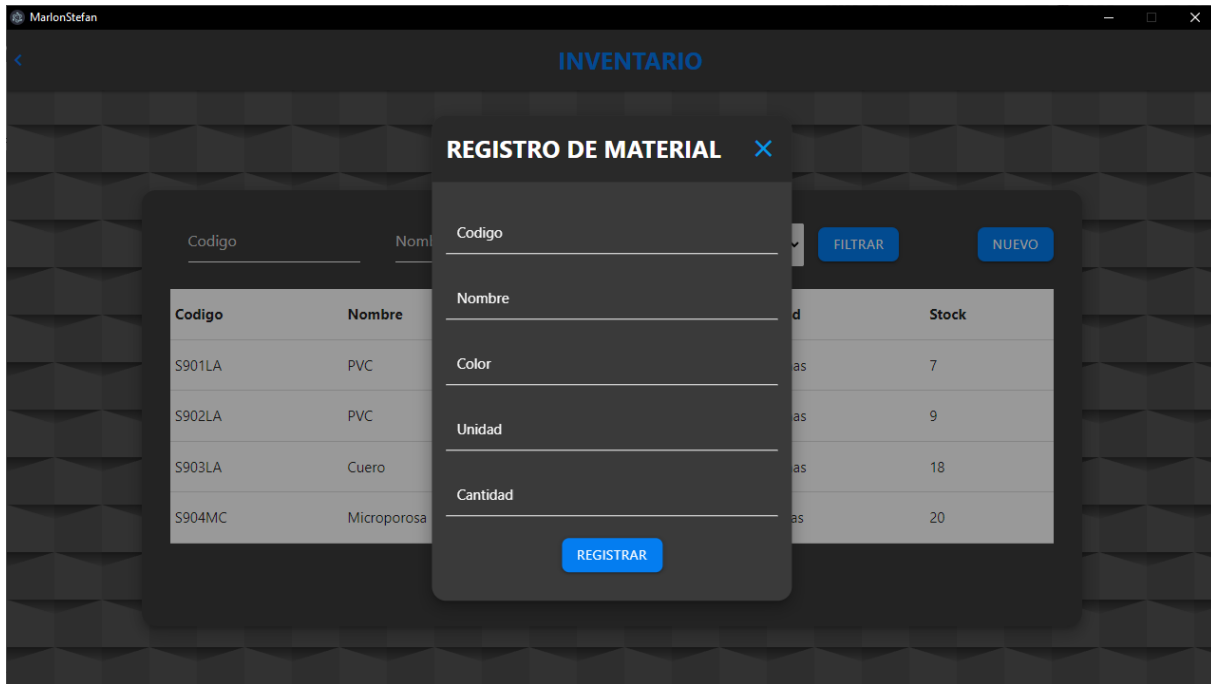
a. Productos

Al seleccionar Productos en la lista desplegable se podrá gestionar los productos del inventario, primeramente se podrán visualizar los productos mediante una tabla la cual contiene los detalles de cada uno, así mismo se cuenta con un filtro el cual cuenta con las opciones de nombre, tipo, calzado y genero para filtrar. Para la creación de un nuevo producto se debe seleccionar el botón “nuevo” el cual desplegará el siguiente modal en el cual se podrá realizar el registro de un nuevo producto.

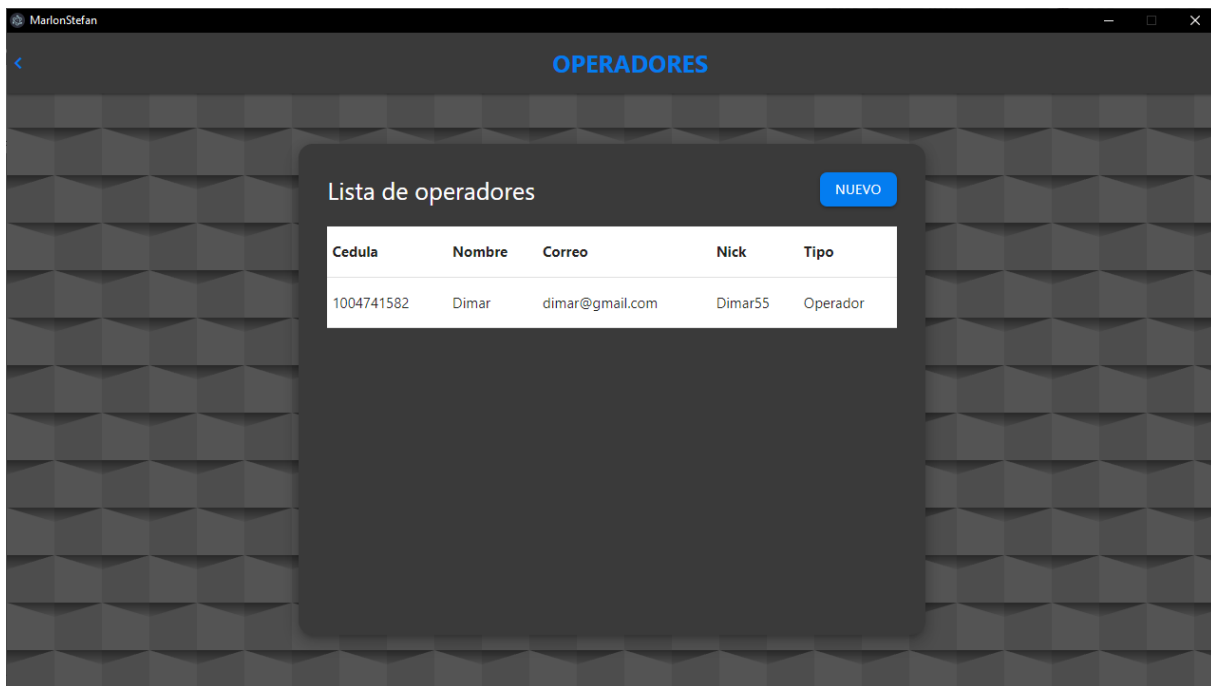


b. Materiales

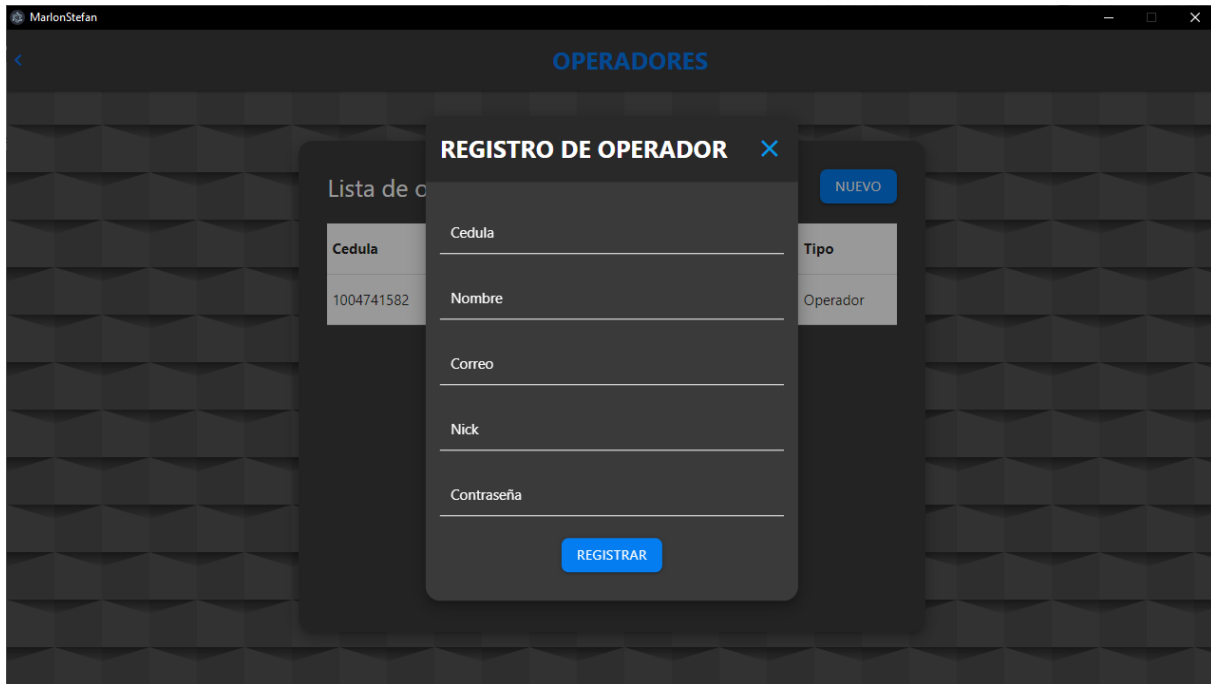
Al seleccionar Materiales en la lista desplegable se podrá gestionar los materiales del inventario, al igual que en los productos se podrán visualizar los materiales mediante una tabla la cual contiene los detalles de cada uno, así mismo se cuenta con un filtro el cual cuenta con las opciones de código, nombre, color y unidad para filtrar. Para la creación de un nuevo material se debe seleccionar el botón “nuevo” el cual desplegará el siguiente modal en el cual se podrá realizar el registro de un nuevo material.



5. Operadores (solo Administrador)

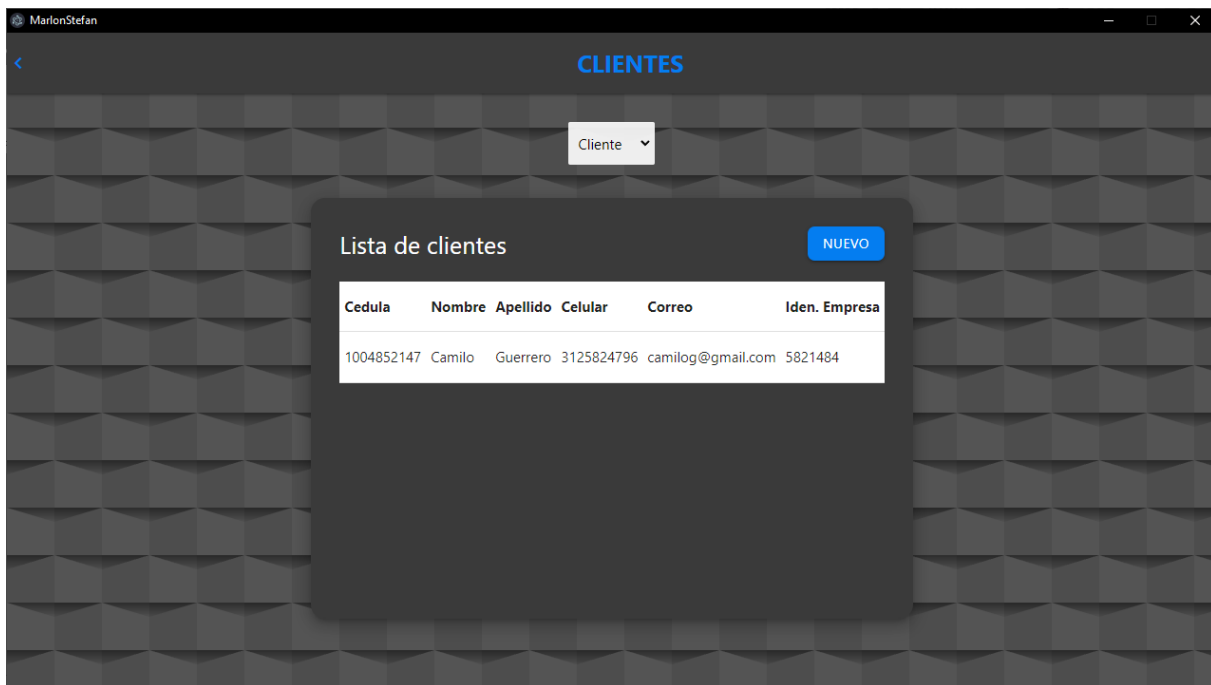


Esta interfaz solo esta disponible para el Administrador, aquí se podrán visualizar los operadores que tienen acceso al sistema. Para crear un nuevo operador se debe seleccionar el botón “nuevo” el cual desplegara un modal para permitir el registro.



6. Clientes

Al seleccionar “Visualizar” clientes desde el menú principal se mostrará la siguiente interfaz.



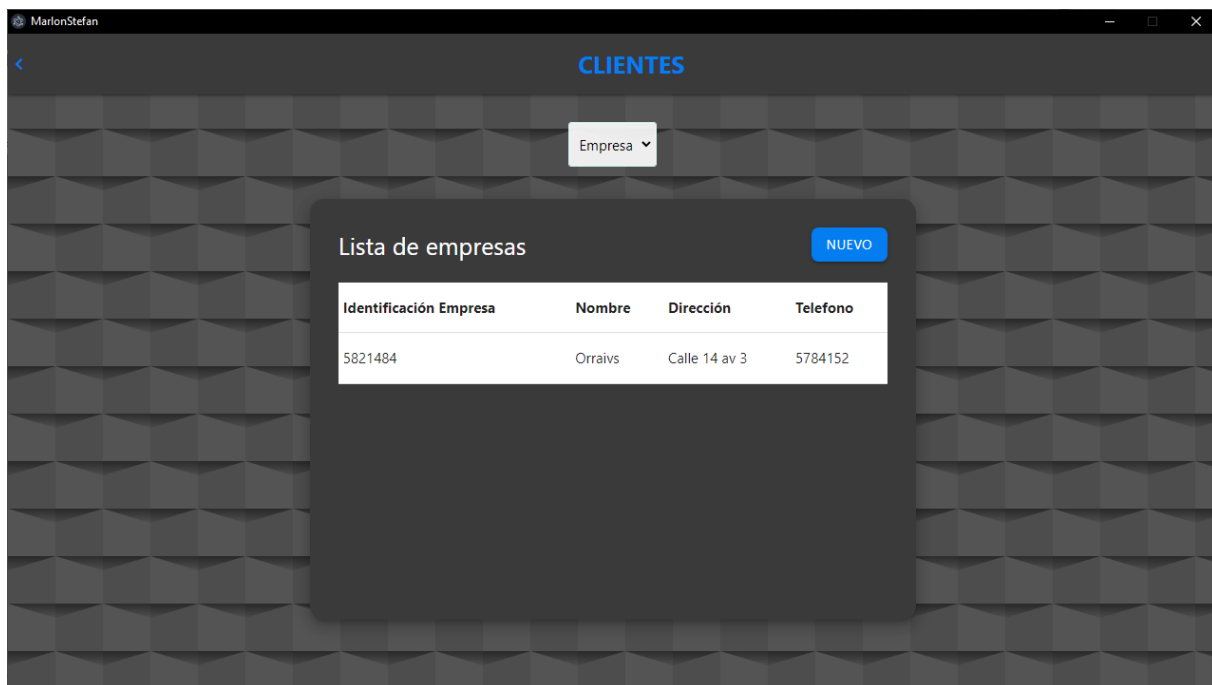
a. Clientes

Esta permite la visualización de los clientes que están registrados en el sistema junto a sus detalles. Para el registro de un nuevo cliente se debe seleccionar el botón “nuevo” en el cual se desplegará un modal donde se deberá llenar los datos del cliente junto a la id de la empresa a la que pertenece.

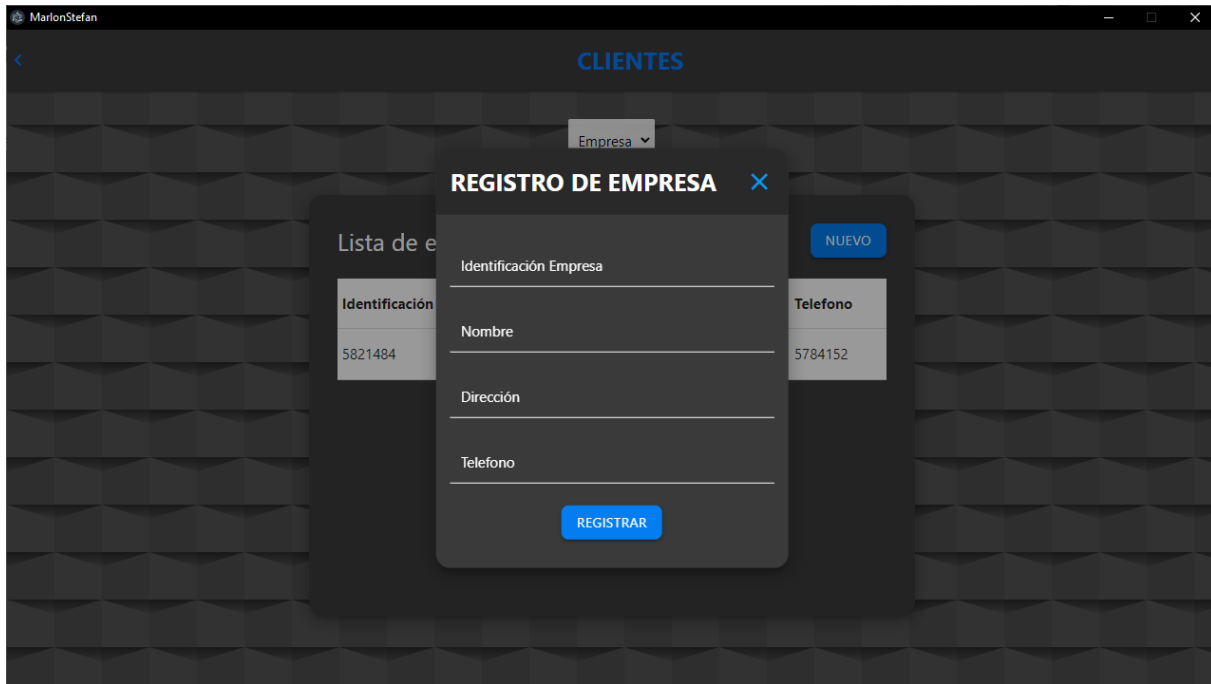
The screenshot shows a web application interface with a dark theme. At the top, there's a header bar with the text 'CLIENTES' in blue. Below the header, a modal titled 'REGISTRO DE CLIENTE' is open, featuring a close button (X) in the top right corner. The modal contains several input fields for client registration: 'Cedula', 'Nombre', 'Apellido', 'Celular', 'Correo', and 'Identificación Empresa'. A blue 'REGISTRAR' button is located at the bottom of the modal. In the background, a 'Lista de clientes' table is partially visible, showing columns for 'Cedula', 'Nombre', and 'Iden. Empresa'. A blue 'NUEVO' button is also visible on the right side of the background interface.

Cedula	Nombre	Iden. Empresa
1004852147	Ce	5821484

b. Empresas

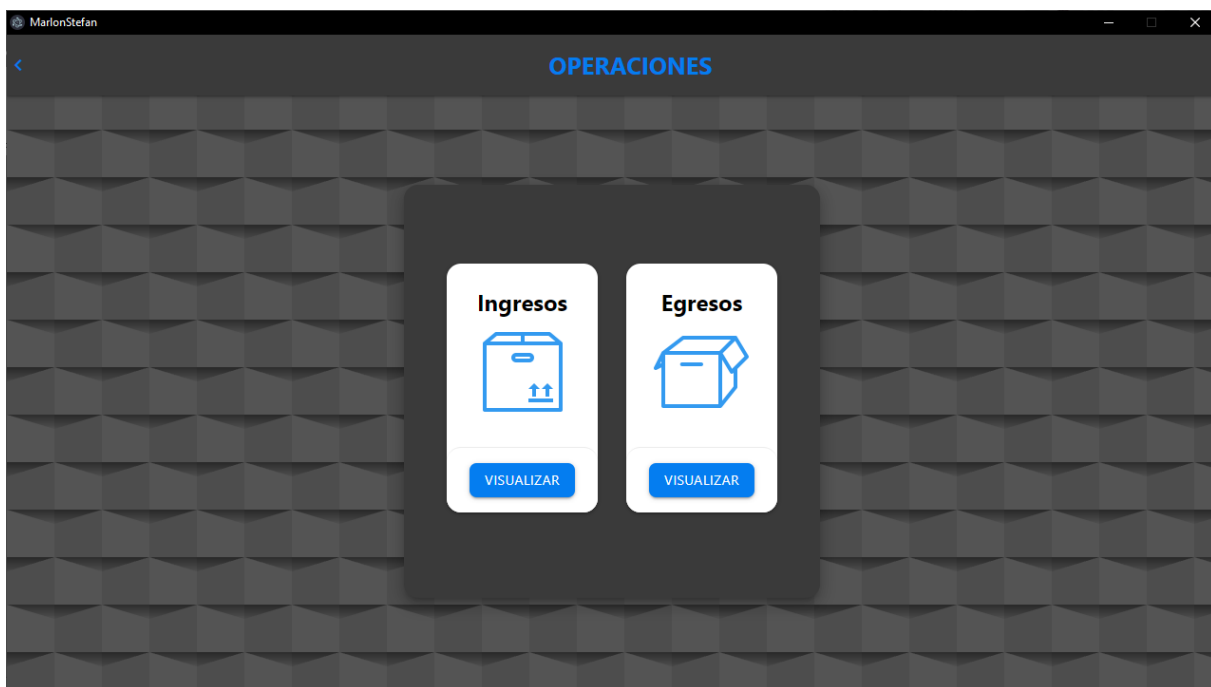


Esta permite la visualización de las empresas que están registradas en el sistema junto a sus detalles. Para el registro de una nueva empresa se debe seleccionar el botón “nuevo” en el cual se desplegará un modal donde se deberá llenar los datos de la empresa como se muestra a continuación.



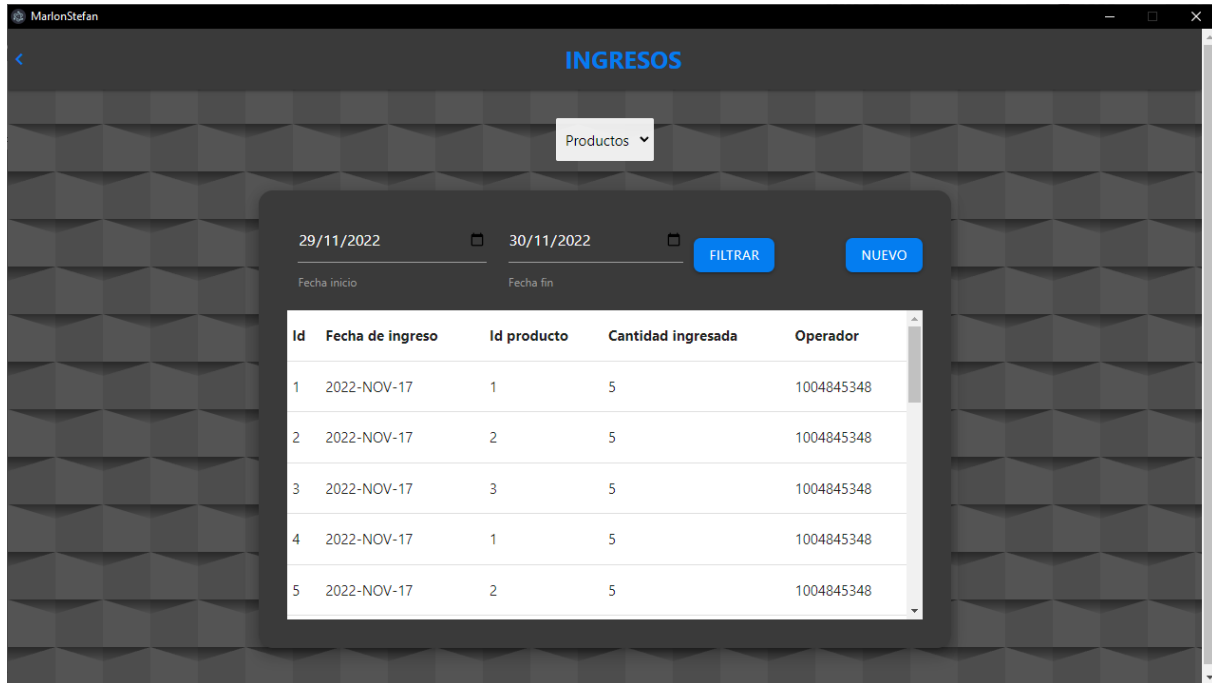
7. Operaciones

Al seleccionar “Visualizar” operaciones desde el menú principal se mostrará la siguiente interfaz, la cual cuenta con 2 botones uno el cual redirige al apartado de ingresos y el otro a egresos como se puede ver a continuación.



a. Ingresos

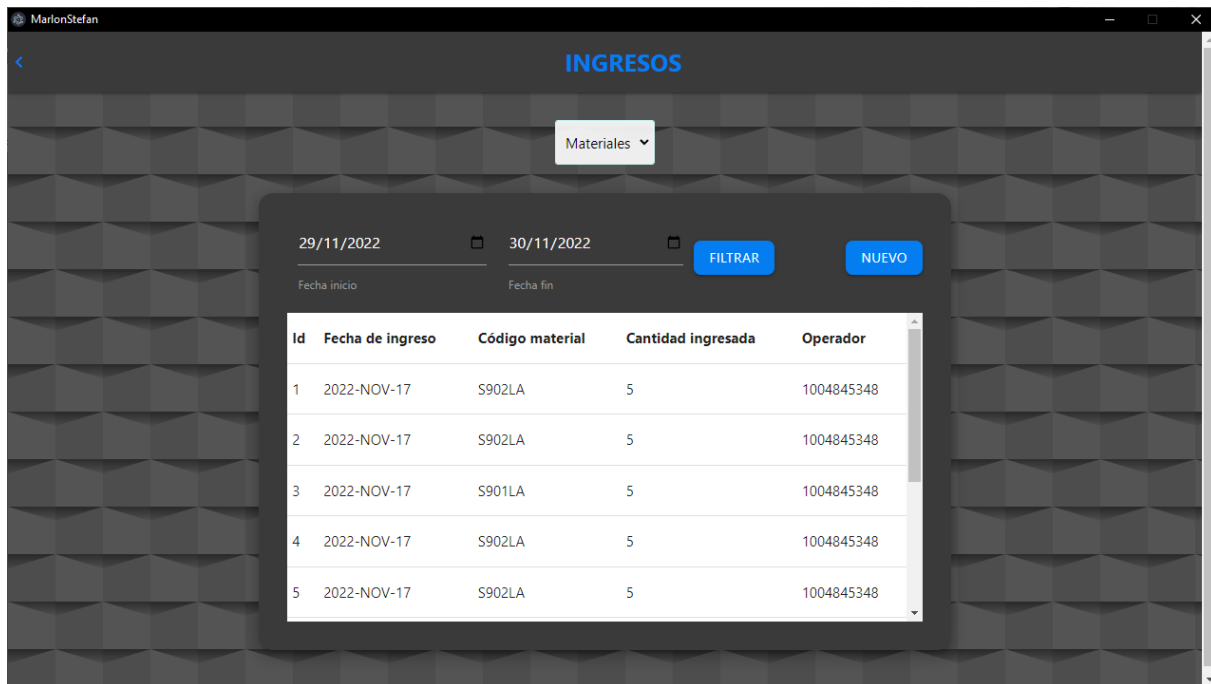
Al seleccionar “Visualizar” ingresos desde el menú de operaciones se mostrará la siguiente interfaz.



The screenshot shows a web application window titled 'INGRESOS'. At the top, there is a dropdown menu labeled 'Productos'. Below this, there are two date pickers: '29/11/2022' (labeled 'Fecha inicio') and '30/11/2022' (labeled 'Fecha fin'). To the right of the date pickers are two buttons: 'FILTRAR' and 'NUEVO'. Below these elements is a table with the following data:

Id	Fecha de ingreso	Id producto	Cantidad ingresada	Operador
1	2022-NOV-17	1	5	1004845348
2	2022-NOV-17	2	5	1004845348
3	2022-NOV-17	3	5	1004845348
4	2022-NOV-17	1	5	1004845348
5	2022-NOV-17	2	5	1004845348

Esta interfaz cuenta con la función de poder cambiar entre gestionar los productos o materiales a través de una lista desplegable.



En ambas opciones se podrá gestionar los ingresos de los productos como de los materiales según se desee, como se puede ver en la ilustración anterior, en esta podremos filtrar los ingresos por un rango de fechas para obtener rápidamente información sobre algún ingreso en concreto. Además, se cuenta con el botón “nuevo” el cual permite el registro de un nuevo ingreso como se verá a continuación.

MarlonStefan

2

Corte de valeta

Corte

Cuero

Valeta

Dama

38-40

51

AÑADIR

3

Corte de valeta

Corte

Cuero

Valeta

Caballero

38-40

23

AÑADIR

4

Corte de valeta 2

Corte

Cuero

Valeta

Caballero

40-42

5

AÑADIR

Detalles ingreso

Id	Nombre	Tipo	Material	Calzado	Genero	Talla	Cantidad	Acción
2	Corte de valeta	Corte	Cuero	Valeta	Dama	38-40	- 10 +	ELIMINAR
1	Corte de sandalia	Corte	Microporosa	Sandalia	Dama	38-40	- 15 +	ELIMINAR

REGISTRAR INGRESO

MarlonStefan

S902LA

PVC

Beige

Docenas

9

AÑADIR

S903LA

Cuero

Marron

Docenas

18

AÑADIR

S904MC

Microporosa

Marron

Laminas

20

AÑADIR

Detalles ingreso

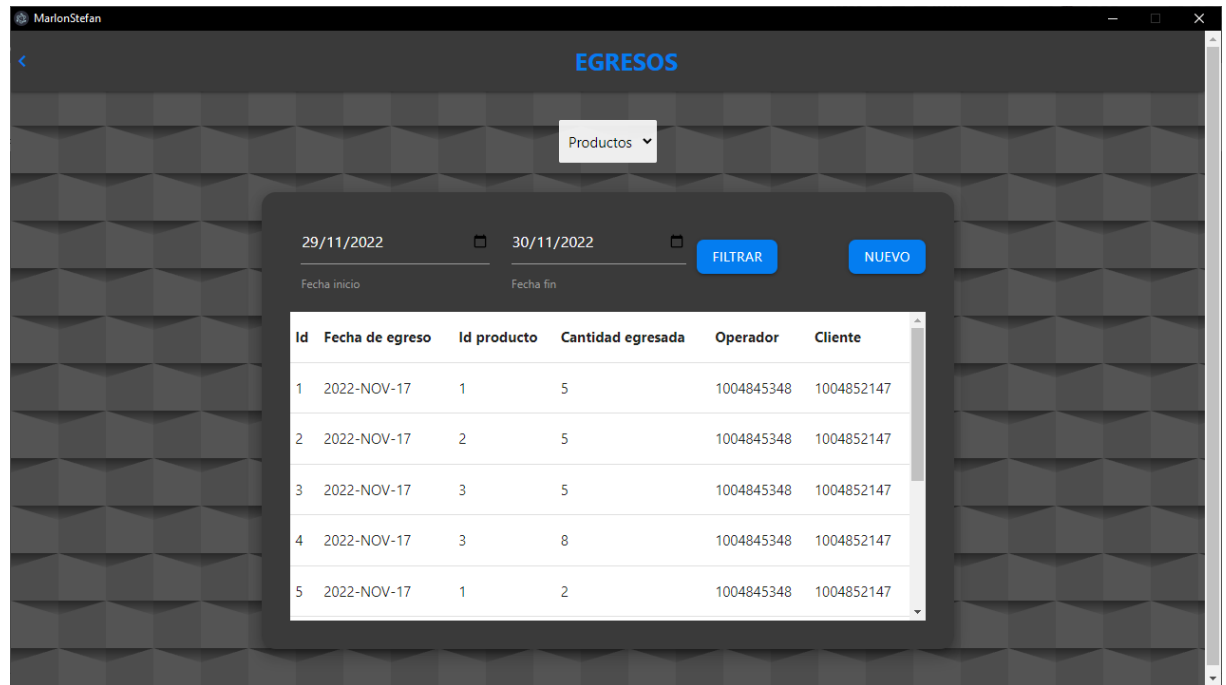
Codigo	Nombre	Color	Unidad	Cantidad	Acción
S901LA	PVC	Negro	Docenas	- 6 +	ELIMINAR
S902LA	PVC	Beige	Docenas	- 4 +	ELIMINAR

REGISTRAR INGRESO

Una vez se hayan agregado los productos o materiales deseados junto a las cantidades se podrá seleccionar “Registrar ingreso” lo cual actualizara la cantidad de los productos o materiales ingresados en el sistema.

b. Egresos

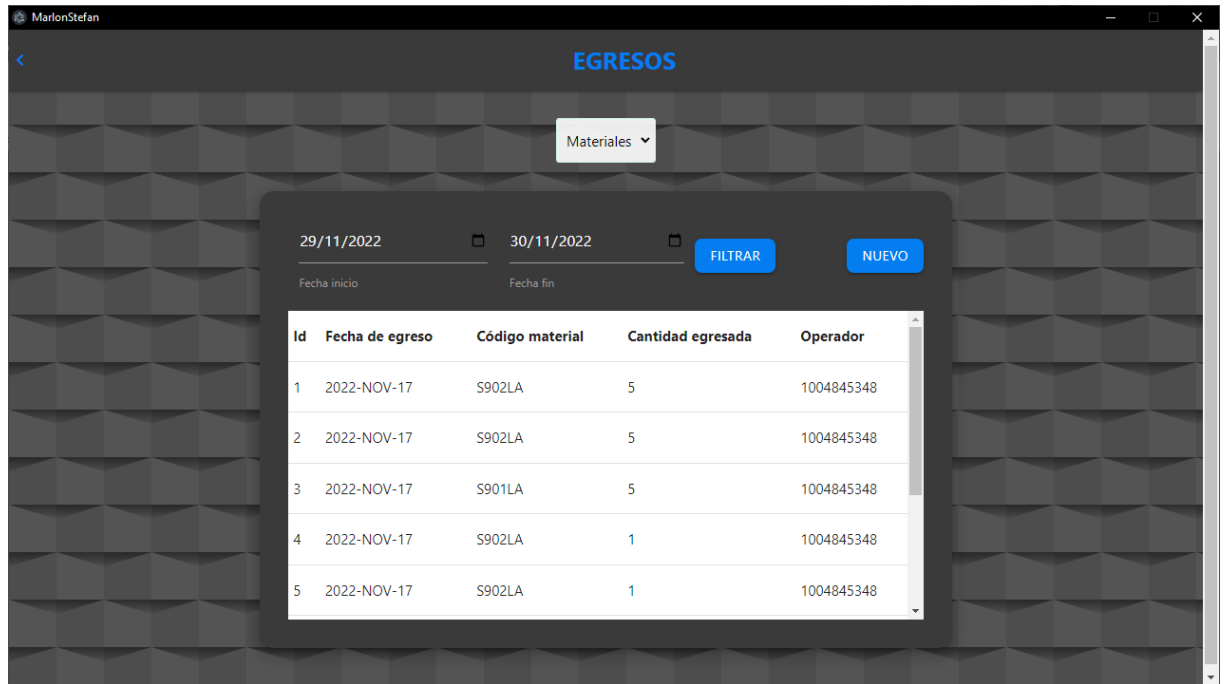
Al seleccionar “Visualizar” egresos desde el menú de operaciones se mostrará la siguiente interfaz.



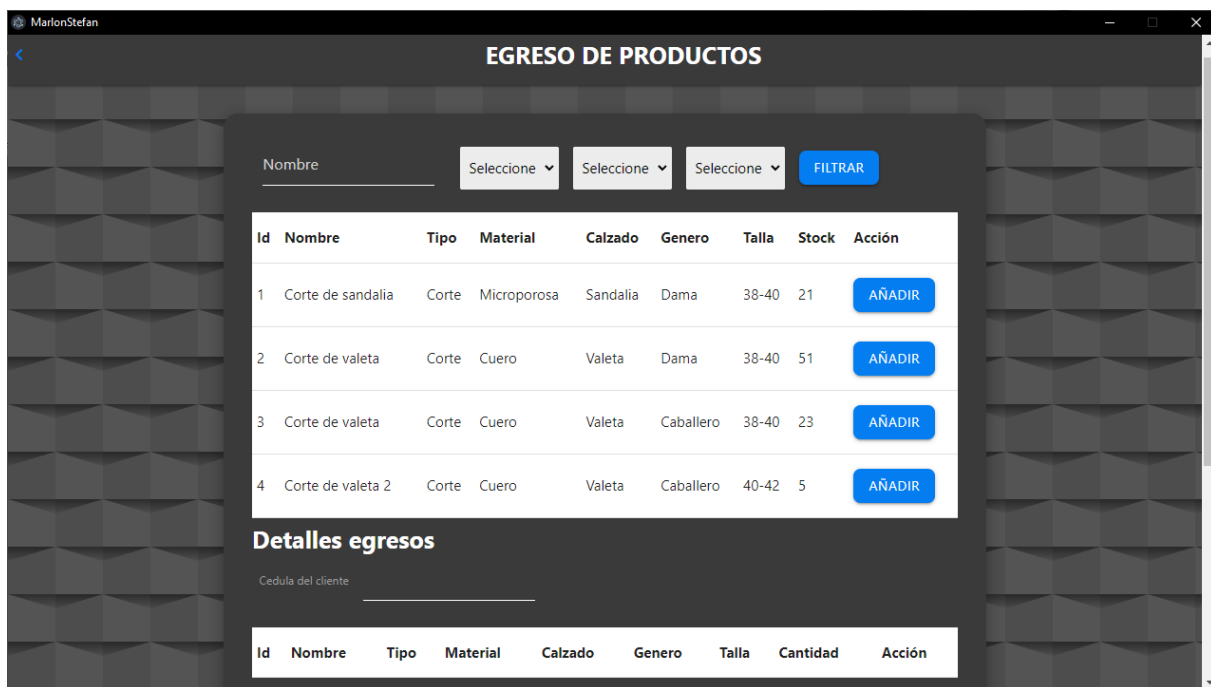
The screenshot displays a web application window titled "MarlonStefan" with a dark theme. The main heading is "EGRESOS" in blue. Below it is a "Productos" dropdown menu. A date range filter is set from "29/11/2022" to "30/11/2022", with "Fecha inicio" and "Fecha fin" labels. There are "FILTRAR" and "NUEVO" buttons. The table below shows five rows of transaction data.

Id	Fecha de egreso	Id producto	Cantidad egresada	Operador	Cliente
1	2022-NOV-17	1	5	1004845348	1004852147
2	2022-NOV-17	2	5	1004845348	1004852147
3	2022-NOV-17	3	5	1004845348	1004852147
4	2022-NOV-17	3	8	1004845348	1004852147
5	2022-NOV-17	1	2	1004845348	1004852147

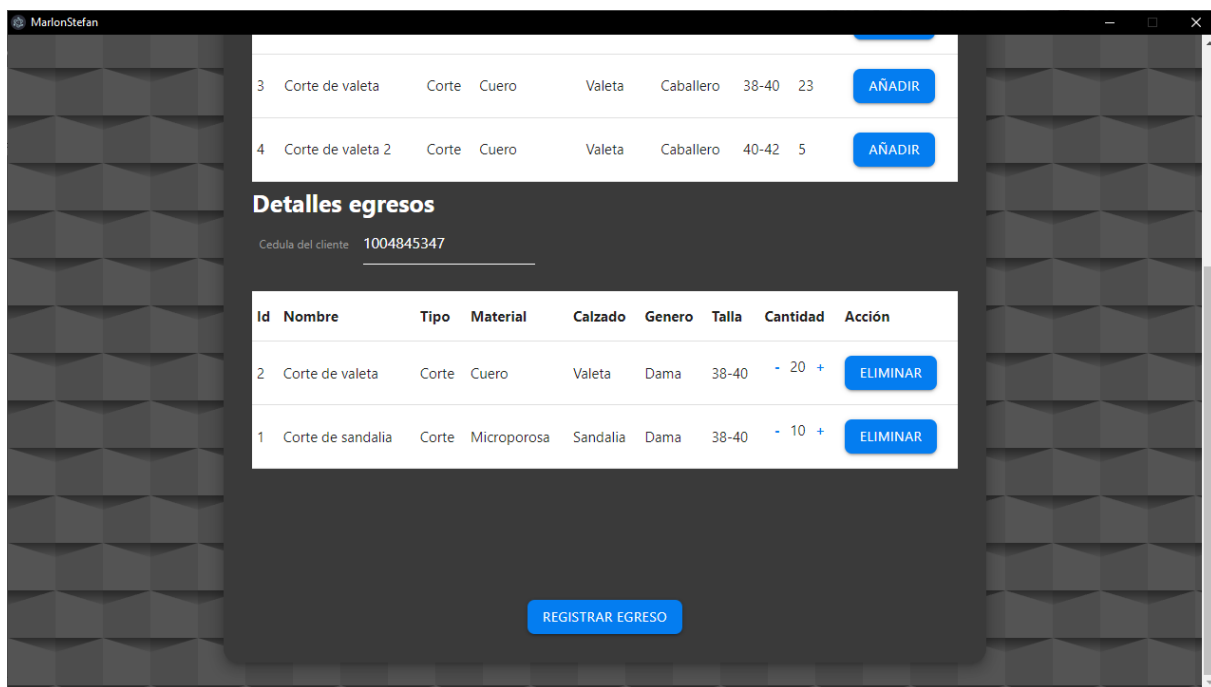
Esta interfaz cuenta con la función de poder cambiar entre gestionar los productos o materiales a través de una lista desplegable.



En ambas opciones se podrá gestionar los egresos de los productos como de los materiales según se desee, como se puede ver en la ilustración anterior, en esta podremos filtrar los egresos por un rango de fechas para obtener rápidamente información sobre algún egreso en concreto. Además, se cuenta con el botón “nuevo” el cual permite el registro de un nuevo egreso como se verá a continuación.



En esta interfaz el usuario podrá visualizar los productos y filtrarlos mediante las mismas opciones del inventario, para agregar un producto al egreso se debe dar clic en “Añadir” lo cual lo agregará a los detalles de egreso como se ve a continuación. Además, se deberá digitar la cedula del cliente al cual se le esta realizando el egreso.



Una vez se hayan agregado los productos deseados junto a las cantidades se podrá seleccionar “Registrar egreso” lo cual actualizara la cantidad de los productos o materiales egresados en el sistema.

EGRESO DE MATERIALES

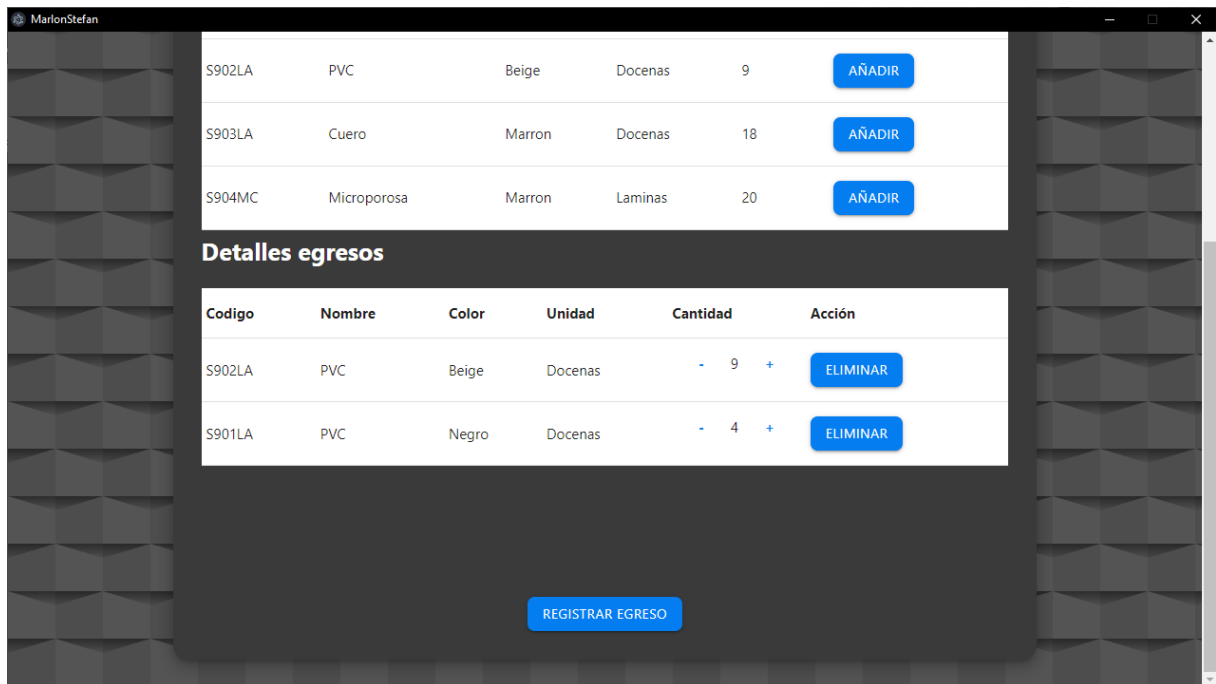
Codigo Nombre Seleccione Seleccione FILTRAR

Codigo	Nombre	Color	Unidad	Stock	Acción
S901LA	PVC	Negro	Docenas	7	AÑADIR
S902LA	PVC	Beige	Docenas	9	AÑADIR
S903LA	Cuero	Marron	Docenas	18	AÑADIR
S904MC	Microporosa	Marron	Laminas	20	AÑADIR

Detalles egresos

Codigo	Nombre	Color	Unidad	Cantidad	Acción
--------	--------	-------	--------	----------	--------

En esta interfaz se podrá registrar el egreso de materiales, añadiéndolos a los detalles de egresos mediante el botón “Añadir”, luego de añadir todos los materiales a egresar y seleccionar sus cantidades se vera de la siguiente manera.



Por último, se debe seleccionar el botón “Registrar egreso” lo cual hará la respectiva actualización de las cantidades de los materiales egresados en el sistema.

ANEXO B. ENCUESTA VIABILIDAD

A continuación se mostrarán los resultados de la encuesta realizada a la persona encargada del inventario de la fábrica Prefabricados Marlon – Stefan. Dicha encuesta tiene como propósito conocer la satisfacción del cliente sobre la aplicación web de escritorio y el plan que se tiene para poner en práctica la aplicación. Para la respuesta (*ver ilustración 108 a la 112*).

La primera pregunta realizada al cliente fue: ¿Está de acuerdo usted con afirmar que la aplicación web de escritorio cumple con los requerimientos definidos para el desarrollo del proyecto? La respuesta que se obtuvo fue un “Sí”, por lo cual el cliente confirma que la aplicación cumplió con todos los requerimientos definidos. Para la respuesta (*ver ilustración 108*).

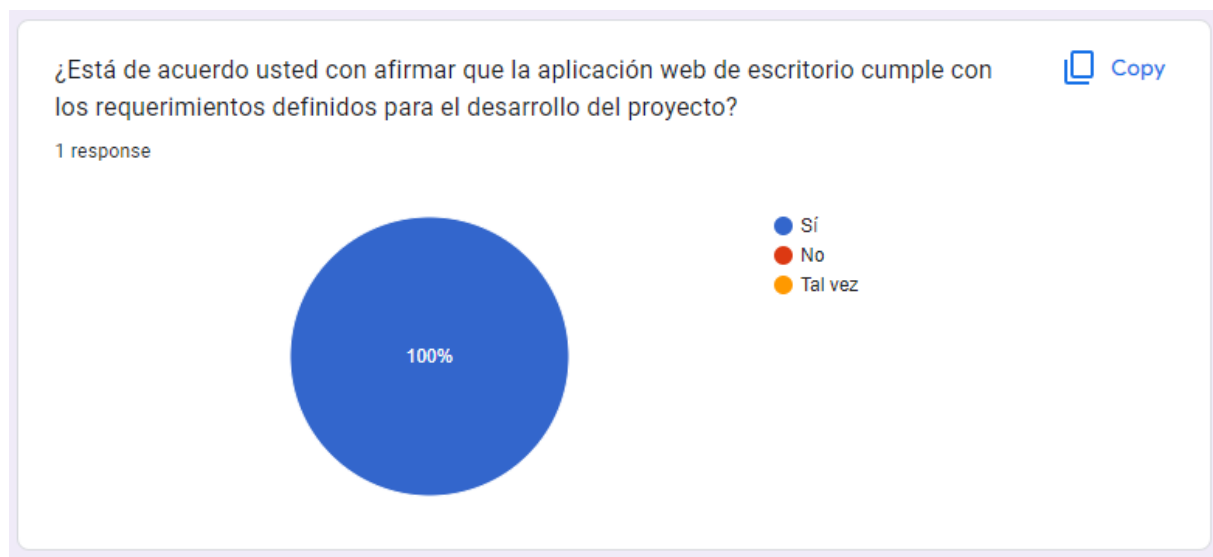


Ilustración 108. Resultado pregunta 1

La segunda pregunta realizada al cliente fue: ¿Considera usted que la aplicación web de escritorio cuenta con un diseño simple, intuitivo y fácil de manejar? La respuesta que se obtuvo fue un “Sí”, por lo cual el cliente afirma que la aplicación cumple con un diseño simple e intuitivo que le permite tener un flujo en la aplicación con facilidad. Para la respuesta (ver *ilustración 109*)

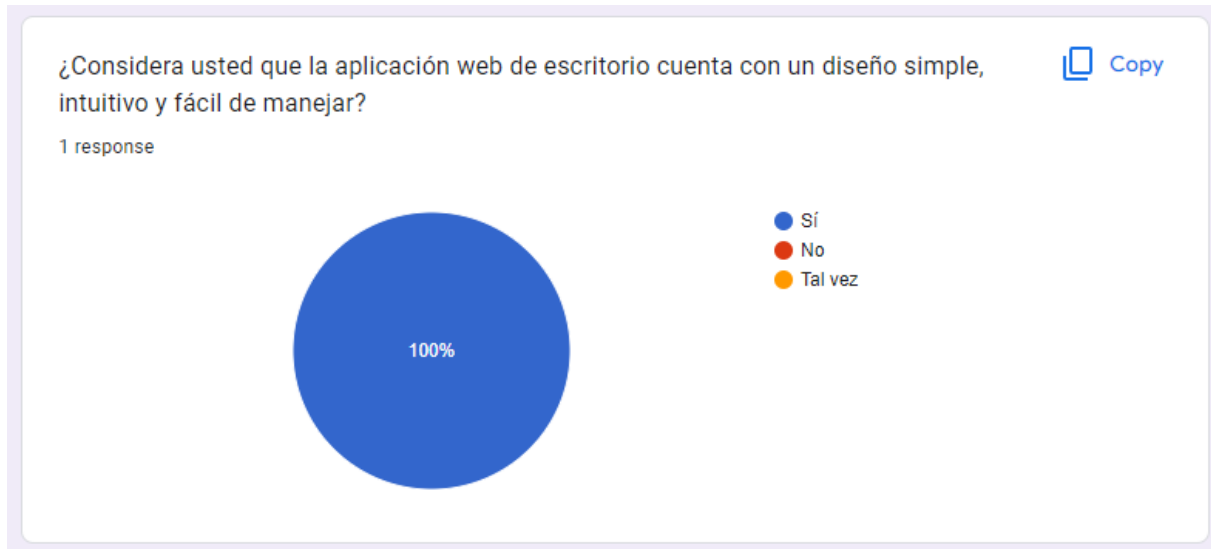


Ilustración 109. Resultado pregunta 2

La tercera pregunta realizada al cliente fue: ¿Considera usted que las funcionalidades desarrolladas en la aplicación web de escritorio cumplen con las necesidades del proyecto? La respuesta que se obtuvo fue un “Sí”, por lo cual el cliente afirma que las funcionalidades que se encuentran en la aplicación cumplen con las necesidades planteadas para el proyecto. Para la respuesta (ver *ilustración 110*)



Ilustración 110. Resultado pregunta 3

La cuarta pregunta realizada al cliente fue: ¿Considera usted que la aplicación web de escritorio facilitara el manejo del inventario de la fábrica? La respuesta que se obtuvo fue un “Tal vez”, por lo cual el cliente todavía no está seguro debido a que no se ha probado por un tiempo la aplicación, lo que se puede evidenciar en la siguiente pregunta. Para la respuesta (ver ilustración 111)

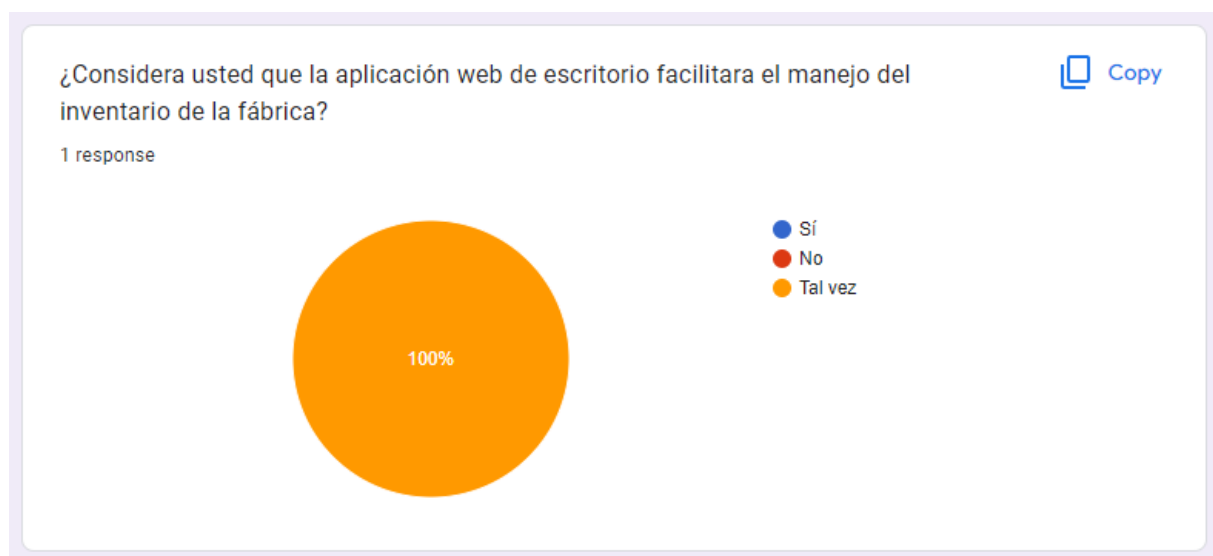


Ilustración 111. Resultado pregunta 4

La quinta pregunta realizada al cliente fue: A continuación responda, como se piensa poner en práctica la aplicación web de escritorio en la fábrica. En la respuesta que se obtuvo se puede observar que primero pondrán la aplicación en un periodo de prueba con información que ya se tiene en la fábrica para poder comprobar si les presenta una mejora en el control del inventario, dependiendo del resultado anterior se pondrá en uso real a posterior. Para la respuesta (*ver ilustración 112*)

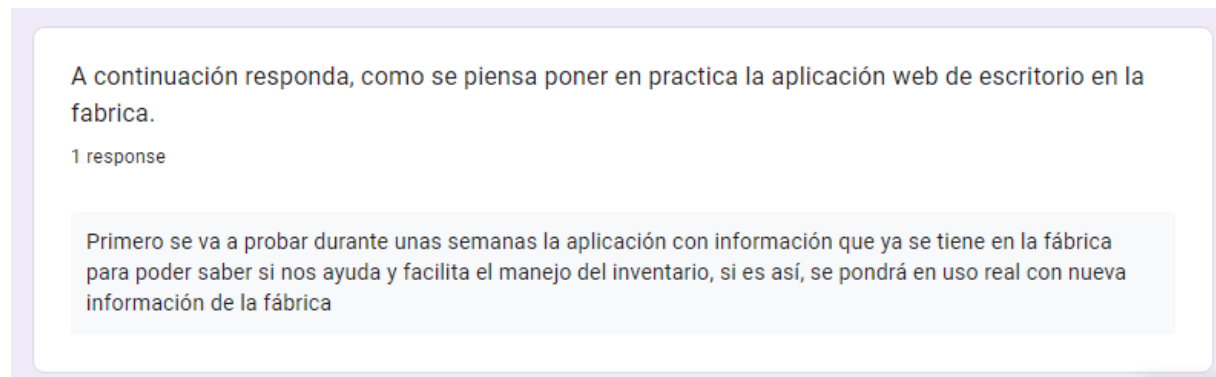


Ilustración 112. Resultado pregunta 5