Universidad de San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ingeniería en Ciencias y Sistemas



# EVALUACION DE LOS BENEFICIOS QUE PROPORCIONAN LAS HERRAMIENTAS LOW-CODE Y NO-CODE PARA EL DESARROLLO DE SOLUCIONES TECNOLOGICAS WEB EN PEQUEÑAS EMPRESAS

**Edwin Alfredo López Gómez**
**Santiago Gilberto Antonio Rivadeneira Ruano**
Asesorado por Ing. Herman Igor Véliz Linares

Guatemala, julio de 2025

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA


FACULTAD DE INGENIERÍA

# EVALUACION DE LOS BENEFICIOS QUE PROPORCIONAN LAS HERRAMIENTAS LOW-CODE Y NO-CODE PARA EL DESARROLLO DE SOLUCIONES TECNOLOGICAS WEB EN PEQUEÑAS EMPRESAS

TRABAJO DE GRADUACIÓN

PRESENTADO A LA JUNTA DIRECTIVA DE LA
FACULTAD DE INGENIERÍA
POR

**EDWIN ALFREDO LOPEZ GOMEZ**
**SANTIAGO GILBERTO ANTONIO RIVADENEIRA RUANO**
ASESORADO POR EL ING. HERMAIN IGOR VÉLIZ LINARES

AL CONFERÍRSELE EL TÍTULO DE

**INGENIERO EN CIENCIAS Y SISTEMAS**

GUATEMALA, JULIO DE 2025

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA


## NÓMINA DE JUNTA DIRECTIVA

| | |
|---|---|
| DECANO (a. i.) | Ing. José Francisco Gómez Rivera |
| VOCAL II | Ing. Mario Renato Escobedo Martínez |
| VOCAL III | Ing. Juan Carlos Molina Jiménez |
| VOCAL IV | Ing. Kevin Vladimir Cruz Lorente |
| VOCAL V | Ing. Fernando José Paz González |
| SECRETARIO | Dr. Hugo Humberto Rivera Pérez |

## TRIBUNAL QUE PRACTICÓ EL EXAMEN GENERAL PRIVADO

| | |
|---|---|
| DECANO (a. i.) | Ing. José Francisco Gómez Rivera |
| EXAMINADOR | Ing. Edgar Estuardo Santos |
| EXAMINADOR | Ing. Edgar René Ornelis Hoil |
| EXAMINADOR | Ing. Álvaro Giovanni Longo Morales |
| SECRETARIO | Dr. Hugo Humberto Rivera Pérez |

**Edwin Alfredo López Gómez**

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA


## NÓMINA DE JUNTA DIRECTIVA

| | |
|---|---|
| DECANO (a. i.) | Ing. José Francisco Gómez Rivera |
| VOCAL II | Ing. Mario Renato Escobedo Martínez |
| VOCAL III | Ing. Juan Carlos Molina Jiménez |
| VOCAL IV | Ing. Kevin Vladimir Cruz Lorente |
| VOCAL V | Ing. Fernando José Paz González |
| SECRETARIO | Dr. Hugo Humberto Rivera Pérez |

## TRIBUNAL QUE PRACTICÓ EL EXAMEN GENERAL PRIVADO

| | |
|---|---|
| DECANO | Ing. José Francisco Gómez Rivera (a. i.) |
| EXAMINADORA | Inga. Virginia Victoria Tala Ayerdi |
| EXAMINADOR | Ing. Herman Igor Véliz Linares |
| EXAMINADOR | Ing. Byron Rodolfo Zepeda Arévalo |
| SECRETARIO | Dr. Hugo Humberto Rivera Pérez |

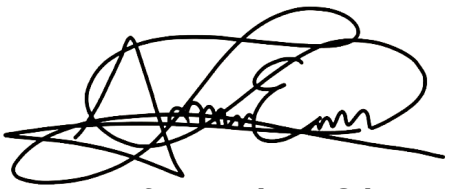
**Santiago Gilberto Antonio Rivadeneira Ruano**

# HONORABLE TRIBUNAL EXAMINADOR

En cumplimiento con los preceptos que establece la ley de la Universidad de San Carlos de Guatemala, presento a su consideración mi trabajo de graduación titulado:

**EVALUACION DE LOS BENEFICCIOS QUE PROPORCIONAN LAS HERRAMIENTAS LOW-CODE Y NO-CODE PARA EL DESARROLLO DE SOLUCIONES TECNOLOGICAS WEB EN PEQUEÑAS EMPRESAS**

Tema que me fuera asignado por la Dirección de la Escuela de Ingeniería en Ciencias y Sistemas con fecha 22 de febrero de 2023.



**Edwin Alfredo López Gómez**

**Santiago Gilberto Antonio Rivadeneira Ruano**

Guatemala, 22 de abril de 2025

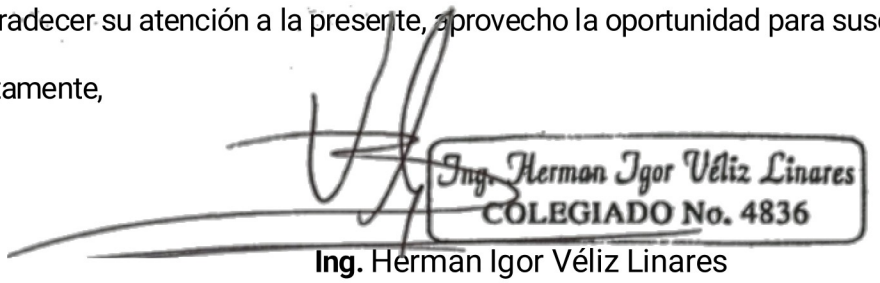
Ingeniero
**Carlos Alfredo Azurdia**
**Coordinador de Privados y Trabajos de Tesis**
**Escuela de Ingeniería en Ciencias y Sistemas**
**Facultad de Ingeniería - USAC**

Respetable Ingeniero Azurdia:

Por este medio hago de su conocimiento que en mi rol de asesor del trabajo de investigación realizado por el estudiante **Santiago Gilberto Rivadeneira Ruano** con carné **201313722** y CUI **2666 91048 0101** y el estudiante **Edwin Alfredo López Gómez** con carné **201314007 y CUI 2247 57431 0101** titulado **"EVALUACION DE LOS BENEFICIOS QUE PROPORCIONAN LAS HERRAMIENTAS LOW-CODE Y NO-CODE PARA EL DESARROLLO DE SOLUCIONES TECNOLOGICAS WEB EN PEQUEÑAS EMPRESAS",** luego de corroborar que el mismo se encuentra finalizado, lo he revisado y doy fé de que el mismo cumple con los objetivos propuestos en el respectivo protocolo, por consiguiente, procedo a la aprobación correspondiente.

Al agradecer su atención a la presente, aprovecho la oportunidad para suscribirme,

Atentamente,




**Ing.** Herman Igor Véliz Linares
Colegiado No. 4836

Universidad San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ingeniería en Ciencias y Sistemas

Guatemala, 6 de mayo del 2025

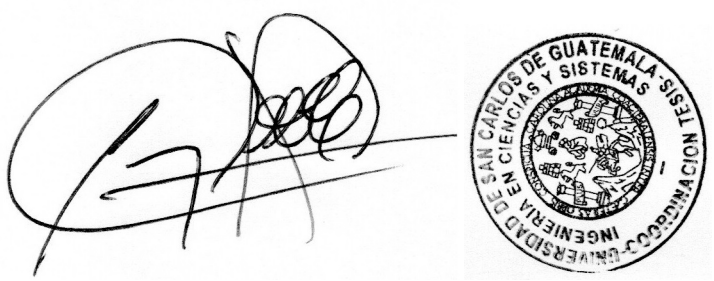
Ingeniero
**Carlos Gustavo Alonzo**
**Director de la Escuela de Ingeniería**
**En Ciencias y Sistemas**

Respetable Ingeniero Alonzo:

Por este medio hago de su conocimiento que he revisado el trabajo de graduación de los estudiantes **Santiago Gilberto Antonio Rivadeneira Ruano** con carné **201313722** y CUI **2666910480101** y **Edwin Alfredo Lopez Gomez** con carné **201314007** y CUI **2247574310101,** titulado**: "EVALUACIÓN DE LOS BENEFICIOS QUE PROPORCIONAN LAS HERRAMIENTAS LOW-CODE Y NO-CODE PARA EL DESARROLLO DE SOLUCIONES TECNOLÓGICAS WEB EN PEQUEÑAS EMPRESAS",** y a mi criterio el mismo cumple con los objetivos propuestos para su desarrollo, según el protocolo.

Al agradecer su atención a la presente, aprovecho la oportunidad para suscribirme,

Atentamente,




**Ing. Carlos Alfredo Azurdia**
Coordinador de Privados
y Revisión de Trabajos de Graduación

**UNIVERSIDAD DE SAN CARLOS DE GUATEMALA**

**FACULTAD DE INGENIERÍA**

SIST.LNG.DIRECTOR.22.EICCSS.2025

El Director de la Escuela de Ingeniería en Ciencias y Sistemas de la Facultad de Ingeniería de la Universidad de San Carlos de Guatemala, luego de conocer el dictamen del Asesor, el visto bueno del Coordinador de área del trabajo de graduación titulado: **EVALUACIÓN DE LOS BENEFICIOS QUE PROPORCIONAN LAS HERRAMIENTAS LOW-CODE Y NO-CODE PARA EL DESARROLLO DE SOLUCIONES TECNOLÓGICAS WEB EN PEQUEÑAS EMPRESAS**, presentado por: **Santiago Gilberto Antonio Rivadeneira Ruano, Edwin Alfredo Lopez Gomez**, procedo con el Aval del mismo, ya que cumple con los requisitos normados por la Facultad de Ingeniería.

"ID Y ENSEÑAD A TODOS"

Ingeniero Carlos Gustavo Alonzo
DIRECTOR
Escuela de Ingeniería en Ciencias y Sistemas

Guatemala, mayo de 2025

Decanato
Facultad e Ingeniería

24189101- 24189102

LNG.DECANATO.OIE.570.2025

El Decano de la Facultad de Ingeniería de la Universidad de San Carlos de Guatemala, luego de conocer la aprobación por parte de la Dirección de Escuela de Ingeniería en Ciencias y Sistemas, al Trabajo de Graduación titulado: **EVALUACIÓN DE LOS BENEFICIOS QUE PROPORCIONAN LAS HERRAMIENTAS LOW-CODE Y NO-CODE PARA EL DESARROLLO DE SOLUCIONES TECNOLÓGICAS WEB EN PEQUEÑAS EMPRESAS**, presentado por: **Santiago Gilberto Antonio Rivadeneira Ruano, Edwin Alfredo Lopez Gomez** después de haber culminado las revisiones previas bajo la responsabilidad de las instancias correspondientes, autoriza la impresión del mismo.

IMPRÍMASE:

Firmado electrónicamente por: José Francisco Gómez Rivera
Motivo: Autorización de entrega de trabajo de graduación
Fecha: 08/07/2025 13:23:53
Lugar: Facultad de Ingeniería, USAC.

Ing. José Francisco Gómez Rivera
Decano a.i.

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
DECANO a.i.
Facultad de Ingeniería

Guatemala, julio de 2025

# ACTO QUE DEDICO A:

| | |
|---|---|
| **Dios** | A la fuerza suprema que guía mis pasos y me brinda sabiduría en cada desafío. |
| **Mis padres** | Verónica Elizabeth Gómez Salguero y José Alfredo López Estrada, pilares inquebrantables de mi vida, cuyo amor y sacrificio han sido el cimiento de mis logros. |
| **Mis hermanos** | Compañeros de vida y cómplices en cada aventura, por su apoyo incondicional en este viaje académico. |
| **Mis tíos** | Extensión de mi familia, por su constante aliento y por creer en mí incluso cuando yo dudaba. |
| **Mi amigo** | Incansable y fuente inagotable de ideas, por ser luz en los bloqueos creativos y compañero en las largas noches de estudio. |

**Edwin Alfredo López Gómez**

## ACTO QUE DEDICO A:

| | |
|---|---|
| **Dios** | Por darme la sabiduría, la paz y templanza necesarias para alcanzar esta meta. |
| **Mis padres** | Santiago Belarmino Rivadeneira Rodas y Vilma Patricia Ruano Reynoso, por brindarme su apoyo incondicional y su constante motivación. |
| **Mis hermanos** | Edson André y Kamyla Celeste Rivadeneira Ruano, por estar a mi lado apoyándome. |
| **Mis abuelos** | Marco Antonio Ruano García (q. e. p. d) y María Francisca Reynoso, por motivarme y estar pendientes de mi carrera. |
| **Mi novia** | Khrishaa Bbettzabely Kristabely Santizo Barrera, por estar a mi lado, animarme y confiar en mí. |

**Santiago Gilberto Antonio Rivadeneira Ruano**

# AGRADECIMIENTOS A:


| | |
|---|---|
| **Universidad de San Carlos de Guatemala** | Por ser la cuna de mi formación profesional y por su inquebrantable compromiso con la excelencia académica. |
| **Facultad de Ingeniería** | Por proporcionar las herramientas y conocimientos necesarios para enfrentar los desafíos de la ingeniería moderna. |
| **Mi familia** | Por sostenerme con su amor incondicional durante cada desafío y victoria en mi camino universitario; sin sus sacrificios, paciencia y fe inquebrantable en mí, este logro no sería posible. |
| **Mis amigos** | Compañeros de batalla en esta aventura académica. Gracias por las noches de estudio compartidas, por los ánimos en los momentos de flaqueza y por las celebraciones en cada pequeña victoria. Su amistad ha sido el apoyo en los momentos difíciles y la alegría en los triunfos. Este logro no sería el mismo sin el apoyo incondicional. |

**Edwin Alfredo López Gómez**

# AGRADECIMIENTOS A:


| | |
|---|---|
| **Universidad de San Carlos de Guatemala** | Por permitirme formarme como profesional en sus aulas. Agradezco la educación de calidad y los valores que me han inculcado durante estos años de estudio. |
| **Facultad de Ingeniería** | Por proporcionarme los conocimientos y herramientas necesarias para mi desarrollo académico. |
| **Mi familia** | Por su apoyo incondicional, paciencia y animo durante toda mi carrera universitaria. |
| **Mis amigos** | Por acompañarme durante toda la carrera, compartiendo alegrías y desafíos. Gracias por las largas noches de estudio, los momentos de risa que aliviaron el estrés y el apoyo mutuo que nos dimos para alcanzar nuestras metas. |

**Santiago Gilberto Antonio Rivadeneira Ruano**

# ÍNDICE GENERAL

# ÍNDICE DE ILUSTRACIONES

## FIGURAS



## TABLAS

# LISTA DE SÍMBOLOS

| Símbolo | Significado |
|---|---|
| **$** | Dólar |
| **Gb** | Giga Bite |
| **Mb** | Mega Bite |

# GLOSARIO

**Abstracción** Es una técnica que se centra en identificar y sintetizar las partes más importantes de un proceso.

**Algoritmo** Son una serie de pasos ordenados que tienen como fin realizar una tarea.

***API*** Son un conjunto de estándares que tienen como fin crear reglas de comunicación entre sistemas.

**Arquitectura** Conjunto tecnologías que trabajan en conjunto como un único sistema para brindar servicios.

**Automatización** Es un análisis que se enfoca en diseñar procesos autónoma sin intervención humana.

**Biblioteca** Son conjuntos de programas empaquetados que trabajan en conjunto para realizar procesos estandarizados.

**Boolean** Es un tipo de dato que permite almacenar valores lógicos.

**Bug** Se denomina así a un error de programación.

| | |
|---|---|
| **CAPEX** | Se refiere a inversión por parte de una empresa en cuanto mejora de activos. |
| **Chatbot** | Es un programa informático que tiene como fin gestionar procesos de forma autónoma por medio de un chat. |
| **Checkbox** | Es un componente gráfico utilizado en informática, que guarda un valor. |
| **Cliente-Servidor** | Es un tipo de sistema informático en el cual existe una separación entre la lógica y la presentación de la información. |
| **CRUD** | Es referencia a siglas en ingles las cuales tienen como objetivo gestionar información por medio de procesos de crear, leer, actualizar y eliminar. |
| **DBMS** | Son sistemas informáticos que se encargan de gestionar la información de bases de datos. |
| **Dependencia** | Son programas los cuales son necesarios para el correcto funcionamiento de un sistema informático. |
| **Drag and Drop** | Hace referencia metodología de desarrollo en la que los componentes visuales son arrastrados y soltados en un entorno gráfico para crear una vista. |

| | |
|---|---|
| **Encapsulamiento** | Es un principio informático el cual se basa en contener la implementación por motivos de seguridad. |
| **Encriptación** | Proceso de codificación, utilizado para la protección de información sensible. |
| **Endpoint** | Punto de comunicación de red donde se interactúa con un API, que permite interactuar con servicios y recursos. |
| **Firewall** | Dispositivo que se encarga de filtrar el tráfico de red entrante y saliente, protegiendo los sistemas de intrusiones. |
| **Framework** | Son herramientas y componentes reutilizables que proporcionan una base para la implementación de aplicaciones de software. |
| **Hacker** | Individuos con habilidades técnicas avanzadas que manipulan o atacan sistemas informáticos. |
| **Herencia** | Mecanismo de la programación orientada a objetos que permite trasladar las propiedades de una clase a otra, facilitando la reutilización de código. |
| **Input** | Elemento HTML que permite la creación de controles interactivos con el fin de obtener datos del usuario. |

| | |
|---|---|
| **Interfaz** | Es el medio por el cual interactúan componentes de software, hardware o usuarios, que permite el intercambio de información. |
| **JSON** | Formato de texto que se utiliza para el intercambio de datos estructurados. |
| **JWT** | Es un estándar abierto para la transmisión de información segura, se utiliza para autenticar usuarios. |
| **Librería** | Son un conjunto de funciones y recursos que ya existen que pueden ser utilizados para tareas sin escribir código desde cero. |
| **MFA** | Método de autenticación que requiere dos o más factores para validar la identidad de un usuario. |
| **Nube** | Recursos informáticos que son utilizados a través de internet que permiten el uso de aplicaciones bajo demanda. |
| **OPEX** | Gasto operativo recurrente asociado con el funcionamiento diario de una empresa, como salarios, alquileres y servicios. |
| **Paradigma** | Son un conjunto de conceptos y prácticas que definen un enfoque para resolver problemas de programación y diseño de sistemas. |

| | |
|---|---|
| **Persistencia** | Es la capacidad de mantener y recuperar datos almacenados a largo plazo, asegurando la disponibilidad de la información. |
| **Plugin** | Componente que añade funcionalidades a una aplicación principal. |
| **Polimorfismo** | Propiedad de la programación orientada a objetos que facilita la flexibilidad y reutilización de código. |
| **Proxy** | Servidor que actúa como intermediario entre un cliente y otro servidor que mejora el rendimiento, seguridad y la privacidad. |
| **QR** | Código de barras que puede ser escaneado por dispositivo para acceder a datos o enlaces. |
| **Radio button** | Elemento HTML que permite al usuario seleccionar una opción entre varias alternativas. |
| **REST** | Tipo de arquitectura que se utiliza para la comunicación entre un cliente y servidor. |
| **Scrum** | Marco de trabajo para la gestión y desarrollo de proyectos que promueve colaboración continua y flexibilidad ante cambios. |
| **Scrum master** | Es un profesional que dirige un equipo dentro del marco de trabajo Scrum guiando al quipo para seguir |

| | |
|---|---|
| | las mejores prácticas agiles, remover impedimentos y mejorar continuamente. |
| **Software** | Conjunto de programas y aplicaciones que proporcionan funcionalidades al usuario. |
| **SQL** | Lenguaje de programación para la gestión de bases de datos relacionales. |
| **Trigger** | Instrucciones que se ejecutan automáticamente en una base de datos en respuesta a los diferentes eventos, como inserción, actualización o eliminación. |
| **Wiki** | Sitio web colaborativo cuyas páginas son editadas directamente desde el navegador, donde los usuarios son los encargados de la gestión de las páginas. |

# RESUMEN

En las últimas décadas, las pequeñas empresas han desempeñado un papel fundamental en el desarrollo económico. Diversos estudios han señalado que su crecimiento depende en gran medida de la digitalización y el uso de herramientas tecnológicas. En este contexto, la presente investigación se enfocó en analizar las necesidades más comunes en aplicaciones web para pequeñas empresas, con el propósito de identificar soluciones eficientes y accesibles.

Para abordar esta problemática, se realizó un análisis comparativo entre las tecnologías *low-code, no-code* y la programación convencional para el desarrollo de soluciones web en pequeñas empresas. A través de la creación de un sistema de inventarios, se examinaron las ventajas y limitaciones de cada una de estas tecnologías. Además, se evaluaron las principales necesidades de las pequeñas empresas para optimizar sus procesos y mantenerse competitivas en un entorno digital en constante evolución.

Este documento se encuentra estructurado en once capítulos. En los primeros seis capítulos, se presenta el marco teórico que sustenta la investigación. Entre el séptimo y decimo capitulo, se detalla la metodología empleada. Finalmente, en el décimo primer capítulo se expone el *benchmarking* junto con análisis comparativo y la interpretación de resultados.

# OBJETIVOS

- General

Analizar y comparar los beneficios que proveen las distintas herramientas l*ow-code* y *no-code*, con los beneficios que ofrece la programación convencional para el desarrollo de soluciones tecnológicas web, con el fin de obtener los mejores resultados e implementar software que se adapte a las necesidades y requerimientos de pequeñas empresas.

- Específicos

1. Indagar e identificar las necesidades comunes que se presentan en pequeñas empresas, en cuanto a soluciones tecnológicas.

2. Investigar las capacidades y características que brindan las herramientas low-code y no-code en contraste con las herramientas convencionales.

3. Analizar los beneficios de *low-code* y *no-code* en el desarrollo de implementación de soluciones web.

4. Comparar las herramientas *low-code* y *no-code*, con las herramientas de la programación convencional web para sugerir una mejor opción de implementación de software que mejor se adapte a las necesidades de una pequeña empresa.

5. Dar a conocer recomendaciones que ayuden a las pequeñas empresas a implementar soluciones tecnológicas web.

# INTRODUCCIÓN

El crecimiento en empresas y negocios ha impulsado que los grupos de trabajo busquen soluciones agiles para adaptarse a los ambientes tecnológicos, esto provoca que personas sin un alto grado de conocimientos, referentes a programación, indaguen sobre nuevas tecnologías que proporcionen soluciones que se adapten mejor a las necesidades; Es allí cuando aparece el *no-code* y *low-code*. Otorgando la posibilidad de obtener los beneficios y ventajas que proporciona la programación convencional optimizando el tiempo, reduciendo costos y velocidad de implementación.

# 1. TECNOLOGÍA EN PEQUEÑAS EMPRESAS

En la actualidad, es importante y necesario para las empresas mantener la producción, la gestión de recursos y las gestiones financieras, tratando de mantenerse a la vanguardia en cuanto a la innovación y el desarrollo tecnológico. Esto no es un problema para las medianas y grandes empresas, pero representa un desafío para las pequeñas. Las pequeñas empresas, en la mayoría de las ocasiones, no cuentan con los recursos necesarios para poder iniciar proyectos. En estos escenarios, se ven condicionadas por una serie de elementos, tales como la estructura, el entorno, el sector y los recursos financieros.

## 1.1. Transformación digital

Las reglas del negocio para las empresas han cambiado. Si una empresa quiere prosperar, necesita iniciar un proceso de transformación, es decir, un proceso de transformación digital. Este no es un proceso en el cual se implementan las últimas tecnologías, sino un proceso complicado que no solamente necesita cambios estructurales, sino también un cambio de mentalidad en la empresa.

Cuando se habla de transformación digital se tiene que abordar desde tres puntos de vista diferentes: tecnología, procesos y personas. El último de estos es donde se encuentra la esencia de éxito, manteniendo una buena gestión del cambio con empleados, proveedores y clientes.

### 1.1.1. ¿Qué es la transformación digital?

La transformación digital es un proceso mediante el cual una organización busca incorporar la tecnología digital en todas las áreas del negocio con el objetivo de cambiar la forma en que se otorga valor a sus clientes. Las compañías implementan tecnologías digitales innovadoras para llevar a cabo cambios en su cultura y operaciones, de modo que puedan ajustarse a las cambiantes necesidades de sus clientes.

### 1.1.2. Importancia de la transformación digital

La importancia de la transformación digital es sumamente notable, ya que permite a las organizaciones optimizar procesos, utilizar mejor los recursos y tener una mejor perspectiva hacia el futuro, obteniendo así las mejores ventajas competitivas del mercado. Esto es posible debido a la simplificación de procesos, al uso adecuado de tecnologías apropiadas y al análisis de los datos generados, con el fin de tomar mejores decisiones.

### 1.1.3. Ventajas de la transformación digital

Cuando se implementa la transformación digital dentro de una organización, se obtienen diversas ventajas, tales como:

- Mejora en la productividad, reducción de tiempos y aumento en la eficiencia de todo tipo de procesos empresariales.

- Mejorar la experiencia del cliente al ofrecer disponibilidad de servicios a través de múltiples canales, como sitios web, aplicaciones móviles,

comunicación en tiempo real y seguimiento de pedidos, lo que se traduce en un mejor servicio al cliente en general.

- Reducir los costos operativos, incluyendo el mantenimiento de equipos, la logística de entrega, los gastos energéticos, los gastos en recursos humanos y los gastos de asistencia al cliente.

### 1.1.4. Pilares de la transformación digital

Para que la transformación digital sea eficaz, no es necesario aplicar las tecnologías más nuevas. En su lugar, es fundamental actualizar todos los aspectos de la organización con el fin de aprovechar al máximo los beneficios que la digitalización empresarial puede ofrecer.

#### 1.1.4.1. Experiencia del cliente

Este es uno de los pilares clave que impulsan la transformación digital, ya que se busca adaptar la tecnología solo si se conoce el comportamiento y las expectativas del cliente.

#### 1.1.4.2. Personas

Los nuevos modelos de negocios digitales tienen éxito si sus empleados se adaptan plenamente. Esto se puede lograr mediante la capacitación de los empleados, la atracción de nuevo personal y la retención del personal existente mediante oportunidades que ofrezcan crecimiento profesional.

#### 1.1.4.3. Cambio

Es necesario prepararse para el cambio, ya que la transformación digital altera todas y cada una de las partes de la empresa. Por lo tanto, se requiere una planificación adecuada para obtener todas las herramientas y el entorno de trabajo necesarios para lograr una transformación exitosa.

#### 1.1.4.4. Innovación

La innovación consiste en crear nuevas ideas que fomenten los cambios y transformaciones, por lo que se relaciona directamente con la transformación digital. Cuando se dice que es uno de los pilares de la transformación digital, es porque se busca que los empleados se involucren directamente en todo el proceso de transformación digital, cambiando su forma de pensar y aceptando dicho cambio.

#### 1.1.4.5. Liderazgo

El liderazgo es uno de los pilares más importante dentro de la transformación digital, puesto que se necesitan lideres proactivos que mantengan el control de todo el proceso de transformación.

#### 1.1.4.6. Cultura

El último de los pilares, aunque no menos importante, depende de los pilares anteriores, ya que se busca crear una cultura de innovación dentro de los empleados, convirtiéndolos en apasionados y deseosos de proporcionar valor a los clientes con el máximo rendimiento que sus habilidades puedan

ofrecer. Llegado a este punto, donde los empleados aceptan el cambio, las iniciativas por la transformación digital se amplían y tienen éxito.

### 1.1.5. ¿Cuáles son los tipos de transformación digital?

Existen cuatro tipos principales por los que las organizaciones pueden optar para la transformación digital: procesos empresariales, modelos de negocios, dominio empresarial y organización o cultura.

#### 1.1.5.1. Proceso empresarial

La optimización de procesos tiene como objetivo encontrar métodos innovadores para mejorar los flujos de trabajo internos y externos ya establecidos. La introducción de nuevas tecnologías, con frecuencia, transforma significativamente los procesos, lo que contribuye a obtener mejores resultados.

#### 1.1.5.2. Modelo de negocio

Este tipo de transformación se enfoca en rediseñar un modelo de negocio ya existente con tecnología. El principal propósito es brindar servicios empresariales fundamentales de forma innovadora o a través de canales diferentes para aumentar los ingresos como el incremento de clientes.

#### 1.1.5.3. Dominio empresarial

La transformación digital de dominios ocurre cuando una organización logra capturar exitosamente una nueva parte o dominio del mercado. Las empresas logran esto con proyectos de transformación digital que mejoran sus

ofertas o crean nuevas oportunidades de negocio a través de tecnologías digitales.

#### 1.1.5.4. Organización o cultura

Es una renovación de la cultura u organización interna dentro de la empresa, centrándose en brindar el máximo valor a los clientes. Este enfoque se aplica generalmente cuando se busca adelantarse a la competencia y alcanzar los objetivos empresariales de forma rápida.

# 2. PROGRAMACIÓN POR CAPAS

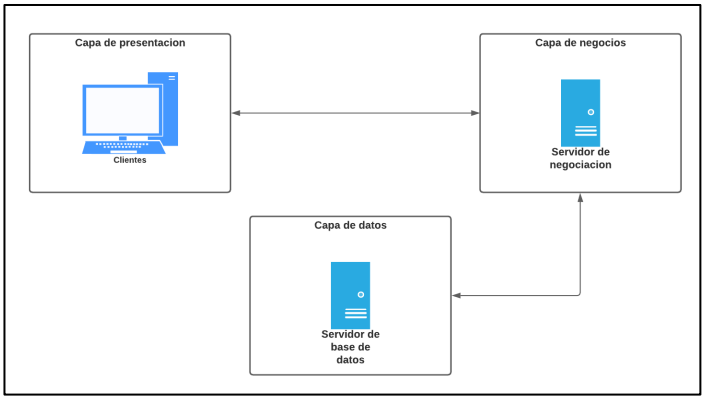
El desarrollo de sistemas informáticos ha evolucionado a lo largo del tiempo, empleando diversas técnicas de programación. En un principio se utilizaba la programación estructurada, la cual posteriormente se combinó con la programación por eventos para dar lugar a la programación orientada a objetos. Actualmente, se ha alcanzado un alto grado de madurez en este tipo de programación, lo que ha permitido la reutilización de código y el surgimiento de la programación por capas.

## 2.1. ¿Qué es la programación por capas?

La programación por capas consiste en dividir el código en diferentes componentes según su rol, funcionalidad y responsabilidad. Esto permite una mayor reutilización del código, un desarrollo más rápido y un mantenimiento sencillo del mismo.

**Figura 1.**

*Modelo tres capas*




*Nota*. Modelo por capas que muestra el flujo de datos. Elaboración propia, realizado con Lucidchart.

## 2.2. ¿Cuáles son las ventajas de la programación por capas?

Las ventajas que ofrece la programación por capas, en comparación con otros tipos de arquitecturas, son las siguientes:

- Una implementación más rápida: el mismo modulo puede ser desarrollado por diferentes equipos al mismo tiempo, lo que permite que la aplicación de la empresa sea lanzada al mercado más rápidamente.

- Escalabilidad: cada uno de los niveles puede ser escalado de forma independiente.

- Alta disponibilidad: es menos probable que el mal rendimiento de un nivel afecte el resto.

- Mejora de la seguridad: puesto que los niveles de presentación y de datos no se comunican entre sí, la programación por capas actúa como un firewall interno, lo que impide todo tipo de ataques.

## 2.3. ¿Cuáles son las capas principales?

Las capas principales de la programación por capas pueden variar según el contexto y la tecnología utilizada, pero en general se incluyen las siguientes.

### 2.3.1. Presentación

La capa de presentación es responsable de la interfaz visual de la aplicación. Su función principal es permitir la interacción del usuario con el sistema, comunicándose con la capa de negocio para obtener, editar y almacenar datos.

### 2.3.2. Lógica del negocio

La capa de negocio es responsable de interactuar con la capa de presentación, recolectar los datos necesarios, procesarlos y transformar esta información con la capa de datos para realizar operaciones.

### 2.3.3. Acceso a datos

La capa de datos es responsable del acceso a los datos y se utiliza para almacenar y recuperar toda la información procesada dentro del sistema. Es una parte esencial de cualquier sistema, ya que garantiza que los datos se almacenen de manera segura y estén disponibles cuando se soliciten.

# 3. TECNOLOGÍAS WEB

En los últimos años, la informática ha adquirido un papel fundamental en las empresas, y la presencia en la web ha transformado el comercio digital de manera significativa. Las tecnologías web impulsan la forma en que se diseñan, producen y venden los productos y servicios, así como también la manera en que se diseñan los procesos. Actualmente, comprar en línea es un proceso simple, rápido y seguro. Además, las empresas pueden beneficiarse de una mayor visibilidad de marca y un mayor alcance al público objetivo.

## 3.1. ¿Qué son las tecnologías web?

Son un conjunto de herramientas que se encargan de brindar un alto nivel de funcionalidad a los desarrollos web. Estas herramientas trabajan mediante la implementación de distintos tipos de arquitecturas y se apoyan en lenguajes bien establecidos para su diseño y desarrollo. Normalmente, se utilizan arquitecturas de tres capas para este tipo de desarrollos y aportan beneficios a corto y mediano plazo.

## 3.2. Capas del desarrollo web

Las aplicaciones cliente-servidor son una de las mejores alternativas para desarrollos en pequeñas empresas. Estas aplicaciones se forman a partir de arquitecturas de tres capas: la capa de presentación o *frontend*, la capa de lógica de negocio o *backend* y la capa de datos. Estas capas proporcionan una mayor rapidez en el desarrollo, escalabilidad y seguridad a las aplicaciones

cliente-servidor, convirtiéndolas en una de las mejores alternativas para pequeñas empresas.

### 3.2.1. Frontend

Con la evolución de la tecnología, las páginas web han sufrido diversos cambios a lo largo del tiempo. La necesidad de crear sitios web cada vez con más funcionalidades ha hecho necesario cambiar la forma en la que se crean y diseñan estas páginas. A partir de esto, se adopta la terminología de *frontend*. Esta filosofía se centra en diversificar y agilizar la forma en que se desarrollan las páginas web y en convertir, cada vez más, las páginas web estáticas en aplicaciones web; para las cuales el único límite, en cuanto a funcionalidad, diseño y características se refiere, son los límites que indican los requerimientos de la solución que se está desarrollando.

#### 3.2.1.1. ¿Qué es el *frontend*?

Es la parte de la capa de presentación que se ejecuta en el navegador y se encarga de mostrar a los usuarios la información de la capa del negocio y la forma de interactuar con ella. Utiliza *frameworks* y librerías para agilizar el desarrollo e implementación, y se apoya en HTML para realizar la maquetación y diseño, en CSS para brindar estilos agradables e intuitivos, y en JavaScript para agregar interactividad y funcionalidad.

#### 3.2.1.2. Campos de uso

Los desarrollos *frontend* se encuentran en una amplia gama de aplicaciones, entre las que se destacan el comercio electrónico, las redes sociales, las aplicaciones bancarias, de servicios, juegos, entre otro.

Normalmente se utilizan en soluciones con funcionalidades complejas, en las que se requiere una alta interactividad por parte del usuario**.**

#### 3.2.1.3. Tecnologías

Las aplicaciones web se basan en tres grandes pilares: HTML, CSS y Javascript los cuales en conjunto conforman las páginas web modernas.

- HTML es el lenguaje de marcado de hipertexto utilizado por el navegador para maquetar y presentar el contenido dentro del navegador. Utiliza etiquetas que delimitan los objetos que se presentan en pantalla, y su sintaxis se basa en una estructura bien delimitada que separa en bloques, las partes que componen los desarrollos web.

- CSS, o lenguaje de hojas de estilo en cascada, es un lenguaje de estilos estandarizado cuyo principal objetivo es presentar los componentes HTML y describir dónde y cómo deben ser renderizados cada uno de los componentes dentro del navegador.

- JavaScript es un lenguaje de programación interpretado multiparadigma, también conocido como lenguaje de scripting, utilizado por los navegadores modernos para proporcionar funcionalidades dinámicas complejas a las páginas web.

#### 3.2.1.4. Principales herramientas

A continuación, se describirán algunas de las herramientas más utilizadas en el diseño y desarrollo de aplicaciones web, diferenciándolas en dos grandes filosofías: los *frameworks* y las bibliotecas o librerías.

- React es una de las bibliotecas o librerías más famosas de los últimos años. Fue creada por Facebook en 2013 y se ha convertido en una de las herramientas más populares para desarrollar aplicaciones de página única (SPA). React se originó con la filosofía de unificar los componentes básicos de una aplicación web y es utilizada en aplicaciones web populares como Facebook, Instagram, Reddit y Netflix. Esta biblioteca está diseñada para permitir el desarrollo de aplicaciones complejas en un corto periodo de tiempo.

- Angular es un *framework* mantenido por Google y diseñado para crear aplicaciones de página única (SPA). Está escrito en TypeScript y se enfoca en la creación de módulos que facilitan el desarrollo de aplicaciones web. Angular permite crear soluciones robustas y ligeras que se adaptan a las necesidades y requerimientos. Este *framework* es utilizado por empresas como Google, HBO y Nike, entre otras, y proporciona un entorno de trabajo completo para el desarrollo de aplicaciones web y móviles.

### 3.2.2. Backend

Además de la presentación de la información, algo que es tanto o más importante es el manejo de la información, la capa de la lógica del negocio se encarga del manejo, integridad y la seguridad de los datos, estos procesos son totalmente transparentes para los usuarios finales y facilitan a la capa de presentación de los procesos que son sensibles para el negocio.

#### 3.2.2.1. ¿Qué es el *backend*?

Son las aplicaciones de la capa de la lógica del negocio que se encargan de la interacción entre la capa de presentación o *frontend* y las bases de datos. Existe una amplia gama de alternativas, en cuanto a herramientas de desarrollo se refiere, para diseñar e implementar estas aplicaciones. Son las responsables de mantener la integridad de la información, así como también de la seguridad, usualmente estas aplicaciones trabajan del lado del servidor.

#### 3.2.2.2. Campos de uso

El principal uso del *backend*, es gestionar la comunicación entre los diferentes componentes que componen el desarrollo web. Se encarga de la autenticación, gestión y administración de los datos, almacena la información en las bases de datos, provee de seguridad, permite la integración con otros servicios y se encarga de comunicar de forma eficiente, segura y confiable.

#### 3.2.2.3. Tecnologías

Existen una gran variedad de tecnologías en las que es posible desarrollar aplicaciones para el *backend*, desde lenguajes de programación especialmente diseñados hasta lenguajes robustos con décadas de desarrollo, los cuales son confiables. En general, para desarrollar soluciones para *backend*, es posible utilizar un lenguaje que se adapte a las necesidades del proyecto. Entre los más utilizados, tenemos JavaScript, Python, Ruby, Java, C#, C++ y otros.

#### 3.2.2.4. Principales herramientas

A continuación, se presentarán algunos de los lenguajes de programación y herramientas más utilizados para el desarrollo de aplicaciones de *backend*.

- Node.js es un entorno de desarrollo para aplicaciones web del lado del servidor, multiplataforma, orientado principalmente al desarrollo de servidores. Esta herramienta utiliza JavaScript y se ejecuta principalmente sobre un único núcleo. Además, se enfoca en el desarrollo de eventos asíncronos.

- *Flask* es un entorno de desarrollo para aplicaciones web que trabaja con el lenguaje de programación Python. Proporciona herramientas de desarrollo y es ligero y escalable. Además, tiene una curva de aprendizaje simple, lo que permite que los desarrolladores se adapten a la herramienta y al lenguaje en poco tiempo.

- Spring *Boot* es un *framework* para desarrollar aplicaciones web que admite múltiples lenguajes de programación, como Java, *Kotlin* y *Groovy*. Provee herramientas que permiten un desarrollo inicial mínimo y mantiene una filosofía de trabajo simple.

### 3.2.3. Bases de datos

La información en la actualidad desempeña un papel muy importante, desde su uso personal, donde las personas almacenan los contactos de su teléfono móvil, hasta en las grandes empresas, donde la información es

relevante para realizar análisis sobre las ventas, compras y predecir el mercado en general.

#### 3.2.3.1. ¿Qué es una base de datos?

Las bases de datos son colecciones de datos estructurados que se utilizan para almacenar información. Son ampliamente utilizadas en las empresas para gestionar de forma organizada, rápida y fiable los datos. La información se representa mediante filas y columnas, donde cada columna representa una propiedad. Las bases de datos son administradas principalmente por programas llamados Sistemas de Gestión de Bases de Datos (DBMS, por sus siglas en inglés), los cuales se encargan de gestionar la información de manera eficiente.

#### 3.2.3.2. Tipos de bases de datos

Hoy en día existen diferentes tipos de bases de datos, y las principales diferencias entre ellos tienen que ver con la forma en que se almacena la información. Esto afecta desde el archivo lógico donde se aloja la información, hasta la estructura interna que determina la accesibilidad de los datos, a continuación, se describirán algunas de las más importantes.

- Bases de datos relacionales: Basado en el modelo relacional este tipo de base de datos tiene como enfoque principal representar los datos como tablas en los que cada fila es un registro y las columnas representan los campos. Los datos se relacionan por campos que se denominan llaves primarias y permite asociar la información de forma fácil y rápida.

- Bases de datos NoSQL: las bases de datos no relacionales organizan los datos en estructuras flexibles, brindan una respuesta más rápida de la información y proporcionan una mejor escalabilidad. En la actualidad existe una amplia variedad de tipos de bases de datos NoSQL, las más comunes son:

    - Clave-valor: utiliza un sistema de almacenamiento en el que se define una clave y un valor para obtener la información.

    - Documentales: trabajan con un modelo en el que se representan los datos como un objeto, regularmente en formato JSON, lo que es fácil e intuitivo para recuperar la información.

#### 3.2.3.3. Usos de las bases de datos

Las bases de datos en la actualidad tienen una amplia gama de usos. Estos abarcan desde los usos empresariales en los que las grandes, medianas y pequeñas empresas almacenan, gestionan y analizan sus datos, hasta los usos particulares en los que un teléfono móvil almacena información del usuario. A continuación, se presentarán algunos escenarios importantes para las bases de datos.

- Gestión de inventarios: llevar un registro detallado de los bienes tangibles e intangibles es y será uno de los usos más recurrentes para el sector empresarial. El potencial de los inventarios en una base de datos está en poder acceder a la información de forma precisa y rápida para tomar decisiones.

- Comercio electrónico: es importante llevar un control detallado de las compras y ventas, las empresas usan esos datos para manejar las compras gestionar la cadena de suministros y analizar las ventas para optimizar el rendimiento financiero.

- Análisis de datos: una parte muy importante de almacenar información para una empresa es poder analizar los datos y tomar decisiones en consecuencia. Las bases de datos proporcionan esas capacidades, poder analizar los datos de las ventas para posteriormente actuar en consecuencia en la compra de productos puede ser un ejemplo claro de esto, pero el análisis de los datos va aún más allá, ya que permite hacer predicciones sobre el comportamiento del mercado.

- Videojuegos: el campo de las bases de datos también abarca una multitud de usos, y los videojuegos no son una excepción. En general, los juegos necesitan almacenar información de todo tipo para su funcionamiento interno, como los modelos en 3D, por ejemplo, pero su campo de uso es muy amplio. Las empresas productoras de videojuegos también hacen uso de los registros para analizar estadísticas y tomar decisiones en relación con el diseño y la producción de los juegos.

#### 3.2.3.4. Principales bases de datos

Existen muchas bases de datos diferentes que se utilizan ampliamente en la actualidad. Algunas de las bases de datos más populares y ampliamente utilizadas son las siguientes:

- Oracle: es el líder indiscutible en el mercado de bases de datos, proporciona soluciones robustas y confiables. Sus bases de datos relacionales son reconocidas por manejar altos volúmenes de datos y tener alta disponibilidad. También ofrece otros tipos de bases de datos, como NoSQL.

- MongoDB: es una base de datos NoSQL principalmente orientada a documentos, la estructura de los documentos muestra una similitud con los objetos JSON lo que permite que sea intuitiva y fácil de implementar. MongoDB está diseñada para funcionar en un clúster de servidores, lo que la convierte en una base de datos distribuida con alta disponibilidad.

- REDIS: redis es una base de datos NoSQL clave-valor en memoria, diseñada principalmente para almacenar información tipo caché. Ofrece velocidades de lectura y escritura extremadamente rápidas, lo que la hace ideal para manejar grandes volúmenes de datos que pueden producir cuellos de botella y necesitan ser accedidos frecuentemente con poca latencia. Con su arquitectura en memoria, Redis proporciona una alta disponibilidad y escalabilidad, y es capaz de manejar grandes cargas de trabajo en tiempo real.

# *4. NO-CODE*

El *no-code* es un término que se utiliza para referirse a la creación de aplicaciones y herramientas digitales sin necesidad de escribir código. En lugar de utilizar programación, se emplean plataformas de desarrollo visual y herramientas de arrastrar y soltar para construir y personalizar aplicaciones o sitios web.

Se le conoce como un movimiento o una filosofía que se ha vuelto popular en los últimos años debido a que permite que personas con habilidades limitadas de desarrollo puedan crear aplicaciones y automatizaciones de manera rápida y eficiente. Estas plataformas están diseñadas para ser accesibles y fáciles de usar.

## 4.1. ¿Cómo funciona el *no-code*?

El funcionamiento del *no-code* varía según la plataforma o herramienta que se esté utilizando. Sin embargo, en general, el proceso para construir una aplicación con *no-code* es agregar componentes o elementos y arrastrarlos. El *no-code* funciona como un intermediario visual entre la programación y el usuario que realiza el diseño de la aplicación.

## 4.2. ¿Cuáles son las ventajas del *no-code*?

Entre los beneficios que existen al crear aplicaciones o sitios web *no-code,* se encuentran:

- Accesibilidad: permite que personas con habilidades limitadas de programación puedan construir aplicaciones y herramientas sin necesidad de conocimientos en programación.

- Flexibilidad: las plataformas *no-code* ofrecen plantillas y herramientas personalizables, y permiten adaptar dichas herramientas a las necesidades de cada proyecto.

- Agilidad: las empresas, emprendedores y usuarios han apostado por utilizar *no-code*, ya que permite crear aplicaciones web y móviles de forma más rápida. Esto se debe a que se utilizan módulos y elementos visuales ya existentes, lo que reduce el tiempo de estas tareas y aumenta la agilidad al crear cualquier sistema.

- Bajos costos: un proyecto creado sin programar resulta más rentable que uno que involucra programadores, *scrum masters*, líderes de equipo, software, licencias, bases de datos, entre otros gastos. Todos estos costos se reducen con el uso de soluciones *no-code.*

- Modificaciones simples: en un software o aplicación creada con programación convencional, los procesos de cambios o mejoras pueden ser complicados, consumiendo tiempo y recursos. Sin embargo, el uso de herramientas *no-code* simplifica estos procesos, ya que su interfaz *drag and drop* facilita las modificaciones y mejoras.

- Tiempo y recursos: las tecnologías *no-code* posibilita la creación de aplicaciones y herramientas con mayor celeridad y eficiencia, y disminuye tanto el tiempo como los recursos necesarios para su desarrollo.

## 4.3. ¿Cuáles son las desventajas del *no-code*?

Los beneficios que ofrecen las tecnologías *no-code* son diversos, sin embargo, aunque sean tecnologías innovadoras, existen algunas desventajas que se deben tomar en cuenta:

- Personalización: aunque las tecnologías *no-code* permitan implementar aplicaciones de manera rápida y eficiente, no siempre es posible personalizar la aplicación acorde a las necesidades del usuario, ya que se cuentan con plantillas o componentes ya preestablecidos.

- Límites: el uso de tecnologías *no-code* permite la implementación de diversas soluciones a través de sus herramientas. Sin embargo, cuando se trata de sistemas masivos con procesos complicados, puede resultar difícil encontrar una solución adecuada con estas herramientas.

- Dependencias del proveedor: cuando se habla de aplicaciones creadas con tecnologías *no-code*, dependen de proveedores específicos y de las funcionalidades que este ofrece, lo que limita las opciones del usuario y aumenta el riesgo que el proveedor deje de ofrecer dicho servicio.

- Escalabilidad: las aplicaciones *no-code* pueden ser menos escalables que soluciones personalizadas, lo que limita la gestión de estas cuando se habla de un gran volumen de datos.

- Seguridad: la seguridad dentro de una solución web o aplicación es muy importante. Debido a la importancia de esto, dentro de la programación convencional existen diferentes procesos y formas de ejecutarla

correctamente. Sin embargo, en las soluciones *no-code* pueden existir vulnerabilidades, ya que estas son aplicaciones simplificadas y existen técnicas que no pueden ser aplicadas para crear una aplicación completamente segura.

### 4.4. Usos del *no-code*

Los usos son diversos, desde la creación de un sitio web, aplicaciones móviles hasta la automatización de procesos y la gestión empresarial. Entre los propósitos más destacables se incluyen:

- Creación de sitios web: las plataformas de *no-code* pueden utilizarse para crear sitios web con diversas funcionalidades, desde sitios web pequeños y simples hasta sitios más complejos con integraciones de aplicaciones de terceros.

- Creación de aplicaciones móviles: las herramientas *no-code* permiten la creación de aplicaciones móviles iOS y Android sin la implementación de código.

- Automatización de procesos: existen herramientas como *Zapier* y IFTT que permiten la automatización de procesos de una forma rápida y simple, mejorando la eficiencia y productividad de los procesos.

- Creación de *chatbots*: la transformación digital ha tenido un gran impacto en los procesos empresariales; ha llevado a la automatización de procesos y la implementación de tecnología en el servicio al cliente, entre otros aspectos. La innovación ha llegado al punto de ofrecer

soluciones como un *chatbot*, con el que los usuarios pueden interactuar para obtener información sobre una empresa o un servicio en línea.

- Análisis de datos: existen herramientas que permiten el análisis de datos a gran escala, como *Tableau*, que permite a los usuarios crear visualizaciones y análisis de información sin necesidad de escribir código.

## 4.5. Principales herramientas del *no-code*

Las tecnologías *no-code* ofrecen numerosas ventajas y diversas formas de implementar soluciones. Sin embargo, independientemente del proyecto que se desee desarrollar, existen diversas herramientas en el mercado que se adaptan a cada necesidad, como las siguientes:

- *Webflow*: es una plataforma que permite crear sitios web y aplicaciones web sin conocimientos de programación. Ofrece al usuario una interfaz para diseñar, desarrollar, lanzar sitios y aplicaciones web responsive.

- *Adalo*: es una herramienta que permite el desarrollo de aplicaciones móviles y web, así mismo su publicación en App Store o Google Play Store.

- *Glide*: es una plataforma para desarrollar aplicaciones móviles y web de forma sencilla y rápida. Aunque no permite la publicación directa en las tiendas de aplicaciones móviles, ofrece otras opciones para compartir y distribuir la aplicación a través de enlaces web o códigos QR.

- *Bubble*: es una herramienta para la creación de aplicaciones móviles y web, con una curva de aprendizaje alta y con un potencial alto para el desarrollo de soluciones móviles.

- *Airtable*: es una plataforma *no-code* cuya principal función es la creación de bases de datos relacionales.

- *Notion*: es una herramienta *no-code* que pone a disposición de los usuarios distintas formas de implementar soluciones. Algunas de estas soluciones son la creación de documentos, bases de datos y wikis de una forma colaborativa.

- *Zapier*: es una plataforma de automatización que permite al usuario realizar integraciones entre herramientas y automatizaciones de tareas repetitivas, y funciona mediante la creación de flujos de trabajo.

- *Make*: es una herramienta de automatización que permite a los usuarios realizar conexiones o automatizaciones de acciones específicas dentro de una aplicación.

- *Landbot*: es una plataforma *no-code* que permite la creación de *chatbots* y aplicaciones de mensajería con características personalizables, como la integración con servicios de terceros, la automatización de respuestas y la creación de flujos de conversación.

# *5. LOW-CODE*

El *low-code* (también conocido como *low-code development* o *low-code platform*) es un enfoque que permite a los equipos empresariales o de desarrollo crear soluciones tecnológicas y aplicaciones con una cantidad mínima de programación. Las plataformas de *low-code* proporcionan a los usuarios un conjunto de herramientas, componentes de código personalizado y scripts reutilizables, lo que les permite desarrollar procesos y aplicaciones de una forma eficiente sin tener que escribir código o realizar pruebas de scripts.

Las plataformas *low-code* proporcionan un entorno de desarrollo visual que utiliza funciones simples de arrastrar y soltar, permitiendo que cualquier persona dentro de un grupo u organización puedan diseñar y crear aplicaciones empresariales sin la necesidad de disponer de habilidades especializadas en programación.

## 5.1. ¿Cómo funciona el *low-code*?

El funcionamiento del *low-code* se basa en el uso de herramientas y plataformas que permiten a programadores y usuarios implementar aplicaciones de toda clase de forma rápida y eficiente con una cantidad mínima de programación.

En general el desarrollo de una aplicación con tecnologías *low-code* se inicia con la definición de los requisitos que se desean cumplir. A partir de ahí, los usuarios pueden seleccionar los componentes y módulos que proporcionan

las plataformas. Estos módulos y componentes pueden ser bases de datos, herramientas de automatización y conectores de integración de sistemas.

Una vez concluida la selección de componentes, los usuarios interactúan con una interfaz gráfica para arrastrar y soltar los componentes dentro de un lienzo y diseñar la interfaz de usuario de la aplicación. Durante el proceso de desarrollo, estas plataformas proporcionan asistencias al usuario, como detección y corrección de errores, así como la realización de pruebas dentro de un entorno seguro.

## 5.2. ¿Cuáles son las ventajas del *low-code*?

Las plataformas *low-code* proporcionan una gran cantidad de beneficios para las organizaciones, equipos y desarrolladores. Dentro de estos beneficios podemos encontrar:

- Innovación: el *low-code* permite que la innovación dentro de las empresas aumente, ya que cualquier empleado puede tomar la iniciativa para desarrollar aplicaciones.

- Agilidad: en la actualidad, las empresas tienen que poder adaptarse a los cambios del mercado y satisfacer las necesidades de los consumidores. El desarrollo con tecnologías *low-code* permite que las empresas se vuelvan más agiles al utilizar componentes prefabricados y herramientas de automatización se crean las aplicaciones rápidamente, lo que acelera los procesos de comercialización.

- Ahorrar costos: el uso de herramientas *low-code* permite reducir significativamente los costos de desarrollo de software, ya que se

reducen considerablemente el tiempo y los recursos para crear código personalizado.

- Seguridad: el *low-code* permite agregar características de seguridad para la protección de las aplicaciones y datos.

- Mantenimiento: el desarrollo de aplicaciones *low-code* es más simple en comparación de la programación convencional, ya que este requiere una cantidad significativa de líneas de código. Esto implica que los equipos empresariales o usuarios pueden optar por realizar su propio mantenimiento y actualización de sus productos, generando procesos de desarrollo más eficientes.

- Eficiencia: al utilizar herramientas *low-code* se reduce el tiempo de codificación por lo que la eficiencia dentro de los procesos de desarrollo aumenta.

- Facilidad de uso: el *low-code* permite que los usuarios puedan crear aplicaciones sin tener habilidades especificas en programación o que tengan que recurrir a cursos, tutoriales tediosos de conceptos que para un usuario común pueden tornarse complicados. Esto significa que una gran cantidad de usuarios pueden crear aplicaciones o soluciones tecnológicas, únicamente conociendo los requisitos del usuario que se quieren aplicar.

### 5.3. ¿Cuáles son las desventajas del *low-code*?

Aunque el *low-code* cuenta con muchas ventajas, también presenta algunas desventajas que se incluyen a continuación:

- Personalización: las herramientas *low-code* están diseñadas para ser fáciles de usar, esta característica limita la personalización de la aplicación en comparación con la programación convencional. Esto puede hacer que los desarrolladores o usuarios se vean limitados por las herramientas dentro de la plataforma, y no logren a cabalidad lo que desean realizar.

- Procesos de migración: en el caso hipotético que se decidiera migrar de una plataforma *low-code* a una plataforma hecha con código convencional, este proceso puede volverse complejo, largo y costoso, especialmente si se han utilizado muchos componentes de las plataformas de bajo código.

- Control de rendimiento y seguridad: debido a que son herramientas de bajo código, no se tiene el mismo control de rendimiento y seguridad como en la programación convencional. Estas plataformas tienen limitaciones y reglas ya predefinidas a las cuales los desarrolladores deben ajustarse, lo que afecta claramente el rendimiento y la seguridad de las aplicaciones.

- Costos de plataforma: al utilizar herramientas de bajo código, a menudo los costos se tornan altos en comparación con las herramientas de programación convencional de código abierto. Esto presenta un problema para las pequeñas empresas o para las empresas que cuentan con un presupuesto limitado para el desarrollo de software.

- Mantenimiento por fabricantes: los proveedores de las herramientas *low-code* tienen la capacidad de realizar actualizaciones y cambios en las diferentes soluciones cuando lo deseen. En cambio, en los casos de

aplicaciones basadas en la nube, se puede negociar con los proveedores para que las actualizaciones y cambios se realicen en un momento conveniente para la empresa.

## 5.4. Diferencias entre *low-code* y *no-code*

Las plataformas y *software low-code* están evolucionando y ha surgido una nueva filosofía conocida como el *no-code,* son dos conceptos que tienden a confundirse, ya que suelen mencionarse en conjunto. A diferencia del no-code, las aplicaciones desarrolladas con *low-code* ofrecen mayor flexibilidad y complejidad de funcionalidades, ya que se pueden agregar códigos personalizados. Las aplicaciones *no-code* están diseñadas para usuarios que no tienen conocimientos de programación puedan crearlas, ya que todos los módulos necesarios ya estas preconstruidos y solo deben elegir cual se acopla más a sus necesidades y agregar los datos.

A continuación, se muestra una tabla con las principales diferencias entre *low-code* y *no-code:*

**Tabla 1.**

*Principales diferencias entre low-code y no-code*

| **Low-code** | | **No-code** |
|---|---|---|
| Propósito | Establecer una nueva generación de herramientas de desarrollo rápido para desarrolladores | Crear aplicaciones de autoservicio para usuarios empresariales |
| Complejidad de las aplicaciones | Puede crear aplicaciones de complejidad media-alta | Puede crear aplicaciones simples |
| Perfil de los creadores de la aplicación | Desarrolladores de software | Usuarios empresariales/usuarios no técnicos |
| Objetivo principal | Desarrollo rápido | Facilidad de uso |
| Código requerido | Poco | Ninguno |

Continuación de Tabla 1.

| **Low-code** | | **No-code** |
|---|---|---|
| Personalización | Media – su capacidad de personalización depende de la LCAP | Baja – solo algunos elementos admiten cierta personalización (por ejemplo, las plantillas preconstruidas) |
| Dependencia del proveedor | Baja – facilidad para cambiar o moverse entre distintas plataformas | Alta – a menudo toto tiene que estar en la misma plataforma |

*Nota*. Principales diferencias entre el Low-Code y No-Code. Obtenido de TIC Portal (2022). *Low Code (de poco código).* (https://www.ticportal.es/glosario-tic/low-code-poco-codigo). Consultado el 19 de marzo de 2023. De dominio público.

## 5.5. Usos del *low-code*

Es utilizado para implementar soluciones empresariales y soluciones tecnológicas de una forma rápida y eficiente. Algunos de los usos comunes incluyen:

- Creación de aplicaciones móviles y web sin tener que preocuparse por escribir grandes cantidades de líneas de código.

- Las herramientas *low-code* permiten la automatización de procesos empresariales y flujos de trabajo complejos.

- Las empresas pueden utilizar el *low-code* para generar soluciones personalizadas acordes a sus necesidades, sin depender de soluciones preconstruidas que no se acoplan al cien por ciento de sus necesidades.

- El *low-code* facilita la integración de nuevas aplicaciones con los sistemas empresariales existentes, lo que mejora la eficiencia y productividad de la empresa.

## 5.6. Principales herramientas del *low-code*

A continuación, se presentan algunas de las principales herramientas *low-code:*

- Power Apps es una plataforma de desarrollo de aplicaciones empresariales de Microsoft. Esta se integra con el paquete de herramientas de Microsoft Office 365 y permite la creación de aplicaciones móviles y web personalizadas sin la necesidad de habilidades de programación. Permite conectarse a más de 200 fuentes de datos, y automatizar los procesos de negocio.

- Appian es un sistema que permite a sus usuarios crear aplicaciones con facilidad, con integración con otras herramientas y aplicaciones empresariales. Asimismo, permite la automatización de procesos y flujos de trabajo.

- Mendix es una plataforma de desarrollo de aplicaciones basadas en la nube. Esta herramienta permite a los usuarios crear aplicaciones móviles y web, con integración con múltiples servicios en la nube o herramientas empresariales. Mendix se enfoca en la colaboración y el rápido desarrollo de aplicaciones.

- OutSystems es una de las principales herramientas de bajo código que permite a los usuarios la creación de sistemas móviles o web.

OutSystems ofrece una amplia gama de herramientas para la integración de aplicaciones, la automatización de procesos y la generación de código en múltiples lenguajes.

- Salesforce Lightning es una herramienta para la creación de aplicaciones móviles y web. Ofrece a los usuarios la facilidad de integrar herramientas de Salesforce, así como la automatización de procesos y flujos de trabajo.

- Bubble ofrece a los usuarios una amplia variedad de servicios y herramientas empresariales para el desarrollo de aplicaciones web.

- Google App Maker es una herramienta de bajo código proporcionada por Google que permite la creación de aplicaciones con integración de servicios de Google como Drive y Sheets. Google App Maker también permite la automatización de procesos y flujos de trabajo.

La diferencia entre estas aplicaciones se encuentra en el enfoque, características y funcionalidades específicas. Algunas de estas aplicaciones se centran en la creación de aplicaciones móviles, mientras que otras se enfocan en aplicaciones web. Además, cada una de estas ofrece un grado de integración con otras aplicaciones y servicios diferente.

# 6. CODE

La programación es un proceso que implica el análisis, diseño y desarrollo de aplicaciones de software. Este proceso se basa en la implementación de soluciones mediante algoritmos, que se traducen en instrucciones que los ordenadores pueden entender y ejecutar.

## 6.1. ¿Para qué sirve la programación?

La programación tiene una amplia gama de usos en diversos campos, como las ciencias, las artes, el comercio, el entretenimiento, entre otros. En general, la programación se utiliza para desarrollar aplicaciones que resuelven problemas simples o complejos.

## 6.2. Tipos de programación

En la actualidad, existen varios tipos de programación utilizados comúnmente en las empresas. A continuación, se describen algunos de los más populares:

### 6.2.1. Programación estructurada

La programación estructurada es un tipo de programación que se basa en estructurar y organizar de manera lógica el código. Es una evolución de la programación secuencial y se apoya en el uso de estructuras de control. En la programación estructurada, las instrucciones se ejecutan de forma secuencial. Esta forma de codificación sentó las bases para el desarrollo de filosofías de

programación más modernas, como la programación orientada a objetos o la programación funcional.

### 6.2.2. Programación funcional

Es un paradigma de programación que se basa en el desarrollo e implementación de programas con funciones principalmente declarativas. En la actualidad, es un paradigma bastante utilizado en el desarrollo de aplicaciones web.

### 6.2.3. Programación orientada a objetos

La programación orientada a objetos es un paradigma de programación el cual utiliza como base objetos como abstracciones de la realidad. Los objetos son representaciones que se definen por sus atributos y comportamientos, y utilizan la filosofía de reutilización de código. Este paradigma se basa en cuatro conceptos principales: la herencia, el polimorfismo, el encapsulamiento y la abstracción.

# 7. DEFINICIÓN DE APLICACIÓN PARA IMPLEMENTACIÓN DE *BENCHMARKING*

La aplicación es un sistema de inventarios, el cual cuenta con diferentes módulos aplicados a la gestión de una pequeña empresa. Las gestiones que se pueden realizar a través de este sistema es la gestión del inventario, control de ventas y generación de reportes. Dicha aplicación será implementada con filosofías *no-code, low-code* y programación convencional.

## 7.1. Módulos de la aplicación

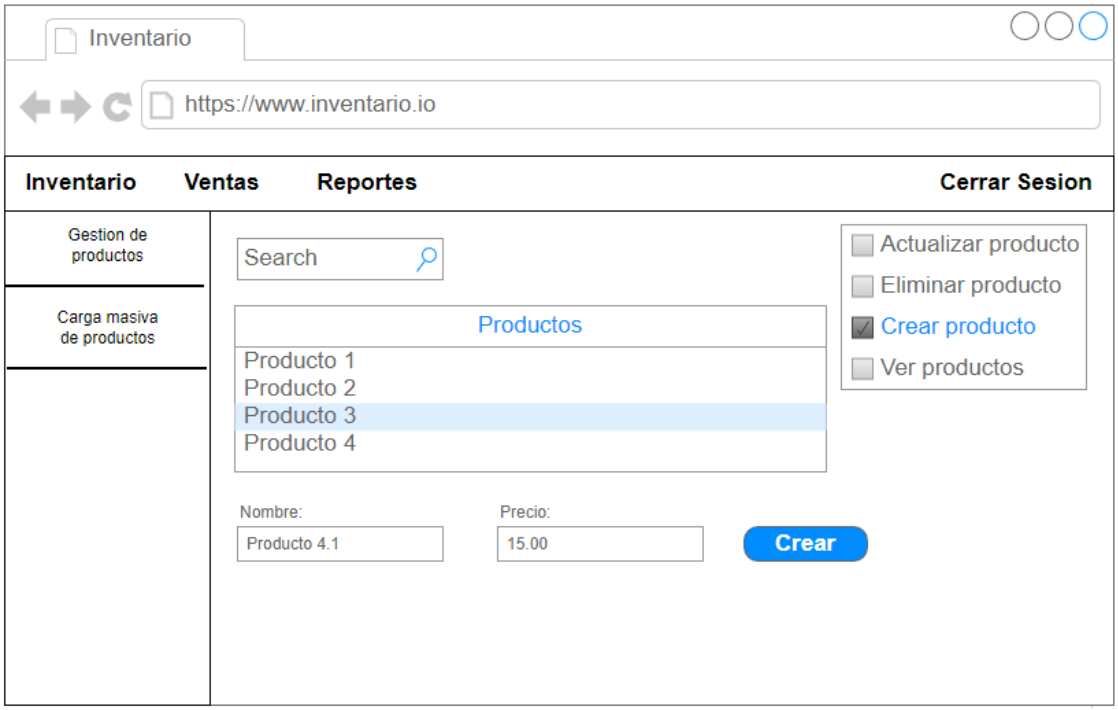
La aplicación se divide en diferentes módulos, entre los cuales podemos encontrar los siguientes:

### 7.1.1. Inventario

En el módulo de inventario se lleva a cabo el control de los productos asociados a la empresa. Las funcionalidades que se pueden encontrar es una carga masiva, creación de productos, visualizar los productos, actualizar productos y eliminar los productos.

**Figura 2.**

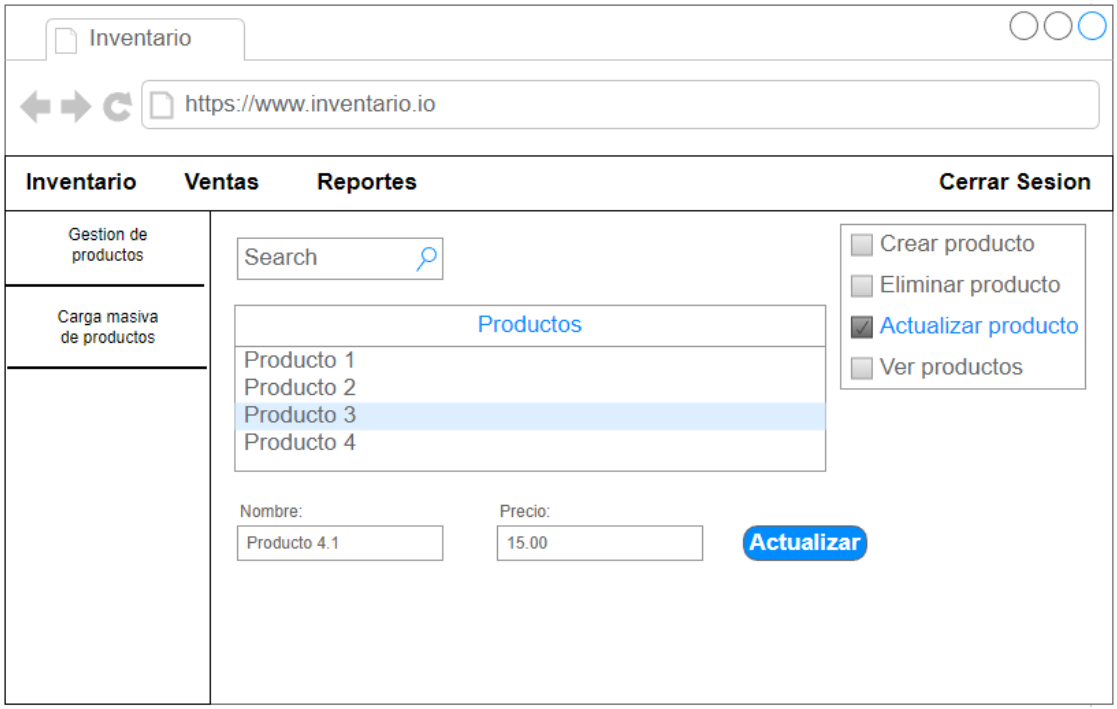
*Crear producto*




*Nota*. Vista que muestra la creación de productos. Elaboración propia, realizado con Draw.io.

**Figura 3.**

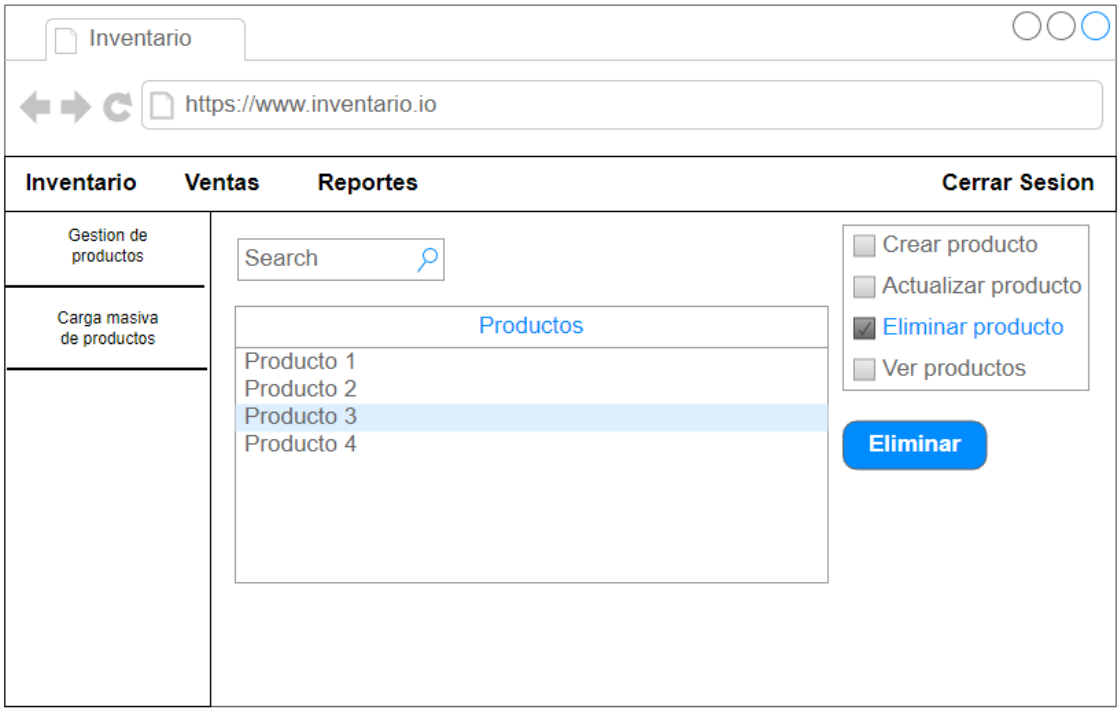
*Actualizar producto*




*Nota*. Vista que muestra la actualización de productos. Elaboración propia, realizado con Draw.io.

**Figura 4.**

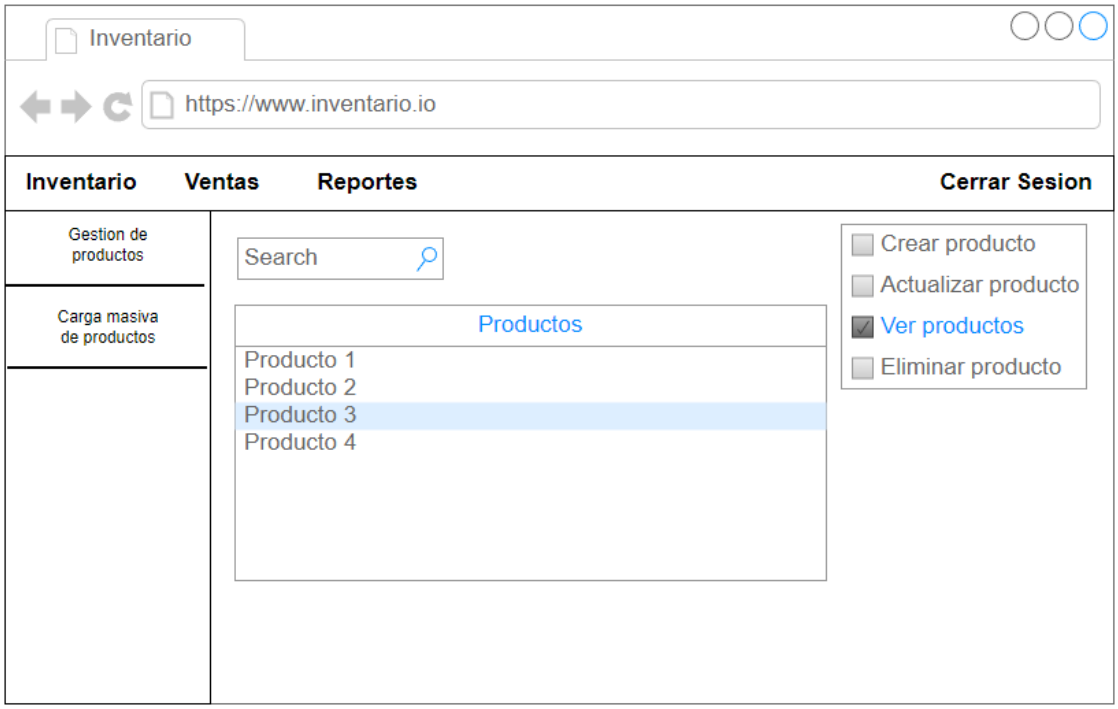
*Eliminar producto*




*Nota*. Vista que muestra la eliminación de productos. Elaboración propia, realizado con Draw.io.

**Figura 5.**

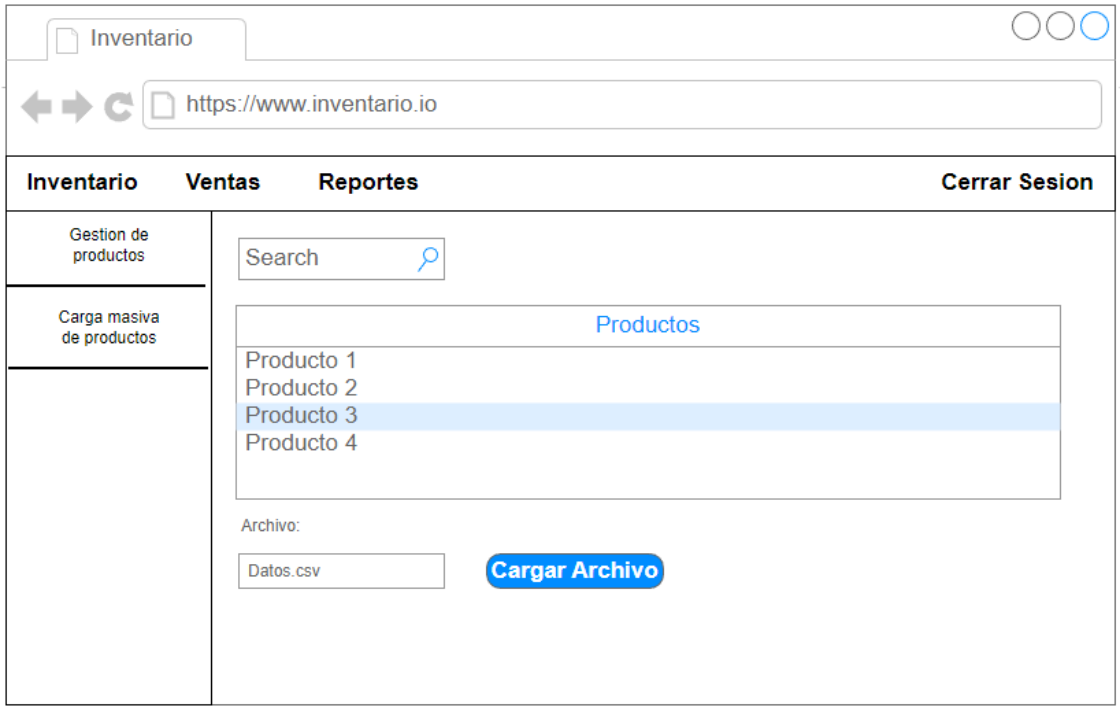
*Visualizar producto*




*Nota*. Vista que muestra la búsqueda de productos. Elaboración propia, realizado con Draw.io.

**Figura 6.**

*Carga masiva*



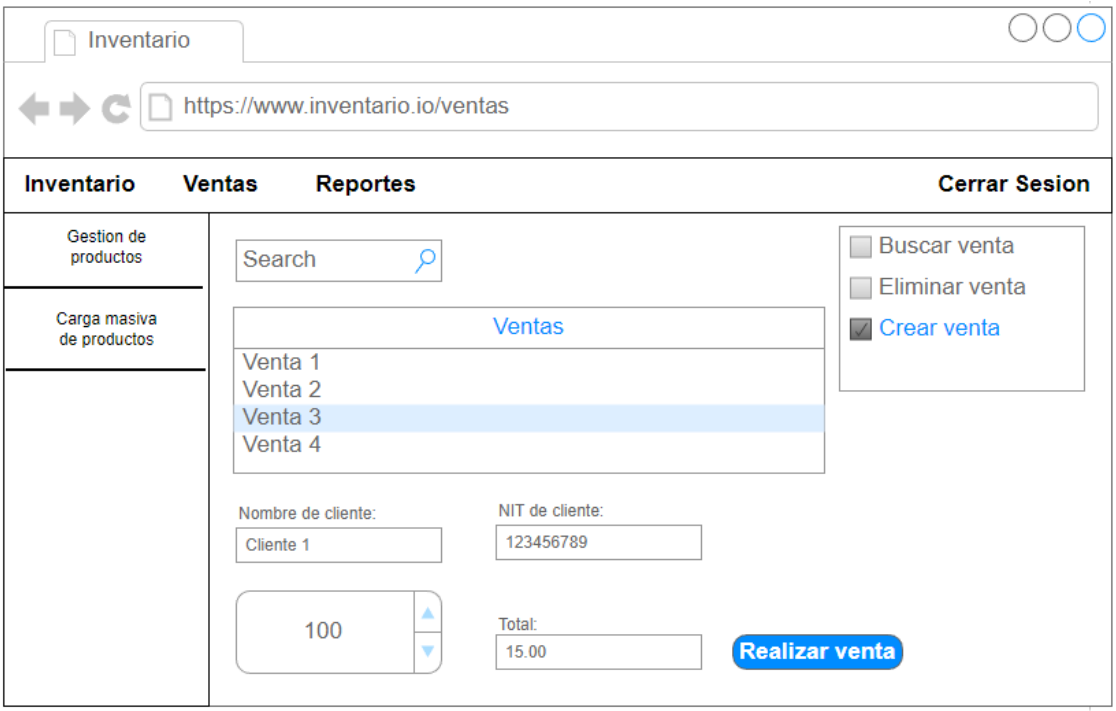

*Nota*. Vista que muestra la carga masiva de productos. Elaboración propia, realizado con Draw.io.

### 7.1.2. Ventas

El módulo de ventas es el componente que permite a los usuarios llevar un registro y control de las ventas realizadas en la empresa. Algunas de las funciones que ofrece este módulo son la generación de ventas y consultar información de cada venta que se realice por cliente o por fecha.

**Figura 7.**

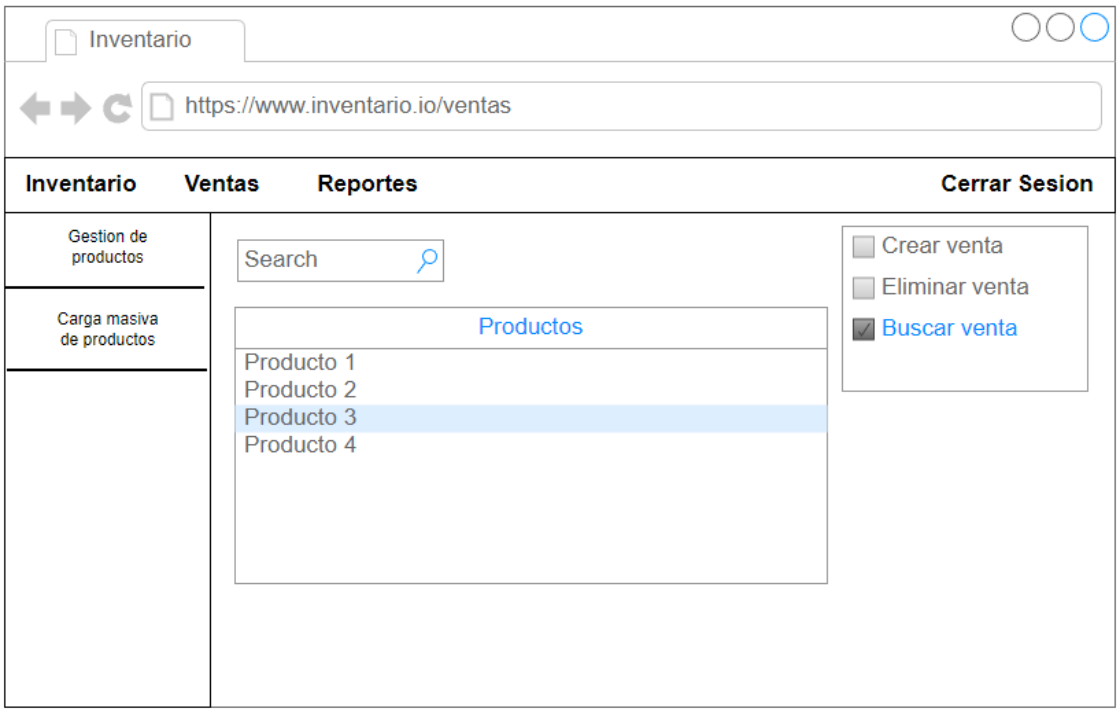
*Crear venta*




*Nota*. Vista que muestra la creación de ventas. Elaboración propia, realizado con Draw.io.

**Figura 8.**

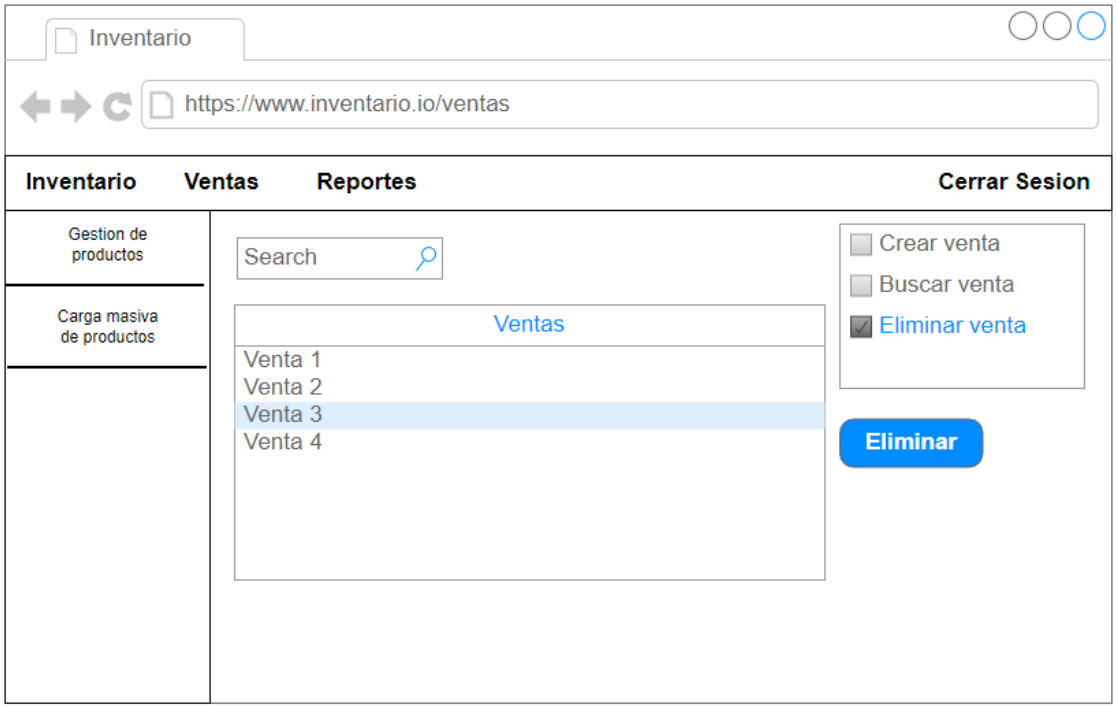
*Buscar venta*




*Nota*. Vista que muestra la búsqueda de ventas. Elaboración propia, realizado con Draw.io.

**Figura 9.**

*Eliminar venta*




*Nota*. Vista que muestra la eliminación de ventas. Elaboración propia, realizado con Draw.io.

### 7.1.3. Reportes

En un sistema de inventarios la principal función y no menos importante es la generación de reportes, este módulo permite la generación de reportes de ventas, de productos y el inventario en general. A continuación, se detalla cada uno de los reportes que se pueden encontrar dentro de la aplicación:

#### 7.1.3.1. Ventas por fecha

Muestra al usuario información de las ventas por un rango de fechas, es decir, muestra toda la información de las ventas como los productos vendidos y total de ganancias.

#### 7.1.3.2. Productos más vendidos

Presenta al usuario la información sobre los productos más vendidos en determinado rango de fechas.

#### 7.1.3.3. Inventario disponible

Permite al usuario visualizar la información el *stock* actual que se encuentra dentro del sistema de inventarios.

# 8. DESARROLLO DE APLICACIÓN WEB CON FILOSOFÍA *NO-CODE*

En la era digital en que vivimos, la capacidad de crear aplicaciones web personalizadas es fundamental para el éxito de cualquier empresa. Es por lo que a continuación se detalla el análisis y la selección de una plataforma *no-code* que cumpla con las necesidades básicas de una pequeña empresa.

## 8.1. Definición de la herramienta *no-code* para implementación

*Bubble* es una herramienta de programación sin código que permite crear y desplegar aplicaciones web sin necesidad de codificar. A continuación, se presentan algunas de las razones por las que se consideró utilizar *Bubble* para dicha implementación:

- Facilidad de uso, ofrece una interfaz de usuario intuitiva para la creación y edición de páginas web.

- Amplia gama de herramientas y recurso, desde plantillas y diseños predefinidos hasta herramientas de edición y personalización avanzadas.

- Integración con otras plataformas, como Google Analytics y WordPress.

- Seguridad, protege la página web de amenazas y vulnerabilidades. Incluyendo copias de seguridad para garantizar la persistencia de la página web.

- Soporte técnico, permite realizar consultas ante cualquier problema que se pueda presentar.

- Cuenta con una capa gratuita, para prueba de sus funcionalidades.

## 8.2. Análisis de implementación

La implementación de una aplicación *no-code* se ha vuelto una solución popular para la creación rápida de aplicaciones personalizadas sin la necesidad de conocimientos de programación avanzados. Sin embargo, las herramientas *no-code* parezcan sencillas de usar, la implementación de una aplicación sigue siendo un proceso complejo que requiere de un análisis especifico y una planificación estratégica. En este análisis de una aplicación *no-code,* se analizan los puntos más importantes de la implementación de una aplicación.

### 8.2.1. Funcionalidad

La plataforma *Bubble* otorga la esencia de la filosofía no-code, ya que proporciona herramientas visuales que son características de las plataformas pertenecientes a esta filosofía. Dentro de estas herramientas, se pueden encontrar plantillas de componentes web, definición de flujos de trabajo para la elaboración de la funcionalidad de cada componente o formulario, y definición de datos para persistencia y almacenamiento de información.

Los componentes web dentro de *Bubble* son variados, existen componentes predefinidos por la plataforma, como Input, *Checkbox*, Radio *Buttons*, entre otros. En su mayoría son componentes básicos de HTML.

Los flujos de trabajo son definidos por el usuario es decir se agrega funcionalidad en el componente que se necesite y seleccionando las funciones predefinidas por la plataforma. Las funciones predefinidas se clasifican por cuenta o usuario, navegación, bases de datos, email, pagos, estadísticas, acciones por componente y *plugin*. Como ejemplo de estas funciones predefinidas se pueden encontrar funciones de redirección a otras páginas, inicio de sesión de un usuario, cierre de sesión de un usuario, actualización de contraseñas, envío de correos, entre otros.

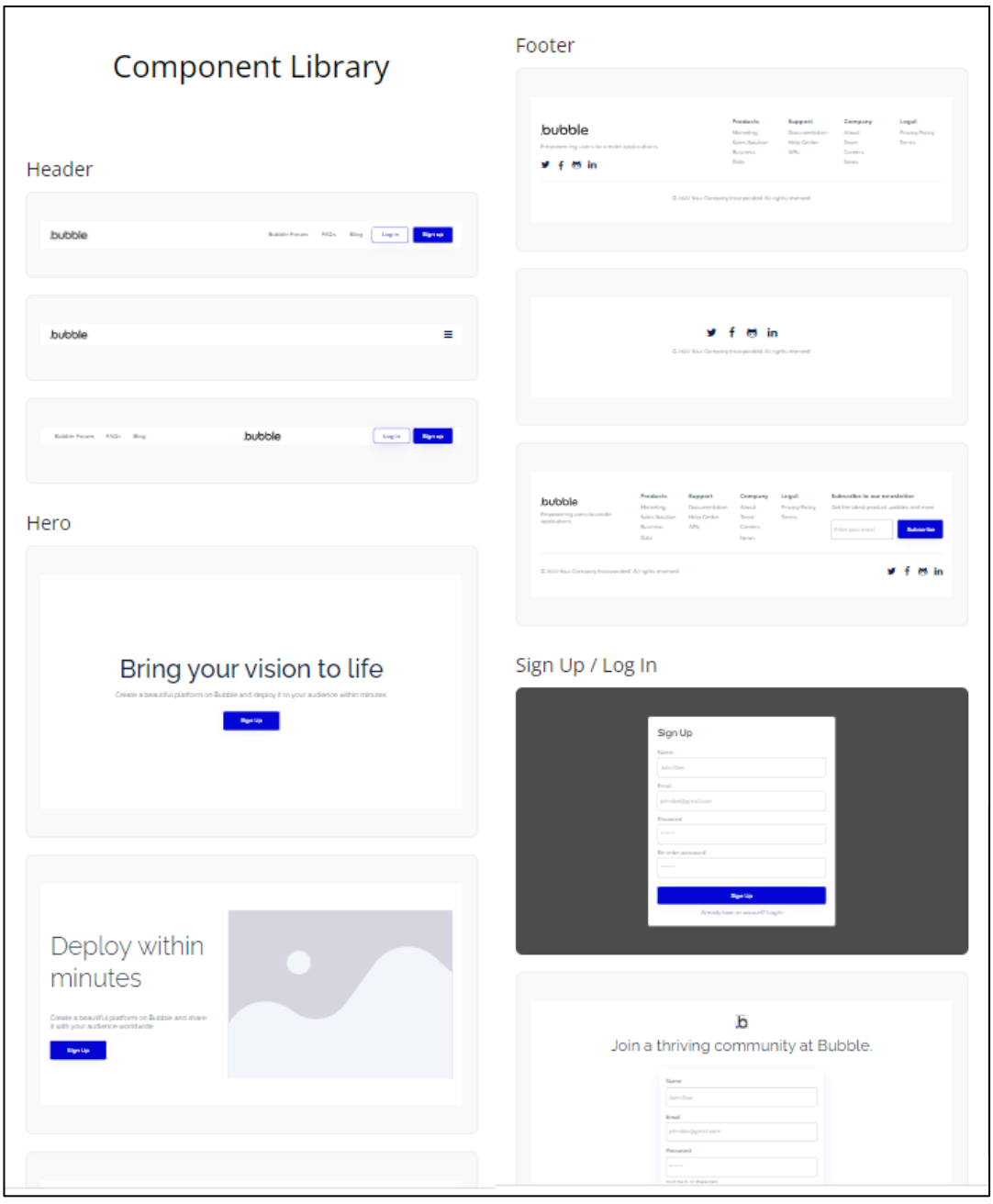
Dentro del apartado de datos, se permite la creación de tipos de datos personalizados, los cuales pueden ser interpretados como una tabla de una base de datos relacional o como una colección dentro de una base de datos no relacional. Estas tablas permiten que se agreguen los diferentes campos, según las necesidades de la implementación, sin embargo, establece ciertos campos de manera preestablecida, para los cuales no existe una explicación del por qué se agregan de forma automática.

### 8.2.2. Flexibilidad

La plataforma *Bubble* es una plataforma muy versátil ya que permite crear componentes visuales personalizados de todo tipo, formularios, paginas, barras de menú y todo tipo de componentes para crear una página web atractiva y funcional. Adicionalmente permite crear bases de datos personalizadas, de los tipos de datos estándar como cadenas, números, booleanos, entre otros.

**Figura 10.**

*Componentes Bubble.io*




*Nota*. Componentes de la capa de flexibilidad de una plataforma no-code. Elaboración propia, realizado con Paint.

### 8.2.3. Personalización

La personalización de la herramienta *no*-code *Bubble* es muy variada, permite la utilización de plantillas, la creación y la modificación de estas. Ofrece una amplia gama de personalización de componentes para las vistas de una página web, permite los cambios de color, cambios en bordes e incluso la creación de los propios componentes.

### 8.2.4. Integración

*Bubble* cuenta con una amplia gama de integraciones dentro de la plataforma se denominan *plugins*. Estos *plugins* ofrecen una amplia gama de funciones que se agrupan por categorías, tipos, precio y si es un *plugin* construido por desarrolladores de la misma plataforma o por terceros.

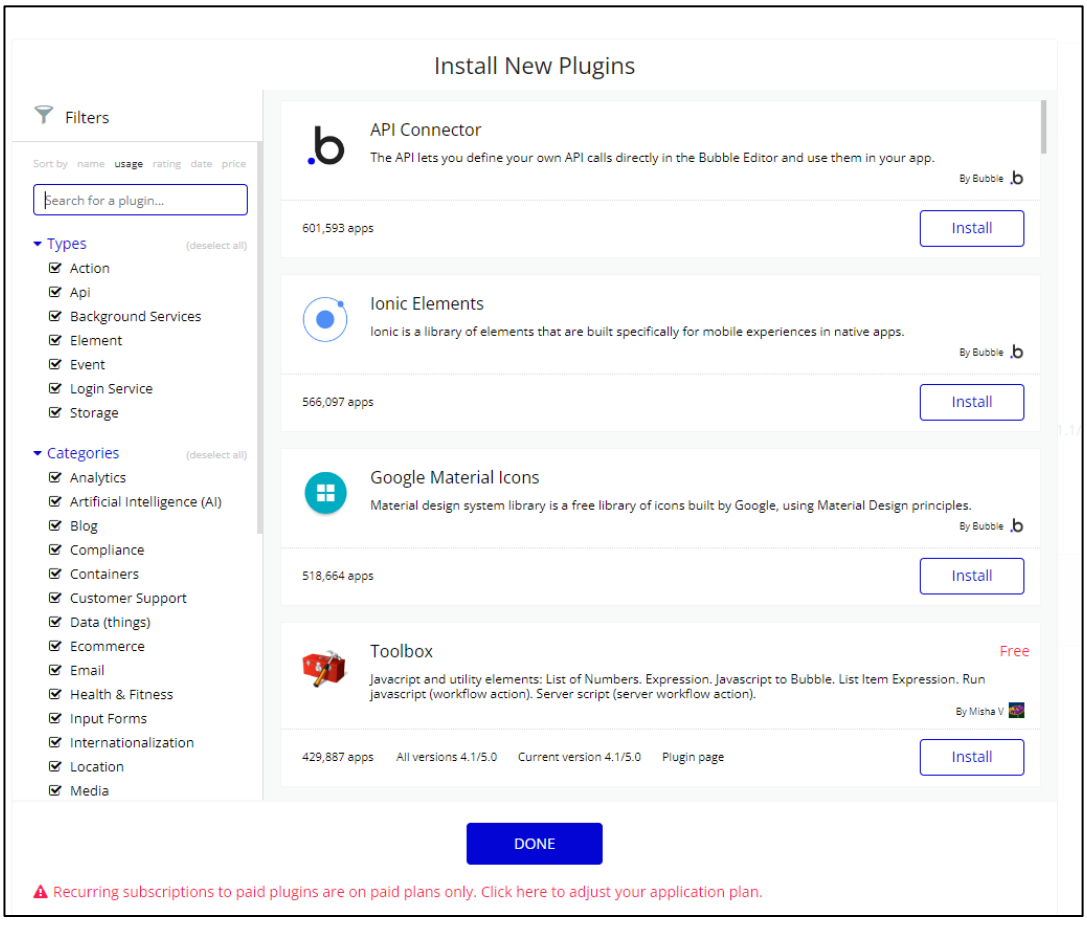
Algunas de las funciones que se incluyen dentro de los *plugins* son:

- Elementos HTML personalizados como iconos, elementos de Ionic, plantillas de gráficas, calendarios, entre otros.
- Integración con APIS de las aplicaciones de redes sociales más destacadas como Facebook, WhatsApp, Discord y Google.
- Herramientas de análisis de datos como Google Analytics y Google Charts.
- *Plugin* de correos como SendGrid, Email de Canvas y Mailchimp.

Para la instalación de *plugins* únicamente es necesario ubicar el que se desea utilizar e instalar como que fuera una aplicación normal. Este *plugin* se instala dentro de la plataforma de *Bubble* y se agrupan dentro del apartado de *plugins*. El uso de estos es sencillo, ya que únicamente se piden los enlaces y claves de APIS. La configuración de cada uno de esos servicios se realiza en cada una de las plataformas relacionadas.

**Figura 11.**

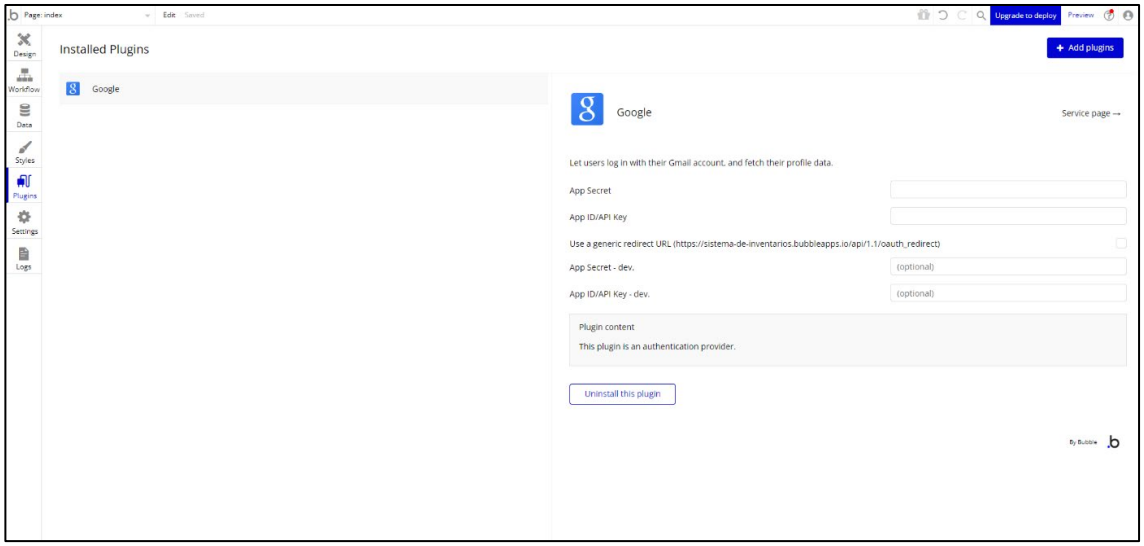
*Instalación de plugins*




*Nota*. Instalación de *plugins* dentro de la plataforma Bubble. (https://bubble.io), consultado el 26 de abril de 2023. De dominio público.

**Figura 12.**

*Listado de plugins instalados*




*Nota*. Listado de *plugins* instalados dentro de la plataforma Bubble. Obtenido de la Plataforma sin código impulsada por IA (s.f.). *Conozca el desarrollo de aplicaciones impulsado por IA*. (https://bubble.io), consultado el 26 de abril de 2023. De dominio público.

### 8.2.5. Facilidad de uso

La aplicación cuenta con tutoriales de diferente índole. Cuenta con videos, manuales de uso, sesiones interactivas y blogs de contenido. Adicionalmente cuando se crea una aplicación proporciona un tutorial guiado a través de los componentes y como utilizar cada una de las funcionalidades y partes de la plataforma.

### 8.2.6. Soporte

El soporte es una característica primordial dentro de cualquier plataforma dirigida a usuarios programadores, así como a no programadores.

*Bubble* no es la excepción dentro de las plataformas que ofrecen esta característica. Cuenta con foros en los que se encuentran las dudas más frecuentes de los usuarios, así como un apartado de contacto si la duda que tenga el usuario no es resuelta dentro de los foros. El tiempo de respuesta dentro de los foros es rápido, existen usuarios moderadores que tardan minutos en responder ante las dudas que presenten los usuarios.

### 8.2.7. Mantenimiento

El mantenimiento no será un problema, ya que la lógica de programación, así como los componentes y su funcionamiento, son fáciles de comprender. Las páginas web pueden crearse y modificarse sin afectar el resto de proyecto o vistas que existan dentro de la página web. No existen inconvenientes con querer mejorar la funcionalidad dentro de la plataforma o hacer crecer el proyecto.

### 8.2.8. Costos

Los costos para una pequeña empresa son primordiales, especialmente si se encuentra en sus inicios, ya que se busca economizar y mantenerse a flote con las pequeñas ganancias que se puedan generar. La plataforma *Bubble* cuenta con una capa gratuita que otorga los siguientes beneficios:

- Logo de *Bubble* que certifica la autenticidad de la pagina
- Protección de contraseñas
- Soporte de la comunidad
- Servidor con capacidad básica
- Editor para una aplicación
- Almacenamiento de logs durante treinta minutos

- 0.5 GB de almacenamiento

Los beneficios de esta plataforma se ven afectados por el tipo de plan que se escoja. Este tipo de plataformas limita todo por planes y conforme a ellos, la gama de funciones crece, es decir, pueden agregar almacenamiento, conexiones con API, funciones de importación con archivos, control de versiones, copias de seguridad y restauración de datos.

### 8.2.9. Escalabilidad horizontal y vertical

La escalabilidad es una parte importante dentro de todos los proyectos de programación, ya que es vital el funcionamiento óptimo de la aplicación en cuanto a tiempos de respuesta y alta disponibilidad para los clientes. La escalabilidad dentro de cualquier plataforma perteneciente a la filosofía no-code depende del proveedor. Es decir, el crecimiento de servidores para un proyecto, así como su optimización, ya no es responsabilidad del usuario que escoge una herramienta *no-code* para el desarrollo de un proyecto.

### 8.2.10. Seguridad

La seguridad en una página web es crucial para proteger la información sensible del usuario, como nombres, direcciones de correo electrónico, números de tarjetas de crédito y contraseñas, de ser robada por *hackers*. Dentro de *Bubble*, la seguridad no puede ser gestionada por el usuario como en la programación convencional, ya que allí se puede definir una llave y configuración para la encriptación de esta información, así como el uso de JWT Token para diseñar *endpoints* seguros y asegurar que la información del cliente está siendo procesada y almacenada de manera correcta.

Estas funciones de seguridad de la programación convencional no son proporcionadas por esta plataforma, se hace un manejo por el proveedor. Dentro de las pruebas realizadas se observó que, al iniciar la sesión dentro del sistema de inventarios mediante correo electrónico, este se adjuntaba a la ruta de navegación dentro del navegador. Vulnerando la dirección de correo electrónico.

# 9. DESARROLLO DE APLICACIÓN WEB CON FILOSOFÍA *LOW-CODE*

En este capítulo se detallarán los criterios que se han considerado para seleccionar la herramienta de trabajo que se utilizará para elaborar la aplicación web. También se describe el proceso de diseño de la aplicación web utilizando la filosofía *low-code*, lo que incluye la personalización y configuración de estos, la integración de los datos y la creación de los flujos de trabajo necesarios para el correcto funcionamiento de la aplicación.

## 9.1. Definición de la herramienta low-code para implementación

Budibase es una herramienta en la nube que permite la implementación y desarrollo de aplicaciones web de bajo código. Entre las características principales se encuentran una interfaz intuitiva y atractiva, así como también una curva de aprendizaje sencilla, lo que permite desarrollar soluciones web en un corto periodo de tiempo. Además, posee integración con *plugin* de terceros para ampliar las funcionalidades y permite el despliegue seguro en la plataforma y otras plataformas en la nube.

## 9.2. Análisis de implementación

En este análisis, se examinarán algunos aspectos importantes para la implementación de una aplicación *low-code* con Budibase. Se definirán sus ventajas y características, teniendo en cuenta los criterios por los que fue seleccionada y los fundamentos que llevaron a este análisis.

### 9.2.1. Funcionalidad

Budibase utiliza la filosofía *low-code*, que proporciona múltiples funcionalidades para agilizar el desarrollo de aplicaciones web. La interfaz es intuitiva y agradable, y resulta útil para diseñar y crear aplicaciones web capaces de gestionar información, automatizar procesos, crear sistemas de control, vistas de información, CRUD de datos, entre otras.

La interfaz está compuesta por un sistema *drag and drop*, lo que hace que sea realmente sencillo diseñar y crear las aplicaciones. Además, cada componente tiene la capacidad de ampliar su funcionalidad mediante la posibilidad de agregar funcionalidad mediante código escrito en JavaScript.

Provee un amplio catálogo de opciones de almacenamiento, que van desde alternativas básicas como utilizar archivos de datos en línea como Google Sheets, hasta integración con bases de datos relacionales y no relacionales robustas. Esta característica es importante para ampliar los alcances de los desarrollos.

El trabajo colaborativo es otra herramienta importante para ampliar alcances, aumentar características y reducir tiempos. Budibase provee la capacidad de integrar y promover el desarrollo colaborativo mediante *plugins* que permiten integrar el desarrollo con herramientas como Slack, Zapier y Stripe.

### 9.2.2. Flexibilidad

Es importante mencionar que Budibase es altamente flexible, ya que brinda una amplia gama de funcionalidades y herramientas para que el

desarrollo sea altamente eficiente. Además, utiliza sistemas de versionamiento robustos y brinda la posibilidad de despliegue en la nube de aplicaciones, incluyendo otras nubes altamente utilizadas. También permite trabajo colaborativo, integración con herramientas empresariales, entre otras características.

### 9.2.3. Personalización

Personalizar es una funcionalidad importante en la actualidad, tener la posibilidad de personalizar las soluciones y adaptarlas a las necesidades de los requerimientos es clave para desarrollar aplicaciones adaptables, Budibase cuenta con una amplia variedad de personalización, permite configurar las aplicaciones y que se adapten a los dispositivos, permite configurar temas, cuenta con una amplia variedad de conectividad con almacenamiento de datos y permite personalizar los procesos y las integraciones con otras herramientas.

### 9.2.4. Integración

Budibase permite integrar distintas tecnologías para aprovechar los beneficios que estas proveen. Las ventajas que esto supone para las aplicaciones web son importantes para la implementación y el desarrollo. Entre estas se encuentran la integrar *plugins* de terceros, lo que permite que Budibase establezca las bases y los *plugins* las expandan, como es el caso de, las compras en línea. Otra integración importante es la característica que tiene esta herramienta de desplegar las aplicaciones en diferentes nubes, lo que permite decidir el alcance de estas.

### 9.2.5. Facilidad de uso

Esta herramienta posee una interfaz agradable e intuitiva, ideal para personas que no cuentan con conocimientos avanzados en este tipo de herramientas. Además, su enfoque de arrastrar y soltar permite a los usuarios diseñar y personalizar aplicaciones visualmente, lo que facilita la comprensión y reduce los errores. Esto es importante porque permite reducir costos y disminuir los tiempos de desarrollo.

### 9.2.6. Soporte

Una característica importante para cualquier herramienta es el soporte, el cual es necesario en el momento del desarrollo e implementación. Budibase cuenta con documentación en su página y un foro en el que se resuelven dudas e inconvenientes. Adicionalmente, es posible resolver dudas mediante chats en vivo y correo electrónico.

### 9.2.7. Mantenimiento

Al tratarse de una herramienta intuitiva, cualquier desarrollador, incluso sin tantos conocimientos, tiene la posibilidad de adaptarse a cualquier desarrollo y solucionar inconvenientes. Además, gracias a estas mismas características, es importante mencionar que agregar más funcionalidades no tendría un impacto tan alto como lo sería en un desarrollo tradicional.

### 9.2.8. Costos

Budibase cuenta con una capa gratuita que permite el desarrollo, implementación y despliegue de hasta cuatro aplicaciones, una cantidad

ilimitada de usuarios y un almacenamiento limitado. Además, ofrece la posibilidad de realizar hasta doscientas automatizaciones de ejecución por mes.

Adicionalmente, Budibase cuenta con planes de pago que se adaptan de acuerdo con las necesidades de uso. Estos planes ofrecen características adicionales como mayor capacidad de almacenamiento, mayor cantidad de automatizaciones de ejecución, integración con bases de datos externas, soporte técnico, entre otros.

#### 9.2.9. Escalabilidad horizontal y vertical

Ampliar los recursos es una característica clave en cualquier aplicación web y Budibase cuenta con una alta capacidad de escalado, tanto horizontal como vertical. En cuanto al escalado vertical, Budibase permite incrementar los recursos a través de su sistema de arquitectura, que se puede ajustar desde las configuraciones de despliegue en línea. En términos de escalado horizontal, Budibase permite ampliar la capacidad de respuesta creando instancias en diferentes nodos, tanto en su núcleo como en su sistema de base de datos.

Adicionalmente, es posible exportar su núcleo directamente a tecnologías como Docker, lo que permite implementar la arquitectura directamente con un proveedor de nube e integrarse con Kubernetes, lo que lo hace altamente escalable.

### 9.2.10. Seguridad

En términos de seguridad, Budibase integra una capa altamente confiable y segura a través de su proxy, lo que le permite proporcionar datos mediante solicitudes HTTPS forzadas. El equipo de Budibase realiza auditorías periódicas de seguridad para reducir y detectar posibles vulnerabilidades. En cuanto a la gestión de datos, esta herramienta utiliza sistemas de bases de datos no relacionales robustos alojados en la nube de AWS y utiliza los sistemas más actualizados para mantener la integridad de la información.

# 10. IMPLEMENTACIÓN DE APLICACIÓN WEB CON PROGRAMACIÓN CONVENCIONAL

La programación convencional es la forma tradicional de desarrollo utilizada desde la creación de las computadoras. Ha evolucionado a lo largo del tiempo y su principal ventaja es que no existe un límite aparente en cuanto a la implementación y solución de problemas. Además, constantemente se crean herramientas que ofrecen mejores características y funcionalidades, ampliando el alcance de los lenguajes de programación, incluso los más antiguos. En este capítulo, se describirá una implementación tradicional de una arquitectura cliente-servidor diseñada para una pequeña empresa que satisfaga las necesidades y pueda ser tomada como referencia para comparar con los diseños de las aplicaciones *low-code* y *no-code* descritos en capítulos anteriores.

## 10.1. Definición de herramientas para implementación

Para la solución de *frontend* se utilizará el *framework* de Angular, el cual posee una amplia gama de características, esto lo hacen ideal para el desarrollo de aplicaciones web; además, cuenta con una amplia comunidad la cual provee de información y soporte, que contribuyen al desarrollo de la herramienta.

Para el desarrollo de la lógica del negocio se seleccionó una herramienta con una amplia documentación y soporte por parte de la comunidad: Node.js. Es la respuesta para la implementación de nuestro backend, el cual se encargará de gestionar la lógica del negocio. Node cuenta

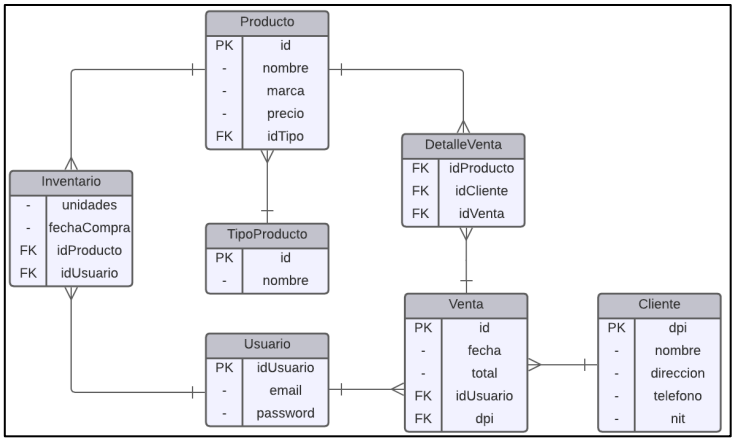
con muchas características que lo hacen ideal para el desarrollo cliente-servidor de aplicaciones web, ya que trabaja bajo el lenguaje de programación JavaScript, que es el motor que provee funcionalidad a todos los desarrollos de aplicaciones web actuales.

En cuanto a la herramienta para el almacenamiento de información, se optó por utilizar una instancia de SQL Server Express, que es una versión gratuita y limitada de este motor de base de datos. Esta versión es ideal para pequeñas empresas que no necesitan almacenar grandes volúmenes de información y que requieren de una solución de base de datos eficiente y económica. Además, la adquisición de una licencia de una versión más avanzada de SQL Server en el futuro, puede ser una opción para satisfacer las necesidades de crecimiento del negocio y tener acceso a funcionalidades más avanzadas.

Dentro de la capa de datos de la aplicación de inventario se necesita almacenar la información es por lo que el modelo entidad relación propuesto para esta aplicación es el siguiente:

**Figura 13.**

*Modelo entidad relación de aplicación de inventarios*

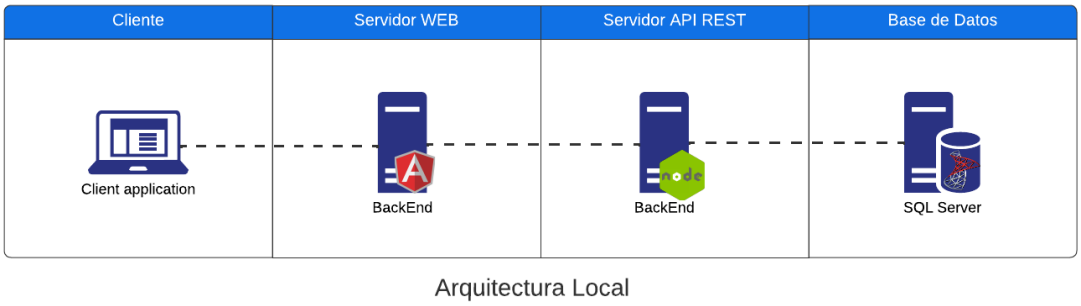



*Nota*. Modelo entidad relación propuesto para el almacenamiento de los datos de la aplicación de inventarios. Elaboración propia, realizado con Lucidchart.

La arquitectura propuesta para la aplicación de inventario es la siguiente:

**Figura 14.**

*Arquitectura cliente-servidor*




*Nota*. Arquitectura propuesta para el desarrollo tradicional. Elaboración propia, realizado con Lucidchart. Licencia Non-commercial use, DMCA.

## 10.2. Análisis de implementación

El desarrollo de aplicaciones con programación convencional proviene desde los inicios de la informática, lo que ha causado una revolución a través del tiempo en cuanto a programas y avances tecnológicos. Sin embargo, este tipo de programación no puede ser implementado por cualquier persona, es necesario tener un nivel de abstracción alto para poder comprender ciertas funcionalidades, paradigmas y tipos de ejecución. Es por lo que en este análisis se incluye la retroalimentación de los puntos más esenciales que resaltan de una aplicación implementada con este tipo de programación.

### 10.2.1. Funcionalidad

La gama de funcionalidades dentro de una aplicación con programación convencional es de un nivel muy alto, ya que todo requerimiento propuesto por un usuario puede realizarse. Dentro de la programación existen diferentes capas de personalización que pueden ser personalizadas directamente con el código fuente de la aplicación. En el caso de la aplicación de inventarios las funcionalidades se expanden gracias a los módulos de Node.js que son incluidos tanto en el *backend* como en el *frontend*. Al ser tecnologías que utilizan Javascript, se vuelven muy versátiles como este lenguaje de programación.

Sin embargo, algunos de estos módulos de Node.js cuentan con poca documentación, por lo que su uso se vuelve tedioso o incluso nulo en dado caso no se comprendiera la funcionalidad de estos, ya que varios de estos módulos son creados por terceros. Es por lo que al utilizar cualquiera de estos módulos es necesario validar si existe suficiente documentación y ejemplos para comprender su funcionamiento.

### 10.2.2. Flexibilidad

La flexibilidad es uno de los puntos fuertes de la programación tradicional. A diferencia de herramientas como el *low-code* o *no-code*, la programación tradicional es considerablemente más flexible. Desde el punto de vista del desarrollo, no existe un límite específico para lo que se puede o no implementar en una aplicación web. Un claro ejemplo de esto son las bibliotecas desarrolladas por la comunidad, ya que agilizan y brindan funcionalidades adicionales en poco tiempo de implementación.

Angular proporciona un modelo de trabajo por módulos que agiliza el desarrollo de aplicaciones y permite reutilizar el código. Además, es compatible con bibliotecas creadas por la comunidad que agregan funcionalidad adicional. Node.js permite incorporar módulos y un entorno de desarrollo capaz de crear desarrollos escalables. Utiliza el potente lenguaje de JavaScript, que es uno de los más adaptables y flexibles, y permite la instalación fácil de módulos o bibliotecas. SQL Server es una de las mejores opciones en el mercado para el almacenamiento de información. Se utiliza en una multitud de empresas y posee las cualidades necesarias para escalar los servidores, almacenar datos, además de brindar compatibilidad con un gran número de aplicaciones.

### 10.2.3. Personalización

A diferencia de *low-code* y *no-code*, que ofrecen cierto nivel de personalización, pero limitado, la metodología tradicional de desarrollo ofrece múltiples ventajas en cuanto a diseño, flexibilidad y personalización. Angular, por ejemplo, permite diseñar módulos personalizables y exportables, la utilización de bibliotecas, y una amplia gama de temas y estilos para diseñar

sus componentes. Node.js cuenta con bibliotecas fáciles de implementar y diseños de arquitecturas que se adaptan a las necesidades de los sistemas informáticos. SQL Server es uno de los sistemas más personalizables en el mercado; entre sus principales características se encuentran configuraciones avanzadas y personalizadas, la implementación de rutinas con funciones, procedimientos y triggers, así como configuraciones personalizadas en temas de seguridad y almacenamiento de información.

### 10.2.4. Integración

Actualmente, los sistemas informáticos se diseñan para que los sistemas interactúen y se acoplen. Las arquitecturas modernas se componen de la vista o *frontend*, la lógica del negocio o *backend* y un sistema de almacenamiento. Hablando puntualmente de Angular, Node.js y SQL Server, estos comparten herramientas que garantizan la correcta integración de estos sistemas. Algunas ventajas podrían ser que comparten lenguajes de programación, lo que facilita el desarrollo. Los métodos de comunicación por protocolos HTTP agilizan la comunicación entre los distintos componentes del sistema y brindan seguridad y robustez.

### 10.2.5. Facilidad de uso

En términos de desarrollo, las aplicaciones web modernas ofrecen una amplia variedad de funcionalidades, las cuales dependen de los requerimientos de la solución que se desea implementar. Sin embargo, en términos de desarrollo, estas están limitadas al equipo de desarrollo y a las habilidades de este. En términos generales, la dificultad dependerá del lenguaje de programación y del *framework* o librería que se utilice.

Angular es un *framework* muy completo que ofrece desarrollos robustos enfocados en la reutilización de código, pero esto supone que el equipo de desarrollo debe dominar los conocimientos necesarios para satisfacer las necesidades de los requerimientos. Node.js, por su parte, es un entorno de desarrollo diseñado para el desarrollo de aplicaciones del lado del servidor. La curva de aprendizaje de este es moderada, pero se simplifica en caso de que los desarrolladores cuenten con conocimientos del lenguaje de JavaScript. En cuanto al uso de SQL Server, este depende de la complejidad de la lógica del negocio y los requisitos de información del sistema.

En resumen, para el desarrollo de una aplicación tradicional, resulta que la curva de aprendizaje es alta para un equipo con poca o nula experiencia.

#### 10.2.6. Soporte

Una de las ventajas que posee el desarrollo tradicional es que la mayoría de las empresas que mantienen los *frameworks* o librerías se encargan de mantener actualizada la documentación con los cambios más recientes y nuevas funcionalidades. Además, las comunidades aportan conocimiento profesional y de campo, así como apoyo en la resolución e identificación de bugs y nuevas funcionalidades, en contraste con algunas herramientas de low-code y *no-code* que tienen deficiencias en cuanto a documentación e información.

#### 10.2.7. Mantenimiento

En la actualidad el mantenimiento de una aplicación es muy importante, ya que permite ver el crecimiento del negocio agregando nuevas reglas o funcionalidades. Así como las correcciones necesarias para que se ajuste la

funcionalidad de la aplicación a las necesidades del cliente. Para el sistema de inventarios se utilizó una arquitectura cliente servidor es decir un servidor que se encarga de distribuir la página web y otro que se encarga de responder a las solicitudes de este cliente web. Al utilizar esta arquitectura se permite mantener el control de cada una de las aplicaciones como independientes, ya que conforme a las necesidades se pueden ir agregando actualizaciones o corrigiendo en el servidor que sea necesario.

Dentro del desarrollo de la aplicación se utilizó la arquitectura que provee angular, haciendo el uso de componentes, servicios y módulos para organizar y gestionar la aplicación de forma eficiente. Para el *backend* se utilizó una arquitectura MVC (Modelo Vista Controlador) y servicios REST permitiendo una mejor organización de *endpoints* y del código.

### 10.2.8. Costos

Las aplicaciones que utilizan desarrollos convencionales pueden ser desplegadas de dos maneras, de forma local y en la nube. Al implementar un despliegue en la nube para el sistema de inventarios se incurre en gastos como dominio para el sitio web, para el cliente web, para el servidor *backend* y para la base de datos. así como la licencia para base de datos si se desea algún tipo de soporte para este servidor. Estos gastos pueden ser disminuidos utilizando algún servicio de nube como Google Cloud o AWS para el cual se pagaría por mes el tiempo que estén en línea los servicios y la cantidad de almacenamiento que se necesite en la base de datos.

### 10.2.9. Escalabilidad horizontal y vertical

Al ser una aplicación desarrollada desde cero, las capacidades de escalabilidad son muy grandes. El único límite es el presupuesto del que se disponga, ya que depende de donde se tenga el despliegue se incurre en gastos Capex u Opex. El sistema de inventarios al encontrarse dentro de AWS se incurre en gastos Opex, esto permite que la aplicación sea muy estable en cuanto a la disponibilidad de servidores, dando una capacidad de aumentar los recursos si es necesario dando una amplia escalabilidad horizontal.

Dentro del margen de la escalabilidad vertical, al ser una aplicación con despliegue en la nube el único límite de personalización es la capacidad de configuración que ofrecen dentro de AWS.

### 10.2.10. Seguridad

Dentro de la seguridad de la aplicación de inventarios y para una aplicación implementada con desarrollos convencionales la capacidad de mejorar la seguridad es diversa, se implementó utilizando métodos como JWT, módulos de NodeJS como Helmet los cuales agregan capas de seguridad al sistema utilizando las cabeceras y estableciendo que únicamente ciertas personas puedan acceder directamente al contenido del inventario.

A diferencia de una aplicación *no-code* y *low-code*, la seguridad queda del lado del desarrollador, la cual debe ser gestionada de forma correcta para no ser afectados por personas mal intencionadas.

# 11. *BENCHMARKING*

Un *benchmarking* es la evaluación y comparación de un producto, servicio o proceso. Con el cual se busca identificar las mejores prácticas, establecer los mejores estándares de rendimiento y aplicar conocimientos creando oportunidades de mejora en eficiencia, calidad y competitividad.

Se establecieron diferentes medidas de calificación para cada métrica, adaptándose al contexto del aspecto evaluado. Algunos ejemplos incluyen: no cumple, no aplica, limitado y cumple. Estas medidas se utilizaron para evaluar cada uno de los aspectos más relevantes de cada métrica, en base a la experiencia adquirida durante la implementación de aplicaciones web tipo inventario en las diferentes tecnologías.

## 11.1. Métricas

Las métricas son las herramientas que permiten la evaluación del desempeño de cada una de las herramientas, permitiendo identificar mejoras, fortalezas y debilidades de cada una. A continuación, se incluyen las métricas que se utilizaron para realizar la comparativa entre las tecnologías de programación *no-code, low-code* y la programación convencional.

### 11.1.1. Funcionalidad

Como parte de la evaluación de la métrica de funcionalidad se consideraron los siguientes aspectos: es importante evaluar el comportamiento de las distintas metodologías de programación ante

escenarios con procesos complejos, así como también el nivel de adaptación que ofrecen estos ante este tipo de situaciones. El análisis incluye aspectos importantes para el desarrollo de aplicaciones robustas y escalables, pero sin descuidar otros aspectos importantes como lo son la integridad y la consistencia de la información.

**Tabla 2.**

*Funcionalidad en las tecnologías y programación convencional*

| ASPECTO | TECNOLOGÍA | | |
|---|---|---|---|
| | **LOW-CODE** | **NO-CODE** | **CONVENCIONAL** |
| Desarrollo de flujos de trabajo complejo | Limitado | No cumple | Cumple |
| Estandarización | Limitado | No cumple | Cumple |
| Integridad y consistencia | Cumple | Cumple | Cumple |
| Interoperabilidad | Limitado | Limitado | Cumple |
| Reutilización | Limitado | Limitado | Cumple |

*Nota.* Evaluación de la funcionalidad en las tecnologías no-code, low-code y programación convencional. Elaboración propia, realizado con Word.

### 11.1.2. Flexibilidad

Dentro de la evaluación de la flexibilidad, se consideraron tres aspectos clave. En primer lugar, se analizó la adaptabilidad a cambios, es decir la capacidad de ajustarse a las necesidades o modificaciones que existan dentro del proyecto. En cuanto a modularidad se consideró la estructuración de las aplicaciones en módulos reutilizables. Finalmente, se evaluó si dichas tecnologías permiten la configuración sin programación y la facilidad de llevar a cabo las configuraciones sin un vasto conocimiento en programación.

Se obtuvieron los siguientes resultados:

**Tabla 3.**

*Evaluación de los aspectos principales de la flexibilidad*

| ASPECTO | TECNOLOGÍA | | |
|---|---|---|---|
| | **LOW-CODE** | **NO-CODE** | **CONVENCIONAL** |
| Adaptable a cambios | Limitado | Limitado | Cumple |
| Modularidad | Cumple | Cumple | Cumple |
| Configurable sin programación | Cumple | Cumple | No cumple |

*Nota.* Evaluación de los aspectos principales de la flexibilidad. Elaboración propia, realizado con Word.

### 11.1.3. Personalización

Como parte de la evaluación de la métrica de personalización se consideraron los siguientes aspectos: Como principal objetivo se evaluaron las capacidades de las herramientas para crea y diseñar entornos visuales atractivos, adicionalmente también se evaluaron otros aspectos importantes como lo son la capacidad de configuración y flexibilidad de adaptarse a las necesidades de los posibles tipos de requerimiento.

**Tabla 4.**

*Personalización en las tecnologías y programación convencional*

| ASPECTO | TECNOLOGÍA | | |
|---|---|---|---|
| | **LOW-CODE** | **NO-CODE** | **CONVENCIONAL** |
| Configuración y parametrización | Limitado | Limitado | Cumple |
| Desarrollo de interfaces | Cumple | Cumple | Cumple |
| Flexibilidad de diseño | Limitado | Limitado | Cumple |
| Reutilización de componentes | Limitado | Limitado | Cumple |

*Nota.* Evaluación de la personalización en las tecnologías *no-code, low-code* y programación convencional. Elaboración propia, realizado con Word.

### 11.1.4. Integración

La integración es una de las métricas más importantes, ya que se evalúa la compatibilidad con otros sistemas. Esto porque la mayor parte de instituciones ya cuentan con algún sistema para el cual se busque una sustitución o una mejora. También se valoró la facilidad de integración, que libertad existe para la interconexión entre sistemas. Por último, se analizó el consumo de recursos externos, especialmente a través de APIs, un método esencial para desarrollos web.

Al realizar una evaluación de la integración se obtuvieron las siguientes calificaciones:

**Tabla 5.**

*Evaluación de los aspectos principales de la integración*

| ASPECTO | TECNOLOGÍA | | |
|---|---|---|---|
| | LOW-CODE | NO-CODE | CONVENCIONAL |
| Compatibilidad con otros sistemas | Limitado | Limitado | Cumple |
| Facilidad de integración | Fácil | Fácil | Complejo |
| Consumo de recursos externos | Limitado | Limitado | Cumple |

*Nota*. Evaluación de los aspectos principales de la integración de las tecnologías de programación no-code, low-code y la programación convencional con otros sistemas. Elaboración propia, realizado con Word.

### 11.1.5. Facilidad de uso

Esencialmente se evaluaron tres aspectos muy importantes para el manejo, el aprendizaje y la experiencia de usuario. El enfoque del marco de

trabajo en el cual se centró la investigación es en el personal poco capacitado pero dispuesto afrontar retos. Principalmente es necesario establecer que, una herramienta ideal debe de contar con un nivel moderado de dificultad, esto evitara la frustración de no entender el inicio o fin del aprendizaje y que invite al desarrollador a seguir investigando y progresando.

**Tabla 6.**

*Facilidad de uso en las tecnologías y programación convencional*

| ASPECTO | TECNOLOGÍA | | |
|---|---|---|---|
| | LOW-CODE | NO-CODE | CONVENCIONAL |
| Curva de aprendizaje | Media | Baja | Alta |
| Experiencia de usuario | Fácil | Fácil | Muy Difícil |
| Satisfacción del usuario | Alta | Alta | Baja |

*Nota*. Evaluación de la facilidad de uso en las tecnologías no-code, low-code y programación convencional. Elaboración propia, realizado con Word.

### 11.1.6. Soporte

Para la evaluación de esta métrica se utilizaron aspectos como la disponibilidad, calidad de soporte, canales disponibles, el tiempo en que se obtiene respuesta y la documentación existente.

La disponibilidad se refiera a la capacidad de acceder al soporte siempre que sea necesario, asegurando que los usuarios puedan obtener asistencia en el momento adecuado. La calidad del soporte implica que la ayuda o la información que fue otorgada sea precisa y efectiva para resolver los problemas que puedan surgir. Los canales de soporte hacen referencia a los medios a través de los cuales se ofrece asistencia, medios como el correo electrónico, llamadas telefónicas, chat en vivo o el uso de plataformas como

WhatsApp, lo que garantiza que los usuarios puedan elegir la vía más conveniente para obtener ayuda. El análisis aplicado sobre esta métrica devolvió los siguientes resultados:

**Tabla 7.**

*Principales aspectos del soporte*

| ASPECTO | TECNOLOGÍA | | |
|---|---|---|---|
| | LOW-CODE | NO-CODE | CONVENCIONAL |
| Disponibilidad | Limitado | Limitado | Cumple |
| Calidad de soporte | Cumple | Cumple | Cumple |
| Canales de soporte | Limitado | Limitado | Cumple |
| Tiempo de respuesta | Limitado | Limitado | Limitado |
| Documentación | Limitado | Limitado | Cumple |

*Nota*. Evaluación de los principales aspectos relacionados con el soporte para las tecnologías no-code, low-code y programación convencional. Elaboración propia, realizado con Word.

### 11.1.7. Mantenimiento

El uso de métricas como la facilidad de depuración, facilidad de diagnóstico, control de versiones y cobertura de pruebas es esencial para garantizar un desarrollo de software eficiente y de alta calidad. Estas métricas permiten identificar y resolver errores rápidamente, diagnosticar problemas con precisión, gestionar cambios de manera ordenada y asegurar que gran parte del código esté probado para minimizar fallos. En conjunto, fomentan la colaboración entre equipos, mejoran la estabilidad del producto y reducen el tiempo y los costos asociados al mantenimiento y desarrollo continuo.

**Tabla 8.**

*Mantenimiento en las tecnologías y programación convencional*

| ASPECTO | TECNOLOGÍA | | |
|---|---|---|---|
| | LOW-CODE | NO-CODE | CONVENCIONAL |
| Facilidad de depuración | Limitado | Limitado | Cumple |
| Facilidad de diagnostico | Limitado | Limitado | Cumple |
| Control de versiones | Limitado | Limitado | Cumple |
| Cobertura de pruebas | Limitado | Limitado | Cumple |

*Nota*. Evaluación del mantenimiento en las tecnologías no-code, low-code y programación convencional. Elaboración propia, realizado con Word.

### 11.1.8. Costos

Los costos se encuentran distribuidos en diferentes clasificaciones, desde un punto de vista contable. Es por lo que se analizaron cada una de sus clasificaciones, analizando cada uno de los aspectos más relevantes para la implementación de la aplicación web. A continuación, se detallan estos aspectos:

#### 11.1.8.1. Costo inicial

Dentro del costo inicial para las tecnologías *low-code/no-code* implica el uso de suscripciones o licencias de plataforma, así mismo con lleva un tiempo de configuración inicial y capacitaciones sobre la plataforma que se esté utilizando.

La programación convencional hace el uso de desarrolladores, lo que implica contrataciones, salarios, beneficios. Herramientas y entornos de desarrollo. Licencias de software, servidores de desarrollo y hardware.

#### 11.1.8.2. Costo de operación

Para las plataformas *low-code/no-code* existen cuotas recurrentes, costos de soporte y actualizaciones de la plataforma. Los costos de operación para la programación convencional incluyen los salarios de los desarrolladores y equipos de soporte, así como el mantenimiento de servidores y el hardware.

#### 11.1.8.3. Costo de integración

La integración con otros sistemas *legacy* aumenta los costos ya que se utilizan conectores o Apis adicionales. La programación convencional incluye los costos de desarrollo y el mantenimiento de las integraciones personalizadas.

#### 11.1.8.4. Costo de escalabilidad

Los costos de escalabilidad para las plataformas *low-code/no-code* varían en función de la cantidad de usuarios y los datos que se dispongan dentro de estas. Estos costos para la programación convencional se incrementan ya que a medida que incrementa la demanda es necesario ampliar la infraestructura y optimizar el código para manejar mayor carga.

**Tabla 9.**

*Costos de implementación de una aplicación web*

| ASPECTO | TECNOLOGÍA | | |
|---|---|---|---|
| | LOW-CODE | NO-CODE | CONVENCIONAL |
| Costo inicial | Bajo | Bajo | Alto |
| Costo de operación | Medio | Medio | Alto |
| Costo de integración | Bajo | Medio | Alto |
| Costos de escalabilidad | Alto | Alto | Alto |

*Nota*. Evaluación de los costos de implementación de una aplicación web utilizando tecnologías *low-code, no-code* y programación convencional. Elaboración propia, realizado con Word.

### 11.1.9. Escalabilidad horizontal y vertical

El uso de métricas como escalabilidad horizontal, escalabilidad vertical, monitoreo, gestión y optimización de recursos es clave para garantizar un sistema eficiente y adaptable. Estas métricas permiten evaluar la capacidad del sistema para manejar cargas crecientes mediante la adición de servidores (horizontal) o mejoras en el hardware existente (vertical). Además, el monitoreo asegura una supervisión continua del rendimiento, mientras que la optimización de recursos maximiza el uso eficiente de infraestructura, reduciendo costos y mejorando la sostenibilidad. En conjunto, estas métricas aseguran un sistema robusto, flexible y preparado para responder a demandas cambiantes.

**Tabla 10.**

*Escalabilidad horizontal y vertical*

| ASPECTO | TECNOLOGÍA | | |
|---|---|---|---|
| | LOW-CODE | NO-CODE | CONVENCIONAL |
| Escalabilidad horizontal | Media | Limitada | Muy alta |
| Escalabilidad vertical | Media | Limitada | Muy alta |
| Monitoreo y gestión de recursos | Media | Media | Muy alta |
| Optimización de recursos | Limitada | Limitada | Muy alta |

*Nota*. Evaluación de la escalabilidad horizontal y vertical en las tecnologías no-code, low-code y programación convencional. Elaboración propia, realizado con Word.

### 11.1.10. Seguridad

Para evaluar la seguridad en las tecnologías, se consideraron cuatro aspectos clave. Primero, la protección de datos, que abarca el aislamiento de información, la replicación de los datos y las medidas para proteger la información ante desastres o pérdidas. En segundo lugar, se evalúo el uso de cifrado, considerando si se aplican algoritmos modernos y efectivos para proteger los datos sensibles.

El tipo de autenticación también fue analizado, verificando si se implementa algún tipo de mecanismo como la autenticación multifactor (MFA) o protocolos seguros de autenticación en red. Finalmente, se examinó la gestión de vulnerabilidades, es decir, que medidas se toman para prevenir, detectar y mitigar ataques o brechas de seguridad en la plataforma. Esta evaluación de seguridad arrojo los siguientes resultados para cada tipo de tecnología:

**Tabla 11.**

*Seguridad en las tecnologías y programación convencional.*

| ASPECTO | TECNOLOGÍA | | |
|---|---|---|---|
| | LOW-CODE | NO-CODE | CONVENCIONAL |
| Protección de datos | Limitado | Limitado | Cumple |
| Cifrado | No cumple | No cumple | Cumple |
| Autenticación | Cumple | Cumple | Cumple |
| Gestión de vulnerabilidades | No cumple | No cumple | Cumple |

*Nota*. Evaluación de la seguridad en las tecnologías *no-code, low-code* y programación convencional. Elaboración propia, realizado con Word.

## 11.2. Análisis comparativo

A continuación, se incluye el análisis de los resultados al evaluar las diferentes métricas incluidas en la sección anterior:

### 11.2.1. Funcionalidad

Según el análisis realizado a las herramientas bajo las metodologías de desarrollo no-code, low-code y desarrollo tradicional para las siguientes métricas se obtuvieron los siguientes resultados.

#### 11.2.1.1. Desarrollo de flujos de trabajo complejo

- *No-code*: esta metodología de trabajo es la más intuitiva y amigable para el desarrollador, permite implementar funcionalidades de manera ágil, además de ser sumamente accesible para desarrolladores con conocimientos básicos y poca experiencia. La principal desventaja de esta metodología es el bajo nivel de complejidad que puede ofrecer en

cuanto a funcionalidades, las cuales se limitan a las que las distintas herramientas provean.

- *Low-code*: en comparación con *no-code,* esta metodología permite la personalización mediante la incorporación de fragmentos de codificación, lo que la hace más adaptable para desarrollos complejos. Por su parte, esto aumenta la complejidad del aprendizaje para desarrolladores con conocimientos básicos de programación. Adicionalmente, se debe tomar en cuenta que el nivel de personalización para funcionalidades complejas se limita a lo que la herramienta permita.

- Desarrollo tradicional: el desarrollo tradicional permite la implementación de cualquier tipo de funcionalidad compleja. A diferencia de las metodologías anteriores, esta no se limita a un entorno de desarrollo específico. Como contraparte, tiene la particularidad de que el desarrollador debe contar con conocimientos específicos para poder implementar las funcionalidades, lo que hace que el proceso sea tan complejo como la funcionalidad que se desee desarrollar.

#### 11.2.1.2. Estandarización

- *No-code* y *low-code*: en cuanto a la estandarización de funcionalidad, las plataformas no-code y low-code comparten muchas similitudes, esto se debe a que están limitadas a las capacidades que la herramienta o plataforma provea. Esta limitación puede ocasionar desarrollos con bajos niveles de reutilización de funcionalidad, lo que posteriormente se traduce en aplicaciones ineficientes.

- Desarrollo tradicional: el desarrollo tradicional permite altos niveles de estandarización al crear aplicaciones web eficientes, esto gracias a diversos factores como el lenguaje de programación utilizado, las herramientas disponibles y la filosofía del lenguaje.

#### 11.2.1.3. Integridad y consistencia

En términos generales, las filosofías de desarrollo están diseñadas para cumplir estándares que proporcionan a los desarrolladores ciertos niveles de confiabilidad en la creación de aplicaciones robustas. Por lo tanto, se puede establecer que todas las metodologías cumplen con estándares que dependen de tres factores principales: la tecnología utilizada, las herramientas implementadas y el equipo de trabajo.

#### 11.2.1.4. Interoperabilidad

- *Low-code* y *no-code*: en cuanto a interoperabilidad, las plataformas *low-code* y *no-code* ofrecen características limitadas, ya que estas dependen de la herramienta o plataforma que se utilice. En general, proporcionan el soporte necesario para la interconexión de tecnologías más comunes. Sin embargo, como consideración adicional, si se requiere funcionalidad de una tecnología específica, es probable que esta no esté incluida entre las tecnologías soportadas por la plataforma.

- Desarrollo tradicional: la metodología de desarrollo tradicional ofrece soporte completo para casi cualquier tipo de funcionalidad que requiera comunicación con otras tecnologías. Sin embargo, su principal desventaja radica en que este nivel de soporte frecuentemente requiere un alto grado de especialización por parte del desarrollador.

#### 11.2.1.5. Reutilización

- *Low-code* y *no-code*: estas metodologías utilizan mecanismos similares para brindar funcionalidad, entre los que se pueden mencionar la modularización, la reutilización de flujos y las plantillas. Sin embargo, estos mecanismos están limitados por el nivel de funcionalidad que permite la herramienta utilizada.

- Desarrollo tradicional: por su parte, la metodología tradicional posee numerosas herramientas y frameworks creados con el fin de reutilizar código. Sin embargo, su principal desventaja radica en el alto nivel de conocimiento técnico necesario para implementar este tipo de desarrollo.

### 11.2.2. Flexibilidad

El análisis para la métrica flexibilidad es el siguiente:

#### 11.2.2.1. Adaptable a cambios

- *Low-code* se clasificó como limitado. Esto porque permite realizar diferentes ajustes a través de interfaces visuales, las opciones para adaptarse a cambios importantes o radicales son limitadas. La estructura predefinida y los componentes estandarizados limitan la personalización, lo que podría convertirse en una desventaja en proyectos que requieren cambios continuos.

- *No-code* se clasificó como limitado. Esta tecnología es aún más limitada en cuanto a adaptabilidad, dado que los usuarios dependen

completamente de las herramientas visuales y las configuraciones predefinidas que otorgue el proveedor. Aunque es excelente por la simplicidad y la rapidez, tiene dificultad para adaptarse a cambios complejos o personalizados ya que no permite la intervención a través de código.

- La programación convencional se le otorgo una clasificación de cumple. Ya que es la opción más flexible en este aspecto, no se encuentra restringido por algún tipo de plantilla, ni por componentes predefinidos. Permite realizar ajustes en el código, lo que es ideal para proyectos que requieran adaptaciones continuamente y cambios dentro de la estructura.

#### 11.2.2.2. Modularidad

- *Low-code* se clasifica como cumple. Permite a los desarrolladores la creación de módulos reutilizables, lo que facilita el desarrollo de las aplicaciones de forma rápida y eficiente. Es una ventaja cuando se trabaja en proyectos con elementos comunes o repetidos que pueden reutilizarse en diferentes partes del sistema.

- *No-code* se le dio una clasificación de cumple. A pesar de que otorga una menor flexibilidad para personalizar los módulos, esto permite crear aplicaciones modulares con bloques predefinidos, lo que ayuda a escalar y a replicar funciones de manera eficiente.

- La programación convencional se otorgó una clasificación de cumple también ofrece una estructura modular, con la ventaja de que los desarrolladores tienen control sobre cómo se crean, configuran y

reutilizan los módulos. Es la opción más completa para crear soluciones altamente personalizadas y modulares, aunque genere un costo mayor de complejidad y tiempo de desarrollo.

#### 11.2.2.3. Configurable sin programación

- *Low-code* cumple. Estas plataformas están diseñadas para facilitar el desarrollo sin tener mucha intervención de programación, que permite a los usuarios configurar aplicaciones usando interfaces gráficas. Aunque algunas personalizaciones más avanzadas pueden requerir conocimientos básicos de programación, la mayor parte del desarrollo puede hacerse sin codificar.

- *No-code* cumple. Está diseñado para los usuarios sin habilidades técnicas, permitiendo que la mayoría de las configuraciones se realicen sin necesidad de escribir una sola línea de código. Todo se realiza a través de interfaces visuales, lo que vuelve a las plataformas *no-code* de configuración accesible para usuarios sin experiencia en la programación.

- La programación convencional no cumple, requiere un conocimiento técnico considerable, ya que cualquier ajuste, cambio o configuración debe hacerse programando. Esto limita su accesibilidad a usuarios no técnicos, quienes dependerán completamente de desarrolladores para realizar cualquier cambio dentro del sistema.

### 11.2.3. Personalización

Se presenta el análisis correspondiente a la métrica personalización, con el fin de comparar la capacidad de adaptación de cada tecnología evaluada. Los siguientes resultados muestran las diferencias entre los enfoques.

#### 11.2.3.1. Configuración y parametrización

- Todas las metodologías ofrecen la capacidad de configuración y parametrización de interfaces en mayor o menor medida, estando limitadas únicamente por la herramienta o plataforma utilizada. Cabe destacar que la programación tradicional ofrece el nivel más alto de configuración posible.

#### 11.2.3.2. Desarrollo de interfaces

- *No-code y low-code*: estas plataformas ofrecen sistemas de arrastrar y soltar (*drag and drop*) y plantillas prediseñadas para crear interfaces de manera intuitiva y rápida. Sin embargo, una desventaja significativa es que estas interfaces pueden verse limitadas por las opciones de personalización que proveen las herramientas de desarrollo.

- Programación convencional: posee una gran capacidad de personalización y adaptación para el diseño de interfaces de usuario escalables y reutilizables. Sin embargo, su principal desventaja radica en que la mayoría de estas interfaces necesitan codificarse desde cero,

lo que hace que su desarrollo sea más complejo y requiera mayor tiempo.

#### 11.2.3.3. Flexibilidad de diseño

- *Low-code y no-code:* estas metodologías ofrecen un nivel de funcionalidad que cumple con los requisitos de diseño más comunes en la actualidad. Sin embargo, su principal desventaja es que, si se requiere un diseño muy específico o se desean utilizar estilos particulares, es probable que las herramientas no proporcionen el soporte necesario para implementarlos.

- Desarrollo tradicional: la programación convencional proporciona todas las características necesarias para brindar flexibilidad en los diseños de aplicaciones web. Sin embargo, como en otros casos, su implementación está limitada por las capacidades técnicas del desarrollador y la complejidad de su utilización.

#### 11.2.3.4. Reutilización de componentes

- *Low-code y no-code*: este tipo de herramientas o plataformas permite crear módulos y plantillas que reutilizan componentes para desarrollar aplicaciones. Una de sus principales ventajas es la facilidad y velocidad con la que se pueden crear los desarrollos.

- Desarrollo tradicional: la programación tradicional cuenta con una gran capacidad para reutilizar componentes que se adaptan mejor a las necesidades de los requerimientos. Sin embargo, su principal desventaja es que, si no existe un componente que se ajuste a las

necesidades específicas, el desarrollo de uno nuevo conlleva un tiempo considerable.

### 11.2.4. Integración

A continuación, se detalla el análisis para cada aspecto de la métrica Integración para cada una de las diferentes tecnologías.

#### 11.2.4.1. Compatibilidad con otros sistemas

- *Low-code* se clasifica como limitado. Las plataformas que se enfocan en este tipo de tecnologías suelen tener capacidades de integrarse con diferentes sistemas u otros tipos de tecnologías sin embargo estas integraciones se encuentran limitadas, ya que dependen de las APIs y módulos que ofrece la plataforma. Aunque permita compatibilidad de otros sistemas, está sujeta a las herramientas integradas que proporciona la misma plataforma, que en cierto momento puede llegar a restringir la interconectividad con sistemas más complejos o personalizados.

- *No-code* se clasifica como limitado. Las herramientas *no-code* llegan a ser más simples que las plataformas *low-code,* limitando aún más la capacidad de integración. Normalmente permiten integración con otros sistemas mediante conectores predefinidos o integraciones básicas, pero no ofrecen mucha flexibilidad para adaptarse a sistemas complejos o personalizados.

- La programación convencional se clasifica como cumple, ya que ofrece mayor compatibilidad, permite un desarrollo personalizado y flexible.

Los programadores pueden implementar las integraciones necesarias para interoperar con diferentes sistemas, teniendo el control total sobre las conexiones y la comunicación entre sistemas. Siendo una característica clave para poder trabajar ambientes empresariales con infraestructuras heterogéneas.

#### 11.2.4.2. Facilidad de integración

- *Low-code* y *no-code* son tecnologías que resaltan por ofrecer una integración fácil, lo que significa que facilitan la conexión entre diferentes servicios o sistemas sin necesidad de esfuerzo mediante programación, procesos engorrosos y tediosos. Este tipo de soluciones ofrecen conectores preconstruidos o flujos visuales lo que simplifica la integración con otras herramientas o sistemas.

- La programación convencional se considera compleja en los términos de integración. Dado que es necesario realizar un esfuerzo técnico mayor, ya que las conexiones deben de implementarse a medida y no existen herramientas predefinidas por terceros para facilitar dicha tarea. Sin embargo, aunque el proceso de configuración e implementación se torne complejo, ofrece una mayor personalización y control sobre las integraciones.

#### 11.2.4.3. Consumo de recursos externos

- *Low-code* es limitado, permiten la interacción con APIs y el consumo de recursos externos, sin embargo, estas herramientas se encuentran limitadas a los conectores que proporciona la plataforma. Si bien es

posible realizar integraciones básicas, la capacidad de personalizar o gestionar recursos externos complejos puede verse restringida.

- *No-code* es limitado, estas están diseñadas para poder simplificar el proceso de desarrollo, pero puede implicar restricciones cundo se trata de consumir APIS externas de forma compleja.

- La programación convencional cumple, ya que ofrece un mayor control y flexibilidad acerca del consumo de cualquier recurso externo, permitiendo a los programadores la implementación de cualquier API o servicio sin ninguna limitación, permitiendo una mayor capacidad para consumir datos y una gestión personalizada, útil en aplicaciones web que dependen de la comunicación de múltiples servicios externos.

### 11.2.5. Facilidad de uso

Respecto a la facilidad de uso, se analizó el desempeño de las tecnologías en entornos con distintos niveles de experiencia técnica. Los hallazgos se presentan a continuación.

#### 11.2.5.1. Curva de aprendizaje

- *No-code:* es la mejor opción para equipos de trabajo con muy poco o nulos conocimientos técnico, permite desarrollar soluciones en cortos periodos de tiempo y suelen ser herramientas intuitivas y fáciles de aprender.

- *Low-code:* ideal para desarrolladores con nociones de desarrollo bajas, permite desarrollar soluciones en moderados lapsos de tiempo y para

este tipo de herramientas o plataformas es necesario hacer uso de la documentación y tutoriales para poder utilizar de manera adecua estas herramientas.

- Programación convencional: es necesario contar con un equipo de trabajo el cual tenga los conocimientos técnicos adecuados para desarrollar aplicaciones, en cuanto al tiempo de desarrollo, depende de la complejidad de los requerimientos, pero estos suelen llevar planificaciones para lapsos de tiempo prolongados.

#### 11.2.5.2. Experiencia de usuario

- *Low-code* y *no-code:* las herramientas proporcionan interfaces intuitivas que facilitan el diseño y la implementación de funcionalidad y agilizan el desarrollo por medio del uso de plantilla y módulos preestablecidas que agilizan tiempos.

- Desarrollo tradicional: es necesario el conocimiento técnico adecuado para el manejo de las herramientas de desarrollo, así como también para el uso del lenguaje de programación. No todos los lenguajes y herramientas proporcionan documentación adecuada para el manejo de estas herramientas lo que es un poco desalentador y confuso.

#### 11.2.5.3. Satisfacción del usuario

- *Low-code* y *no-code:* regularmente este tipo de herramientas cuentan con interfaces de usuario simples que facilitan su utilización lo que promueve el aprendizaje y genera una percepción de realización por parte de los desarrolladores.

- Programación convencional: hablando en términos generales la programación convencional es poco amigable con el desarrollador, la forma de trabajar de estas tecnologías no facilita el desarrollo de aplicaciones además de que no promueve el interés por el aprendizaje.

### 11.2.6. Soporte

El análisis de resultados para la métrica Soporte es el siguiente:

#### 11.2.6.1. Disponibilidad

- *Low-code* y *no-code* cuentan con una disponibilidad limitada. Esto significa que es posible acceder al soporte, pero la disponibilidad puede ser restringida, ya sea en términos de horario, frecuencia o acceso a expertos. Los usuarios podrían no tener acceso a soporte inmediatamente o continua cuando lo necesiten.

- La programación convencional cumple, ya que el soporte es más sólido ya que cuenta con documentación a la cual se puede acceder en cualquier momento otorgando mayor flexibilidad al usuario.

#### 11.2.6.2. Calidad de soporte

*Low-code, no-code* y la programación convencional se califican con cumple, ya que el soporte que se otorga es satisfactorio. Ofreciendo soluciones técnicas mediante documentación, tutoriales tanto de terceros como de los sitios oficiales. No existe una diferencia significativa de calidad en el soporte ofrecido por las tres tecnologías.

#### 11.2.6.3. Canales de soporte

- *Low-code* y *no-code* son limitados, los canales que pueden utilizar los usuarios para solventar sus preguntas son reducidas por ejemplo solo correo electrónico, chat. Esto podría presentar complicaciones a los usuarios para elegir el canal de comunicación correcto o dificultades para obtener respuesta.

- La programación convencional cumple, ya que ofrece canales de comunicación extensos, permitiendo acceder a diferente información no únicamente a través de canales oficiales que se proporcionan por la librería o software que se está utilizando si no también mediante tutoriales proporcionados por terceros, ya sea por video tutoriales, blogs, páginas web, apoyo de la comunidad de desarrolladores, entre otros.

#### 11.2.6.4. Tiempo de respuesta

En este aspecto las tecnologías *low-code, no-code* y la programación convencional se clasifican como limitados, indicando que, aunque exista el soporte el tiempo de espera para recibir una respuesta puede ser prolongado o no tan rápido como los usuarios lo requieren, afectando la productividad si estos problemas no se abordan con brevedad.

#### 11.2.6.5. Documentación

- *Low-code* y *no-code* son limitados, ya que la mayoría de la documentación se encuentra únicamente en la página oficial de la plataforma que se desee utilizar, es decir, aunque se navegue por

internet buscando alguna fuente diferente no es posible encontrar una diferente a la documentación oficial.

- La programación convencional cumple, esto porque no únicamente se cuenta con la documentación proporcionada por el implementador. Al ser tecnologías, librerías o software se cuenta con el apoyo de comunidades que suelen utilizar todas las tecnologías que entran en esta categoría. Extendiendo la cantidad de recursos no únicamente a la página oficial de la plataforma si no a video tutoriales, *blogs* entre otros.

#### 11.2.7. Mantenimiento

Como parte del análisis de la métrica de mantenimiento se tomó en cuenta la facilidad de las herramientas para realizar el diagnóstico y depuración de errores, así como también la facilidad para proporcionar soluciones correctas por medio pruebas y la adaptabilidad a cambios y lanzamiento de nuevas versiones.

##### 11.2.7.1. Facilidad de depuración

- *No-code* y *low-code:* las plataformas o herramientas suelen proporcionar poca información relacionada con depuración esto depende de la plataforma, los mensajes de depuración logs suelen ser simples, lo que ayudan a corregir problemas simples, pero no ayuda demasiado en problemas lógicos y complejos.

- Programación convencional: la programación convencional ofrece múltiples métodos de depuración que permiten una depuración especializada a bajo nivel.

#### 11.2.7.2. Facilidad de diagnostico

- *No-code* y *low-code:* la mayoría de las plataformas ofrece poca información del estado de las aplicaciones lo que dificulta las tareas de diagnóstico y corrección de errores, regularmente estas se limitan a logs simples que no proporcionan información clara.

- Programación convencional: existen herramientas que permiten monitorear en tiempo real las aplicaciones con logs claros que permiten conocer el estado actual de las aplicaciones y monitorear no solo si ocurrieron errores si no también el consumo de recursos.

#### 11.2.7.3. Control de versiones

- *No-code* y *low-code:* principalmente las herramientas permiten un muy bajo nivel de control para versiones, principalmente se limitando a revertir cambios en un historial de cambios.

- Programación convencional: permite un completo control de versiones por medio de la utilización de herramientas específicas para controlar flujos de desarrollo complejos.

#### 11.2.7.4. Cobertura de pruebas

- *No-code* y *low-code:* ofrece pocas opciones de pruebas, estas se limitan a la herramienta, regularmente se trata de pruebas de integración y unitarias en ciertos componentes.

- Programación convencional: permite total integración con herramientas específicas para pruebas automatizadas de todo tipo y brinda reportes detallados de las pruebas realizadas.

### 11.2.8. Costos

El análisis de resultados para la métrica de Costos de implementación por tecnología es el siguiente:

#### 11.2.8.1. Costo inicial

- *Low-code* y *no-code* presentan un costo inicial bajo, lo que es beneficioso para empresas o proyectos con presupuestos limitados. Esto se debe a que no requieren inversiones en desarrolladores o infraestructura costosas al inicio.

- En cuanto a la programación convencional el costo inicial de cualquier proyecto es elevado, con lleva la contratación de desarrolladores especializados, infraestructura, compra de hardware, licencias entre otras cosas que serán necesarias para la elaboración del proyecto.

#### 11.2.8.2. Costo de operación:

- *Low-code* y *no-code* tienen costos de operación medios, lo cual indica que, aunque no es tan costoso como la programación convencional, requiere gastos de mantenimiento, para mejoras continuas, licencias de plataforma o soporte.

- En la programación convencional el costo de operación es alto, estos costos se asocian al mantenimiento de los sistemas, el soporte continuo, actualizaciones y así como la necesidad de contar con especialistas que se encarguen de controlar estos sistemas.

#### 11.2.8.3. Costo de integración

- *Low-code* tiene un costo de integración bajo, estas plataformas están diseñadas para facilitar la integración con aplicaciones y servicios a través de componentes o conectores predefinidos que reducen el esfuerzo del desarrollo.

- *No-code* tiene un costo de integración medio, ya que también ofrece opciones simplificada de integración, este puede ser más limitado en su capacidad para conectarse a sistemas más complejos o personalizados.

- El costo de integración para una aplicación con tecnologías convencional presenta un costo alto, ya que las soluciones suelen ser personalizadas consumiendo más tiempo, esfuerzo y recursos para lograr una integración adecuada con otras aplicaciones o plataformas.

#### 11.2.8.4. Costo de escalabilidad

Las tres tecnologías presentan altos costos de escalabilidad. Esto porque a medida que la solución crece en complejidad o volumen de usuarios, los recursos necesarios como infraestructura, licencia, desarrollo aumenta significativamente en cada una de ellas. Para las tecnologías *low-code* y *no-code* cuentan con limitaciones, por lo que si requiere una mejora en cuanto

a recursos será necesario una mejora del plan o licencia a la que se encuentra suscrito. En el caso de la programación convencional, la escalabilidad puede requerir mayores esfuerzos de desarrollo e infraestructura personalizada.

### 11.2.9. Escalabilidad horizontal y vertical

Para analizar los distintos aspectos relacionados con la escalabilidad se tomaron en cuenta las distintas maneras en cómo puede evolucionar una arquitectura, la forma en que se pueden gestionar los recursos, cual es nivel de monitoreo de las soluciones y optimización de recursos.

#### 11.2.9.1. Escalabilidad horizontal

- *No-code* y *low-code:* muy limitada por las plataformas permite algún nivel de distribución de cargas el cual se gestiona directamente por el proveedor, y no soporta arquitecturas distribuidas.

- Programación convencional: permite diseñar arquitecturas distribuidas complejas. Existen herramientas específicas las cuales distribuir la carga y regulan el rendimiento de las arquitecturas.

#### 11.2.9.2. Escalabilidad vertical

- *No-code* y *low-code*: permiten un nivel de escalado bajo el cual se limita por las opciones que proporciona la plataforma, una de la desventaja es que no es posible optimizar ni configurar la manera en la que se escalan las aplicaciones.

- Programación convencional: permite control total sobre la forma en que se configuran los recursos de las aplicaciones en los servidores o en la nube y permite bajar costos por medio de configuraciones avanzadas y detalladas.

#### 11.2.9.3. Monitoreo y gestión de recursos

- *No-code* y *low-code:* ofrece pocas opciones de monitoreo generalmente estas limitadas a logs y notificaciones simples los cuales poseen pocas opciones para gestionar recursos.

- Programación convencional: ofrece integración compleja con herramientas personalizadas y configurables capaces de gestionar los recursos. Permiten configurar alertas y ver el estatus de los recursos en tiempo real.

#### 11.2.9.4. Optimización de recursos

- *No-code* y *low-code:* ofrece pocas o nulas opciones para gestionar los recursos, estas se limitas a las opciones que ofrecen las herramientas y las practicas empleadas por parte de los desarrolladores de las plataformas.

- Programación convencional: permite optimizar los recursos a nivel de código, configuración e infraestructura, esto permite ahorras costos y consumo energético. Adicionalmente permite el uso de estrategias específicas que se adaptas a los diferentes tipos de aplicaciones.

### 11.2.10. Seguridad

El análisis que califica los aspectos principales de la métrica seguridad es el siguiente:

#### 11.2.10.1. Protección de datos

- *Low-code* y *no-code* son limitados, ya que ambas tecnologías ofrecen un nivel básico de protección de datos, lo que significa que no son completamente robustos en este aspecto. Es decir, pueden tener medidas de protección, pero no ser suficientes para escenarios en donde se requiera un alto nivel de seguridad, como replica de datos o protección ante desastres.

- La programación convencional cumple, es más robusta y completa. Permite a los desarrolladores implementar métodos de seguridad personalizados y adaptados a las necesidades específicas del proyecto. Esto garantiza una mayor protección de la integridad en diversos contextos y escenarios.

#### 11.2.10.2. Cifrado

- *Low-code* y *no-code* no cumple, ninguna de estas tecnologías implementa el cifrado de manera adecuada o absoluta, lo que representa un riesgo considerable cuando se trata de proteger datos sensibles. Es decir, toda la parte del cifrado se realiza por parte del implementador sin especificar claramente o de una forma técnica que cifrado se aplica sobre los datos.

- La programación convencional cumple, se manejan los mejores cifrados utilizando algoritmos modernos para asegurar la protección de los datos durante su transmisión y almacenamiento.

#### 11.2.10.3. Autenticación

La autenticación en los tres tipos de tecnologías estudiados cumple, cuentan con los mecanismos de seguridad adecuados, incluyendo opciones como autenticación multifactor (MFA) o protocolos seguros para proteger el acceso no autorizado.

#### 11.2.10.4. Gestión de vulnerabilidades

- *Low-code* y *no-code* no cumple. En estas tecnologías no se aplican medidas adecuadas para gestionar las vulnerabilidades. Esto implica que no existen o son insuficientes las herramientas para detectar, mitigar y prevenir brechas de seguridad, lo que aumenta el riesgo de ataques o fallos en la protección.

- La programación convencional cumple, ya que se implementan mecanismos solidos para la gestión de las vulnerabilidades, lo que permite tomar medidas preventivas y correctivas para mantener la seguridad del sistema frente a posibles amenazas.

# CONCLUSIONES

1. En la actualidad, las pequeñas empresas buscan optimizar sus operaciones mediante soluciones tecnológicas que aumenten la eficiencia y reduzcan la dependencia de procesos manuales. La transformación digital permite automatizar tareas, eliminar el uso excesivo de papel y modernizar flujos de trabajo anticuados, facilitando la adopción de herramientas de software accesibles y fáciles de usar. Entre las principales necesidades tecnológicas de las pequeñas empresas destacan la automatización de procesos, la gestión eficiente de recursos y la integración de plataformas que mejoren la productividad y la comunicación.

2. Las plataformas *low-code* y *no-code* permiten la creación de soluciones de software de una forma rápida y accesible, facilitando a los usuarios la implementación de reglas del negocio sin necesidad de experiencia técnica avanzada. Estas tecnologías ofrecen interfaces graficas simples e intuitiva que eliminan la complejidad asociada a la programación convencional. A diferencia de las herramientas tradicionales, las plataformas low-code y no-code disminuyen la necesidad de grandes equipos de trabajo, reduciendo gastos en producción y mantenimiento de desarrollo de software.

3. Las herramientas de desarrollo *low-code* y *no-code* simplifican significativamente la creación de soluciones web, permitiendo a los usuarios diseñar y lanzar aplicaciones sin necesidad de conocimientos técnicos especializados. Estas plataformas agilizan los tiempos de

desarrollo al ofrecer plantillas y componentes preconfigurados, lo que reduce la dependencia de equipos de desarrollo tradicionales. Además, facilitan la iteración rápida, permitiendo ajustes y mejoras en tiempo real, lo que aumenta la eficiencia y reduce costos. Al simplificar el desarrollo, las herramientas *low-code* y *no-code* permiten a las empresas adaptarse rápidamente a los cambios del mercado y acelerar la implementación de soluciones web.

4. Después de analizar las tecnologías *low-code, no-code* y la programación convencional en relación con las necesidades de las pequeñas empresas, se recomienda el uso de plataformas *no-code.* Dado que las pequeñas empresas generalmente no cuentan con áreas especializadas en desarrollo de *software*, las plataformas *no-code* son ideales, ya que permiten crear aplicaciones de manera sencilla sin necesidad de conocimientos técnicos avanzados. Aunque la programación convencional ofrece mayor personalización, su complejidad y los recursos necesarios la hacen menos adecuada para este tipo de empresas.

   Las herramientas *no-code,* en cambio, permiten a las pequeñas empresas implementar rápidamente soluciones funcionales y adaptadas a su negocio, sin la necesidad de un equipo técnico especializado, lo que las convierte en la opción más accesible y eficiente.

# RECOMENDACIONES

1. Realizar un análisis para determinar cuál de los procesos manuales actuales, al digitalizarse mediante una herramienta *no-code*, generaría el mayor beneficio en términos de eficiencia, ahorro de costos y mejora en la productividad. Priorizar aquellos procesos que sean repetitivos, consuman tiempo y recursos, o afecten la calidad de servicio.

2. Al seleccionar una herramienta para la transformación digital es recomendable priorizar herramientas con planes flexibles y que permitan realizar pruebas gratuitas o demos para asegurar que se adapten al flujo de la empresa sin generar complicaciones o costos innecesarios. Así mismo considerar la facilidad de uso, soporte técnico y costos a mediano y largo plazo.

3. La seguridad debe gestionarse proactivamente configurando adecuadamente permisos y roles, implementando validación de datos para proteger contra entradas maliciosas, auditando regularmente el acceso a tus aplicaciones, y comprendiendo qué aspectos de seguridad gestiona la plataforma y cuáles son tu responsabilidad.

4. La documentación efectiva para proyectos *low-code* y *no-code* debe ir más allá de los aspectos técnicos, abarcando decisiones de diseño, lógica de negocio y flujos de trabajo visuales con abundantes elementos gráficos como capturas de pantalla y diagramas. Es esencial establecer un sistema de documentación dinámico y colaborativo que evolucione con la aplicación, incluyendo un glosario de términos, registro de

problemas resueltos, limitaciones identificadas y soluciones alternativas implementadas.

# REFERENCIAS


Amazon Web Services. (s.f.). *¿Qué es la codificación manual mínima? - Explicación sobre la codificación manual mínima*. Amazon Web Services, Inc. https://aws.amazon.com/es/what-is/low-code/

Amazon Web Services. (s.f.). *¿Qué es la transformación digital? - Explicación de la transformación digital*. Amazon Web Services, Inc. https://aws.amazon.com/es/what-is/digital-transformation/#:~:text=La%20transformaci%C3%B3n%20digital%20es%20el,ofrece%20valor%20a%20los%20clientes

Amazon Web Services. (s.f.). *Desarrollador - Capacitación digital y presencial*. Amazon Web Services, Inc. https://aws.amazon.com/es/training/learn-about/developer/

Amazon Web Services. (s.f.-a). *¿Qué es una aplicación web? - Explicación de las aplicaciones web*. Amazon Web Services, Inc. https://aws.amazon.com/es/what-is/web-application/

Amazon Web Services. (s.f.-b). *¿Qué es una aplicación web? - Explicación de las aplicaciones web*. Amazon Web Services, Inc. https://aws.amazon.com/es/what-is/web-application/

Ashok, A. (13 de mayo de 2021). *Council Post: Low-Code and No-Code in 2021: ¿Are they as useful as they seem?* [Publicación del Consejo: Low-Code y No-Code en 2021: ¿Son tan útiles como parecen?]. https://www.forbes.com/sites/forbestechcouncil/2021/05/13/low-code-

and-no-code-in-2021-are-they-as-useful-as-they-seem/?sh=6fc6b2d4160f

International Business Machines. (20 de marzo de 2023). *¿Qué es la arquitectura de tres niveles?* https://www.ibm.com/mx-es/topics/three-tier-architecture

Korpi. (20 de agosto de 2024). *Qué es No Code, el movimiento que va a transformar el desarrollo de software*. https://platzi.com/blog/no-code-learning-path/

Lores, F. (2024). *¿Qué es una plataforma low-code?* Velneo. https://www.velneo.com/blog/que-es-plataforma-lowcode

No Code Family. (s.f.-b). *Best No-Code Backend Tools in 2024* [Las mejores herramientas backend sin código de 2024]. https://nocodefamily.com/tools/backend

Oracle Cloud Infrastructure. (s.f.). *¿Qué es una base de datos?* Oracle México. https://www.oracle.com/mx/database/what-is-database/

Red Hat. (2 de junio de 2022). *¿What is an API?* https://www.redhat.com/en/topics/api/what-are-application-programming-interfaces

SAP. (s.f.). *Low-code/no-code: el futuro del desarrollo*. SAP. https://www.sap.com/products/technology-platform/build/what-is-low-code-no-code.html

Tic Portal. (26 de septiembre de 2022). *Low-Code (de poco código)*. https://www.ticportal.es/glosario-tic/low-code-poco-codigo