UNIVERSIDAD POLITÉCNICA SALESIANA

SEDE QUITO

CARRERA DE INGENIERÍA ELECTRÓNICA

AUTOMATIZACIÓN DE LA ASIGNACIÓN DE ESPACIOS DENTRO DE UNA BODEGA SEMIAUTOMÁTICA PARA LA EMPRESA ROBOTRÓN A TRAVÉS DE INTELIGENCIA ARTIFICIAL

Trabajo de titulación previo a la obtención del

Título de Ingeniero Electrónico

AUTOR: Alexander Andrés Guerrero Saavedra

TUTOR: Carmen Johanna Celi Sánchez

Quito - Ecuador

2022

CERTIFICADO DE RESPONSABILIDAD Y AUTORÍA DEL TRABAJO DE TITULACIÓN

Yo, Alexander Andrés Guerrero Saavedra con documento de identificación N° 1723902001; manifiesto que:

Soy el autor y responsable del presente trabajo; y, autorizo a que sin fines de lucro la Universidad Politécnica Salesiana pueda usar, difundir, reproducir o publicar de manera total o parcial el presente trabajo de titulación.

Quito, 21 de febrero del año 2022

Atentamente,

Alexander Andrés Guerrero Saavedra
1723902001

CERTIFICADO DE CESIÓN DE DERECHOS DE AUTOR DEL TRABAJO DE TITULACIÓN A LA UNIVERSIDAD POLITÉCNICA SALESIANA

Yo, Alexander Andrés Guerrero Saavedra con documento de identificación No. 1723902001, expreso mi voluntad y por medio del presente documento cedo a la Universidad Politécnica Salesiana la titularidad sobre los derechos patrimoniales en virtud de que soy autor del Proyecto técnico: "Automatización de la asignación de espacios dentro de una bodega semiautomática para la empresa Robotrón a través de inteligencia artificial", el cual ha sido desarrollado para optar por el título de: Ingeniero Electrónico en la Universidad Politécnica Salesiana, quedando la Universidad facultada para ejercer plenamente los derechos cedidos anteriormente.

En concordancia con lo manifestado, suscribo este documento en el momento que hago la entrega del trabajo final en formato digital a la Biblioteca de la Universidad Politécnica Salesiana.

Quito, 21 de febrero del año 2022

Atentamente,

Alexander Andrés Guerrero Saavedra
1723902001

CERTIFICADO DE DIRECCIÓN DEL TRABAJO DE TITULACIÓN

Yo, Carmen Johanna Celi Sánchez con documento de identificación N° 1717437808, docente de la Universidad Politécnica Salesiana, declaro que bajo mi tutoría fue desarrollado el trabajo de titulación: AUTOMATIZACIÓN DE LA ASIGNACIÓN DE ESPACIOS DENTRO DE UNA BODEGA SEMIAUTOMÁTICA PARA LA EMPRESA ROBOTRÓN A TRAVÉS DE INTELIGENCIA ARTIFICIAL, realizado por Alexander Andrés Guerrero Saavedra con documento de identificación N° 1723902001, obteniendo como resultado final el trabajo de titulación bajo la opción Proyecto técnico que cumple con todos los requisitos determinados por la Universidad Politécnica Salesiana.

Quito, 21 de febrero del año 2022

Atentamente,

Ing. Carmen Johanna Celi Sánchez, Mgtr
1717437808

DEDICATORIA

A mi familia, especialmente a madre María Augusta Saavedra y a mi padre Alexander Raúl Guerrero quienes me han brindado su apoyo incondicional, compromiso y amor durante todas las etapas de mi vida. A mis amigos quienes me han alentado a seguir adelante y no rendirme a lo largo de este proceso de crecimiento personal. A todas las personas que han aportado con conocimientos, sabiduría y consejos para que este proyecto de titulación se realice satisfactoriamente.

Alexander Andrés Guerrero Saavedra

AGRADECIMIENTO

Mi más profundo agradecimiento a la Universidad Politécnica Salesiana por brindarme las herramientas necesarias para avanzar durante mi formación profesional, a todos los docentes que han compartido sus conocimientos durante mi proceso de aprendizaje.

A mi tutora: Mgtr. Carmen Johanna Celi Sánchez por su apoyo y orientación durante el desarrollo de este proyecto de titulación; a la empresa Robotrón, especialmente al Ing. Víctor Moreno y a la Ing. Marcia Carrión por abrirme las puertas dentro de la empresa, por haber sido una guía durante las etapas del desarrollo del proyecto y por permitirme formar parte del mismo. Un agradecimiento especial a mis padres y a mi familia por motivarme a culminar esta etapa de mi proceso de educación profesional.

Alexander Andrés Guerrero Saavedra

ÍNDICE DE CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

RESUMEN

Dado el creciente y demandante avance tecnológico, aplicaciones y técnicas de inteligencia artificial en la industria, la automatización de procesos logísticos, sumado ante la escasa implementación de sistemas inteligentes en bodegas de pequeñas y medianas empresas, surge la oportunidad de diseñar, ensayar e implementar un algoritmo basado en inteligencia artificial (Deep learning) haciendo uso de herramientas y tecnología de vanguardia.

El presente proyecto está enfocado al desarrollo de un software haciendo uso en la arquitectura de red neuronal LSTM, basándose en estudios previos y software libre, con el fin de realizar predicciones en base a datos históricos de productos, mismas que permitan realizar segmentación de áreas y reservas de espacio físico, a su vez de un algoritmo de asignación el cual designa un espacio único dentro de las áreas previamente reservadas.

Mediante la implementación del software en un servidor de la empresa Robotrón, haciendo uso de datos de 10 clientes de una empresa externa y 30 productos de venta se logró validar la efectividad de la arquitectura diseñada para los datos de entrada con un error RMSE máximo de 3.49. Dentro de las pruebas de validación experimentales, se obtuvo un error mínimo del 1.13% en el primer mes de predicción y un error máximo de 12.3% en el segundo mes de predicción. A su vez, haciendo uso del software inteligente, se logró reducir hasta en aproximadamente 6 minutos el tiempo de asignación e ingreso a bodega.

Palabras clave: LSTM, Deep learning, algoritmo inteligente, series de tiempo.

ABSTRACT

Given the growing and demanding technological advancement, applications and techniques of artificial intelligence in the industry, the automation of logistics processes, added to the scarce implementation of intelligent systems in warehouses of small and medium-sized companies, the opportunity arises to design, test and implement an algorithm based on artificial intelligence (Deep learning) using cutting-edge technology and tools.

This project is focused on the development of software using the LSTM neural network architecture, based on previous studies and free software, in order to make predictions based on historical product data, which allow for area segmentation. and physical space reserves, in turn of an allocation algorithm which designates a single space within the previously reserved areas.

By implementing the software on a server of the Robotrón company, making use of data from 10 clients of an external company and 30 sales products, it was possible to validate the effectiveness of the architecture designed for the input data with a maximum RMSE error of 3.49. Within the experimental validation tests, a minimum error of 1.13% was obtained in the first month of prediction and a maximum error of 12.3% in the second month of prediction. In turn, by making use of the intelligent software, it was possible to reduce the time of allocation and entry to the warehouse by up to approximately 6 minutes.

Keywords: LSTM, Deep learning, intelligent algorithm, time series.

INTRODUCCIÓN

El presente trabajo de titulación se basó en la aplicación de inteligencia artificial (Deep learning) como herramienta para el desarrollo de un software para la automatización de una bodega semiautomática, se describe la importancia y el potencial del uso de estas tecnologías dentro de los procesos de almacenamiento de la industria.

El Capítulo 1 describe el planteamiento del problema, la justificación y los objetivos tanto general como específicos, que se plantearon para este proyecto.

El Capítulo 2 consta de una visión general acerca de inteligencia artificial, redes neuronales y Deep learning, conceptos esenciales para la comprensión de redes neuronales recurrentes y su relación series temporales, también se describen conceptos acerca del funcionamiento de una red LSTM.

En el Capítulo 3 se exponen las características de las librerías y elementos de software que se emplearon para la integración del programa con el servidor de la empresa Robotrón, se presenta la arquitectura de la red neuronal y algoritmos utilizados, para la automatización de una bodega semiautomática.

Finalmente, en el Capítulo 4 se presentan las conclusiones y recomendaciones obtenidas en el presente proyecto de titulación, a través del uso de metodología experimental, mediante la cual se realizó un análisis de pruebas con una base de datos alojada en un servidor de la empresa Robotrón, donde se determinó que el trabajo de titulación cumplió con los objetivos propuestos.

Robotrón es una empresa con sede en el Valle de los Chillos – Ecuador, dedicada a la electrónica, robótica y automatización de procesos industriales. Robotrón desempeña tareas desde la fase de diseño, maniobra, hasta la puesta en marcha y optimización de todo tipo de sistemas de control industriales, así como también, realiza cursos y capacitaciones personalizadas a sus clientes.

CAPÍTULO 1
ANTECEDENTES

1.1 Planteamiento del problema

En las últimas décadas las bodegas han evolucionado exponencialmente a nivel mundial. Todo ello en consecuencia a la alta competitividad laboral, al ritmo acelerado de producción y la alta demanda que existe en la actualidad dentro de la industria.
Con el pasar de los años las técnicas de automatización no tardaron en llegar al área de logística de almacenamiento, dando lugar herramientas de software como WMS (Warehouse Management System) Sistemas de gestión de bodega o herramientas de hardware como Racks (estanterías) para facilitar el proceso de almacenamiento dentro de una bodega.

En la actualidad en el Ecuador en pequeñas y medianas empresas, las técnicas utilizadas dentro de la industria para la organización del espacio físico dentro de bodegas siguen siendo tradicionales, es decir: el operario recibe una orden de ingreso a bodega con los productos correspondientes, a continuación, el operario busca y coloca el producto en un espacio vacío en zonas estáticas previamente definidas, a su vez el operario puede almacenar el producto de manera aleatoria. Lo cual no garantiza que dicha posición sea la óptima para el producto en cuestión, ya que es posible que el producto tenga una rotación lenta dentro de la bodega y obstaculice que otros productos con ciclos de rotación rápida sean despachados de manera eficaz. El uso de software WMS en su mayoría son aplicados en empresas multinacionales como grandes distribuidoras y empacadoras, dejando atrás a otras empresas en el uso de este tipo de tecnologías.

De esta forma se evidencia la escasez de sistemas automatizados e inteligentes dentro del control de bodegas a nivel nacional, lo cual tiene efectos negativos como: pérdidas de existencias dentro de la propia bodega, en ocasiones es necesario mover una gran cantidad de mercadería para encontrar un producto en específico lo que produce ineficiencia al momento de realizar despachos, entre otros.

De esta forma el proyecto busca dar solución al problema de falta de productividad dentro de los almacenes, perdida, desorganización de productos y retrasos a la hora de hacer ingresos o despachos en el almacén.

1.2 Justificación del proyecto

El presente proyecto se enfocará en el desarrollo de un software basado en el algoritmo LSTM (Long Short - Term Memory). Ya que, con el uso de dicho algoritmo se puede analizar una gran cantidad de datos en forma de series temporales, es decir datos almacenados periódicamente, así se puede realizar predicciones del espacio que será necesario para almacenar los productos dentro de la bodega semiautomática en base a datos históricos.

Además, se desarrollará un software para la asignación individual de productos en una posición única dentro de las áreas previamente reservadas, dado que los productos se pueden mezclar en la cola de ingreso a bodega. De esta forma se garantiza un área y un espacio único para cada producto dentro de la bodega.

La importancia de implementar un sistema moderno basado en inteligencia artificial recae en mejorar los procesos logísticos dentro una bodega, mejorar su productividad, dando la posibilidad de reducir tiempos a la hora de ingresar o despachar productos. A su vez, tener productos organizados en áreas de tal forma que se mejoren los tiempos de búsqueda de un producto en concreto dentro del almacén. Además, al hacer uso del algoritmo LSTM (Long Short - Term Memory) se puede realizar una predicción a largo plazo y preparar el espacio interno para los meses con más demanda de producción y de esta forma evitar las pérdidas económicas que conlleva el tener un espacio de bodega mal organizado.

1.3 Objetivos

1.3.1 Objetivo general

Diseñar un sistema automatizado para la asignación de espacios físicos dentro de una bodega semiautomática a través de inteligencia artificial.

1.3.2 Objetivos específicos

- Investigar librerías y herramientas de Python mediante artículos científicos relacionados para la profundización del estudio del algoritmo de la red neuronal LSTM.
- Diseñar un software modular mediante inteligencia artificial para la automatización del ingreso de productos dentro de una bodega semiautomática.
- Simular el comportamiento de la bodega mediante el software FlexSim para la planificación y demostración de los algoritmos utilizados.
- Implementar el algoritmo de inteligencia artificial para la automatización de una bodega con racks móviles mediante un servidor y una base de datos de la empresa Robotrón.
- Realizar pruebas para la validación del funcionamiento mediante tablas de productividad de bodega adquiridas experimentalmente.

CAPÍTULO 2
MARCO TEÓRICO

2.1 Introducción

En el presente capítulo se detallan generalidades y conceptos básicos que engloban el estudio de la inteligencia artificial para la predicción de datos a futuro mediante la red neuronal LSTM, partiendo de sus bases teóricas hasta el análisis previo a su implementación dentro de una bodega semiautomática; así como también los conocimientos necesarios para el diseño y desarrollo del proyecto, descripciones del software utilizado, librerías de desarrollo, así como también de comunicación software con base de datos, se describe el estado del arte de la red neuronal LSTM, a su vez de aspectos generales acerca del entorno de simulación FLEXSIM.

2.2 Inteligencia artificial

La inteligencia artificial se puede definir como la capacidad que adquiere una máquina de aprender y tomar decisiones por sí misma, a través de distintas metodologías y algoritmos. La inteligencia artificial pretende simular el comportamiento del cerebro humano, así como también el comportamiento animal comunitario.

A diferencia de los seres humanos una máquina, más específicamente un programa desarrollado en base a inteligencia artificial, no necesita descansar, puede analizar una gran cantidad de datos simultáneamente y a su vez, el margen de error se disminuye considerablemente en comparación si la misma tarea fuese realizada por un ser humano. (Rouhiainen, 2018)

2.2.1 Machine learning

Es el enfoque principal de la inteligencia artificial (aprendizaje automático). Es el campo de estudio de la informática que brinda a las computadoras la capacidad de aprender y tomar decisiones sin necesidad que el algoritmo haya sido programado explícitamente para ello. (Gerón, 2017), (Rouhiainen, 2018)

El machine learning (aprendizaje automático) se divide principalmente en tres grupos:

a) Aprendizaje supervisado

El aprendizaje supervisado (basado en tareas), se caracteriza por ser un método que requiere la intervención humana para su funcionamiento, es decir, los algoritmos deben ser entrenados con un set de datos que previamente ha sido etiquetado u organizado para categorizar los elementos no conocidos (entrada de la red neuronal) y obtener una predicción (salida de la red neuronal), el aprendizaje supervisado requiere intervención humana para su retroalimentación. (Rouhiainen, 2018)

b) Aprendizaje no supervisado

El aprendizaje no supervisado (basado en datos), se caracteriza por ser un método que no requiere la intervención humana para categorizar elementos no conocidos, este tipo de algoritmo no necesita que los datos tengan algún tipo de etiquetado u organización previa para proporcionar una predicción, es decir, el algoritmo busca por sí mismo la forma más óptima, para lograr clasificar los elementos no conocidos en base a predicciones. (Rouhiainen, 2018)

c) Aprendizaje de refuerzo

El aprendizaje de refuerzo (aprende a reaccionar a su entorno), se caracteriza por aprender en base a "experiencia", es decir, el algoritmo requiere una recompensa positiva cada vez que realizan una predicción correcta. (Rouhiainen, 2018). El aprendizaje de refuerzo permite que el propio algoritmo aprenda automáticamente que acciones tomar para maximizar la recompensa obtenida. (Pina, 2021)

2.2.2 Deep Learning

El Deep Learning (Aprendizaje profundo) es una rama del Machine learning (aprendizaje automático) que permite a las computadoras aprender a partir de experiencias, deduciendo conceptos de manera jerárquica y construyendo dichos conceptos a partir de su relación con conceptos más simples. (Goodfellow, Bengio, & Courville, 2016).

El Deep Learning utiliza modelos compuestos por múltiples capas de procesamiento, basadas en el uso de redes neuronales artificiales (ANN's).

Las redes neuronales artificiales son sistemas parametrizados y nacen de modelos computacionales no lineales, los mismos que basan su arquitectura y funcionamiento con la estructura neuronal del cerebro humano, pudiendo realizar acciones tales como: clasificación, predicción, toma de decisiones y visualización. Las neuronas están organizadas en tres capas interconectadas: capa de entrada, capas ocultas y capa de salida. Cada neurona consta de una entrada ponderada, la misma que cuenta con parámetros ajustables, una función de activación y una salida. (Pina, 2021)

Existen dos modelos principales de arquitectura de redes neuronales basadas en Deep Learning (redes neuronales profundas). Las redes neuronales FeedForward (FNN), mismas donde la información viaja de manera unidireccional desde la capa de entrada hacia la capa de salida, pasando por las capas ocultas. Y las redes neuronales Recurrentes (RNN) siendo esta una variante recursiva, donde las neuronas se interconectan a manera de ciclos dirigidos. (Pina, 2021)

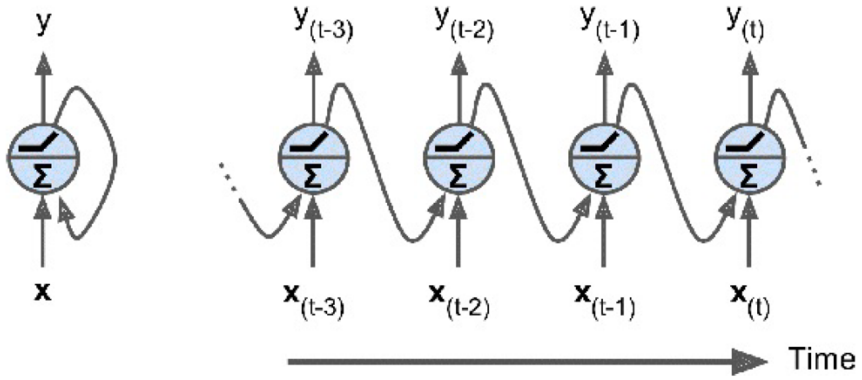
2.3 Redes neuronales recurrentes (RNN)

Las redes neuronales recurrentes se diferencian de las redes neuronales convencionales por su capacidad de trabajar de manera dinámica, es decir, tienen la capacidad de enviar la información de manera bidireccional. De esta forma, se dice que la salida de la red neuronal depende tanto de los valores que tomen las variables de entrada de la red neuronal como de los valores anteriores de la entrada de la red neuronal y de los valores anteriores de la salida de la red neuronal, etc. (Machado, 2018)

La arquitectura de redes neuronales recurrentes tiene la habilidad de analizar y trabajar con series temporales, las mismas que son secuencias de datos de longitudes arbitrarias, con el objetivo de predecir un dato a futuro en una secuencia temporal. (Gerón, 2017)

A continuación, en la imagen izquierda de la Figura 2.1, se muestra la arquitectura más simple de RNN, compuesta por una neurona de entrada, la misma que produce una salida y retroalimenta su resultado a sí misma.

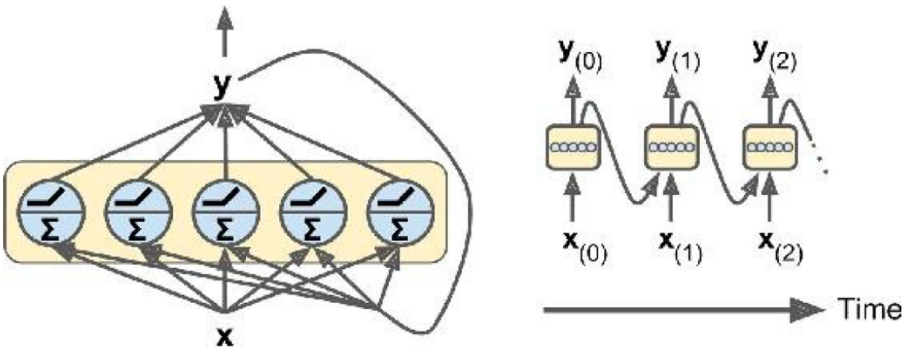
Figura 2.1: Neurona recurrente simple (izquierda), desarrollo en el tiempo de neurona recurrente (derecha)



Neurona recurrente retroalimentándose (Izquierda), Neurona recurrente retroalimentándose a lo largo de una serie temporal (Derecha). Fuente: (Gerón, 2017)

En la imagen derecha de la Figura 2.1, se puede observar el comportamiento de la neurona en una serie temporal. La neurona recibe dos entradas, una entrada $x(t)$ y la salida del paso anterior de la serie temporal $y(t-1)$. De esta forma están formadas las capas de las redes neuronales recurrentes. Para cada paso t de la serie temporal todas las neuronas reciben una entrada $x(t)$ y la salida del paso anterior $y(t-1)$, como se muestra a continuación en la Figura 2.2. (Gerón, 2017)

Figura 2.2: Capa de una red neuronal recurrente (izquierda), desarrollo en el tiempo de una capa de una red neuronal recurrente (derecha)



Capa de una red neuronal recurrente retroalimentándose (Izquierda), capa de una red neuronal recurrente retroalimentándose a lo largo de una serie temporal (Derecha). Fuente (Gerón, 2017)

Como se puede observar en la Figura 2.2, cuando se trabaja con capas de redes neuronales recurrentes, tanto la entrada como la salida se transforman a vectores, a diferencia de cuando era una sola neurona, esta poseía una entrada y una salida con valores escalares. (Gerón, 2017)

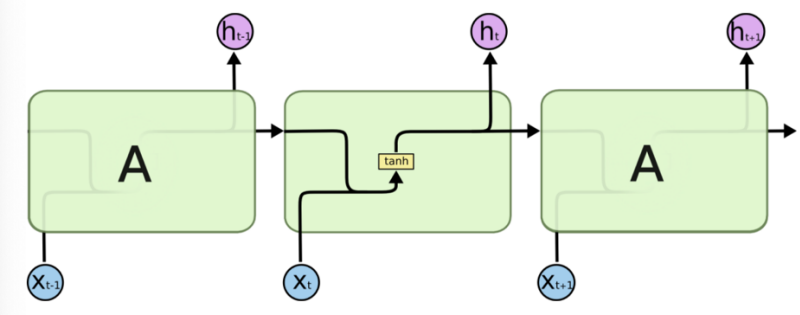
2.4 Red neuronal LSTM

Por sus siglas en inglés “Long Short – Term Memory” o Memoria larga a corto plazo, es un tipo de arquitectura de Deep learning, basada en redes neuronales recurrentes (RNN). La red neuronal La ventaja del uso de este tipo de arquitectura recae en su capacidad para procesar grandes cantidades de datos con gran precisión al momento de realizar predicciones. (Wu, Yao , & Yu, 2018)

2.4.1 Arquitectura de la red neuronal LSTM

Las redes LSTM tienen una arquitectura muy similar a las RNNs a manera de cadena (Figura 2.3), la diferencia radica en su interior, las RNNs poseen una sola capa de red neuronal, al contario de las redes LSTM que poseen cuatro capas de redes neuronales que interactúan entre sí como se muestra en la Figura 2.4.

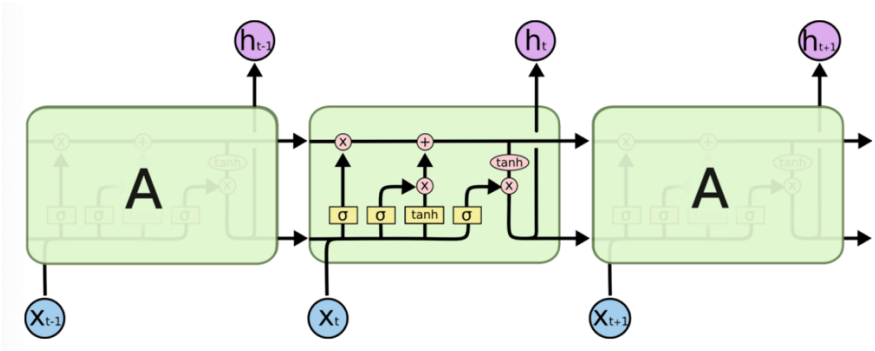
Figura 2.3: Arquitectura básica de una RNN



Arquitectura básica de una RNN. Fuente: (Olah, 2015)

La característica principal de la red LSTM recae en que posee la capacidad de eliminar o almacenar información del estado de la celda. Esta función la realiza mediante estructuras llamadas puertas, las cuales actúan a manera de válvulas, dejando o no pasar la información a través de la celda LSTM.

Figura 2.4: Arquitectura básica de una red LSTM

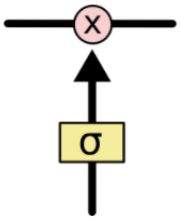


Arquitectura básica de una red LSTM, los círculos de color rosa representan operaciones matemáticas, los cuadros amarillos representan capas de redes neuronales, en este caso podemos observar tres capas sigmoides y una capa tangente hiperbólica. Fuente: (Olah, 2015)

Las puertas están compuestas por una capa con función de activación sigmoidea y una operación de multiplicación. La función sigmoidea genera números entre 0 y 1, estos valores están relacionados de la siguiente manera: 0 no pasa información y 1 pasa toda la información.

La arquitectura básica de una red LSTM cuenta con tres puertas que controlan el estado de la celda LSTM, puerta de entrada, una puerta de salida y una puerta de olvido. (Wu, Yao , & Yu, 2018)

Figura 2.5: Estructura de una puerta LSTM



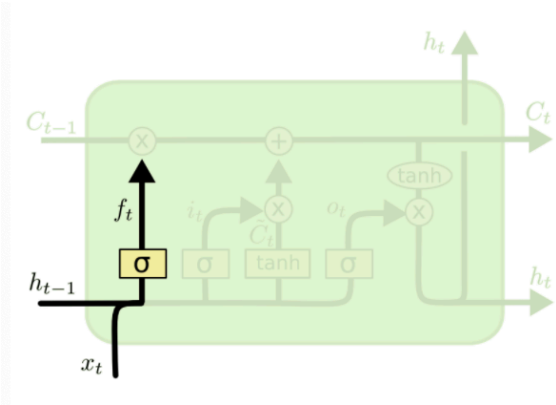
Puerta LSTM formada por una capa con función de activación sigmoidea y una operación de multiplicación. Fuente: (Olah, 2015)

A continuación, se detalla paso a paso el funcionamiento de una red neuronal LSTM y en el orden que las puertas entran en funcionamiento.

2.4.2 Puerta de olvido

El primer paso que realiza la red neuronal LSTM es decidir los datos que serán eliminados o almacenados dentro del estado de la celda LSTM, esta decisión la toma la puerta denominada puerta de olvido mediante el dato de entrada X_t y el estado oculto anterior h_{t-1}, estos datos entran a una capa sigmoidea la cual decide si el dato es relevante, si este es el caso la función sigmoidea arroja el valor de 1 y mantiene el dato generando un f_t con el valor del dato relevante, caso contrario la función sigmoidea arroja el valor de 0 desechando el dato. (Olah, 2015)

Figura 2.6: Puerta de olvido LSTM



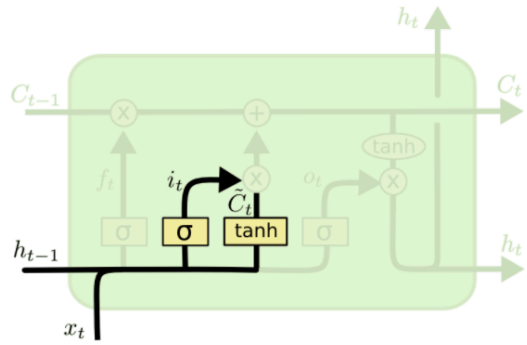
$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

Puerta de olvido LSTM. Fuente: (Olah, 2015)

2.4.3 Puerta de entrada

El siguiente paso que realiza la red neuronal LSTM es decidir que nueva información será almacenada en el estado de la celda, esto se realiza mediante la puerta de entrada, la misma que consta de dos capas, la capa sigmoidea, decide que valores serán actualizados i_t y la capa tanh, crea un vector con los nuevos valores candidatos $\bar{C}_t$. (Olah, 2015)

Figura 2.7: Puerta de entrada LSTM

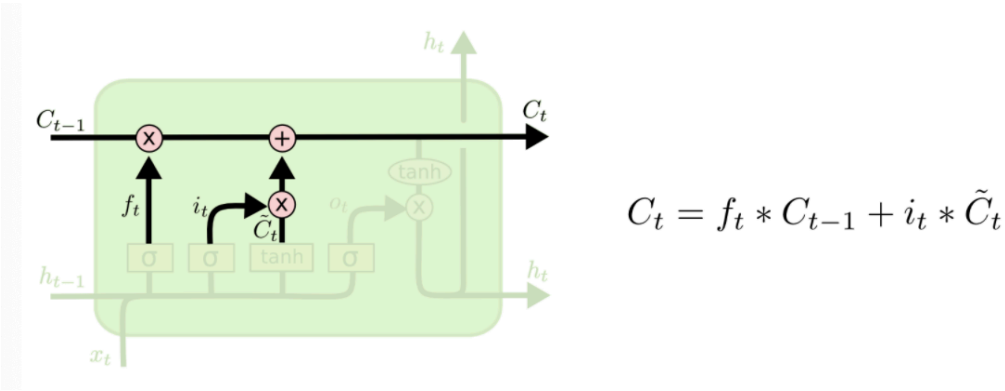


$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

Puerta de olvido LSTM. Fuente: (Olah, 2015)

A continuación, la red neuronal actualiza el estado de la celda anterior C_{t-1} con el nuevo estado de la celda C_t. Se multiplica el estado antiguo de f_t con el estado anterior de la celda C_{t-1} y se agrega los nuevos valores candidatos obtenidos en la puerta de entrada. (Olah, 2015)

Figura 2.8: Actualización de estado en la celda LSTM

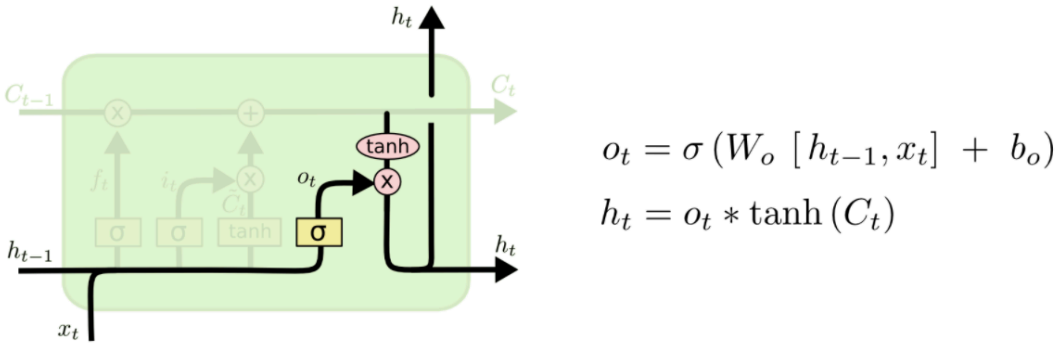


Actualización de estado en la celda LSTM. Fuente: (Olah, 2015)

2.4.4 Puerta de salida

Finalmente se decide que se va a generar como salida, la puerta de salida se basa en el estado actual de la celda a la cual se le aplicará un filtro y genera el nuevo estado oculto. El estado oculto almacena información sobre las entradas anteriores. Como primer paso se aplica la capa sigmoidea al estado oculto anterior y la entrada actual O_t. A continuación, se aplica una celda tanh al nuevo estado de la celda C_t y se lo multiplica por el resultado O_t, generando el nuevo estado oculto h_t.

Figura 2.9: Puerta de salida LSTM



Puerta de salida LSTM y generación de nuevo estado oculto h_t . Fuente: (Olah, 2015)

2.5 Bodegas Semiautomáticas

Se llama bodega semiautomática a la automatización parcial de una bodega convencional, incorporando hardware al estilo racks y software especializado para la administración de la bodega.

Un rack es un soporte principalmente de construcción metálica el cual tiene la función de almacenar objetos dentro de una bodega a manera de matriz, dichos objetos almacenados pueden ser cajas, bins, palets, etc.

Con la implementación de racks móviles o Moviracks dentro de un espacio limitado como es el de un almacén o una bodega, se consigue compactar el espacio entre racks y aumentar la capacidad máxima de la bodega. Los racks se colocan sobre bases móviles que mediante motores y rieles se desplazan lateralmente. De esta forma se suprimen los pasillos estáticos que existen en las bodegas tradicionales.

Al momento que sea necesario realizar un movimiento dentro de la bodega se abre únicamente el pasillo donde se desea realizar la acción dando paso al operario con el montacargas.

El concepto de bodega semiautomática recae en la combinación de un hardware de optimización de espacio (Moviracks) y un software inteligente para comandar los movimientos dentro de la bodega. (Esmena, 2021).

2.6 Estado del arte de sistemas basados en el algoritmo LSTM

En las últimas décadas el interés por predecir comportamientos o valores en varias áreas de la tecnología, ha dado paso a que los investigadores dediquen gran cantidad de tiempo a desarrollar, implementar y mejorar algoritmos predictivos en varias áreas como: el análisis de datos, ciencia e ingeniería. Dado el panorama actual, donde la industria tiende a la automatización y control de procesos, los modelos de inteligencia artificial basados en la arquitectura LSTM han sido considerados eficaces a la hora de trabajar con datos secuenciales o con series temporales.

Este tipo de redes neuronales han servido para tratar problemas como: reconocimiento de textos, traducción de texto, predicciones de clima, predicciones de estructuras de proteínas, análisis de datos de audio y video, etc. (Greff, Rupesh K., Jan , Bas R. , & Jurgen , 2017)

A lo largo del tiempo el modelo LSTM ha tenido varias aplicaciones; en el campo del análisis de datos, en el año 2005 en la conferencia "The annual International Joint Conference on Neural Networks (IJCNN)" siendo esta una conferencia insignia de la IEEE, se publicó un artículo llamado "Framewise Phoneme Classification with Bidirectional LSTM Networks" en el cual fue uno de los estudios pioneros en la aplicación de una red LSTM con propósito de reconocer y analizar de forma continua señales de voz. (Graves & Schmidhuber, 2005)

Otra aplicación destacable de las redes LSTM recae en la conducción autónoma. En el año 2017, una publicación realizada en la "IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)", con el título "An LSTM network for highway trajectory prediction" estudió la utilidad de las redes LSTM para la predicción de trayectorias y longitudes laterales de vehículos en la carretera para mejorar la conducción autónoma. (Altché & de La Fortelle, 2017)

Dentro del campo de la medicina, en el año 2015, un artículo publicado en la mundialmente reconocida conferencia de computación neuronal "Advances in Neural Information Processing Systems 28", llamado: "Parallel Multi-Dimensional LSTM, With Application to Fast Biomedical Volumetric Image Segmentation", logró satisfactoriamente la segmentación y análisis de imágenes cerebrales. (Stollenga, Byeon, Liwicki, & Schmidhuber, 2015)

En el año 2020, un grupo de investigadores canadienses de la universidad de Regina, realizaron un estudio llamado "Time series forecasting of COVID-19 transmission in Canada using LSTM networks" en el cual se pronosticaron los casos futuros del virus, a su vez se compararon las tasas de transmisión entre Canadá, Italia y Estados Unidos. (Reddy Chimmula & Lei, 2020)

Como se puede apreciar, la red LSTM se ha utilizado y se sigue utilizando para el análisis de datos y predicción de valores en diversas áreas de la ciencia y de la tecnología a nivel mundial.

CAPÍTULO 3
DISEÑO E IMPLEMENTACIÓN

En el presente capítulo se busca desarrollar un software modular para automatizar una bodega semiautomática mediante inteligencia artificial, más específicamente mediante la red neuronal LSTM, para lo cual se requiere un procesamiento de series temporales, con dicho software se logra la reserva de áreas dentro de una bodega, tomando en cuenta la rotación de los productos. Además, se requiere el desarrollo de un software dedicado a la asignación de productos en las áreas definidas con anterioridad.

Adicionalmente, se requiere una comunicación entre el script de Python y el servidor, con el objetivo de leer, actualizar y escribir en la base de datos, de esta forma se logra la integración completa con distintos módulos (datos, navegación e interfaz de usuario) correspondientes al equipo de trabajo de la empresa Robotrón.

3.1 Selección de herramienta de desarrollo de software

Para la selección de la herramienta de desarrollo de software, se optó por analizar distintos lenguajes de programación, librerías, facilidad de comunicación con el servidor y facilidad para exportar los scripts en formato ejecutable. Existen varios lenguajes de programación utilizados para desarrollar aplicaciones basadas en Inteligencia Artificial. Los más utilizados en este campo son Python y R, así como también el software Matlab por su orientación matemática y estadística. (Rojas, 2020)

La selección del software idóneo para el siguiente caso de estudio se la realizó mediante un análisis con criterios que faciliten la implementación del proyecto, los cuales se detallan a continuación.

a) Matlab

Matlab es una plataforma de programación dedicada a la computación numérica científica y de ingeniería, se utiliza para analizar datos, desarrollar algoritmos, crear modelos y representar datos y resultados de manera gráfica. (MathWorks, 2021)

Matlab cuenta con una gran cantidad de librerías integradas, llamadas Toolboxes. Todas las herramientas y funciones de Matlab son diseñadas para funcionar juntas de manera óptima. Gracias al uso de Matlab se han logrado miles de aplicaciones en diversas áreas de la ciencia y la tecnología, como son: sistemas de control, aprendizaje automático y

aprendizaje profundo, robótica, mantenimiento predictivo, etc. (Rojas, 2020) (MathWorks, 2021)

Matlab cuenta con la capacidad de implementar toda clase de algoritmos de Machine Learning, como algoritmos de clasificación, regresión, agrupación, etc. Así como también es capaz de implementar algoritmos complejos de Deep Learning. (Rojas, 2020)

Al ser una herramienta muy completa de análisis matemático y estadístico, al contar con una gran cantidad librerías y documentación dedicada, Matlab, es una gran opción como herramienta para el desarrollo de software basado en Inteligencia Artificial.

b) R

R es un lenguaje de programación dedicado al análisis y manipulación de datos estadísticos, cuanta con la capacidad de producir gráficos de alto nivel e interfaces de un conjunto de datos o resultados. (Team, 2021)

Cuenta con una gran variedad de herramientas para la realización de modelos lineales y no lineales, pruebas estadísticas, cálculos matemáticos complejos, manipulación de matrices vectores, y desarrollo de algoritmos de clasificación y agrupación. (Rojas, 2020)

R tiene potencial para el desarrollo de aplicaciones basadas en Inteligencia Artificial debido a librerías como Caret, la misma que permite construir modelos de Machine Learning proporcionando herramientas para la preparación de datos, división de conjuntos de datos, evaluar modelos de Machine Learning, etc. (Rojas, 2020)

La desventaja con la que cuenta R en comparación a otros lenguajes de programación como Python es la complejidad del lenguaje, ya que generalmente la curva de aprendizaje de R suele ser más lenta en comparación con la de Python. (Rojas, 2020)

c) Python

Python es un lenguaje de programación interpretado y orientado a objetos, posee una gran variedad de campos donde se lo utiliza como lenguaje de desarrollo, desde aplicaciones de Windows, páginas web, hasta servidores de red.

Python cuenta con una ventaja sustancial en la simplicidad de sintaxis y velocidad de desarrollo, es decir, se puede desarrollar un programa en Python con menos líneas de código que el mismo programa desarrollado en otros lenguajes de programación como Java o C. A su vez, al ser un lenguaje de sintaxis simple, se facilita la codificación

colaborativa. La desventaja con la que cuenta este lenguaje de programación es la velocidad de ejecución, la misma que es inferior frente a otros lenguajes.
(Rojas, 2020)
Otra ventaja del desarrollo con Python es el gran número de librerías que se pueden encontrar en comparación con otros lenguajes de programación, librerías dedicadas a Machine Learning y diversas áreas, facilitan el flujo de trabajo de inicio a fin. Python es de código abierto y cuenta con una extensa cantidad de recursos, documentación de calidad y una amplia comunidad en línea. (Rojas, 2020)

A continuación, en la Tabla 3.1 se resume las características principales de los 3 lenguajes de programación a manera de comparación

Tabla 3.1: Características principales de los lenguajes de programación más usados para el desarrollo de inteligencia artificial

Matlab	**R**	**Python**
Plataforma de programación orientada a la computación numérica.	Lenguaje de programación orientado al análisis y manipulación de datos estadísticos	Lenguaje de programación interpretado de propósito general.
Gran cantidad de Toolboxes con distintos propósitos: sistemas de control, robótica, análisis estadístico, inteligencia artificial, etc.	Cuenta con librerías para el desarrollo de modelos lineales y no lineales de análisis estadístico e inteligencia artificial.	Líder en cantidad de librerías destinadas al análisis numérico, matricial, cuenta con librerías para el desarrollo de algoritmos de inteligencia artificial de manera sencilla.
Curva de aprendizaje media, sintaxis de complejidad media.	Alta complejidad en la curva de aprendizaje del lenguaje.	Curva de aprendizaje rápida y sintaxis sencilla.
Documentación, recursos y comunidad online amplias.	Documentación, recursos y comunidad online amplias.	Documentación, recursos y comunidad online amplias.

Cuenta con planes de pago, costo estándar de $940 anuales y costo vitalicio de $2350.	Sin costo	Sin costo

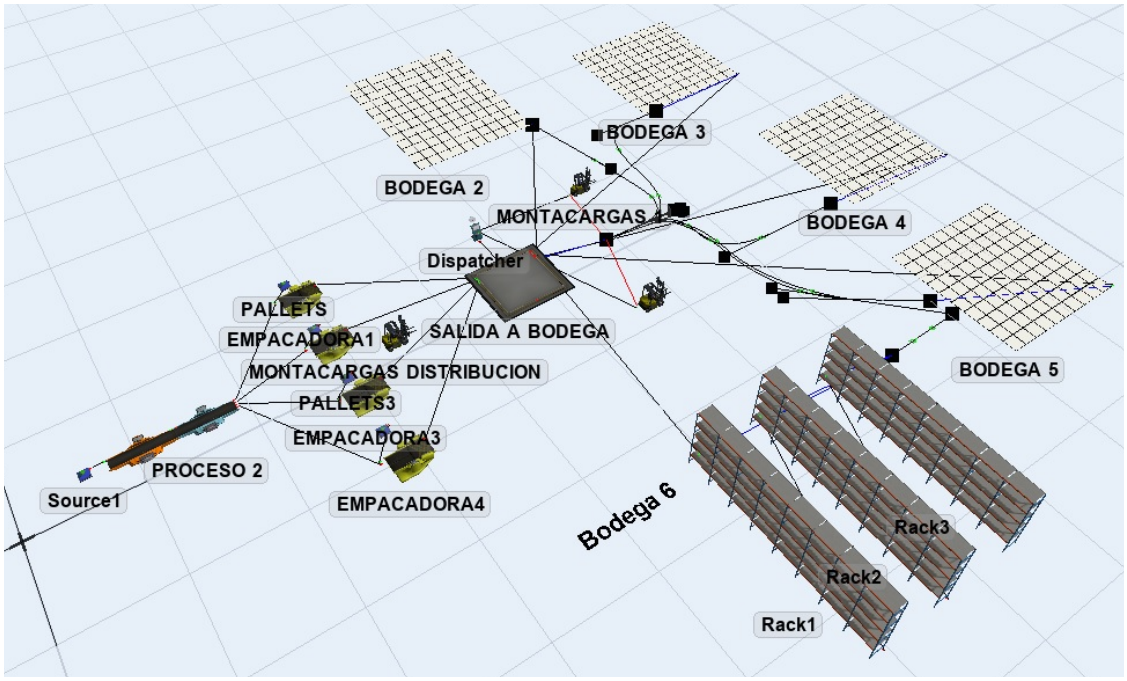
Resumen de características de los principales lenguajes de programación utilizados para el desarrollo de inteligencia artificial. Elaborado por: Alexander Guerrero

A partir del análisis realizado de los lenguajes de programación más comúnmente utilizados para el desarrollo software basado en Inteligencia Artificial, se concluye que todos los lenguajes son aptos para el caso de estudio ya que todos poseen las características y capacidades necesarias para hacer el trabajo. La diferencia radica en la facilidad de implementación e integración del software. Además de la cantidad de información y recursos referentes a librerías y herramientas que se puede encontrar en línea, por lo tanto, se optó por utilizar el lenguaje de programación Python, debido a que cumple a cabalidad con los requerimientos mencionados anteriormente.

3.2 Simulación y diseño de entorno

Para la simulación del entorno en bodega se optó por hacer uso del software FlexSim debido a su potencial de modelado 3D y su facilidad de modelado de procesos logísticos. Para la colocación de objetos, FlexSim utiliza los controles de arrastrar y soltar, haciéndolo bastante intuitivo y cómodo al momento de simular una línea de producción.

Figura 3.10: Vista general de simulación en FlexSim



Simulación de línea de producción, montacargas y disposición de bodegas en FlexSim. Elaborado por: Alexander Guerrero

Para el entorno de simulación se utiliza de un generador (Source 1), el cual produce cajas, es decir productos con diferentes propiedades al azar. A cada caja se le asigna una etiqueta y un color como se observa en la Figura 3.11, dichas etiquetas están dadas por la clasificación de prioridad ABCD expuestas en la Tabla 3.2, la misma que representa la rotación que tiene dicho producto dentro de la bodega. Esto se realiza con el objetivo de visualizar en que sectores de la bodega se están guardando los productos, planear su ubicación óptima priorizando a productos con ciclos de rotación más rápida en cada uno de los espacios designados de bodega.

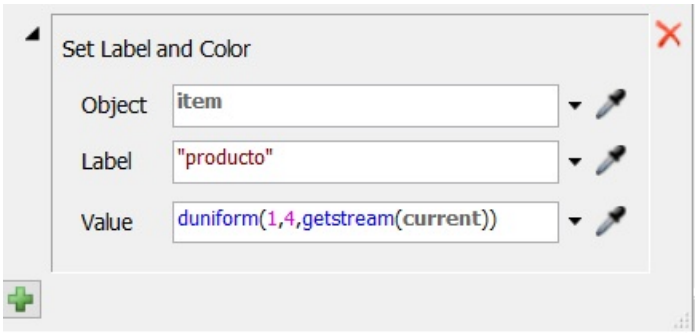
Las etiquetas designadas para los productos son las siguientes:

Tabla 3.2: Clasificación de prioridad ABCD y etiquetas asignadas

Etiqueta real	**Etiqueta en entorno de simulación**	**Rotación en bodega**
Producto A	1	Muy alta
Producto B	2	Alta
Producto C	3	Mediana
Producto D	4	Baja

Explicación detallada de la rotación de los productos con sus respectivas etiquetas utilizadas. Elaborado por: Alexander Guerrero.

Figura 3.11: Propiedades asignadas a generador de cajas en FlexSim



Generación de color y etiqueta dentro del entorno de simulación, Elaborado por: Alexander Guerrero.

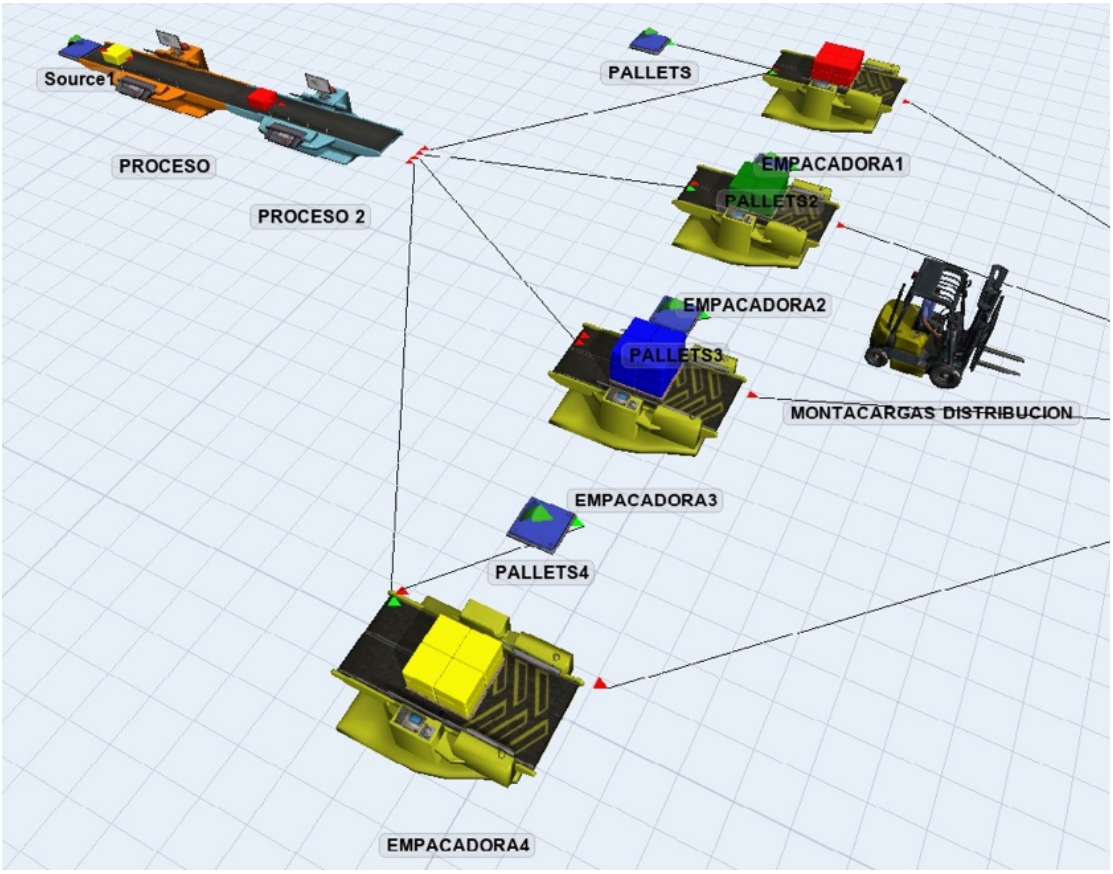
A continuación, los productos pasan por dos procesos que simulan una línea de producción y tratamiento de los productos. Al salir de dichos procesos, los productos son llevados a la zona de empaquetado donde son agrupados en pallets por tipo, es decir, por etiqueta. Una vez un pallet llega a su máxima capacidad este es considerado un Bin. De esta forma se consigue tener los Bines listos para empezar la distribución a zonas designadas de bodega.

Tabla 3.3: Colores distintivos de productos con prioridad ABCD para simulación en FlexSim

Prioridad de producto	Color en simulación
Producto A	Rojo
Producto B	Verde
Producto C	Azul
Producto D	Amarillo

Explicación detallada de prioridades de productos utilizadas por la simulación en FlexSim. Elaborado por: Alexander Guerrero.

Figura 3.12: Simulación de línea de producción en FlexSim



Simulación ejecutándose de línea de producción y empaquetado. Elaborado por: Alexander Guerrero

Una vez los productos son empaquetados en Bines, son llevados por un montacargas a la zona de salida a bodega, la misma que actúa como una línea de espera. Dentro del espacio de bodegas, dos montacargas son los encargados de llevar los productos a las zonas designadas. Mediante configuraciones realizadas a los objetos del entorno de simulación, se designaron zonas para cada producto, teniendo en cuenta la rotación de los mismos.

Es de vital importancia tener en cuenta que se trabajó con dos tipos de bodegas: espacios planos y estanterías (racks), cada bodega cuenta con una capacidad diferente.

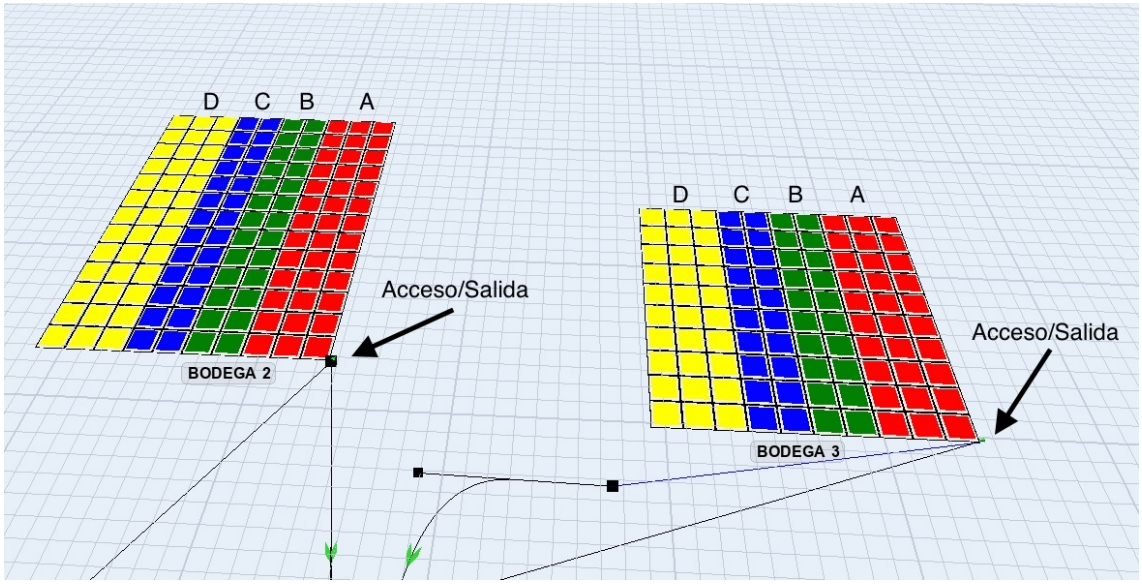
Tabla 3.4: Número de posiciones disponibles en cada espacio de bodega

Número de bodega	Capacidad (número de Bines)
Bodega 2	400
Bodega 3	400
Bodega 4	880
Bodega 5	880
Bodega 6	2040

Posiciones disponibles en cada espacio de bodega, no se toma en cuenta la bodega 1 ya que esta tiene como propósito almacenar otro tipo de productos. Elaborado por: Alexander Guerrero

Para Bodegas de tipo espacio plano, a los productos de tipo A y B se los ubica en zonas cercanas a la puerta de acceso/salida. A los productos de tipo C se los ubica en zonas cercanas al centro de los espacios de bodega y a productos tipo D al otro extremo de la bodega.

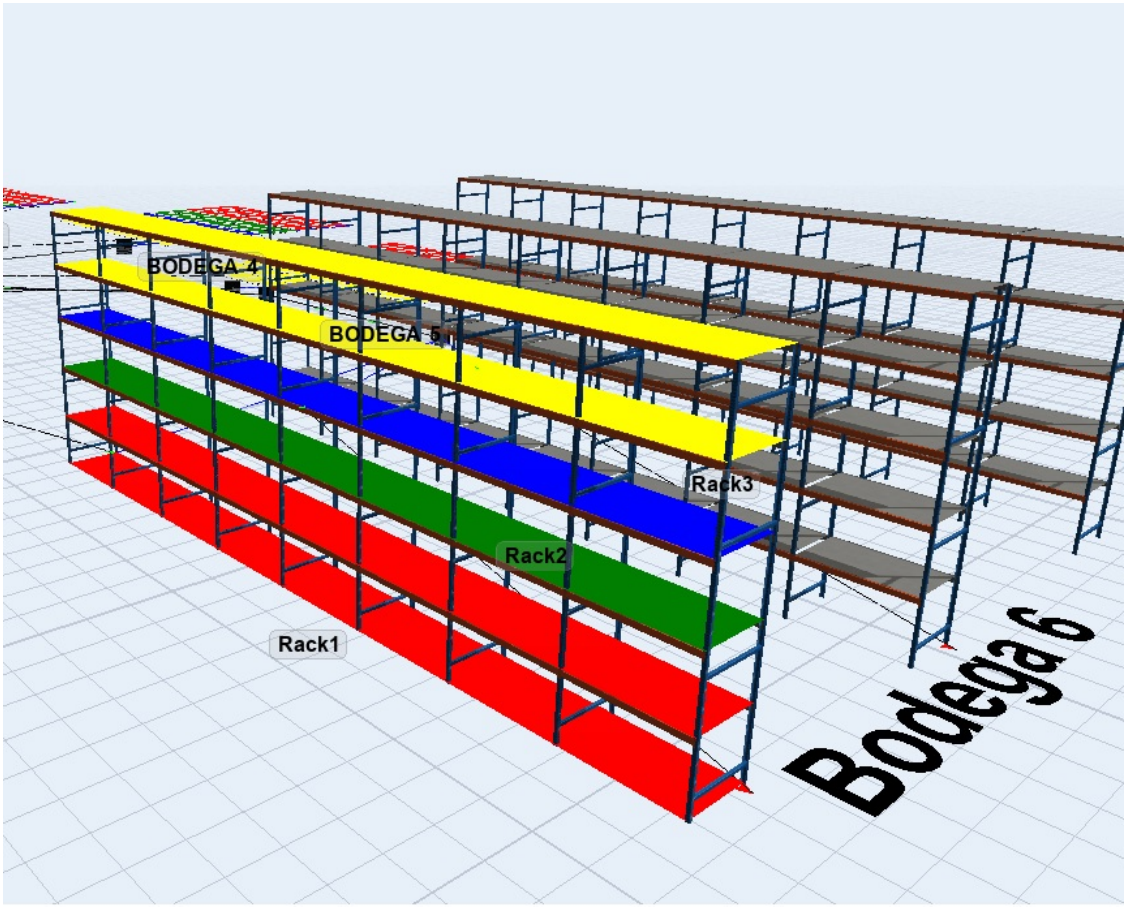
Figura 3.13: Disposición de zonas ABCD para bodegas de tipo espacio plano en FlexSim



Zonas de prioridad ABCD marcadas son sus respectivos colores dentro de bodegas de tipo espacio plano. Elaborado por: Alexander Guerrero

Para Bodegas de tipo Rack, los productos de tipo A y B se los ubica en los niveles más bajos, es decir nivel 1 y nivel 2. A los productos de tipo C se los ubica en el nivel 3 y a productos tipo D en los niveles 4 y 5.
Ambas disposiciones se diseñaron con el propósito de reducir los tiempos de ingreso y de despacho.

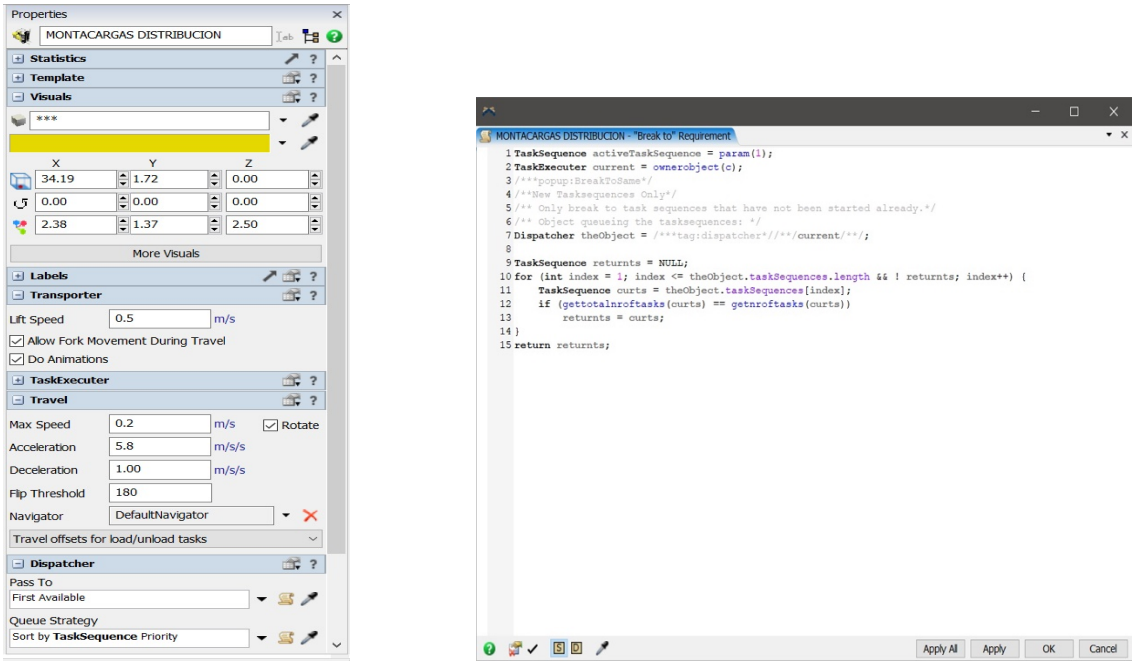
Figura 3.14: Disposición de zonas ABCD para bodegas de tipo Rack en FlexSim



Zonas de prioridad ABCD marcadas son sus respectivos colores dentro de bodega de tipo Rack. Elaborado por: Alexander Guerrero

Para la configuración de objetos, FlexSim cuenta con interfaces que permiten su configuración y parametrización dentro del espacio de trabajo, a su vez cuentan con propiedades pre programadas como se puede apreciar en la Figura 3.15 izquierda y también con la opción de programarlos a manera de código y personalizarlos a gusto propio como se muestra en la Figura 3.15 derecha.

Figura 3.15: Interfaz gráfica de configuración de propiedades (izquierda), interfaz de configuración de propiedades por código (derecha)

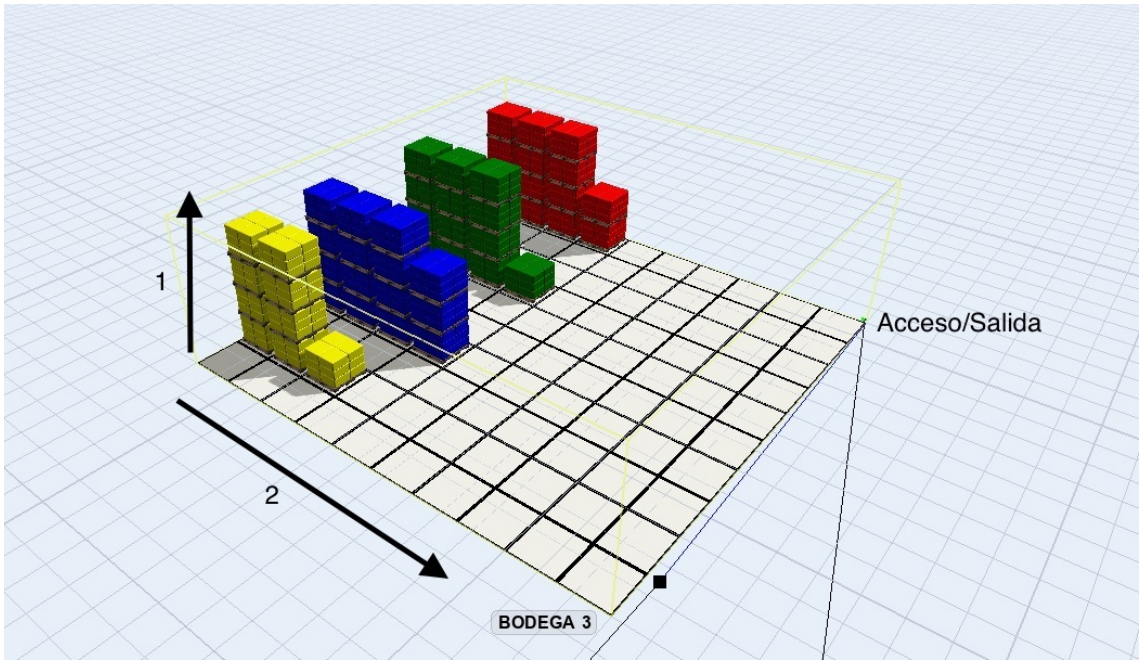


Distintas opciones de interfaces de configuración de un montacargas en FlexSim. Elaborado por: Alexander Guerrero

Una vez definida la disposición de zonas con prioridad ABCD y realizada la configuración básica de procesos y montacargas, se procede a diseñar el orden de llenado en las diferentes configuraciones de bodegas (espacio plano y racks).

Para espacios planos el orden de llenado debe ser formando columnas desde atrás hacia delante. Es decir, se debe asegurar que toda una columna sea llenada primero antes de llenar la siguiente.

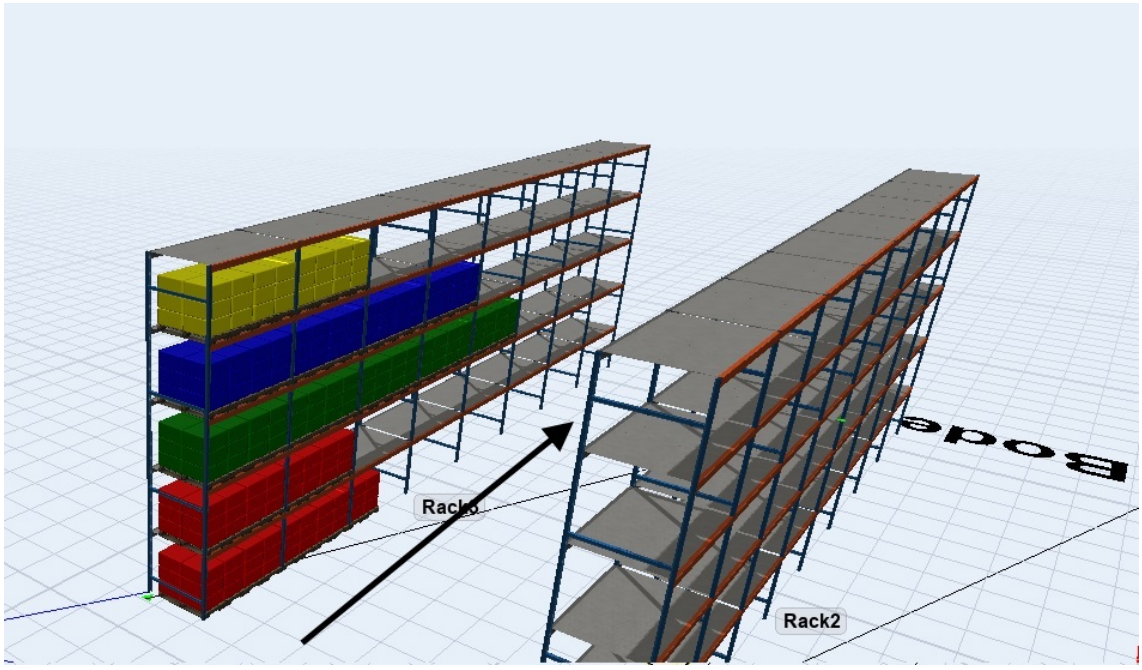
Figura 3.16: Orden de llenado para bodegas de tipo espacio plano en FlexSim



Orden de llenado en bodega de tipo espacio plano, llenan formando columnas de atrás hacia delante. Elaborado por: Alexander Guerrero

A diferencia del llenado en espacios planos, en Racks el orden de llenado debe ser a manera de niveles, y llenando un rack a la vez.

Figura 3.17: Orden de llenado para bodegas de tipo Rack en FlexSim



Orden de llenado en bodega de tipo rack, se llenan los niveles y un rack a la vez. Elaborado por: Alexander Guerrero

Una vez planteado el diseño de zonas de prioridad ABCD y teniendo claro el tipo de algoritmo se va a implementar en el desarrollo del software tanto para la fase de reservas, orden de llenado y asignación dentro de la bodega. Se procede a analizar las librerías requeridas para desarrollar el caso de estudio.

3.3 Librerías y recursos utilizados

a) Pandas

Pandas es una librería de Python dedicada específicamente para el manejo y análisis de datos. Esta librería permite leer archivos CSV, Excel, bases de datos, etc.

Con la ayuda de la librería Pandas, se puede realizar diversas acciones a un set de datos ya sean unidimensionales (listas) o bidimensionales (tablas). Para el caso de estudio, se usa la librería Pandas para trabajar con diversas tablas almacenadas un servidor basado en SQL (Structured Query Language).

b) Numpy

Numpy es una herramienta de Python orientada a la computación numérica. Con el uso de Numpy es posible trabajar con funciones matemáticas complejas, trabajar con vectores y matrices, realizar cálculos de algebra lineal, etc.

Para el caso de estudio se utiliza la librería Numpy para trabajar con vectores y matrices para la red neuronal LSTM.

c) TensorFlow y Keras

TensorFlow es una librería diseñada para Python cuyo objetivo es realizar modelos de Machine Learning y Deep Learning de manera sencilla. Keras es una API (Application Programming Interface) de TensorFlow, la cual es utilizada para construir modelos de Deep Learning.

Para el caso de estudio se utiliza TensorFlow y Keras para la construcción del modelo inteligente LSTM, para entrenarlo, y exportarlo para su implementación.

d) Scikit-learn

Scikit-learn (Sklearn) es una librería de Python dedicada al Machine Learning, presenta herramientas para el análisis de datos, es de código abierto y permite implementar algoritmos de clasificación, regresión, agrupación, etc.

Para el caso de estudio se utiliza la librería Sklearn y sus herramientas de métricas para obtener el error cuadrático medio y la raíz del error cuadrático medio.

e) Pyodbc

Pyodbc es una librería de código abierto diseñada para Python la cual permite crear una conexión ODBC con un servidor y una base de datos. Mediante esta librería además de permitir la conexión, lectura y escritura en bases de datos.

Para el caso de estudio se utiliza la librería Pyodbc con el objetivo de interactuar directamente con la base de datos, leer y escribir en ella.

f) Matplotlib

Matplotlib es una librería de Python dedicada a la visualización de datos, mediante esta librería se pueden representar gráficos y figuras de calidad y de manera precisa.

Para el caso de estudio se utiliza la librería Matplotlib con el objetivo de visualizar los datos de entrenamiento y resultados de las predicciones del algoritmo inteligente.

g) Auto-py-to-exe

Auto-py-to-exe es una interfaz gráfica que hace uso de la librería PyInstaller con el objetivo de transformar un script de Python .py a un archivo ejecutable .exe.

En el caso de estudio esta herramienta se utiliza para la conversión de los archivos .py a .exe, con el objetivo de que estos puedan ser ejecutados desde el servidor.

3.4 Diseño de software

El software fue diseñado cumpliendo los requerimientos propuestos por la empresa Robotrón, para el manejo de bodegas, mismos que se exponen a continuación. Dichos requerimientos que fueron previamente simulados en el software FlexSim. A su vez el software se divide en dos etapas: etapa de reservas, la cual es realizada por el software inteligente, basado la red neuronal LSTM y la etapa de asignación, la cual es realizada por otro software. Ambos trabajan en conjunto para lograr la automatización

3.4.1 Requerimientos para el funcionamiento del software

El software:

- Debe ser versátil y configurable de manera externa.
- Debe tener la capacidad de comunicarse con un servidor, leer y escribir en bases de datos de SQL Server.
- Debe cumplir con el orden de llenado definido para los distintos tipos de bodegas y con las características de zonas de prioridad ABCD.
- Debe dividirse en dos etapas: reservas con Inteligencia Artificial y asignación de espacios internos en áreas previamente reservadas.

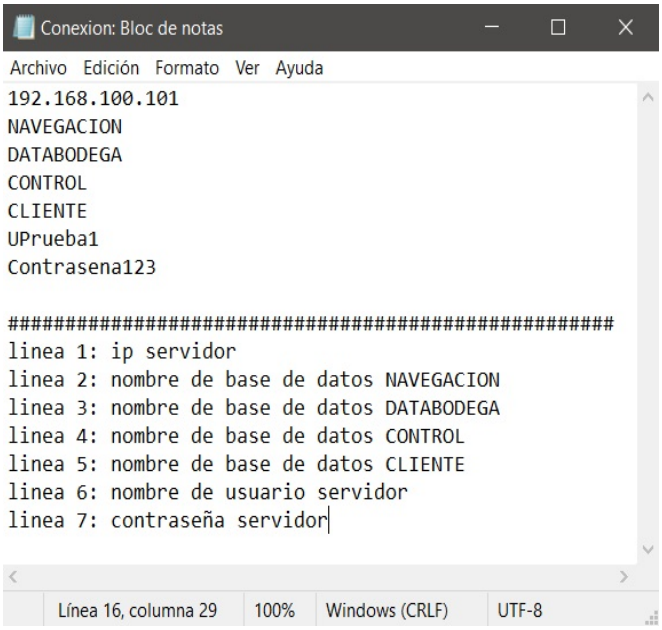
3.4.2 Configuración externa del software

La configuración externa del software se logró mediante la lectura de ficheros de texto .txt, alojados en un directorio en específico. Se implementó dos archivos de texto configurables.

El primer archivo de configuración es denominado Conexión.txt como se puede observar en la Figura 3.18, mediante el cual se manipulan parámetros como: dirección IP, Nombres de bases de datos, nombre de usuario del servidor y contraseña del servidor. De esta forma el software es libre de conectarse y leer datos, aun cuando se hagan cambios de IP o de nombres en las bases de datos del servidor. Estos parámetros son utilizados por la librería Pyodbc para realizar la conexión con el servidor.

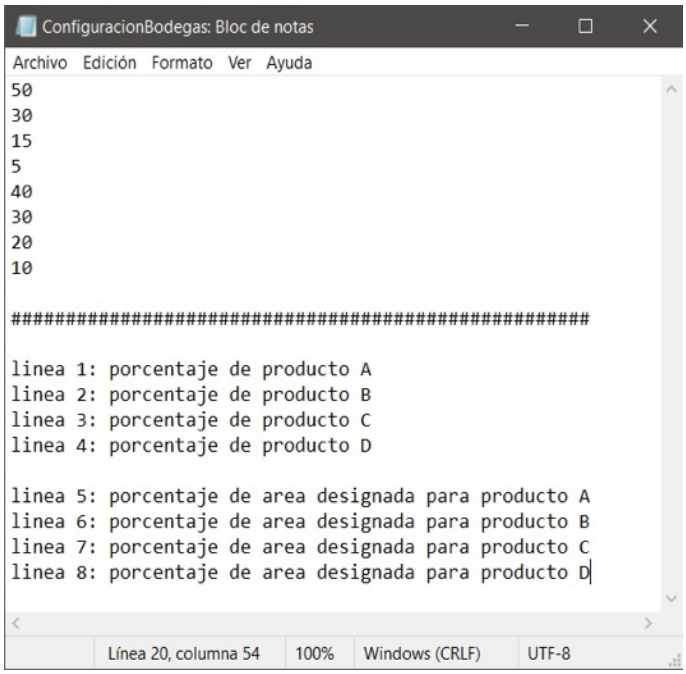
El segundo archivo de configuración es denominado ConfiguracionBodegas.txt como se puede observar en la Figura 3.19 y como su nombre lo indica, tiene la función de definir ciertos parámetros necesarios para las zonas de prioridad ABCD.

Figura 3.18: Configuración de conexión a servidor mediante Bloc de notas



Archivo .txt de configuración de conexión donde se detalla dirección IP, nombres de bases de datos, nombre de usuario y contraseña del servidor. Elaborado por: Alexander Guerrero.

Figura 3.19: Configuración de bodegas mediante Bloc de notas



Archivo .txt de configuración de porcentajes para zonas ABCD por bloc de notas. Elaborado por: Alexander Guerrero

El primer parámetro hace referencia a la cantidad de productos de forma porcentual que se desea que pertenezcan a las zonas de prioridades A, B, C y D del historial de ventas

respectivamente. El segundo parámetro hace referencia al porcentaje del área física que se desea designar para cada una de las zonas de prioridad A, B, C y D. Este parámetro solo influye a espacios planos ya que las zonas de prioridad ABCD en Racks permanece estática.

3.4.3 Conexión, lectura y escritura en base de datos de SQL Server

Para establecer la conexión y poder interactuar con el servidor y una base de datos, se procede a utilizar la librería Pyodbc en Python y el driver correspondiente para Windows, en el caso de estudio se utilizó ODBC (Open DataBase Connectivity) 17.

3.4.3.1 Conexión con servidor y bases de datos

Una vez almacenadas los datos de las filas de texto en variables internas de programa se procede a hacer uso de la librería anteriormente mencionada mediante un “try, except”.

Figura 3.20: Conexión a servidor y base de datos mediante Pyodbc

```
#Conexión con base de datos "CLIENTE"
try:
    conexion = pyodbc.connect('DRIVER={ODBC Driver 17 for SQL Server};SERVER=' +
                              IP + ';DATABASE=' + CLIENTE + ';UID=' + USUARIO + ';PWD=' + PASSWORD)
    print("OK! conexión exitosa con:", CLIENTE)
except Exception as e:
    print("Ocurrió un error al conectar a SQL Server: ", e)
```

Parámetros requeridos por la librería Pyodbc para realizar una conexión con un servidor y una base de datos. Elaborado por: Alexander Guerrero

3.4.3.2 Lectura de base de datos

Pyodbc permite realizar consultas directamente a un servidor SQL y a bases de datos alojadas en su interior, de esta forma con la librería Pandas se puede extraer las tablas que sean necesarias para el desarrollo del software.

Figura 3.21: Consulta a base de datos de SQL Server

```
#consulta a base de datos / obtener datos historicos para hacer predicción
df = pd.read_sql("SELECT * FROM datos_programa where estado_reserva is NULL order by producto, cod_cliente asc", con=conexion)
```

Consulta y extracción de tablas de base de datos mediante Pyodbc y Pandas. Elaborado por: Alexander Guerrero

3.4.3.3 Escritura en base de datos

Para modificar o insertar datos dentro de una tabla almacenada en una base de datos se hace uso de la conexión establecida previamente mediante Pyodbc, se llama a un procedimiento almacenado, el cual será el encargado de actualizar los campos dentro de la base de datos, requiere que se indiquen las variables a ser actualizadas.

Figura 3.22: Escritura en base de datos de SQL Server

```
try:
    with conexion.cursor() as cursor:
        update = " exec reservas_camaras_estado ?,? "
        cursor.execute(update, (num_producto, num_cod_cliente ))
    print('Estado actualizado')
except Exception as e:
    print("Ocurrió un error al insertar: ", e)
```

Ejecución de procedimiento almacenado invocado desde Python mediante la librería Pyodbc para actualizar campos de una tabla en una base de datos de SQL Server. Elaborado por: Alexander Guerrero

3.4.4 Orden de llenado de espacios de bodega

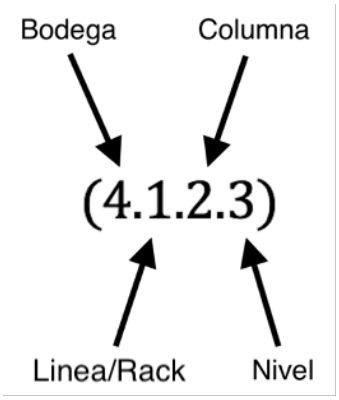
Para cumplir con las condiciones de orden de llenado para los dos tipos de espacios de bodegas del caso de estudio, es necesario primero comprender la notación con la que se describe a una ubicación dentro de la bodega, además de comprender los tipos de productos con los que se trabaja y a las áreas a las que estos pertenecen.

3.4.4.1 Notación de una ubicación dentro de la bodega

La notación se refiere a la manera en la que el equipo de trabajo de Robotrón decidió llamar a una ubicación en específico dentro del espacio de bodega.

La notación consta de cuatro números separados por puntos, como se indica la Figura 3.23.

Figura 3.23: Notación para ubicaciones en espacios de bodega



Notación de ubicación dentro de bodega de tipo espacio plano de tipo Rack. Elaborado por: Alexander Guerrero

De esta forma, al referirse a un producto en específico ubicado en la posición (4.1.2.3), se puede interpretar de la siguiente manera: el producto se encuentra en la bodega 4, Línea/Rack 1, columna 2, nivel 3.

3.4.4.2 Tipos de producto y áreas

Dentro de un espacio de bodega pueden estar almacenados más de un tipo de producto, los cuales están asociados directamente a un código de área (Figura 3.24). Esta información consta en la base de datos y puede ser modificada en cualquier momento. El nombre con el que se asocia a cada área se conoce como tipo de producto o nombre de área.

Figura 3.24: Código de área y nombre de área / Tipo de producto

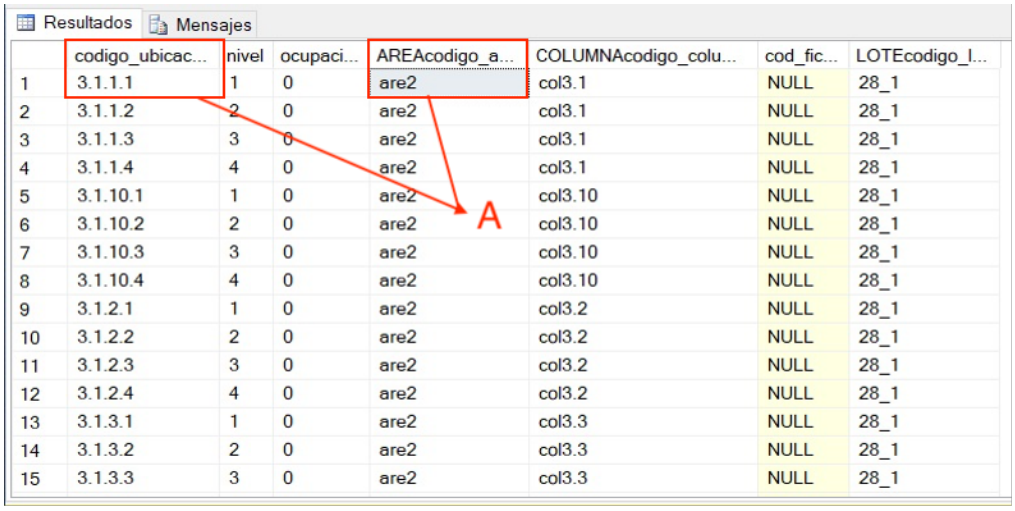
Resultados | Mensajes

	codigo_area	nombre_area
1	are1	Saldo Contenedor
2	are2	Traslape
3	are3	Producto Testigo
4	are4	Producto Terminado
5	are5	Lista de desechos
6	are6	Producto Nuevo
7	are7	Listo para despacho
8	are8	Cuarentena

Captura de base de datos exponiendo los ocho diferentes códigos de área con sus respectivos nombres. Elaborado por: Alexander Guerrero.

Las áreas están conjuntamente asociadas a las posiciones dentro de la bodega como se muestra en la Figura 3.25, etiqueta “A”, de este modo se puede afirmar que un área es un conjunto de posiciones dentro de un espacio de bodega. Esto se realiza con el propósito de tener espacios determinados con un mismo tipo de producto dentro de las zonas de prioridad ABCD.

Figura 3.25: Ubicaciones de bodega asociadas a un código de área



	codigo_ubicac...	nivel	ocupaci...	AREAcodigo_a...	COLUMNAcodigo_colu...	cod_fic...	LOTEcodigo_l...
1	3.1.1.1	1	0	are2	col3.1	NULL	28_1
2	3.1.1.2	2	0	are2	col3.1	NULL	28_1
3	3.1.1.3	3	0	are2	col3.1	NULL	28_1
4	3.1.1.4	4	0	are2	col3.1	NULL	28_1
5	3.1.10.1	1	0	are2	col3.10	NULL	28_1
6	3.1.10.2	2	0	are2	col3.10	NULL	28_1
7	3.1.10.3	3	0	are2	col3.10	NULL	28_1
8	3.1.10.4	4	0	are2	col3.10	NULL	28_1
9	3.1.2.1	1	0	are2	col3.2	NULL	28_1
10	3.1.2.2	2	0	are2	col3.2	NULL	28_1
11	3.1.2.3	3	0	are2	col3.2	NULL	28_1
12	3.1.2.4	4	0	are2	col3.2	NULL	28_1
13	3.1.3.1	1	0	are2	col3.3	NULL	28_1
14	3.1.3.2	2	0	are2	col3.3	NULL	28_1
15	3.1.3.3	3	0	are2	col3.3	NULL	28_1

Captura de base de datos exponiendo las posiciones de la bodega 3 asociadas al área 2, esta a su vez es llamada Traslape (Figura 3.24). por: Alexander Guerrero.

Para tener una secuencia de llenado correcta es necesario ordenar las ubicaciones de tal forma que cumplan con las condiciones deseadas para cada tipo de espacio de bodega. Para realizar un ordenamiento de posiciones es necesario leer la tabla dbo.UBICACIÓN de la base de datos DATABODEGA (Figura 3.25), en la misma se encuentran todas las posiciones de la todas las bodegas. Se realiza una consulta general y se almacena las posiciones en distintas listas dependiendo su estado de ocupación.

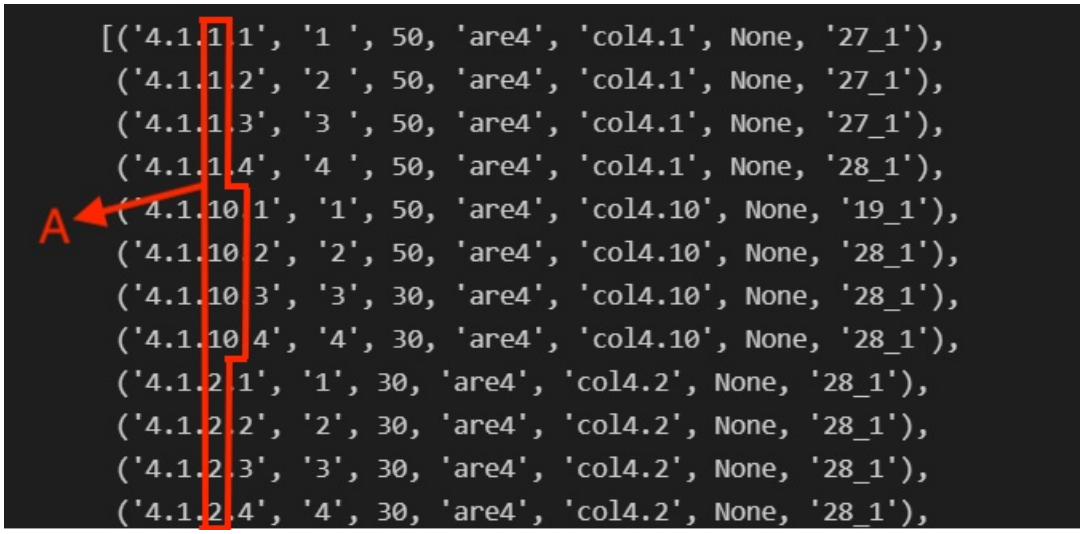
Tabla 3.5: Estado de ocupación de ubicaciones dentro de espacios de bodega

Estado de ocupación	**Descripción**
0	Ubicación libre
30	Ubicación con reserva
50	Ubicación asignada
100	Ubicación ocupada

Representación de los estados de ocupación utilizados por el equipo de trabajo de Robotrón. por: Alexander Guerrero.

Previo el ordenamiento especifico requerido por cada bodega, es necesario ordenar las posiciones de manera general de menor a mayor. Como se puede observar en la Figura 3.26, la etiqueta “A” resalta el orden de las columnas sin ordenar, como se puede apreciar el orden inicia en 1, seguido salta a 10 y posteriormente a 2. Esto se debe a que las posiciones extraídas directamente de la base de datos están ordenadas tomando en cuenta el primer número, es decir la base de datos ordena primero todos los números 1, posteriormente los números 2 y así sucesivamente. Además, como se puede apreciar, varios de los datos extraídos son innecesarios para el ordenamiento.

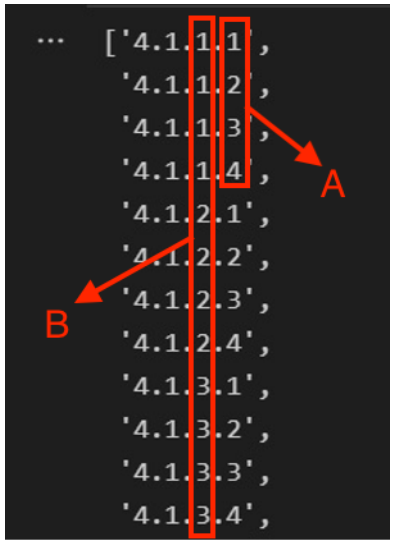
Figura 3.26: Lectura inicial de ubicaciones en base de datos, secuencia desordenada



Secuencia de ubicaciones desordenada en base de datos, misma que se ordena automáticamente teniendo en cuenta los números uno. Elaborado por: Alexander Guerrero.

Una vez aplicado el primer algoritmo de ordenamiento se logra tener una secuencia que va de menor a mayor, como se muestra en la Figura 3.27, la etiqueta “A” muestra el orden de los niveles de menor a mayor, en la Figura 3.27, etiqueta “B” se observa como el orden a diferencia de la Figura 3.26, ahora es secuencial, de menor a mayor. Además, que se eliminan datos no necesarios para procesos futuros.

Figura 3.27: Secuencia de ubicaciones ordenada de menor a mayor



Secuencia de ubicaciones ordenada en Python. Elaborado por: Alexander Guerrero.

Una vez aplicados algoritmos de ordenamiento se obtiene la secuencia deseada para el llenado de productos dentro de la bodega, a esta nueva secuencia de ubicaciones se le debe quitar las posiciones ocupadas (estado de ocupación 50) y reservadas previamente (estado de ocupación 30) con el objetivo de no asignar una ubicación equivocada

3.4.4.3 Orden de llenado en espacios planos

Para el ordenamiento de espacios planos es necesario tener en cuenta que se prioriza el ingreso formando columnas en la dirección desde el fondo de la bodega hacia la entrada, como se muestra en Figura 3.16.

La Figura 3.28 muestra la secuencia ya ordenada de llenado en bodegas de tipo espacios planos. Como se puede apreciar en la etiqueta “A”, el número de nivel aumenta hasta llegar al nivel máximo. Al observar la etiqueta “B”, se aprecia como el número de columna disminuye, es decir que se está llenando de atrás hacia delante.

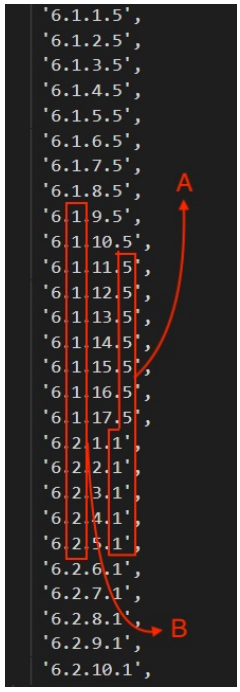
Figura 3.28: Orden de llenado deseado para bodegas de tipo espacio plano

Secuencia de ubicaciones ordenada para bodegas de tipo espacios planos en Python. Elaborado por: Alexander Guerrero.

3.4.4.4 Orden de llenado en Racks

Para el ordenamiento en Racks es necesario tener en cuenta que se prioriza el ingreso por niveles como se muestra en Figura 3.17. En la Figura 3.29, se muestra la secuencia ya ordenada de llenado en bodegas de tipo Rack.

Figura 3.29: Orden de llenado deseado para bodegas de tipo Rack



Secuencia de ubicaciones ordenada para bodegas de tipo Rack en Python. Elaborado por: Alexander Guerrero.

Como se puede apreciar en la etiqueta “A”, el número de nivel se mantiene constante hasta que todas las demás columnas se hayan llenado. Al observar la etiqueta “B”, se aprecia como el número de Rack cambia de 1 a 2 cuando el nivel ha llegado a su valor máximo de 5 y todas las columnas se han llenado.

3.4.5 Diseño de la red neuronal LSTM

Para el diseño e implementación de la red LSTM, se dispone de 4 capas principales en la arquitectura de la red, en rasgos generales, se tiene una capa de entrada, la cual serán los datos del históricos de almacenamiento dichos datos pasarán a través de 2 capas ocultas la cual contendrá la arquitectura de la red neuronal LSTM para producir una salida, en este caso una predicción en series temporales.

a) Capa de entrada

En la entrada de la red neuronal, se tienen datos históricos de 10 clientes y 30 productos, almacenados de forma diaria desde enero de 2017 hasta mayo de 2021. Se tiene un total de 48360 datos por cliente. El dato objetivo a predecir es el espacio ocupado por cada cliente y cada producto.

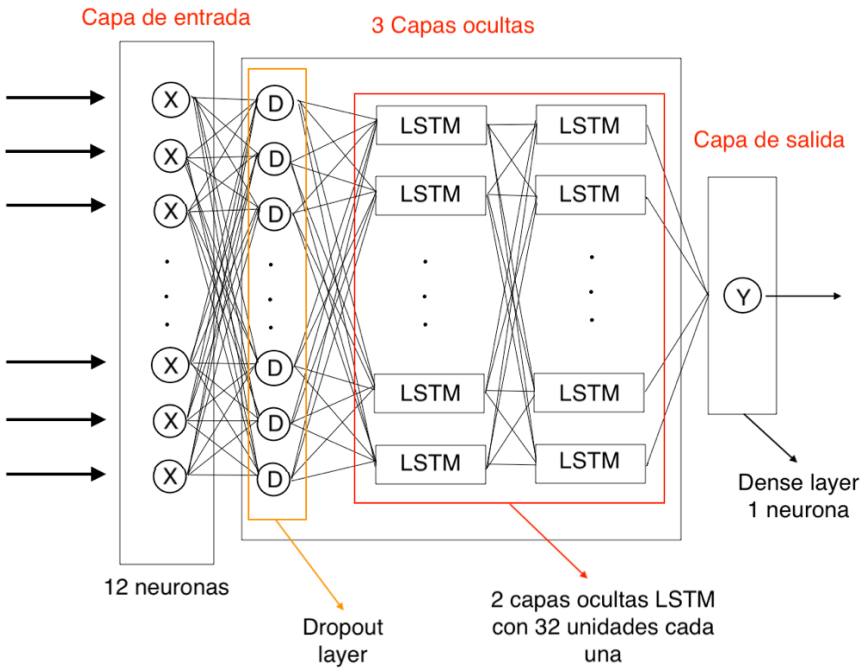
b) Capas ocultas

Para las capas ocultas se tiene la arquitectura de la red LSTM, la misma que en el caso de estudio consta de 1 capa de Dropout al 20% y 2 capas LSTM con 32 unidades cada una, donde se analiza la serie de tiempo del comportamiento de cada cliente y de cada producto, de esta forma para obtener la predicción de salida

c) Capa de salida

Como capa de salida se tiene una capa tipo Densa con una neurona de salida, la cual recoge toda la información generada por las capas ocultas y genera predicciones individuales para la serie temporal dentro de un ciclo repetitivo.

Figura 3.30: Diagrama de la arquitectura de la red neuronal



Esquema detallado a manera de capas de la red neuronal implementada para el caso de estudio. Elaborado por: Alexander Guerrero.

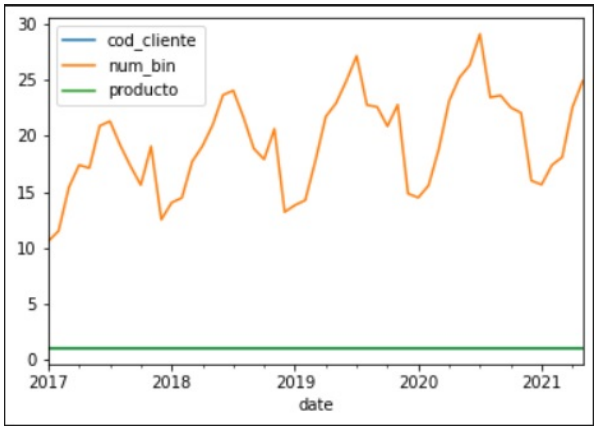
3.4.5.2 Entrenamiento de la red neuronal

Para el entrenamiento de la red neuronal se desglosaron los datos de tal forma en que cada cliente y cada producto tenga su serie temporal independiente, dado que no todos los productos ni todos los clientes tienen el mismo comportamiento a lo largo del tiempo.

A su vez, los datos de entrenamiento deben estar en el formato que se espera la predicción, en el caso de estudio, la predicción será mensual por lo tanto los datos de entrenamiento deben ser agrupados de manera mensual.

De esta forma se obtiene una gráfica a manera de serie temporal, como se puede apreciar el comportamiento es creciente desde el primer registro en 2017 hasta el último en 2021.

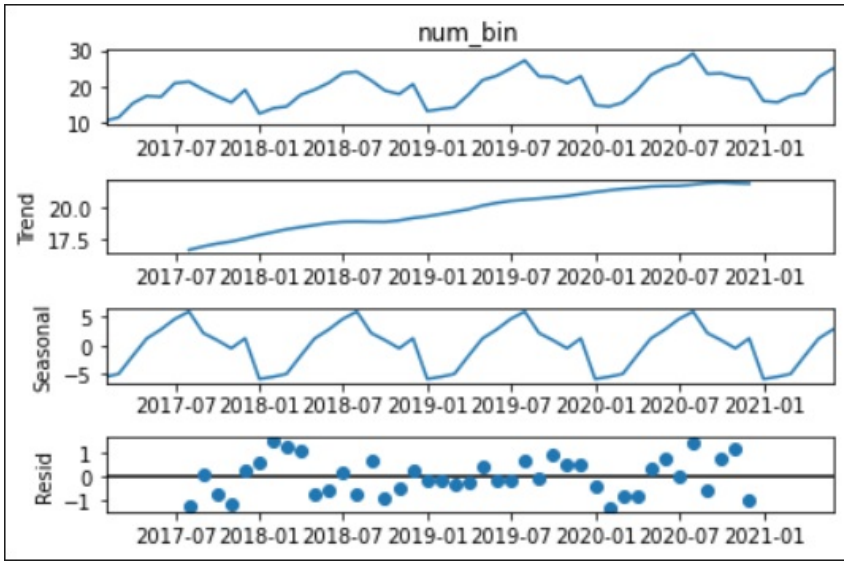
Figura 3.31: Gráfica de serie temporal de datos de entrada



Gráfica temporal de número de posiciones ocupadas en bodega desde 2017 hasta 2021. Elaborado por: Alexander Guerrero.

Analizando la gráfica más a profundidad se observa claramente la línea de tendencia, la componente estacionaria de la gráfica y los datos residuales o ruido, como se puede apreciar en la Figura 3.32.

Figura 3.32: Análisis de datos de entrada



Análisis de tendencia, componente estacionaria y ruido en serie temporal del número de posiciones ocupadas en bodega desde 2017 hasta 2021. Elaborado por: Alexander Guerrero.

Para entrenar redes neuronales recurrentes es de vital importancia dividir el set de datos de entrada en 2 partes, entrenamiento y validación.

El set de datos de entrenamiento como su nombre lo indica, se encarga de entrenar el modelo haciéndolo pasar por la red LSTM.

El set de validación, al contrario, son datos desconocidos para la red LSTM, por lo tanto, con estos datos se prueba el modelo generando predicciones y así validando la eficacia de las mismas sobre estos datos.

Para el caso de estudio se dividió el set de datos en 80% para entrenamiento y 20% para validación. Es decir, los datos de entrenamiento son desde enero de 2017 hasta junio de 2020 y los datos de validación desde julio de 2020 hasta mayo de 2021

a) Configuración de la red neuronal LSTM

Se diseñó un modelo con capas LSTM mediante la herramienta Keras en Python, los parámetros de diseño como el número de capas LSTM, las unidades por capa, las épocas de entrenamiento, la función de activación, se manipularon hasta obtener un comportamiento adecuado de la red neuronal con los datos de entrenamiento. Es importante aclarar que el manipular los parámetros de la red neuronal puede afectar positiva o negativamente al modelo.

Figura 3.33: Configuración del modelo de red neuronal LSTM

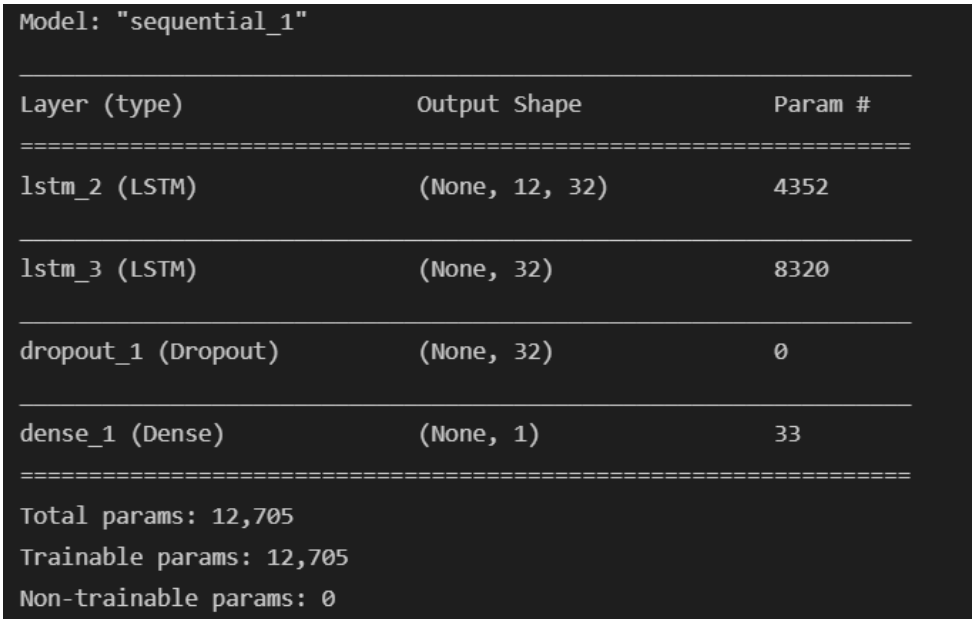
```
from keras.models import Sequential
from keras.layers import Dense, LSTM, Dropout

n_features = 1
n_input = 12
generator = TimeseriesGenerator(scaled_train, scaled_train, length=n_input, batch_size=1)

model = Sequential()
model.add(LSTM(32, activation='relu', return_sequences=True, input_shape=(n_input, n_features)))
model.add(LSTM(32, activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(1))
model.compile(optimizer='adam', loss='mean_squared_error')
model.summary()
history = model.fit(generator,epochs=100)
```

Parámetros óptimos para entrenamiento de red neuronal LSTM con los datos de entrada. Elaborado por: Alexander Guerrero.

Figura 3.34: Resumen de la arquitectura de la red neuronal

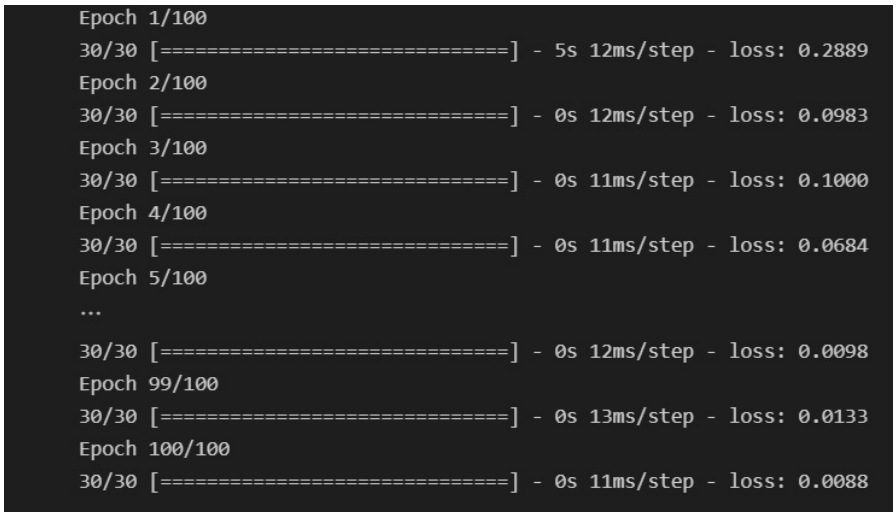


Resumen de arquitectura de red neuronal LSTM detallada por número de capas y neuronas. Elaborado por: Alexander Guerrero.

La Figura 3.34, muestra el resumen detallado de la red neuronal donde se aprecia las dimensiones de cada una de las capas, la primera capa oculta LSTM tiene una dimensión de (1x12x32), la segunda capa LSTM y la capa de Dropout tienen una dimensión de (1x32), finalmente la capa de salida tiene una dimensión de (1x1)

Posterior a realizar las configuraciones de entrenamiento de la red neuronal LSTM, se procede a entrenar el modelo durante 100 épocas

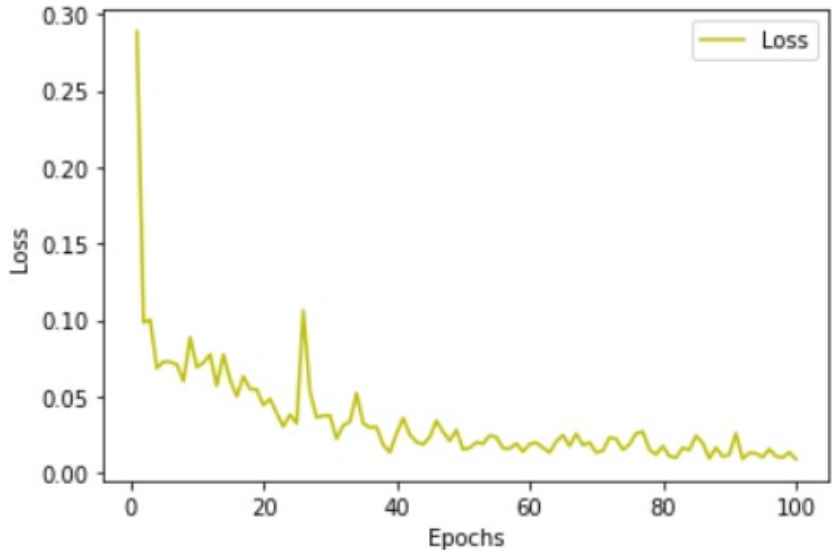
Figura 3.35: Entrenamiento de la red neuronal LSTM



Entrenamiento de red neuronal LSTM 100/100 épocas. Elaborado por: Alexander Guerrero.

En las Figuras 3.35 y 3.36, se muestra el comportamiento a lo largo de las 100 épocas de entrenamiento de la red neuronal, como se puede apreciar conforme se van pasando los ciclos de entrenamiento, la perdida se hace cada vez más pequeña.

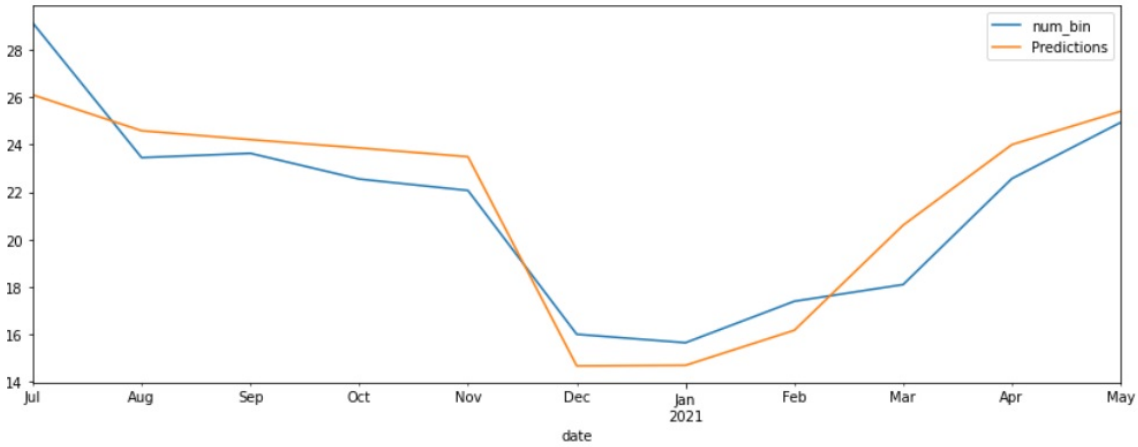
Figura 3.36: Comportamiento de la perdida durante el entrenamiento de la red neuronal LSTM



Perdida durante el entrenamiento de red neuronal LSTM 100/100 épocas. Elaborado por: Alexander Guerrero.

Una vez el modelo ha sido entrenado es hora de ponerlo a prueba haciendo una predicción con datos distintos a los que la red neuronal fue entrenada, es decir, con los datos de validación. Como se puede observar en la Figura 3.37, la línea azul representa los datos reales del espacio ocupado dentro de la bodega del set de validación y la línea naranja representa la predicción generada por la red neuronal LSTM.

Figura 3.37: Predicción obtenida con red neuronal LSTM



Predicción de red neuronal LSTM. Elaborado por: Alexander Guerrero.

A continuación, se procede a guardar y exportar el modelo para posteriormente probarlo con los diferentes datos de otros clientes y productos para validar el funcionamiento el modelo.

Figura 3.38: Exportar modelo previamente entrenado

```
import os.path
if os.path.isfile('models/Prediccion_LSTM.h5') is False:
    model.save('models/Prediccion_LSTM.h5')
    print('Modelo exportado')
✓ 2.2s
Modelo exportado
```

Código de exportación de modelo en Python. Elaborado por: Alexander Guerrero.

b) Algoritmo de reservas basado en red neuronal LSTM

Las Figuras 3.39, 3.40 y 3.41, describen a manera de diagrama de flujo el funcionamiento del algoritmo de reservas haciendo uso de la red neuronal previamente entrenada y exportada.

Figura 3.39: Diagrama de flujo de algoritmo de reservas con red neuronal LSTM

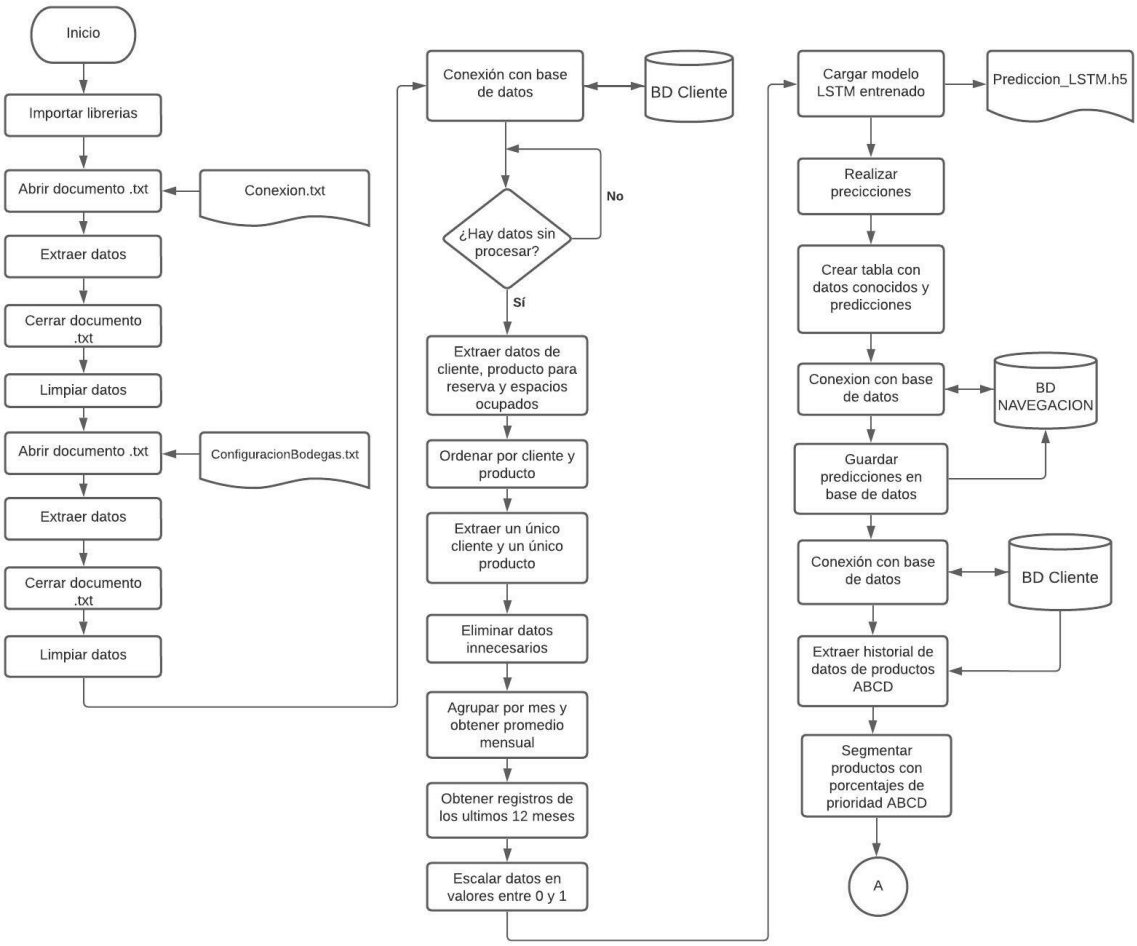


Figura 3.40: Continuación de diagrama de flujo de algoritmo de reservas con red neuronal LSTM

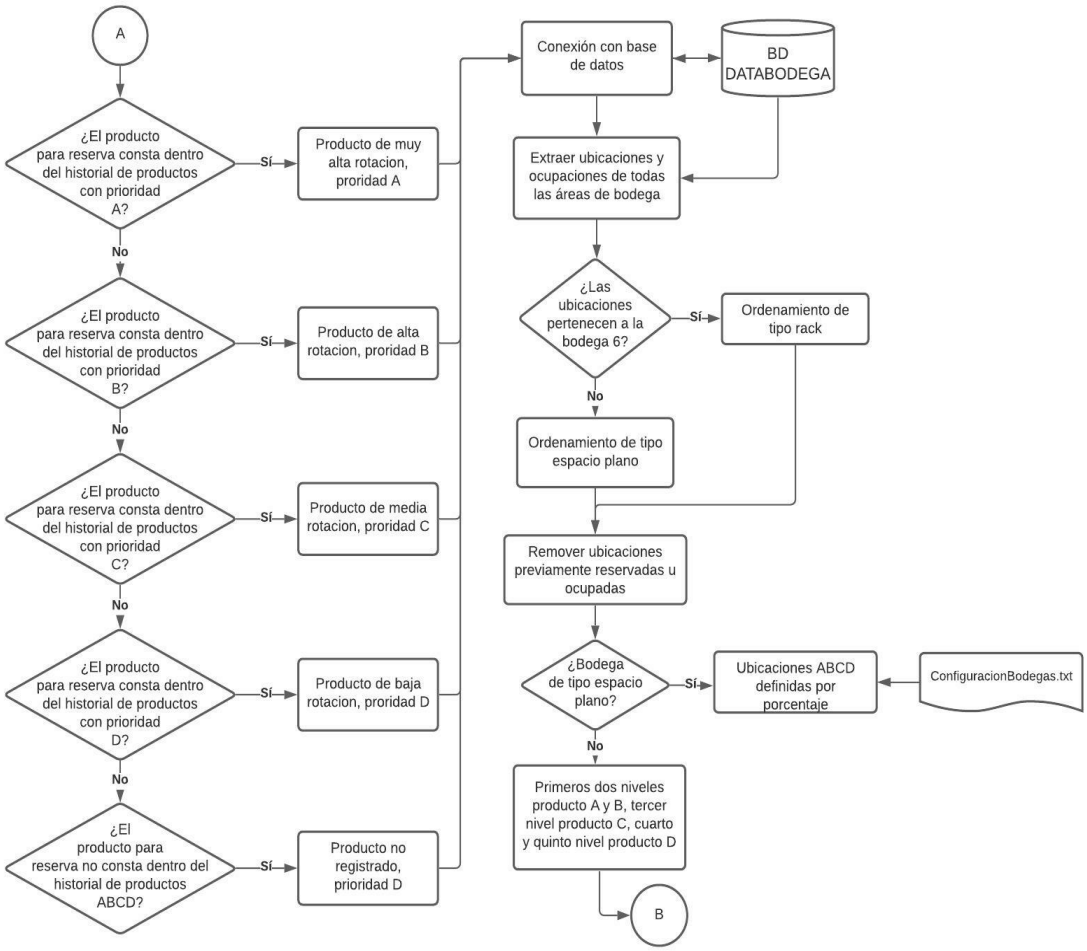
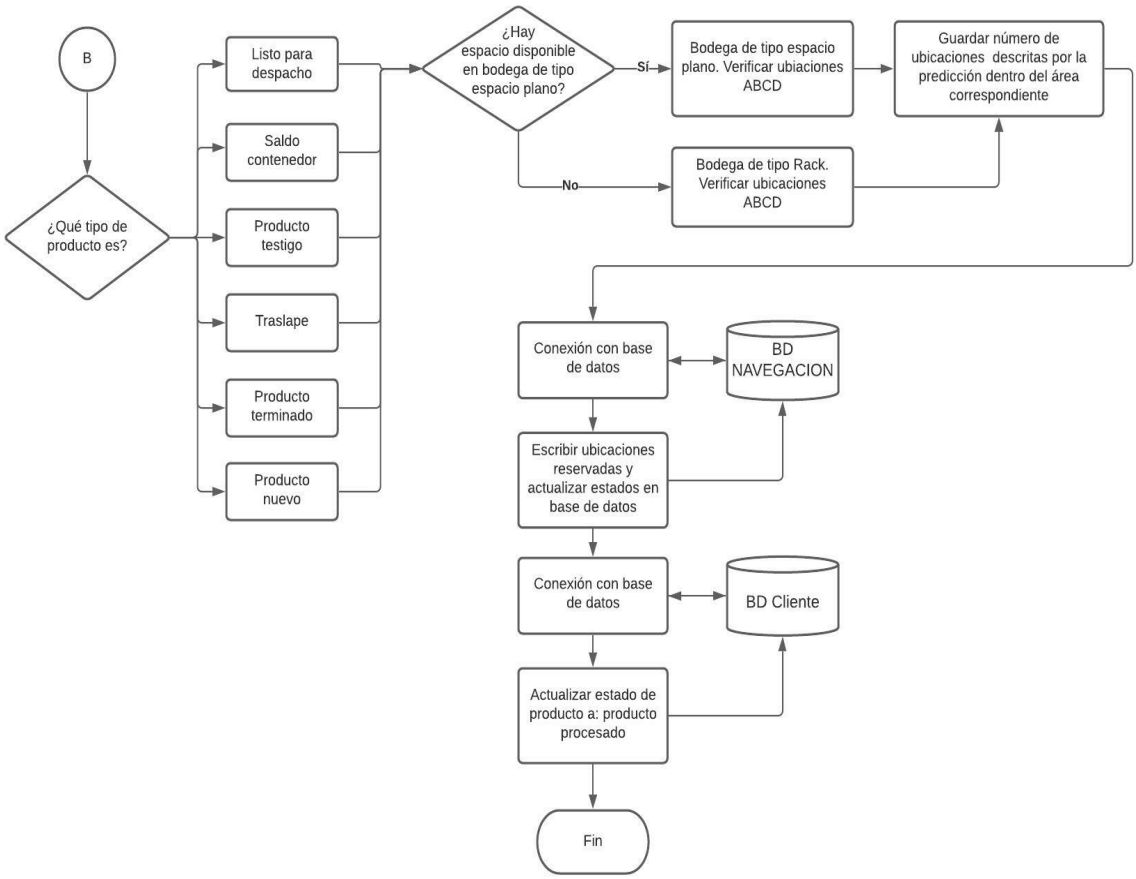


Figura 3.41: Continuación de diagrama de flujo de algoritmo de reservas con red neuronal LSTM



El algoritmo se ejecutará a manera de un archivo ejecutable Reservas.exe. Como primer paso, el algoritmo importa las librerías necesarias para su funcionamiento, lee, extrae y almacena las configuraciones de los archivos .txt (Bloc de notas) para la conexión con el servidor y bases de datos y configuración de bodegas.

A continuación, el algoritmo accede a la base de datos Cliente, donde extrae datos de la tabla dbo.datos_programa, en esta se encuentran los datos históricos de clientes y productos no conocidos por la red neuronal, se procede a agrupar obteniendo una serie temporal a la vez de un cliente y un producto a manera mensual.

Seguido, se escala y se da formato a los datos para servir de entrada a la red neuronal. Se carga el modelo y mediante un ciclo “for” se realizan predicciones en un periodo de 3 meses.

Figura 3.42: Cargar modelo previamente entrenado

```
model = load_model("D:\LSTM\models\Prediccion_LSTM.h5")
```

Código para carga modelo en Python. Elaborado por: Alexander Guerrero.

A continuación, las predicciones realizadas son almacenadas en una tabla de la base de datos.

Para las reservas de espacio dentro de la bodega, se toma en cuenta únicamente la primera predicción que corresponde al mes siguiente de cuando se realiza.

El algoritmo se conecta a la base de datos DATABODEGA, extrae las ubicaciones disponibles, las ordena dependiendo el tipo de producto y al tipo de bodega que pertenezca y procede a realizar la reserva, actualizando el estado de ocupación a 30 en la tabla dbo.UBICACIÓN. El proceso se realiza a manera secuencial hasta culminar con las reservaciones correspondientes a cada producto y cada cliente.

Figura 3.43: Ubicaciones reservadas

```
tipo de producto: Producto Terminado
prediccion: 88
numero de ubicaciones reservadas: 88
ubicaciones reservadas: ['4.3.5.1', '4.3.5.2', '4.3.5.3', '4.3.5.4', '4.3.4.1',
'4.3.4.2', '4.3.4.3', '4.3.4.4', '4.3.3.1', '4.3.3.2', '4.3.3.3', '4.3.3.4',
'4.3.2.1', '4.3.2.2', '4.3.2.3', '4.3.2.4', '4.3.1.1', '4.3.1.2', '4.3.1.3',
'4.3.1.4', '4.4.10.1', '4.4.10.2', '4.4.10.3', '4.4.10.4', '4.4.9.1', '4.4.9.2',
'4.4.9.3', '4.4.9.4', '4.4.8.1', '4.4.8.2', '4.4.8.3', '4.4.8.4', '4.4.7.1',
'4.4.7.2', '4.4.7.3', '4.4.7.4', '4.4.6.1', '4.4.6.2', '4.4.6.3', '4.4.6.4',
'4.4.5.1', '4.4.5.2', '4.4.5.3', '4.4.5.4', '4.4.4.1', '4.4.4.2', '4.4.4.3',
'4.4.4.4', '4.4.3.1', '4.4.3.2', '4.4.3.3', '4.4.3.4', '4.4.2.1', '4.4.2.2',
'4.4.2.3', '4.4.2.4', '4.4.1.1', '4.4.1.2', '4.4.1.3', '4.4.1.4', '4.5.10.1',
'4.5.10.2', '4.5.10.3', '4.5.10.4', '4.5.9.1', '4.5.9.2', '4.5.9.3', '4.5.9.4',
'4.5.8.1', '4.5.8.2', '4.5.8.3', '4.5.8.4', '4.5.7.1', '4.5.7.2', '4.5.7.3',
'4.5.7.4', '4.5.6.1', '4.5.6.2', '4.5.6.3', '4.5.6.4', '4.5.5.1', '4.5.5.2',
'4.5.4.2', '4.5.4.3', '4.5.4.4', '4.5.3.1', '4.5.3.2', '4.5.3.3']
```

Se realiza una reserva de 88 posiciones dentro de la bodega 4 correspondiente al tipo de producto, producto terminado. Elaborado por: Alexander Guerrero.

3.5 Diseño del software de asignación

La Figura 3.44, describe a manera de diagrama de flujo el funcionamiento del algoritmo de asignación de ubicaciones.

Figura 3.44: Diagrama de flujo de algoritmo de asignación de ubicaciones

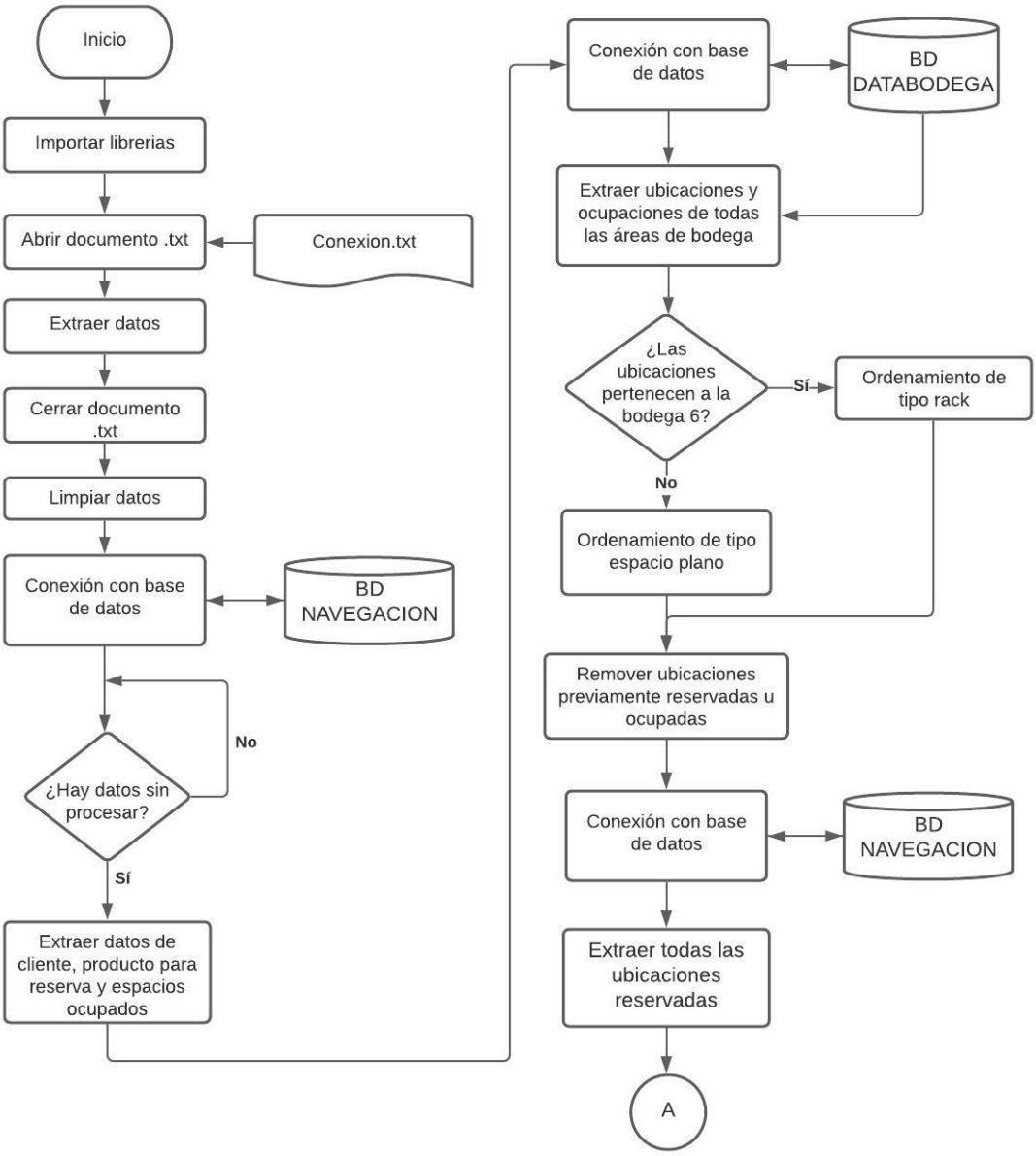
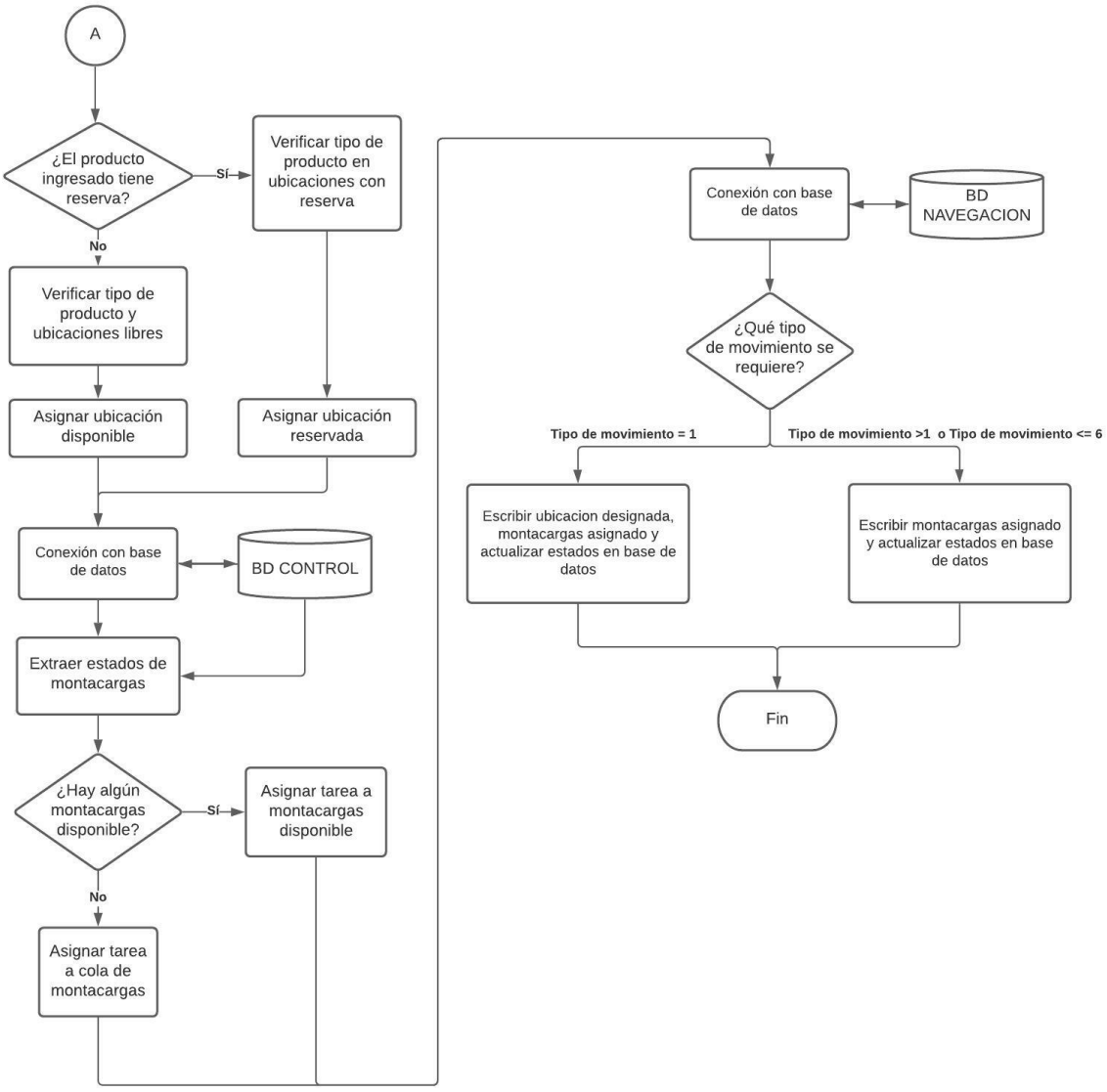


Figura 3.45: Continuación de diagrama de flujo de algoritmo de asignación de ubicaciones



el algoritmo trabaja a manera de un archivo ejecutable: Ingresos.exe. como primer paso el algoritmo importa las librerías necesarias para su funcionamiento, lee, extrae y almacena las configuraciones de los archivos .txt (Bloc de notas) para la conexión con el servidor y bases de datos y configuración de bodegas. A continuación el software accede a la base de datos NAVEGACIÓN y a la tabla dbo.INGRESO_UBICACIÓN, (Figura 3.46) la cual contiene la información de los productos que van a ser ingresados a bodega.

Figura 3.46: Base de datos de ingresos a bodega

Resultados | Mensajes

	cod_fic...	nombre_producto	ubicaci...	montacar...	esta...	t_producto
25	100	BROCCOLI CUTS - 30/70mm	NULL	NULL	1	Producto Terminado
26	101	IQF ORGANIC BROCCOLI FLORETS 12/50 - 12/50	NULL	NULL	1	Producto Terminado
27	102	WP BROCCOLI 4060 - 4060 TALLO LARGO	NULL	NULL	1	Producto Terminado
28	103	IQF CONVENTIONAL SPINACH - 0,12mm	NULL	NULL	1	Producto Terminado
29	104	IQF ORGANIC BROCCOLI FLORETS 10/25 - 10/25m	NULL	NULL	1	Producto Terminado
30	105	IQF ORGANIC SPINACH - 0,12mm	NULL	NULL	1	Producto Terminado
31	106	IQF ORGANIC SPINACH - 0,12mm	NULL	NULL	1	Producto Terminado
32	107	IQF BROCCOLI FLORETS 30/50 - 30/50mm	NULL	NULL	1	Producto Terminado
33	108	IQF ROMANESCO FLORETS 15/35 - 15/30mm	NULL	NULL	1	Producto Terminado
34	109	ORGANIC BLEND KALE&SPINACH - 50%50%	NULL	NULL	1	Producto Terminado

Tabla de ingresos extraída de la base de datos, se muestra nombre de productos, calibre, fecha de creación del ingreso, ubicación, montacargas asignado, estado, orden de producción y tipo de producto. Elaborado por: Alexander Guerrero.

El software captura información puntual de un ingreso a la vez. La información extraída mediante Python se muestra a continuación en la Figura 3.47.

Figura 3.47: Información extraída con Python para Ingresos a bodega

```
ficha:  1
nombre:  IQF BROCCOLI CONVENTIONAL IMPROVED - 30/50 mejorado
tipo:  Producto Terminado
ubicacion None
estado:  1
Disponible o cuarentena: Disponible
```

Extracción de datos de un ingreso por el software basado en Python. Elaborado por: Alexander Guerrero.

La información extraída se interpreta de la siguiente forma: se dice ficha a un número único correspondiente a un Bin, mismo que está preparado para ser almacenado en bodega. Se cuenta con el nombre del producto y con su calibre de corte, a continuación, se observa que el tipo de producto es “Producto terminado”, mismo que pertenece a un área en específico dentro de un espacio de bodega. La ubicación cuenta con un valor de None, ya que el software aún no ha designado una ubicación a dicho producto. El estado en valor 1 representa un ingreso a bodega. Cabe recalcar que este es el único estado que el software de ingresos interpreta para realizar su trabajo. Por último, se puede observar si el producto se encuentra disponible para ser llevado a un espacio de bodega o si este presentó algún inconveniente y debe ser llevado al área de cuarentena para su revisión.

Como paso siguiente, el software ingresa a la base de datos DATABODEGA y a la tabla dbo.UBICACION, de la que extrae todas las ubicaciones de bodega. Las ordena y procede a filtrar las ubicaciones disponibles dentro de las zonas reservadas. El software verifica el área a la que corresponde el producto, también comprueba si el nombre de dicho producto cuenta con una reservada.

Figura 3.48: Posiciones reservadas para un producto en base de datos

Resultados | Mensajes

	producto_calibre	ubicaci...	ocupaci...	tipo	area
1	IQF BROCCOLI CONVENTIONAL IMPROVED - 30/50 mejorado	4.1.1.2	30	Producto Terminado	are4
2	IQF BROCCOLI CONVENTIONAL IMPROVED - 30/50 mejorado	4.1.1.3	30	Producto Terminado	are4
3	IQF BROCCOLI CONVENTIONAL IMPROVED - 30/50 mejorado	4.1.1.4	30	Producto Terminado	are4
4	IQF BROCCOLI CONVENTIONAL IMPROVED - 30/50 mejorado	4.1.10.1	30	Producto Terminado	are4
5	IQF BROCCOLI CONVENTIONAL IMPROVED - 30/50 mejorado	4.1.10.2	30	Producto Terminado	are4
6	IQF BROCCOLI CONVENTIONAL IMPROVED - 30/50 mejorado	4.1.10.3	30	Producto Terminado	are4
7	IQF BROCCOLI CONVENTIONAL IMPROVED - 30/50 mejorado	4.1.10.4	30	Producto Terminado	are4

Captura de base de datos donde se muestran las ubicaciones que han sido reservadas con su nombre respectivo y el área al que pertenecen. Elaborado por: Alexander Guerrero.

Una vez verificado que el producto tenga una zona designada dentro del espacio de bodega, el software designa una a una ubicación dentro de la misma, siguiendo el orden de llenado establecido. Si el producto no tiene una zona reservada, el software le asigna una ubicación disponible dentro del área que pertenezca.

Figura 3.49: Asignación de ubicación reservada

```
Tipo de producto:  Producto Terminado
Reserva:  True
UBICACION FINAL:  4.1.1.2
```

Designación de una ubicación previamente reservada con el software inteligente a un ingreso a bodega. Elaborado por: Alexander Guerrero.

A continuación, el software actualiza la ubicación al valor de 50, mismo que corresponde a “ubicación asignada”.

Como paso siguiente, el software extrae la información de los montacargas que están disponibles en ese momento de la base de datos, y designa la tarea a uno de ellos. Si no existe ningún montacargas disponible, el software asigna las tareas a los montacargas a modo de cola. El proceso se realiza a forma de bucle hasta concluir con los ingresos a bodega.

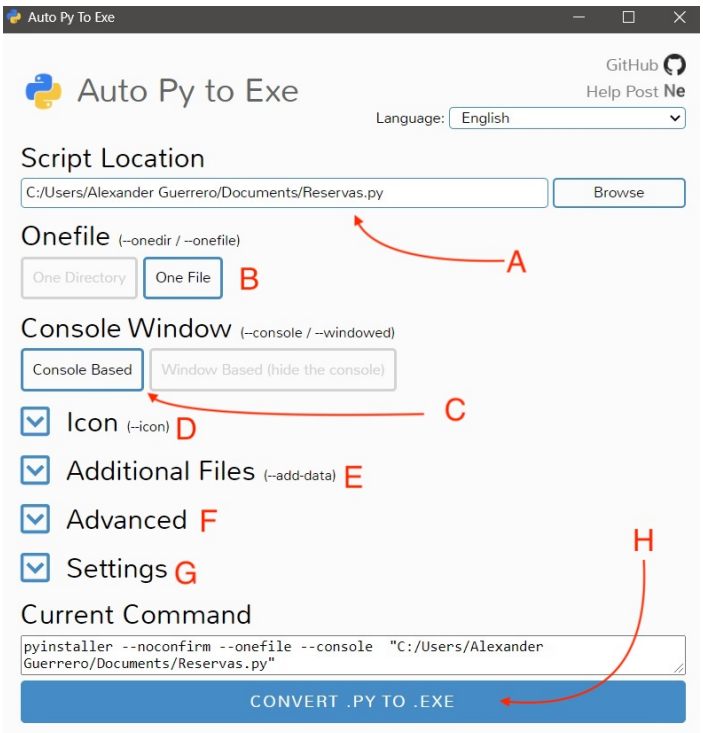
3.6 Implementación del software en servidor

La implementación del software en el servidor fue a manera de archivos ejecutables, dichos archivos fueron creados mediante programación en Python y posteriormente convertidos haciendo uso de la librería Auto-py-to-exe.

3.6.1 Configuración de Auto-py-to-exe

La librería Auto-py-to-exe, incluye gran variedad de propiedades configurables para crear un archivo ejecutable .exe; a continuación, se describe a manera general las opciones que presenta esta librería.

Figura 3.50: Configuración de archivo ejecutable con Auto-py-to-exe



Interfaz de usuario de la herramienta Auto-py-to-exe. Elaborado por: Alexander Guerrero.

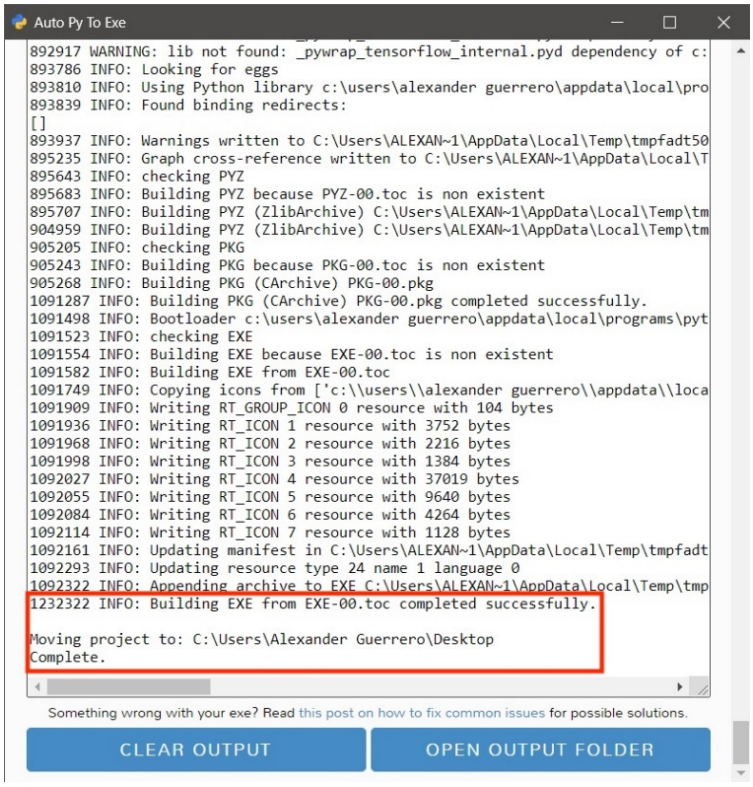
Como se puede apreciar en la Figura 3.50, Auto-py-to-exe cuenta con varios apartados de configuración, mismos que se listan a continuación:

- A, corresponde al directorio donde se ubica el archivo que se desea convertir de formato .py a formato .exe.
- B, presenta opciones de exportación del archivo, pudiendo ser: directorio (en una carpeta con varios archivos) o en un solo archivo ejecutable.

- C, hace referencia a si el software será ejecutado a mediante Consola o por una interfaz de usuario.
- D, corresponde a la opción de cambiar de icono al archivo ejecutable.
- E, permite añadir más de un archivo .py a la conversión.
- F, presenta configuraciones avanzadas de empaquetado, opciones para distintos sistemas operativos etc.
- G, presenta opciones de directorio de exportación, opciones de importación y exportación de archivos JSON. Finalmente, H es un botón el cual inicia la conversión de archivos.

Para el caso de estudio, los parámetros que fueron alterados fueron A, B y C, con la configuración que se muestra en la Figura 3.50

Figura 3.51: Conversión a archivo ejecutable culminada con Auto-py-to-exe



Construcción del archivo ejecutable completada. Elaborado por: Alexander Guerrero.

La Figura 3.51 muestra un mensaje de conversión satisfactoria y exportación al directorio de salida definido.

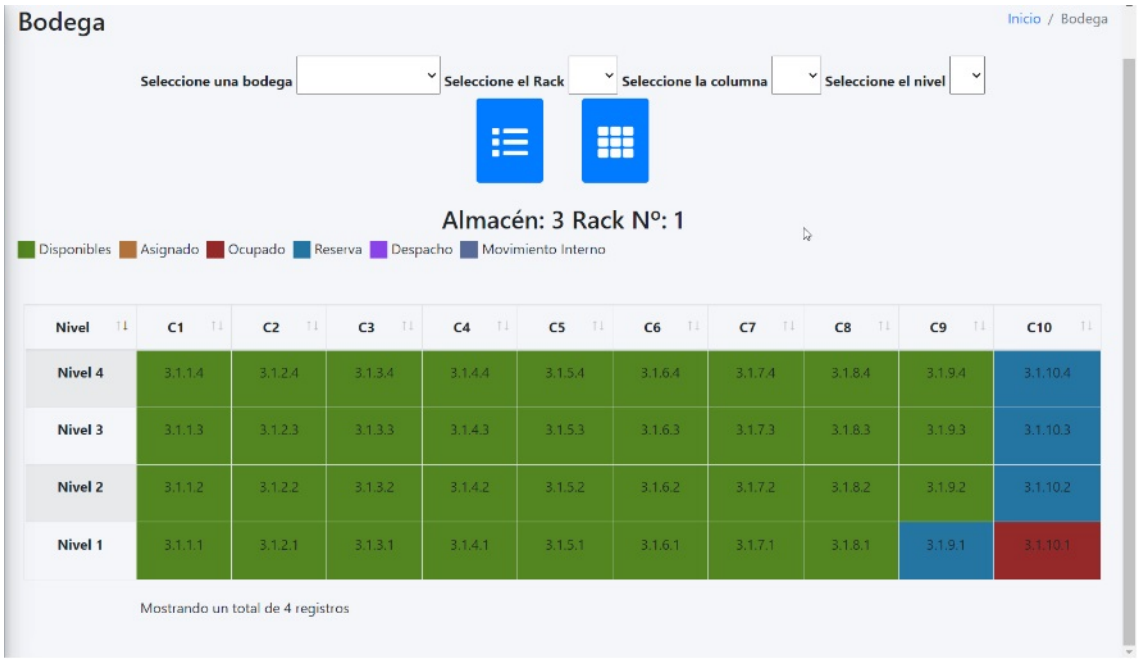
3.7 Integración del software con Interfaz de usuario

Para la visualización de usuario, el software hace uso de la interfaz gráfica Web Front-end diseñada por la empresa Robotrón.

La interfaz de usuario interpreta los valores escritos por el software en la base de datos, específicamente el estado de ubicación de cada espacio de bodega y los representa con casillas de colores.

A continuación, se muestra a manera de un ejemplo real, la vinculación del software con la interfaz de usuario. La Figura 3.52 muestra el estado inicial de una bodega.

Figura 3.52: Interfaz de usuario de bodega estado inicial

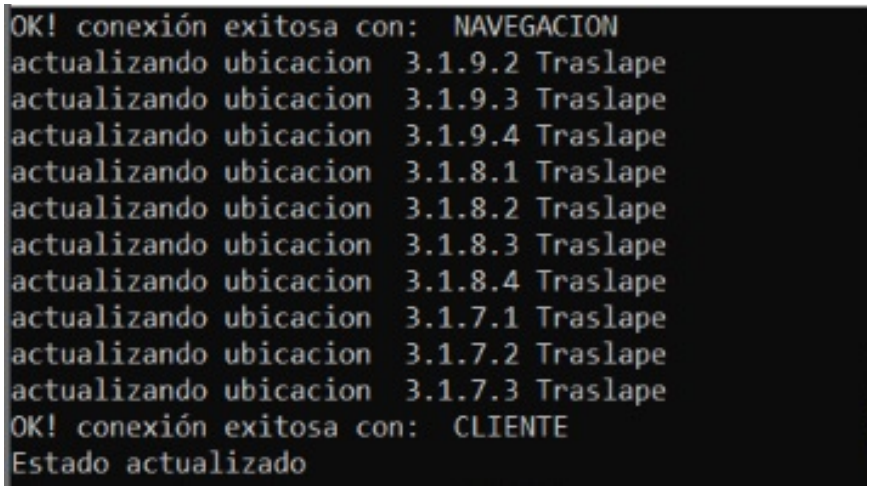


Interfaz de usuario conectada a base de datos mostrando las ocupaciones y reservas en bodega 3. Elaborado por: Robotrón.

La bodega de la Figura 3.52, cuenta con una única posición ocupada de color rojo, 4 posiciones reservadas de color azul y 35 posiciones disponibles de color verde.

A continuación, el software realiza una reserva de 10 posiciones como se muestra en la Figura 3.53.

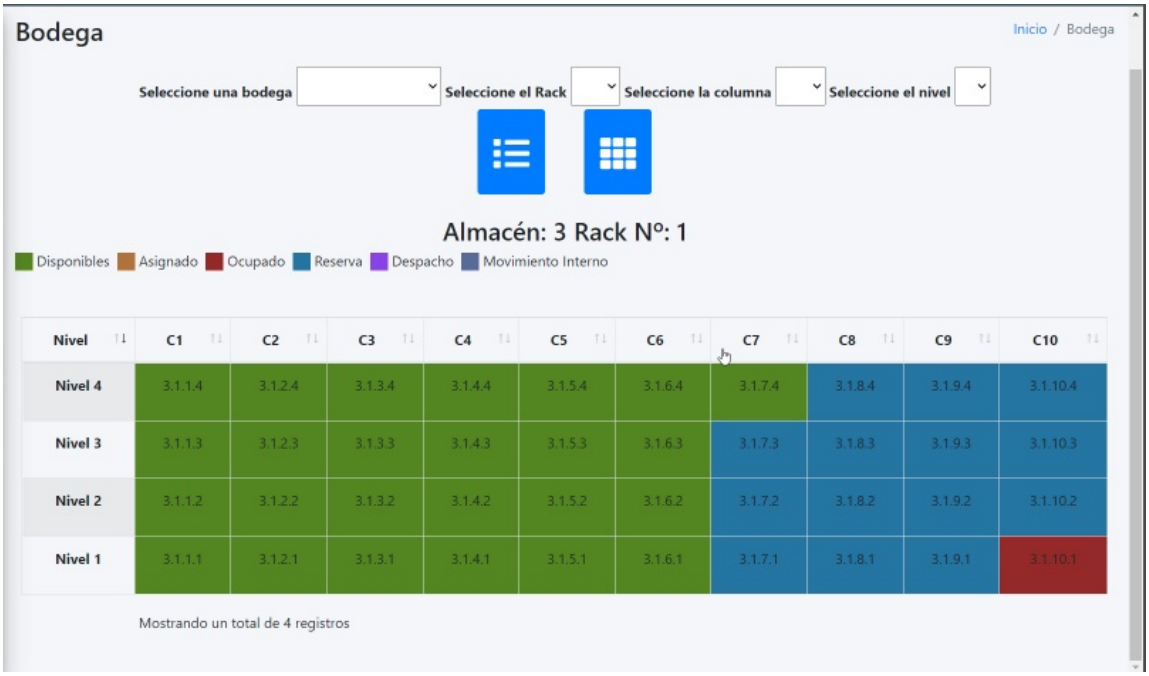
Figura 3.53: Software haciendo una reserva



Ubicaciones reservadas por software ejecutable en bodega 3. Elaborado por: Alexander Guerrero.

Al actualizar la interfaz web, se observa en la Figura 3.54 como el número de posiciones reservadas pasa de 4 a 14.

Figura 3.54: Interfaz de usuario posterior a reservas



Interfaz de usuario conectada a base de datos mostrando las ocupaciones y reservas actualizadas en bodega 3 Elaborado por: Robotrón.

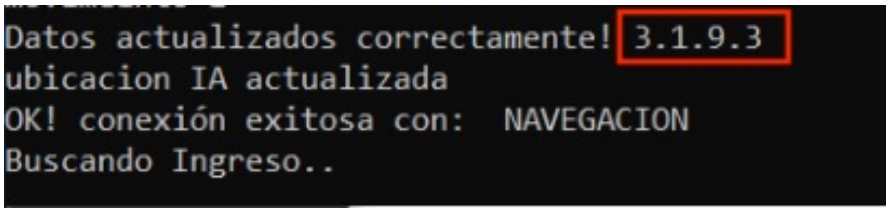
CAPÍTULO 4
PRUEBAS Y RESULTADOS

Con el objetivo de validar el funcionamiento del software, se realizaron pruebas en dos etapas: la primera, en un escenario controlado donde se evalúan distintos modelos del sistema de red neuronal para encontrar el más adecuado para el caso de estudio. La segunda etapa, consta de pruebas de funcionamiento dentro de la bodega, donde se compara el número de ubicaciones predichas y el número real de ubicaciones ocupadas. Además, se realizan comparaciones del tiempo que tarda un operario en ubicar un producto en una posición dentro de la bodega con y sin el uso del software.

4.1 Pruebas de asignación y visualización

A continuación, se ejemplifica la visualización mediante la interfaz de usuario de la asignación de un espacio dentro del área reservada, posterior a que varias posiciones hayan sido ocupadas por productos correspondientes al área.

Figura 4.55: Asignación de ubicación por software



Designación de una ubicación previamente reservada con el software inteligente a un ingreso a bodega. Elaborado por: Alexander Guerrero.

Al actualizar la interfaz de usuario, se observa como la ubicación 3.1.9.3 cambia de color verde a color naranja, color que representa el estado asignado en bodega.

Figura 4.56: Interfaz de usuario, visualización de asignación

Visualización de la ubicación 3.1.9.3 asignada a un montacargas (naranja), las ubicaciones de color verde representan espacios libres, las ubicaciones de color rojo (ocupadas) fueron previamente reservadas, asignadas y llenadas con cargas. Elaborado por: Robotrón.

4.2 Pruebas de diseño de red neuronal

Con el objetivo de encontrar el modelo de red neuronal más adecuado para el caso de estudio, se realizaron pruebas modificando el número de capas LSTM, número de unidades por capa en un período de 100 épocas de entrenamiento como se muestra en la Tabla 4.6.

Como método de error se utiliza el error cuadrático medio (MSE) y la raíz del error cuadrático medio (RMSE), mismos que miden la diferencia entre los valores reales y la predicción realizada. El MSE es más susceptible a cambios abruptos en la predicción, al contrario del RMSE, mismo que es más susceptible a cambios pequeños en la predicción. Para ambos casos, mientras el valor sea más cercano a 0, más cercana está la predicción de los valores reales.

Tabla 4.6: Pruebas para diseño de red neuronal

Número de capas LSTM	Número de neuronas capa 1	Número de neuronas capa 2	MSE	RMSE
1	32	-	22.67	5.34
1	64	-	20.97	4.16
1	128	-	14.54	3.49
2	32	32	10.40	2.95
2	64	32	18.86	3.98
2	64	64	13.29	3.34
2	128	64	14.51	3.48
2	64	128	12.52	3.23
2	128	128	15.96	3.64

Comparativa de comportamiento del software con diferente número de neuronas en la capa oculta. Elaborado por: Alexander Guerrero.

Como se aprecia en la Tabla 4.6, se obtuvo un mejor desempeño implementando 2 capas ocultas LSTM con 32 neuronas cada una, obteniendo un MSE de 10.40 y un RMSE de 3.49. De esta forma se valida que la red neuronal con las configuraciones seleccionadas, es la más acertada para los datos de entrada del caso de estudio.

4.3 Pruebas de implementación

A continuación, se analizó el comportamiento del software implementado en la bodega semiautomática, se realizó una única predicción realizada en agosto 2021 para 3 meses a futuro (septiembre 2021 – noviembre 2021), el análisis se basa en comparar el número de reservas realizadas por el software inteligente y compararlo con el número de ubicaciones ocupadas durante dicho período de tiempo.

Tabla 4.7: Resultados: Predicciones vs ubicaciones ocupadas primer mes

Producto	Predicción mensual [número de ubicaciones]	Ocupado [número de ubicaciones]	Error [%]
1	65	62	4,838
2	47	49	4,0816
3	74	75	1,3333
4	28	27	3,7037
5	71	70	1,4285

Predicción realizada en agosto vs ubicaciones ocupadas en septiembre. Elaborado por: Alexander Guerrero.

En la Tabla 4.7, se compara la predicción realizada en agosto con los resultados medidos en septiembre. Se determina que la predicción hecha por el software inteligente fue bastante acertada, teniendo un error porcentual máximo de 4.8% y un error mínimo de 1.33% para los productos analizados en el periodo de tiempo de un mes.

Tabla 4.8: Resultados: Predicciones vs ubicaciones ocupadas segundo mes

Producto	Predicción mensual [número de ubicaciones]	Ocupado [número de ubicaciones]	Error [%]
1	59	60	1,6666
2	47	43	9,3023
3	72	69	4,3478
4	27	26	3,8461
5	73	65	12,3076

Predicción realizada en agosto vs ubicaciones ocupadas en octubre. Elaborado por: Alexander Guerrero.

En la Tabla 4.8, se puede apreciar los resultados de la predicción realizada en agosto comparada con los datos medidos en octubre, el error ha aumentado a un máximo de 12.3% y el error mínimo es de 1.66%, en el intervalo de dos meses.

Tabla 4.9: Resultados Predicciones vs ubicaciones ocupadas tercer mes

Producto	Predicción mensual [número de ubicaciones]	Ocupado [número de ubicaciones]	Error [%]
1	47	64	26,5625
2	35	47	25,5319
3	58	74	21,6216
4	21	27	22,2222
5	53	72	26,3888

Predicción realizada en agosto vs ubicaciones ocupadas en noviembre. Elaborado por: Alexander Guerrero.

En la Tabla 4.9, se comparan los resultados de la predicción realizada en agosto con los datos reales del mes de noviembre, se puede apreciar como el error crece considerablemente en el intervalo de 3 meses, siendo el error máximo 26.56% y el mínimo 21.62%. Este error incrementa drásticamente ya que el software toma los últimos 12 meses para realizar la predicción, tomando en cuenta las predicciones anteriores, es decir en este caso el software realiza predicciones sobre sus propias predicciones.
Al analizar las tablas de resultados de ocupaciones, se determina que, con una única predicción realizada por el software inteligente, las predicciones obtenidas en el intervalo de uno a dos meses son bastante acertadas teniendo un error máximo de 12.3%.

Finalmente, se midió de forma experimental el tiempo que toma a un operario asignar una ubicación e ingresar un producto mediante un montacargas dentro de un espacio de bodega haciendo y no uso del software inteligente.

Tabla 4.10: Comparativa tiempos de asignación e ingreso

Número de medición	Tiempo sin software [min]	Tiempo con software [min]	Diferencia [min]
1	15	10	5
2	14	9	5
3	14	8	6
4	13	9	4
5	14	10	7
6	14	8	6
7	15	9	6
8	17	10	7
9	13	8	5
10	15	7	8
Promedio [min]	14,2	8,8	5,6

Comparación de tiempos haciendo uso del software y sin hacer uso del software. Elaborado por: Alexander Guerrero.

Como se puede apreciar en la Tabla 4.10, con la implementación del software inteligente se logró reducir el tiempo de asignación e ingreso de productos a bodega hasta aproximadamente 6 minutos, debido a que, sin la implementación del software, la asignación de ubicaciones se realiza de manera manual, sin tener un registro computacional del entorno de trabajo. Con la implementación del software la asignación de ubicaciones es instantánea y el tiempo únicamente depende del montacargas. Al analizar los resultados se puede decir que en una jornada laboral de 8 horas sin el software se logran hacer aproximadamente 34 ingresos a bodega, con el uso del software aproximadamente 55. Lo que resulta en una optimización de 21 ingresos por jornada laboral y aproximadamente de 3 ingresos por hora.

4.4 Pruebas de exportación con Auto-py-to-exe

La herramienta Auto-py-to-exe cuenta con dos opciones de exportación: directorio (carpeta con varios archivos) o un solo archivo ejecutable. Existen dos diferencias destacables las cuales son: tiempo de arranque y tamaño de archivo. Los archivos generados son generalmente grandes ya que, al momento de compilar y generar el archivo ejecutable, el software almacena las librerías completas que se hayan importado en el

script de Python. El método de exportación en directorio tuvo un tamaño de aproximadamente 1.19 Gb con un tiempo de ejecución de aproximadamente 30 segundos. El tamaño en un solo archivo ejecutable tuvo un tamaño de 480Mb con un tiempo de arranque aproximado de 50 segundos. Para el caso de estudio se implementó un solo archivo ejecutable por razones de seguridad, para evitar que se modifiquen o eliminen archivos ya que existe una alta rotación de personal con acceso al servidor.

CONCLUSIONES

Se logró diseñar e implementar el software basado en inteligencia artificial para la automatización de una bodega semiautomática, mismo que permite tener control sobre los espacios de bodega, realizar reservas mediante predicciones en base a datos históricos de cada producto y asignar ubicaciones únicas dentro de las mismas; el sistema se validó con pruebas tanto prácticas como pruebas en un entorno controlado. Obteniendo resultados satisfactorios de predicción en un lapso de dos meses, con un error máximo de 12.3% entre el valor predicho y el valor real.

Mediante la investigación del estado del estado del arte, se encontraron sistemas inteligentes implementados a lo largo de los años en diversas áreas del análisis de datos, ciencia y tecnología que hicieron uso de la red neuronal LSTM para obtener predicciones de datos a futuro. De esta forma se logró comprender el funcionamiento de la red neuronal para el diseño de un algoritmo predictivo del comportamiento de 30 productos y su posterior implementación en la bodega.

Haciendo uso del software FlexSim se modeló y simuló el comportamiento de los distintos espacios del almacén como espacios planos y racks. También se pudo planificar la ubicación de las zonas de prioridad ABCD dentro de los espacios de bodega y diseñar el orden de llenado para cada tipo de bodega respectivamente. La visualización y validación del software se realizó mediante el propio sistema de visualización web de Robotrón.

Para el diseño del algoritmo inteligente, se determinó mediante investigación que el lenguaje de programación con las características óptimas para el desarrollo del caso de estudio es Python. Mediante el cual, se pudo realizar la conexión con 1 servidor, lectura y escritura en 4 diferentes bases de datos, programar la red neuronal con 48360 datos por cliente, realizar algoritmos de ordenamiento, reservar áreas por prioridad ABCD y asignar 4600 ubicaciones disponibles en todos los espacios de bodega, a su vez la exportación de los archivos en formato ejecutable para su uso en el servidor.

Para mejorar la fase del entrenamiento de la red neuronal LSTM, se utilizó una capa de Dropout con un valor mínimo del 20% para evitar el sobre entrenamiento o overfiting de

la red neuronal, a su vez, al momento de trabajar con algoritmos que basen su funcionamiento en series temporales como lo es la red LSTM, es importe tener en cuenta las dimensiones de los vectores de entrada con los que se alimenta a la red neuronal para este caso, ejemplos estructurados en 12 columnas para los elementos de entrada, para las capas oculta ejemplos estructurados en 32 columnas; caso contrario el algoritmo puede presentar inconvenientes en la etapa de entrenamiento o predicción.

Mediante tablas de productividad obtenidas experimentalmente dentro de la bodega, en el periodo de 3 meses, se logró validar el funcionamiento del algoritmo inteligente, teniendo una efectividad mayor para los 2 primeros meses de predicción con un error máximo del 12.3%, en el tercer mes, el error aumenta hasta un 26.56%. Dicho error se debe a que el algoritmo toma una entrada de 12 meses en la serie temporal. Al aumentar el tiempo de predicción, el algoritmo descarta valores reales, el software realiza predicciones sobre sus propias predicciones disminuyendo su efectividad.

Al hacer uso de la librería Auto-py-to-exe, los archivos generados son generalmente grandes ya que, al momento de compilar y generar el archivo ejecutable, el software almacena las librerías completas que se hayan importado en el script de Python. Esto a su vez tiene una desventaja en el tiempo de arranque del programa. En el caso de estudio el archivo tuvo un tamaño aproximado de 480Mb y un tiempo de arranque de aproximadamente 50 segundos.

Mediante la implementación del software, se redujo el tiempo de asignación e ingreso de los productos en aproximadamente 6 minutos. Dicho tiempo se logró gracias a que la asignación de una ubicación dentro de un espacio de bodega es instantánea. Previo a la implementación del software la búsqueda se realizaba de manera manual.

Haciendo uso del software inteligente, se logró tener un control total sobre las ubicaciones de los productos dentro de los espacios de bodega. Mediante el sistema de reservas y asignación, apoyado con el sistema de visualización se puede localizar un producto en concreto o un tipo de producto en específico de manera instantánea dentro de los espacios de bodega, ya que dicha información es escrita en una base de datos.

RECOMENDACIONES

Al momento de trabajar con redes LSTM multicapa, se debe tener en cuenta definir el valor del parámetro “return_sequences” a True, ya que este es el que habilita la capacidad de las neuronas de las capas ocultas de comunicarse y transmitir datos a través de las capas adyacentes. A su vez, en la última capa LSTM este parámetro debe ser definido con el valor de False ya que la capa en cuestión será relacionada con otro tipo de capas para obtener una salida u predicción.

Para evitar conflictos de conectividad entre los diferentes clientes y el servidor se recomienda utilizar misma versión de driver de ODBC, para el caso de estudio, surgieron inconvenientes al trabajar con las versiones 11 y 17 del mismo.

Con motivos de optimización de tiempos de arranque de un archivo ejecutable creado a base de la herramienta Auto-py-to-exe, se recomienda exportar el archivo a manera de directorio, esta forma de exportación no comprime las librerías y recursos en un solo archivo, para el caso de estudio el tiempo de ejecución se redujo en aproximadamente un 40%.

Al momento de dividir un set de datos en parte de entrenamiento y validación se sugiere utilizar aproximadamente un 70% u 80% de los datos totales para entrenamiento y un 30% o 20% para validación, de esta forma se garantiza que el modelo no carezca de datos para la etapa de entrenamiento y que la validación de las predicciones se realice con datos suficientes.

REFERENCIAS

Rouhiainen, L. (2018). Introducción a la inteligencia Artificial. En S. Russel J., & P. Norvig, *Inteligencia Artificial 101 cosas que debes saber hoy sobre nuestro futuro* (págs. 16-19). Barcelona: Alienta.

Gerón, A. (2017). Machine learning. En A. Gerón, *Hands-on Machine Learning with Scikit-Learn and Tensorflow* (pág. 4). O'Reilly.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning. En I. Goodfellow, Y. Bengio, & A. Courville, *Deep Learning* (págs. 96-97). MIT Press.

Wu, G., Yao , L., & Yu, S. (9-11 de Junio de 2018). *IEEE Xplore.* Obtenido de Simulation and Optimization of Production Line Based on FlexSim: https://bibliotecas.ups.edu.ec:2095/stamp/stamp.jsp?tp=&arnumber=8407704

Dávila García, T. (2014). *Sistema WMS (Warehouse Management System) para una empresa PYME (Pequeña y Mediana Empresa) del Ecuador dedicada a la comercialización de productos.* Tesis de postgrado, Escuela superior politécnica del litoral, Guayaquil.

Celi de la Torre, P. (22 de Diciembre de 2017). *Ministerio de Telecomunicaciones y de la Sociedad de la Información.* Obtenido de https://www.telecomunicaciones.gob.ec/wp-content/uploads/2018/01/REGLAMENTO-DE-BIENES.pdf

Brozzi, R., Forti, D., Rauch, E., & T. Matt, D. (30 de Marzo de 2020). *MDPI.* Obtenido de The Advantages of Industry 4.0 Applications for Sustainability: Results from a Sample of Manufacturing Companies: https://www.mdpi.com/2071-1050/12/9/3647/htm

Esmena, M. (2021). *Estanterías sobre bases móviles Movirack.* Obtenido de https://www.mecalux.pe/estanterias-paletizacion/estanterias-paletizacion-movirack

Pina, E. E. (2021). *Aprendizaje por Refuerzo mediante Deep Learning para las Ciudades Inteligentes. Tesis Mag.* Madrid, España: Universidad Politécnica de Madrid.

Machado, M. C. (2018). *Predicción demanda eléctrica española. Implementación de redes neuronales recurrentes en Python. Tesis Mag.* Madrid: Universidad Complutense de Madrid.

Rojas, E. M. (2020). Machine Learning: análisis de lenguajes de programación y herramientas para desarrollo. *Iberian Journal of Information Systems and Technologies*, 586–599.

Team, R. C. (2021). *R Language Definition*. Obtenido de The Comprehensive R Archive Network: https://cran.r-project.org/doc/manuals/r-release/R-lang.html

MathWorks, T. (2021). *MathWorks*. Obtenido de Matlab: https://www.mathworks.com/products/matlab.html

Pedamkar, P. (s.f.). *Educba.* Obtenido de Matlab vs R: https://www.educba.com/matlab-vs-r/

blog, c. (2015). *colah`s blog.* Obtenido de Understanding LSTM Networks: http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Olah, C. (2015). *Stanford University.* Obtenido de Stanford University: https://web.stanford.edu/class/cs379c/archive/2018/class_messages_listing/content/Artificial_Neural_Network_Technology_Tutorials/OlahLSTM-NEURAL-NETWORK-TUTORIAL-15.pdf

Greff, K., Rupesh K., S., Jan , K., Bas R. , S., & Jurgen , S. (2017). LSTM: A Search Space Odyssey. *TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS*.

Altché, F., & de La Fortelle, A. (2017). An LSTM Network for Highway Trajectory Prediction. *IEEE 20th International Conference on Intelligent Transportation Systems (ITSC): Workshop*.

Stollenga, M., Byeon, W., Liwicki, M., & Schmidhuber, J. (2015). Parallel Multi-Dimensional LSTM, With Application to Fast Biomedical Volumetric Image Segmentation. *Neural Information Processing Systems (NIPS)*.

Graves , A., & Schmidhuber, J. (2005). Framewise Phoneme Classification with Bidirectional LSTM Networks. *The annual International Joint Conference on Neural Networks (IJCNN)*.

Reddy Chimmula, V., & Lei, Z. (2020). Time series forecasting of COVID-19 transmission in Canada using LSTM networks. *Chaos, Solitons and Fractals*.

ANEXOS

Anexo 1: Importación de librerías

```
#Importar librerias
from tensorflow.keras.models import load_model
import numpy as np
import pandas as pd
import pyodbc
from sklearn.preprocessing import MinMaxScaler
from tensorflow.keras.models import load_model
from sklearn.preprocessing import MinMaxScaler
import time
```

El segmento de código importa las librerías utilizadas en el caso de estudio. Elaborado por: Alexander Guerrero.

Anexo 2: Lectura y limpieza de datos de archivos .txt

```
#Lectura de datos por filas de texto
f1 = open(r"C:\IA\Conexion.txt")

for x, line in enumerate(f1):

    if x == 0:
        ip_txt = line
    elif x == 1:
        nombre_db_NAVEGACION_txt = line
    elif x == 2:
        nombre_db_DATABODEGA_txt = line
    elif x == 3:
        nombre_db_CONTROL_txt = line
    elif x == 4:
        nombre_db_CLIENTE_txt = line
    elif x == 5:
        nombre_usuario_txt = line
    elif x == 6:
        password_db_txt = line

f1.close()

IP = ip_txt.replace('\n', '')
NAVEGACION = nombre_db_NAVEGACION_txt.replace('\n', '')
DATABODEGA = nombre_db_DATABODEGA_txt.replace('\n', '')
CONTROL = nombre_db_CONTROL_txt.replace('\n', '')
CLIENTE = nombre_db_CLIENTE_txt.replace('\n', '')
USUARIO = nombre_usuario_txt.replace('\n', '')
PASSWORD = password_db_txt.replace('\n', '')
```

El segmento de código lee un archivo llamado Conexión.txt del cual extrae su información, a continuación, elimina los saltos de línea denotados como '\n'. Elaborado por: Alexander Guerrero.

Anexo 3: Conexión con servidor y base de datos

```
#Conexión con servidor y base de datos
try:
    conexion = pyodbc.connect('DRIVER={ODBC Driver 17 for SQL Server};SERVER=' +
                              IP + ';DATABASE=' + CLIENTE + ';UID=' + USUARIO + ';PWD=' + PASSWORD)
    print("OK! conexión exitosa con:", CLIENTE)
except Exception as e:
    print("Ocurrió un error al conectar a SQL Server: ", e)
```

El segmento de código conecta el programa de Python con un servidor y una base de datos mediante dirección IP, nombre de base de datos, usuario y contraseña. Elaborado por: Alexander Guerrero.

Anexo 4: Consulta a base de datos

```
#consulta a base de datos / obtener datos historicos para hacer predicción
df = pd.read_sql("SELECT * FROM datos_train where estado_reserva is NULL order by producto, cod_cliente asc", con=conexion)
num_cod_cliente = int(df.iloc[0]['cod_cliente'])
num_producto = int(df.iloc[0]['producto'])
```

El segmento de código realiza una consulta a la base de datos en la cual extrae todos los datos de la tabla dbo.datos_train, posteriormente extrae el primer dato de las columnas ‘cod_cliente’ y ‘producto’. Elaborado por: Alexander Guerrero.

Anexo 5: Ordenamiento por producto y cliente, obtención de tipo de producto

```
#ordenar datos por producto
producto = df.iloc[0]['nombre_producto']
grouped = df.groupby('nombre_producto')
df = grouped.get_group(producto)

#ordenar datos por cliente
cod_cliente = df.iloc[0]['cod_cliente']
grouped = df.groupby('cod_cliente')
df = grouped.get_group(cod_cliente)

#Obtener tipo para clasificacion en bodega
tipo = df.iloc[0]['tipo_bodega']
```

El segmento de código lee un archivo llamado Conexión.txt del cual extrae su información, a continuación, elimina los saltos de línea denotados como “\n”. Elaborado por: Alexander Guerrero.

Anexo 6: Escalamiento de datos de entrada de red neuronal y carga de modelo previamente entrenado

```
#Escalar los datos en el rango de 0 a 1
scaler = MinMaxScaler()
scaled_train = scaler.fit_transform(train)

#cargar modelo previamente entrenado
features = 1
tamaño = 12
tiempo_prediccion = 7

model = load_model("C:\IA\Prediccion_LSTM.h5")
```

El segmento de código escala los datos en el rango de 0 a 1, a continuación, carga el modelo previamente entrenado y guardado como "Prediccion_LSTM.h5". Elaborado por: Alexander Guerrero.

Anexo 7: Dar formato a datos de entrada de la red neuronal, ciclo repetitivo para realizar predicciones y regreso a escala normal de los datos

```
#reescalar datos para realizar prediccion
test_predictions = []
first_eval_batch = scaled_train[-tamaño:]
current_batch = first_eval_batch.reshape((1, tamaño, features))

#ciclo repetitivo, hacer predicciones con modelo cargado
for i in range(tiempo_prediccion):
    current_pred = model.predict(current_batch)[0]
    test_predictions.append(current_pred)
    current_batch = np.append(current_batch[:,1:,:],[[current_pred]],axis=1)

#regresar a escala normal
true_predictions = scaler.inverse_transform(test_predictions)
true_predictions = true_predictions.astype(int)
```

El segmento de código da el formato a los datos de entrada en el formato (1x12x1) para la entrada de la red neuronal, a continuación se realizan las predicciones haciendo uso del modelo y se regresa los datos a la escala inicial con el comando "scaler.inverse_transfrom". Elaborado por: Alexander Guerrero.

Anexo 8: Segmentar datos por producto

```
broccoli_df = plan_df.loc[plan_df['producto_calibre'].str.contains("BROCCOLI", case=False)]
broccoli_total = broccoli_df['kilos'].sum()
broccoli_total_list = broccoli_df.values.tolist()
```

El segmento de código divide el historial de productos ABCD en listas. En el Anexo correspondiente, se ejemplifica con el uso del producto brócoli, sin embargo, existen otros tipos de producto como coliflor, zanahoria, etc. Elaborado por: Alexander Guerrero.

Anexo 9: Obtener número de productos que pertenecen a cada tipo de prioridad ABCD

```
posicionesA = round((porcentajeA * len(broccoli_total_list)) / 100)
posicionesB = round((porcentajeB * len(broccoli_total_list)) / 100)
posicionesC = round((porcentajeC * len(broccoli_total_list)) / 100)
posicionesD = round((porcentajeD * len(broccoli_total_list)) / 100)
```

El segmento de código hace uso de las listas previamente creadas en el Anexo 8, para dividirlas en prioridades ABCD, la división se realiza con los datos extraídos del archivo editable ConfiguracionBodegas.txt. Elaborado por: Alexander Guerrero.

Anexo 10: Formato de nombres de productos para compararlos con tablas de prioridad ABCD

```
#LIMPIAR NOMBRE DE INGRESO PARA COMPARAR CON TABLAS DE PRODUCTOS A,B,C,D
#BROCOLI
if 'BROCCOLI' in producto and '10/25' in producto:
    nombre_prod = 'BROCCOLI1025'
if 'BROCCOLI' in producto and '10/30' in producto:
    nombre_prod = 'BROCCOLI1030'
if 'BROCCOLI' in producto and '12/50' in producto and 'TL' in producto:
    nombre_prod = 'BROCCOLI1250TL'
if 'BROCCOLI' in producto and '15/30' in producto:
    nombre_prod = 'BROCCOLI1530'
```

El segmento de código da formato al nombre del producto de entrada con el formato de los nombres pertenecientes a las listas de prioridades ABCD. En el Anexo correspondiente, se ejemplifica con el uso del producto brócoli, sin embargo, existen otros tipos de producto como coliflor, zanahoria, etc. Elaborado por: Alexander Guerrero.

Anexo 11: Verificación de producto en tablas de prioridad ABCD

```
#VERIFICAR EN TABLAS DE PRIORIDAD ABCD
#BROCCOLI

for i in range(0,len(broccoli_A_list)):
    if nombre_prod == broccoli_A_list[i][2]:
        tipo_historico = 'A'
        productosA_B = True
for i in range(0,len(broccoli_B_list)):
    if nombre_prod == broccoli_B_list[i][2]:
        tipo_historico = 'B'
        productosA_B = True
```

El segmento de código verifica y encuentra coincidencias del nombre del producto de entrada en las diferentes listas de prioridades ABCD dependiendo del producto. Elaborado por: Alexander Guerrero.

Anexo 12: Ordenamiento inicial áreas y ubicaciones de menor a mayor

```
#ordenamiento inicial de areas de mayor a menor
for i in range(0, len(a_saldos)):
    area1.append(a_saldos[i][0])
area1 = sorted(area1, key=lambda v: [int(i) for i in v.split('.')])
```

El segmento de código ordena de menor a mayor las ubicaciones extraídas de una base de datos. El comando Split se utiliza debido al formato de las ubicaciones (Figura 3.23) y funciona como un separador. Elaborado por: Alexander Guerrero.

Anexo 13: Ordenamiento de areas y ubicaciones, bodega tipo espacio plano

```
#ordenamiento final de areas, dependiendo del tipo de bodega
#Area1
a1_otros = []
a1_b6 = []
a1_test = []
for i in range(0,len(area1)):
    if area1[i][0] != '6':
        a1_otros.append(area1[i])

for i in range(0,len(area1)):
    if area1[i][0] == '6':
        a1_b6.append(area1[i])

for i in range(0,len(a1_otros),40):
    a1_test.append(a1_otros[i:i + 40])

aux1=[]
area1_orden=[]
for i in range(0,len(a1_test)):
    for j in range(len(a1_test[i]),0,-1):
        aux1.append(a1_test[i][j-1])
cont=3
cont2=0
for i in range(0,len(aux1)):
    area1_orden.append(aux1[cont])
    cont = cont-1
    cont2 =cont2+1
    if cont2%4 == 0:
        cont=i+4
        if cont == 0:
            cont=3

area1_orden = list(dict.fromkeys(area1_orden))
```

El segmento de código ordena las ubicaciones con los requerimientos para bodegas de tipo espacio plano como se indica en la sección 3.4.4.3. Elaborado por: Alexander Guerrero.

Anexo 14: Quitar posiciones ocupadas y reservadas

```
for i in range(0,len(posiciones_ocupadas)):
    if posiciones_ocupadas[i] in area1_orden:
        area1_orden.remove(posiciones_ocupadas[i])
for i in range(0,len(res_area1)):
        if res_area1[i][0] in area1_orden:
            area1_orden.remove(res_area1[i][0])
```

El segmento de código elimina las ubicaciones reservadas y ocupadas de las áreas ordenadas. Elaborado por: Alexander Guerrero.

Anexo 15: Ordenamiento de áreas y ubicaciones, bodega tipo rack

```
area1_orden_camara6=[]
racks=[]
for i in range(0,len(a1_b6)):
    racks.append(a1_b6[i])
racks= sorted(racks, key=lambda x: x[-1])
area1_orden_camara6 = list(dict.fromkeys(racks))

for i in range(0,len(res_area1)):
    if res_area1[i][0] in area1_orden_camara6:
        area1_orden_camara6.remove(res_area1[i][0])
```

El segmento de código ordena las ubicaciones con los requerimientos para bodegas de tipo rack como se indica en la sección 3.4.4.4. Elaborado por: Alexander Guerrero.

Anexo 16: Porcentaje de ubicaciones dentro de un espacio de bodega que pertenecen a las distintas prioridades ABCD

```
#AREA 1 ABCD
a1_tipoA_num_pos = int(len(area1_orden) * pA_are/100)
a1_tipoB_num_pos = int(len(area1_orden) * pB_are/100)
a1_tipoC_num_pos = int(len(area1_orden) * pC_are/100)
a1_tipoD_num_pos = int(len(area1_orden) * pD_are/100)

a1_tipoA = []
a1_tipoB = []
a1_tipoC = []
a1_tipoD = []

lim_A = a1_tipoA_num_pos
lim_B = (a1_tipoA_num_pos + a1_tipoB_num_pos)
lim_C = (a1_tipoA_num_pos + a1_tipoB_num_pos + a1_tipoC_num_pos)
lim_D = (a1_tipoA_num_pos + a1_tipoB_num_pos + a1_tipoC_num_pos + a1_tipoD_num_pos)
```

El segmento de código divide una bodega de tipo espacio plano en áreas de prioridad ABCD dentro de la misma a manera porcentual controlada por los datos extraídos del archivo editable ConfiguracionBodegas.txt. Elaborado por: Alexander Guerrero.

Anexo 17: Crear lista con ubicaciones pertenecientes a prioridades ABCD

```
for i in range(0,lim_A):
    a1_tipoA.append(area1_orden[i])

for i in range(lim_A , lim_B):
    a1_tipoB.append(area1_orden[i])

for i in range(lim_B , lim_C):
    a1_tipoC.append(area1_orden[i])

for i in range(lim_C , lim_D):
    a1_tipoD.append(area1_orden[i])
```

El segmento de código guarda las ubicaciones correspondientes de las áreas de prioridad ABCD dentro de una bodega de tipo espacio plano en listas individuales. Elaborado por: Alexander Guerrero.

Anexo 18: Verificación de tipo de producto, prioridad ABCD y creación de listas con ubicaciones para ser reservadas

```
if tipo == 'Listo para despacho':
    m_ubic_res=[]
    cnt = 0
    num_bines = prediccion_to_db[1]
    if num_bines == 0:
        num_bines = 1
    if len(area7_orden) != 0:

        if tipo_historico == 'A':
            for ubi in a7_tipoA:
                m_ubic_res.append(a7_tipoA[cnt])
                cnt = cnt + 1
                if num_bines == cnt:
                    break
        elif tipo_historico == 'B':
            for ubi in a7_tipoB:
                m_ubic_res.append(a7_tipoB[cnt])
                cnt = cnt + 1
                if num_bines == cnt:
                    break
```

El segmento de código verifica a que tipo de producto pertenece el producto de entrada (Figura 3.24), a que tipo de prioridad ABCD pertenece y en base a la predicción realizada con el modelo inteligente, guarda el número de ubicaciones correspondientes en una lista llamada “m_ubic_res”. Elaborado por: Alexander Guerrero.

Anexo 19: Ciclo repetitivo para escribir en base de datos ejecutando procedimientos almacenados dentro del servidor

```
for i in range(0, len(m_ubic_res)):
    try:
        with conexion.cursor() as cursor:
            update = " exec update_Ubicacion_IA_Reservas_pro  ?, ?, ? "
            cursor.execute(update, (m_ubic_res[i]), producto, tipo)
            update = " exec update_Ubicacion_IA_30  ? "
            cursor.execute(update, (m_ubic_res[i]))
            update = " exec update_Ubicacion_30  ? "
            cursor.execute(update, (m_ubic_res[i]))
            print('actualizando ubicacion ', m_ubic_res[i] , producto, tipo)

    except Exception as e:
        print("Ocurrió un error al insertar: ", e)
```

El segmento de código ejecuta procedimientos almacenados en el servidor y a manera de ciclo repetitivo envía datos para la escritura y actualización de bases de datos. Elaborado por: Alexander Guerrero.

Anexo 20: Verificación de productos sin reserva

```
if reservado == False:
    print('SIN RESERVA')

    for i in range(0,len(reservas)):
        if reservas[i] in area1_orden:
            area1_orden.remove(reservas[i])

        if reservas[i] in area2_orden:
            area2_orden.remove(reservas[i])

        if reservas[i] in area3_orden:
            area3_orden.remove(reservas[i])
```

El segmento de código verifica si el producto dispone de una reserva previa guardada en el servidor, si no dispone de una, elimina todas las posiciones reservadas. Elaborado por: Alexander Guerrero.

Anexo 21: Verificación de tipo de producto y asignación de una ubicación única dentro de un espacio de bodega sin reserva

```
if tipo == 'Producto Testigo':
    if len(area3_orden) != 0:
        m_ubic = area3_orden[0]
        print(m_ubic)
    elif len(area3_orden) == 0:
        m_ubic = area3_orden_camara6[0]
        print(m_ubic)
```

El segmento de código se ejecuta cuando el producto no tiene reserva. Verifica el tipo de producto (Figura 3.24) y asigna una ubicación disponible no reservada. Elaborado por: Alexander Guerrero.

Anexo 22: Verificación de productos con reserva y asignación de una ubicación única dentro de un espacio de bodega con reserva previa

```
if reservado == True:
    print('PRODUCTO CON RESERVA')
    if tipo == 'Producto Terminado' and tipo in tipos_res:
        m_ubic = reservas[0]
    if tipo == 'Saldo Contenedor' and tipo in tipos_res:
        m_ubic = reservas[0]
    if tipo == 'Traslape' and tipo in tipos_res:
        m_ubic = reservas[0]
```

El segmento de código se ejecuta cuando el producto tiene una reserva previa guardada en la base de datos. Verifica el tipo de producto (Figura 3.24) y asigna una ubicación disponible dentro de las ubicaciones reservadas para dicho producto. Elaborado por: Alexander Guerrero.

Anexo 23: Escritura en base de datos con ayuda de procedimientos almacenados dependiendo el tipo de movimiento

```
if tipo_mov == '1':
    try:
        with conexion.cursor() as cursor:
            update = " exec update_ubicacion_montac ?,?,?,?,? "
            cursor.execute(update, (ficha, m_ubic, montac_disp[0], 2, bajar_carga))
            print("Datos actualizados correctamente!", m_ubic)

    except Exception as e:
        print("Ocurrió un error al insertar: ", e)
```

El segmento de código verifica el tipo de movimiento 1 que corresponde a “ingreso a bodega” y actualiza los parámetros en la base de datos haciendo uso de procedimientos almacenados. Elaborado por: Alexander Guerrero.

Anexo 24: Fotografía de montacargas

Anexo 25: Fotografía de bodega llena

Anexo 26: Fotografía de oficina de control