Universidad de Castilla-La Mancha

Escuela Superior de Ingeniería Informática

Trabajo Fin de Grado

Grado en Ingeniería Informática

Tecnología Específica de Ingeniería del Software

Algoritmos de optimización en tiendas para pedidos online

Eduardo Pastor de la Cruz

Febrero, 2021

Trabajo Fin de Grado

Grado en Ingeniería Informática

Tecnología Específica de Ingeniería del Software

Algoritmos de optimización en tiendas para pedidos online

Autor: Eduardo Pastor de la Cruz

Directores: Francisco Parreño Torres

Víctor Manuel López Jaquero

Febrero, 2021

Dedicado a mi familia

Declaración de Autoría

Yo, Eduardo Pastor de la Cruz con DNI 04614951R declaro que soy el único autor del trabajo fin de grado titulado "Algoritmos de optimización en tiendas para pedidos online" y que el citado trabajo no infringe las leyes en vigor sobre propiedad intelectual y que todo el material no original contenido en dicho trabajo está apropiadamente atribuido a sus legítimos autores.

Albacete, a 10 de Febrero de 2021

Fdo.: Eduardo Pastor de la Cruz

Resumen

Los hábitos de compra están siempre en constante cambio. Las empresas invierten mucho dinero y esfuerzo en adaptarse a estos cambios. El cambio más notable que estamos viviendo en la última década es el comercio a través de internet.

Este cambio se ha visto especialmente acelerado en la reciente pandemia. Sectores como el de la alimentación, en el que la transición a la venta online no se había conseguido, han visto cómo sus ventas a través de sus plataformas digitales se han incrementado exponencialmente, provocando el bloqueo incluso de grandes distribuidores.

Ante los nuevos desafíos que provoca la venta online, se requieren nuevas aproximaciones y procedimientos para que el coste añadido que lleva la venta online no provoque mermas en las métricas de rendimiento y productividad.

Por ello, este trabajo propondrá soluciones enfocadas a mejorar el rendimiento en las tareas de recolección de productos y la gestión de pedidos online en almacenes de tiendas de alimentación que dispongan de plataformas de comercio electrónico, enfocándonos en la reducción del coste hora-persona a través de una optimización en el orden de recogida y en las rutas a seguir.

Agradecimientos

A Mercedes Sahuquillo Oliver, sin ella nunca lo habría terminado (ni empezado).

A mis tutores, por los consejos que me han dado tanto para el proyecto como para la vida. Unos cracks. No obstante quiero resaltar la especial ayuda de Francisco Parreño Torres a la hora de reengancharme a la vida académica.

A todos los profesores que tuve durante la carrera. Gracias a ellos he conseguido llegar a dónde estoy, que no está mal. En especial a Bernardino del Campo López. Solamente pude disfrutar de él durante una asignatura pero realmente esos dos cuatrimestres cambiaron mi futuro. Me animaron a continuar con la carrera, a pesar de todas las dudas e inseguridades que tenía. El mejor profesor que he tenido nunca, y de las mejores personas que he conocido en mi vida.

Por último agradecérselo a mi familia, que me educaron para ser una persona con inquietudes y que confiaron en mí hasta cuando no tenían motivos para ello.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

1.1 Introducción

Los sistemas de gestión de pedidos de compras a través de internet están en un proceso constante de redimensionamiento debido al incremento en la demanda de los mismos. Escalar los mismos son cuestiones técnicas, ampliamente estudiadas, ya que en la actualidad hay infinidad de sistemas de gran demanda. No obstante, el gestionar un pedido compuesto múltiples referencias, cada una con unos volúmenes y pesos diferentes, y sobre todo, con unas necesidades de almacenaje y manipulación muy variables, y en algunos casos críticos, conlleva aplicar procedimientos que no son fácilmente escalables.

La necesidad de optimizar y redimensionar estos procesos para afrontar picos de demanda ha sido recurrente, desde la aparición de fenómenos como el "Black Friday". No obstante, este año debido a la situación excepcional provocada por el COVID-19, se ha producido un incremento de la demanda inusual, como podemos observar en la **Ilustración 1**, lo que ha llevado a que los comercios peor preparados hayan sufrido problemas de desabastecimiento o de imposibilidad de cumplir con todas las ventas (Klaviyo, 2020).

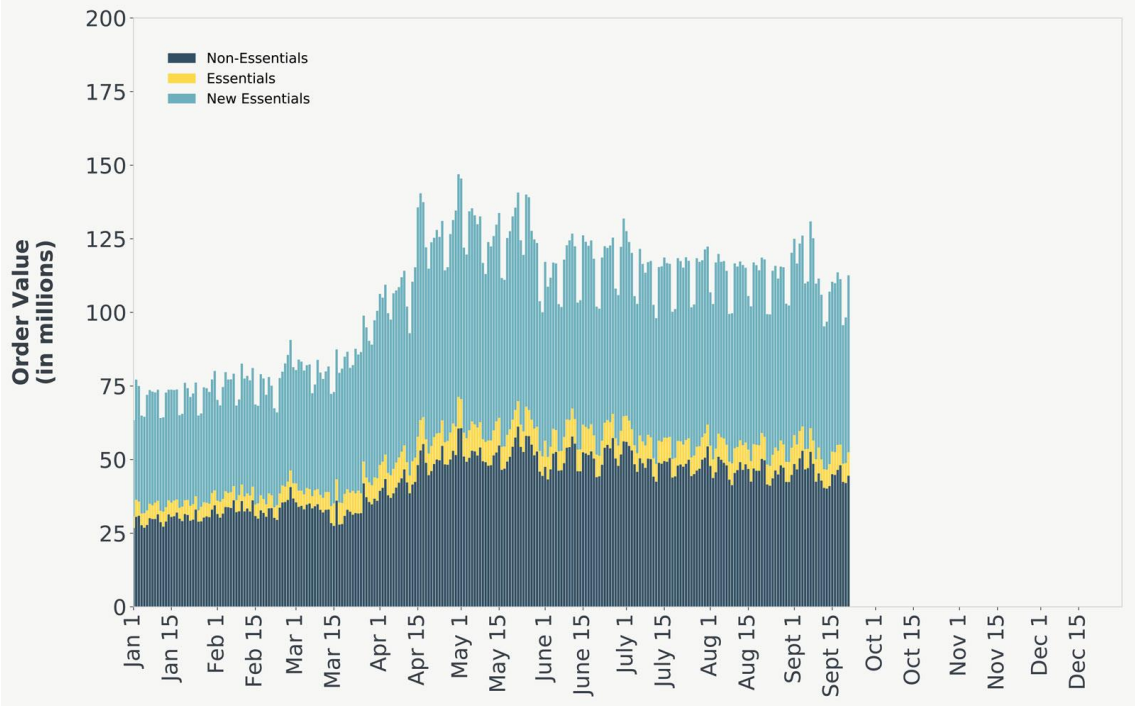


Ilustración 1: Evolución ventas online en el mundo

El trabajo de preparación de pedidos se define no sólo por la naturaleza de los productos a recoger, sino que una parte fundamental es la propia organización del almacén, así como el concepto de tienda. Algunos centros de distribución actuales no son más que las tiendas clásicas, en las que se hacen convivir los clientes tradicionales con el personal de la compañía que se dedica a realizar los pedidos online. Este tipo de tiendas las denominaremos como **tiendas abiertas** y la principal característica de ellas es que aparte de convivir personal de la tienda con los clientes, los productos se organizan de forma que sea más sencilla para la compra que realizan los clientes, que no tiene por qué ser óptimo para una recogida eficaz.

En contraposición a este tipo de tiendas, existen las **tiendas cerradas**, definidas por ser tiendas en las que sólo el personal de la empresa tiene acceso, y por lo tanto se puede enfocar tanto la organización como el stock para optimizar la recogida.

Ambos tipos de tiendas tienen en común que hay que controlar el **stock,** aunque como veremos, el stock es considerado una variable que está ampliamente controlada por los procesos internos de las organizaciones, y en muchos casos se puede considerar este stock, en algunos productos, como infinito, sobre todo en las **tiendas mixtas**, que como

su nombre indica, es un tipo de tienda que combina una zona de tienda abierta con una de tienda cerrada. Un ejemplo lo podemos ver en la **Ilustración 2**.

Este tipo de tiendas ofrece muchas oportunidades nuevas, pero también nuevas complejidades, como son el duplicar el control de stock y la opción de **reaprovisionar** las tiendas cerradas con productos existentes en la zona abierta de la tienda.

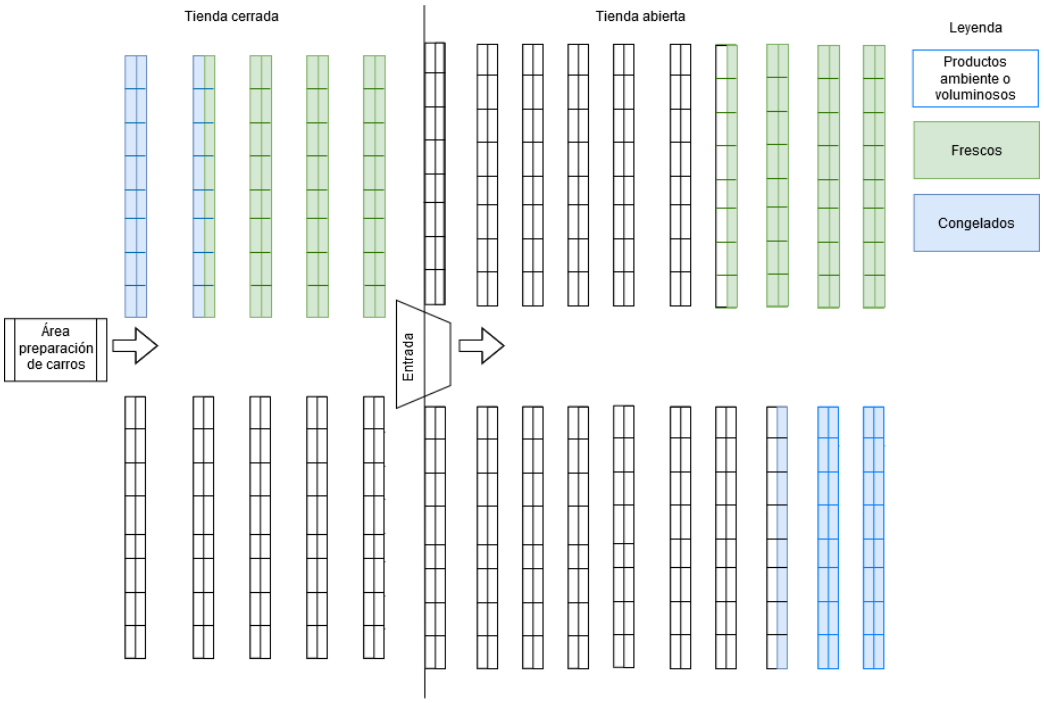


Ilustración 2: Ejemplo de un almacén con tienda abierta y cerrada

El caso que nos ocupa presenta ciertas peculiaridades adicionales que comprometen el control de la recogida, más allá de la inherente a la variedad de productos. La recogida de elementos en tienda abierta conlleva que los trabajadores encargados de la recogida de los productos (a los que denominaremos **pickers** a partir de ahora, ya que este anglicismo es masivamente usado) tengan que interactuar en muchos casos con los clientes habituales de la tienda donde se generan los pedidos para ser enviados.

Pese a que la automatización de los sistemas de recogida de producto en los almacenes de distribución son cada días más fiables y baratos, en las tiendas de concepto de recogida abierta o mixta la tecnología no coexiste armónicamente con los clientes, debido a que aún la convivencia entre máquinas y clientes no está lo suficientemente

avanzada, y pueden generarse situaciones peligrosas para los propios clientes, así como dificultades operativas en las máquinas de recogida.

Esta convivencia hace que el control del tiempo pueda sufrir retrasos (una medida básica de cortesía es dar prioridad a los clientes presenciales frente a los online) y dificulte el control de existencias. No obstante, para el caso que nos atañe definiremos las características generales de la siguiente manera:

- Cuando se va a preparar una ronda de pedidos (puede hacerse una vez al día o varias, dependiendo de las capacidades de la tienda), se preparan todos los carros que van a ser necesarios. Cada carro tendrá de 1 a 12 cajas, diferenciadas por colores. Estas cajas tienen características especiales según los productos que van a recoger, como vemos en la **Tabla 1**.

Color	Productos	Características	Medidas Recomendadas
Gris	Ambiente	Los productos no requieren de un tratamiento especial	60x40x40 cm Largo x ancho x alto
Verde	Refrigerados Frescos	Se añaden 2 placas de hielo y una espuma aislante especial	60x40x40 cm Largo x ancho x alto Medidas interiores tras espumas incluir y refrigeración: 52x34x34 cm
Azul	Congelados	Se añaden 5 placas de hielo y una espuma aislante especial	60x40x40 cm Largo x ancho x alto Medidas interiores tras incluir espumas y refrigeración: 52x32x32 cm
Voluminoso	Voluminosos	Bandeja de 60x40 cm	Bandeja de 60x40 cm. Al ser una bandeja el alto se fijará en función del carro.

Tabla 1: Definición tipos de cajas

Las cajas pueden tener varios tamaños, siempre teniendo en cuenta que lo óptimo es intentar llenar los conocidos como **europalet** al máximo (un ejemplo de europalet se puede observar en la **Ilustración 3**).

Ilustración 3: Ejemplo de cajas en un europalet

- Los pedidos se preparan en dos zonas. La tienda cerrada, que se define como un área de trabajo dónde se almacenan las referencias más utilizadas, y la tienda abierta, anexa a la primera. Por lo tanto, nos encontramos ante un concepto de tienda mixta con reaprovisionamiento desde la segunda a la primera. Primero se recogerán siempre los productos que se puedan desde la tienda cerrada, pasando a recoger a posteriori los productos que no estén en esta a la tienda abierta.
- Los productos en ambas tiendas se clasifican en: refrigerados, congelados, voluminosos, y productos ambiente. Para facilitar la conservación de los

productos refrigerados y congelados, estos se intentarán que sean los últimos recogidos, y tendrán que ser almacenados en cajas especiales.

- El stock se considera infinito en las tiendas cerradas. El menor número de referencias – unas decenas frente a varios miles – provoca que se pueda tener siempre un stock suficiente, además existen procesos de reaprovisionamiento, que no es otra cosa que pasar artículos de la tienda abierta a las estanterías de almacenaje de la tienda cerrada. No obstante, en las tiendas abiertas el control de stock se debe gestionar por los encargados de la tienda y está sujeta a diversos factores, por lo que no se puede descartar que se produzcan **faltantes** en los pedidos. Para minimizar el impacto negativo que tiene el cerrar pedidos sin todos los productos, se pueden generar tablas de **sustitutos** que no son más que productos similares al ordenado, y que los respectivos responsables de sección han definido previamente.
- Los pedidos se pueden gestionar de dos formas: un picker realiza un pedido completo, dejándolo preparado antes de pasar al siguiente (tiendas **monopedido**) o se le proporciona una orden de trabajo con varios pedidos que va rellenando de manera simultánea (tiendas **multipedido)**. Obviamente, esta segunda aproximación reduce enormemente el coste hora-persona, aumentando en la misma proporción la complejidad de procesar las órdenes de trabajo (complejidad que aumenta aún más si añadimos la posibilidad de trabajar con sustitutos). En una tienda multipedido, cada pedido irá en un único carro, pero en un carro pueden ir de 1 a 12 pedidos (el caso más extremo, 12 pedidos, sólo ocurriría si hay 12 pedidos que pueden ser ubicados en una única caja cada uno).
- Los sustitutos sumaran una cantidad de tiempo fija, ya que los productos que van a ser sustituidos están colocado muy próximos al faltante. Los faltantes en productos frescos sí que llevan una operativa que afecta directamente a la productividad en ese, pero para nuestro caso de estudio, la operativa anunciada es irrelevante y se considerará que en todos los casos los productos frescos en rotura de stock se considerarán como faltantes.
- En caso de faltantes que no dispongan de sustitutos, se informará al cliente de que su pedido es incompleto, no cobrando estos productos (los cobros de los

pedidos se realizan una vez este esté completado, con los artículos que hayan podido ser añadidos) e informando al cliente de la ausencia de los mismos.

1.2 Objetivos

El objetivo principal de este trabajo consiste en analizar los algoritmos existentes de gestión de pedidos y la recogida de los mismos, y una vez analizados, adaptarlos a las necesidades específicas de las tiendas propuestas e implementar los algoritmos adaptados, analizando posteriormente los resultados para concretar la mejora de rendimiento en los procesos sobre los que se aplican los algoritmos.

Para ello vamos a seguir una hoja ruta consistente en:

- Describir formalmente el problema.
- Estudiar algoritmos para la resolución de los problemas planteados. Los algoritmos actuarán sobre los procesos de optimización de los procesos de recogida de productos y en la optimización de las rutas.
- Adaptar estos algoritmos al modelo de negocio propuesto.
- Desarrollar los algoritmos más prometedores.
- Analizar los resultados obtenidos en pedidos de ejemplo y proponer posibles mejoras atendiendo a los mismos.

1.3 Motivación

La complejidad que presenta la organización de la recogida de productos en este tipo de tiendas, debido a la gran variabilidad de condiciones, tamaños, pesos e incluso las configuraciones propias de cada tienda provoca que haya una gran variedad de factores sobre los que actuar. Cada aspecto de un proceso industrial o comercial puede ser estudiado y mejorado, lo que repercute directamente en una mejora de la productividad.

Por lo tanto, la adaptación de algoritmos de optimización en procesos con un gran coste hora/hombre aportando incluso ganancias marginales, pueden llevar a incrementos de productividad muy importantes, más aun teniendo en cuenta que son procesos que pueden ser ejecutados cientos de veces al día.

Además, abrimos la puerta a futuras reflexiones sobre cómo optimizar la estructura de los propios almacenes, la distribución de los productos, la gestión de su stock y los procesos de tratamiento de los pedidos, no sólo actuando sobre la recogida, sino también en los procesos de preparación de transporte.

1.4 Estructura del proyecto

- En el Capítulo 1 se detalla la introducción al proyecto, desarrollando las motivaciones que sirven de base al mismo, así como los objetivos alcanzados.
- En el Capítulo 2 se define el problema que se va a estudiar y las posibles soluciones al mismo.
- En el capítulo 3 se introduce el estado del arte actual centrándonos en los algoritmos existentes.
- En el Capítulo 4 comentamos la metodología seguida para la realización del proyecto, así como el desarrollo del mismo.
- En el Capítulo 5 mostramos los datos extraídos de las diversas pruebas a las que hemos sometido a nuestra aplicación.
- En el Capítulo 6 presentamos las conclusiones extraídas durante el desarrollo del proyecto y sentamos las bases de futuras mejoras que puedan usar de base este trabajo.

Capítulo 2

Definición del problema y variantes

2.1 Introducción

Actualmente existen multitud de soluciones para la preparación de pedidos. Casi podemos afirmar que tantas como empresas dedicadas al comercio online. Incluso más, ya que como hemos comentado, una misma empresa puede tener distintos tipos de almacenes o incluso productos diferentes, que pueden necesitar ser procesados siguiendo criterios distintos. Esto es debido a la multitud de variables que definen tanto los almacenes en sí mismos como los procesos implementados y la variabilidad en la naturaleza misma de los productos vendidos.

Los sistemas de gestión de mantenimiento de almacenes (WMS por sus siglas en inglés) llevan décadas desarrollándose (un ejemplo de una organización típica de un almacén y sus procesos asociados lo podemos ver en la **Ilustración 4**), y actualmente hay decenas de soluciones ofertadas por empresas, no obstante, soluciones genéricas implican que no siempre un algoritmo está optimizado al máximo, ya que se debe acudir a soluciones de compromiso que permitan poder ser utilizados por el mayor número de empresas posibles.

Como vemos en (Hompel, 2007), para mejorar los procesos de preparación y envío de bienes, se puede actuar a tres niveles:

- Optimización de la estructura técnica, que define el espacio físico sobre el que se va a trabajar, desde reparto de volúmenes a material y tecnología usada en el almacén. Es un elemento clave, ya que define las distancias a recorrer para realizar tanto la recogida como el guardado, aunque difícilmente optimizable.
- Optimización de la estructura operacional, optimizando la gestión de stock, la organización interna o la interoperabilidad con los transportistas. Este punto es importante, sobre todo en tiendas donde se implemente la opción de la sustitución de faltantes, ya que una buena gestión de almacén tendrá asignados como faltantes productos que se encuentre realmente cerca.
- Optimización de la gestión del almacén. Este aspecto es sobre el que vamos a trabajar, ya que engloba las operaciones de recepción, almacenaje, preparación de pedidos y el propio envío. Como vemos en la ilustración, el flujo normal de procesamiento de productos en un almacén tiene un gran número de pasos, la mayoría de ellos optimizables, pero el ámbito de nuestro proyecto se limita a la recogida (picking), y tratando tangencialmente el almacenaje (store) y el empaquetado (packing).

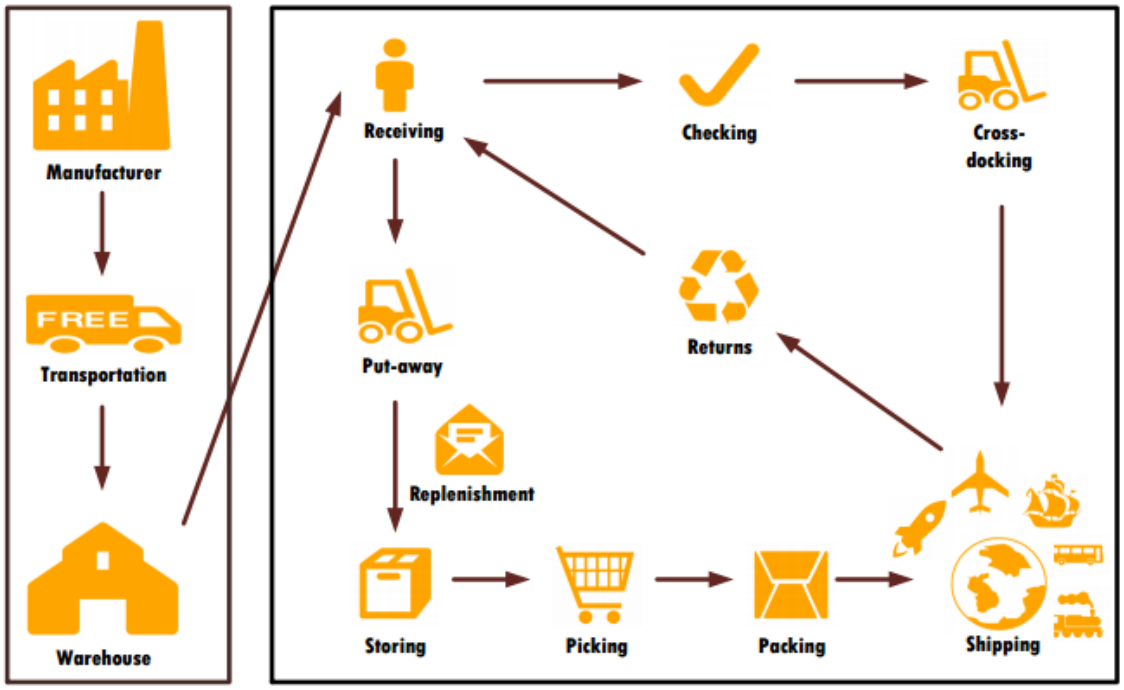


Ilustración 4: Flujo normal de mercancías en un almacén. (Karasek, 2013)

2.2 Descripción del problema

Como hemos descrito anteriormente, nos centraremos en la optimización de la gestión del almacén. Particularmente, trabajaremos en la optimización del tiempo de recogida, minimizando desplazamientos y optimizando la carga de las cajas que componen el pedido. Como podemos ver en la **Ilustración 5**, el proceso que consume el mayor tiempo de trabajo es el desplazamiento hasta la localización del producto, pero no se deben obviar los tiempos de recogida y colocación en la caja.

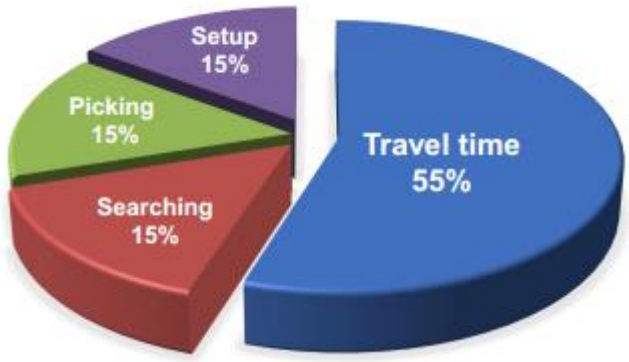


Ilustración 5: Tiempos en procesos de recogida (Johanna Bolaños Zuñiga, 2020)

No obstante, es necesario definir cómo será la estructura del almacén. La composición típica de los almacenes abiertos es la de un pasillo central principal y perpendicular a este las estanterías, con productos en ambas caras. Los productos de frescos (carnicería, frutería, pescadería y panadería) suelen tener espacios específicos, pero no se tomarán en cuenta, ya que la inclusión de frescos no es optimizable ya que todos los productos de la misma categoría se recogen en el mismo lugar. Esta estructura se repite en las tiendas cerradas, ya que son las organizaciones óptimas. En la **Ilustración 6** podemos observar los elementos mínimos a tomar en cuenta para definir la estructura de un almacén, como son los pasillos secundarios, tanto el número como la longitud, si va a haber uno o varios principales que los atraviesen, la localización de los puntos de entrada y de los almacenes o zonas de carga.

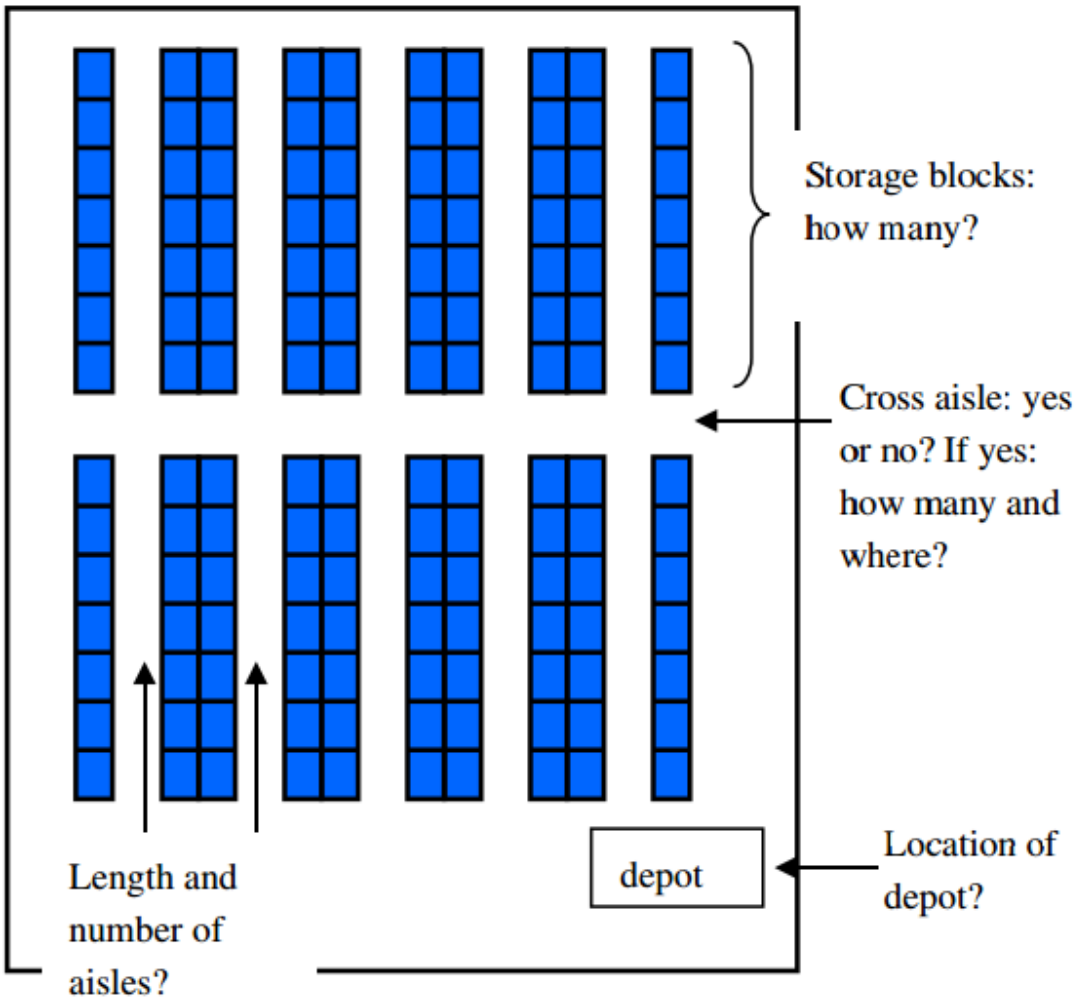


Ilustración 6: Elementos claves en la estructura de almacén. (René de Koster, 2007)

En nuestro caso en la tienda abierta, vamos a definir una estructura de veinte pasillos paralelos (diez a cada lado del pasillo central), con productos a ambos lados, divididos en dos por un pasillo central, en uno de cuyos extremos se localizará el almacén y la tienda cerrada. La tienda cerrada tiene una estructura idéntica a la tienda abierta, pero con sólo cinco pasillos.

Cada pasillo estará dividido en ocho segmentos. Normalmente estos pasillos tienen varias alturas de almacenaje, pero en tiempos de recogida no afecta la altura, siempre y cuando no se necesiten medios adicionales (escaleras o sistemas elevadores, por ejemplo). En la **Ilustración 7** podemos observar una representación del almacén sobre el que vamos a trabajar. En él apreciamos las áreas destinadas a cada tipo de tienda, las zonas de entrada y de almacenaje, el número de pasillos y su estructura.

Por lo tanto, tenemos como objetivo el crear algoritmos que, primero reparten los pedidos entre las cajas necesarias, y posteriormente optimicen la recogida de productos en tiendas con dos ambientes diferentes y definidas para trabajar con multipedido, en un almacén con una estructura clásica de dos pasillos principales y N secundarios perpendiculares.

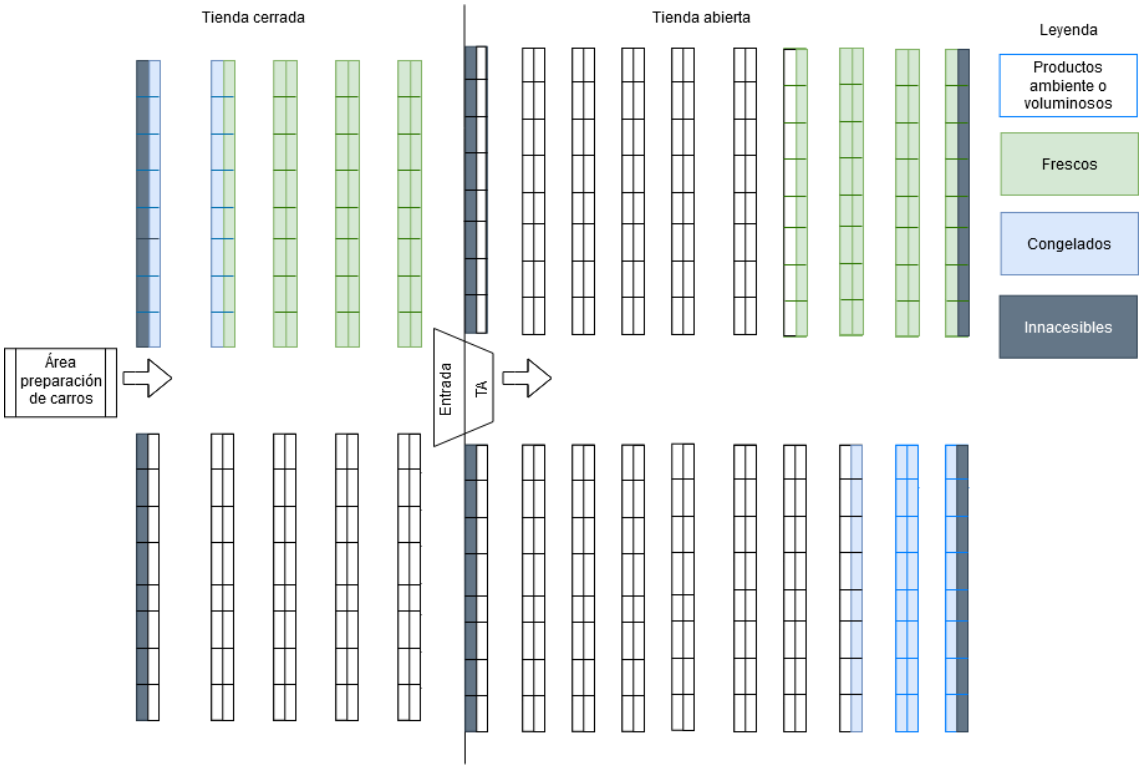


Ilustración 7: Esquema de tienda sobre la que vamos a trabajar

El ancho de los pasillos secundarios es de 2 metros. El ancho de las estanterías será de un metro, dejando una profundidad de balda a cada lado de 0.5 metros.

El pasillo principal tiene un ancho de 4 metros y las estanterías tienen una longitud de 8 metros. A ambos extremos finales de las estanterías, hay un pasillo de 2 metros de ancho, donde los clientes y pickers pueden dar la vuelta.

El tamaño entre pasillos y de los pasillos, tanto central como lateral de la tienda cerrada será el mismo que los de la tienda abierta.

Por lo tanto, tenemos un tamaño total de tienda abierta de 28 metros de largo y 24 de ancho en tienda abierta, y de 17 metros de largo por 24 de ancho en tienda cerrada.

Aparte de trabajar con este tipo de almacenes, definiremos nuestra tienda como multipedido (conocido como **batching**). Esto modifica ligeramente los algoritmos, ya que introduce una mayor cantidad de elementos a recoger.

Los carros de recogida llevarán 12 cajas de recogida (pertenecientes desde a 1 hasta a 12 pedidos), pudiendo ser de frescos, ambiente o congelados. El tamaño de las cajas es de 60cm de ancho, 40 de profundo y 40 de alto, siendo estas una cuarta parte de un europalet, y permitiendo un apilamiento en 3 niveles de hasta 120cm, no sobrepasando las recomendaciones de altura máxima, establecidas en 135 cm y un peso máximo de 15 kilos (AECOC, 2012). Debido a que las cajas tienen apertura superior, las bandejas de los niveles inferior e intermedio deben de ser extraíbles.

Hay que tener en cuenta a la hora de calcular los volúmenes de los refrigerados y congelados que estas cajas necesitan de un aislante térmico y un sistema de refrigeración que limita el tamaño de las mismas.

En un carro puede haber de 1 a 12 pedidos, pero cada pedido debe estar completamente incluido en un único carro. Por lo tanto, cuando el carro vuelve al almacén, las cajas están preparadas para ser depositadas en los europalet que recogerán los transportistas.

La base de datos tendrá información sobre productos de tipo ambiente, voluminosos (que comparten ubicación con los productos de tipo ambiente), frescos y congelados. De los productos almacenaremos al menos la siguiente información:

- Tipo de producto (ambiente, fresco, congelado, voluminoso).
- Tamaño y Peso.
- Cantidad de unidades disponibles y disponibilidad en tienda cerrada, tienda abierta, o ambas.
- Posición dentro de la tienda (número de pasillo y número de segmento).

Con estos productos, generaremos pedidos. Como hemos comentado, estos pedidos pueden tener productos de uno o varios tipos, pero todo pedido debe ser completado en un carro. El sistema deberá de organizar de una forma óptima los pedidos en una o más cajas, respetando la naturaleza de los productos que contienen, y repartir estas cajas en uno o más carros.

Una vez creados los carros, el sistema generará una ruta lo más corta posible, teniendo en cuenta que no siempre va a ser óptima, y que van a ser necesarias aproximaciones heurísticas. Esta ruta incluirá todos los productos que deberán incluir, y en qué caja deben ir estos productos. Tras concluir la ruta debe haber entre 1 y 12 pedidos terminados completamente, y esta ruta debe ser lo más corta posible.

Por lo tanto, a partir de este punto el proyecto se divide en dos estudios separados, que serán de nuevo unificados en las conclusiones. La primera rama estudiará los algoritmos de optimización de llenado de cajas. Si bien estos algoritmos están ampliamente estudiados (Borja Menéndez, 2017), la casuística especial que proponemos (diferentes tipos de productos y recipientes para los mismos, obligatoriedad de que los pedidos sean completamente contenidos en un único carro) genera casos de estudio suficientemente interesantes para su análisis.

La segunda rama del proyecto se centrará en los algoritmos de solución de problemas conocidos como **batching**, o recogida de productos de varios pedidos en una ruta de recogida. Como en el caso anterior, hay estudios sobre este tema, pero mayoritariamente centrados en almacenes sin pasillo central, sin dos ambientes de recogida diferenciados, y sobre todo sin diferenciar entre varios tipos de productos. Por lo tanto, está más que justificado el estudio aquí presentado.

Capítulo 3

Algoritmos existentes

Partimos de una base en la que no se dispone de sistemas de optimización en la creación de carros ni en las rutas que tienen que seguir los mismos.

Los pedidos se añaden en los carros por orden de pedido, otorgando una caja de congelados si los hay, una de frescos si en el pedido hay productos frescos o refrigerados, una bandeja de voluminosos por cada producto voluminoso que haya en los pedidos (independientemente de su tamaño) y una caja de artículos ambiente por cada 50 productos que haya en el pedido.

Como podemos observar, la falta de optimización a la hora de generar las cajas de pedidos puede llevar a una ineficiente preparación de los mismos. Puede darse el caso de que haya 3 artículos voluminosos que podrían ser recogidos en una bandeja o dos (por ejemplo, es posible que 3 paquetes de 16 rollos de papel quepan en una o dos bandejas, mientras que para paquetes más grandes sí que necesites los 3), pero que debido a esta falta de optimización ocuparían un cuarto del carro, en lugar de una doceava parte.

Además, no todos los artículos ocupan el mismo volumen. Un pedido con especias, chicles o frascos de conservas pequeños no llenará una caja en 50 artículos. Sin embargo, un pedido con bebidas o bolsas de gran volumen es posible que necesiten más de una caja para estos 50 artículos.

Igualmente, un pedido en el que haya mucho refrigerado o congelado (compras familiares mensuales) pueden necesitar más de una caja de cada tipo. Esto provocaría que la persona encargada de la recogida de este producto tenga que preparar un nuevo carro tras volver al almacén con el primer carro, y asignarlo manualmente al carro que ya ha preparado, aumentando las posibilidades de fallos en pedidos incompletos y el tiempo total de recogida de los pedidos. En la **Tabla 2** definimos un pedido posible que debido a las características de los artículos que lo componen, supone fallos a la hora de generar las cajas para los pedidos.

Pedidos (Artículos y medidas)	Cajas Previstas	Cajas Reales
3 bolsas de patatas 300g - (34.5x23x9 – 21424.5cm3) **6 botellas Agua 1.5l -** (30x8x8 – 11520cm3) **6 cartones de leche 1l -** (16.5x9.5x6.4 – 6019.2cm3) **Cacao 450g -** (14x9x9 - 1134cm3) **2 paquetes de Galletas 800gr -** (22x12.5x13 – 7150cm3) **2 pan hamburguesa -** (32.6x14.6x8.3 –7901cm3) **12 latas refresco -** (11.5x6.5x6.5 – 5830.5cm3) **2 botellas aceite de oliva -** (26x8x8 – 3328cm3) **1 paquete de chicles -** (12.2x7.1x7.1 – 615.1cm3) **2 barras de pan -** (54x10x5 – 5400cm3) **Detergente Líquido -** (31.8x18.2x7.2 – 4167.1cm3) **Gel Corporal -** (16x6x6 – 575cm3) **Rollo Papel de Cocina -** (39.6x23.2x25 – 22968cm3)	1 caja Ambiente (40 productos – 97416 cm3)	El volumen máximo de una caja es **96000cm3** por lo que necesitaríamos **2 cajas**
Sandía 8 kg - (18x18x18 – 5832cm3) **Saco de naranjas 5kg -** (44x32x32 – 45056cm3) **2 lechugas - (35x22x22 – 16940cm3)** **1 kg melocotones -** (20x20x20 – 4000cm3)	1 caja frescos	**2 cajas** de frescos en **4 productos**

Tabla 2: Ejemplos desajustes carros

En cuanto a las rutas a seguir, se parte de un sistema en el que se asigna manualmente un valor incremental a los productos disponibles, siguiendo una ruta definida por los responsables del almacén según sus estimaciones de qué ruta es la óptima. Por lo tanto, estamos dejando al criterio del personal el estimar qué ruta es la mejor en relación a distancia recorrida.

Si bien esta aproximación puede funcionar aceptablemente (puesto que en los trabajos de almacén puede tenerse en cuenta información no disponible a priori, como estado físico de los empleados, masificación de determinados pasillos, conflictos con los procesos de reaprovisionamiento o tareas de limpieza y mantenimiento, entre otras), siempre será más eficiente optimizar el proceso de recogida y adaptar el resto de los trabajos o casuísticas a este.

Otro factor a tener en cuenta es que al trabajador encargado de la recolección sólo se le muestra la localización e identificador del siguiente elemento a recoger. Esto puede generar problemas como que un recolector esté próximo a un artículo que debe recoger, pero el sistema le indique que vaya hasta otro punto del pasillo, pasando por delante de un artículo que debe recoger posteriormente, o como hemos indicado anteriormente, que el encargado de la preparación del pedido se vea parado en una aglomeración en un pasillo.

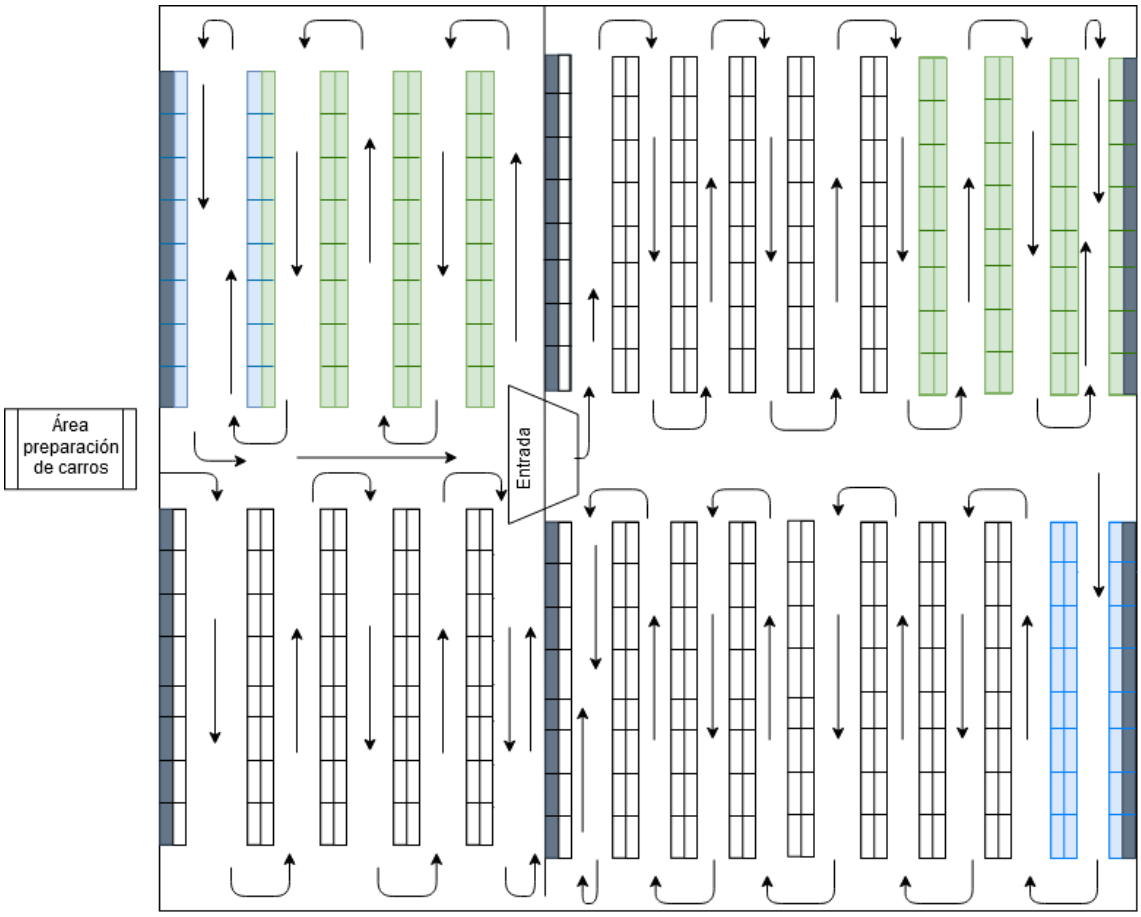


Ilustración 8: Ejemplo de ruta sin optimizar

En la **Ilustración 8** observamos una posible ruta propuesta por un responsable de almacén. Partiendo del área de preparación de carros, el encargado decide que lo mejor es empezar por el pasillo de la derecha, ya que parece lógico empezar por los productos ambiente, y pasar después a los que necesitan de un trato especial, como son los refrigerados y congelados. Tras recoger la zona situada a la derecha según salimos del área de preparación de carritos, pasamos a recorrer la zona situada a la izquierda, pasando dos veces por el mismo pasillo antes de acceder a esta. Al terminar esta zona, volvemos a recorrer dos veces el último pasillo antes de encarar el pasillo principal y acceder a la tienda abierta. Una vez en ella, el encargado decide que es más beneficioso empezar por la zona situada a la izquierda de la entrada, según se accede a la misma. Tras realizar un circuito en S, se procede a atravesar el pasillo central, recorriendo previamente dos veces el mismo pasillo.

Como vemos, si bien esta solución pude parecer la más clara, vemos como el personal se ve forzado a recorrer pasillos por duplicado.

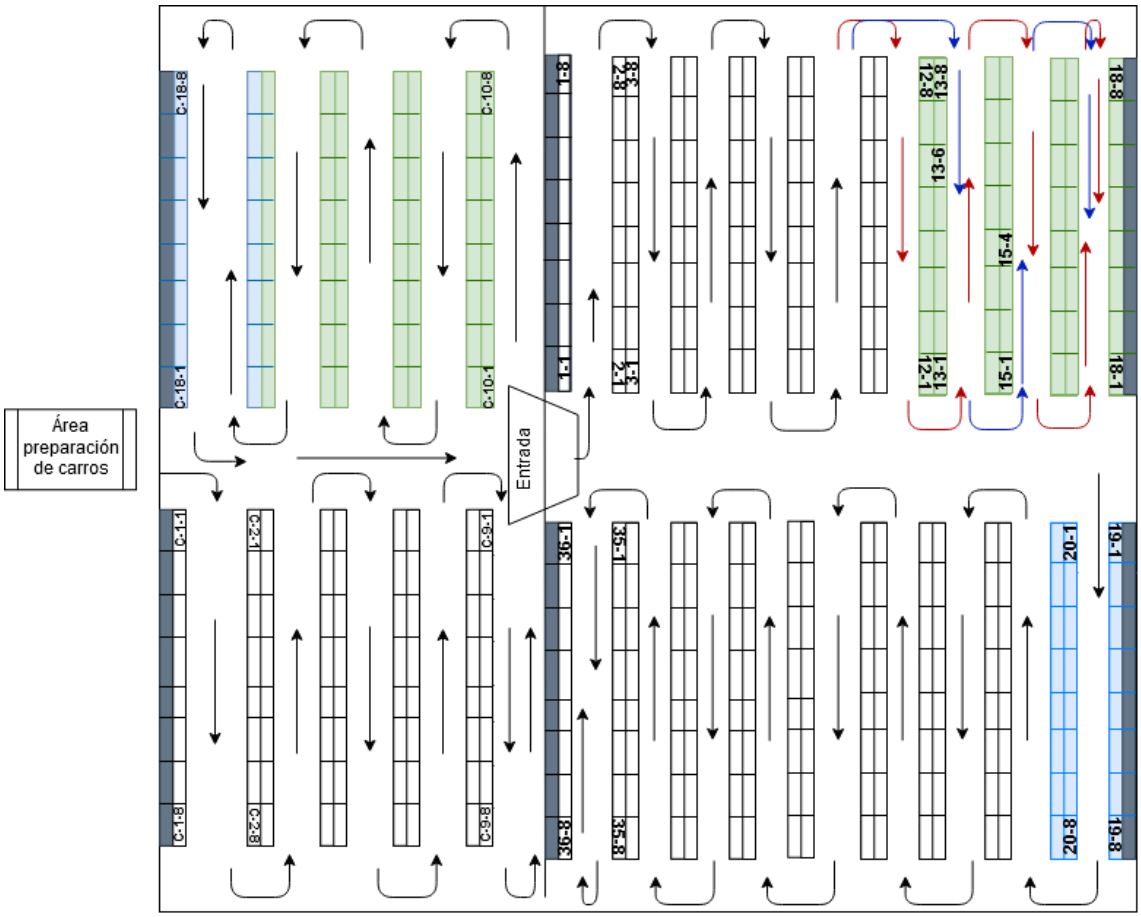


Ilustración 9: Ejemplo de rutas no óptimas

Otro problema que se puede presentar debido a esta aproximación lo vemos en la **Ilustración 9.** Si por ejemplo tuviéramos que recoger tres productos situados en las localizaciones **12-8, 13-6, 15-1, 15-4,** la ruta más rápida sería atravesando los pasillos 13-14 y 15-16 en dirección contraria a la predefinida, lo que aparte evitaría recorrer el pasillo 17-18 en dos ocasiones. No obstante, como el repartidor no dispone de esta información (recordamos que en el dispositivo de recogida sólo se le muestra la información del siguiente artículo a recoger) no puede "optimizar" sobre la marcha la ruta a seguir.

Se han marcado en rojo las giros y pasillos que pasan a ser innecesarias, redundantes o que se recorren en una dirección incorrecta, y en azul la nueva ruta optimizada por el reponedor, saltándose las rutas predefinidas y reduciendo el número de pasillos recorridos, así como la doble pasada al último pasillo.

Tras describir la operativa actual de la que se parte en los trabajos de almacén, pasamos a describir los algoritmos que van a ser estudiados.

El siguiente bloque va a estar dividido en dos partes, la primera definiendo qué algoritmos van a ser estudiados para la creación de las cajas y los carros por pedidos, centrándonos en la segunda parte en los algoritmos de enrutamiento más eficaces para la preparación de los carros previamente definidos.

3.1 Algoritmos para generación de cajas y carros

El problema de reparto de cajas lo vamos a subdividir en dos, primero generaremos el número mínimo de cajas posible para cada pedido. A continuación, agruparemos los pedidos en carros, maximizando el número de pedidos por carro.

3.1.1 Algoritmos para el cálculo del número de cajas – One dimensional bin packing

Partimos de un sistema que genera las cajas basándose en el número de artículos. Como hemos visto en el planteamiento del problema, estos son altamente ineficiente y puede provocar que los encargados de la preparación de pedidos tengan que pasar dos veces por la zona de preparación de carros.

La solución más intuitiva pasa por controlar los volúmenes de los productos y de las cajas, y tomando la cota inferior de esta relación como valor válido. Sumando volúmenes y dividiendo esta suma por el volumen de la caja, tendríamos en la parte entera el número de cajas de cada tipo. Este tipo de problemas se llaman de una dimensión, ya que sólo se tiene en cuenta el volumen de los productos, por lo que se simplifica el cálculo. No obstante, cuando el pedido puede llenar varias cajas y los productos tienen una gran variabilidad de volúmenes el problema se convierte en NP-completo (Krzysztof Fleszar, 2002).

3.1.2 Algoritmos para el cálculo del número de cajas – Three dimensional bin packing

Esta aproximación tiene en cuanta las tres dimensiones del producto a recoger. Trabajaremos directamente con las dimensiones de los productos, no con los volúmenes de los mismos ya que los algoritmos bin-packing de más dimensiones ofrecen mejores resultados (HenrikI Christensen, 2017), aunque estos también impactan en los tiempos de resolución de los mismos.

Como no es el objetivo principal el desarrollo de este tipo de algoritmos, adaptaremos una implementación publicada en (davidmchapman, 2019). Esta librería implementa un algoritmo que genera cajas atendiendo a las dimensiones de las mismas y a las dimensiones de los productos a introducir.

3.1.3 Algoritmos generación carros con batching

Una vez generado el número de cajas que vamos a necesitar de cada tipo para cada pedido, pasamos a agrupar los pedidos de tal forma que se optimicen los carros de pedidos.

Como hemos comentado, nos encontramos con dos factores limitantes. Los carros pueden transportar un máximo de doce cajas, y los pedidos deben ser completados en un carro, no se puede dividir un pedido en varios carros. No obstante, en un carro pueden viajar varios pedidos.

Una de las aproximaciones posibles es la de generar estos lotes (o batch) con pedidos que tengan productos iguales o en su defecto, lo más cercanos posibles. Esta aproximación esta estudiada en (Borja Menéndez, 2017). La aproximación más simple que proponen es la de generar un lote por pedido, y una vez generados, estudiar cuál los siguientes pedidos es más similar, generando previamente una lista de candidatos (Elite Candidate List - ECL). Para ello generan una Lista Maestra, estudiando cómo impacta el quitar pedidos del lote y añadirlos en otros pedidos con suficiente espacio libre. Con esta lista se crea el ECL con los movimientos de la lista que generan una mejora en el resultado final. En la Ilustración 10 podemos observar el pseudocódigo generado para esta solución.

Algorithm 2. InitSolution(α).

1: $S \leftarrow NaiveSolution()$
2: **while** $ThereAreImprovingMoves(S)$ **do**
3: $ML \leftarrow ConstructMasterList(S)$
4: $I_{max} \leftarrow \max_{S \in ML} f(S)$
5: $I_{min} \leftarrow \min_{S \in ML} f(S)$
6: $th = I_{min} + \alpha \cdot (I_{max} - I_{min})$
7: $ECL \leftarrow ConstructEliteCandidateList(ML, th)$
8: $S \leftarrow GetAtRandom(ECL)$
9: **end while**
10: **return** S

Ilustración 10 - Pseudoalgoritmo para problemas de batching

Por último, queremos comentar otra aproximación que dejamos planteada en la sección de trabajo futuro. Esta aproximación consiste en agrupar pedidos por tipo de producto contenido, agrupando, por ejemplo, los pedidos que sólo tengan productos de tipo ambiente, productos que no tengan algún tipo de producto (o varios) o pedidos que puedan ser completados completamente en tienda cerrada.

3.2 Algoritmos para recogida

Los algoritmos de recogida de multipedido con priorización de elementos son **NP-complejo** para recogidas con 3 o más pedidos, mientras que el enrutamiento de los pickers con los condicionantes anteriormente explicados tiene una complejidad de O(N2(P + 1)N) siendo N el número de órdenes y P el número máximo de ítems. (Marek Matusiak, 2014).

A esta limitación matemática se le suman dos limitaciones debido a la naturaleza de la recogida a mano. La primera de ellas es que determinadas rutas pueden parecer ilógicas para los encargados de recoger los productos, por lo que podrían tomar la decisión unilateral de saltarse la ruta propuesta, y segundo, se pueden producir congestiones en los pasillos.

Por lo tanto, se requieren soluciones heurísticas para encontrar soluciones en un tiempo aceptable de tiempo (los recursos para calcular órdenes de recogida no pueden ser ilimitados).

Además, como propone (Makusee Masae, 2020) los algoritmos usados en almacenes con varios bloques de pasillos (**multi-block warehouses**) difieren de los usados en distribuciones mono bloque. Por ejemplo, los Algoritmos Exactos no son aplicables debido a la complejidad que añade el enrutamiento a través de varios subniveles de almacenaje.

Por lo tanto, las soluciones propuestas son **Heurísticas** (o bien **metaheurísticas** si se deciden añadir complejidades extras a la solución, tales como la consideración de posibles congestionamientos en determinados pasillos, pero que, al exceder el ámbito del presente documento, no serán tenidas en cuenta) y se desarrollarán de forma detallada en los siguientes apartados.

3.2.1 Recorrido en S en múltiples bloques

Similar al ejemplo presentado en la **Ilustración 11**, consiste en empezar en un lateral del almacén e ir recorriendo los pasillos en los que haya artículos a recoger hasta llegar al final de un bloque (izquierdo o derecho) y una vez alcanzado el final del mismo, cambiar al otro bloque, a través del primer pasillo en el que encontremos artículos para recogida. En la **Ilustración 10** tenemos un ejemplo propuesto en (Makusee Masae, 2020) de cómo un algoritmo basado en recorrido en S, genera una ruta para un pedido de ejemplo.

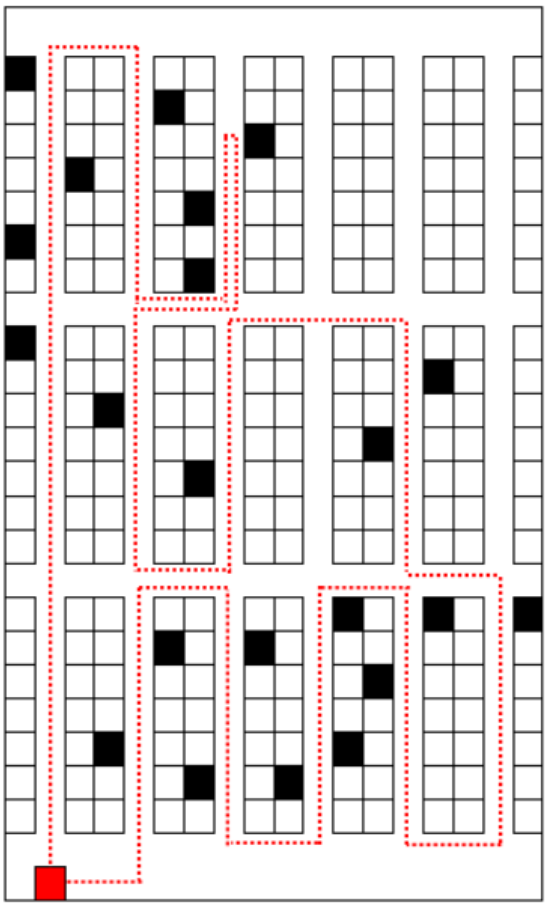


Ilustración 11: Ejemplo de recorrido en S

Este tipo de algoritmos es muy adoptado por la extrema sencillez de las rutas, ya que los recolectores siempre terminan los pasillos auxiliares al completo, porque los puntos de giro de los carros suelen ser anchos. En nuestro caso no es determinante, ya que los carros de recogida no son especialmente voluminosos, pero en almacenes en los que se use para la recogida maquinaria (especialmente maquinaria en altura) sí que puede ser un factor determinante.

3.2.2 Espacio más largo en múltiples bloques

Este algoritmo secuencia los bloques de la misma manera que el algoritmo en forma de S, con la salvedad de que divide los pasillos secundarios en dos mitades, usando el espacio más largo entre dos artículos requeridos o entre la salida del pasillo y el siguiente ítem para definir qué extremo se considera como inicial o final del pasillo.

En la **Ilustración 12** tenemos un ejemplo propuesto en (Makusee Masae, 2020) de cómo un algoritmo basado en recorrido en S, genera una ruta para un pedido de ejemplo.

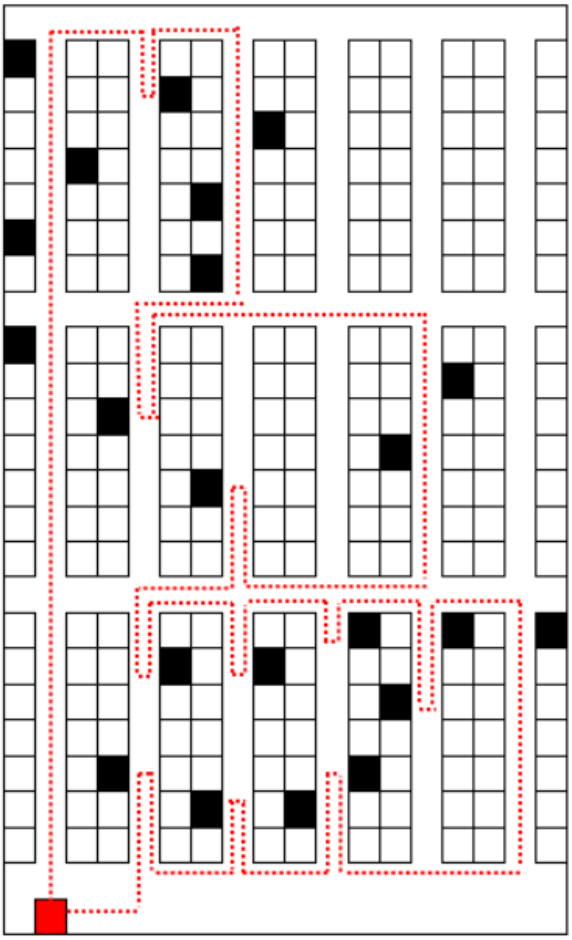


Ilustración 12: Ejemplo de recorrido con espacio más largo

Como vemos, ahora no siempre se recorre el pasillo completo, forzando a los encargados de la recolección de girar en mitad del pasillo a veces, y pasando dos veces frente al mismo producto.

3.2.3 Combinado

Comienza como el recorrido en S, pero con la salvedad de que calcula los puntos más lejanos en los que hay artículos para definir una nueva cabecera de entrada al siguiente pasillo y el punto en el que debe retornar la persona encargada de la recolección, en caso de que en ese pasillo no haya más elementos, y teniendo en cuenta la distancia más larga hasta el siguiente artículo.

También se añade la división en dos mitades de los pasillos secundarios, lo que modifica el punto de salida de los mismos, en caso de que no haya más artículos a recoger en ellos.

En la **Ilustración 13** tenemos un ejemplo propuesto en (Makusee Masae, 2020) de cómo un algoritmo basado en recorrido combinado genera una ruta para un pedido de ejemplo.

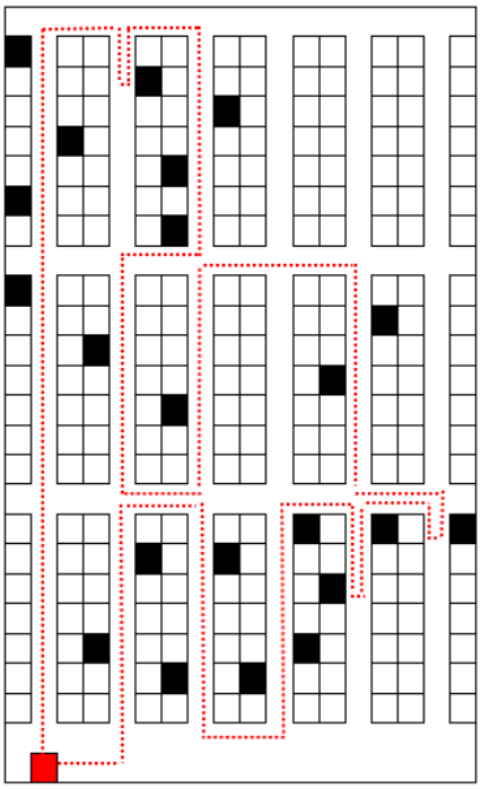


Ilustración 13: Ejemplo de recorrido combinado

Como vemos, la tendencia inicial es la de completar el recorrido siguiendo una forma de S. No obstante hay ciertos puntos en que la distancia al siguiente elemento provoca un cambio de dirección en un punto intermedio del pasillo.

3.2.4 Combinado+

Partiendo del algoritmo anterior, para los elementos en el bloque más cercano al punto de partida, se tiene en cuenta cómo van a ser recogidos de derecha a izquierda, ya que los elementos situados más a la izquierda no tienen por qué recogerse en el camino de vuelta, si no que podemos añadirlos modificando la ruta de vuelta.

En la **Ilustración 14** tenemos un ejemplo propuesto en (Makusee Masae, 2020) de cómo un algoritmo basado en recorrido combinado+, genera una ruta para un pedido de ejemplo.

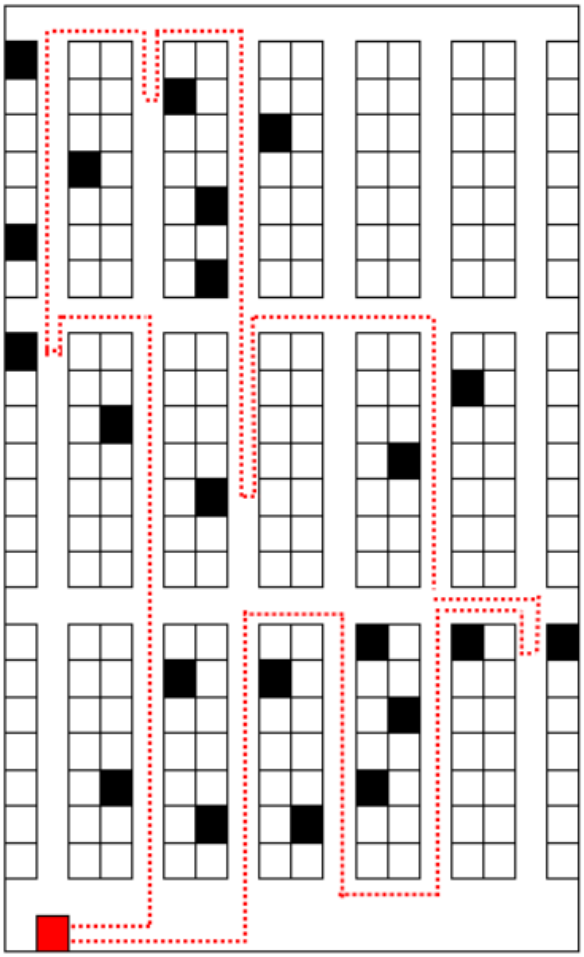


Ilustración 14: Ejemplo de recorrido combinado+

En este caso nos encontramos el primer caso en el que la vuelta al almacén no se hace por el pasillo más cercano, sino que se hace por el que tiene artículos en el bloque más cercano al mismo.

3.2.5 Óptimo

En (Kees Jan Roodbergen, 2001) tenemos además de una nueva descripción de algunos de los métodos anteriormente enumerados, un nuevo algoritmo llamado Óptimo. Esta aproximación trata el problema como un caso especial del Problema del Vendedor Viajero (Travelling Salesman Problem -TSP en inglés y cuya explicación podemos ver en más detalle en (Ma, 2020)). Este problema consiste en visitar todos los puntos necesarios reduciendo la distancia recorrida al máximo posible. Para ello usan una aproximación basada en el algoritmo Branch and Bound (ver **Ilustración 15**) aplicado al algoritmo TSP anteriormente descrito.

Algorithm 1: Branch-and-Bound(X, f)

```
1 Set L = {X} and initialize x̂
2 while L ≠ ∅:
3     Select a subproblem S from L to explore
4     if a solution x̂' ∈ {x ∈ S | f(x) < f(x̂)} can be found: Set x̂ = x̂'
5     if S cannot be pruned:
6         Partition S into S_1, S_2, ..., S_r
7         Insert S_1, S_2, ..., S_r into L
8     Remove S from L
9 Return x̂
```

Ilustración 15: Algoritmo Branch-and-Bound

En la Ilustración 16 podemos ver una ruta de ejemplo calculada en (Kees Jan Roodbergen, 2001).

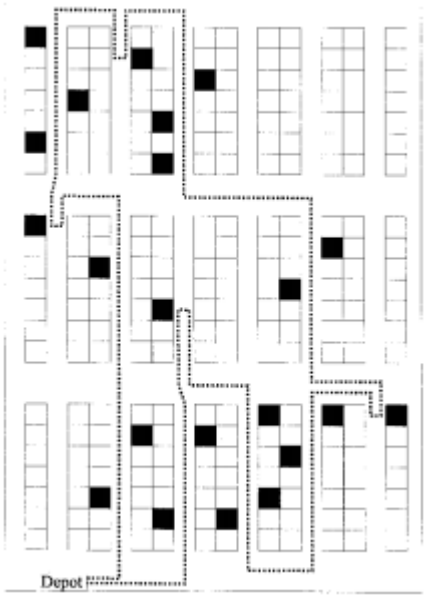


Ilustración 16: Ejemplo ruta algoritmo Óptimo (Kees Jan Roodbergen, 2001)

Capítulo 4

Metodología

4.1 Arquitectura y Framework de desarrollo

Se ha optado por una aplicación de escritorio alimentada por una base de datos relacional. Se ha elegido esta aproximación por ser robusta y eficiente, características básicas en una aplicación que necesita ser fiable y tener absoluta disponibilidad y fiabilidad en los datos.

Para ello, como framework de desarrollo se usará un entorno basado en el IDE Visual Studio Community ya que nos proporciona las herramientas necesarias para generar una aplicación como la propuesta y gestionar la base de datos conjuntamente. Además, se trata de un IDE gratuito y ampliamente extendido, por lo que podemos animar a los lectores a continuar con esta línea de investigación sin necesidad de comprar licencia de la herramienta.

Para el desarrollo de la base de datos se ha utilizado una distribución gratuita de SQL Server Management Studio. En particular la versión 18, si bien es posible utilizar cualquier software que permita la gestión de BBDD relacionales.

4.2 Descripción de la metodología elegida

Una vez analizada la naturaleza del problema, se ha decidido adoptar una metodología ágil, ya que el desarrollo incremental y la adaptación a requisitos cambiantes, dos de los

casos de uso para los que se definió Scrum (Ken Schwaber, 2020), se consideran como básicos para este proyecto.

En concreto, Scrum se postula como candidato ideal, ya que podemos partir de la definición general de la solución e ir depurándola en las sucesivas iteraciones (Sprint), actualizando los puntos clave entre las mismas. No obstante, a pesar de seguir los valores que rigen esta metodología, tenemos que adaptar de una forma extremadamente flexible los roles de usuario, que quedarían definidos como sigue:

- **Product Owner:** Francisco Parreño Torres y Víctor López Jaquero. Definen los artefactos que servirán de guía al equipo, optimizándolos para dar el máximo valor añadido a cada iteración (sprint).
- **Equipo de desarrollo:** Eduardo Pastor de la Cruz. Un equipo Scrum debe ser capaz de auto-organizarse y todos sus miembros deberían ser capaces de añadir valor en todas las competencias demandadas al equipo. Como vemos este rol, si bien está pensado para equipos más numerosos, encaja perfectamente con el espíritu de los proyectos de fin de estudios.
- **Scrum Master:** Eduardo Pastor de la Cruz. Si bien es cierto que podemos encontrar casos en los que el Scrum Master es miembro del equipo de desarrollo (Varios, 2020), es posible que surjan ciertos conflictos de intereses. No obstante, el principal papel del Scrum Master es hacer comprender al equipo de desarrollo las ideas que ha querido transmitir el Product Owner en la definición de requisitos, y a la vez, transmitir a este los problemas que está encontrando el equipo de desarrollo, por lo que no es descabellado que se solapen los puestos de Scrum Master y desarrollador.

Una vez definidos los roles pasamos a enumerar los **artefactos** sobre los que descansa la responsabilidad de encontrar necesidades y de aportar valor añadido al proyecto.

- **Product Backlog**: Si bien este documento debe estar sometido a revisión durante todo el proyecto, en un estadio inicial del mismo se puede usar como guía de referencia, en la que se describen a alto nivel las necesidades, las prioridades de las misas y el valor que añaden al proyecto. El Product Owner es el encargado de realizarlo y actualizarlo.

- **Sprint Backlog**: Se puede definir como un plan suficientemente detallado para ser guía del trabajo diario del equipo de desarrollo. Proviene de la segmentación y priorización del Product Backlog. Al igual que el anterior, puede ser actualizado si los requisitos cambian durante la iteración.
- **Incremento**: son todos los ítems completados durante una iteración. Estos incrementos deben de ser productos usables, que tengan un valor añadido.

Scrum por último define los eventos en los que debemos de programar para no caer en un excesivo número de reuniones y para fomentar la comunicación y colaboración.

- **Sprint**: Es cada iteración en la que se divide el proyecto. En un sprint se debe finalizar los requisitos definidos en el Sprint Backlog y debe entregarse una versión completamente usable.
- **Panificación de Sprint**: Reunión previa al inicio del Sprint, de corta duración en la que se define qué y cómo se va a ejecutar.
- **Daily Scrum**: Reunión diaria de 15 minutos en la que el equipo de desarrollo planifica el trabajo para las siguientes 24 horas.
- **Sprint Review**: Tiene lugar cuando acaba un Sprint y su objetivo es el de adaptar el Product Backlog, si se han detectado nuevos requisitos.
- **Sprint Retrospective**: Se usa para analizar el último Sprint a nivel técnico y de relaciones entre el equipo, con el objetivo de mejorar los procesos en la siguiente iteración.

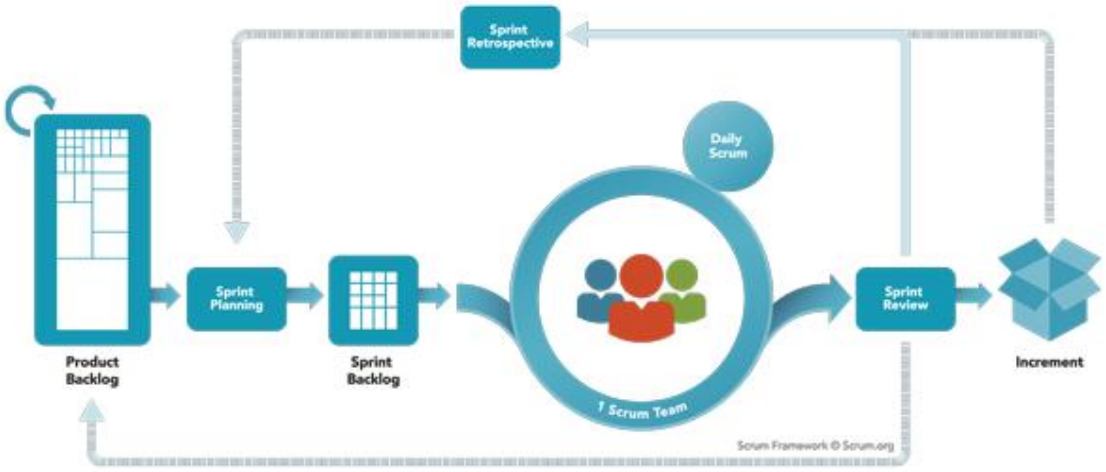


Ilustración 17: Marco de trabajo de Scrum (Francia, 2020)

En la Ilustración 17 podemos ver cómo interactúan los artefactos con los eventos.

4.3 Dinámica de trabajo

Basándonos en la metodología expuesta, y una vez definidos los roles necesarios, pasamos a definir cómo se va a estructurar el desarrollo y la creación del TFG (que no deja de ser un Incremento final, mejor documentado).

El Product Backlog se irá definiendo a través de las reuniones iniciales con los tutores del proyecto y se añadirá en el apartado 2.2 del presente documento.

Para definir este documento nos apoyaremos en historias de usuario. Si bien no es un artefacto definido por Scrum, sí que es una práctica recomendada en las metodologías ágiles para la toma de requisitos, ya que hay una relación directa entre una historia de usuario con una funcionalidad de software, escrita desde la perspectiva del usuario final o cliente (Rehkopf, 2020).

Los Sprint Backlog se definirán formalmente y se añadirán en Anexos.

Entre cada uno de los Sprint se realizará una reunión en la que se englobarán el Sprint Review y la Sprint Retrospective para analizar la entrega del producto, así como el Sprint Planning para dejar preparadas las bases del siguiente Sprint. Las Daily Scrum se sustituyen por consultas a través de email o chats de texto y las Sprint Review pasan a hacerse semanalmente (2 por Sprint) para suplir las deficiencias en comunicación por la supresión de las Daily Scrum.

Por último, si bien es cierto que el Product Owner debería producir ciertos documentos, la naturaleza de un TFG hace que esta aproximación no sea adecuada, por lo que toda la documentación será creada por el alumno, encargándose el tutor de su validación.

Todos los artefactos generados se pueden consultar en el **Anexo I** del presente documento.

4.4 Desarrollo

Primero definiremos las historias de usuario, sobre las cuales construiremos el Product Backlog, y que también usaremos posteriormente para redefinir las tareas en los Sprint Backlog (ver Anexo I para consultar estos artefactos).

Una vez definidas las historias de usuario y el Product Backlog, pasamos a definir los Sprint. En los siguientes puntos se puede consultar un resumen de los 3 Sprint en los

que se dividió la implementación del proyecto, si bien la información detallada de los mismos se puede consultar en el Anexo I.

4.4.1 Historias de Usuario

Si bien las historias de usuario tienen que ser sencillas, claras y concisas, creemos que debido a la naturaleza académica del problema, es recomendable que además sean fácilmente probadas (aunque creemos que esta recomendación debería ser tenida en cuenta en todos los desarrollos ágiles). Como detallan en (Cohn, 2004) una historia definida como "*Un usuario no debe esperar mucho para que aparezca la siguiente pantalla tras una acción*" no pueden ser probadas empíricamente (¿cuánto tiempo es poco tiempo?), por lo que la automatización de los test y la satisfacción en el cumplimiento de las historias.

Teniendo en consideración esta limitación y adoptando las recomendaciones de (Rehkopf, 2020) seguiremos esta estructura en las historias de usuario:

"Como – Quiere – Para".

Tras el "*como"* se define el destinatario último del desarrollo, en "*Quiere*" se describe la intención del usuario, no funcionalidades explícitas y tras "*Para*" se definen los problemas que debe resolver esta historia. Por lo tanto, se definen las siguientes historias para resolver los problemas propuestos en este proyecto:

- Como encargado de pedidos online, quiero tener información del número de cajas que tengo que preparar por cada pedido, para evitar cometer errores en mi trabajo.
- Como encargado del almacén, quiero saber qué cajas van en cada carro, para organizar de una forma más eficiente mi trabajo y el de mi equipo.
- Como preparador de pedidos, quiero saber la ruta que tengo que seguir, para ahorrarme trabajo y ser más eficiente.

4.4.2 Primer Sprint

En este primer Sprint se desarrollará una BBDD funcional y se alimentará, y un entorno básico desde el que se podrán acceder a los datos almacenados en la misma.

Como producto entregable se presentará el sistema de reparto de pedidos en cajas. Los encargados de la recogida sabrán exactamente cuántas cajas necesitan, ahorrándose

posibles problemas de falta de cajas, y entregándoles información para que puedan asegurar que cada pedido se termina en una sola ruta.

Tras entregarse esta versión se revisará si durante el Sprint se han detectado nuevos requisitos a nivel de definición de elementos.

4.4.3 Segundo Sprint

En este segundo sprint se entrega el optimizador de rutas. Tras este sprint se revisa si han aparecido nuevos requisitos y se acepta el plan de trabajo del tercer sprint.

Tras este Sprint se pueden considerar cumplidos los objetivos principales, ya que dispondremos de un sistema capaz de gestionar los pedidos y optimizar las rutas, mejorando la productividad del personal de recogida. No obstante, se integra un tercer sprint, con el objetivo de aumentar el rendimiento de los algoritmos anteriores.

4.4.4 Tercer Sprint

En el tercer Sprint usaremos los datos recogidos en la iteración anterior y se implementan las mejoras decididas y se entrega el incremento final, incluyendo la optimización de la creación de los carros. Con esta entrega no sólo garantizamos que los pedidos se terminan en una sola ruta, sino que al optimizar la generación de carros se procede a un reparto más equitativo de los mismos, ya que se intentará que los primeros carros vayan cargados al máximo.

Tras este sprint se revisa si el desarrollo completo ha sido exitoso.

4.4.5 Incremento Final

Se entrega la versión 1.0 del software. Al ser un desarrollo fundamentalmente académico, se presentarán también los datos recopilados durante la implementación y se definirán posibles mejoras para trabajos futuros.

Capítulo 5

Experimentos y Resultados

5.1 Experimentos y resultados

En esta sección se muestra un resumen de la información utilizada en los experimentos, así como los resultados obtenidos.

Primero se presenta una tabla (**Tabla 3**) con información relacionada con los artículos, así como su disposición en tienda. Posteriormente, en la **Tabla 4** se presentan ejemplos de pedidos que podría realizar un cliente. Sobre pedidos similares a estos se aplicarán los algoritmos de creación de carros y de optimización de rutas, y se presentarán los resultados obtenidos.

ID Artículo	Artículo	Medidas	Localización TA	Localización TC
1	Bolsas de patatas 300g	34.5x23x9	4 - 1	-
2	Cartón de leche 1l	16.5x9.5x6.4	15 - 11	C1 - 12
3	Paquete de galletas 800gr	22x12.5x13	7 - 15	-
4	Pan hamburguesa	32.6x14.6x8.3	7 - 2	-
5	Lata refresco	11.5x6.5x6.5	15 - 5	C2 - 3
6	Botella aceite de oliva 1.5l	26x8x8	1 - 1	C1 - 1
7	Paquete de chicles	12.2x7.1x7.1	5 -6	-
8	Barra de pan	54x10x5	6 -17	-
9	Detergente líquido	31.8x18.2x7.2	18 - 8	-
10	Gel Corporal	16x6x6	14 - 3	-
11	Botella Agua 1.5l	30x8x8	16 - 12	C8 - 5
12	Rollo papel de cocina	39.6x23.2x25	17 - 3	-
13	Sandía 8 kg	18x18x18	7 – 3	-
14	Saco de naranjas 5kg	44x32x32	7 – 12	C7 - 3
15	Lechuga	35x22x22	8 – 5	-
16	Melocotón	20x20x20	9 – 2	-
17	Helado Chocolate 1l	21x11x11	10 -6	-
18	Fideos 450g	5.9x8.7x15	3 - 8	
19	Pizza congelada 340g	27.5x26.8x2.6	10 -12	C10 - 2
20	Guisantes congelados 200g	3.63x2.43x0.99	10 -17	-
21	Papel higiénico 63 rollos	77.5x31.75x29	18 – 11	C5 - 12
22	Chocolate 300g	23.5x9.6x1.3	2 - 12	C4 - 9

Tabla 3: Artículos

ID Pedido	ID Artículo	Cantidad
1	1	3
1	4	1
1	5	2
1	7	4
1	8	1
1	14	1
1	15	2
1	17	1
1	21	1
2	11	6
2	16	1
2	6	2
2	12	1
2	18	2
2	1	1
2	3	2

Tabla 4: Ejemplo de pedidos

5.2 Resultados algoritmo creación de cajas

Para probar los algoritmos de creación de cajas usamos los siguiente pedidos (en la BBDD son los pedidos correspondientes a la fecha del 03/01/2021), recogidos en la **Tabla 5**.

EAN_linea_pedido	Id_pedido_linea_pedido	cantidad
1	4	2
2	4	6
5	4	12
6	4	2
11	4	6
13	4	1
15	4	2
17	4	1
19	4	2
21	4	1
2	5	6
3	5	1
8	5	1
10	5	1
9	5	1
18	5	2
22	5	1
2	6	6
4	6	1
5	6	24
6	6	2
12	6	2
8	6	1
3	6	1
11	6	6
20	6	2
17	6	1
13	9	2

14	9	2
16	9	2
5	10	48
21	10	2

Tabla 5: Pedidos sobre los que se ejecutan los algoritmos

Tras tener la BBDD cargada con esta tanda de pedidos inicial, y una vez finalizada la interfaz gráfica, se procede a reproducir una simulación de cómo sería el reparto en cajas en el estado inicial del problema. En este punto no tenemos optimización de cajas, ni de rutas ni de carros. Tras esta ejecución podemos ver los datos presentados en la interfaz.

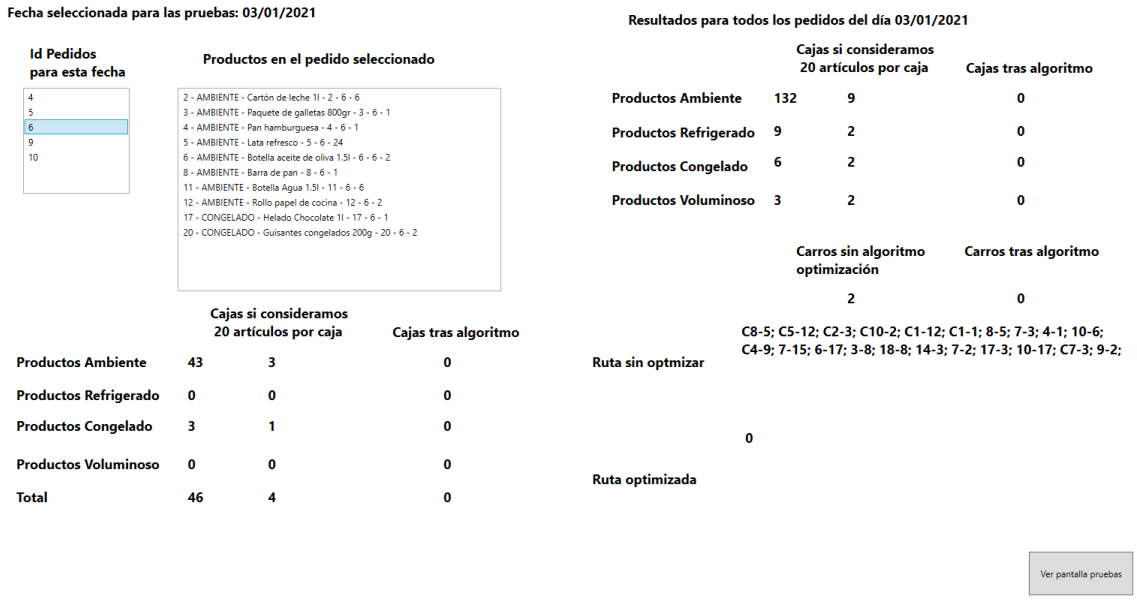
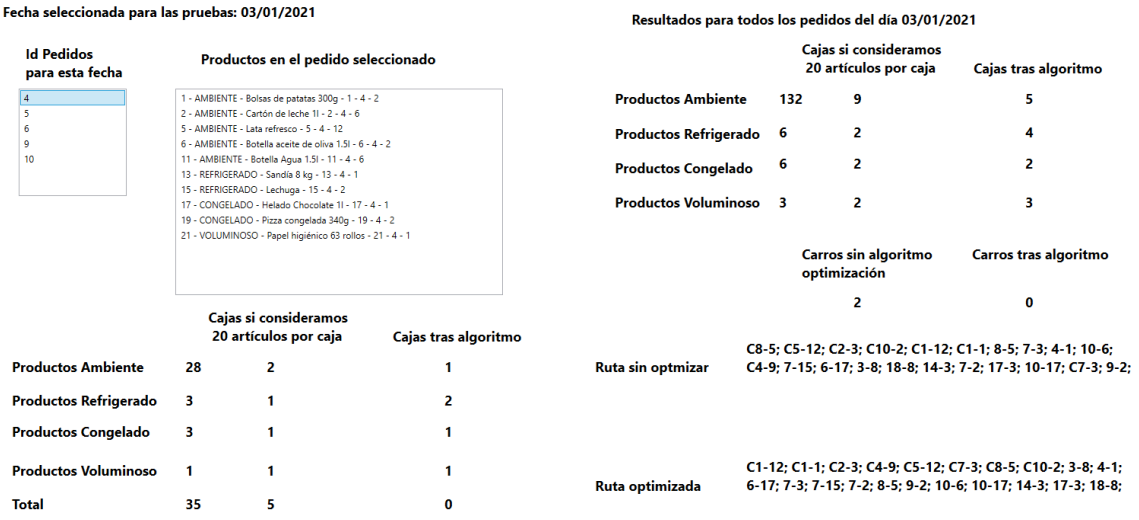


Ilustración 18: Resultados previos a la optimización

Como podemos observar en la **Ilustración 18**, se ha optado por implementar en la parte izquierda de la interfaz un módulo que permite al usuario consultar datos de los pedidos por la fecha preseleccionada, mientras que en la parte derecha se ha dejado la visualización global de los datos, con los que el usuario no puede interactuar, pero que reflejan la información más relevante para ejercer el trabajo de recolección de los pedidos.

Tras implementar el algoritmo de optimización de creación de cajas, los resultados son bien diferentes, como podemos ver en la **Ilustración 19**.



Fecha seleccionada para las pruebas: 03/01/2021

Id Pedidos para esta fecha

4
5
6
9
10

Productos en el pedido seleccionado

1 - AMBIENTE - Bolsas de patatas 300g - 1 - 4 - 2
2 - AMBIENTE - Cartón de leche 1l - 2 - 4 - 6
5 - AMBIENTE - Lata refresco - 5 - 4 - 12
6 - AMBIENTE - Botella aceite de oliva 1.5l - 6 - 4 - 2
11 - AMBIENTE - Botella Agua 1.5l - 11 - 4 - 6
13 - REFRIGERADO - Sandía 8 kg - 13 - 4 - 1
15 - REFRIGERADO - Lechuga - 15 - 4 - 2
17 - CONGELADO - Helado Chocolate 1l - 17 - 4 - 1
19 - CONGELADO - Pizza congelada 340g - 19 - 4 - 2
21 - VOLUMINOSO - Papel higiénico 63 rollos - 21 - 4 - 1

	Cajas si consideramos 20 artículos por caja		Cajas tras algoritmo
Productos Ambiente	28	2	1
Productos Refrigerado	3	1	2
Productos Congelado	3	1	1
Productos Voluminoso	1	1	1
Total	35	5	0

Resultados para todos los pedidos del día 03/01/2021

		Cajas si consideramos 20 artículos por caja	Cajas tras algoritmo
Productos Ambiente	132	9	5
Productos Refrigerado	6	2	4
Productos Congelado	6	2	2
Productos Voluminoso	3	2	3

	Carros sin algoritmo optimización	Carros tras algoritmo
	2	0

Ruta sin optmizar	C8-5; C5-12; C2-3; C10-2; C1-12; C1-1; 8-5; 7-3; 4-1; 10-6; C4-9; 7-15; 6-17; 3-8; 18-8; 14-3; 7-2; 17-3; 10-17; C7-3; 9-2;
Ruta optimizada	C1-12; C1-1; C2-3; C4-9; C5-12; C7-3; C8-5; C10-2; 3-8; 4-1; 6-17; 7-3; 7-15; 7-2; 8-5; 9-2; 10-6; 10-17; 14-3; 17-3; 18-8;

Ilustración 19: Cajas tras la optimización

Como podemos observar, el total de las cajas es muy diferente. Las cajas de producto ambiente han pasado de nueve a cinco. Esto es debido a que los productos de tipo Ambiente suelen ser pequeños y muy numerosos, por lo que en la mayoría de las ocasiones podremos incluir más de 20 artículos por caja. Por otro lado, los artículos Refrigerado y Voluminoso necesitan de más cajas. Esto es debido a que los productos refrigerados incluidos en estos pedidos son más grandes. Lo mismo ocurre en los productos Voluminoso. En cada caja sólo podemos incluir un producto de nuestros pedidos, por lo que en los pedidos con más de un voluminoso, se le asignará una caja por cada producto.

Si atendemos a los resultados por artículo, podemos ver que en el pedido con **Id Pedido 4**, las cajas pasan de dos cajas de tipo Ambiente a una, debido a que los productos son más de veinte, pero de pequeño tamaño. Sin embargo, se genera una caja adicional para pedidos Refrigerado. Esto es debido a que uno de los productos es muy voluminoso, lo que provoca que a pesar de contener en este pedido solamente tres productos de Refrigerado, estos necesitan dos cajas para ser completamente almacenados.

Fecha seleccionada para las pruebas: 03/01/2021

Id Pedidos para esta fecha

4
5
6
9
10

Productos en el pedido seleccionado

5 - AMBIENTE - Lata refresco - 5 - 10 - 48
21 - VOLUMINOSO - Papel higiénico 63 rollos - 21 - 10 - 2

		Cajas si consideramos 20 artículos por caja	Cajas tras algoritmo
Productos Ambiente	**48**	**3**	**1**
Productos Refrigerado	**0**	**0**	**0**
Productos Congelado	**0**	**0**	**0**
Productos Voluminoso	**2**	**1**	**2**

Ilustración 20: Ejemplo de pedido

En la **Ilustración 20** vemos un ejemplo donde se aprecia más claramente el cómo esta optimización afecta especialmente a los productos Voluminoso o a los productos de pequeñas dimensiones. Con sólo dos artículos de tipo Voluminoso en el pedido con **Id Pedido 10** necesitamos dos cajas de tipo Voluminoso. No obstante, 48 artículos Ambiente pasan a tener asignadas tres cajas a sólo una. Esto es debido a que el volumen de una lata es relativamente pequeño, por lo que pueden ser perfectamente acomodadas en una sola caja. En este pedido, la optimización ha mejorado la eficiencia de las recogidas en dos aspectos clave. El recolector no necesita volver al almacén a por una caja de tipo Voluminoso extra, y por otro lado, no vuelve de su recorrido con dos

cajas vacías, pudiéndose aprovechar ese hueco para añadir uno o dos pedidos adicionales. Podemos ver más en detalle como el algoritmo trabaja sobre las cajas para llegar a sacar estas conclusiones.

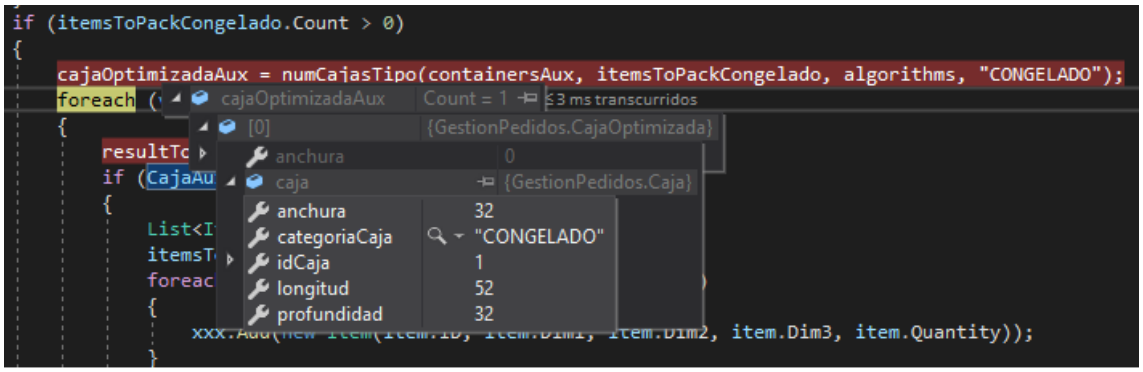


Ilustración 21: Ejemplo implementación Caja

En la **Ilustración 21** podemos observar una caja de tipo Congelado sobre la que hemos hecho una iteración de cara a optimizar los pedidos que contiene.

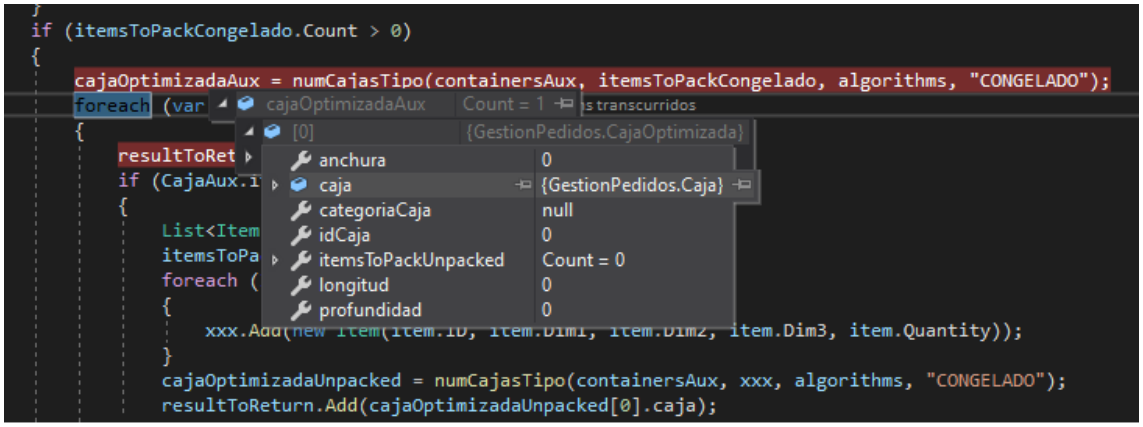


Ilustración 22: Ejemplo implementación Caja sin elementos sin empaquetar

En la **Ilustración 22** tenemos un ejemplo de una caja de tipo Congelado sobre la que hemos hecho una iteración de cara a optimizar los pedidos que contiene. Vemos que en esta iteración todos los productos han entrado en la primera iteración, por lo que se marca como cero el número de productos no empaquetados.

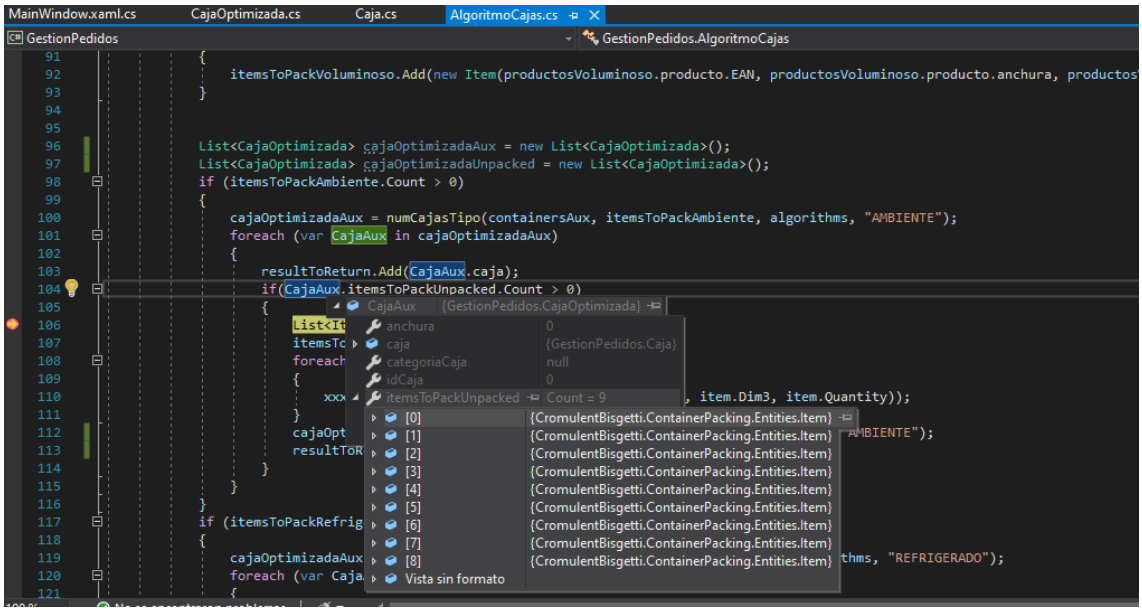


Ilustración 23: Lista de elementos sin empaquetar en una primera iteración

En la **Ilustración 23** vemos como se tratan los elementos que no han podido ser introducidos en una caja en la primera iteración. Sobre este listado iteratemos nuevamente para introducirlos en una nueva caja, generada dinámicamente.

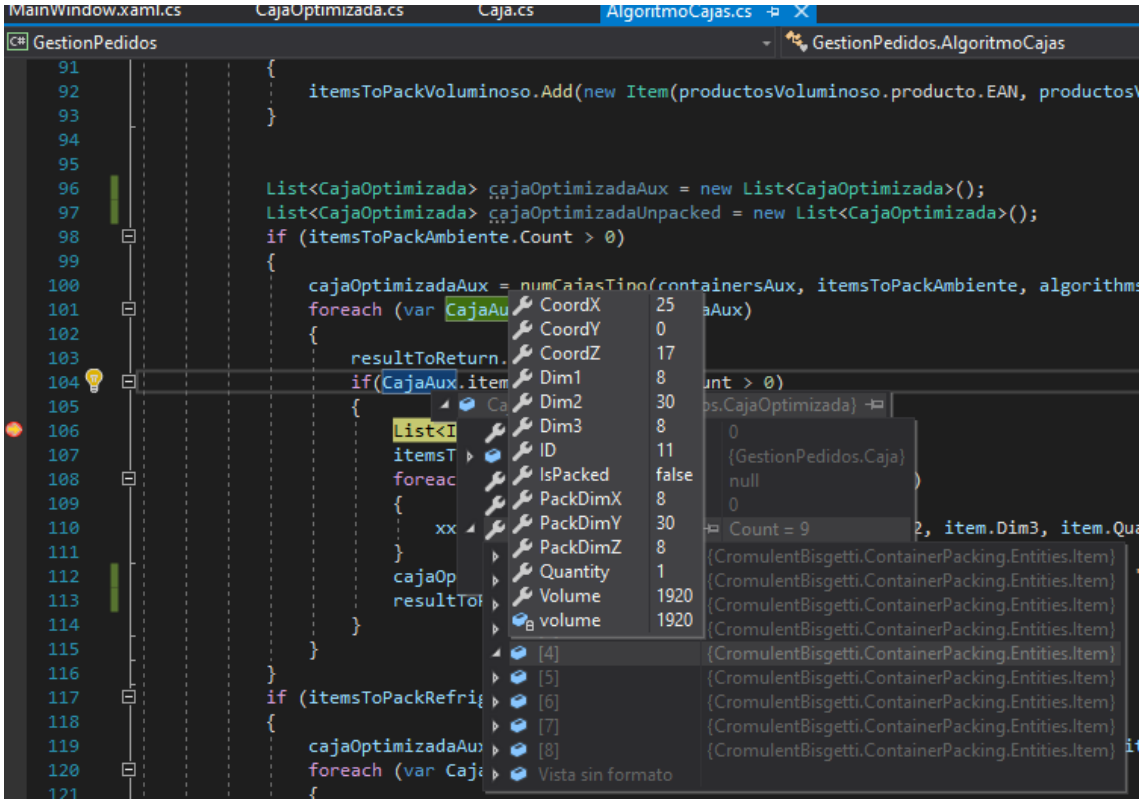


Ilustración 24: Ejemplo intento colocación producto

En la **Ilustración 24** podemos observar la información que proporciona el algoritmo sobre unos de los items que no han podido ser introducidos. Como vemos, tenemos disponible las coordenadas del espacio libre con más volúmen en la caja. Como en ese espacio no cabe el producto sobre el que estamos trabajanado, es incluido en la lista de productos no incluidos en esta caja.

Tras la implementación de este algoritmo podemos concluir que los objetivos se han cumplido, ya que garantizamos que los problemas de salidas de carros con cajas de menos, provocando la obligatoriedad de los recolectores y del personal de almacén de generar un carro nuevo se han resuelto.

Además, minizamos los problemas de sobre-estimación en el número de cajas necesarias por pedido, lo que impacta en la generación de carros, pudiendo meter más pedidos por carro, minimizando las rutas de salida de los recolectores.

5.3 Resultados algoritmo creación de rutas

Antes de analizar la forma en que se calcula la ruta, hemos de recordar que en caso de que un elemento se encuentre tanto en Tienda Abierta como en Tienda Cerrada, sólo marcaremos en la lista la localización en Tienda Cerrada, ya que es más probable que esté el producto en stock, y para limitar al mínimo el tiempo que pasa el recogedor en Tienda Abierta, reduciendo problemas de congestión.

En la **Ilustración 25** tenemos una representación cómo están distribuidos los productos de los pedidos de prueba a lo largo del almacén. Recordamos que el área comprendida entre el **Área preparación de carros** y **la Entrada** es lo que consideramos la Tienda Cerrada y la parte derecha se conoce como Tienda Abierta. En todos los casos empezaremos las rutas desde el **Área preparación de carros**.

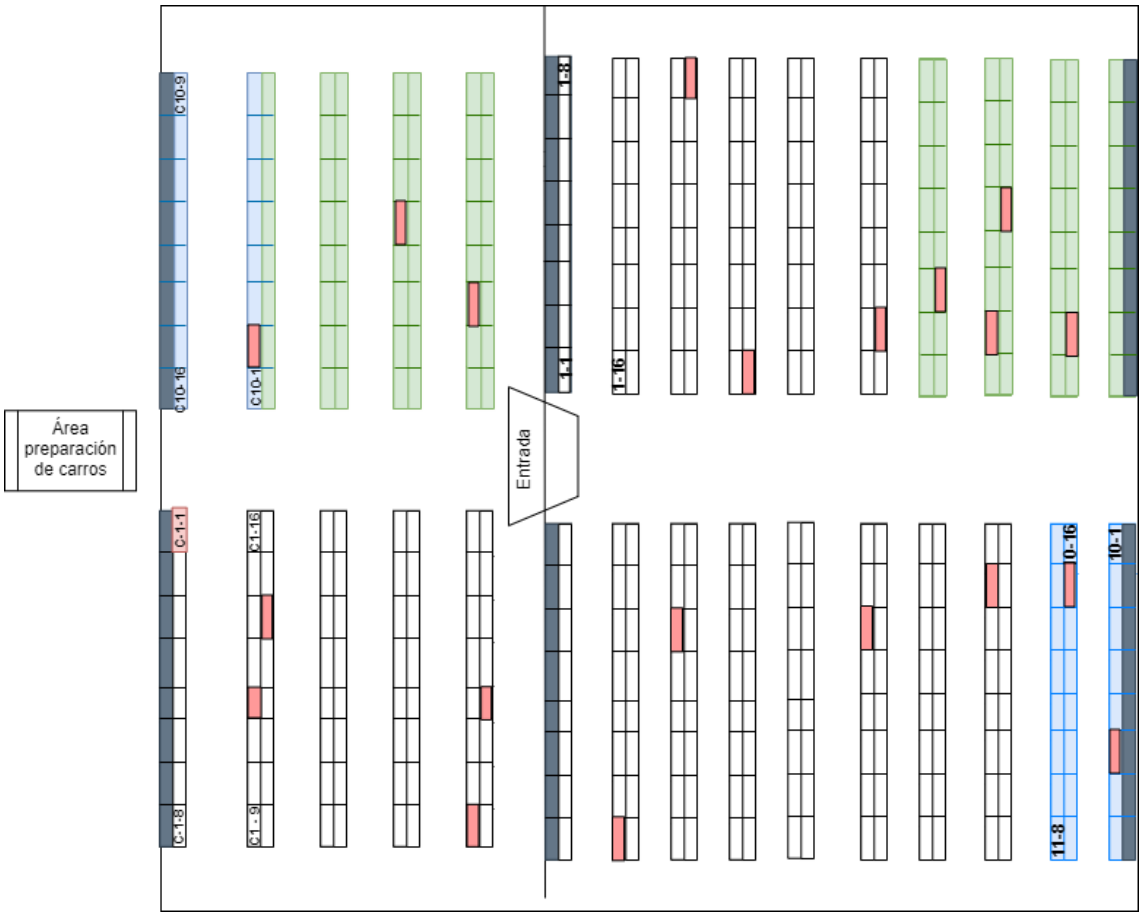


Ilustración 25: Mapa con los productos de los pedidos de prueba

Partimos de un punto en el que no se optimiza la ruta, por lo que el orden de elementos que mostramos para la recogida se basa en el orden de los elementos según se recuperan de la base de datos.

Como vemos en la **Ilustración 26**, la ruta viene caracterizada por una cantidad notable de productos localizados en Tienda Cerrada, como es esperado. No obstante el orden en el que se muestran no es muy óptimo. Al ir montando las localizaciones (también llamados segmentos) por orden de pedido, tenemos que se nos mezclan productos de Tienda Abierta y Tienda Cerrada.

Ruta sin optmizar	**C8-5; C5-5; C2-3; C10-2; C1-12; C1-1; 8-5; 7-3; 4-1; 10-6; C4-9; 7-15; 6-15; 3-8; 18-8; 14-3; 17-3; 12-2; 10-15; C7-3; 9-2;**

Ilustración 26: Ruta sin optimizar

Con estos datos, podemos sugerir una ruta, en la que comenzamos por el primer pasillo y nos desplazamos hasta el producto más cercano de ese mismo pasillo o de su simétrico con respecto al pasillo central, antes de cambiar de pasillo. Con este método se puede dar el caso que tengamos algún producto más cercano, pero evitamos tener que rotar el carro más veces al final de pasillo o incluso tener que recorrer un pasillo simétrico a uno ya recorrido. Con esta aproximación, la ruta a seguir es la que podemos ver en la **Ilustración 27**.

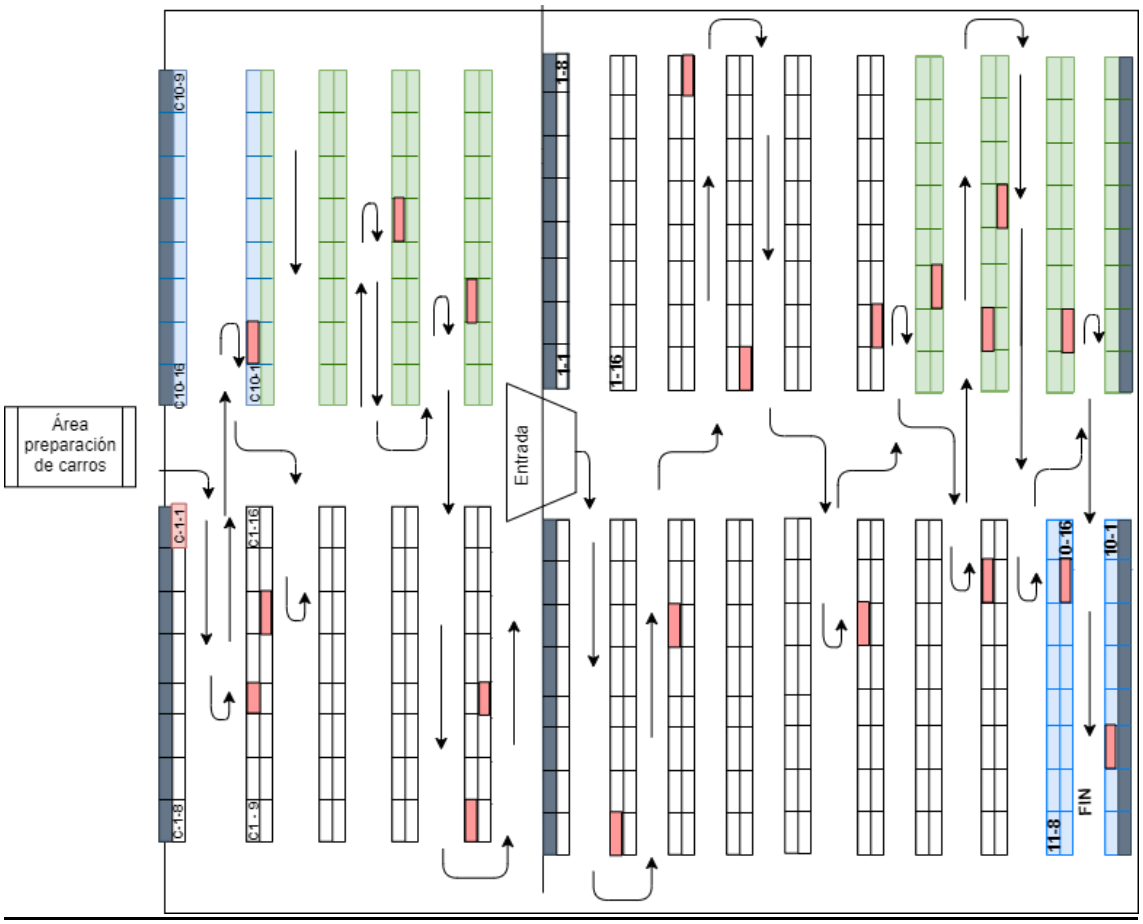


Ilustración 27: Ruta generada sin optimizar

Teniendo en cuenta que el ancho del pasillo central es de tres metros, que el ancho de los pasillos es de tres metros y que cada segmento mide un metro (la longitud del pasillo es de ocho metros, dividido en ocho segmentos) y que el almacén mide 45 metros de

largo, la ruta total asciende a 205 metros. El objetivo es reducir esta distancia lo máximo posible.

No obstante, para esta primera aproximación hemos infringido algunos preceptos básicos que no podemos trasladar a la realidad. La primera es que esta ruta la hemos hecho sabiendo la secuencia completa de segmentos a los que tenemos que acudir y dejando a criterio del operador la ruta que quiere coger siguiendo unos preceptos muy vagos. Como hemos comentado anteriormente, a la persona encargado de recoger el pedido se le debe mostrar sólo el siguiente pedido a recoger, para minimizar la toma de decisiones del recogedor y para estandarizar una forma de trabajo, simplificando la carga de trabajo de los recolectores.

Por lo tanto, si bien la aproximación anterior puede ser válida, no es aceptable bajo los criterios que define el presente trabajo.

Para acercarnos a los postulados expuestos y cumplir los requisitos definidos, vamos a minimizar la toma de decisiones en tema de ruta del recolector, y le vamos a proporcionar una ruta automática que pueda seguir siguiendo simplemente siguiendo el orden indicado. En una primera aproximación vamos a implementar un algoritmo muy sencillo. Vamos a agrupar todos los productos de todos los pedidos que sean de Tienda Cerrada entre sí, y posteriormente haremos lo mismo con los de Tienda Abierta. Una vez agrupados los productos por área, ordenaremos los pedidos por orden de pasillo ascendente. Esta aproximación si bien puede parecer anti intuitiva, se usa en varios algoritmos que hemos estudiado en puntos anteriores, como son los de Espacio más largo o en S.

Ruta optimizada	**C1-12; C1-1; C2-3; C4-9; C5-5; C7-3; C8-5; C10-2; 3-8; 4-1; 6-15; 7-3; 7-15; 8-5; 9-2; 10-6; 10-15; 12-2; 14-3; 17-3; 18-8;**

Ilustración 28: Ruta generada primera optimización

Este set de datos es el que vamos a usar a partir de ahora para las optimizaciones de las rutas de recogida.

Para generar la ruta que obtuvimos en la **Ilustración 28** hemos seguido el precepto de que el recogedor sólo tiene acceso a la localización del siguiente elemento a recoger. Antes de analizarla, tenemos que puntualizar que, por la forma en que hemos generado el set de datos, seguramente el recolector seguirá una ruta en la que recorrerá un sub-sector del cada tienda (inferior o superior respecto del pasillo central) antes de cambiar al contrario, si bien no podemos descartar que en algún momento se cambie antes del final de la tienda de sub-sector.

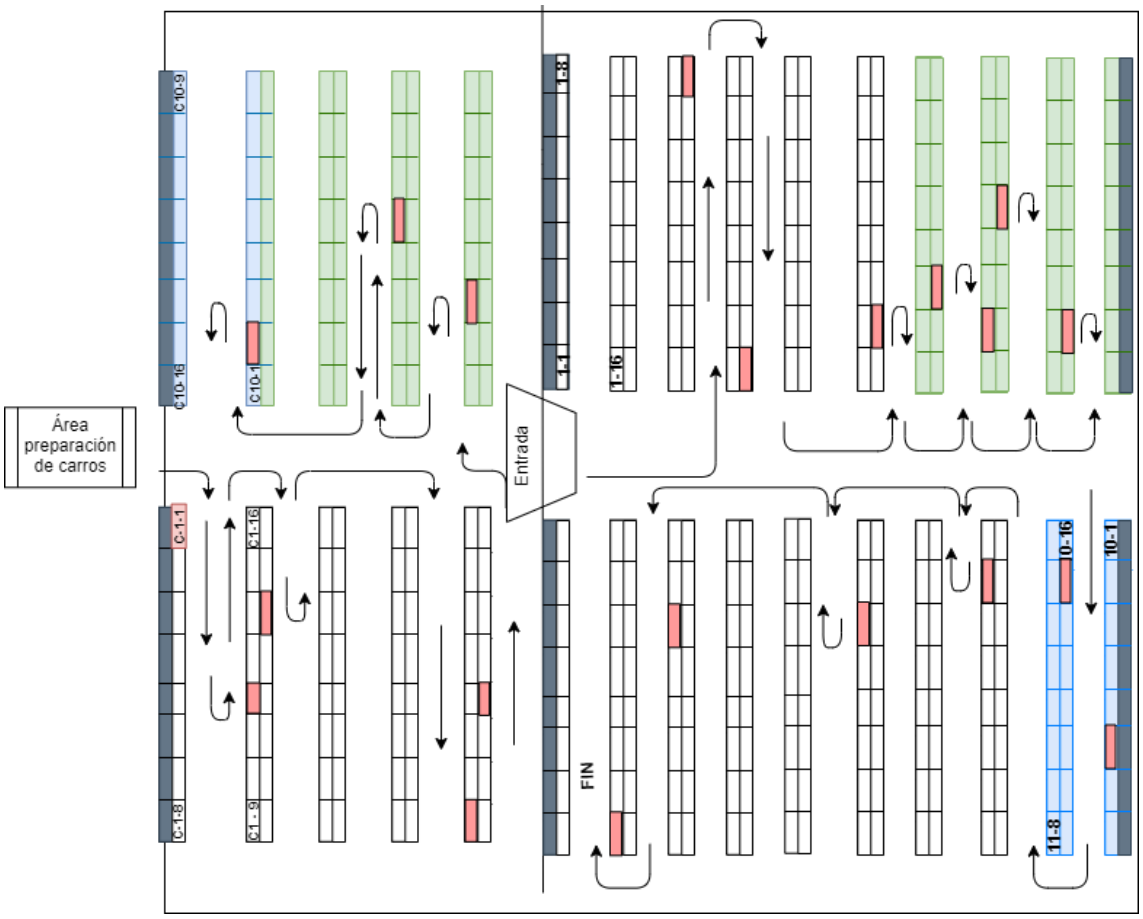


Ilustración 29: Ruta primera optimización

Tras esta primera optimización vemos (**Ilustración 29**) que la distancia total es de 237 metros. Es ligeramente superior a la distancia recorrida en la ruta anterior, pero reduciendo enormemente la toma de decisión del recolector.

Tras esta primera aproximación en la que hemos dejado sentadas las bases sobre las que vamos a trabajar, pasamos a implementar un algoritmo específico de rutas. En este caso hemos optado por la solución conocida como Óptima (descrita en el punto 3.2.5 del presente documento).

Para ello ejecutamos el algoritmo que hemos extraído de (Roodbergen, s.f.). En concreto hemos extraído el algoritmo Óptimo, como hemos comentado anteriormente. Tras esta ejecución, tenemos una longitud de ruta de 195 metros, lo que supones una mejora aproximada de un 5% respecto a la mejor ruta que teníamos anteriormente. La ruta obtenida la podemos analizar en la **Ilustración 30**.

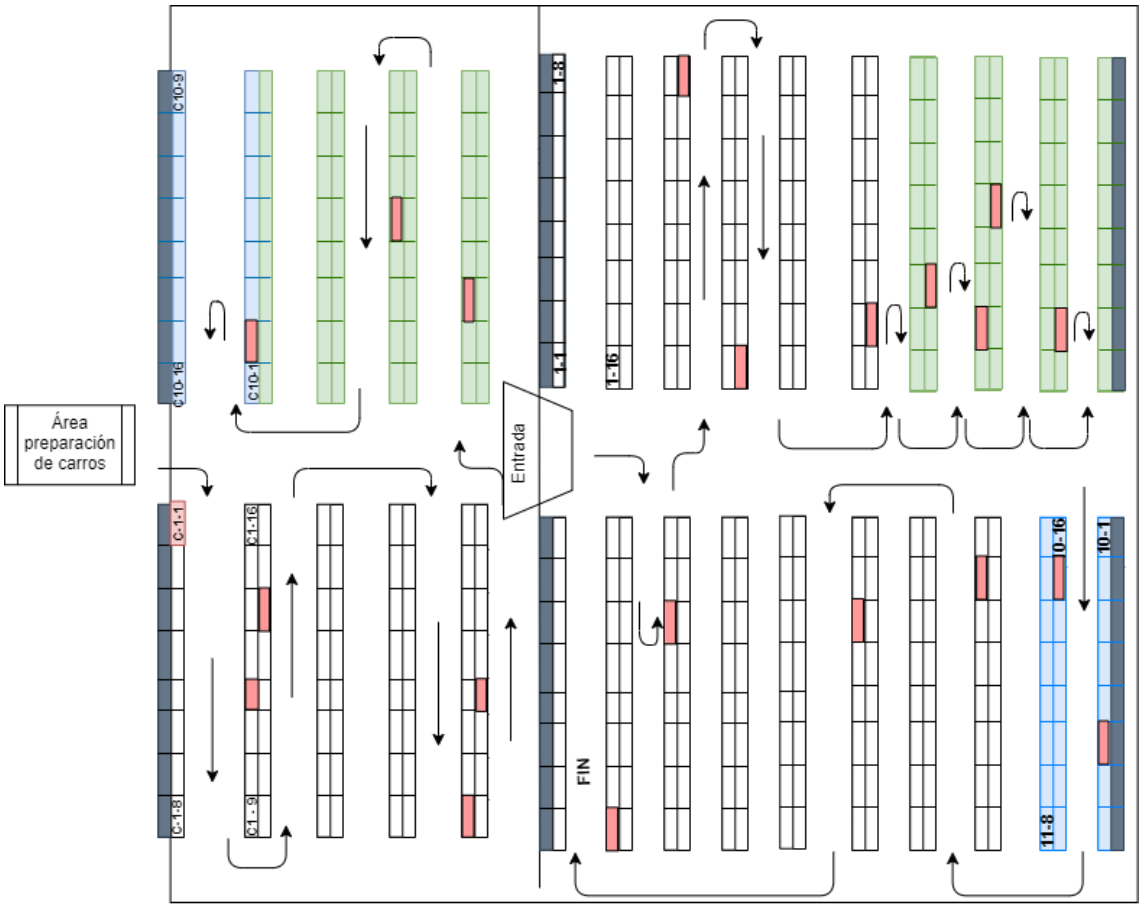


Ilustración 30: Ruta tras algoritmo Óptimo

Una mejora de la productividad de un 5% es algo realmente apreciable. No obstante, se espera una mejora aún mayor en cuanto los pedidos contengan un mayor número de elementos y en almacenes con un mayor número de pasillos.

También podemos inferir que el algoritmo arrojaría mejores resultados si podemos establecer una ruta de finalización diferente a la de inicio. No obstante, esa práctica la dejamos para futuras mejoras, ya que en sí mismo supondría un nuevo sprint.

Tras estas primeras pruebas, se ejecutó una batería de pruebas en la que se analizaron pedidos generados simulando una semana de trabajo. Estos pedidos se han generado manualmente intentando abarcar el abanico más amplio de casuísticas, manteniéndose siempre en pedidos que puedan ser considerados reales. En la **Tabla 6** tenemos la representación de estos datos. En ella se muestra la información de la longitud en metros de los recorridos que tiene que realizar el personal encargado de la recolección antes de aplicar algoritmos de optimización, tras aplicar el algoritmo de agrupación que diseñamos nosotros y por último la longitud tras aplicar el algoritmo Óptimo, que hemos analizado anteriormente.

Fecha	Longitud ruta previa optimización	Algoritmo diseñado por nosotros	Algoritmo Óptimo
31/12/2020	155 metros	151 metros	147 metros
01/01/2021	112 metros	106 metros	99 metros
02/01/2021	195 metros	202 metros	183 metros
03/01/2021	205 metros	237 metros	195 metros
04/01/2021	172 metros	166 metros	158 metros
05/01/2021	184 metros	178 metros	175 metros
06/01/2021	192 metros	175 metros	168 metros

Tabla 6: Resultado algoritmos optimización rutas

Capítulo 6

Conclusiones y Trabajo Futuro

En este último capítulo se presentan las conclusiones recabadas tras la terminación de este proyecto. Además se introducen ideas que pueden servir como trabajos futuros.

6.1 Conclusiones

En el presente trabajo se ha abordado un problema real y actual, de gestión de pedidos en un almacén, así como de tratamiento de las rutas que han de realizar los encargados de recoger esos pedidos por el almacén.

Este software proporciona una herramienta a empresas que tengan este modelo de negocio que impacta directamente en la productividad de los empleados de la misma, reduciendo las horas/persona que necesita un pedido en ser recogido y preparado para el envío. Para ello se han trabajado sobre dos puntos:

- Primero se ha optimizado el número de cajas que se necesita por pedido, minimizando para ello las salidas de carros con cajas que no se van a rellenar por no ser necesarias y evitando así que el trabajador encargado de la recogida tenga que volver al almacén a que le preparen un nuevo carro, debido a que hay un exceso productos que no han podido ser incluido en las cajas preparadas en primera instancia.
- Posteriormente, basándonos en la información generada en el paso anterior, conseguimos optimizar las rutas que deben ser seguidas por el personal. Para ello

trabajamos desde dos puntos de vista. Primero reducimos la toma de decisiones del trabajador, mostrándole el siguiente producto a recoger, y por otro lado, le ofrecemos una ruta pensada para que el recorrido sea el menor posible, consiguiendo reducir tanto su fatiga física como el tiempo que tarda en completar cada pedido.

Como hemos visto en algunos ejemplos, los algoritmos no siempre reducen el tiempo de recogida todo lo deseado, pero el tratar los problemas abordando múltiples aspectos, produce soluciones a varios niveles, que combinados producen grandes beneficios y que cubren los improbables escenarios en que un algoritmo presente un resultado subóptimo.

Una vez analizado el trabajo propuesto podemos concluir que cumple el objetivo general. El software entregado es completamente funcional y produce una solución fácilmente adaptable para el mercado. Como hitos más interesantes del proyecto podemos destacar:

- Se ha desarrollado una BBDD plenamente funcional, que se ha alimentado con productos y pedidos que podrían ser perfectamente extraídos de la realidad diaria de un almacén de tipo hipermercado.
- Se han proporcionado herramientas para trabajar sobre esos datos a través de métodos que hacen trasparente al desarrollo la estructura de la BBDD.
- Se ha proporcionado una interfaz gráfica que muestra la información recogida de la BBDD de forma clara y precisa a los usuarios de la herramienta. Se ha proporcionado una interfaz auxiliar que permite probar los cambios tanto de datos, como de algoritmia.
- Se integra un proyecto de generación de cajas siguiendo algoritmos basados en tres dimensiones licenciado bajo licencia MIT. Este software ha tenido que ser ampliamente modificado para adaptar los usos definidos por este software a los requisitos de nuestro proyecto. Creemos en el uso de software libre como una forma de mejorar tanto la calidad de nuestros proyectos como los de otros desarrolladores, creando una inercia que permita agilizar cada vez más los desarrollos. Por ello este proyecto se puede encontrar en https://github.com/EduPastor/rutasAlmacenes bajo licencia MIT.

- Se crea un módulo que permite mejorar las rutas a seguir por los empleados del almacén. Como nota diferencial, se integra el concepto de doble tienda (Tienda Abierta + Tienda Cerrada). Este concepto por sí mismo implica mejoras en la productividad de los propios trabajos de recogida (a costa de necesitar más espacio de almacenaje) y permite que los algoritmos de recogida incrementan su eficacia.

En gran parte de estos ejemplos hemos analizado un problema o solución con una perspectiva multidimensional ha sido la guía principal de todo el proyecto. En algunos puntos nos hemos permitido la licencia de apartarnos de una visión puramente académica para añadir una perspectiva empresarial (como reducir la toma de decisiones innecesarias de los empleados). Recomendamos a todos los lectores de este humilde documento una última lectura (Bono, 1970), que puede parecer ajena al tema que estamos tratando, pero que consideramos que debería ser la lectura base de todos los proyectos de este tipo, sobre todo en esta rama de la ciencia, ya que el principal problema al que se enfrentan los alumnos cuando terminan un TFG no es la falta de habilidades técnicas, si no escapar de la verticalidad del sistema.

6.2 Competencias adquiridas

El desarrollo de este proyecto de fin de grado ha ayudado a consolidar competencias adquiridas durante mis estudios [CO19], como aquellas que he ido trabajando en mi trayectoria profesional, tales como la gestión de un proyecto desde cero [BA5], recopilando todos los requisitos necesarios, traduciéndolos en necesidades y objetivos, y traduciéndolos estos en un producto software [IS1], apoyándonos en una metodología ágil estudiada durante la carrera, y que he necesitado utilizar en el ámbito laboral. Este producto se ha desarrollado con software con licencias gratuitas, lo que facilita futuros trabajos partiendo de esta base [IS3].

También hemos sido capaces de identificar hasta qué profundidad podía llegar un desarrollo que se adapte a las necesidades de un TFG y a un posible producto de mercado [IS2], evitando desarrollar módulos que no aportaban elementos interesantes en las vertientes académica o de producto [IS5], [CO6]. Además, he conseguido nuevas competencias gracias al apoyo de mis tutores. En concreto he sido capaz de afrontar la

modificación e implementación de algoritmos complejo [IS4], gracias al apoyo de mi tutor, Francisco Parreño López y he diseñado, implementado y alimentado una BBDD compleja desde 0 [BA4], siguiendo los consejos de mi otro tutor, Víctor Manuel López Jaquero. Por último, destacar que el proyecto se ha apoyado [CO7] en desarrollos con licencia MIT y sin licencia de software libre [IS6]. En todos los casos hemos respetado estas licencias y hemos liberado este proyecto para que la comunidad lo utilice si lo estima necesario [CO1].

6.3 Trabajo futuro

Lamentablemente no todas las mejoras estudiadas han podido ser implementadas. El trabajo de mejora a la hora de generar carros, si bien muy interesante desde el punto de vista teórico, no parece ofrecer grandes ventajas en pedidos pequeños. El ratio coste/beneficio de este desarrollo es muy bajo, y no se recomienda a afrontarlo a no ser que tengamos un volumen de pedidos muy alto. Incluso encontrándonos en esta situación (un volumen superior a los 50 pedidos por ronda de recogida) no parece que vaya a ser un elemento diferenciador, ya que un empleado encargado de la recolección puede manejar un número limitado de carros.

No obstante, cancelar un Sprint durante la fase de planificación del mismo, si bien no es óptimo ni recomendable, ya que implica que la toma de requisitos y el cálculo del esfuerzo y el valor añadido esperado han fallado, siempre es mejor que realizar un sprint que no va a ser útil. Mejor perder un día que veintiuno.

Además, a lo largo del estudio del problema aquí presentado se han encontrado casos de uso o mejoras que tienen el potencial para convertirse en mejoras del trabajo aquí expuesto. Hemos resumido las más interesantes en esta entrada con el ánimo de que en un futuro puedan ser estudiadas e implementadas.

- Añadir la opción de que el punto terminal de las rutas sea diferente al inicial, lo que puede impactar directamente en las rutas de la Tienda Cerrada, ya que los algoritmos podrían obtener mejores resultados si dejamos al recolector en la entrada a la Tienda Abierta, en lugar de en la entrada al almacén.
- Tener en cuenta también el peso de los artículos a la hora de generar las cajas, introduciendo puntos de paso prioritarios en los recorridos (lo que más pesa se recoge antes).

- Limitar el peso incluido en las cajas a no más de los 15 kilos que contempla la legislación actual en materia de protección de los trabajadores.
- Estudiar si sale a cuenta modificar las rutas atendiendo al orden de ordenación de cajas. Puede ser que añadir una dimensión al algoritmo de generación de cajas puede generar rutas menos óptimas y que el incremento en el tiempo de recorrido no compenso la reducción en el tiempo de empaquetado.
- Recalculo en "vivo" de las rutas si en el almacén no hay sustitutos adyacentes. Para ello hay que permitir al usuario la opción de comunicarse con el software, ya que actualmente es un software de sólo consulta.
- Reducir congestión alternando lateral de inicio de recogida en tiendas con pasillo central, o utilizando diferentes algoritmos de enrutamiento en almacenes "mono bloque".
- Añadir artículos a las cabeceras de pasillos. En muchos almacenes abiertos al público se aprovechan las cabeceras de los pasillos para mostrar producto al usuario. Estos artículos siguen estando accesibles en sus localizaciones habituales, pero se les habilita un espacio prioritario por visibilidad para aumentar sus ventas. Estudiar el impacto de añadir estos productos a la ruta de recogida, así como su impacto en el stock de los mismos y en problemas de congestión adicionales.
- Permitir al personal encargado de recoger los pedidos que pueda modificar el orden de recogida de productos en caso de congestión. Para ello el algoritmo debe recalcular las nuevas rutas atendiendo a los productos pendientes de recogida, sin recalcular en el orden ni el número de cajas.
- Estudiar subdividir los productos ambientes en productos de alimentación y productos de limpieza, para separar los envíos de alimentación de los potencialmente tóxicos.
- Estudiar si es óptimo dividir la recogida entre producto ambiente y voluminoso en un recorrido, y los frescos y congelados en otro, estudiando si esta división impacta positivamente en la productividad.

- Estudiar cómo afecta al algoritmo el modificar el ancho de algunos pasillos. En almacenes de tipo Tienda Abierta es común que algunos pasillos tengan características especiales de longitud o anchura.
- Introducir una generación propia de cajas/carros a pedidos con una cantidad de productos muy alta o con pocos productos diferentes pero en cantidades muy elevadas. Algunos clientes y algunas promociones realizan pedidos que tienen características especiales y que un tratamiento diferente puede ahorrar tiempos de recogida.

Capítulo 7

Bibliografía

AECOC. (2012). *AECOC.* Obtenido de https://www.aecoc.es: https://www.aecoc.es/recomendaciones/ral-unidades-de-carga-eficiente/

Bono, E. d. (1970). *Lateral Thinking - A Textbook of Creativity.* Penguin Books.

Borja Menéndez, E. G.-A. (2017). Variable Neighborhood Search strategies for the Order. *Computers & Operations Research*, 500-512.

Cohn, M. (2004). *User Stories Applied for Agile Software Development.* Addison Wesley.

davidmchapman. (2019). *Github*. Obtenido de https://github.com/davidmchapman/3DContainerPacking

Francia, J. (7 de 10 de 2020). *Scrum.org - The home of scrum*. Obtenido de Scrum.org - The home of scrum: https://www.scrum.org/resources/blog/que-es-scrum

HenrikI Christensen, A. K. (2017). Approximation and online algorithms for multidimensional bin packing: A survey. *Computer Science Review - 24*, 63-79.

Hompel, M. S. (2007). *Warehouse Managmenet - Automation.*

Johanna Bolaños Zuñiga, J. A. (2020). Optimization of the Storage Location Assignment and the Picker-Routing Problem by Using Mathematical Programming. *Applied sciences*.

Karasek, J. (2013). An Overview of Warehouse Optimization. *International Journal of Advances in Telecommunications Electrotechnics Signals and Systems*, 2-7.

Kees Jan Roodbergen, R. K. (2001). Routing methods for warehouses with multiple cross aisles. *International Journal of Production Research*, 1869-1883.

Ken Schwaber, J. S. (7 de 10 de 2020). *Scrum Guides*. Obtenido de Scrum Guides: https://www.scrumguides.org/scrum-guide.html

Klaviyo. (4 de Octubre de 2020). *Klaviyo*. Obtenido de Klaviyo: https://www.klaviyo.com/covid-19-ecommerce-marketing-poll

Krzysztof Fleszar, K. S. (2002). New heuristics for one-dimensional bin-packing. *Computers & Operations Research* , 821-839.

Ma, S. (2 de Enero de 2020). *Routific*. Obtenido de Routific: https://blog.routific.com/travelling-salesman-problem

Makusee Masae, C. H. (2020). Order picker routing in warehouses: A systematic literature review. *International Journal of Production Economics*.

Marek Matusiak, R. d. (2014). A fast simulated annealing method for batching precedence-constrained customer orders in a warehouse. *European Journal of Operational Research*, 968-977.

Microsoft. (s.f.). *https://visualstudio.microsoft.com/es/vs/community*.

Rehkopf, M. (2020). *Atlassian Agile Coach.* Obtenido de https://www.atlassian.com/es/agile/project-management/user-stories

René de Koster, T. L.-D. (2007). Design and control of warehouse order picking:. *European Journal of Operational Research*, 481-501.

Roodbergen. (s.f.). *Interactive Warehouse*. Obtenido de http://www.roodbergen.com/warehouse/frames.htm?demo

Varios. (7 de 10 de 2020). *Scrum.org - The home of Scrum*. Obtenido de Scrum.org - The home of Scrum: https://www.scrum.org/forum/scrum-forum/5887/scrum-master-member-development-team

Anexo I. Artefactos Scrum

En esta sección añadimos los artefactos generados en todos los Sprint, así como el Product Backlog, y sus diferentes actualizaciones.

En las subsecciones principales de los Sprint se añadirá una actualización del Product Backlog con las Historias terminadas y un Sprint Backlog con las tareas realizadas en el Sprint anterior y las que se han priorizado para el siguiente Sprint.

I.1 Product Backlog

	Historia	Estado	Tiempo Total Estimado (días)	Fecha inicio estimada	Dependencia	% Finalizado
A	Diseño BBDD	Sin Comenzar	2	05/10/2020		0%
B	Implementación BBDD	Sin Comenzar	2	07/10/2020	A	0%
C	Interfaz pruebas BBDD	Sin Comenzar	3	9/10/2020	B	0%
D	Interfaz usuario básica	Sin Comenzar	4	12/10/2020		0%
E	Implementación algoritmo reparto en cajas	Sin Comenzar	7	16/10/2020	C, D	0%
F	Interfaz usuario definitiva	Sin Comenzar	4	23/10/2020	D	
G	Implementación algoritmo creación rutas	Sin Comenzar	8	26/10/2020	E	0%
H	Implementación algoritmo carros	Sin Comenzar	10	30/10/2020	G	0%
I	Análisis de resultados	Sin Comenzar	1	9/11/2020	H	0%
J	Optimización algoritmos	Sin Comenzar	2	10/11/2020	I	0%
K	Análisis Optimización	Sin Comenzar	1	12/11/2020	J	0%

Tabla 7: Product Backlog Inicial – Historias

Tarea ID	Historia ID	Tarea	Estimación (días)	Dependencia	Prioridad	Estado (%)	Condición de aprobación	Aprobado
1	**A**	Definir cualidades productos	1		**100**	**0%**	Se deben dejar clara la información que hay que almacenar de los productos	**No OK**
2	**A**	Definir relaciones entre las tablas	1	1	**100**	**0%**	Los diagramas presentados deben adecuarse a la solución	**No OK**
3	**B**	Creación tablas	1	1,2	**100**	**0%**	Las tablas deben estar implementadas	**No OK**
4	**B**	Carga de datos	1	3	**50**	**0%**	Debe haber una cantidad suficiente de datos para poder ejecutar pruebas	**No OK**
5	**C**	Queries de selección de datos	3	4	**80**	**0%**	Las consultas deben recuperar los datos esperados	**No OK**
6	**D**	Inicialización del proyecto	0.5		**100**	**0%**	Se debe presentar la arquitectura del software	**No OK**
7	**D**	Conexión con la BBDD	0.5	6	**50**	**0%**	El entorno de desarrollo debe recuperar información de la BBDD	**No OK**
8	**D**	Interfaz básica	1		**60**	**0%**	El entorno gráfico presenta un diseño mínimo pero funcional	**No OK**
9	**E**	Implementación algoritmo reparto en cajas	2		**100**	**0%**	Se entrega una versión funcional del cálculo de cajas	**No OK**
10	**F**	Diseño final	4	8	**20**	**0%**	Se entrega la versión final de la interfaz gráfica	**No OK**
11	**G**	Implementación algoritmo creación de rutas	8	9	**100**	**0%**	Se entrega una versión funcional del cálculo de carros	**No OK**
12	**G**	Definición plan de pruebas	1		**50**	**0%**	Se entrega un plan de pruebas que debe cumplir la entrega final	**No OK**
13	**H**	Implementación algoritmo	9	14,15	**100**	**0%**	Se entrega la versión final del producto	**No OK**

		creación de carros						
14	**I**	Persistir datos	1	16	**80**	**0%**	Los datos obtenidos son usables para las siguientes iteraciones	**No OK**
15	**J**	Implementar mejoras en algoritmos	2	16,17	**100**	**0%**	Documento con las pruebas a seguir y los resultados esperados	**No OK**
16	**K**	Persistir datos	1	17	**80**	**0%**	Los datos obtenidos son comparados con los obtenidos previamente	**No OK**

Tabla 8: Product Backlog Inicial – Tareas

I.2 Sprint Backlog – Sprint 1

El Product Backlog en este punto es el mismo que hemos definido en el punto anterior debido a que no ha habido actuaciones sobre el Proyecto.

Tarea ID	Historia ID	Tarea	Estimación (días)	Dependencia	Prioridad	Estado (%)	Condición de aprobación	Aprobado
1	**A**	Definir cualidades productos	1		**100**	**0%**	Se deben dejar clara la información que hay que almacenar de los productos	**No OK**
2	**A**	Definir relaciones entre las tablas	1	1	**100**	**0%**	Los diagramas presentados deben adecuarse a la solución	**No OK**
3	**B**	Creación tablas	1	1,2	**100**	**0%**	Las tablas deben estar implementadas	**No OK**
4	**B**	Carga de datos	1	3	**50**	**0%**	Debe haber una cantidad suficiente de datos para poder ejecutar pruebas	**No OK**
5	**C**	Queries de selección de datos	3	4	**80**	**0%**	Las consultas deben recuperar los datos esperados	**No OK**
6	**D**	Inicialización del proyecto	0.5		**100**	**0%**	Se debe presentar la arquitectura del software	**No OK**
7	**D**	Conexión con la BBDD	0.5	6	**50**	**0%**	El entorno de desarrollo debe recuperar información de la BBDD	**No OK**
8	**D**	Interfaz básica	1		**60**	**0%**	El entorno gráfico presenta un diseño mínimo pero funcional	**No OK**

Tabla 9: Sprint Backlog - Sprint 1

I.3 Artefactos generados

Tras este Sprint se liberan dos artefactos para apoyar el desarrollo del equipo. Son los diagramas E/R y lógico usados para la implementación de la primera fase de la BBDD y que será obligatorio seguir en los desarrollos de los Sprint siguientes. En la **Ilustración 31** tenemos un diagrama E/R con las entidades básicas que compondrán la BBDD.

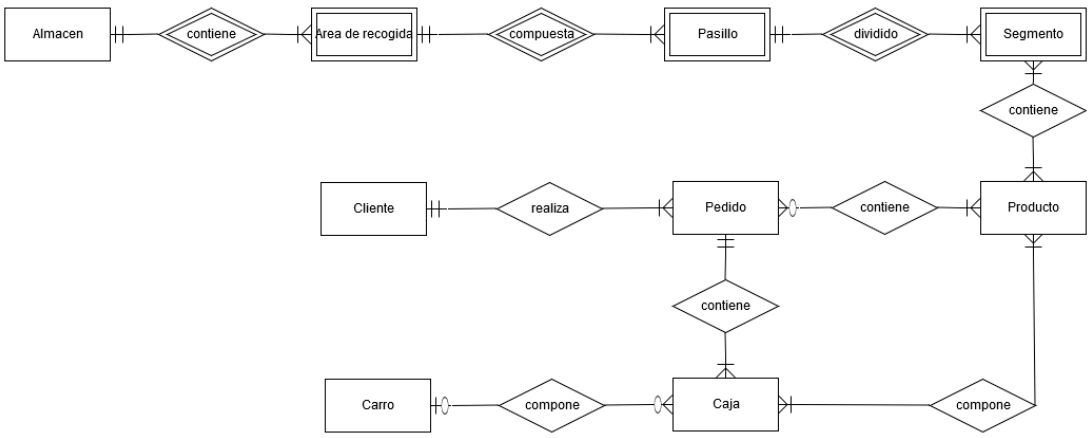


Ilustración 31: Diagrama E/R BBDD almacenes

Por su parte, en la **Ilustración 32** tenemos el diagrama lógico de la BBDD, en el que se detalla más en profundidad las relaciones entre las entidades previamente definidas y se definen las propiedades de las mismas.

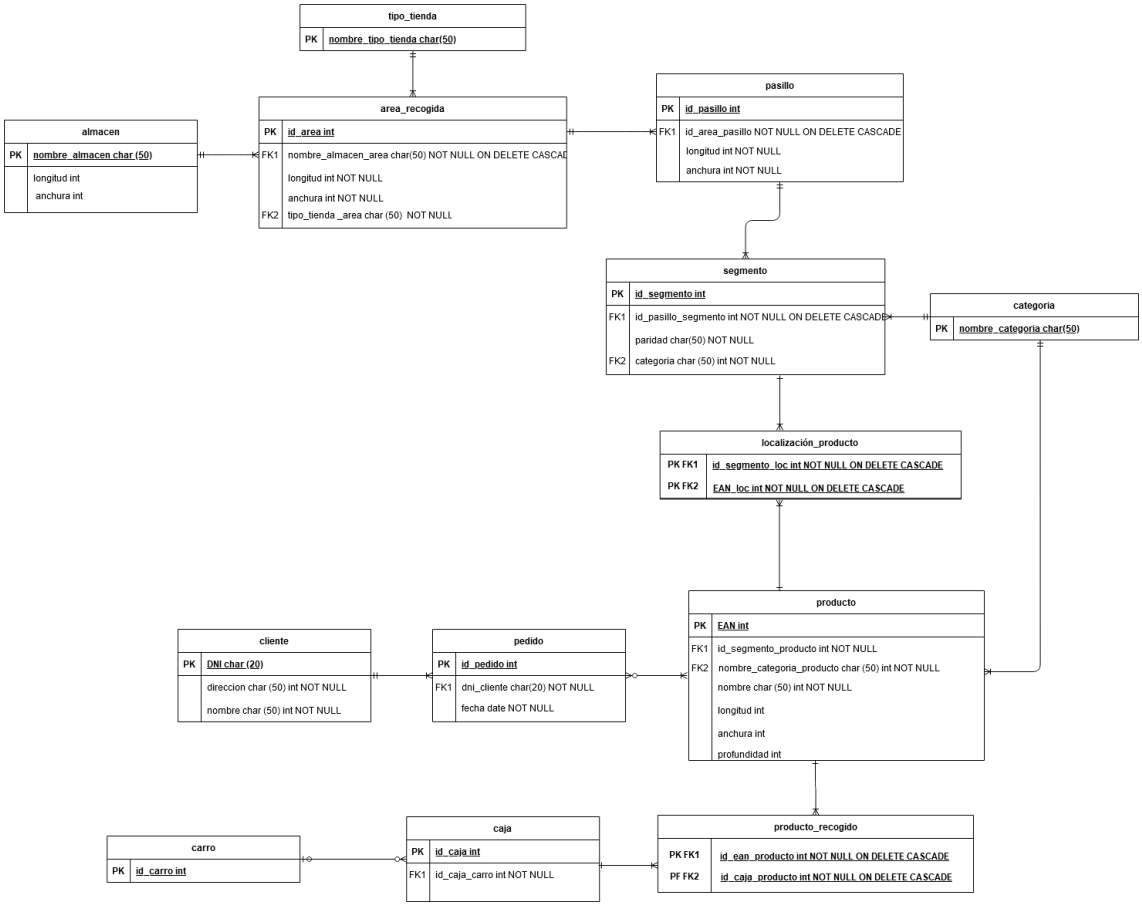


Ilustración 32: Diagrama lógico de BBDD

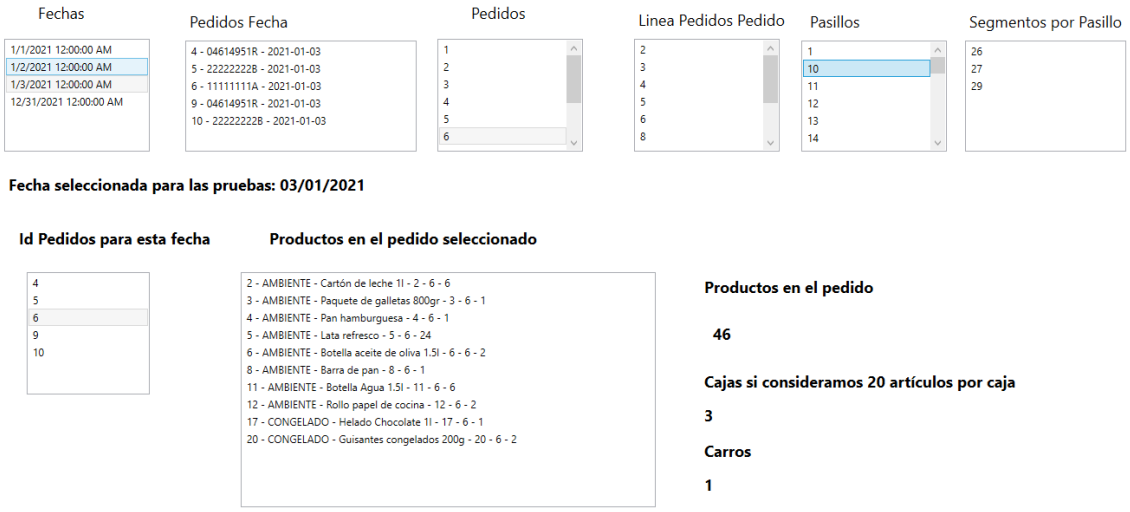


Ilustración 33: Maqueta de apoyo al desarrollo

En la **Ilustración 33** vemos una pantalla de apoyo que se creó para mostrar visualmente las peticiones a la BBDD, y que podría servir como una primera propuesta sobre la que desarrollar el aspecto visual de la aplicación definitiva. En ella podemos encontrar diferentes consultas que son necesarias para alimentar los algoritmos que se van a implantar tanto en este primer sprint como en los siguientes.

Algunas de las consultas usadas en las pantallas anteriores pueden verse a continuación. La mayoría de ellas serán usadas por los algoritmos finales.

I.4 Problemas encontrados tras el primer Sprint

Tras este primer Sprint se detecta que en el diseño de la BBDD no incluye el campo CP en la tabla **cliente.** En el siguiente Sprint debe ser subsanado.

Igualmente, se detectan errores en la nomenclatura en los diagramas de algunos campos en tablas de la BBDD. Se corrigen y se añade el esquema final de la BBDD como producto de consulta para apoyo a los siguientes desarrollos.

I.5 Product Backlog y Sprint Backlog – Sprint 2

	Historia	Estado	Tiempo Total Estimado (días)	Fecha inicio estimada	Dependencia	% Finalizado
A	Diseño BBDD	Sin Comenzar	2	05/10/2020		100%
B	Implementación BBDD	Sin Comenzar	2	07/10/2020	A	100%
C	Interfaz pruebas BBDD	Sin Comenzar	3	9/10/2020	B	100%
D	Interfaz usuario básica	Sin Comenzar	4	12/10/2020		100%
E	Implementación algoritmo reparto en cajas	Sin Comenzar	7	16/10/2020	C, D	100%
F	Interfaz usuario definitiva	Sin Comenzar	4	23/10/2020	D	50%
G	Implementación algoritmo creación rutas	Sin Comenzar	8	26/10/2020	E	0%
H	Implementación algoritmo carros	Sin Comenzar	10	30/10/2020	G	0%
I	Análisis de resultados	Sin Comenzar	1	9/11/2020	H	0%
J	Optimización algoritmos	Sin Comenzar	2	10/11/2020	I	0%
K	Análisis Optimización	Sin Comenzar	1	12/11/2020	J	0%

Tabla 10: Product Backlog tras Sprint 1

Tarea ID	Historia ID	Tarea	Estimación (días)	Dependencia	Prioridad	Estado (%)	Condición de aprobación	Aprobado
1	**A**	Definir cualidades pasillos y productos	2		**100**	**100%**	Se deben dejar claros el número de pasillos así como su numeración, disposición y todos lo dates que hay que almacenar de los productos.	**OK**
2	**A**	Definir relaciones entre las tablas	2	1	**100**	**100%**	Los diagramas presentados deben adecuarse a la solución.	**OK**
3	**B**	Creación tablas	4	1,2	**80**	**100%**	Las tablas deben estar implementadas.	**OK**
4	**B**	Carga de datos	1	3	**50**	**100%**	Debe haber una cantidad suficiente de datos para poder ejecutar pruebas.	**OK**
5	**C**	Queries de selección de datos	3	4	**80**	**100%**	Las consultas deben recuperar los datos esperados.	**OK**
6	**D**	Inicialización del proyecto	0.5		**100**	**100%**	Se debe presentar la arquitectura del software	**OK**
7	**D**	Conexión con la BBDD	0.5	6	**50**	**100%**	El entorno de desarrollo debe recuperar información de la BBDD	**OK**
8	**D**	Interfaz básica	1		**60**	**100%**	El entorno gráfico presenta un diseño mínimo pero funcional	**OK**
9	**E**	Implementación algoritmo reparto en cajas	2		**100**	**0%**	Se entrega una versión funcional del cálculo de cajas	**No OK**
10	**F**	Diseño final	4	8	**20**	**0%**	Se entrega la versión final de la interfaz gráfica	**No OK**
11	**G**	Implementación algoritmo creación de rutas	8	9	**100**	**0%**	Se entrega una versión funcional del cálculo de carros	**No OK**
12	**G**	Definición plan de pruebas	1		**50**	**0%**	Se entrega un plan de pruebas que	**No OK**

							debe cumplir la entrega final	
13	**H**	Implementación algoritmo creación de carros	9	14,15	**100**	**0%**	Se entrega la versión final del producto	**No OK**
14	**I**	Persistir datos	1	16	**80**	**0%**	Los datos obtenidos son usables para las siguientes iteraciones	**No OK**
15	**J**	Implementar mejoras en algoritmos	2	16,17	**100**	**0%**	Documento con las pruebas a seguir y los resultados esperados	**No OK**
16	**K**	Persistir datos	1	17	**80**	**0%**	Los datos obtenidos son comparados con los obtenidos previamente	**No OK**

Tabla 11: Sprint Backlog - Sprint 2

I.6 Artefactos generados tras el Sprint 2

Como comentamos, tras el primer Sprint se detectaron errores en la BBDD que fueron subsanados en este ciclo. Se presenta a continuación (**Ilustración 34**) el diagrama final de la BBDD como referencia para futuros desarrollos.

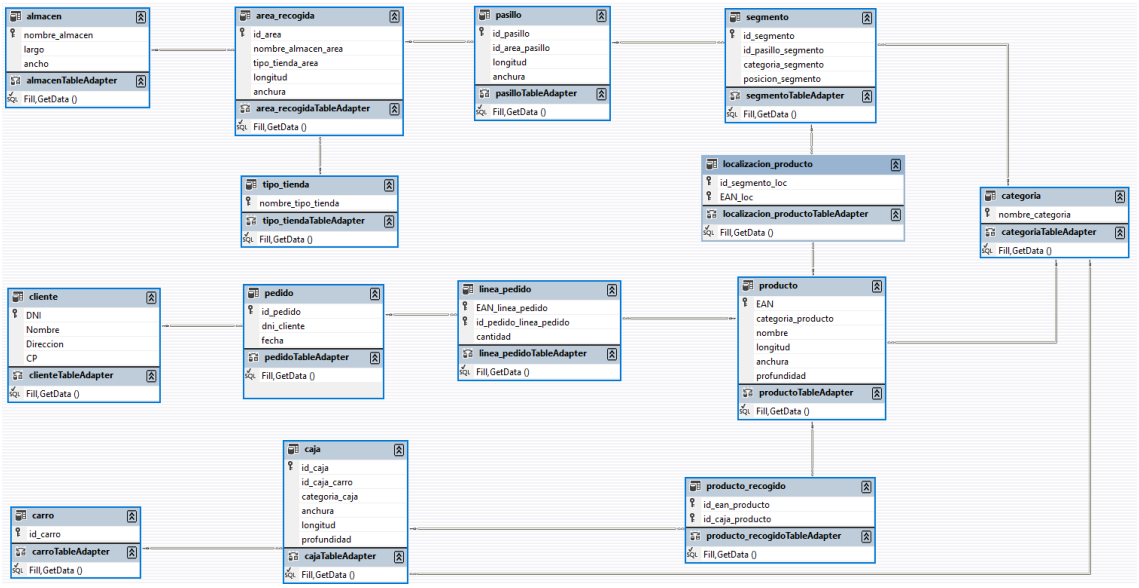


Ilustración 34: Diagrama final de la BBDD

I.7 Product Backlog y Sprint Backlog – Sprint 3

	Historia	Estado	Tiempo Total Estimado (días)	Fecha inicio estimada	Dependencia	% Finalizado
A	Diseño BBDD	Sin Comenzar	2	05/10/2020		100%
B	Implementación BBDD	Sin Comenzar	2	07/10/2020	A	100%
C	Interfaz pruebas BBDD	Sin Comenzar	3	9/10/2020	B	100%
D	Interfaz usuario básica	Sin Comenzar	4	12/10/2020		100%
E	Implementación algoritmo reparto en cajas	Sin Comenzar	7	16/10/2020	C, D	100%
F	Interfaz usuario definitiva	Sin Comenzar	4	23/10/2020	D	100%
G	Implementación algoritmo creación rutas	Sin Comenzar	8	26/10/2020	E	100%
H	Implementación algoritmo carros	Sin Comenzar	10	30/10/2020	G	0%
I	Análisis de resultados	Sin Comenzar	1	9/11/2020	H	0%
J	Optimización algoritmos	Sin Comenzar	2	10/11/2020	I	0%
K	Análisis Optimización	Sin Comenzar	1	12/11/2020	J	0%

Tabla 12: Product Backlog tras Sprint 2

Tarea ID	Historia ID	Tarea	Estimación (días)	Dependencia	Prioridad	Estado (%)	Condición de aprobación	Aprobado
1	**A**	Definir cualidades productos	1		**100**	**100%**	Se deben dejar clara la información que hay que almacenar de los productos	**OK**
2	**A**	Definir relaciones entre las tablas	1	1	**100**	**100%**	Los diagramas presentados deben adecuarse a la solución	**OK**
3	**B**	Creación tablas	1	1,2	**100**	**100%**	Las tablas deben estar implementadas	**OK**
4	**B**	Carga de datos	1	3	**50**	**100%**	Debe haber una cantidad suficiente de datos para poder ejecutar pruebas	**OK**
5	**C**	Queries de selección de datos	3	4	**80**	**100%**	Las consultas deben recuperar los datos esperados	**OK**
6	**D**	Inicialización del proyecto	0.5		**100**	**100%**	Se debe presentar la arquitectura del software	**OK**
7	**D**	Conexión con la BBDD	0.5	6	**50**	**100%**	El entorno de desarrollo debe recuperar información de la BBDD	**OK**
8	**D**	Interfaz básica	1		**60**	**100%**	El entorno gráfico presenta un diseño mínimo pero funcional	**OK**
9	**E**	Implementación algoritmo reparto en cajas	2		**100**	**100%**	Se entrega una versión funcional del cálculo de cajas	**OK**
10	**F**	Diseño final	4	8	**20**	**0%**	Se entrega la versión final de la interfaz gráfica	**No OK**
11	**G**	Implementación algoritmo creación de rutas	8	9	**100**	**0%**	Se entrega una versión funcional del cálculo de carros	**No OK**
12	**G**	Definición plan de pruebas	1		**50**	**0%**	Se entrega un plan de pruebas que debe cumplir la entrega final	**No OK**
13	**H**	Implementación algoritmo creación de carros	9	14,15	**100**	**0%**	Se entrega la versión final del producto	**No OK**

14	**I**	Persistir datos	1	16	**80**	**0%**	Los datos obtenidos son usables para las siguientes iteraciones	**No OK**
15	**J**	Implementar mejoras en algoritmos	2	16,17	**100**	**0%**	Documento con las pruebas a seguir y los resultados esperados	**No OK**
16	**K**	Persistir datos	1	17	**80**	**0%**	Los datos obtenidos son comparados con los obtenidos previamente	**No OK**

Tabla 13: Sprint Backlog - Sprint 3

Este Sprint es cancelado debido a que no se considera que el valor añadido que provee sea superior a la inversión realizada, ni en el corto ni en el medio plazo.

I.8 Incremento Final

El incremento final, una copia de esta memoria, y material adicional se pueden encontrar en https://github.com/EduPastor/rutasAlmacenes bajo licencia MIT.