INSTITUTO TECNOLÓGICO

SUPERIOR DE GUASAVE

TESIS

“ANÁLISIS EN LA MEJORA DEL TIEMPO Y DISMINUCIÓN DE ERRORES PARA PROCESOS DE MANIPULACIÓN DE DATOS EN EL ÁREA DE TRÁFICO APLICANDO UN SISTEMA DE LOCALIZACIÓN ASISTIDO POR ASISTENTES DE VOZ.”

QUE PRESENTA

AXEL FRANCISCO ASTORGA AGUILAZOCHO

NÚMERO DE CONTROL

1825010202

PARA OBTENER EL GRADO DE

INGENIERO EN SISTEMAS COMPUTACIONALES

ASESOR

ING. GRACIELA LUGO RUBIO

EMPRESA:

INSTITUTO TECNOLÓGICO SUPERIOR DE GUASAVE

GUASAVE, SINALOA. FEBRERO, 2023

Instituto Tecnológico Superior de Guasave	**LIBERACIÓN DE PROYECTO PARA TITULACIÓN INTEGRAL PLAN 2015**	Responsable: Jefatura de División de Ciencias
	Referencia a la Norma:ISO 9001:2015 8.1;8.2.1;8.2.2;8.2.3;8.5.1;8.5.2;8.6;8.7.2 Referencia a la Norma:ISO 21001:2018 8.1;8.1.1;8.1.2;8.2.1;8.2.2;8.2.3;8.3.3;8.3.4;8.6;8.7;9.1.2	Código: ITSG-SIG-AO-PO-19-03
		Revisión: 4
		Fecha de Emisión: Agosto 2022
		Página 1 de 1

Guasave, Sinaloa, Fecha: 03/02/2023
ASUNTO: Liberación de proyecto para titulación integral

ING. LAURA BEATRIZ INZUNZA RAMÍREZ
Coordinador(a) de Titulación
PRESENTE

Por este medio informo que ha sido liberado el siguiente proyecto para la titulación integral:

Nombre del estudiante y/o egresado(a):	Axel Francisco Astorga Aguilazocho
Carrera:	Ingeniería en Sistemas Computacionales
Número de control:	1825010202
Nombre del proyecto:	Análisis en la mejora del tiempo y disminución de errores para procesos de manipulación de datos en el área tráfico aplicando un sistema de localización asistido por asistentes de voz.
Producto:	Tesis

Agradezco de antemano su valioso apoyo en esta importante actividad para la formación profesional de nuestros(as) egresados(as).

ATENTAMENTE

ING. MANUEL ALFREDO FLORES ROSALES.

Nombre y firma del (de la) jefe(a) de División de Ciencias Computacionales.

Graciela Lugo
ING. Graciela Lugo Rubio
Nombre y firma del Asesor(a)

ING. José Esteban Peral Huizar
Nombre y firma del Revisor(a)

Arce Cardenas Fco. ?
MC. Francisco Javier Arce Cárdenas
Nombre y firma del Revisor(a)

AGRADECIMIENTOS

Antes que, a cualquier otra persona, me gustaría agradecer a mi familia, por el gran apoyo brindado, tanto en el aspecto emocional, como económico, por ayudarme ante cualquier necesidad de recursos, por siempre estar ahí en cada momento que se requería, porque a pesar de la distancia nunca me falto nada de ellos, nunca terminaré de agradecerles todo lo que hicieron por mí.

Agradezco a mis compañeros de carrera, a mis amigos, por regalar tan buenos momentos y lograr que estos años hayan sido amenos y llenos de aprendizajes. También agradezco a todos los docentes que han sido parte de mi formación, y que en ellos encontré a buenos amigos.

RESUMEN

El presente trabajo, tiene como propósito apoyar a la empresa CMC Transporta ubicados en la ciudad de Guasave, Sinaloa. El cual consiste en analizar los procesos operativos en el área de Tráfico actuales de la empresa, ya que esta no cuenta con un sistema informático adecuado para un proceso ágil del departamento lo que significa que de manera manual toda la información es registrada en una aplicación de escritorio lo que genera un problema al momento de hacer consultas o registrar nueva información, ya que suelen cometerse errores y complicar el actualizar la información solicitada, con una API REST hecha a medida, es posible guardar información que no ocupara espacio físico mejorando la forma de organizar y almacenar la información, el simple hecho de no depender técnicamente del capital humano puede ayudar a que la empresa mejore potencialmente los procesos que realiza. Se puede acelerar el proceso de consultas rápidas a los registros de los remolques o estadísticas generadas por el sistema de tráfico.

Palabras clave: API REST, capital humano, procesos, consultas, registros, remolques, sistema de tráfico.

ABSTRACT

The elaboration of the project is to support the company CMC Transport located in the city of Guasave, Sinaloa. Which consists of analyzing the operational processes in the current Traffic area of the company, since the company does not have an adequate computer system for an agile process of the department, which means that all the information is manually registered in an application desktop, which generates a problem when making queries or registering new information, since errors are often made and complicate updating the requested information, with a custom-made REST API, it is possible to save information that does not occupy physical space by improving the way to organize and store information, the simple fact of not technically depending on human capital can help the company potentially improve the processes it performs. You can speed up the process of quick queries to trailer records or statistics generated by the traffic system.

Keywords: REST API, human capital, processes, queries, records, trailers, traffic system.

ÍNDICE

INTRODUCCIÓN 1
I PLANTEAMIENTO DEL PROBLEMA 2
II OBJETIVOS 3
2.1 Objetivo general: 3
2.2 Objetivos específicos: 3
III JUSTIFICACIÓN 4
IV MARCO TEÓRICO 5
4.1 Bases de datos 5
4.1.1 Orígenes de las bases de datos. 5
4.1.2 Tipos de bases de datos. 6
4.1.3 Importancias de las bases de datos en las empresas. 18
4.2 Sistemas Gestores de Bases de Datos. 19
4.2.1 Evolución de los Sistemas Gestores de Bases de Datos. 20
4.2.2 Mejores gestores de bases de datos a lo largo de la historia. 22
4.3 API 30
4.3.1 Principios de diseño de REST 32
4.3.2 Funcionamiento de las API 34
4.4 Tipos de Api 36
4.5 Aplicaciones para desarrollar API REST 38
4.6 Asistentes de voz inteligentes 41
4.6.1 Tipos de asistentes de voz 42
V MATERIALES Y MÉTODOS 48
5.1 Modalidad de Investigación 48
5.1.1 Bibliográfica 48
5.1.2 Aplicada 48
5.2 Análisis del sistema de monitoreo de remolques. 48
5.3 Problemáticas detectadas 49
5.4 Creación de Api rest 49
5.4.1 Estructura de datos 50
5.4.2 GETS 50

5.4.3 Modificar 54
5.5 Creación de skill 56
5.5.1 Creación de Invocations - Intents. 57
5.5.2 Programación de Inicio de sesión. 59
5.5.3 Programación de modificar el patio o actualizar la información. 61
5.5.4 Programación de localización de un remolque de carga. 62
VI ANÁLISIS DE RESULTADOS Y DISCUSIÓN. 64
6.1 Pizarra digital 64
6.2 Software 64
6.3 Evaluación de aplicaciones 65
6.3.1 Asignación de viajes 65
6.3.2 Localización 68
6.3.1 Cambio de patio y actualización de información 69
VII CONCLUSIONES Y RECOMENDACIONES. 74
7.1 Recomendaciones: 74
7.2 COMPETENCIAS DESARROLLADAS 75
VIII REFERENCIAS. 77
IX Anexo 79
9.1 Subir API REST A Soome 79
9.1.1 Ingresar a Soome 79
9.1.2 Publicar nuestra aplicación 85
9.1.3 Subir la Aplicación publicada al hosting. 91

ÍNDICE DE FIGURAS

Figura 1 Base de datos relacional. 7
Figura 2 Ejemplo de una base de datos orientadas a objetos. 8
Figura 3 Bases de datos distribuidas 9
Figura 4 Bases de datos NoSQL. 10
Figura 5 Bases de datos orientadas a grafos. 11
Figura 6 Bases de datos de código abierto. 12
Figura 7 Bases de datos de código abierto. 13
Figura 8 Bases de datos en la nube. 15
Figura 9 Bases de datos en la nube. 16
Figura 10 Bases de datos de documentos/JSON. 17
Figura 11 Niveles de abstracción de la arquitectura ANSI. 20
Figura 12 Logo de MySQL 22
Figura 13 Logo de MariaDB. 23
Figura 14 Logo de SQLite 24
Figura 15 Logo de PostgreSQL. 25
Figura 16 Logo de Microsoft SQL Server. 26
Figura 17 Logo de Oracle. 27
Figura 18 Logo de MongoDB. 28
Figura 19 Logo de MongoDB. 29
Figura 20 Logo de Cassandra. 29
Figura 21 Consumo de API REST 36
Figura 22 Express.js 38
Figura 23 Logo de Flask 38
Figura 24 Django Rest Framework. 39
Figura 25 Spring 39
Figura 26 Logo Ruby on Rails. 39
Figura 27 ASP.NET 40
Figura 28 Postman 40
Figura 29 Swagger 41
Figura 30 Loopback 41
Figura 31 Google Assistant. 43
Figura 32 Amazon Alexa. 44
Figura 33 Siri 45
Figura 34 Bixby. 46
Figura 35 Cortana. 47
Figura 36 Mycroft. 47
Figura 37 Diseño de la pizarra 49
Figura 38 Conexión a SQL SERVER 50
Figura 39 Estructura de datos 50

Figura 40 Método GET 51
Figura 41 Método obtener usuario 52
Figura 42 Procedimiento para obtener usuario y contraseña. 53
Figura 43 Llamado a la Api 53
Figura 44 Método de Localización 54
Figura 45 Modificar 55
Figura 46 Estructura de modificar 55
Figura 47 Selección de colores 56
Figura 48 Plataforma de Alexa Developer. 57
Figura 49 Creación de Invocations. 58
Figura 50 Creación de Invocations – Intents 59
Figura 51 Conexión a la API 60
Figura 52 Programación de inicio de sesión 61
Figura 53 programación de modificar patio o actualizarlo 62
Figura 54 Programación de localización 63
Figura 55 Pizarra digital 64
Figura 56 Pantalla de prueba de skill 65
Figura 57 Asignación de camión en pizarra digital 66
Figura 58 Formulario de asignación de viaje en pizarra digital 66
Figura 59 Asignación de viaje 67
Figura 60 asignación de viaje (Parte 2) 67
Figura 61 localización 69
Figura 62 Selección de contenido 70
Figura 63 Seleccionar placa 70
Figura 64 Cambio de patio 71
Figura 65 Cambio de patio (Segunda parte) 72
Figura 66 SOME.COM 80
Figura 67 Free .Net Hosting 80
Figura 68 Plan Gratuito 81
Figura 69 Creación de cuenta 82
Figura 70 Compra del plan gratuito 82
Figura 71 Alojamiento de nuestro Sitio Web 83
Figura 72 Sitio creado 84
Figura 73 Apartado de administración 85
Figura 74 Proyecto que vamos a publicar 86
Figura 75 Carpeta 87
Figura 76 Ubicación de carpeta 88
Figura 77 Publicar 88
Figura 78 Comprobar en la ubicación. 89
Figura 79 Convertir en ZIP 90
Figura 80 Zip 91

Figura 81 Subir la Aplicación publicada al hosting 92
Figura 82 Subir archivo zip..... 92
Figura 83 Descomprimir Zip 93
Figura 84 Vista de archivos descomprimidos..... 93
Figura 85 URL de la API REST 94

INTRODUCCIÓN

La APIS REST han cambiado por completo la ingeniería de software. En la actualidad no existe un proyecto o aplicación que no disponga de una API REST para la creación de servicios profesionales a partir del software. Twitter, youtube, los sistemas de identificación de facebook, hay cientos de empresas que generan negocios gracias a las APIS REST, sin ellas, todo el crecimiento seria prácticamente imposible. Esto es así porque REST es el estándar más lógico, eficiente y habitual en la creación de APIS para servicios de internet.

Por lo cual la mejor opción es crear una API REST a medida para automatizar un proceso el cual tiene reglas específicas, ya que si el capital humano responsable olvida actualizar la información, puede ocasionar un descontrol, además que depende de igual forma de una estricta norma de escritura, que si no se lleva a cabo bajo las reglas la aplicación no reconoce los cambios y esto estropea el funcionamiento correcto e impide continuar realizando cambios, dichos cambios se resuelven directamente con el responsable del mantenimiento de software (servicio externo de la empresa).

I PLANTEAMIENTO DEL PROBLEMA

CMC TRANSPORTA es una empresa de giro de transporte y logística la cual cuenta con más de 50 vehículos y más de 70 remolques que realizan las tareas de llevar día a día la carga de sus clientes a su destino. Para este fin, se cuenta con una "Pizarra digital" que permite controlar la ubicación de los remolques. La desventaja de esta pizarra es que depende de los cambios que realiza el trabajador directamente, si el capital humano responsable olvida actualizar la información, puede ocasionar un descontrol, además que depende de igual forma de una estricta norma de escritura, que si no se lleva a cabo bajo las reglas la aplicación no reconoce los cambios y esto estropea el funcionamiento correcto e impide continuar realizando cambios, dichos cambios se resuelven directamente con el responsable del mantenimiento de software (servicio externo de la empresa).

Por lo tanto, el presente proyecto propone resolver este problema mediante la implementación de una API REST. La ventaja de automatizar este proceso en específico es bastante, Pero mencionando los más importantes son:

La optimización de los recurso y tiempos, ya que actualmente la empresa utiliza una computadora y dos monitores para realizar esta actividad, el ahorro de tiempo se reflejará básicamente en el cambio de interacción del capital humano y la máquina, pues de pasar en un proceso basado en eventos del equipo (clic, doble clic, clic derecho, enter) se aplica un proceso de conversación con los dispositivos utilizados.

Reducción de errores, los sistemas automatizados que utilizan un lenguaje natural tienen la característica de ser en si inteligentes y comprender perfectamente los comando que se requieren ejecutar.

II OBJETIVOS

2.1 Objetivo general:

- Implementar una API REST para la empresa CMC, para disminuir tiempo y errores al consumir y manipular la información de localización de cajas de tráiler, mediante un sistema inteligente asistido por voz Alexa.

2.2 Objetivos específicos:

- Tomar requerimientos para comprender el flujo del proceso que se realiza en el área de tráfico.
- Analizar datos del proceso, obtenidos en el recabado de información en el área de tráfico.
- Implementar prototipo de una API REST.
- Probar el prototipo diseñado.
- Corrección de errores.

III JUSTIFICACIÓN

La incorporación de APIs en el desarrollo de aplicaciones y servicios web se ha convertido en una tendencia creciente debido al aumento exponencial en el número de estos en Internet. La implementación de APIs no solo es una oportunidad para las grandes empresas, sino que cualquier tipo de organización puede descubrir su potencial.

En este contexto, es importante justificar la implementación de APIs en una empresa, especialmente por las ventajas que presenta en comparación con procesos manuales. En este sentido, es fundamental destacar dos aspectos clave: la optimización de los recursos y tiempos, y la reducción de errores.

Por un lado, la automatización del proceso con APIs permite optimizar los recursos y tiempos de la empresa. Actualmente, el proceso requiere una computadora y dos monitores, pero el uso de APIs permitiría una interacción más eficiente entre el capital humano y la máquina, ya que se aplicaría un proceso de conversación en lugar de un proceso basado en eventos.

Por otro lado, la automatización con APIs también permite reducir los errores en el proceso. Los sistemas automatizados que utilizan un lenguaje natural tienen la capacidad de ser inteligentes y comprender perfectamente los comandos que se requieren ejecutar, lo que minimiza la posibilidad de errores humanos.

IV MARCO TEÓRICO

4.1 Bases de datos

Una base de datos es un sistema organizado para almacenar, gestionar y recuperar información. Esta información se almacena en tablas, donde cada tabla tiene un conjunto de campos (o columnas) y registros (o filas), que utilizan para almacenar y gestionar información en un formato estructurado, lo que permite un fácil acceso, actualización y recuperación de la información.

Las bases de datos se pueden utilizar para una variedad de aplicaciones, como el almacenamiento de información de clientes, el seguimiento de inventarios, el procesamiento de transacciones financieras y la recopilación de datos para la toma de decisiones empresariales. Existen varios tipos de bases de datos, como las bases de datos relacionales, las bases de datos no relacionales y las bases de datos en memoria.

4.1.1 Orígenes de las bases de datos.

Los orígenes de las bases de datos se remontan a la necesidad de almacenar y recuperar información de manera eficiente en las organizaciones y empresas. En el siglo XIX y principios del siglo XX, la información se almacenaba en tarjetas perforadas o en papel y se utilizaban sistemas manuales para buscar y recuperar información.

En la década de 1950, se desarrollaron los primeros sistemas automatizados de bases de datos, como el sistema Integrated Data Store (IDS) de IBM. Estos sistemas permitían el almacenamiento y recuperación automatizados de información, pero aún no contaban con un modelo de organización de datos.

En 1970, el matemático Edgar F. Codd propuso el Modelo Relacional de Datos, que es la base de las bases de datos relacionales modernas. Este modelo estableció las reglas para la estructuración de la información en tablas relacionales y para la manipulación de esta información mediante el lenguaje SQL.

En los años 70 y 80 surgieron varias bases de datos relacionales comerciales, tales como Oracle, IBM DB2, y Sybase. Con el surgimiento de la era digital y el big data, las bases de datos no relacionales y las bases de datos en memoria se volvieron cada vez más populares.

4.1.2 Tipos de bases de datos.

Bases de datos relacionales. Una base de datos relacional es un tipo de base de datos en la cual se almacena información en tablas con campos y registros, y estas tablas están relacionadas entre sí mediante claves foráneas, esto se puede ver en la figura 1. Estas bases de datos se basan en el Modelo Relacional de Datos, propuesto por Edgar F. Codd en 1970.

Las tablas de una base de datos relacional contienen campos o columnas, que son los diferentes atributos de un registro o fila. Cada tabla tiene un conjunto de campos únicos llamados clave primaria, que se utilizan para identificar de manera única un registro. Las claves foráneas se utilizan para establecer relaciones entre tablas, permitiendo la integridad referencial.

El lenguaje de consulta estándar para bases de datos relacionales es el SQL (Structured Query Language) el cual permite la manipulación y recuperación de datos de las tablas mediante comandos como SELECT, INSERT, UPDATE, DELETE.

Ejemplos de bases de datos relacionales incluyen MySQL, Oracle, Microsoft SQL Server, PostgreSQL, y SQLite. Estas bases de datos son ampliamente utilizadas en diferentes aplicaciones, incluyendo sistemas de gestión empresarial, sistemas de gestión de contenido y aplicaciones web.

Estas bases de datos almacenan información en tablas con campos y registros. Las tablas están relacionadas entre sí mediante claves foráneas. Ejemplos de bases de datos relacionales incluyen MySQL, Oracle y Microsoft SQL Server.

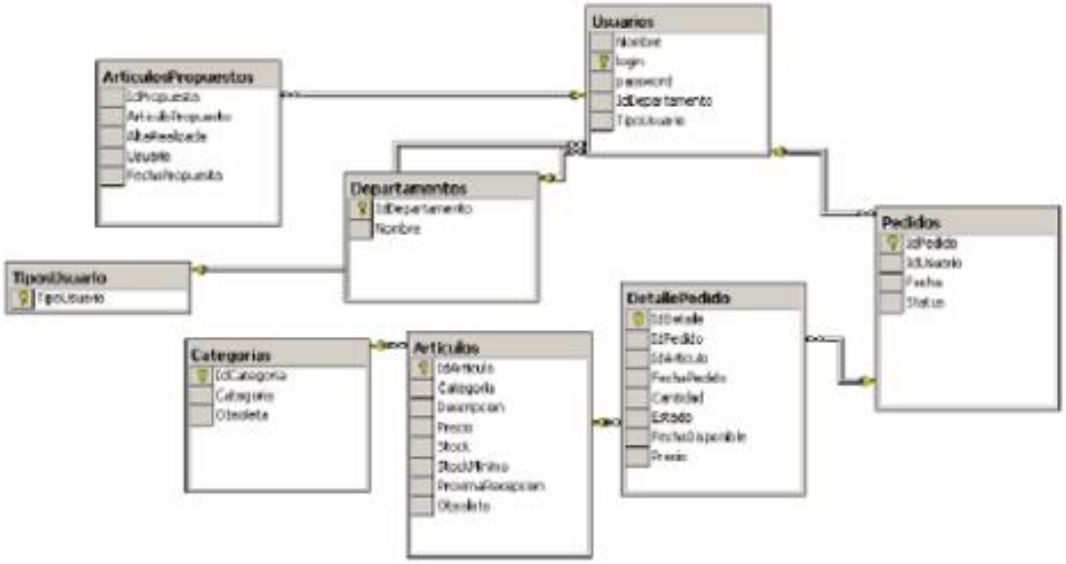


Figura 1 Base de datos relacional.

Bases de datos orientadas a objetos. Las bases de datos orientadas a objetos (OODB) son un tipo de base de datos que utilizan la programación orientada a objetos (POO) para almacenar y recuperar datos. En lugar de utilizar tablas relacionales para almacenar los datos, almacenan los datos en forma de objetos.

Un objeto en una base de datos de este tipo contiene datos y métodos, que son funciones que se pueden aplicar a los datos. Los objetos se almacenan en un repositorio central y se pueden acceder y modificar mediante un lenguaje de programación orientado a objetos.

El sistema maneja automáticamente las relaciones entre los objetos, y las consultas se realizan mediante el uso de objetos y mensajes en lugar de mediante SQL. Esto permite una mayor flexibilidad en la consulta y una mejor integridad de los datos.

Algunos ejemplos de bases de datos orientadas a objetos incluyen ObjectDB, MUMPS, y ZODB. Sin embargo, son menos populares que las bases de datos relacionales y las bases de datos no relacionales, ya que son menos escalables y no son tan ampliamente compatible con la mayoría de las tecnologías existentes.

Clase	Objetos	Atributos/datos
Empleado	Juan Pérez	Edad: 25
		Puesto: Psicóloga social
		Salario: 8000
	María Suárez	Edad: 23
		Puesto: Pedagoga
		Salario: 15 000

Figura 2 Ejemplo de una base de datos orientadas a objetos.

Bases de datos distribuidas. En una base de datos distribuida, los datos se dividen en fragmentos y se replican en varios nodos del sistema, lo que permite un acceso rápido y una alta disponibilidad. También permite la realización de operaciones en paralelo y la distribución de la carga de trabajo entre varios nodos.

Existen varios tipos de bases de datos distribuidas, incluyendo:

Bases de datos distribuidas homogéneas: en las que todos los nodos utilizan la misma tecnología y sistema operativo.

Bases de datos distribuidas heterogéneas: en las que los nodos utilizan diferentes tecnologías y/o sistemas operativos.

Algunos ejemplos de bases de datos distribuidas incluyen:

- MySQL Cluster
- Oracle RAC
- Microsoft SQL Server AlwaysOn
- MongoDB Sharding
- Cassandra

En general, las bases de datos distribuidas son adecuadas para aplicaciones que requieren un alto rendimiento, escalabilidad, alta disponibilidad y tolerancia a fallos, ya que pueden manejar un gran número de peticiones simultáneas y distribuir la carga de trabajo entre varios nodos.

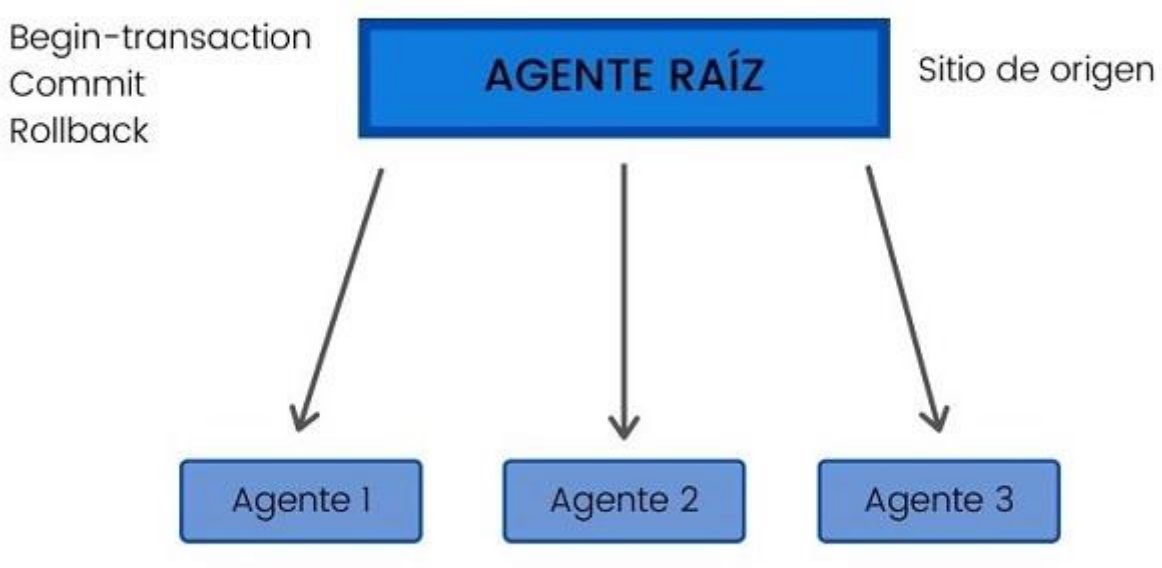


Figura 3 Bases de datos distribuidas

Bases de datos NoSQL. NoSQL es una abreviatura de "No Structured Query Language" (sin lenguaje de consulta estructurado), y se refiere a un tipo de bases de datos que no utilizan el lenguaje SQL para la recuperación y manipulación de datos. En lugar de eso, utilizan diferentes lenguajes y modelos de almacenamiento.

Las bases de datos NoSQL se caracterizan por ser escalables, flexibles y tener un alto rendimiento en comparación con las bases de datos relacionales. Estas bases de datos son adecuadas para aplicaciones que requieren un gran volumen de datos, un alto rendimiento y una gran cantidad de transacciones.

Existen varios tipos de bases de datos NoSQL, incluyendo:

Bases de datos de documentos: MongoDB, Couchbase

Bases de datos de clave-valor: Redis, Riak

Bases de datos de columnas: Cassandra, Hbase

Bases de datos de grafos: Neo4j, OrientDB

Bases de datos en tiempo real: Firebase Realtime Database

Algunas de estas bases de datos se especializan en una funcionalidad específica, como MongoDB en los documentos, Cassandra en la escalabilidad, Redis en la velocidad y almacenamiento en memoria, Neo4j en el manejo de grafos.

Figura 4 Bases de datos NoSQL.

Bases de datos orientadas a grafos. Las bases de datos orientadas a grafos (Graph databases) son un tipo de base de datos no relacionales que se especializan en el almacenamiento y recuperación de datos relacionales. En lugar de almacenar los datos en tablas relacionales, estas bases de datos los almacenan en forma de grafos, que consisten en nodos y relaciones.

Los nodos representan entidades o conceptos, y las relaciones representan las conexiones entre estas entidades. Esto permite una representación más precisa y flexible de las relaciones entre los datos, lo que es especialmente útil para aplicaciones en las que se requiere un alto rendimiento y escalabilidad en la consulta de relaciones complejas.

Algunos ejemplos de bases de datos orientadas a grafos incluyen Neo4j, ArangoDB, OrientDB, y JanusGraph. Estas bases de datos son ampliamente utilizadas en aplicaciones como la recomendación de productos, la detección de fraude, la inteligencia artificial y el análisis de redes sociales.

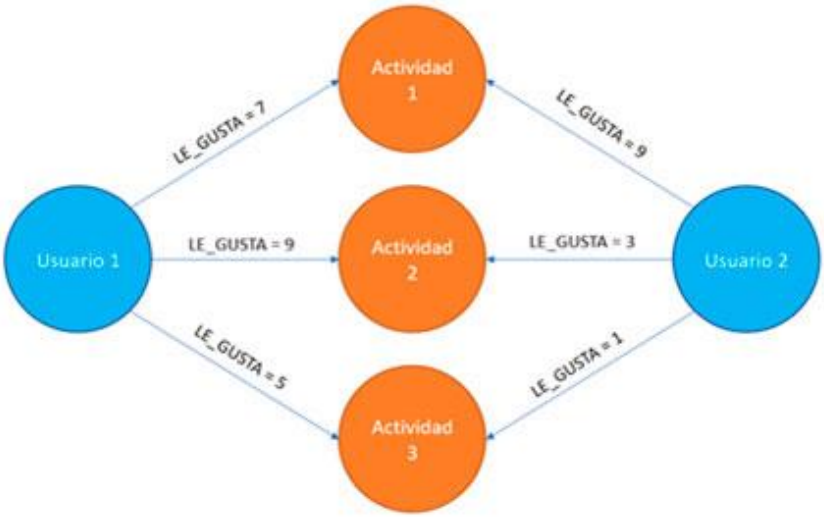


Figura 5 Bases de datos orientadas a grafos.

Bases de datos OLTP. OLTP (Online Transaction Processing) se refiere a un tipo de bases de datos diseñadas para manejar un gran número de transacciones y operaciones de lectura y escritura simultáneas. Estos sistemas están diseñados para procesar transacciones de manera rápida y eficiente, y son utilizados en aplicaciones

de negocio, como sistemas de gestión empresarial, sistemas de gestión de contenido y aplicaciones web.

En una base de datos OLTP, los datos se organizan en tablas relacionales normalizadas, y las consultas se realizan mediante el uso de lenguaje SQL. La integridad de los datos se mantiene mediante restricciones y reglas de negocio. Los sistemas OLTP son altamente escalables y están diseñados para manejar un gran número de transacciones simultáneas.

Algunos ejemplos de bases de datos OLTP incluyen MySQL, Oracle, Microsoft SQL Server, PostgreSQL y SQLite. Estas bases de datos son ampliamente utilizadas en diferentes aplicaciones, incluyendo sistemas de gestión empresarial, sistemas de gestión de contenido y aplicaciones web.

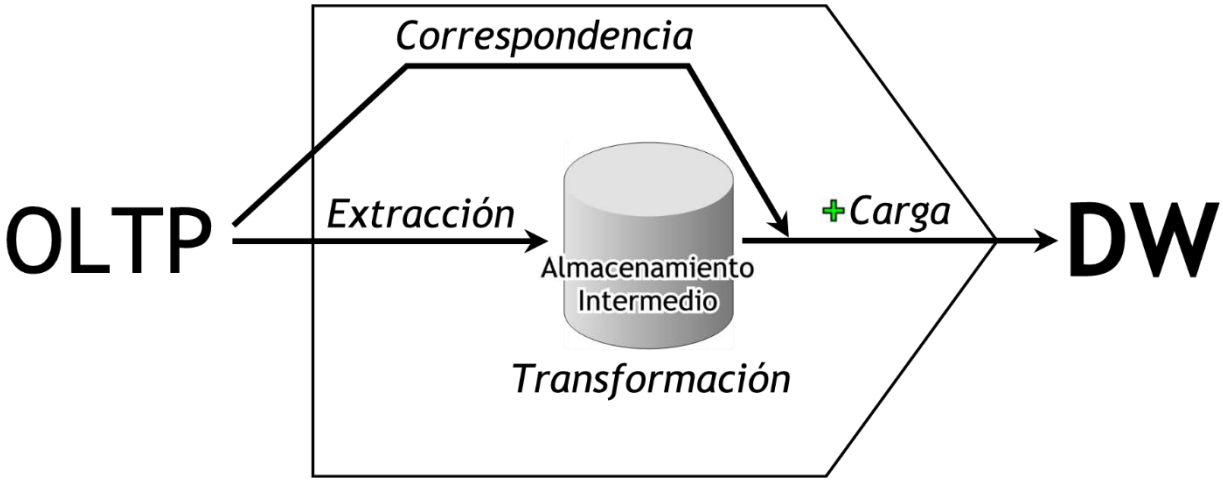


Figura 6 Bases de datos de código abierto.

Bases de datos de código abierto. Una base de datos de código abierto es una base de datos cuyo código fuente está disponible de manera gratuita y puede ser modificado y distribuido por cualquier persona. Esto significa que los desarrolladores pueden verificar y modificar el código fuente para adaptarlo a sus necesidades específicas, y pueden contribuir al desarrollo del software.

Algunos ejemplos de bases de datos de código abierto incluyen:

- MySQL
- PostgreSQL
- MongoDB
- Cassandra
- Redis
- MariaDB
- Neo4j

La ventaja de utilizar bases de datos de código abierto es que son gratuitas y no requieren licencias, lo que reduce los costos. También pueden ser personalizadas para adaptarse a las necesidades específicas de una organización y pueden contar con una gran comunidad de desarrolladores que contribuyen al mejoramiento del software.

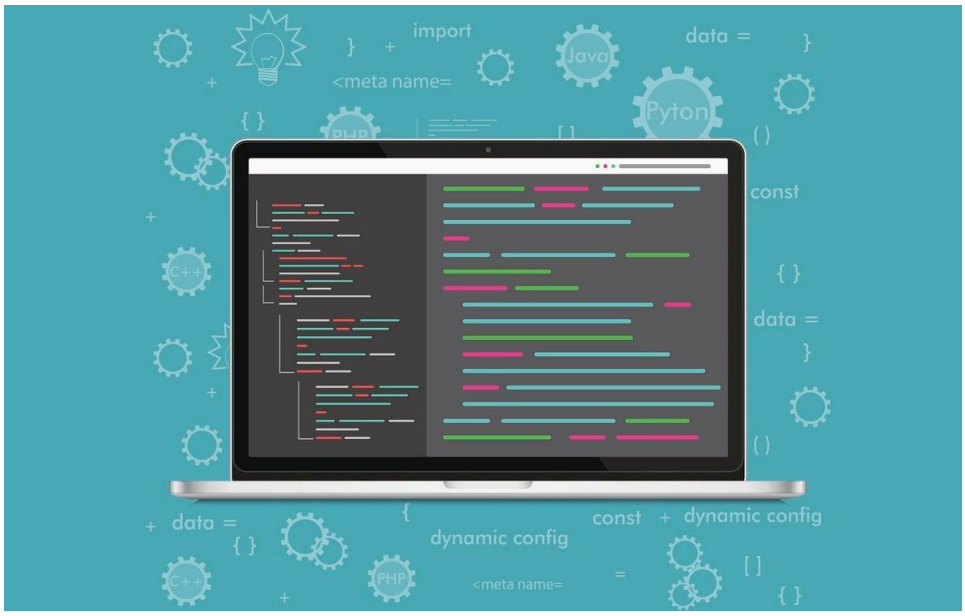


Figura 7 Bases de datos de código abierto.

Bases de datos en la nube. Las bases de datos en la nube son aquellas que se alojan en servidores remotos y se accede a ellas a través de internet, en lugar de alojarlas en un servidor local. Esto permite a los usuarios acceder a sus bases de datos desde cualquier lugar con conexión a internet.

En una base de datos en la nube, la administración y el mantenimiento de la base de datos son responsabilidad del proveedor de la nube, lo que significa que los usuarios no tienen que preocuparse por tareas como la escalabilidad, el rendimiento y la seguridad.

Algunos ejemplos de bases de datos en la nube incluyen:

- Amazon RDS: Es un servicio de bases de datos relacionales en la nube de Amazon Web Services (AWS)
- Google Cloud SQL: Es un servicio de bases de datos relacionales en la nube de Google Cloud Platform (GCP)
- Amazon DocumentDB: Es un servicio de bases de datos no relacionales de documentos en la nube de Amazon Web Services (AWS)
- Microsoft Azure Cosmos DB: Es un servicio de bases de datos no relacionales globales en la nube de Microsoft Azure.

En general, las bases de datos en la nube son adecuadas para aplicaciones que requieren un alto rendimiento, escalabilidad, alta disponibilidad y tolerancia a fallos, ya que permiten una gran escalabilidad y disponibilidad de los recursos, así como una fácil escalabilidad y adaptabilidad según las necesidades del usuario.

Figura 8 Bases de datos en la nube.

Base de datos multimodelo. Una base de datos multimodelo es una base de datos que permite el almacenamiento y la recuperación de datos en varios modelos de datos diferentes. Estos modelos pueden incluir modelos relacionales, no relacionales, orientados a grafos, de documentos, entre otros. El objetivo es poder elegir el modelo que mejor se adapte a las necesidades de cada aplicación.

Los datos se almacenan de manera nativa en cada modelo, permitiendo una mayor flexibilidad en la consulta y una mejor integridad de los datos. Permite trabajar con modelos diferentes en una sola base de datos, lo que permite una mayor eficiencia y escalabilidad en la gestión de datos.

Algunos ejemplos de bases de datos multimodelo incluyen ArangoDB, OrientDB, CosmosDB, y RavenDB. Algunas de estas bases de datos son ampliamente utilizadas en aplicaciones como la inteligencia artificial, el análisis de datos, la ciencia de datos, la recomendación de productos, la detección de fraudes, y el análisis de redes sociales.

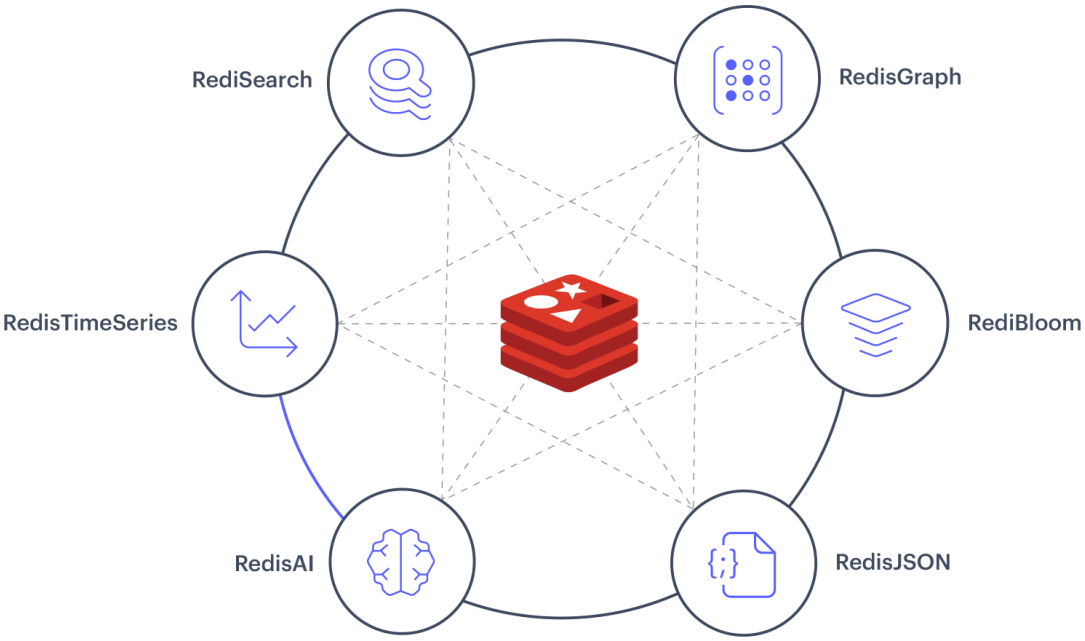


Figura 9 Bases de datos en la nube.

Bases de datos de documentos/JSON. Las bases de datos de documentos, también conocidas como bases de datos de JSON, son un tipo de base de datos no relacionales que se especializan en el almacenamiento y recuperación de documentos en formato JSON (JavaScript Object Notation). En estas bases de datos, cada documento representa un registro, y se almacena en un formato similar a un objeto JSON.

En una base de datos de documentos, los datos se organizan en colecciones, similares a las tablas en una base de datos relacional. Esto permite una mayor flexibilidad en el esquema de los datos, ya que cada documento puede tener un esquema diferente. Los sistemas de estas bases de datos son altamente escalables y están diseñados para manejar un gran número de transacciones simultáneas.

Algunos ejemplos de bases de datos de documentos incluyen MongoDB, Couchbase, and RavenDB. Estas bases de datos son ampliamente utilizadas en

aplicaciones web, sistemas de gestión de contenido, análisis de datos, inteligencia artificial, y sistemas de recomendación.

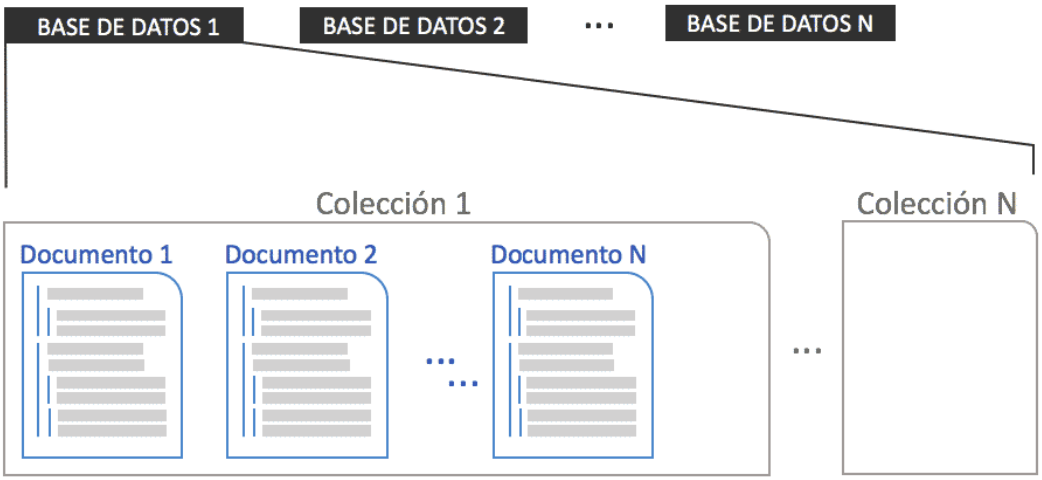


Figura 10 Bases de datos de documentos/JSON.

Bases de datos de autogestión. Las bases de datos de autogestión son un tipo de base de datos que tienen la capacidad de realizar tareas de administración y mantenimiento de manera automática. Esto incluye tareas como escalabilidad, rendimiento, disponibilidad, seguridad, copias de seguridad, y actualizaciones.

En una base de datos de autogestión, los usuarios no tienen que preocuparse por la configuración, el monitoreo y el mantenimiento de la base de datos. El sistema se encarga automáticamente de ajustar la capacidad según las necesidades de rendimiento, de asegurar la disponibilidad, de realizar copias de seguridad y de realizar actualizaciones.

Algunos ejemplos de bases de datos de autogestión incluyen Amazon Aurora, Azure Cosmos DB, Google Cloud Spanner, y CockroachDB. Estas bases de datos son ampliamente utilizadas en aplicaciones que requieren alta disponibilidad, escalabilidad y tolerancia a fallos. (Oracle, 2023)

4.1.3 Importancias de las bases de datos en las empresas.

De acuerdo con el sitio web de Academia Crandi, Las Bases de Datos poseen gran importancia, tanto en el ámbito personal como en el empresarial. Siendo considerado uno de los mayores aportes generados por la informática para las organizaciones empresariales. Actualmente, cualquier organización por más pequeña que sea, debe contar con una 18 Base de Datos. De esta forma, logrará gestionar su productos y clientes desde un sistema fácil de usar. Pero para que sea efectiva, no basta con tenerla: hay que saber cómo gestionarla.

Las principales utilidades que ofrece una base de datos a la empresa son las siguientes:

• Agrupar y almacenar todos los datos de la empresa en un único lugar.

• Facilitar que se compartan los datos entre los diferentes miembros de la empresa.

• Evitar la redundancia y mejorar la organización de la agenda.

• Realizar una interlocución adecuada con los clientes.

Si se gestiona adecuadamente una Base de Datos, se pueden obtener excelentes ventajas. Ayuda a aumentar la eficacia, haciendo que se realicen los trabajos con mayor rapidez y agilidad debido a la simplificación de estos. También se puede mejorar la seguridad de tus datos almacenados. Gracias a estos factores, se maximizan los tiempos y, por tanto, se producirá una mejora en la productividad. Estas funcionalidades aportarán un valor añadido a las empresas. Ya que una base de datos formulada correctamente conseguirá que la información y el conocimiento sean los mayores activos de la compañía. Así, se logra sacar el máximo rendimiento de los colaboradores, además de conseguir datos de los clientes potenciales. Y, por último, sabiendo que la información es poder, cuantos más datos tengamos mayor será la competitividad de la compañía.

En resumen, las bases de datos son una herramienta clave para la gestión de información y el buen funcionamiento de una empresa. Permiten almacenar y

gestionar información de manera eficiente, automatizar procesos, mejorar la comunicación y colaboración entre departamentos, ofrecer un mejor servicio al cliente, y cumplir con regulaciones de privacidad y seguridad de la información.

4.2 Sistemas Gestores de Bases de Datos.

La Mh Education (2018), define un “Sistema Gestor de Bases de Datos o SGBD, también llamado DBMS (Data Base Management System) como una colección de datos relacionados entre sí, estructurados y organizados, y un conjunto de programas que acceden y gestionan esos datos. La colección de esos datos se denomina Base de Datos o BD, (DB Data Base)”.

La Mh Education (2018), explica la historia de la arquitectura de los sistemas de bases de datos “En 1975, el comité ANSI-SPARC (American National Standard Institute - Standards Planning and Requirements Committee) propuso una arquitectura de tres niveles para los SGBD cuyo objetivo principal era el de separar los programas de aplicación de la base de datos física. En esta arquitectura el esquema de una BD se define en tres niveles de abstracción distintos:

Nivel interno o físico: el más cercano al almacenamiento físico, es decir, tal y como están almacenados en el ordenador. Describe la estructura física de la BD mediante un esquema interno. Este esquema se especifica con un modelo físico y describe los detalles de cómo se almacenan físicamente los datos: los archivos que contienen la información, su organización, los métodos de acceso a los registros, los tipos de registros, la longitud, los campos que los componen, etcétera.

Nivel externo o de visión: es el más cercano a los usuarios, es decir, es donde se describen varios esquemas externos o vistas de usuarios. Cada esquema describe la parte de la BD que interesa a un grupo de usuarios en este nivel se representa la visión individual de un usuario o de un grupo de usuarios.

Nivel conceptual: describe la estructura de toda la BD para un grupo de usuarios mediante un esquema conceptual. Este esquema describe las entidades, atributos,

relaciones, operaciones de los usuarios 20 y restricciones, ocultando los detalles de las estructuras físicas de almacenamiento".

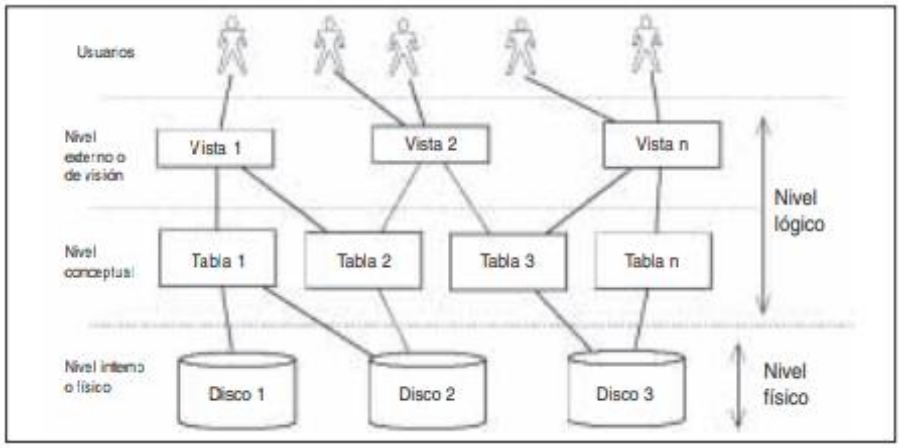


Figura 11 Niveles de abstracción de la arquitectura ANSI.

En mi opinión los Sistemas Gestores de Bases de Datos son una herramienta esencial en la gestión y el manejo de información en las empresas y organizaciones. Permiten almacenar, organizar y acceder a grandes cantidades de datos de manera eficiente y segura, lo que facilita la toma de decisiones y mejora la eficiencia en las operaciones. Además, con el aumento del volumen de datos generado en la actualidad, los DBMS se han vuelto fundamentales en el análisis de datos y el aprendizaje automático. Sin embargo, es importante elegir el adecuado según las necesidades de la empresa y asegurar una adecuada implementación y mantenimiento para asegurar su correcto funcionamiento.

4.2.1 Evolución de los Sistemas Gestores de Bases de Datos.

Los sistemas gestores de bases de datos (DBMS, por sus siglas en inglés) han evolucionado significativamente a lo largo de los años. A continuación, se describen algunos de los principales hitos en la evolución de los DBMS:

- En los años 60 y 70, se desarrollaron los primeros DBMS relacionales, como IBM's System R y Oracle. Estos sistemas introdujeron el concepto de tablas relacionales, lo que permitió una mejor organización y recuperación de la información.
- A finales de los años 70 y principios de los años 80, aparecieron los primeros DBMS cliente-servidor, lo que permitió una mayor escalabilidad y flexibilidad en las aplicaciones.
- A finales de los años 80 y principios de los años 90, aparecieron los primeros DBMS de objetos, que permitieron el almacenamiento de objetos complejos en lugar de solo datos simples.
- En los años 90 y 2000, aparecieron los DBMS basados en XML, que permitieron el almacenamiento y recuperación de datos en un formato estructurado y estandarizado.
- En la última década, los DBMS han evolucionado para incluir características de inteligencia artificial y aprendizaje automático, permitiendo un mejor análisis y toma de decisiones basadas en datos.
- Con el aumento de las aplicaciones en la nube, se han desarrollado DBMS en la nube, lo que permite una mayor escalabilidad y flexibilidad en las aplicaciones y una mayor eficiencia en el costo.

En general, la evolución de los DBMS ha sido impulsada por el aumento de la cantidad y complejidad de los datos, así como por las necesidades cambiantes de las empresas y las aplicaciones. Los DBMS modernos son capaces de manejar grandes volúmenes de datos y ofrecen una amplia variedad de herramientas para analizar y extraer información valiosa de los datos.

4.2.2 Mejores gestores de bases de datos a lo largo de la historia.

Marín (2019), nos dice que en la revista digital inesem que en la actualidad existen multitud de SGBD y pueden ser clasificados según la forma en que administran los datos en:

Un Sistema Gestor de Base de Datos (SGBD) o DataBase Managenent System (DBMS) es un sistema que permite la creación, gestión y administración de bases de datos, así como la elección y manejo de las estructuras necesarias para el almacenamiento y búsqueda de información del modo más eficiente posible.

En la actualidad, existen multitud de SGBD y pueden ser clasificados según la forma en que administran los datos en:

- Relacionales (SQL)
- No relacionales (NoSQL)

A lo largo de este post vamos a mostrar los principales sistemas gestores de bases de datos más usados de cada tipo.

MySQL Es el sistema gestor de bases de datos relacional por excelencia.
Es un SGBD multihilo y multiusuario utilizado en la gran parte de las páginas web actuales. Además, es el más usado en aplicaciones creadas como software libre.

Figura 12 Logo de MySQL

Se ofrece bajo la GNU GPL aunque también es posible adquirir una licencia para empresas que quieran incorporarlo en productos privativos (Desde la compra por parte de Oracle se está orientando a este ámbito empresarial).

Las principales ventajas de este Sistema Gestor de Bases de datos son:

- Facilidad de uso y gran rendimiento
- Facilidad para instalar y configurar
- Soporte multiplataforma
- Soporte SSL

La principal desventaja es la escalabilidad, es decir, no trabaja de manera eficiente con bases de datos muy grandes que superan un determinado tamaño.

MariaDB. Este SGBD es una derivación de MySQL que cuenta con la mayoría de las características de este e incluye varias extensiones.

Nace a partir de la adquisición de MySQL por parte de Oracle para seguir la filosofía Open Source **y** tiene la ventaja de que es totalmente compatible con MySQL.

Figura 13 Logo de MariaDB.

Entre las principales **características** de este Sistema Gestor de Bases de datos se encuentran:

- Aumento de motores de almacenamiento
- Gran escalabilidad
- Seguridad y rapidez en transacciones
- Extensiones y nuevas características relacionadas con su aplicación para Bases de datos NoSQL.

No tiene desventajas muy aparentes salvo algunas pequeñas incompatibilidades en la migración de MariaDB y MySQL o pequeños atrasos en la liberación de versiones estables.

SQLite. Más que un Sistema Gestor de bases de datos como tal, SQLite es una biblioteca escrita en C que implementa un SGBD y que permite transacciones sin necesidad de un servidor ni configuraciones.

Figura 14 Logo de SQLite

Es una biblioteca utilizada en multitud de aplicaciones actuales ya que es open source y las consultas son muy eficientes.
Las principales características de SQLite son:

- El tamaño, al tratarse de una biblioteca, es mucho menor que cualquier SGBD
- Reúne los cuatro criterios ACID (Atomicidad, Consistencia, Aislamiento y Durabilidad) logrando gran estabilidad

- Gran portabilidad y rendimiento

La gran desventaja de SQLite es la escalabilidad ya que no soporta bases de datos que sean muy grandes.

PostgreSQL. Este sistema gestor de base de datos relacional está orientado a objetos y es libre, publicado bajo la licencia BSD.

Figura 15 Logo de PostgreSQL.

Sus principales características son:

- Control de Concurrencias multiversión (MVCC)
- Flexibilidad en cuanto a lenguajes de programación
- Multiplataforma
- Dispone de una herramienta muy fácil e intuitiva para la administración de las bases de datos.
- Robustez, Eficiencia y Estabilidad.

La principal desventaja es la lentitud para la administración de bases de datos pequeñas ya que está optimizado para gestionar grandes volúmenes de datos.

Microsoft SQL Server. Es un sistema gestor de bases de datos relacionales basado en el lenguaje Transact-SQL, capaz de poner a disposición de muchos usuarios grandes cantidades de datos de manera simultánea.

Figura 16 Logo de Microsoft SQL Server.

Es un sistema propietario de Microsoft. Sus principales características son:

- Soporte exclusivo por parte de Microsoft.
- Escalabilidad, estabilidad y seguridad.
- Posibilidad de cancelar consultas.
- Potente entorno gráfico de administración que permite utilizar comandos DDL y DML.
- Aunque es nativo para Windows puede utilizarse desde hace ya un tiempo en otras plataformas como Linux o Docker.

Su principal desventaja es el precio. Cuenta con un plan gratuito (Express) pero lo normal es la elección de alguno de los planes de pago disponibles (Standard, Developer, Enterprise o SQL Azure, la versión de SQL Server en la nube).

Oracle. Tradicionalmente, Oracle ha sido el SGBD por excelencia para el mundo empresarial, considerado siempre como el más completo y robusto, destacando por:

- Soporte de transacciones.
- Estabilidad.
- Escalabilidad.
- Multiplataforma.

Figura 17 Logo de Oracle.

La principal desventaja, al igual que SQL Server, es el coste del software ya que, aunque cuenta con una versión gratuita (Express Edition o XE), sus principales opciones son de pago.

Las opciones de pago disponibles son:

1. Standard Edition (SE)
2. Standard Edition One (SE1)
3. Standard Edition 2 (SE2)
4. Personal Edition (PE)
5. Lite Edition (LE)
6. Entreprise Edition (EE)

MongoDB. Estamos ante el Sistema Gestor de Bases de Datos no relacionales (SGBD NoSQL) más popular y utilizado actualmente.

MongoDB es un SBGD NoSQL orientado a ficheros que almacena la información en estructuras BSON con un esquema dinámico que permite su facilidad de integración.

Empresas como Google, Facebook, eBay, Cisco o Adobe utilizan MongoDB como Sistema Gestor de Bases de datos.

Figura 18 Logo de MongoDB.

Las principales características de MongoDB son:

- Indexación y replicación
- Balanceo de carga
- Almacenamiento en ficheros
- Consultas ad hoc
- Escalabilidad horizontal
- Open Source

Como desventaja principal, MongoDB no es un SGBD adecuado para realizar transacciones complejas.

Redis. Está basado en el almacenamiento clave-valor. Podríamos verlo como un vector enorme que almacena todo tipo de datos, desde cadenas, hashses, listas, etc.

El principal uso de este SGBD es para el almacenamiento en memoria caché y la administración de sesiones**.**

Figura 19 Logo de MongoDB.

Las características principales son:

- Atomicidad y persistencia
- Gran velocidad
- Simplicidad
- Multiplataforma

Cassandra. Al igual que Redis, Cassandra también utiliza almacenamiento clave-valor. Es un SGBD NoSQL distribuido y masivamente escalable.

Figura 20 Logo de Cassandra.

Facebook, Twitter, Instagram, Spotify o Netflix utilizan Cassandra.
Dispone de un lenguaje propio para las consultas denominado CQL (Cassandra Query Languaje).
Las principales características de este SGBD NoSQL son:

- Multiplataforma
- Propio lenguaje de consultas (CQL)
- Escalado lineal y horizontal
- Es un SGBD distribuido
- Utiliza una arquitectura peer-to-peer

Es difícil determinar cuál es el mejor sistema gestor de bases de datos (DBMS) ya que depende de las necesidades y requerimientos específicos de cada empresa o proyecto. Sin embargo, concuerdo con los gestores de bases de datos ya mencionados anteriormente ya son los DBMS más populares y ampliamente utilizado en la actualidad.

4.3 API

Como se mencionó en el principio de este apartado, API (Application Programming Interface, o Interfaz de Programación de Aplicaciones en español) es un conjunto de reglas y estándares que se utilizan para permitir que diferentes aplicaciones se comuniquen entre sí. Una API específica proporciona una interfaz para acceder a los servicios o funciones de un sistema o aplicación en particular. Que permiten a los desarrolladores acceder a una variedad de servicios y funciones, como bases de datos, servicios de mensajería, servicios de nube, servicios de redes sociales, entre otros, sin necesidad de conocer los detalles internos de cómo se implementan esos servicios. Esto permite a los desarrolladores crear aplicaciones más complejas y potentes al integrar servicios de diferentes fuentes.

Existen diferentes tipos, como las API web, que permiten el acceso a servicios a través de protocolos de red estándar, y las API de software, que permiten el acceso a funciones y servicios en un sistema de software específico.

Se puede entender API (Application Programming Interface) como las funcionalidades que aporta un cierto servicio software facilitando que pueda ser utilizado por otro software para mejorar sus resultados. Normalmente no es un resultado en si mismo, sino que sirve de enlace entre un software ya creado y otro al que este le puede resultar útil, lo que se conocería como una interacción “software-to-software”. En este contexto para realizar una API práctica esta tendrá que aportar un concepto claro de las funcionalidades que ofrece, quedando bien definidas, para que el que quiera usarlas posteriormente tenga claro cómo funcionan y cómo puede acceder a ellas. Los beneficios de esto son que, en vez de tener que partir de cero y crear un código necesario, pero ya existente, las APIs proporcionan la posibilidad de obtener los mismos resultados sin tener que perder el tiempo en reimplementaciones. (Plaza Sheila, 2015-2016)

Sin embargo, IBM Cloud Education (2021) menciona que “Una API, o interfaz *de* programación de aplicaciones, es un conjunto de reglas que determinan cómo las aplicaciones o los dispositivos pueden conectarse y comunicarse entre sí. Una API REST es una API que se ajusta a los principios de diseño de REST, un estilo de arquitectura también denominado *transferencia de estado representacional*. Por este motivo, las API REST son a veces denominadas API RESTFUL.”

En resumen, una API es una interfaz que permite a las aplicaciones interactuar y comunicarse entre sí de manera estandarizada, lo que permite a los desarrolladores acceder a una variedad de servicios y funciones de manera más eficiente y sencilla.

4.3.1 Principios de diseño de REST

REST (Representational State Transfer) es un estilo de arquitectura de software que se utiliza para desarrollar servicios web. Los principios de diseño de REST se basan en los principios de la arquitectura de la World Wide Web y son los siguientes:

- Cliente-Servidor: El sistema se divide en clientes y servidores, donde el cliente envía solicitudes y el servidor proporciona respuestas.
- Estado Independiente: Cada solicitud contiene toda la información necesaria para que el servidor pueda entenderla y procesarla, sin necesidad de almacenar el estado del cliente.
- Cacheable: Las respuestas son marcadas como aptas para caché, permitiendo al cliente almacenar la respuesta para su uso futuro.
- Interfaz uniforme: El servidor proporciona una interfaz uniforme para acceder a los recursos, permitiendo que los clientes utilicen la misma interfaz para acceder a todos los recursos.
- Sistema de enrutamiento: El servidor utiliza un sistema de enrutamiento para mapear las solicitudes a los recursos apropiados.
- Sistema de representación: El servidor proporciona diferentes representaciones de los recursos, como HTML, XML o JSON.
- Comunicación síncrona: El cliente y el servidor se comunican de manera síncrona, esperando una respuesta para cada solicitud.

Siguiendo estos principios, REST permite crear servicios web escalables, flexibles y fáciles de utilizar y mantener. Los servicios web RESTful se basan en estos principios y se utilizan ampliamente en aplicaciones web y móviles modernas.

Sin embargo, IBM Cloud Education (2021) dice “En el nivel más básico, una API es un mecanismo que permite a una aplicación o servicio acceder a un recurso dentro de otra aplicación o servicio. La aplicación o servicio que realiza el acceso se

denomina cliente y la aplicación o servicio que contiene el recurso se denomina servidor.

Algunas API, como SOAP o XML-RPC, imponen una estructura estricta a los desarrolladores. Sin embargo, las API REST se pueden desarrollar mediante el uso de prácticamente cualquier lenguaje de programación y son compatibles con una variedad de formatos de datos. El único requisito es que se ajusten a los siguientes seis principios de diseño de REST, también conocidos como restricciones de arquitectura:

- **Interfaz uniforme:** Todas las solicitudes de API para el mismo recurso deben ser iguales, independientemente de la procedencia de la solicitud. La API REST debe asegurarse de que el mismo dato, como el nombre o la dirección de e-mail de un usuario, pertenezca a un único identificador uniforme de recurso (URI). Los recursos no deben ser demasiado grandes, sin embargo, deben contener toda la información que el cliente pueda necesitar.
- **Separación entre cliente y servidor:** En el diseño de API REST, las aplicaciones de cliente y de servidor deben ser completamente independientes entre sí. La única información que la aplicación de cliente debe conocer es el URI del recurso solicitado. No puede interactuar con la aplicación de servidor de ninguna otra forma. Del mismo modo, una aplicación de servidor no debe modificar la aplicación de cliente más allá de entregarle los datos solicitados vía HTTP.
- **Sin estado.** Las API REST son sin estado, lo que significa que cada solicitud debe incluir toda la información necesaria para procesarla. En otras palabras, las API REST no requieren ninguna sesión en el lado del servidor. Las aplicaciones de servidor no pueden almacenar ningún dato relacionado con una solicitud de cliente.
- **Capacidad de almacenamiento en memoria caché.** Siempre que sea posible, los recursos deben poder almacenarse en la memoria caché en el

lado del cliente o el servidor. Las respuestas de servidor también necesitan contener información de si el almacenamiento en memoria caché está permitido para el recurso entregado. El objetivo es mejorar el rendimiento en el lado del cliente, al mismo tiempo que aumenta la escalabilidad en el lado del servidor.

- **Arquitectura de sistema de capas.** En las API REST, las llamadas y respuestas pasan por diferentes capas. Como regla general, no debe suponer que las aplicaciones de cliente y de servidor se conectan directamente entre sí. Puede haber una serie de intermediarios diferentes en el bucle de comunicación. Las API REST deben diseñarse para que ni el cliente ni el servidor puedan notar si se comunican con la aplicación final o con un intermediario.
- **Código bajo demanda (opcional).** Generalmente, las API REST envían recursos estáticos, pero en algunos casos, las respuestas también pueden contener un código ejecutable (como applets de Java). En estos casos, el código solo debería ejecutarse bajo demanda.”

4.3.2 Funcionamiento de las API

El funcionamiento de las API se basa en la comunicación entre el cliente y el servidor. El cliente es la aplicación que desea acceder a los datos o funcionalidades de otro sistema, mientras que el servidor es el sistema que proporciona el acceso a dichos datos o funcionalidades. El cliente envía una solicitud al servidor utilizando un protocolo de comunicación establecido, como HTTP, y el servidor responde con los datos o información solicitada.

Las API también pueden incluir mecanismos de autenticación y autorización para asegurar que solo las aplicaciones autorizadas pueden acceder a los datos o funcionalidades. Además, las API suelen documentarse para que los desarrolladores puedan comprender cómo utilizarlas adecuadamente.

Sin embargo, redhat (2023) menciona que “Las API permiten que sus productos y servicios se comuniquen con otros, sin necesidad de saber cómo están implementados. Esto simplifica el desarrollo de las aplicaciones y permite ahorrar tiempo y dinero. Las API le otorgan flexibilidad; simplifican el diseño, la administración y el uso de las aplicaciones; y ofrecen oportunidades de innovación, lo cual es ideal al momento de diseñar herramientas y productos nuevos (o de gestionar los actuales).

A veces, las API se consideran como contratos, con documentación que representa un acuerdo entre las partes: si una de las partes envía una solicitud remota con cierta estructura en particular, esa misma estructura determinará cómo responderá el software de la otra parte.

También permiten la colaboración entre el equipo comercial y el de TI, ya que simplifican la forma en que los desarrolladores integran los elementos de las aplicaciones nuevas en una arquitectura actual. Las necesidades comerciales suelen cambiar rápidamente en respuesta a los mercados digitales en constante cambio, donde la competencia puede modificar un sector entero con una aplicación nueva. Para seguir siendo competitivos, es importante admitir la implementación y el desarrollo rápidos de servicios innovadores. El desarrollo de aplicaciones nativas de la nube es una forma identificable de aumentar la velocidad de desarrollo y se basa en la conexión de una arquitectura de aplicaciones de microservicios a través de las API.

Las API son un medio simplificado para conectar su propia infraestructura a través del desarrollo de aplicaciones nativas de la nube, pero también le permiten compartir sus datos con clientes y otros usuarios externos. Las API públicas aportan un valor comercial único porque simplifican y amplían sus conexiones con los partners y, además, pueden rentabilizar sus datos (un ejemplo conocido es la API de Google Maps).”

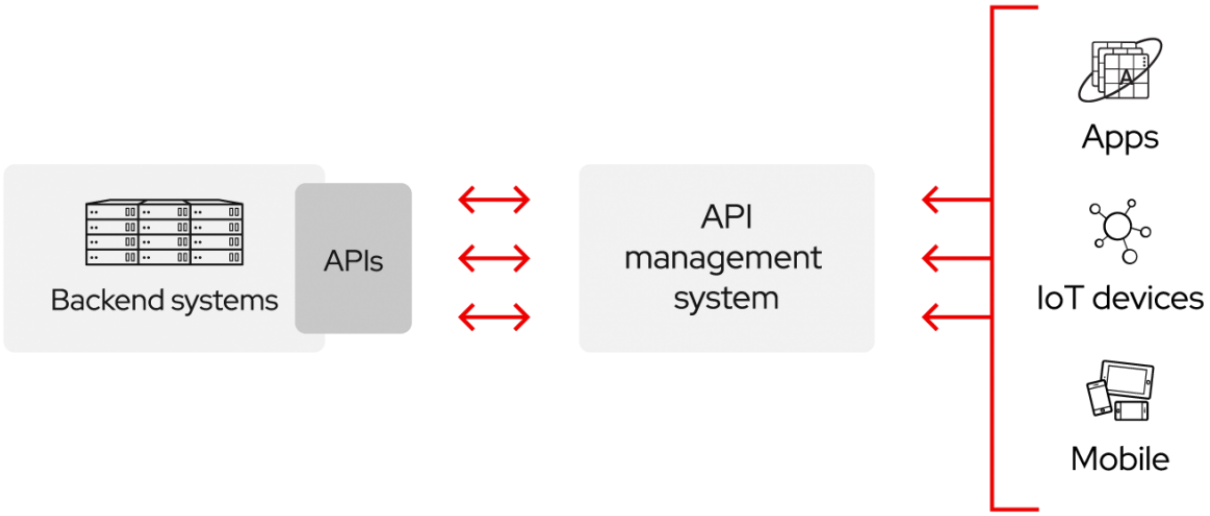


Figura 21 Consumo de API REST

4.4 Tipos de Api

Eugenia (2023) menciona varios tipos de API públicas y privadas, incluyendo bibliotecas de lenguajes de programación, HTTP, SOAP y RESTful.

API públicas. Las API públicas o abiertas son interfaces que están disponibles para cualquier usuario.

Por ejemplo, existen bibliotecas de algunos lenguajes de programación que cuentan con una amplia gama de bibliotecas y aplicaciones que son necesarias para hacer funcionar un software. En lugar de hacer que los desarrolladores programen desde cero, puedes emplear estas reglas o códigos sin restricciones y facilitar el proceso de diseño y desarrollo.

En otros escenarios, tal vez un escritor requiera integrar una imagen o un video dentro de su blog online. Con la API adecuada puede añadir el enlace y hacer que aparezca en su sitio web.

API privada. Las API privadas o cerradas son herramientas a las que únicamente pueden acceder los usuarios que han sido autorizados para ello.

Estas aplicaciones generalmente están diseñadas para el funcionamiento de una empresa en particular, por lo que sirven para hacer llamados a datos de un área, aplicar reglas definidas o establecer formatos preexistentes y propios de una organización. Estas API cuentan con gateways (o puertas) cerradas que solo permiten el acceso a quienes se identifican como parte del grupo de colaboradores o personas usuarias.

HTTP API. Las HTTP API o web API son interfaces diseñadas específicamente para usarse en el desarrollo de sitios web por medio del protocolo de transferencia de hipertexto (HTTP, por sus siglas en inglés).

Al desarrollar un sitio, es necesario transferir información de una fuente a otra disponible en la internet. Estas API están optimizadas para ubicar, extraer o aplicar alguna regla o rutina dentro de una página electrónica o en un servidor que funcione en la red.

SOAP API. Las SOAP API o API de protocolo simple de acceso a objetos consisten en una serie de pautas que permiten a un programa acceder a información básica de otra ubicación.

Como tal, estas API no proveen archivos o datos de otros repositorios. Por el contrario, solamente dan las indicaciones que regulan la extracción e integración de estos elementos de manera segura y estandarizada.

RESTful API. Las RESTful API o API de transferencia de estado representacional son interfaces que sirven para hacer llamados a un estilo específico de arquitectura de software basado en el contenido multimedia.

Las API de este tipo son excelentes para el diseño de plataformas más complejas, como las aplicaciones móviles, que requieren la integración de tablas, imágenes y videos. De manera sencilla, extraen la información solicitada y la arrojan o integran al software.

4.5 Aplicaciones para desarrollar API REST

Existen varias aplicaciones y herramientas que se pueden utilizar para desarrollar API REST, algunas de las más populares incluyen:

Express.js: Es un marco de desarrollo para Node.js que permite crear fácilmente API RESTful. Es muy popular entre los desarrolladores de JavaScript.

Figura 22 Express.js

Flask: Es un marco de desarrollo para Python que permite crear fácilmente API RESTful. Es muy popular entre los desarrolladores de Python.

Figura 23 Logo de Flask

Django Rest Framework: Es un marco de desarrollo para Python que permite crear fácilmente API RESTful. Es muy popular entre los desarrolladores de Python.

Figura 24 Django Rest Framework.

Spring: Es un marco de desarrollo para Java que permite crear fácilmente API RESTful. Es muy popular entre los desarrolladores de Java. (Yasmani, 2022)

Figura 25 Spring

Ruby on Rails: Es un marco de desarrollo para Ruby que permite crear fácilmente API RESTful. Es muy popular entre los desarrolladores de Ruby.

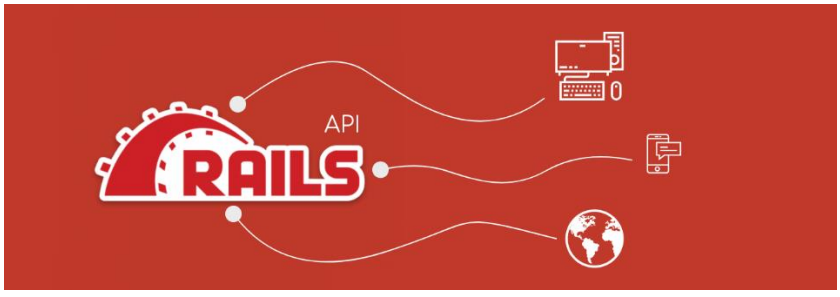


Figura 26 Logo Ruby on Rails.

ASP.NET: Es un marco de desarrollo de Microsoft que permite crear aplicaciones web utilizando C# y otros lenguajes de programación de Microsoft. Es una de las opciones populares para desarrollar API RESTful en plataformas Windows.

Figura 27 ASP.NET

Postman: Es una herramienta para probar y desarrollar API. Es muy popular entre los desarrolladores de todas las plataformas.

Figura 28 Postman

Swagger: Es una herramienta para documentar y probar API RESTful. Es muy popular entre los desarrolladores de todas las plataformas.

Figura 29 Swagger

Loopback: Es un marco de desarrollo para Node.js que permite crear fácilmente API RESTful. Es muy popular entre los desarrolladores.

Figura 30 Loopback

Estas son solo algunas de las herramientas más populares. (rootstack, 2022)

4.6 Asistentes de voz inteligentes

Cada compañía que construye un asistente aplica métodos y acercamientos al desarrollo lo cual afecta al resultado final. Un asistente puede sintetizar palabras más cualitativamente, otro puede hacerlo con más precisión y sin explicaciones ni correcciones adicionales, otros pueden realizar un rango más estrecho de tareas, pero con mayor precisión y como el usuario quiera. El conjunto de características que tiene un asistente depende completamente de las áreas a las que un desarrollador ha prestado mayor atención, dado que todos los sistemas se basan en el aprendizaje automatizado (Polyakov, 2018, págs. 1-5).

Un asistente de voz inteligente se relaciona por medio de software, asociado con el sistema operativo de dispositivos específicos, el cual pueden interactuar con el ser humano a través de comandos de voz. Para esto, los asistentes de voz solo pueden responder preguntas simples, búsquedas de información, así como el pronóstico del tiempo o realizar acciones telefónicas básicas como “llamar a Jordi”.

Es por esto por lo que las versiones móviles de las páginas web comenzaron a priorizar y adaptar sus interfaces a los comandos de voz. Por ejemplo, estos asistentes están disponibles en muchos de los servicios de Google, desde la búsqueda del navegador hasta Google Maps y, por supuesto, en teléfonos Android, donde el usuario tiene solo que indicar lo que necesita saber y el dispositivo responderá de inmediato. Amazon promueve “echo”, una serie de altavoces inteligentes, para tener un asistente digital en casa. Esto es un cambio tan trascendental que significa olvidar la pantalla tradicional y empezar a interactuar de una manera radicalmente diferente por medio de la voz (Nicolás, 2018).

Estos dispositivos introdujeron una interfaz conversacional dentro del hogar que permite a los usuarios el pedir y guardar información, controlar casas inteligentes y una serie de acciones dentro del internet. Por ejemplo, una persona con escasa movilidad podría controlar las luces dentro de la casa o las cerraduras usando la voz, también una persona con discapacidad visual puede preguntar la hora o información sobre el clima. Varios estudios encontraron que este nuevo paradigma de las (Pradhan, 2018)interfaces visuales avanzadas (AVI, por sus siglas en inglés) tiene un tremendo potencial para inclusividad y accesibilidad (Pradhan, Mehta, & Findlater, 2018).

4.6.1 Tipos de asistentes de voz

De acuerdo con Paz (2020, págs. 4-11) las principales asistentes de voz más populares son:

GOOGLE ASSISTANT. Se trata de uno de los asistentes virtuales más recientes, anunciado en la conferencia Google I/O de 2016. Es posible encontrarlo hoy en día implementado en una gran variedad de dispositivos como smartphones, tablets, TV box, Smart TVs, smartwatchs, coches y en dispositivos específicamente dirigidos a este asistente como Google Home o el recientemente presentado Google Nest Hub. Los dispositivos compatibles con dicho asistente suelen presentar el logotipo mostrado en la siguiente figura.

Figura 31 Google Assistant.

AMAZON ALEXA. Alexa es el asistente de voz desarrollado por Amazon que fue lanzado en el año 2014. Está disponible para Fire OS 5.0, iOS 8.0 y Android 4.4 y posteriores. En la actualidad se encuentra disponible en diferentes idiomas, entre ellos, el español. En la actualidad Alexa se puede instalar en dispositivos móviles a través de los markets oficiales de cada plataforma, pero lo que hace realmente interesante a Alexa no es la posibilidad de poder ser instalado en dispositivos móviles, sino la posibilidad de adquirir dispositivos hardware que ya lo incorporan y permiten la interconexión de múltiples dispositivos.

Figura 32 Amazon Alexa.

SIRI. Se trata del asistente personal desarrollado inicialmente por Apple [3] en 2010. A partir de 2016 pasó de estar disponible únicamente en los dispositivos iPhone a incorporarse al resto de dispositivos de Apple con su integración en macOS Sierra. La llegada de Siri fue una auténtica revolución para esta industria ya que fue uno de los pioneros en este campo. Por otro lado, en la actualidad este asistente realiza acciones similares a sus competidores, sin embargo, el ecosistema de dispositivos e integraciones se encuentra más limitado que el de sus competidores Amazon y Google.

En la misma línea que sus competidores, Apple ha desarrollado un kit de integración con dispositivos IoT llamado Apple Homekit pero con una integración de un número menor de dispositivos que sus contrincantes. Los requisitos de cifrado de datos que exige Apple lo convierten en un sistema con unos estándares de seguridad bastante altos lo que hace de él una plataforma a la que no muchos dispositivos pueden acceder.

Figura 33 Siri

BIXBY. Se trata del asistente virtual desarrollado por Samsung Electronics. Su primera presentación en 2017 fue acompañada del famoso dispositivo de la marca Samsung Galaxy S8. Inicialmente este asistente solo se encontraba disponible en Android, pero poco después Samsung anunció Bixby 2.0 el cual podría ser encontrado ya fuera de smartphone y unido a una amplia gama de dispositivos como televisores e incluso refrigeradores. Al mismo tiempo, anunció una SDK para poder desarrollar aplicaciones para este asistente. Con una variedad mucho más modesta de dispositivos compatibles que sus competidores Google y Amazon, Samsung apuesta por un enfoque similar al de Apple, siendo compatible su asistente únicamente con dispositivos Samsung y desarrollando su propio gadget conocido como Galaxy Home.

Figura 34 Bixby.

CORTANA. Es el asistente virtual creado por Microsoft en 2009 y que en 2015 dio el salto a ordenadores de escritorio y dispositivos móviles con Windows 10. Este asistente también está disponible como aplicación en las plataformas Android, iOS y XBOX. En 2017 Microsoft anunció la capacidad de desarrollar aplicaciones para esta plataforma al igual que sucede en sus competidores y en 2018 ha realizado integraciones con diversas marcas IoT. A principios de 2019, ante la gran competición en este mercado, Microsoft ha adoptado la posición de convertir Cortana en una skill (aplicación para Amazon Alexa) que pueda ser empleada por otros asistentes como Alexa o Google Assistant. El enfoque que le han dado a su asistente a partir de esta fecha es el de poder ser utilizado por los demás y no entrar en una competencia directa.

Figura 35 Cortana.

Mycroft. Se trata del primer asistente virtual del mercado open source bajo una licencia de código Apache. Es un asistente desarrollado en Python que se encuentra en una fase aún temprana de su desarrollo. Su enfoque es totalmente abierto en oposición a los que podemos encontrar en el mercado y se postula como opción open source en esta tecnología. El proyecto Mycroft también se encuentra trabajando en un gadget hardware como el de sus competidores, pero también siguiendo una filosofía open hardware. El primer dispositivo fue Mark I y ahora se encuentra el dispositivo Mark II en un proceso de crowfunding en Kickstarer.

Figura 36 Mycroft.

V MATERIALES Y MÉTODOS.

5.1 Modalidad de Investigación

En la elaboración del presente se han utilizado varias modalidades tales como lo son las siguientes:

5.1.1 Bibliográfica

Se optó por esta vía debido a la utilización de Internet como principal fuente de información para adquirir conocimientos esenciales sobre el tema. Se han consultado diversos tipos de documentos, revistas, tesis de grado y sitios web.

5.1.2 Aplicada

Se eligió esta modalidad basándose en la aplicación de los conocimientos adquiridos a lo largo de mis estudios en la carrera universitaria, específicamente en los módulos relacionados con la investigación de operaciones, programación orientada a objetos, ingeniería de software, programación de bases de datos y bases de datos.

5.2 Análisis del sistema de monitoreo de remolques.

Como primer paso para el desarrollo de este proyecto, se entrevistó al responsable del mantenimiento de software, para conocer sobre el funcionamiento del sistema de la pizarra, explicándonos cada uno de los componentes que la conforman y para terminar de comprender mejor el proceso, se hizo un análisis de la documentación de la pizarra digital.

Tal como se muestra en la siguiente figura 37, es el diseño del sistema que actualmente la empresa usa, en base esto se toma en cuenta los componentes necesarios que son necesarios para la realización del proyecto. Una vez hacer un análisis y recaudar información necesaria, se puede tomar en cuenta que elementos utilizar al desarrollar la Skill para el asistente de voz (Alexa).

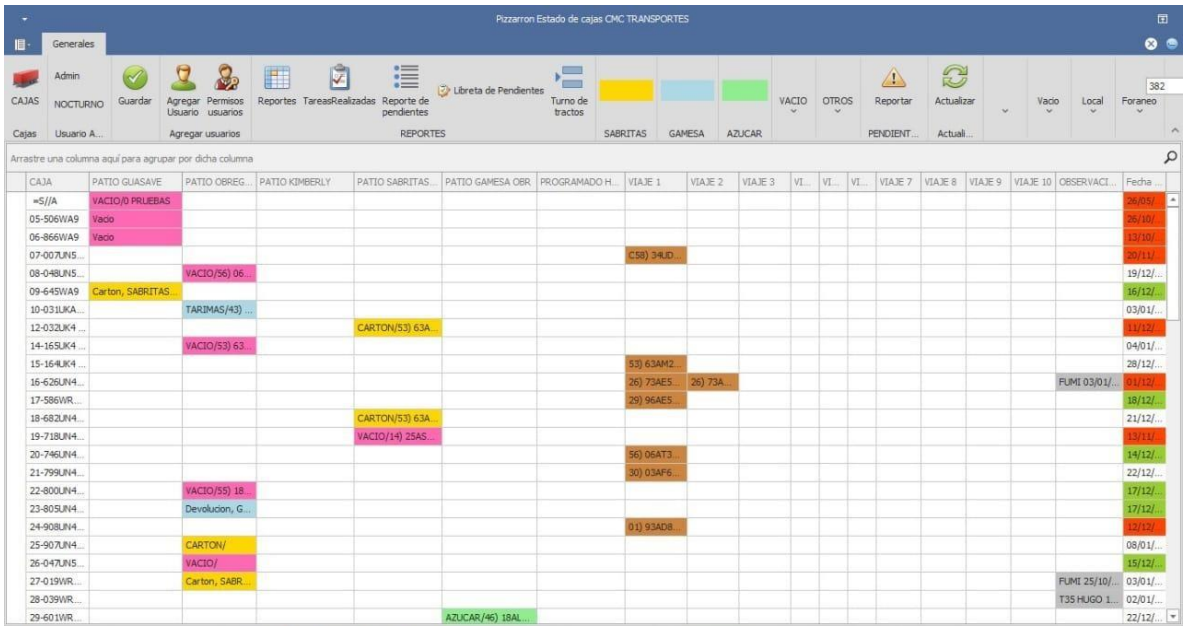


Figura 37 Diseño de la pizarra

5.3 Problemáticas detectadas

La utilización de la pizarra es complicada ya que es poco amigable con el usuario, al momento de utilizarla puede ser confuso, pero con las diferentes pruebas que se realizaron utilizándola se comprendió que faltan muchas validaciones a la aplicación que permite ingresar todo tipo de caracteres que pueden influir mucho en los errores ya que depende de una norma de escritura rigurosa, también es complicado buscar una caja en especifica a cada momentos y como es parecido al diseño de un Excel hay muchas casillas que son muy chicas por lo que en un descuido se puede seleccionar otra caja que no era ocasionando una información errónea.

5.4 Creación de Api rest

En la siguiente figura 38 se muestra el código con el cual se hace la conexión a SQL SERVER, que se le tienen que enviar las claves de conexión que es el servidor, el nombre de la base de datos, el usuario y la contraseña para que nos permita abrir la comunicación entre la API REST hacia la base de datos que utiliza la empresa ya que sin esta no podríamos hacer nada.

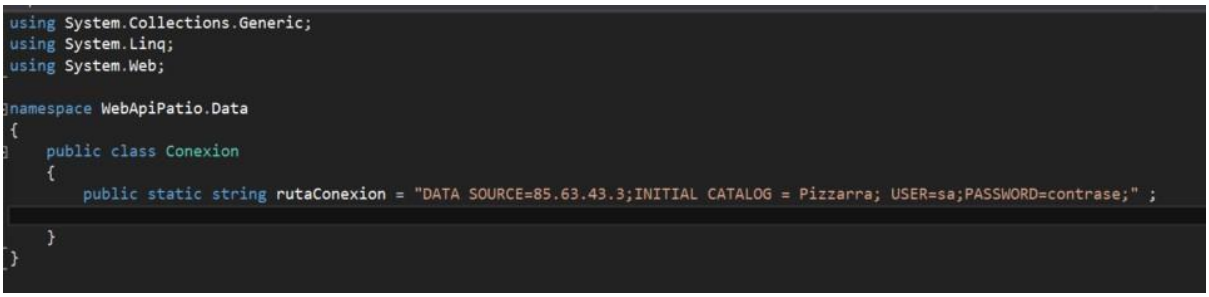


Figura 38 Conexión a SQL SERVER

5.4.1 Estructura de datos

En la siguiente figura 39 se muestra la clase donde se declaran las variables que se van a utilizar para insertar o extraer la información de la base de datos, estas variables deben de coincidir las declaradas en la tabla a la que se quiere hacer uso, ya que, si no son idénticos en nombre y tipo de datos, marcaría error.

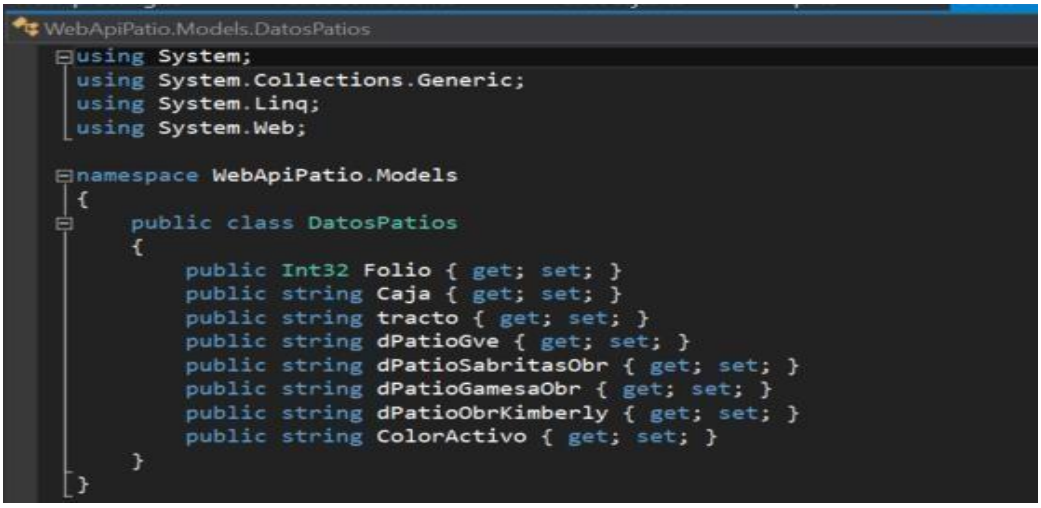


Figura 39 Estructura de datos

5.4.2 GETS

En la siguiente figura 40 se hacen 2 tipos de consultas GET, para poder reconocer el tipo de consulta que se hará hay una validación, que si lo que recibe el id es diferente a uno entonces entra al if donde se validara si el usuario e la contraseña son correctos, pero si el valor del id es igual a uno entonces eso indica que se ira al else

donde se consume la API REST que tiene la empresa para saber la localización a tiempo real de las cajas.

```
// GET api/<controller>/5
public string Get(string id)
{
    string entrega = "";
    string phrase = id;
    string[] words = phrase.Split(',');
    int cont = words.Count();
    if (cont != 1)
    {
        PatiosCajas.ObtenerUsuario(words[0]);
        if (PatiosCajas.Veri == words[1])
        {
            entrega = "si";
        }
        else
        {
            entrega = "no";
        }
    }
    else
    {
        Api api = new Api();
        //PatiosCajas.ObtenerTractos(id);
        string tract = "Caja #" + id;
        api.UltimaPosicionAsyc(tract);
        entrega = Api.Localizacion;
    }
    return entrega;

}
```

Figura 40 Método GET

En la siguiente figura 41 se muestra la función de obtenerUsuario la cual recibe un dato string, el cual se va a ingresar a un método que llama a la función ObtenerUsuarios que está en la clase PatiosCajas,

```
PatiosCajas.ObtenerUsuario(words[0]);
if (PatiosCajas.Veri == words[1])
{
    entrega = "si";
}
else
{
    entrega = "no";
}
```

Figura 41 Método obtener usuario

En la siguiente figura 42 se muestra el método ObtenerUsuario el cual revira el nombre de usuario e ira a buscarlo en la base de datos por medio de un store procedure que llama a la tabla donde se encuentran las credenciales de estos, para saber si existe ese usuario, si lo encuentra traerá sus claves para compararlas y regresar un sí o no.

En la siguiente figura 43 se muestra el segundo GET, el cual su finalidad es llamar una API que contrato la empresa para localizar donde se encuentran los remolques, se le ocupa enviar los primeros 2 dígitos de las placas de los remolques.

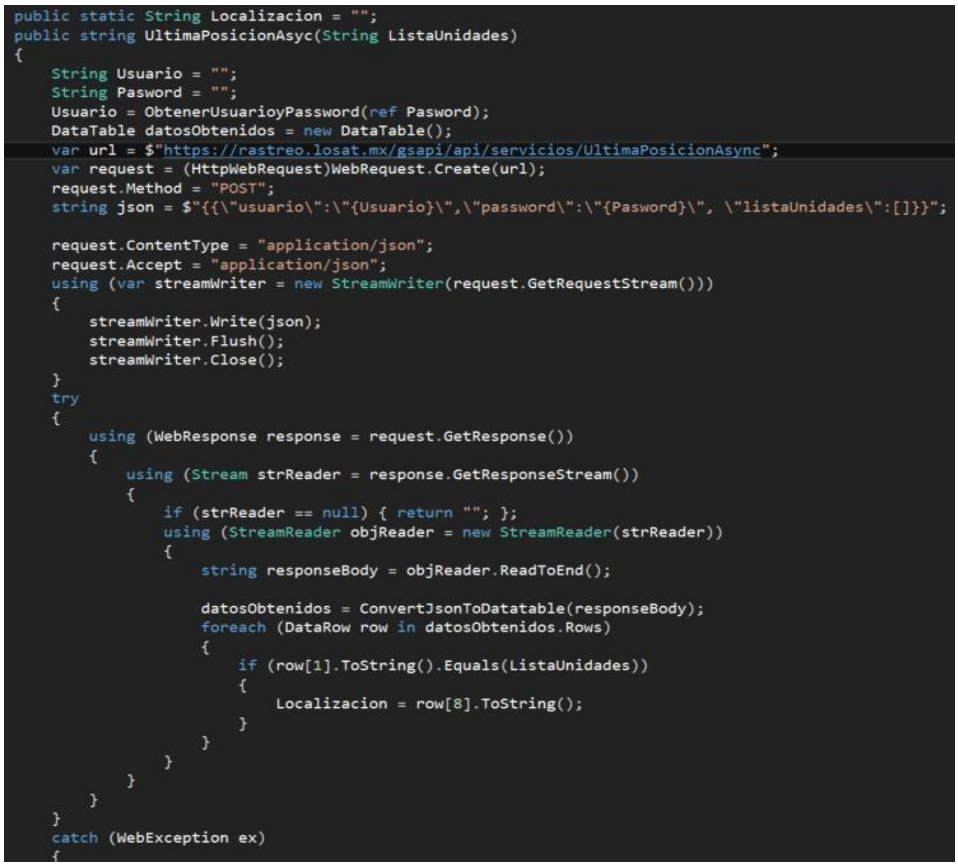
```
public static string ObtenerUsuario(string Usuario)
{

    using (SqlConnection oConexion = new SqlConnection(Conexion.rutaConexion))
    {
        string password = "";
        SqlCommand cmd = new SqlCommand("Usuario", oConexion);
        cmd.CommandType = CommandType.StoredProcedure;
        cmd.Parameters.AddWithValue("@Usuario", Usuario);

        try
        {
            oConexion.Open();
            cmd.ExecuteNonQuery();

            using (SqlDataReader dr = cmd.ExecuteReader())
            {
                while (dr.Read())
                {
                    password = dr["password"].ToString();
                }
                Veri = password;
            }
            return password;
        }
        catch (Exception ex)
        {
            return password;
        }
    }
}
```

Figura 42 Procedimiento para obtener usuario y contraseña.

```
Api api = new Api();
//PatiosCajas.ObtenerTractos(id);
string tract = "Caja #" + id;
api.UltimaPosicionAsyc(tract);
entrega = Api.Localizacion;
```

Figura 43 Llamado a la Api

Los datos de la API REST se tienen que nombrar de cierto modo el cual lo podemos observar en la figura 44, se le tienen que enviar el usuario y contraseña que están de forma estáticas en el programa, para que nos envié los datos que ocupamos en un archivo Json, ya que tenemos el archivo buscamos la placa que necesitamos en el archivo Json y extraemos la ubicación.



```
public static String Localizacion = "";
public string UltimaPosicionAsyc(String ListaUnidades)
{
    String Usuario = "";
    String Pasword = "";
    Usuario = ObtenerUsuarioyPassword(ref Pasword);
    DataTable datosObtenidos = new DataTable();
    var url = $"https://rastreo.losat.mx/gsapi/api/servicios/UltimaPosicionAsync";
    var request = (HttpWebRequest)WebRequest.Create(url);
    request.Method = "POST";
    string json = $"{{\"usuario\":\"{Usuario}\",\"password\":\"{Pasword}\", \"listaUnidades\":[]}}";

    request.ContentType = "application/json";
    request.Accept = "application/json";
    using (var streamWriter = new StreamWriter(request.GetRequestStream()))
    {
        streamWriter.Write(json);
        streamWriter.Flush();
        streamWriter.Close();
    }
    try
    {
        using (WebResponse response = request.GetResponse())
        {
            using (Stream strReader = response.GetResponseStream())
            {
                if (strReader == null) { return ""; };
                using (StreamReader objReader = new StreamReader(strReader))
                {
                    string responseBody = objReader.ReadToEnd();

                    datosObtenidos = ConvertJsonToDatatable(responseBody);
                    foreach (DataRow row in datosObtenidos.Rows)
                    {
                        if (row[1].ToString().Equals(ListaUnidades))
                        {
                            Localizacion = row[8].ToString();
                        }
                    }
                }
            }
        }
    }
    catch (WebException ex)
    {
```

Figura 44 Método de Localización

5.4.3 Modificar

En el siguiente texto se menciona el método PUT y su relación con la clase PatiosCajas. La Figura 44 ilustra este proceso, mostrando cómo los datos enviados por la skill de Alexa son recibidos en formato JSON y enviados a la función "Modificar" en la clase PatiosCajas.

```
// PUT api/<controller>/5
public bool Put([FromBody]DatosPatios Patios)
{
    return PatiosCajas.Modificar(Patios);
}
```

Figura 45 Modificar

En este texto se describen los pasos necesarios para gestionar los datos recibidos en un archivo en formato JSON. La Figura 46 presenta de manera gráfica estos pasos, que incluyen la separación de los datos, la búsqueda en la base de datos de la caja, el tracto y el folio, la verificación de en qué patio agregar la caja, y la actualización de su ubicación en la base de datos.

```
public static bool Modificar(DatosPatios oUsuario)
{

    using (SqlConnection oConexion = new SqlConnection(Conexion.rutaConexion))
    {
        Obtener(oUsuario.Caja);
        ObtenerTracto(oUsuario.tracto);
        ObtenerFolio(oUsuario.Folio);
        SqlCommand cmd = new SqlCommand("SP_Patio", oConexion);
        cmd.CommandType = CommandType.StoredProcedure;
        cmd.Parameters.AddWithValue("@Folio", Fol);
        cmd.Parameters.AddWithValue("@Caja", caj);
        if (oUsuario.dPatioGve != "")
        {
            cmd.Parameters.AddWithValue("@dPatioGve", oUsuario.dPatioGve + "/" + trac);
            cmd.Parameters.AddWithValue("@dPatioSabritasObr", oUsuario.dPatioSabritasObr);
            cmd.Parameters.AddWithValue("@dPatioGamesaObr", oUsuario.dPatioGamesaObr);
            cmd.Parameters.AddWithValue("@dPatioObrKimberly", oUsuario.dPatioObrKimberly);
        }
        else if (oUsuario.dPatioSabritasObr != "")
        {
            cmd.Parameters.AddWithValue("@dPatioGve", oUsuario.dPatioGve);
            cmd.Parameters.AddWithValue("@dPatioSabritasObr", oUsuario.dPatioSabritasObr + "/" + trac);
            cmd.Parameters.AddWithValue("@dPatioGamesaObr", oUsuario.dPatioGamesaObr);
            cmd.Parameters.AddWithValue("@dPatioObrKimberly", oUsuario.dPatioObrKimberly);
        }
        else if (oUsuario.dPatioGamesaObr != "")
        {
            cmd.Parameters.AddWithValue("@dPatioGve", oUsuario.dPatioGve);
            cmd.Parameters.AddWithValue("@dPatioSabritasObr", oUsuario.dPatioSabritasObr);
            cmd.Parameters.AddWithValue("@dPatioGamesaObr", oUsuario.dPatioGamesaObr + "/" + trac);
            cmd.Parameters.AddWithValue("@dPatioObrKimberly", oUsuario.dPatioObrKimberly);
        }
        else if (oUsuario.dPatioObrKimberly != "")
        {
            cmd.Parameters.AddWithValue("@dPatioGve", oUsuario.dPatioGve);
            cmd.Parameters.AddWithValue("@dPatioSabritasObr", oUsuario.dPatioSabritasObr);
            cmd.Parameters.AddWithValue("@dPatioGamesaObr", oUsuario.dPatioGamesaObr);
            cmd.Parameters.AddWithValue("@dPatioObrKimberly", oUsuario.dPatioObrKimberly + "/" + trac);
        }
        if(oUsuario.ColorActivo == "SABRITAS")
        {
            cmd.Parameters.AddWithValue("@ColorActivo", "GOLD");
        }
        else if(oUsuario.ColorActivo == "GAMESA")
        {
```

Figura 46 Estructura de modificar

En el siguiente texto se describen las condiciones que determinan el color de identificación de la caja en la base de datos. La Figura 47 ilustra este proceso, que incluye la selección del color dependiendo de los datos que contenga la caja, y el envío de la información a la base de datos para su correcta modificación en caso de que no haya errores.

```
else if(oUsuario.ColorActivo == "GAMESA")
{
    cmd.Parameters.AddWithValue("@ColorActivo", "LightBlue");
}
else if (oUsuario.ColorActivo == "AZUCAR")
{
    cmd.Parameters.AddWithValue("@ColorActivo", "LightGreen");
}
else if (oUsuario.ColorActivo == "VACIO")
{
    cmd.Parameters.AddWithValue("@ColorActivo", "HotPink");
}
else if (oUsuario.ColorActivo == "OTROS")
{
    cmd.Parameters.AddWithValue("@ColorActivo", "LIGHTSALMON");
}
try
{
    oConexion.Open();
    cmd.ExecuteNonQuery();
    return true;
}
catch (Exception ex)
{
    return false;
}
```

Figura 47 Selección de colores

5.5 Creación de skill.

Una vez adquiridos los componentes necesarios para la creación del sistema, se procede a la creación de la Skill.

En la siguiente figura 68 se muestra la plataforma Alexa Developer que permite desarrollar habilidades o Skills para el asistente virtual de comando de voz Alexa. Para crear una Skill, primero se debe acceder a Alexa Developer y luego seguir los pasos necesarios para su creación y activación.

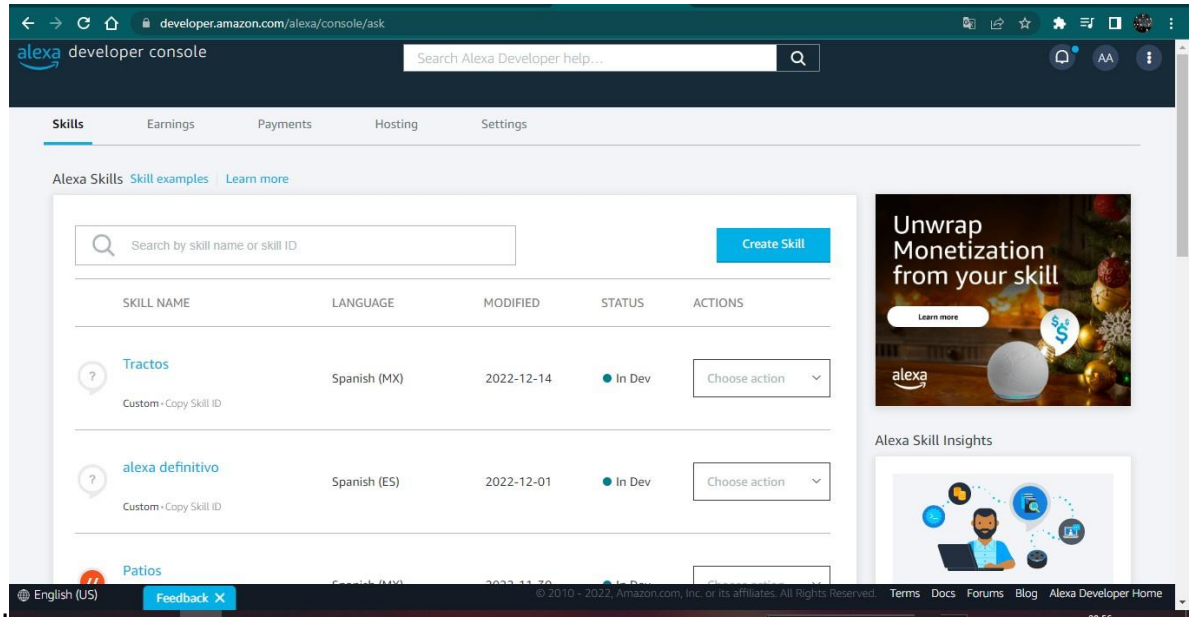


Figura 48 Plataforma de Alexa Developer.

Una vez ingresado dentro de la plataforma se crea lo que es la Skill la cual se va a programar para solicitar una petición deseada, en este caso se crea la Skill que es Tractos dentro de ella se programara las funcionalidades.

5.5.1 Creación de Invocations - Intents.

En la siguiente figura 69 se muestra la invocations creada que permitirá abrir la Skill al momento de solicitar una petición.

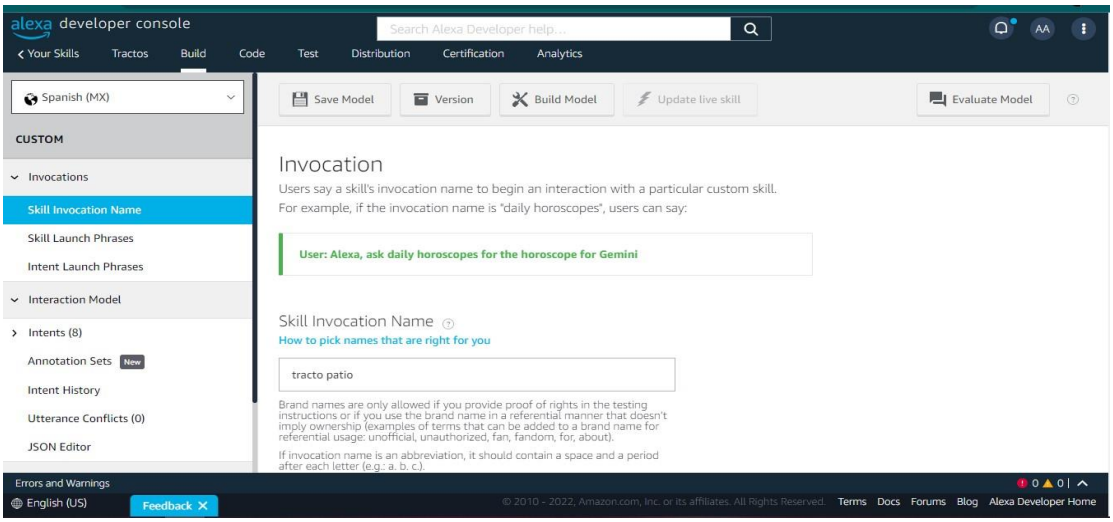


Figura 49 Creación de Invocations.

Una vez ya creada la invocación, esta se abrirá con el nombre registrado, para ello se toma en cuenta que nombre poner para evitar confusiones al asistente de voz.

En la siguiente figura 70 se crearon lo que son Intents, estos son las frases o palabras claves que el asistente de voz va a reconocer y así realizar una consulta de lo que se está solicitando, es por eso que al momento de crear las frases tenían que ser claras y precisas para que la asistente entendiera lo que se está solicitando. También, se creó las variables que guardaran las palabras claves que son para realizar la consulta.

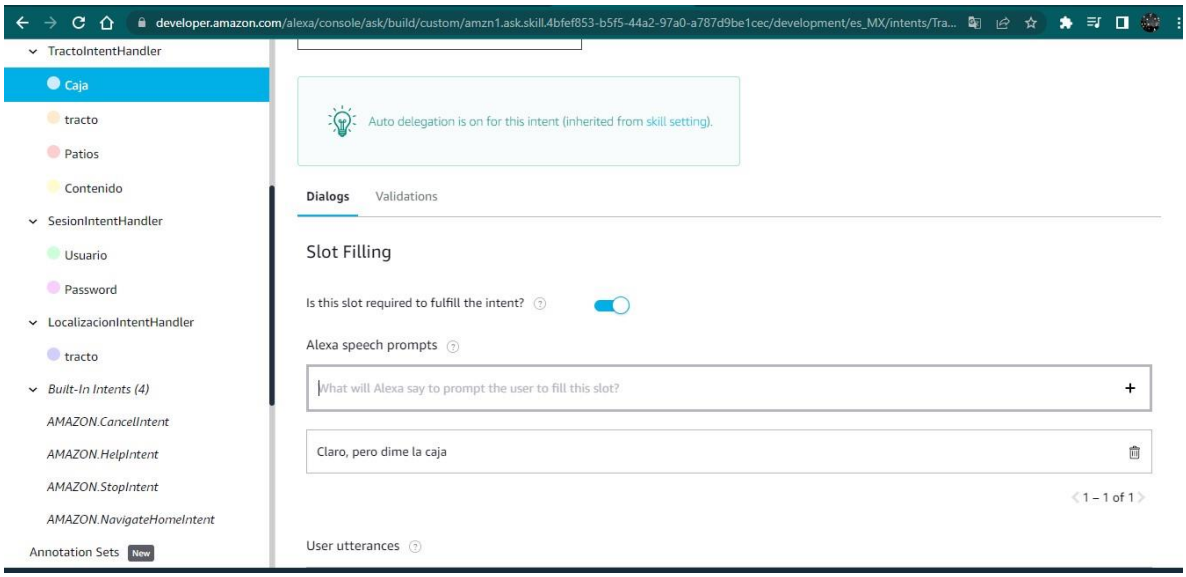


Figura 50 Creación de Invocations – Intents

5.5.2 Programación de Inicio de sesión.

En la siguiente figura 71 se muestran cuatro variables globales la cual primera variable es una librería predeterminada que ya viene en el código, la segunda y tercera es la librería de axios las cuales son muy importantes ya que con ellas permitirán consumir la API REST y la última variable se utilizara para el inicio de sesión

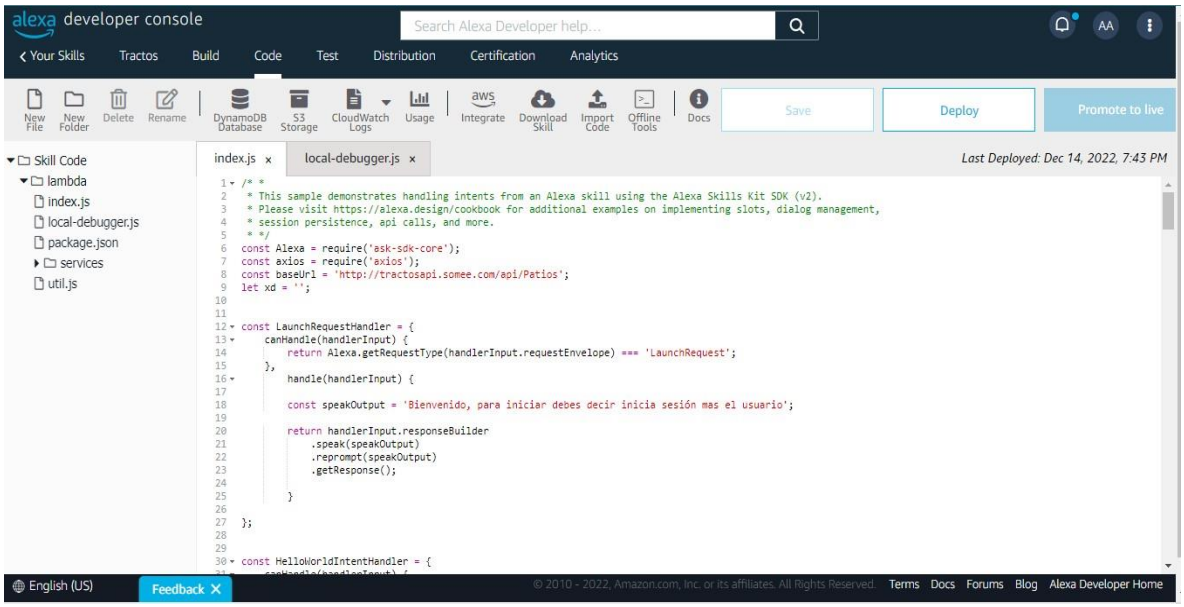


Figura 51 Conexión a la API.

En la siguiente figura 72 se muestra el LaunchRequestHandler donde se hace el proceso de inicio de sesión, que cuando inicia la skill la Alexa te pide ingresar tu usuario y contraseña para enviarle las claves a la API REST, la cual le regresaría un sí o un no dependiendo si son correctas las claves

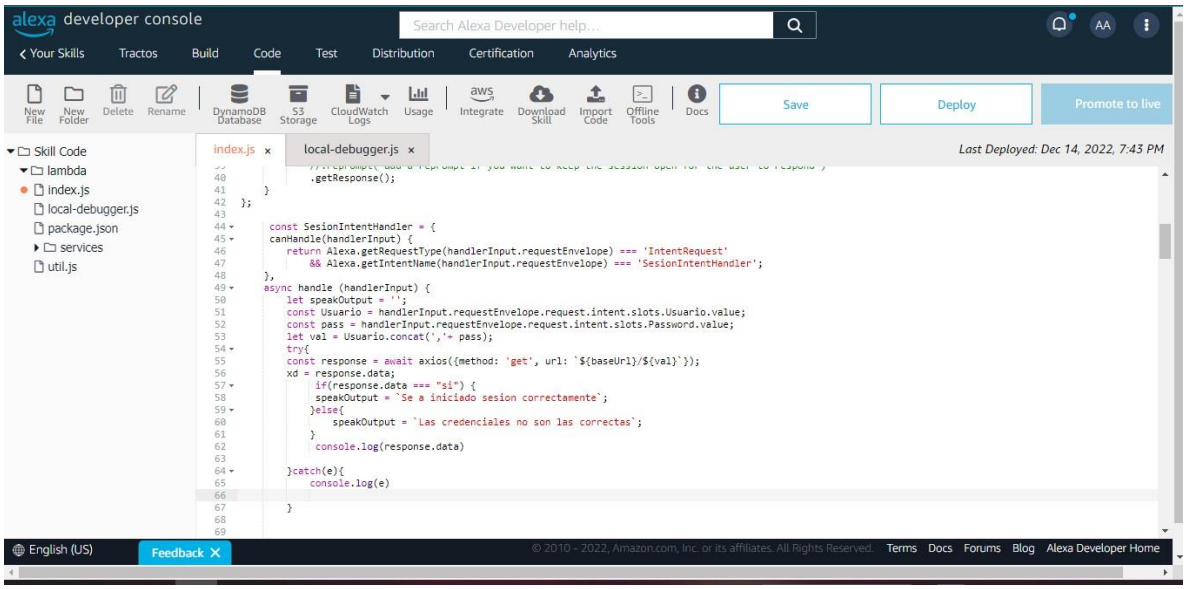


Figura 52 Programación de inicio de sesión

5.5.3 Programación de modificar el patio o actualizar la información.

En la siguiente figura 73 se muestra el intents TractoIntentHandler que toma todos los datos para modificar el patio o actualizar la información, después hay varias condiciones if que se utilizan para saber en qué patio se va a modificar o actualizar y el producto que trae para identificar su color, luego que se hace todo ese proceso de validación, se toman todos los datos y se envían en la cadena de axios para enviar los datos a la API REST donde esta procesara los datos e guardara en la base de datos, si sale un error dirá que ha sucedido un error con el servicio pero si todo sale correctamente mostrara el mensaje que todo se ha guardado correctamente.

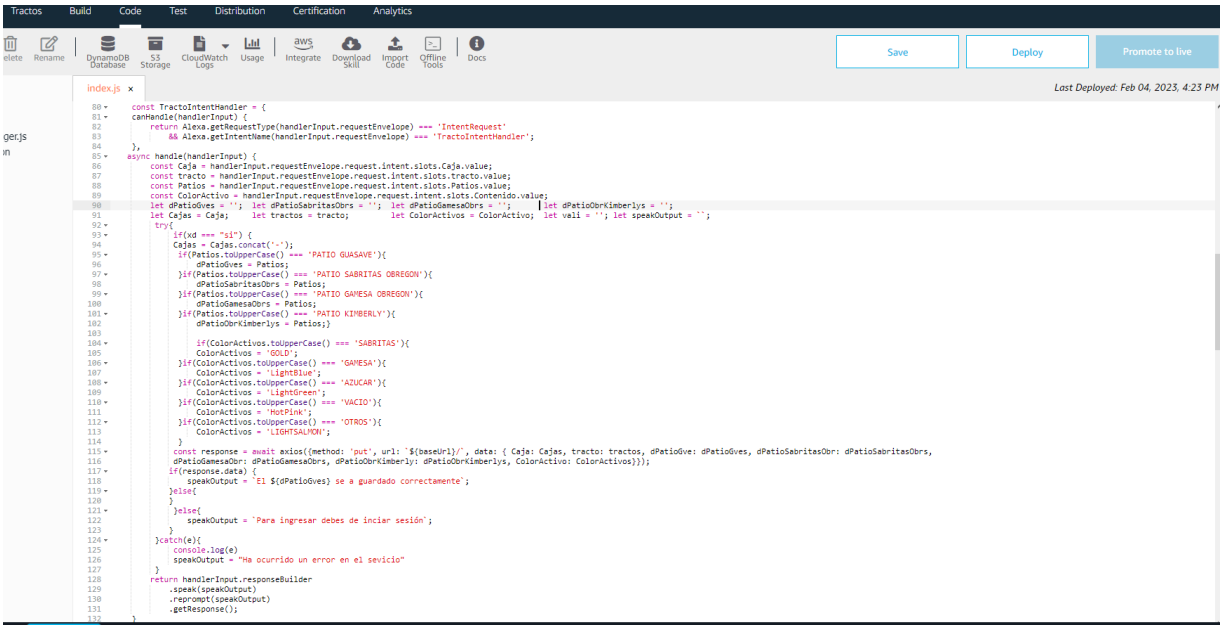


Figura 53 programación de modificar patio o actualizarlo

5.5.4 Programación de localización de un remolque de carga.

En la siguiente figura 74 se muestra el intents LocalizacionIntentHandler el cual solo tomo los primeros dos números de la caja para unirlos ingresarlo a la cadena de conexión axio la cual le enviara la información a la API REST, para que este la busque y si encontró información le enviaría la localización a tiempo real y si no le respondería que la caja esta no se encontró.

```
const LocalizacionIntentHandler={
    canHandle(handlerInput) {
        return Alexa.getRequestType(handlerInput.requestEnvelope) === 'IntentRequest'
            && Alexa.getIntentName(handlerInput.requestEnvelope) === 'LocalizacionIntentHandler';
    },
    async handle (handlerInput){
         let speakOutput = '';
              const localizar = handlerInput.requestEnvelope.request.intent.slots.tracto.value;
              let ubi = localizar;
              const response = await axios({method: 'get', url: `${baseUrl}/${ubi}`});
              speakOutput = response.data;

     return handlerInput.responseBuilder
            .speak(speakOutput)
            .reprompt(speakOutput)
            .getResponse();
}};
```

Figura 54 Programación de localización

VI ANÁLISIS DE RESULTADOS Y DISCUSIÓN.

Al realizar el análisis del proceso transporte para la empresa CMC Transporta se tuvo como resultado conocer cada uno de los pasos que ponen dicho proceso, desde la asignación de viajes hasta la actualización de la información al momento de terminar el viaje.

De dicho análisis se obtuvo como resultado datos sobre el tiempo que se tarda en generar los procesos y los errores de estos.

6.1 Pizarra digital

A continuación, se muestra la figura de la pizarra digital donde se hacen todos los movimientos, lo cual se detecta que a simple vista es difícil comprender cual es el uso correspondiente de este.

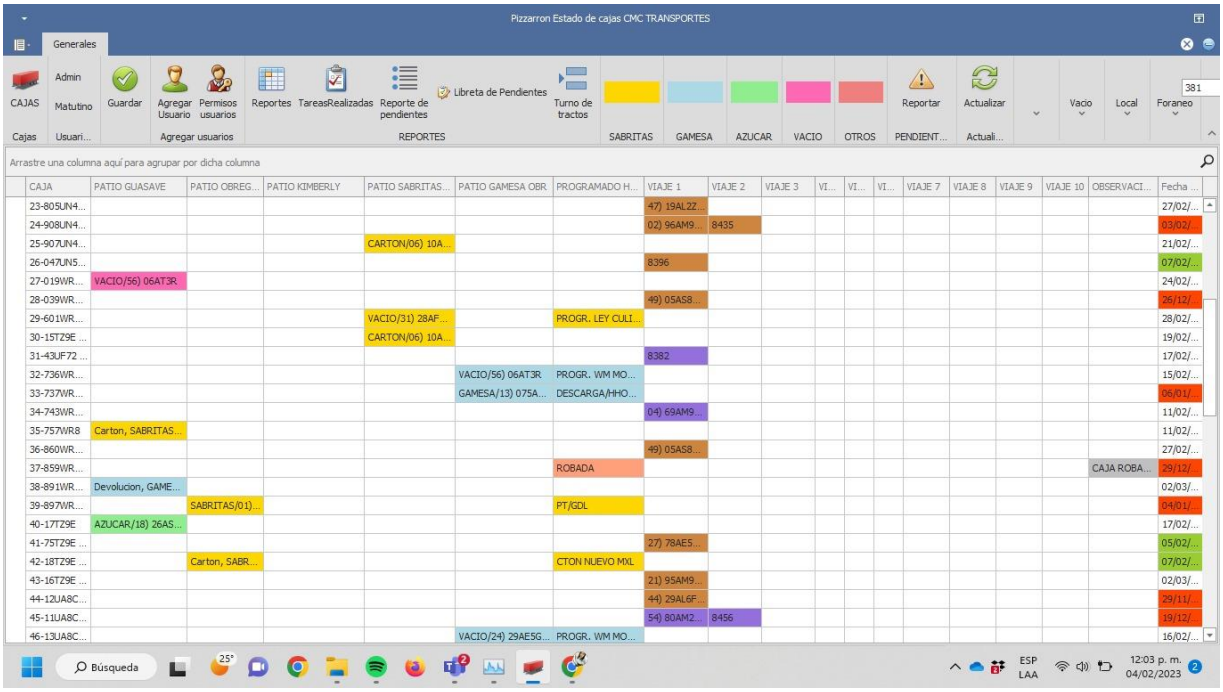


Figura 55 Pizarra digital

6.2 Software

Tras el estudio de la pizarra digital de CMC Transporta se obtuvo como resultado una Api rest que es consumida por una skill de Alexa.

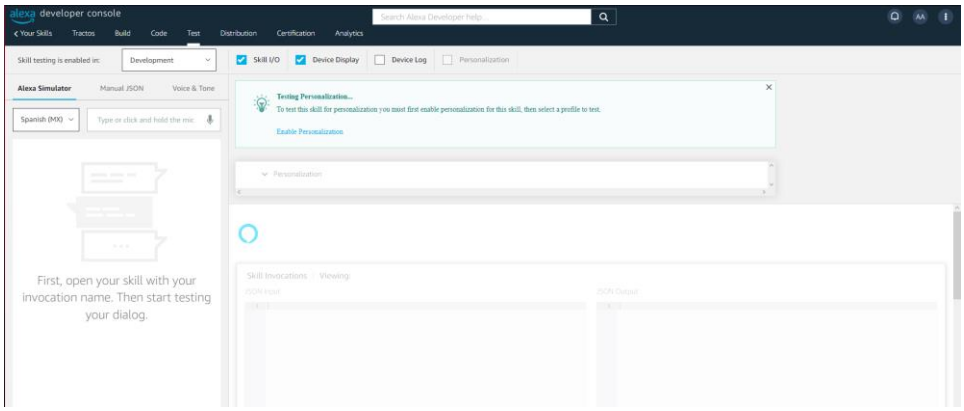


Figura 56 Pantalla de prueba de skill

Se obtuvo como resultado un programa de lenguaje natural (skill de Alexa) automatizada por la problemática que representa el manejo y llenado de la pizarra digital, ya que pueden suceder errores humanos los cuales afectaran el funcionamiento de esta, ocasionando que fallos que se tengan que solucionar manualmente en la base de datos por el encargado en sistemas.

6.3 Evaluación de aplicaciones

Cabe mencionar que la skill de Alexa funciona consumiendo una API REST que consume la base de datos de la empresa así haciendo todo el proceso directamente en la API REST, para entregarle todos los datos limpios de algún error a la base de datos.

6.3.1 Asignación de viajes

La asignación de viajes es un proceso largo pues conlleva la búsqueda de la caja que este vacía en la pizarra para luego dar clic derecho para que muestre una sección en la cual se elegirá el camión que llevara la caja como se muestra en la siguiente figura:

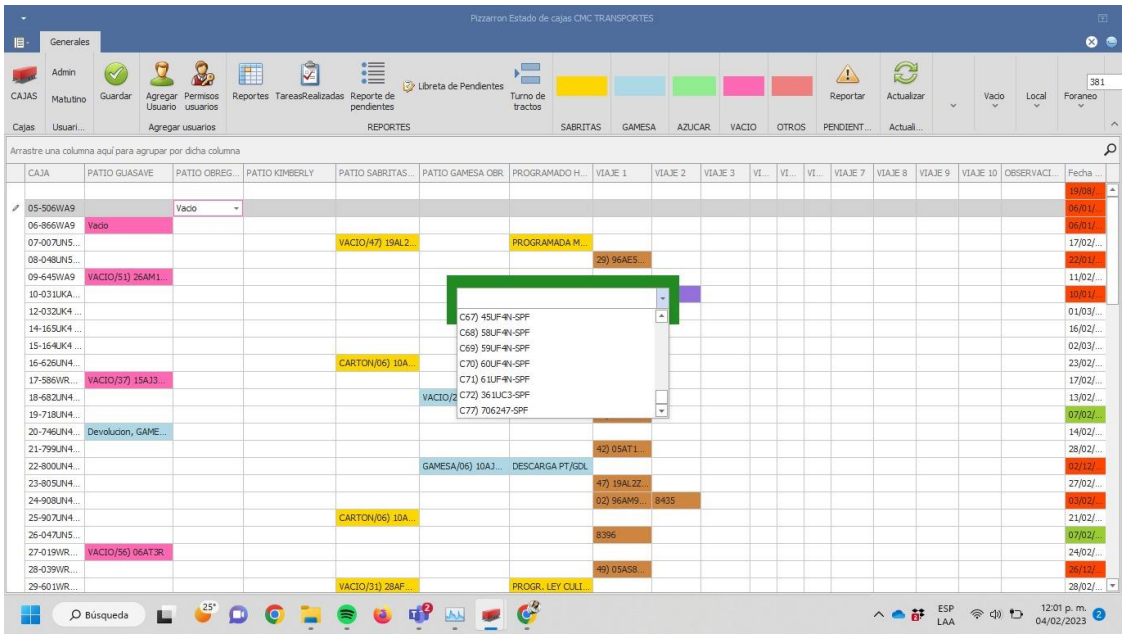


Figura 57 Asignación de camión en pizarra digital

Luego nos abre una ventana emergente donde se debe responder por medio del teclado un formulario de la información básica del viaje.

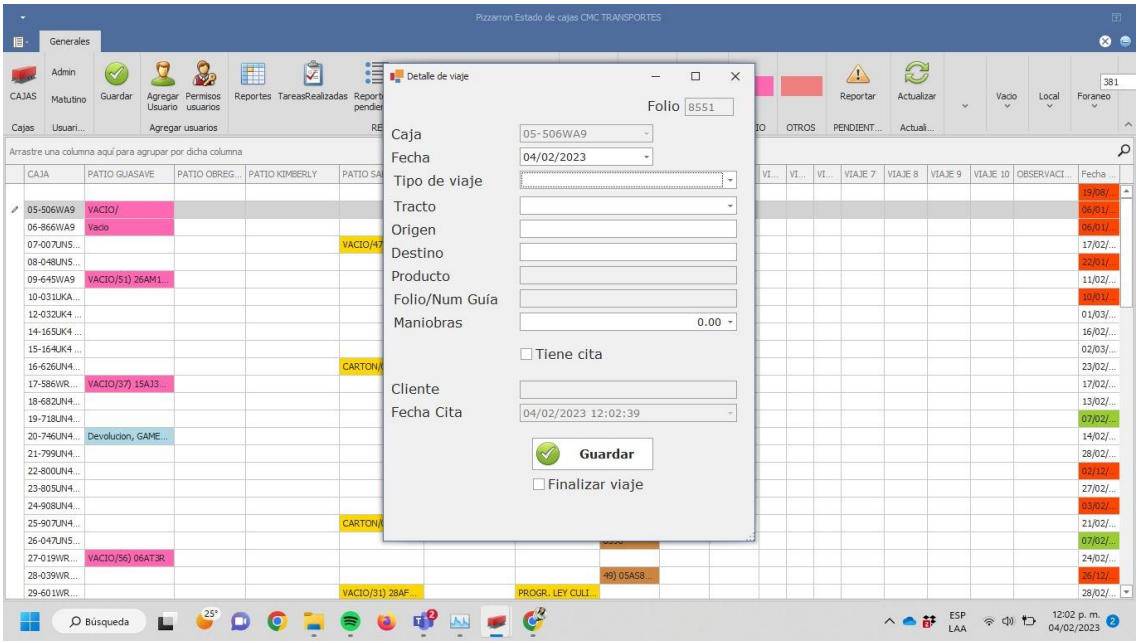


Figura 58 Formulario de asignación de viaje en pizarra digital

Viendo que actualmente el proceso de asignación de viaje se hace de manera manual, Ahora con la skill de Alexa este proceso se realizara automáticamente por comando de voz, En la siguiente figura se mostrara el proceso visual de como seria el proceso para asignar un viaje:

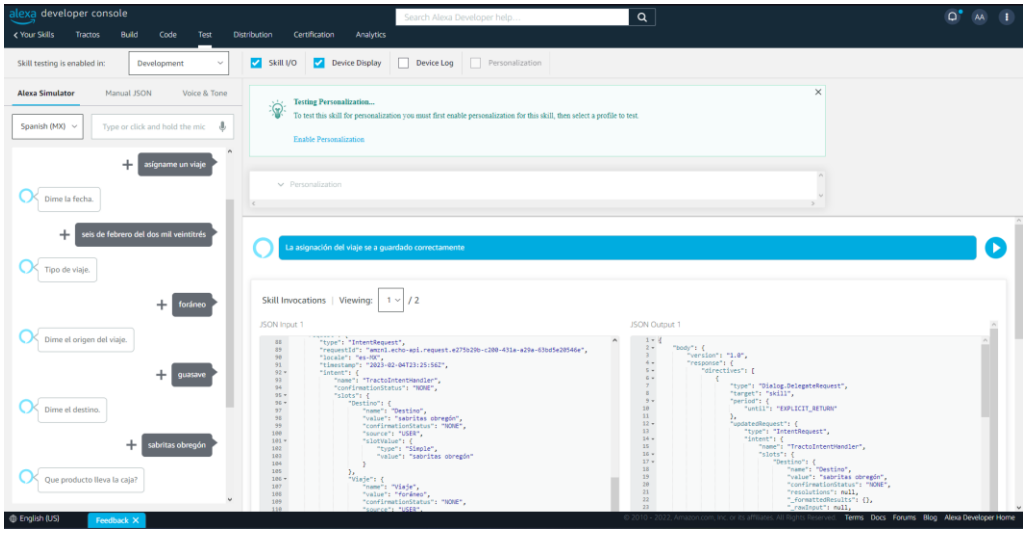


Figura 59 Asignación de viaje

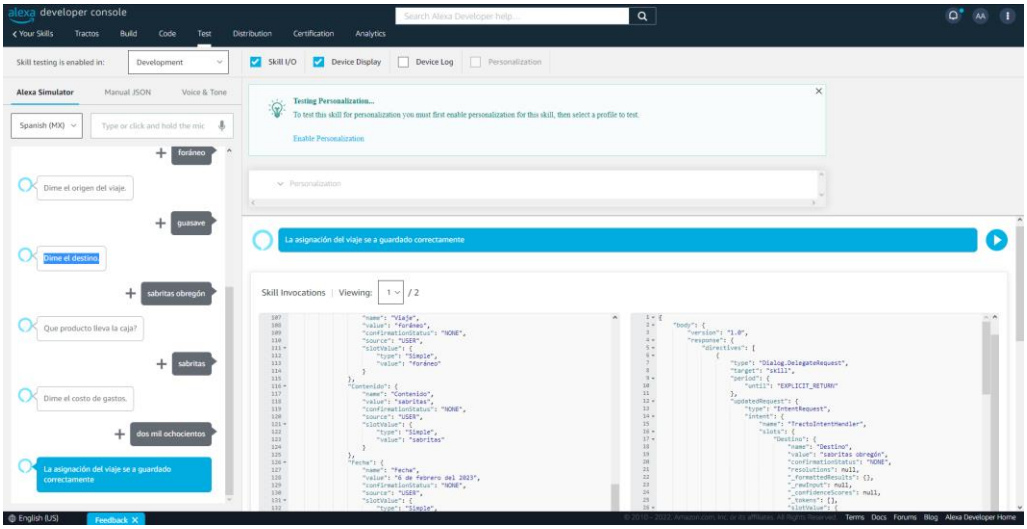
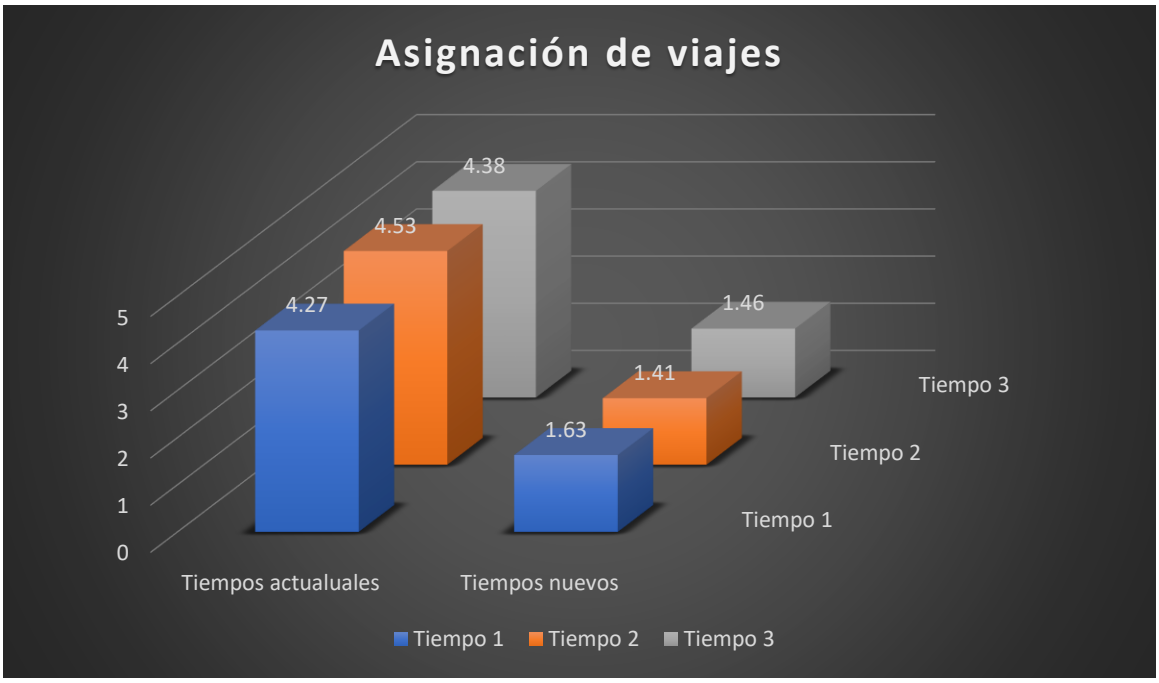


Figura 60 asignación de viaje (Parte 2)

En la siguiente grafica se muestran las pruebas que se hicieron para comprobar las diferencias en tiempo de los tres intentos que se hicieron, logramos ver que en los

tiempos de la asignación de viaje actual tarda más de cuatro minutos en hacerla y los tiempos pueden variar dependiendo que persona utilice el programa ya que puede ser tedioso a diferencia de la skill de Alexa solo es utilizar el comando y ella misma te estaría indicando los datos que ocupa para asignar el viaje, por lo cual el tiempo es muy reducido ya que la Alexa solo es solo un intermediario de recepción de datos y cuando los obtiene se los envía a la API REST que esta los limpiara para pasarlos a la base de datos sin errores.



6.3.2 Localización

Para acceder a la organización especifica de un remolque es necesario iniciar sesión en una plataforma web donde se hace la búsqueda de los remolques navegando por Google Maps o bien recorrer la lista de los nombres de las cajas registradas. Una vez que se selecciona la caja se debe de solicitar a la plataforma un reporte y de esta manera conocerán el historial de la caja y ubicación actual lo cual puede llegar a ser tardado.

Pero la empresa ofrece una API REST la cual se puede consumir para desglosar la localización de cada caja a tiempo real. En la siguiente figura se muestra la

localización al momento de ser utilizada por la Alexa, así haciéndose independiente de la empresa y obteniendo los datos en segundos:

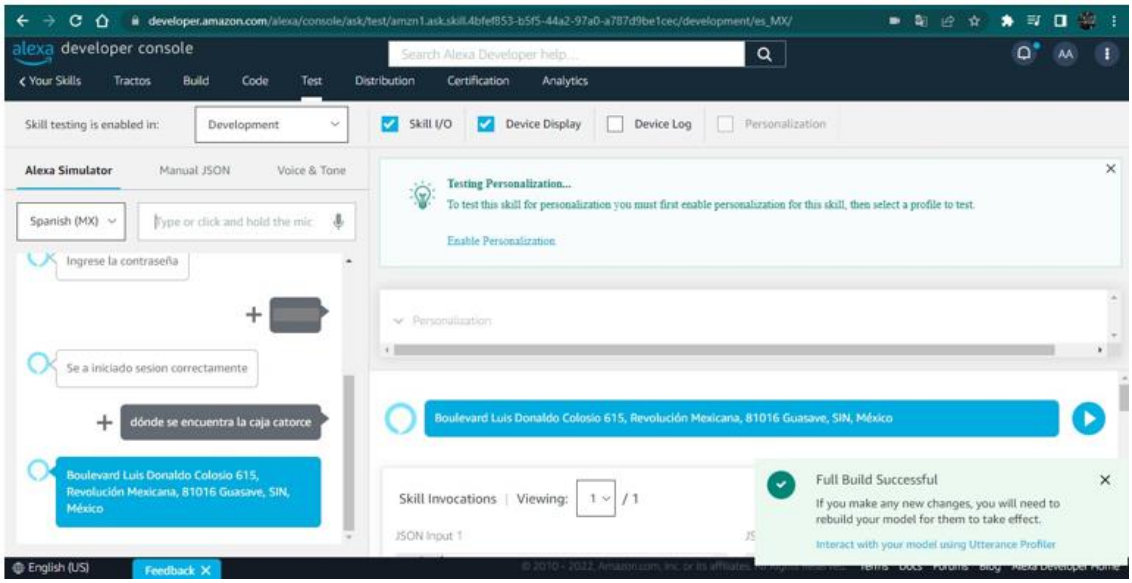


Figura 61 localización

6.3.1 Cambio de patio y actualización de información

Para realizar el cambio de patio o actualización de información se debe de hacer clic en el espacio del patio al que quiere moverlo en la pizarra para poder seleccionar el contenido de la caja como se muestra en la siguiente figura:

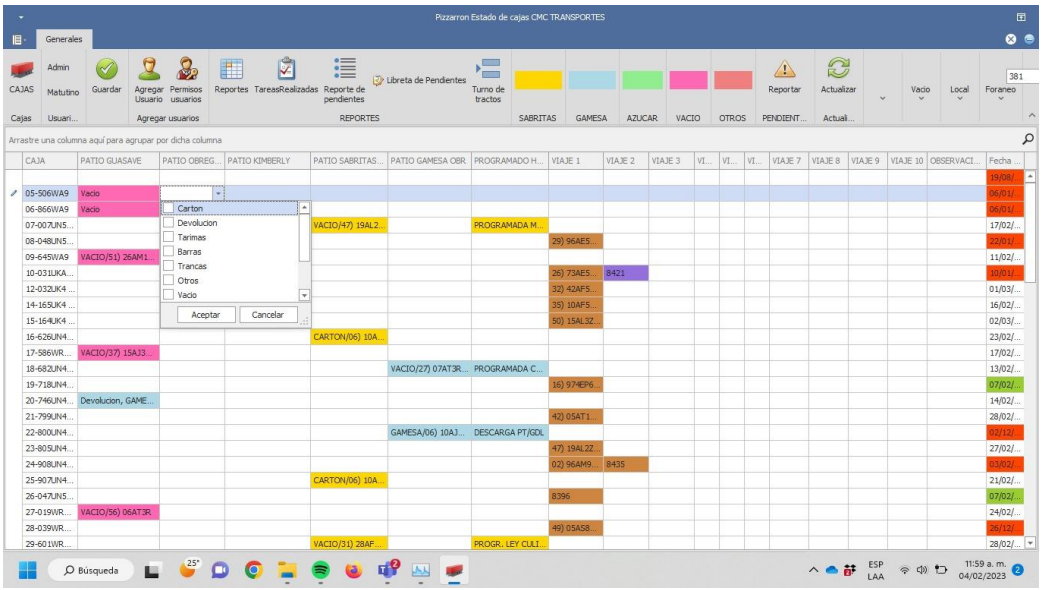


Figura 62 Selección de contenido

Al ya haber seleccionado el contenido de la caja la pizarra mostrara los camiones y se debe de buscar la placa del camión que dejo la caja, como en la siguiente figura:

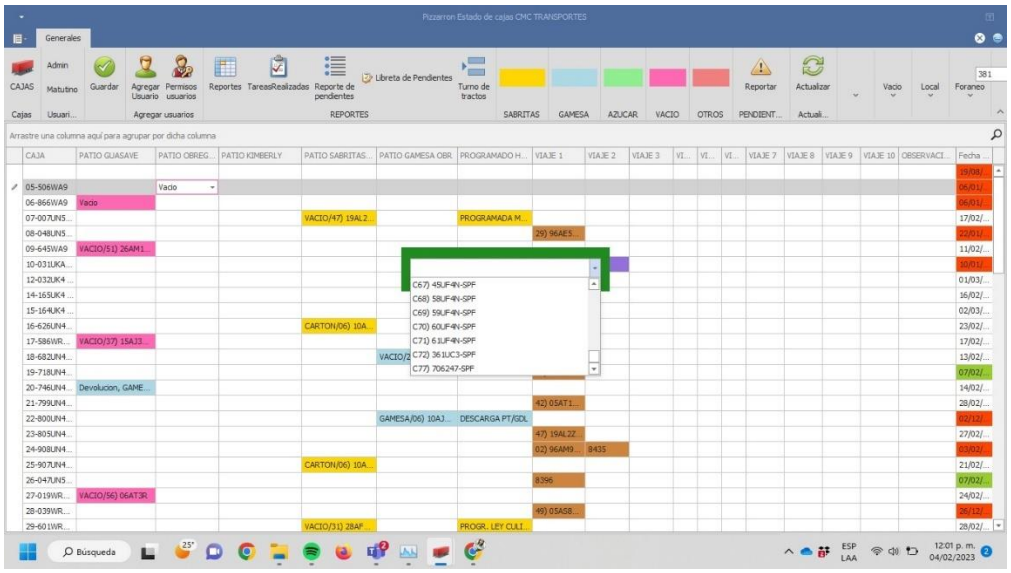
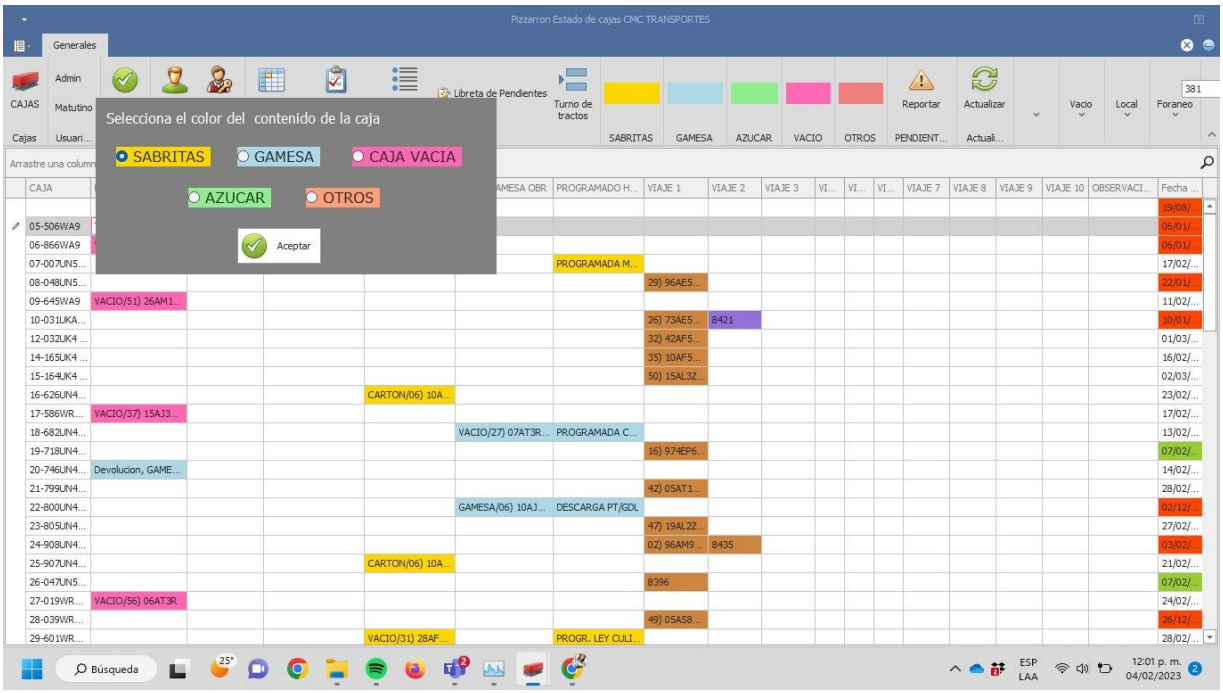


Figura 63 Seleccionar placa

Por último, se mostrará el apartado de colores para seleccionar el tipo de contenido que trae la caja, como se muestra en la siguiente figura:



El proceso de cambio de patio y actualización de información también se hace de manera manual, por lo cual es susceptible a errores humano, por lo cual también este proceso se realizará automáticamente por comando de voz con la skill de Alexa, En la siguiente figura se mostrará el proceso visual:

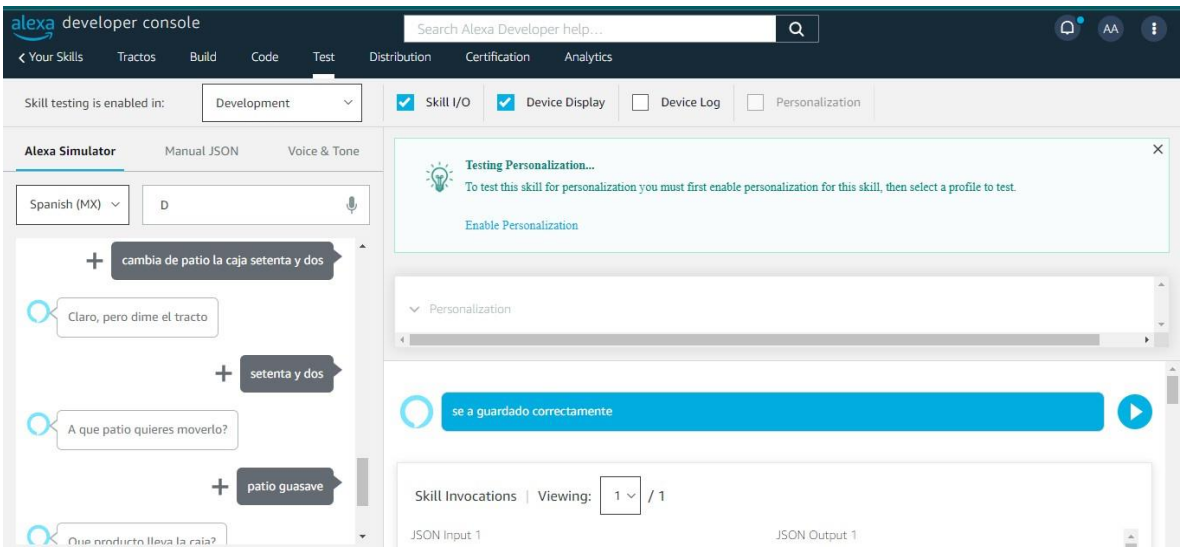


Figura 64 Cambio de patio

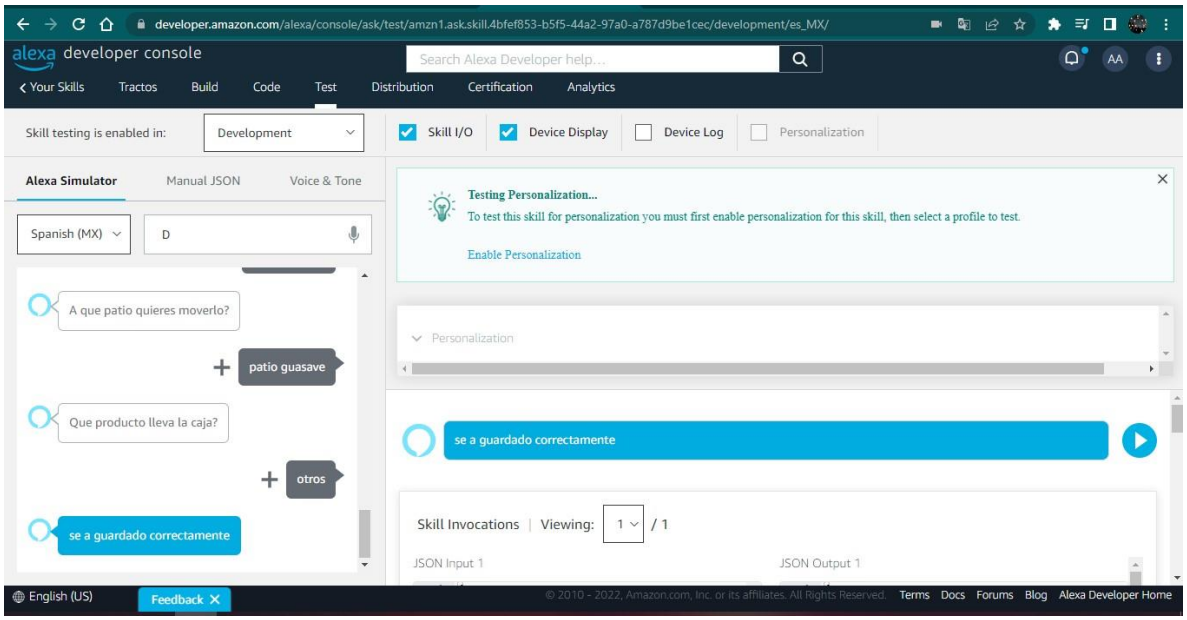
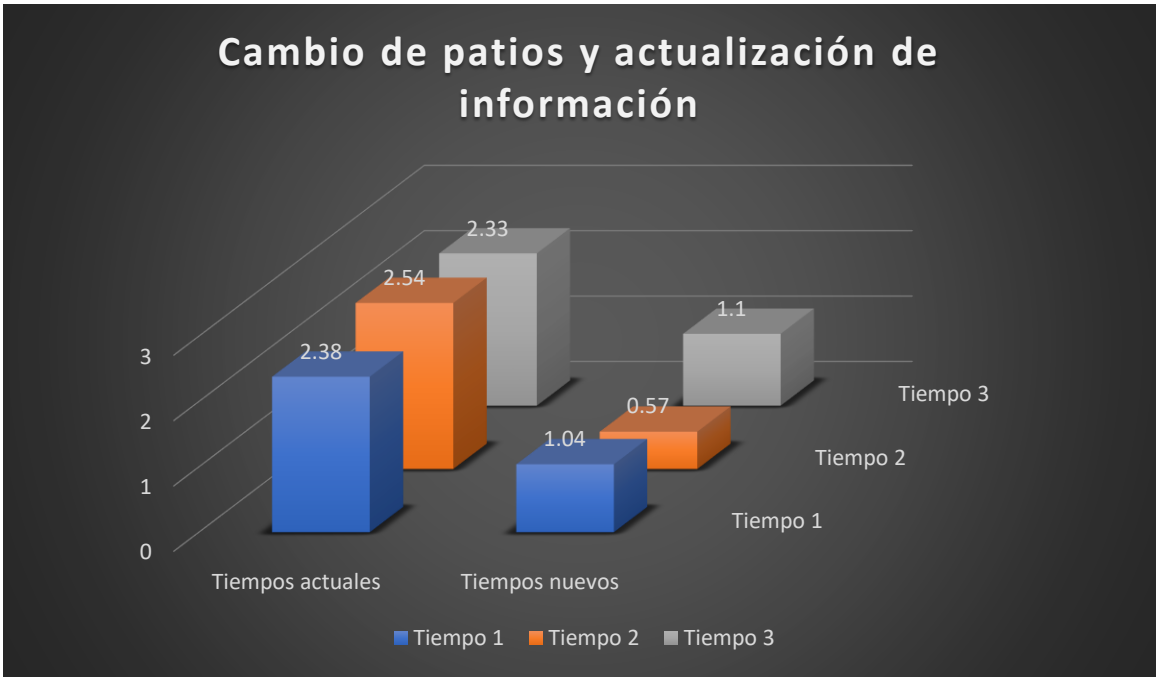


Figura 65 Cambio de patio (Segunda parte)

En la siguiente grafica se muestran los tiempos que se necesitan para realizar estas acciones en la pizarra digital y con los comandos de voz en Alexa, se nota la reducción de tiempos en los diferentes procesos.



Pero lo más importante es la reducción de errores, como con la skill de Alexa se utiliza el lenguaje natural no hay necesidad de preocuparse por los datos erróneos, ya están automatizadas las respuestas y si los datos son diferentes a los que se espera botaría una advertencia que no dejaría insertar los datos.

VII CONCLUSIONES Y RECOMENDACIONES.

En conclusión, luego de llevar a cabo un exhaustivo estudio y evaluación, se ha demostrado la viabilidad e importancia de implementar una API REST en la empresa CMC Transporta. El objetivo de este proyecto de investigación tecnológica era desarrollar una solución para la gestión eficiente de las cajas y la organización del tráfico y logística en la empresa.

Se identificaron los desafíos existentes en la actualidad en la pizarra digital utilizada por la empresa y se propuso la utilización de una API REST para superarlos. Esta tecnología permitirá mejorar la eficiencia y la rentabilidad de la empresa al brindar una gestión más efectiva y controlada de sus operaciones.

Para futuras mejoras se podría terminar de automatizar todos los procesos de la pizarra digital y buscar unir la API REST de localización a tiempo real, la actualización de información y cambio de patio, para que se realice todo el proceso directamente en la base de datos, para que no dependa del capital humano.

7.1 Recomendaciones:

- Se recomienda seguir las indicaciones de uso adecuadamente para obtener los resultados esperados y maximizar los beneficios de esta nueva tecnología. Continuar monitoreando y evaluando el desempeño de la API REST implementada para asegurar su eficacia y eficiencia en la gestión de la información de localización de los remolques.

- Se recomienda implementar medidas de seguridad adecuadas para proteger la información de localización sensibles y garantizar la privacidad de los usuarios.

- Es importante considerar la escalabilidad y la seguridad de la API REST a medida que el número de remolques y la cantidad de información de localización aumenten.

- Considerar la posibilidad de integrar la API REST con otros sistemas de la empresa para mejorar la eficiencia en la gestión de la información.
- Capacitar al personal encargado de manejar la API REST para asegurar su correcto uso y mantenimiento.
- Mantener actualizada la documentación de la API REST para facilitar su uso y mantenimiento.
- Evaluar periódicamente las necesidades de la empresa en cuanto a la gestión de la información de los remolques y considerar posibles mejoras a la API REST en caso de ser necesario.

7.2 COMPETENCIAS DESARROLLADAS

Durante el proceso de investigación e implementación desarrolle las competencias tales como son:

- Capacidad de organizar y planificar.
- Conocimientos básicos de la carrera.
- Comunicación oral y escrita.
- Habilidad para buscar y analizar información proveniente de fuentes diversas.
- Solución de problemas.
- Entender el funcionamiento de una API REST y su importancia en la gestión de información en la era digital.
- Desarrollar habilidades en la programación y uso de lenguajes de programación como HTML, JavaScript y Python.
- Conocer las mejores prácticas en la creación de una API REST, incluyendo la seguridad, la escalabilidad y la compatibilidad con otros sistemas.
- Desarrollar habilidades en la gestión de bases de datos y en la integración de la API con diferentes sistemas y aplicaciones.
- Aprender a trabajar con herramientas y plataformas en la nube.

- Comprender la importancia de la documentación en la creación de una API REST y aprender a documentar de manera clara y concisa.
- Adquirir conocimientos sobre el modelo de arquitectura de la API y su implementación en diferentes proyectos.

En general, esta experiencia me brindó una visión más amplia del mundo de la tecnología y su capacidad para transformar la forma en que las empresas operan y compiten en el mercado.

Me dejó con un sentido de logro y confianza en mis habilidades técnicas y una motivación para seguir aprendiendo en el campo.

Estoy seguro de que estos conocimientos y habilidades serán valiosos para mí en mis futuros proyectos y en mi carrera profesional en general.

VIII REFERENCIAS.

Asier, M. (11 de 04 de 2013). *https://asiermarques.com/2013/conceptos-sobre-apis-rest/*. Obtenido de Conceptos sobre APIs REST: https://asiermarques.com/2013/conceptos-sobre-apis-rest/

Eugenia, M. (20 de 01 de 2023). *hubspot*. Obtenido de hubspot: https://blog.hubspot.es/website/que-como-usar-api

IBM Cloud Education. (6 de 04 de 2021). *IBM*. Obtenido de API REST: https://www.ibm.com/mx-es/cloud/learn/rest-apis

Marín, R. (2019). Los gestores de bases de datos más usados en la actualidad. *Inesem*, https://www.inesem.es/revistadigital/informatica-y-tics/los-gestores-de-bases-de-datos-mas-usados/.

Mh Education. (1 de 2 de 2018). *Sistemas gestores de.* Obtenido de Mh Education: https://www.mheducation.es/bcv/guide/capitulo/8448148797.pdf

Nicolás, M. (04 de 05 de 2018). *Oleoshop*. Obtenido de Principales asistentes de voz y en qué se diferencian: https://www.oleoshop.com/blog/asistentes-de-voz

Oracle. (2023). *Oracle Mexico*. Obtenido de ¿Qué es una base de datos?: https://www.oracle.com/mx/database/what-is-database/

Paz, J. (2020). *UNA GUÍA PARA LA INTEGRACIÓN DE ASISTENTES COMO ALEXA O GOOGLE ASSISTANT PARA MEJORAR EL DÍA A DÍA EN EL HOGAR DEL MAYOR.* Salamanca: universidad de salamanca editorial. Obtenido de https://emeriti.usal.es/guia/guia_asistentes_v1.pdf: https://emeriti.usal.es/guia/guia_asistentes_v1.pdf

Plaza Sheila, R. N. (2015-2016). *API de servicios web.* Madrid: Facultad de Informática.

Polyakov, E. M. (2018). *Investigation and development of the intelligent voice assistant for the Internet of Things using machine learning.* Moscow, Russia.: Moscow Workshop on Electronic and Networking Technologies.

Pradhan, A. M. (2018). *"Accessibility Came by Accident": Use of Voice-Controlled Intelligent Personal Assistants by People with Disabilities.* College Park: University of Maryland.

redhat. (20 de 01 de 2023). *redhat*. Obtenido de ¿Qué es una API y cómo funciona?: https://www.redhat.com/es/topics/api/what-are-application-programming-interfaces

rootstack. (27 de 04 de 2022). *rootstack*. Obtenido de ¿Cuál es la mejor tecnología para construir API REST?

Yasmani, T. (10 de 05 de 2022). *saasradar*. Obtenido de Las 7 mejores tecnologías para el desarrollo de API REST: https://saasradar.net/desarrollo-api-rest/

IX Anexo

9.1 Subir API REST A Soome

Se publico la aplicación en Soome, un hosting que entre sus múltiples servicios nos proporciona un alojamiento gratuito para aplicaciones .NET.

Lo que necesitas tener es:

- La aplicación web hecha en .NET
- Se utilizo Visual Studio 2019
- Microsoft SQL Server Management Studio
- Un correo electrónico válido

En este caso, se tiene la aplicación realizada con .NET Core, Entity Framework.

9.1.1 Ingresar a Soome

En la siguiente figura 48 se muestra la página web del hosting Soome, en la cual se subirá la API REST

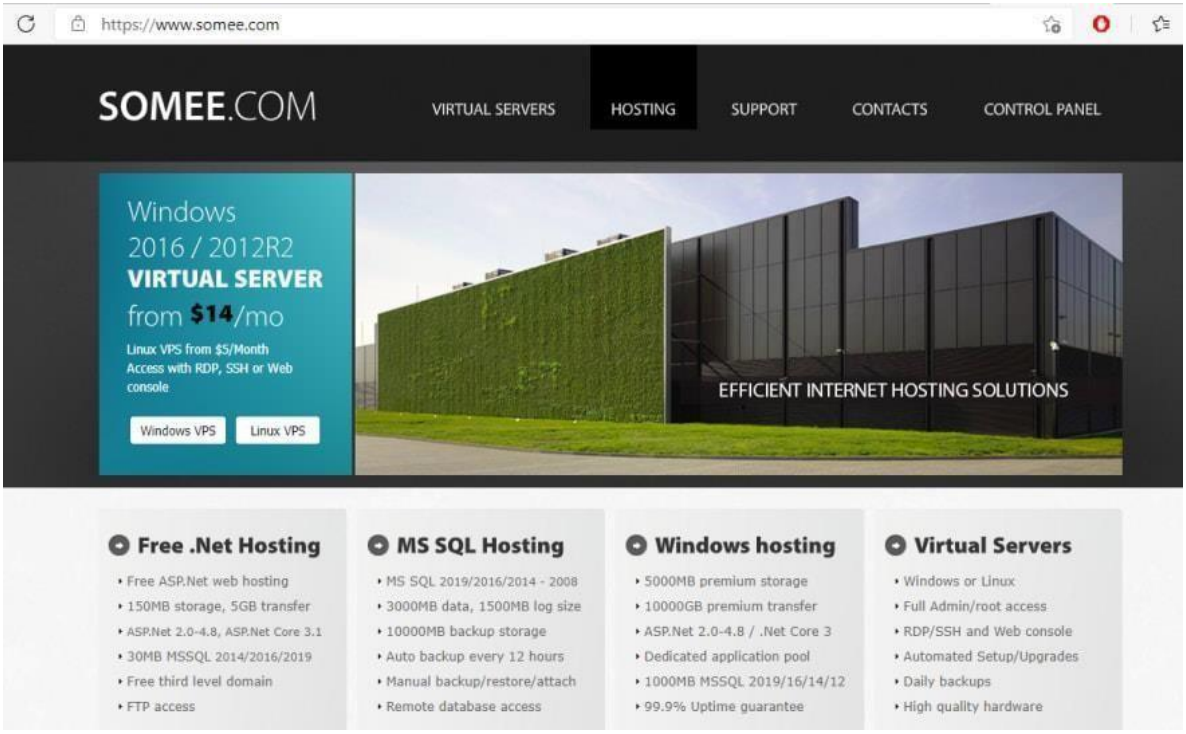


Figura 66 SOME.COM

Aquí podemos observar en la figura 49 en la parte remarcada, esta es la opción que vamos a utilizar Free .Net Hosting y damos clic en Learn More o leer más.

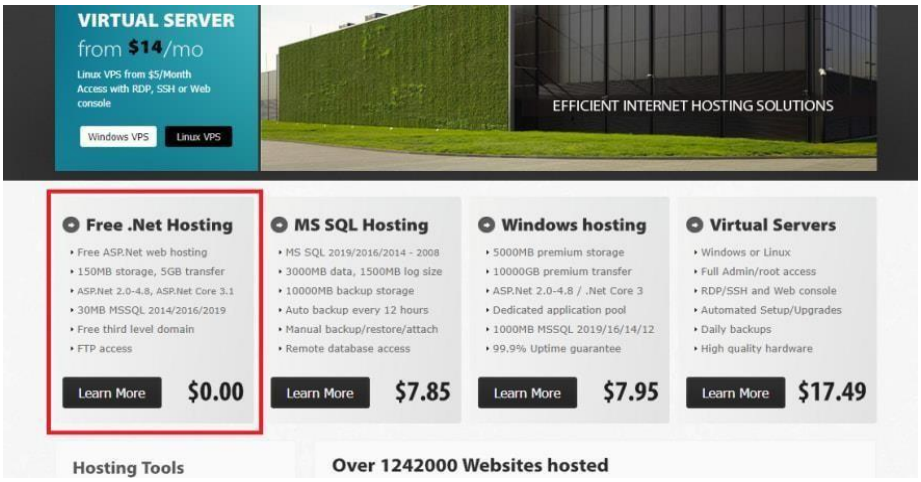


Figura 67 Free .Net Hosting

En la siguiente figura 50 nos da algunas especificaciones acerca del plan gratuito. Damos Clic en Order now

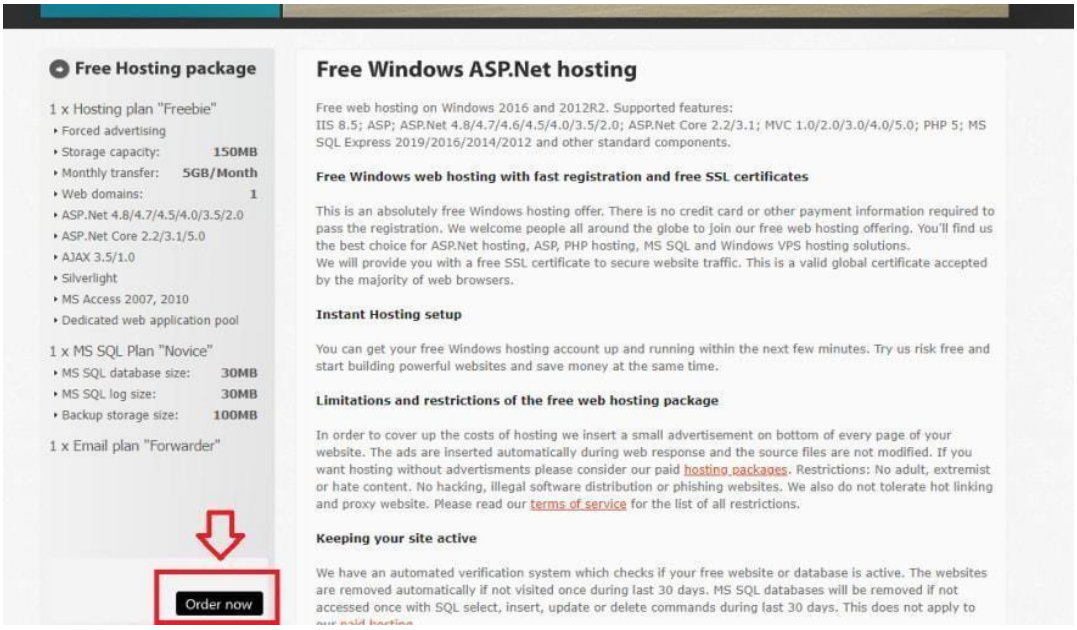


Figura 68 Plan Gratuito

En la siguiente figura 51 inmediatamente nos muestra la ventana de ingreso o registro, aquí debemos llenar todos los datos y crear una nueva cuenta, posteriormente te envía un correo con un código para que puedas seguir el proceso, por eso es importante que tengas esa cuenta activa.

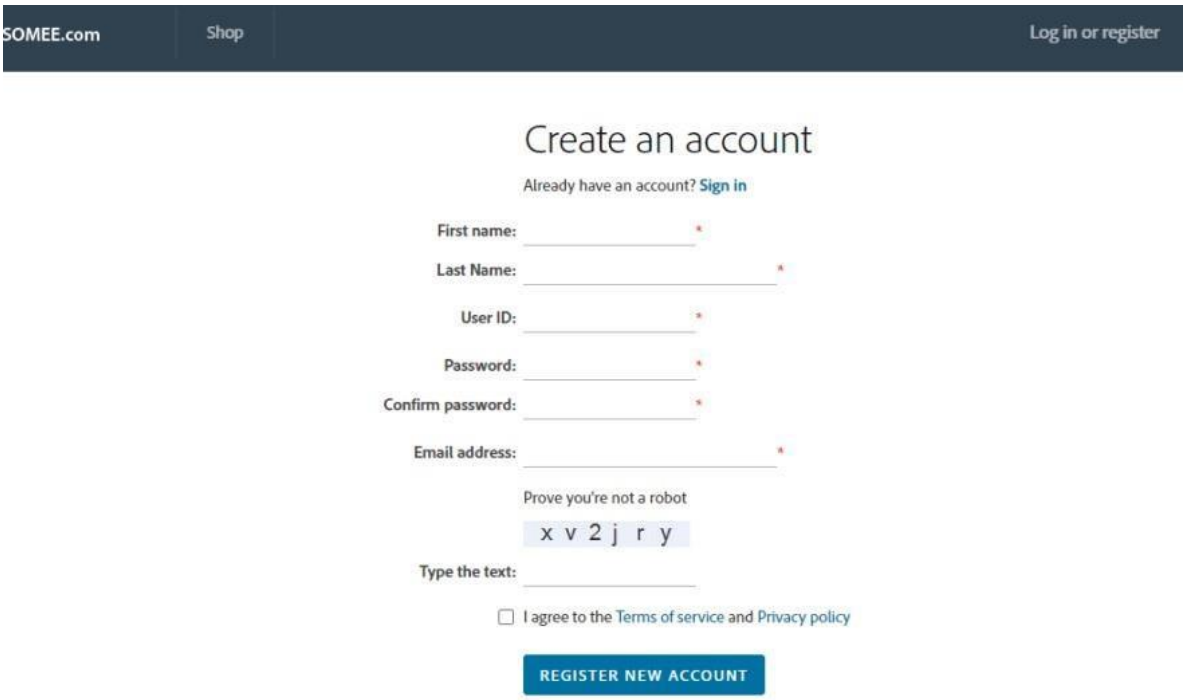


Figura 69 Creación de cuenta

En la siguiente figura 52 tenemos la orden y se dará clic al Checkout

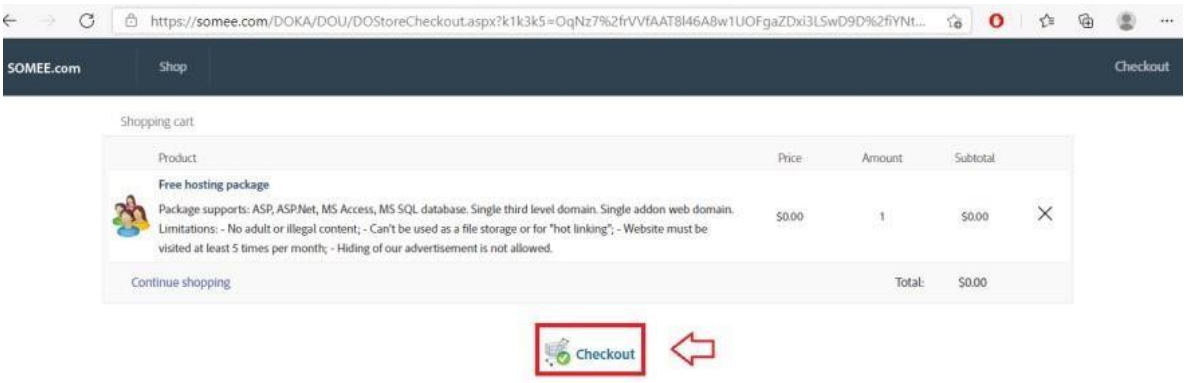


Figura 70 Compra del plan gratuito

Nos envía al panel de control de nuestro nuevo hosting y ya tenemos nuestro hosting listo para configurar.

En la siguiente figura 53 se observa la pantalla de crear el alojamiento del Sitio Web, ingresamos todos los datos necesarios acorde al tipo de aplicación. En nuestro caso se llamará OnSale, esto determinará el subdominio con el que vamos a acceder, elegimos la versión de .Net Core o depende su caso, el título del sitio y opcional una descripción, y creamos el Sitio Web.

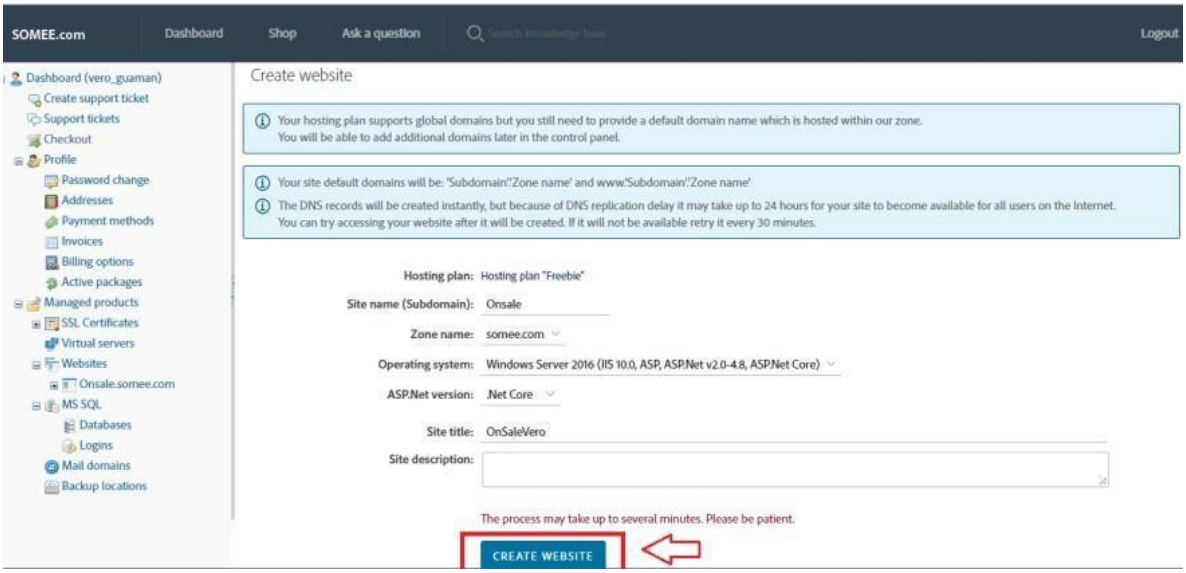


Figura 71 Alojamiento de nuestro Sitio Web

y se ha creado nuestro sitio correctamente.

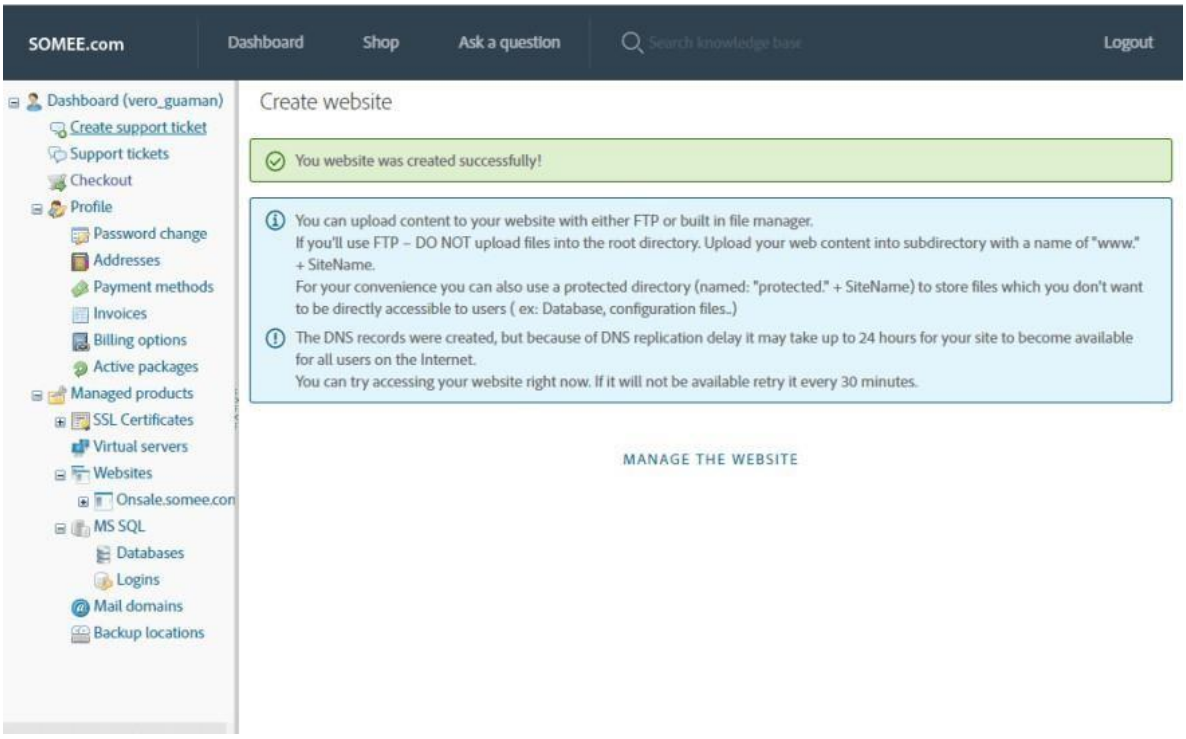


Figura 72 Sitio creado

podemos observar en la parte inferior izquierda ya las opciones para administrarlo

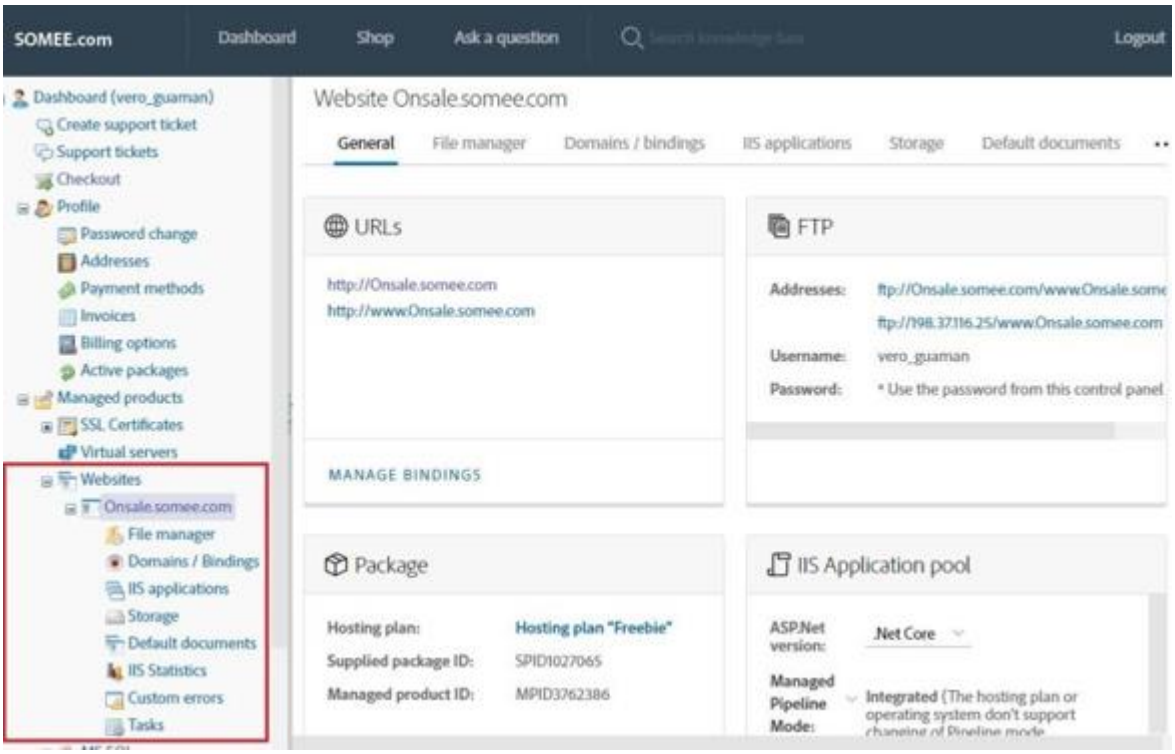


Figura 73 Apartado de administración

9.1.2 Publicar nuestra aplicación

En nuestra aplicación en Visual Studio, damos clic derecho a la solución o al proyecto que vamos a publicar. Entre todas las opciones elegir Publicar.

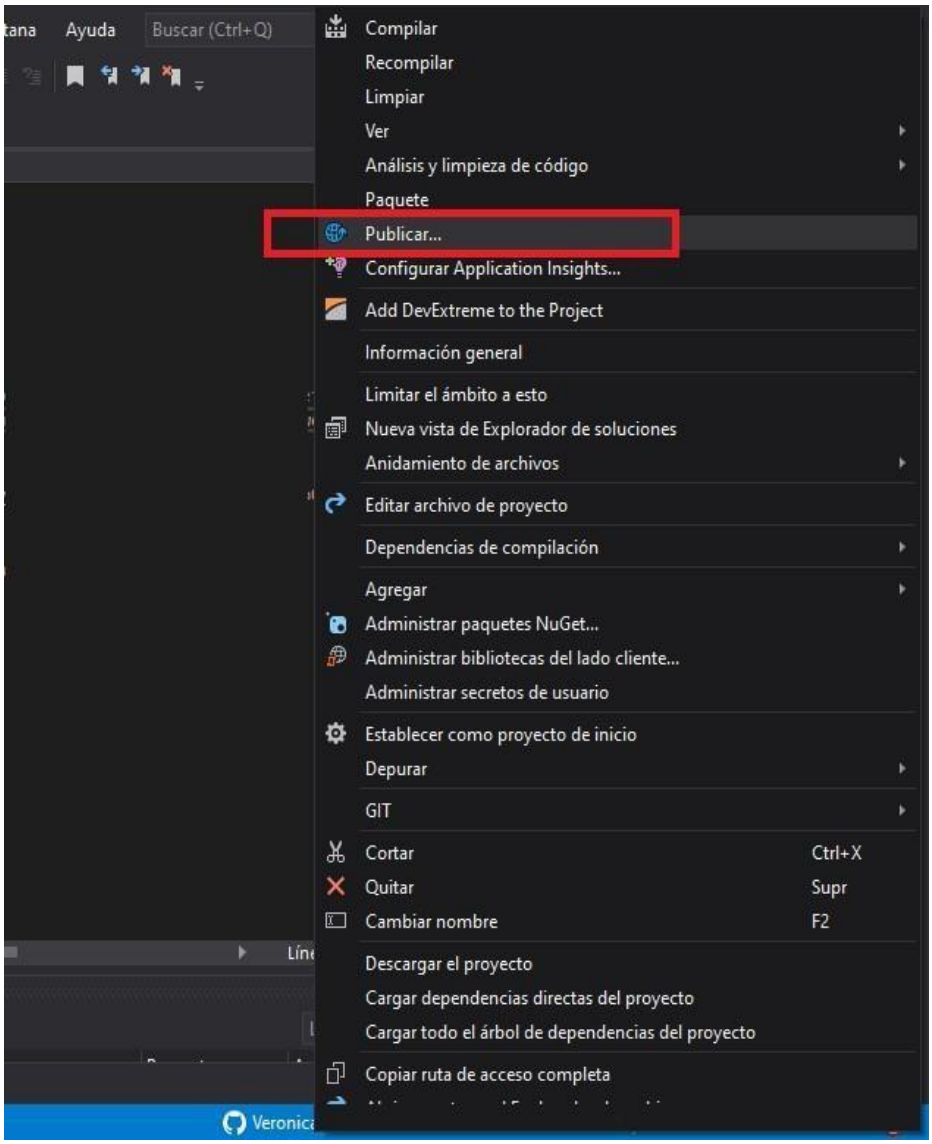


Figura 74 Proyecto que vamos a publicar

En la figura 57 nos muestra una pantalla donde nos dará seis opciones y seleccionaremos la opción "Carpeta"

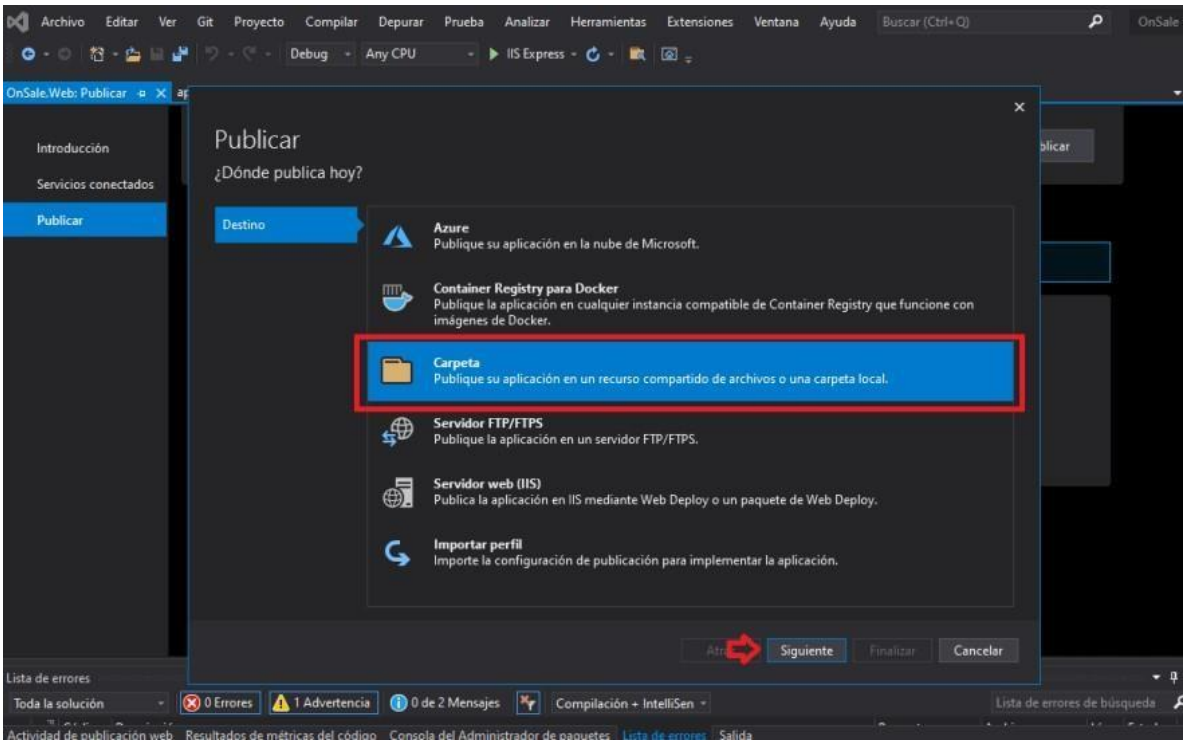


Figura 75 Carpeta

En la figura 58 se muestra una pantalla donde se elegirá la ubicación en la que se van a publicar los archivos, la podemos dejar por defecto o en su lugar, creamos una carpeta vacía en la ubicación de nuestra preferencia y le damos clic en Finalizar.

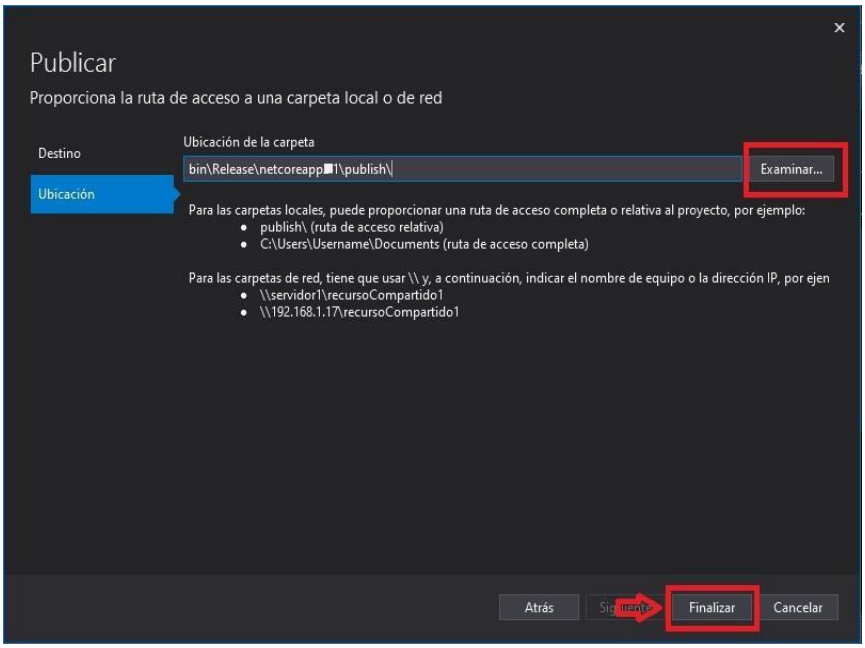


Figura 76 Ubicación de carpeta

En la siguiente figura 59 se muestra la configuración lista, ahora solo nos queda dar clic en Publicar

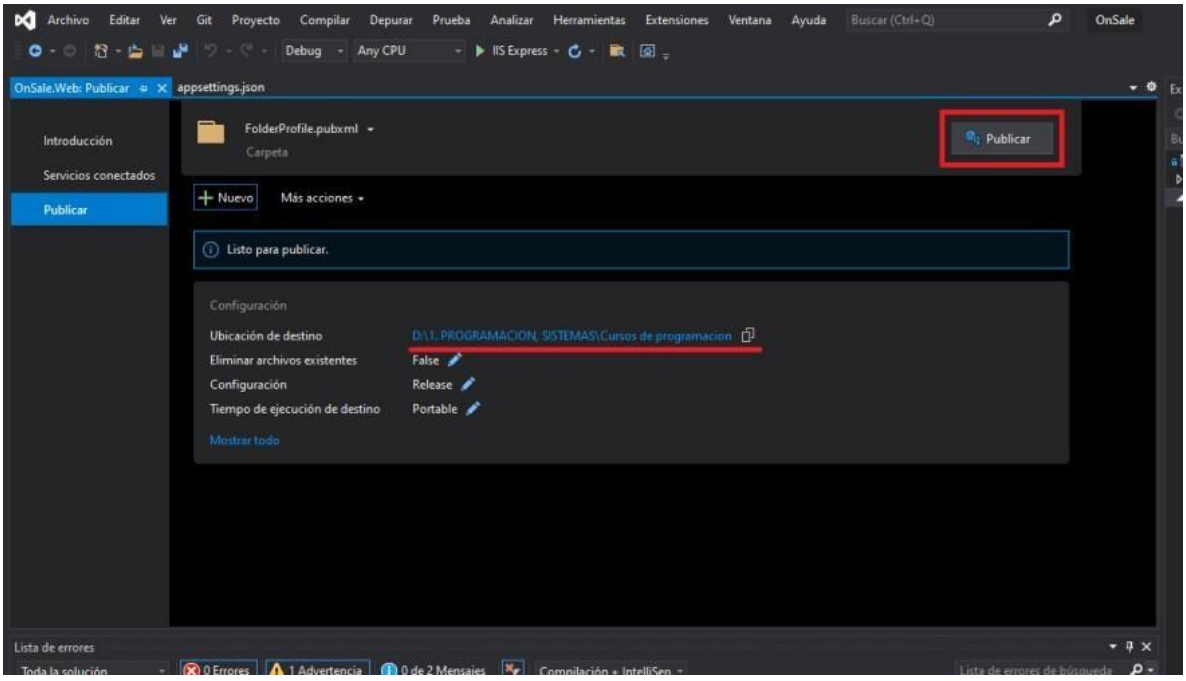


Figura 77 Publicar

En la siguiente figura 60 se muestra que ha sido publicada la aplicación y podemos dirigirnos a comprobar en la ubicación.

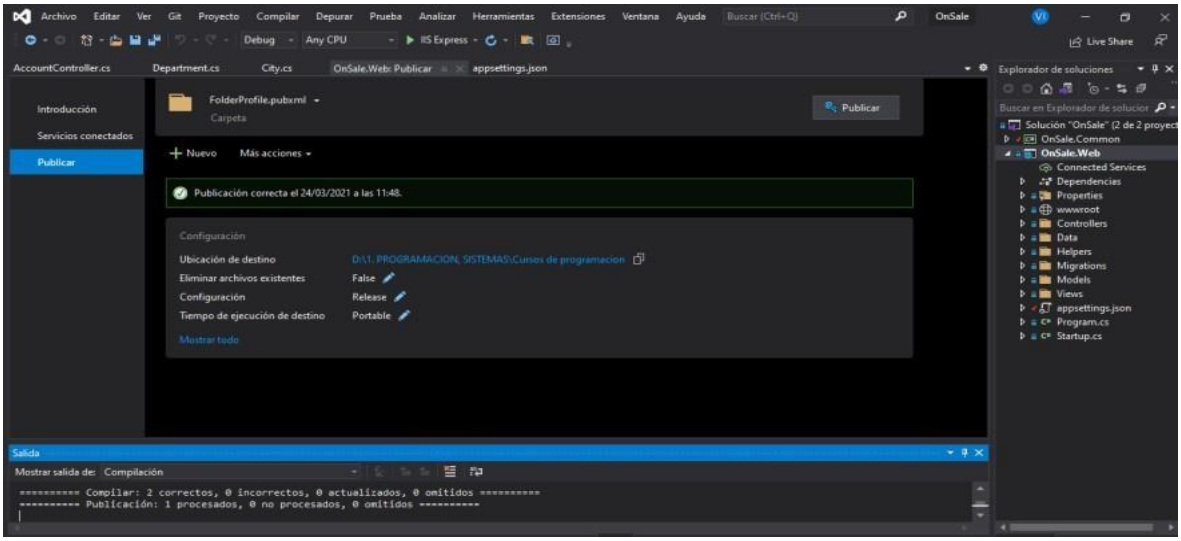


Figura 78 Comprobar en la ubicación.

En la siguiente figura 61 podemos observar que se han creado muchos archivos, mismos que nos servirán para subirlos al hosting, para esto vamos a seleccionar todos los archivos dentro de la carpeta, y lo convertimos en ZIP.

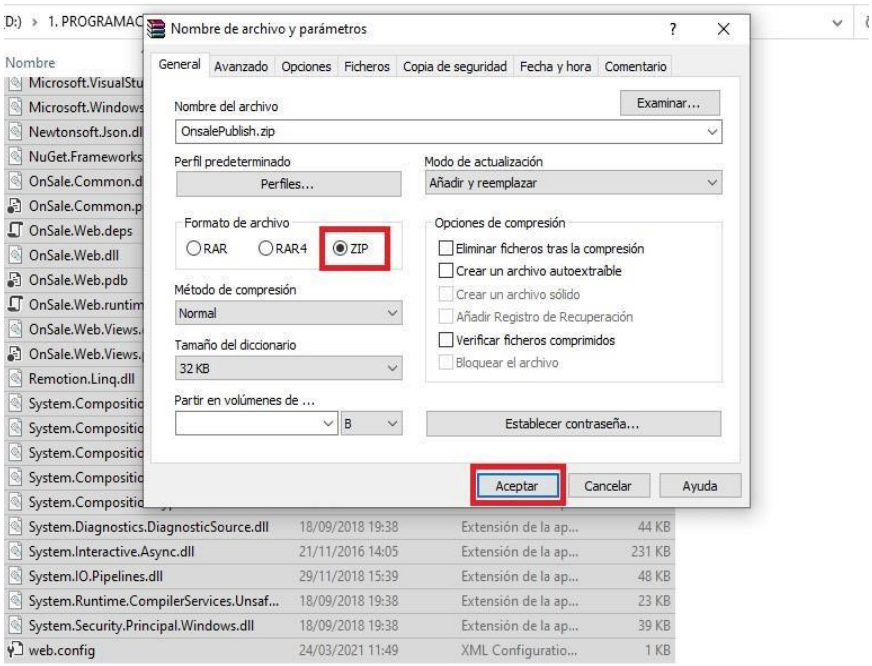


Figura 79 Convertir en ZIP

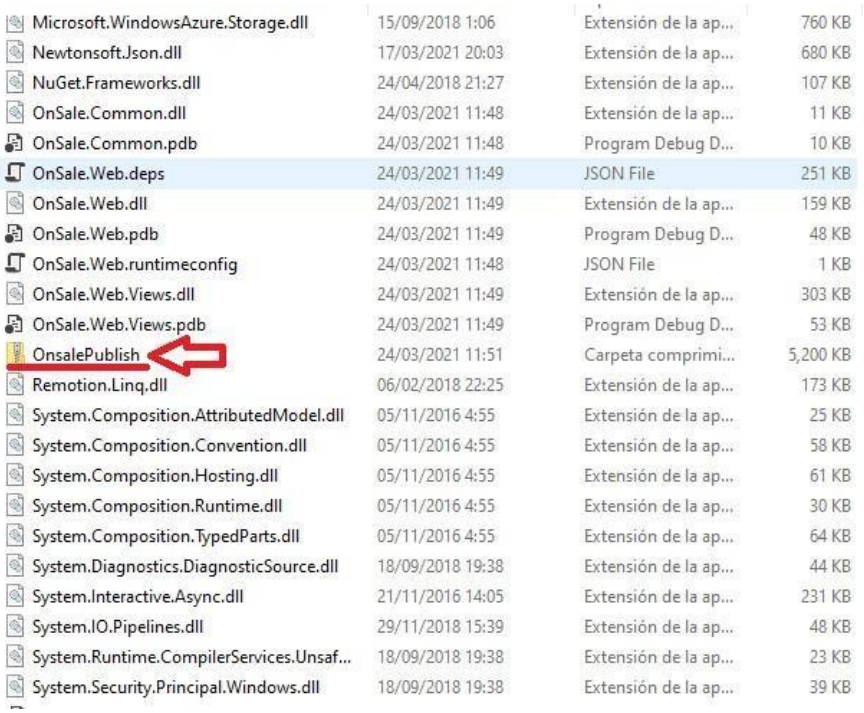


Figura 80 Zip

9.1.3 Subir la Aplicación publicada al hosting.

En la siguiente figura 63 se muestra el panel de administración de Soome. En la parte izquierda inferior del Menú nos dirigimos y damos clic en File Manager o administración de archivos, y damos clic en Upload.

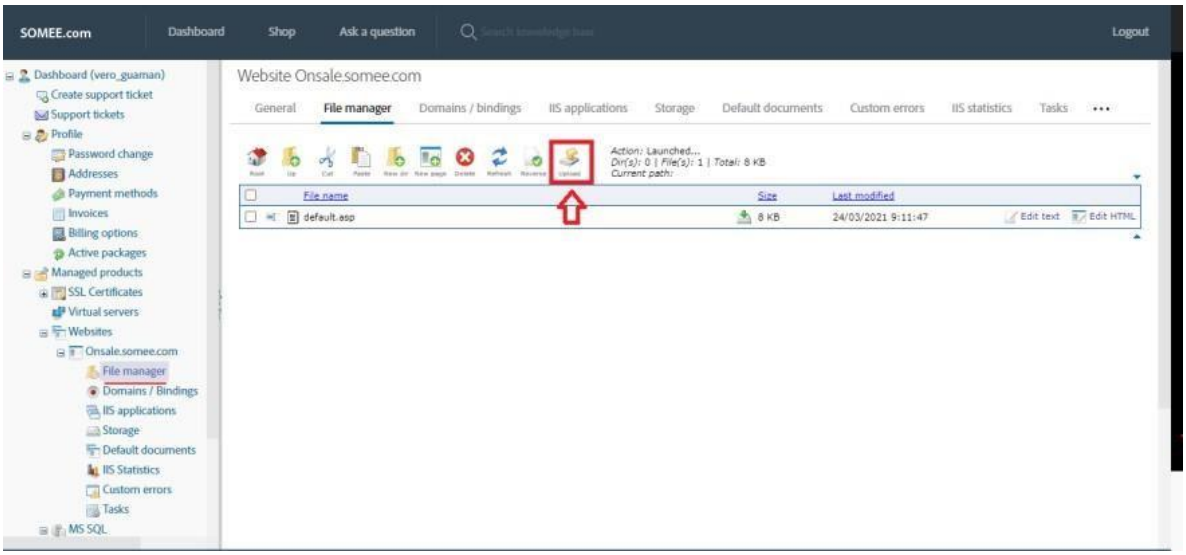


Figura 81 Subir la Aplicación publicada al hosting

Clic en Elegir archivo y se sube el archivo ZIP.

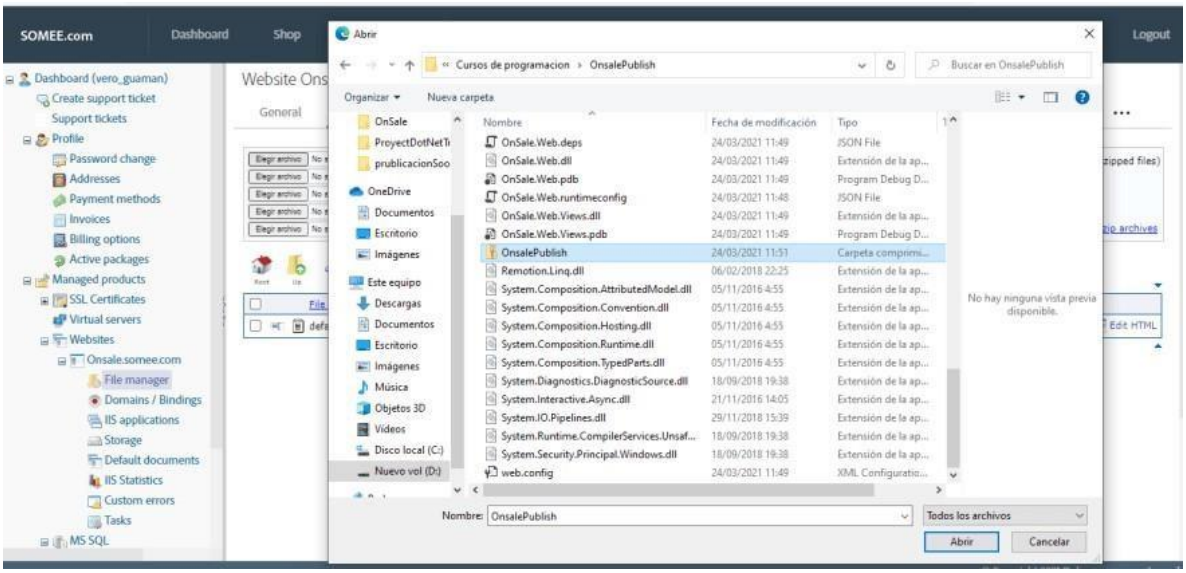


Figura 82 Subir archivo zip

Clic en Upload and Unzip archivos y descomprimir archivos.

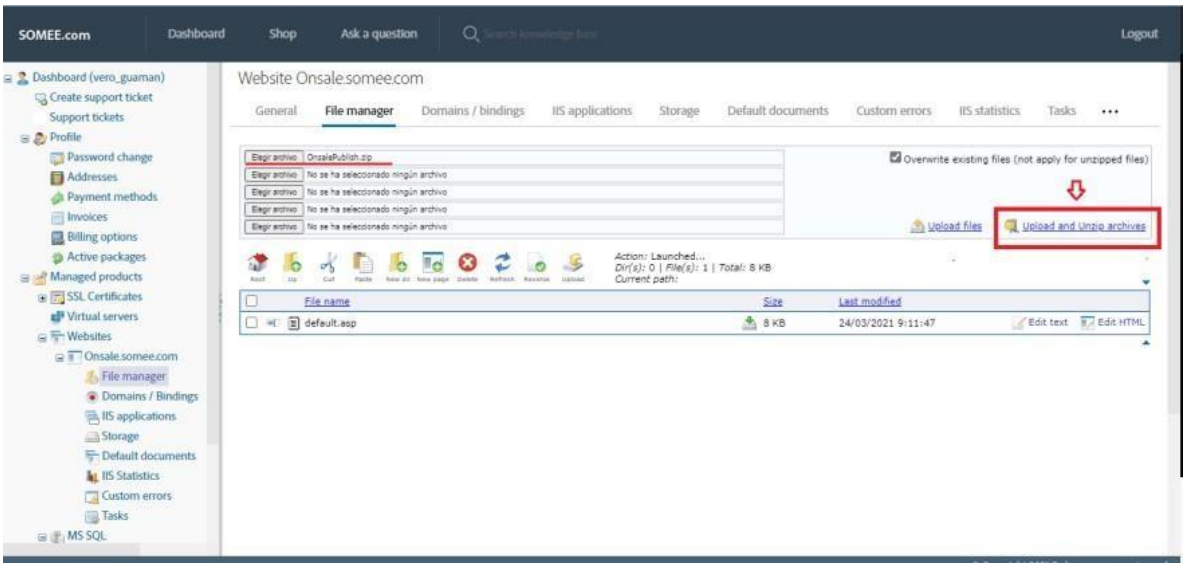


Figura 83 Descomprimir Zip

En la siguiente figura 66 Se mostrará cómo se subirán todos los archivos y se verá de la siguiente manera.

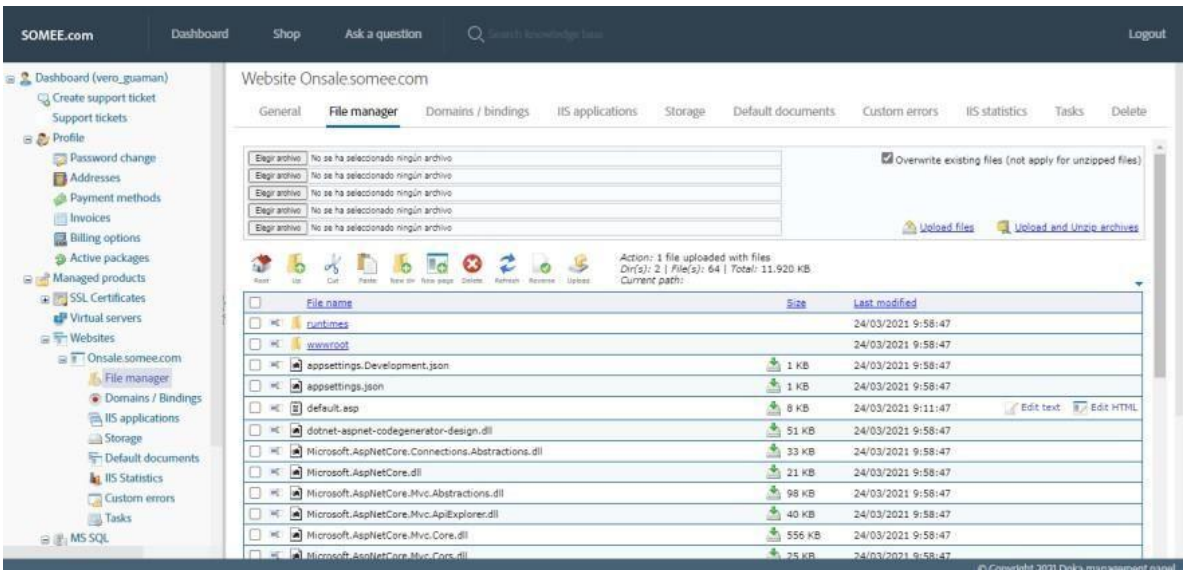


Figura 84 Vista de archivos descomprimidos

En la siguiente figura 67 se ingresa a la URL de la API REST y comprobamos que la aplicación ya está publicada.

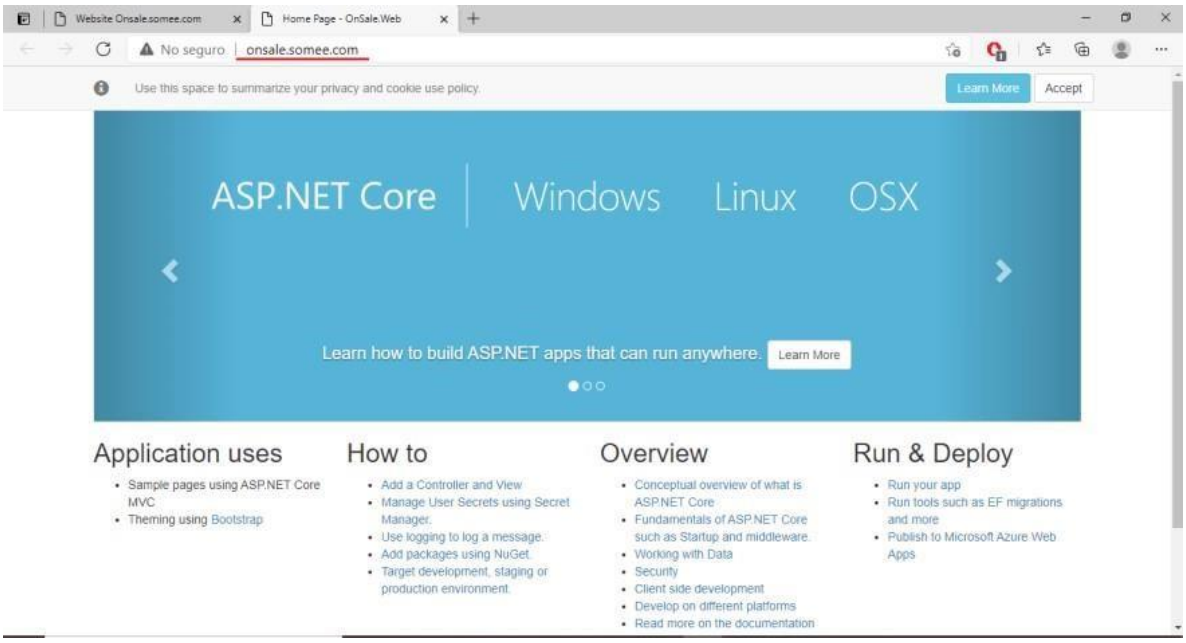


Figura 85 URL de la API REST