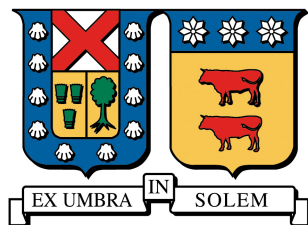


UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA
DEPARTAMENTO DE ELECTRÓNICA
SANTIAGO – CHILE



SISTEMA TECNOLÓGICO DE SIMULACIÓN PARA EL
PROCESO DE RECEPCIÓN DE UVA

MATÍAS YAMIL SOTO SEPÚLVEDA

MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL TELEMÁTICO

PROFESOR GUÍA
NICOLÁS SEBASTIÁN GÁLVEZ RAMÍREZ

PROFESOR CORREFERENTE
PATRICIO OLIVARES RONCAGLIOLO

MAYO 2025

Agradecimientos

A todas las personas mencionadas en este apartado les tengo un cariño inmenso y atesoro recuerdos increíbles con cada una de ellas, muchas gracias por estar en mi vida. Como todos saben no soy de muchas palabras, así que seré breve.

Gracias mi amor por estar siempre a mi lado, apoyarme en los momentos más difíciles de mi vida y recordarme de lo capaz que soy cada vez que dudaba mínimamente de mi mismo. Muchas noches de estudio fueron motivadas por la idea de construir un futuro contigo, te amo con toda mi vida. También agradecer a tu familia por quererme y recibirme desde el día uno.

A mis abuelos, gracias por acogerme en su hogar en el momento que más lo necesitaba, por darme lo que necesité sin pensarlo dos veces, son uno de los pilares fundamentales de todos mis logros y sin ustedes no habría llegado a donde estoy. Gracias a mis padres por enseñarme los valores fundamentales que hoy me representan como persona y profesional, por permitirme perseguir mis sueños y ser ejemplos de superación, esfuerzo y perseverancia. A mis hermanos, Agustín y Antonia, ustedes me motivan a mejorar día a día para ser un digno ejemplo a seguir. Gracias también a mis tíos, Italo y Daniela, por todo su apoyo, consejos y gratos momentos.

A compañeros de universidad y amigos, Cristobal, Vicente, Thomas, Tomás, Diego, Clemente, Chiappe, Giovanni y Bastian, muchas gracias por acompañarme en esta etapa, sobre todo en tiempos extraños de pandemia donde todo el contacto era remoto y conectarnos por Discord a tirar la talla era lo más cercano a salir del hogar que había en ese periodo.

Gracias a mis amigos del Antupiren, Vice, Titi, Cata, Chamorro, Fran, Keko, Marin, Nacho, Ruso, Sol y Wale, por hacerme reír y pasar buenos momentos cada vez que nos vemos. Estoy orgulloso del camino que cada uno tomó.

A mi profesor Nicolás, gracias por la paciencia, comprensión y oportunidades desde la primera ayudantía, espero que sus próximos alumnos no sean tan desordenados como yo.

Muchas gracias a todos por todo.

Dedicatoria

Para la persona que siempre puso todas sus fichas en mí.

Italo Yamil Sepúlveda Salech, mi Tata.

Resumen

En el presente trabajo se expone el diseño, desarrollo e implementación de una herramienta tecnológica de simulación orientada a la optimización del proceso de recepción de uva durante la vendimia, en el contexto operativo de Viña Errázuriz. Este proceso, fundamental en la cadena de producción vitivinícola, presenta desafíos asociados a la identificación y gestión de cuellos de botella, cuya existencia compromete tanto la eficiencia operativa como la calidad del producto final. Para afrontar esta situación se visualiza integralmente el proceso de recepción, con el propósito de mejorar su monitoreo, control y capacidad de análisis estratégico; cumpliendo los siguientes hitos:

- Caracterizar detalladamente las líneas de producción existentes en la planta, identificando su configuración técnica, maquinaria involucrada y principales variables operativas.
- Modelar cada línea de recepción, incorporando parámetros como velocidades de procesamiento, capacidades de acumulación y rendimientos, con el objetivo de simular su comportamiento bajo distintas condiciones.
- Desarrollar e implementar una herramienta digital que integra dicho modelo en una interfaz interactiva, permitiendo visualizar en tiempo real el flujo del proceso y evaluar escenarios operativos críticos

Esta herramienta fue validada mediante la comparación de resultados simulados con registros históricos, asegurando su fiabilidad y aplicabilidad en contextos reales. Los resultados alcanzados evidencian que la herramienta permite representar con precisión la dinámica del proceso de recepción, facilitando la identificación temprana de ineficiencias como acumulaciones excesivas o velocidades subóptimas en la maquinaria. Asimismo, entrega una base sólida para la toma de decisiones estratégicas relativas a la renovación de equipos, posibilitando estimar su impacto sin introducir nuevas restricciones operativas. La solución propuesta destaca además por su arquitectura modular y escalable, lo que la posiciona como una base tecnológica adaptable a futuros desarrollos y transferible a otras líneas o bodegas del sector.

Abstract

This work describes the design, development, and implementation of a technological simulation tool to optimize the grape reception process during harvest within the operational context of Viña Errázuriz. This critical process in the wine production chain challenges the identification and management of bottlenecks, compromising operational efficiency and the quality of the final product. To tackle this problem, a visualization platform of the reception process is implemented to improve the monitoring, control, and strategic analysis capabilities, completing the following checkpoints:

- Detailed characterization of production lines in the plant, identifying their technical configuration, machinery involved, and key operational variables.
- Reception line modeling, including parameters such as processing speeds, accumulation capacities, and yields, with the objective of simulating their behavior under different conditions.
- To develop and implement a digital tool integrating this model into an interactive interface, allowing real-time visualization of the process flow and evaluation of critical operational scenarios.

This tool is validated by comparing simulated results with historical records, ensuring its reliability and applicability in real-world contexts. The results show that the tool accurately represents the dynamics of the reception process, with early identification of inefficiencies such as excessive accumulations or suboptimal machinery speeds. It also provides a solid basis for strategic decision-making regarding equipment renewal, estimating their impact without introducing new operational restrictions. Furthermore, the implemented solution has a modular and scalable architecture, set as an adaptable technology to future developments and transferable to other lines or wineries in the sector.

Índice general

Índice de figuras	8
Índice de tablas	10
1. Introducción	11
1.1. Propósito	11
1.2. Objetivos	12
1.2.1. Objetivo General	12
1.2.2. Objetivos Específicos	12
1.2.3. Contribuciones	13
1.3. Estructura de la memoria	14
2. Contextualización del problema	15
2.1. ¿Como se produce el vino?	16
2.2. ¿Qué es una línea de producción?	17
2.2.1. Líneas en Viña Errazuriz	18
2.2.2. Ícono	18
2.2.3. Línea Confluencia	19
2.2.4. Doesia	20
2.2.5. Tecle	21
2.2.6. Líneas con Prensas	22
2.2.7. Máquinas de las líneas	24
2.3. Variables del proceso	26
2.4. Problemática	26
3. Estado del Arte	28
3.1. Problemas de optimización de líneas de producción	28

3.1.1. Job-Shop Scheduling	29
3.1.2. Line Balancing Problem	33
3.2. Buisiness Process Management	35
4. Diseño de la Solución	39
4.1. Requisitos del Sistema	39
4.1.1. Requerimientos Funcionales	40
4.1.2. Requerimientos de Interfaces	40
4.1.3. Requerimientos de Ambiente	41
4.2. Arquitectura del Sistema	42
4.2.1. Diagrama de Contexto	42
4.2.2. Diagrama de Arquitectura	43
4.2.3. Descripción de Componentes	44
4.2.4. Matriz de requisitos funcionales y componentes	44
4.3. Diseño de interfaces	45
4.3.1. Modelo de navegación	45
4.3.2. Prototipo de Interfaces Usuarías	48
5. Implementación y Despliegue de la Solución	51
5.1. Caso de estudio	51
5.2. Implementación de la Arquitectura	52
5.2.1. Entorno de Desarrollo Local	52
5.2.2. Entorno DevOps en la Nube	53
5.2.3. Proxy Comunicador	54
5.2.4. Base de Datos	55
5.2.5. API	56
5.2.6. Autenticación	58
5.2.7. Motor de la Aplicación Web	60
5.3. Frontend	62
5.3.1. Inicio de sesión	62
5.3.2. Dashboard	63
5.3.3. Formulario de nueva simulación	63
5.3.4. Vista de simulación	65
5.3.5. Historial de simulaciones	66
5.4. Despliegue de la solución	67
5.4.1. Verificación y Validación del Sistema	67

5.4.2. Plan de Pruebas	69
5.4.3. Gestión de Riesgos	70
6. Conclusiones y Perspectivas	72
6.1. Conclusiones Generales	72
6.2. Caso de estudio: Viña Errázuriz	74
6.3. Perspectivas y trabajo futuro	74

Índice de figuras

2.1. Resumen del proceso de producción del vino.	17
2.2. Diagrama línea Ícono.	19
2.3. Diagrama línea confluencia.	20
2.4. Diagrama línea Doesia.	21
2.5. Diagrama línea Tecle.	22
2.6. Diagrama línea Prensa 1.	23
2.7. Diagrama línea Prensa 2.	23
2.8. Ejemplos de máquinas utilizadas en la recepción de la uva.	25
3.1. Ejemplo de Job-Shop Scheduling con 3 maquinas y 3 trabajos.	30
3.2. Solución al ejemplo de Job-Shop Scheduling.	31
3.3. Programación de tareas de descarga de uva.	33
3.4. Ejemplo de MALBP.	35
3.5. Ciclo de vida del BPM [29].	36
4.1. Esquema del sistema.	42
4.2. Diagrama de Arquitectura generalizado del sistema.	43
4.3. Flujos de autenticación y de uso de la herramienta.	46
4.4. Flujo completo de la aplicación.	47
4.5. Mock-Up Inicio de Sesión de la aplicación.	48
4.6. Mock-Up Dashboard de la aplicación.	49
4.7. Mock-Up Formulario de nueva simulación.	50
5.1. Flujo de autenticación con token JWT.	60
5.2. Vista de inicio de sesión	62
5.3. Vista del dashboard principal	63
5.4. Formulario de configuración de simulación.	64

5.5. Vista de simulación en tiempo real	65
5.6. Tabla de historial de simulaciones.	66

Índice de tablas

2.1. Tabla de Procesos de Producción de Vino.	24
4.1. Requerimientos funcionales del sistema.	40
4.2. Requerimientos de interfaces del sistema.	41
4.3. Matriz de requisitos funcionales y componentes.	45
5.1. Descripción de los endpoints de la API	58
5.2. Códigos de colores en la vista de simulación y su interpretación	66

Capítulo 1

Introducción

En la industria vitivinícola, los procesos de recepción y producción de uva son altamente complejos, por lo tanto una comprensión profunda de estos es crucial para garantizar la eficiencia operativa y la calidad del producto final. La vendimia es el proceso de recolección y recepción de las uvas destinadas a la producción de vino y representa uno de los momentos más críticos dentro del ciclo de producción debido al tiempo acotado en el que las uvas alcanzan su punto óptimo de madurez. Este período implica la coordinación de múltiples variables, desde la cosecha, la entrega de la uva en la bodega y el proceso de recepción y limpieza de la fruta antes de comenzar el proceso de fermentación. Un manejo inadecuado de estos factores puede impactar negativamente en la productividad y en la calidad del vino producido. Este trabajo aborda específicamente la problemática del proceso de recepción de uva durante la época de vendimia en Viña Errázuriz, una de las bodegas mejor posicionadas internacionalmente de Chile, con 15 vinos galardonados en la última versión de Decanter World Wine Awards [1]. El proceso de recepción es vital, ya que marca el inicio de la transformación de la materia prima en un producto de alta calidad. Sin embargo, la naturaleza dinámica y volátil de la época de vendimia requiere de un control riguroso y de herramientas adecuadas para asegurar que la materia prima sea procesada en condiciones óptimas [2].

1.1. Propósito

El proceso de recepción de uva en Viña Errázuriz comienza con su llegada a la planta, transportada en bins de plástico o camiones de volteo. En el primer paso, las uvas son volcadas en el inicio de la línea, donde ingresan a la despalladora, que separa las bayas de los racimos. A continuación, las uvas

pasan por una serie de máquinas de limpieza, donde se eliminan impurezas o residuos, desplazándose a través de cintas transportadoras. Finalmente, las uvas limpias son trasladadas mediante bombas de masa a las cubas de fermentación, donde continúa el proceso de producción del vino. En este trabajo, **cuello de botella** en el proceso de recepción de uva se refiere a una etapa específica en la cadena de producción donde el flujo de trabajo se ralentiza o se interrumpe, reduciendo la capacidad global del sistema. Este problema puede surgir cuando alguna maquinaria, como la despalilladora, las cintas transportadoras, las máquinas de limpieza o las bombas de masa, no pueden manejar el volumen de uvas entrante, provocando acumulaciones o retrasos en el flujo de procesamiento. Actualmente, la viña enfrenta desafíos en la identificación de cuellos de botella en este proceso, además de contar con múltiples máquinas que han superado su período útil y necesitan ser reemplazadas. Por lo tanto, es necesario tomar decisiones de inversión estratégicas para determinar qué máquinas reemplazar sin generar nuevas limitaciones en el proceso. A partir de esto se plantea la pregunta que desea resolver este trabajo de ingeniería, **¿Cómo identificar eficazmente los cuellos de botella en el proceso de recepción de uva y entregar información relevante para la toma de decisiones de inversión?** Para abordar esta situación, se plantea el desarrollo de una herramienta digital que permita optimizar el proceso de recepción de uva, mejorando su visualización y control. Esta herramienta no solo ayudará a mejorar la coordinación y seguimiento en tiempo real de la vendimia, sino que también brindará soporte a la toma de decisiones estratégicas, garantizando una mayor eficiencia en la operación.

1.2. Objetivos

Para responder esta pregunta, se establecerá un objetivo general y objetivos específicos orientados a analizar detalladamente el flujo actual del proceso y desarrollar una herramienta digital que mejore su visualización y control.

1.2.1. Objetivo General

Visualizar completa e íntegramente el proceso de recepción de la uva en la vendimia.

1.2.2. Objetivos Específicos

- **OE1: Definir y cuantificar las diferentes líneas de producción en el proceso de recepción de uva.** Definir y detallar cada una de las 7 líneas de producción con las que cuenta la empresa, evidenciando la existencia de cada máquina y su relevancia en el proceso. Se

explicará la relación en el ciclo productivo para obtener un panorama total del proceso, gráfico y documentado, que facilite el desarrollo del proyecto. Además, se obtendrán métricas de cada máquina en funcionamiento actual y de su historial. Se normalizará el sistema de unidades con el que se trabajará, tanto para flujos como para velocidades, dando énfasis a los insumos del proceso y considerando los diversos formatos en los que llega la uva.

- **OE2: Modelar el proceso de recepción de la uva.** Modelar cada línea considerando los parámetros y ruta crítica tal que simule el funcionamiento del proceso actual. Con esto se podrá evaluar diferentes escenarios de funcionamiento.
- **OE3: Desarrollo y validación de una herramienta digital basada en el modelo del proceso.** Desarrollar una herramienta digital basada en el modelo matemático del proceso de recepción de uva obtenido en el OE2, con una interfaz de usuario que permita una visualización completa de las diferentes líneas de producción. Validar la herramienta para escenarios críticos de producción, simulando el proceso en las diferentes líneas utilizando parámetros de bases históricas con las que cuenta la empresa, y contrastar los resultados con los resultados reales. Esto garantizará que la herramienta sea precisa y útil para mejorar la eficiencia operativa y apoyar la toma de decisiones estratégicas.

1.2.3. Contribuciones

Las contribuciones de este trabajo incluyen el desarrollo de una herramienta digital que mejora la visualización del proceso de recepción de uva durante la vendimia, optimizando el control operativo y permitiendo un seguimiento detallado de las etapas involucradas, lo que facilita la identificación de ineficiencias y mejora la toma de decisiones estratégicas. En cuanto a la arquitectura de la herramienta, se plantea el desarrollo una aplicación digital con una estructura modular y una interfaz de usuario intuitiva. Las siete líneas de producción y las máquinas involucradas en el proceso de recepción de uva se integran eficientemente en esta herramienta, considerando una base tecnológica alineada con estándares de la industria para escalabilidad funcional y operativa.

En lo que respecta a la simulación del proceso, se creó un modelo de producción que representa el proceso de recepción de uva. Esta simulación permite la evaluación de diferentes escenarios operativos y la predicción del comportamiento del sistema bajo diversas condiciones, apoyando la toma de decisiones de inversión mediante el análisis del impacto de reemplazar o actualizar máquinas sin generar nuevos cuellos de botella. Finalmente, la herramienta proporciona indicadores gráficos y animaciones que permiten visualizar de forma rápida las máquinas que operan a velocidades sub-óptimas o las acumulaciones de uva en el proceso. Esta funcionalidad facilita la identificación de ineficiencias y

patrones recurrentes en las líneas de producción, mejorando la toma de decisiones estratégicas al ofrecer información clara sobre las limitaciones en el flujo de trabajo.

1.3. Estructura de la memoria

La estructura del presente documento se compone de la siguiente forma:

- En el Capítulo 2 se formaliza y contextualiza el problema.
- En el Capítulo 3 se analizan las diversas técnicas utilizadas para desarrollar soluciones tecnológicas en el contexto de optimización de flujos de procesos.
- En el Capítulo 4 presenta el diseño de la solución propuesta, describiendo el caso de uso, los diagramas detallados del flujo de la aplicación y sus módulos, y su conexión con los requisitos funcionales.
- En el Capítulo 5 aborda la implementación y despliegue de la solución, detallando el proceso realizado en cada uno de sus módulos y la interfaz del sistema que los integra.
- En el Capítulo 6 se presentan los resultados obtenidos tras la implementación de la herramienta.
- En el Capítulo 7 se concluye destacando las principales contribuciones del proyecto, reconociendo las limitaciones encontradas, y proponiendo futuras mejoras y líneas de investigación.

Capítulo 2

Contextualización del problema

El proceso de elaboración del vino se sustenta en una serie de etapas estructuradas y gestionadas, cuyo correcto desarrollo es esencial para obtener un producto de alta calidad y valor comercial. En este capítulo se aborda el proceso general de producción del vino, detallando las etapas involucradas desde la vendimia hasta el embotellado.

Posteriormente, se introduce el concepto de línea de producción, enfocándose particularmente en su aplicación dentro del contexto vitivinícola y describiendo las diversas configuraciones empleadas en Viña Errazuriz, analizando las líneas específicas presentes, destacando aspectos técnicos, características operativas y tecnológicas, y su influencia en la calidad final del producto. Asimismo, se describen las máquinas implicadas en los procesos de recepción y tratamiento inicial de la uva.

Finalmente, el capítulo define y analiza las variables involucradas en la operación de estas líneas, destacando la importancia de identificar y gestionar adecuadamente los cuellos de botella, los cuales constituyen un factor determinante para la optimización del proceso productivo.

2.1. ¿Como se produce el vino?

El vino representa una de las bebidas más ancestrales producidas por la humanidad, resultando del producto final de un proceso intrincado que combina ciencia, arte y tradición [3]. Este proceso incluye la vendimia, el despalillado, la molienda, el prensado, la fermentación, la decantación, la maduración y, finalmente, el embotellado [4]. A continuación se detallan cada una de ellas:

1. **Vendimia:** Constituye el primer paso en la elaboración del vino, donde se recolectan las uvas en su punto óptimo de madurez. Este proceso se realiza generalmente entre finales del verano y principios del otoño. La recolección puede efectuarse de manera manual, permitiendo una selección más precisa de los racimos, o de forma mecánica para una mayor eficiencia.
2. **Despalillado:** Una vez cosechadas, las uvas se someten al despalillado, que consiste en separar los granos de uva de los raspones o tallos. El despalillado se lleva a cabo mediante máquinas especializadas que eliminan los tallos sin dañar los granos. Dependiendo de la calidad del vino a producir, este proceso se realiza en líneas pequeñas con máquinas más eficientes que producen poco mosto, o en máquinas grandes donde se vuelcan camiones de fruta para ser procesados. Las uvas junto con el *mosto*, que es el jugo fresco obtenido de las uvas y que contiene azúcares, ácidos y otros compuestos esenciales para la fermentación, debe ser llevado por una bomba de masa hacia el siguiente paso.
3. **Molienda:** Tras el despalillado, se procede a la molienda de las uvas. Este paso implica romper suavemente las pieles de los granos para liberar el mosto sin triturar las semillas.
4. **Prensado:** El prensado tiene como objetivo extraer el máximo volumen de mosto de las uvas molidas. Mediante prensas hidráulicas o neumáticas, se aplica presión controlada para separar el líquido de los sólidos (pieles, pulpa y semillas). El mosto obtenido se destina al proceso de fermentación, mientras que los restos sólidos, conocidos como orujo, pueden utilizarse para la destilación de aguardientes o como abono orgánico.
5. **Fermentación:** La fermentación es el proceso biológico en el cual las levaduras convierten los azúcares del mosto en alcohol etílico y dióxido de carbono. Este proceso se desarrolla en tanques de acero inoxidable.
6. **Decantación:** Finalizada la fermentación, el vino se somete a la decantación o trasiego para separar el líquido de las lías (sedimentos de levaduras y otros sólidos). Este proceso implica transferir el vino a otro recipiente, dejando atrás las partículas sedimentadas.
7. **Maduración:** El vino decantado entra en un periodo de maduración, donde se afina y desarrolla

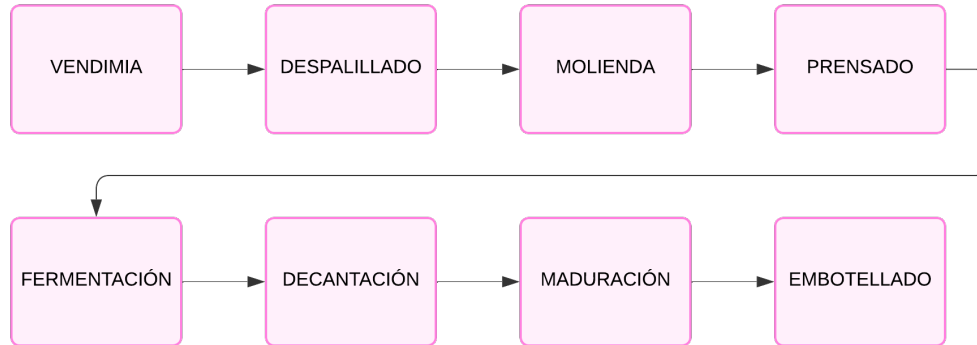


Figura 2.1: Resumen del proceso de producción del vino.

su carácter distintivo. La maduración puede realizarse en barricas de roble o en tanques de acero inoxidable.

8. **Embotellado:** Finalmente, el vino se embotella bajo condiciones controladas para preservar su calidad y longevidad.

2.2. ¿Qué es una línea de producción?

Las líneas de producción se utilizan para automatizar la fabricación de una amplia gama de productos, por lo que su aplicación se ha extendido a numerosos ámbitos industriales. En este trabajo se toma como referencia la definición propuesta por Bierbooms [5], quien describe una línea de producción como un proceso de manufactura o ensamblaje en el que los materiales se someten a un procesamiento secuencial hasta obtener un producto final.

Los procesos en los que se enfoca este trabajo corresponden a la recepción y despalillado de la uva antes de ser enviada a la cuba de fermentación. Una vez cosechadas durante la vendimia, las uvas llegan a la bodega en tres posibles formatos: gamelas para uvas recogidas a mano, bins de mayor tamaño para ser cargados con grúas horquilla y, finalmente, camiones de volteo.

Una vez recibida la carga, esta se introduce en una serie de máquinas destinadas al proceso de despalillado, durante el cual se eliminan los palos del racimo, las uvas de tamaño reducido y cualquier residuo contenido en la carga inicial.

Al finalizar el proceso de despalillado, tanto las uvas como el mosto son transferidos a las cubas de fermentación. Este transporte se efectúa mediante bombas de masa, las cuales están conectadas a tuberías que se extienden hacia diversas áreas de la bodega.

A lo largo de este escrito, se considera el término **“Línea de Producción”** como el conjunto secuencial de todas las máquinas responsables de los procesos previos al envío de la uva a las tuberías de distribución.

En la siguiente sección, se ejemplifica el concepto de Línea de Producción aplicado al caso concreto de Viña Errázuriz. Esta viña, ubicada en el Valle de Aconcagua, cuenta con siete líneas distintas para la recepción de uva. Cada una de estas líneas tiene configuraciones propias, distintos niveles de automatización, capacidades diferentes y está orientada a tipos específicos de producción de vino.

Estas diferencias entre líneas permiten observar una variedad de situaciones reales que ayudan a comparar cómo funciona cada configuración y a estudiar sus dinámicas. El objetivo de esta parte del trabajo es describir detalladamente cada una de estas líneas, usando la información técnica que se recopiló directamente en la viña. Esta caracterización servirá de base para construir el modelo de simulación que se desarrolla más adelante en la memoria.

2.2.1. Líneas en Viña Errázuriz

La configuración y el funcionamiento de las líneas de producción pueden variar significativamente entre una viña y otra. Factores como el tipo de producto, el volumen de elaboración, la tecnología disponible, así como las condiciones y objetivos específicos de cada bodega, influyen en el diseño, la disposición de las máquinas y el flujo del proceso. En el caso de Viña Errázuriz, poseen siete líneas distintas en su planta en Panquehue, las cuales son descritas a continuación, destacando las características principales de cada una.

2.2.2. Ícono

La línea Ícono es actualmente la que emplea la tecnología más avanzada para procesar las uvas de mayor calidad y cuidado, destinadas específicamente a la producción de vino tinto. Equipada con máquinas de la marca Pellenc, importadas desde Francia, sigue una configuración en serie que permite separar con precisión el escobajo, el mosto y las uvas en mal estado, preparando así la materia prima para su posterior fermentación en la cuba.

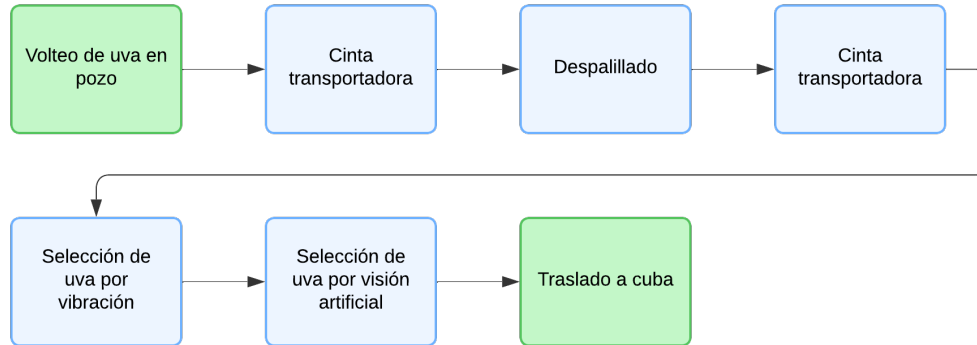


Figura 2.2: Diagrama línea Ícono.

2.2.3. Línea Confluencia

Este proceso se caracteriza por presentar un punto de unión entre dos líneas de producción que siguen la misma configuración (véase Fig. 2.3). Cada una cuenta con un punto de inicio o carga de materia prima independiente, la cual es procesada por el mismo tipo de máquinas, pero a velocidades diferentes. Aquí, el mosto se combina con la uva procedente de la despalilladora, para luego ser conducido a la cuba a través de tuberías. Dado que en esta línea se procesa uva para vino tinto de menor calidad que en la línea Ícono, se trabaja a una mayor velocidad de producción. Esto incrementa la cantidad de escobajo, el cual requiere ser extraído mediante una bomba de masa que integra ambos puntos.

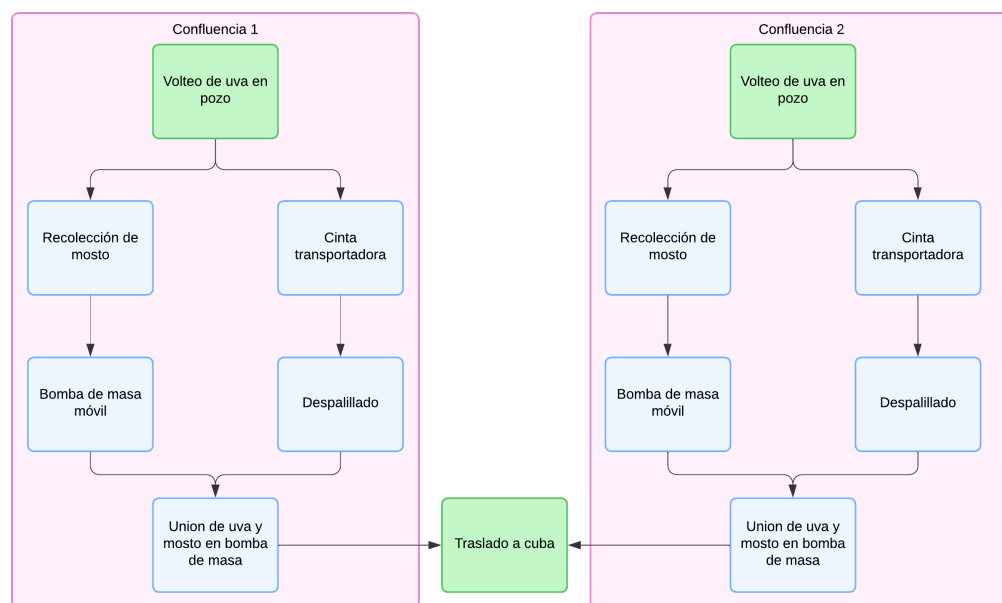


Figura 2.3: Diagrama línea confluencia.

2.2.4. Doesia

La línea Doesia presenta una configuración equivalente a contar con una única entrada de materia prima del proceso Confluencia. No obstante, se encuentra separada del resto de las líneas dentro de una bodega cerrada, ya que en ella se procesa un tipo de uva especial que, debido a su composición química, resulta muy susceptible al ataque de plagas en espacios abiertos.

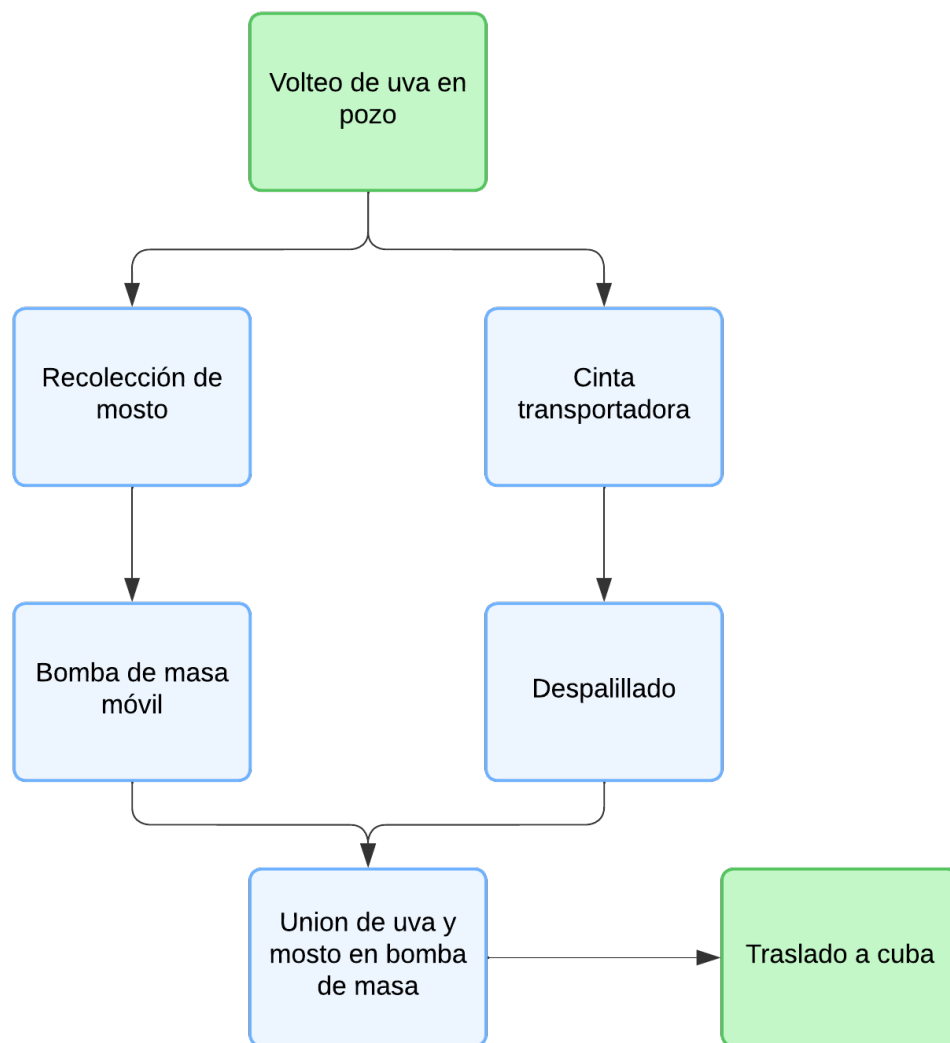


Figura 2.4: Diagrama línea Doesia.

2.2.5. Tecle

Viña Errazuriz cuenta con dos líneas Tecle idénticas que pueden operar de forma independiente. Estas líneas están diseñadas para manejar uvas de calidad inferior y con un mayor volumen de carga, motivo por el cual la etapa inicial consiste en voltear la tolva de un camión, en lugar de utilizar gamelas o bines, como sucede en otras líneas. La uva procesada en estas líneas se destina, en su mayoría, a la producción de vino blanco.

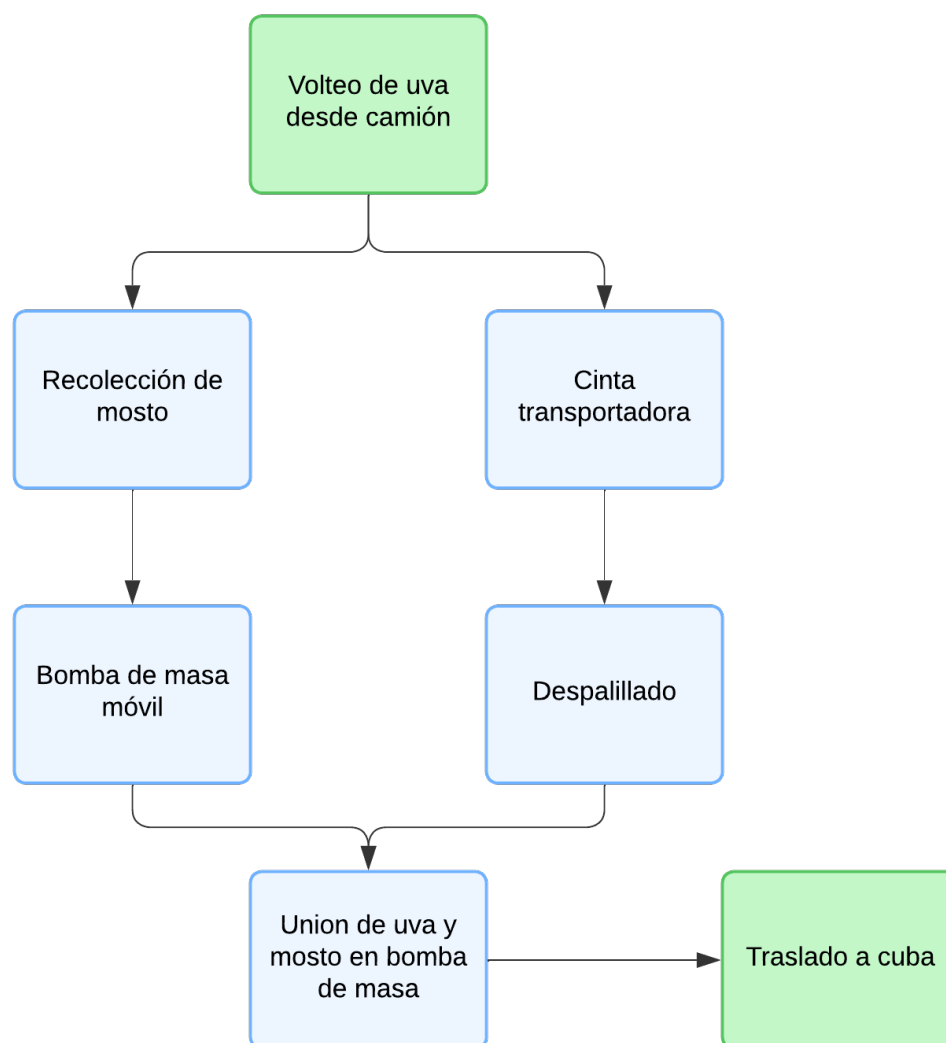


Figura 2.5: Diagrama línea Tecle.

2.2.6. Líneas con Prensas

Este proceso se divide en dos líneas, cada una equipada con prensas industriales de alta presión, capaces de extraer por completo el líquido de las uvas destinadas a la producción de vino blanco. La carga de materia prima se realiza a partir de grandes camiones, al igual que en la línea Tecle, recuperando así el mosto, y a diferencia del resto de las líneas, se extrae también el hollejo. La principal diferencia entre una línea Prensa y otra radica en la presencia de una prensa adicional en la primera (vease Fig. 2.7), lo que genera una mayor cantidad de desechos.

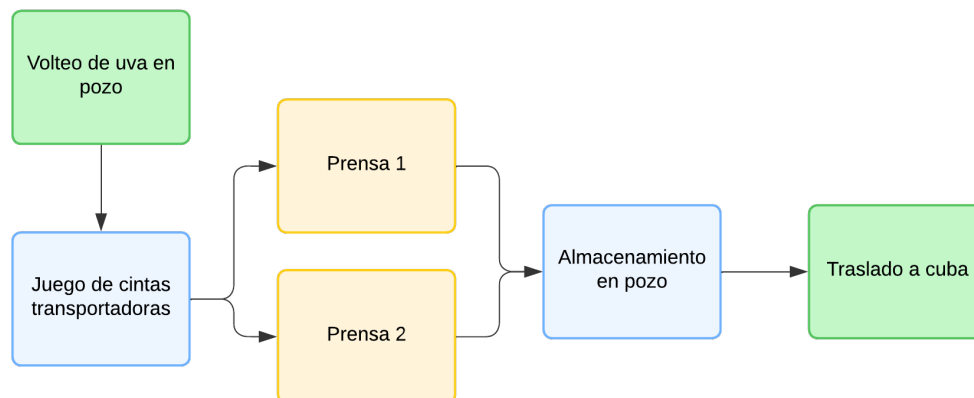


Figura 2.6: Diagrama línea Prensa 1.

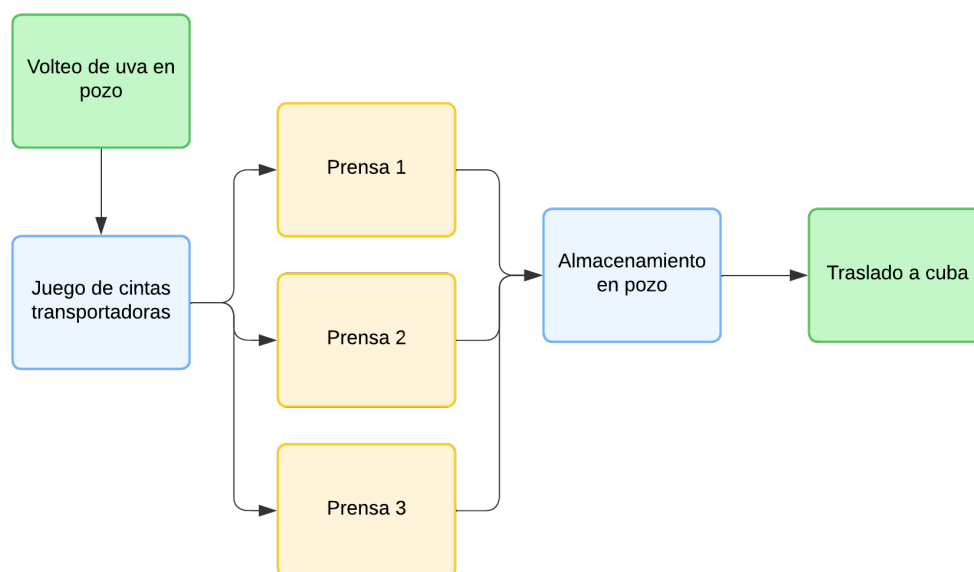


Figura 2.7: Diagrama línea Prensa 2.

En la Tabla 2.1 se presentan de manera resumida las principales características, en caso de que apliquen, de las diferentes líneas discutidas a lo largo de esta sección, abarcando aspectos clave que han sido analizados previamente y que son relevantes dentro del contexto presentado.

Línea	Ícono	Confluencia	Doesia	Tecle	Prensa
Volteo	Gamelas	Bin	Bin	Camión	Camión
Vino	Tinto	Tinto y Blanco	Tinto	Tinto y Blanco	Blanco
Tecnología	Alta	Media	Alta	Media	Media
Recolección de mosto	×	✓	✓	✓	✓
Bomba de masa en escobajo	×	✓	✓	×	×
Costo por botella	Alto	Alto	Medio	Bajo	Medio
Variedades de Vino	9	8	S.I	2	S.I

Tabla 2.1: Tabla de Procesos de Producción de Vino.

2.2.7. Máquinas de las líneas

A continuación se detallan las distintas máquinas de las líneas descritas anteriormente.

1. **Pozo con tornillo:** El pozo es la estructura inicial donde se reciben las uvas una vez descargadas en la bodega. Este pozo cuenta con un tornillo sin fin en su base que permite transportar las uvas de manera controlada hacia la siguiente etapa del proceso. Se asume que posee una capacidad suficiente para recibir toda la carga inicial.
2. **Cinta transportadora:** La cinta transportadora es un mecanismo encargado de trasladar las uvas entre dos máquinas, permitiendo un flujo continuo y uniforme de la fruta. Esta posee un receptáculo encargado de acumular fruta en caso de que la velocidad de la maquina anterior sea mayor a la velocidad de la cinta.
3. **Despalilladora:** La despalilladora es la máquina encargada de separar el fruto de uva de los tallos. Utiliza un tambor giratorio perforado y un eje con paletas que desprenden los granos sin causar daños.
4. **Bomba de masa:** La bomba de masa es el dispositivo que transporta el mosto y las uvas despalilladas desde la línea hasta las cubas de fermentación. A esta bomba se le añade un receptáculo para que posea capacidad de acumulación.
5. **Bomba de agua:** La bomba de agua es el dispositivo que transporta el mosto desde las máquinas donde se produce, ya sea por el golpeo o presión de la uva, hasta las cubas para ser almacenado.
6. **Maquina de vibración:** La máquina de vibración es un equipo avanzado utilizado para distribuir y separar las uvas despalilladas mediante vibraciones controladas. Este comportamiento permite

que las uvas se desplacen sobre una superficie tamizada, eliminando restos de pieles, semillas, fragmentos de tallos y otras impurezas. Esta maquina también posee un receptáculo.

7. **Maquina con visión:** La máquina con visión, o clasificador óptico, es un equipo avanzado que utiliza tecnología de visión artificial para analizar y seleccionar las uvas individualmente. Mediante cámaras de alta resolución y software especializado, esta máquina identifica y separa automáticamente las uvas defectuosas, dañadas o con características indeseadas.
8. **Prensa:** La prensa es la máquina industrial encargada de extraer el jugo de las uvas por medio de una presión controlada. Al ejercer fuerza sobre la uva, separa el mosto de los hollejos y otros sólidos.



(a) Bomba de agua utilizada en la línea Icono.



(b) Bomba de masa móvil utilizada en varias líneas.



(c) Cinta transportadora utilizada en la línea Icono.



(d) Despalilladora de la línea Confluencia.

Figura 2.8: Ejemplos de máquinas utilizadas en la recepción de la uva.

2.3. Variables del proceso

Para la correcta caracterización del proceso, se deben definir las variables numéricas involucradas.

- **Carga inicial:** Al principio del proceso de recepción se posee una carga de uva con palo en kilogramos.
- **Velocidad de la máquina:** Todas las máquinas descritas previamente poseen una velocidad de salida, que representa cuantos kilogramos de uva que recibe la máquina son enviados a la siguiente en una hora (kg/h).
- **Capacidad de acumulación:** Las máquinas que poseen receptáculos son capaces de recibir una cantidad determinada de uva en kilogramos desde una maquina más rápida, esta capacidad se mide en kilogramos.
- **Velocidad de salida:** Al final del proceso y dependiendo de las máquinas involucradas, la uva sale desde la línea hacia la cuba a una velocidad determinada en kg/h , que sirve como métrica de comparación entre la línea como unidad completa y cada uno de sus componentes.
- **Rendimiento:** Hay máquinas, como las prensas, que separan la uva del mosto generando residuo con respecto a la carga inicial, por lo que cantidad de materia prima de salida es menor en un cierto porcentaje.

Teniendo en cuenta estas variables se define un término crucial de este escrito, **cuello de botella**. Se entiende como cuello de botella a una máquina que posea una velocidad menor al resto de la línea, culminando en que la velocidad de salida del proceso sea lenta. Otro caso de cuello de botella o “caso de infacilidad” corresponde a una máquina que, debido a la capacidad de su receptáculo, pueda desbordarse causando una detención inminente del proceso de recepción y despallado.

2.4. Problemática

El proceso de recepción y despallado de la uva es una etapa crítica para Viña Errazuriz, donde la eficiencia y el control son esenciales para mantener la calidad del producto final. A pesar de contar con una línea de producción que incluye máquinas de alta tecnología, es común enfrentar problemas de ineficiencia debido a cuellos de botella en alguna de estas etapas. Estos cuellos de botella pueden generar retrasos, aumentar los costos operativos y afectar negativamente la calidad del mosto y, por ende, del vino producido. Este problema radica en *identificar eficazmente los cuellos de botella en el proceso de recepción de uva y proporcionar información relevante para la toma de*

decisiones de inversión. La identificación precisa de estos puntos críticos permitirá optimizar el flujo de trabajo, mejorar la coordinación entre las diferentes etapas del proceso y garantizar que cada máquina opere a su máxima capacidad sin comprometer la calidad.

Para abordar esta problemática, se propone el desarrollo de una herramienta digital que facilite la visualización y el control del proceso de recepción de uva, aplicada específicamente a la línea “Ícono”. Los detalles sobre la metodología, diseño e implementación de dicha herramienta serán abordados en profundidad en los próximos capítulos.

Capítulo 3

Estado del Arte

La gestión eficiente de los procesos productivos y logísticos en entornos industriales ha sido objeto de extensa investigación a lo largo de las últimas décadas. En particular, la optimización de operaciones dentro de líneas de producción representa un campo fértil en el que confluyen diversas disciplinas como la investigación de operaciones, la ingeniería industrial y la informática. Este capítulo presenta una revisión del estado del arte centrada en metodologías, modelos y enfoques aplicables al contexto de la industria vitivinícola, con especial énfasis en los desafíos asociados a la programación, asignación de recursos y mejora continua de procesos.

Dada la naturaleza compleja, dinámica y altamente estacional del sector vitivinícola, resulta pertinente estudiar problemas clásicos como el *Job-Shop Scheduling*, el *Line Balancing Problem* y enfoques contemporáneos como la *Gestión de Procesos de Negocio* (Business Process Management, BPM). Estos marcos teóricos y metodológicos permiten modelar, analizar y optimizar la producción desde múltiples ángulos, considerando tanto las restricciones operativas como los objetivos estratégicos de las bodegas.

3.1. Problemas de optimización de líneas de producción

La optimización y el modelamiento de problemas asociados a líneas de producción han sido ampliamente estudiados en distintos contextos industriales. En el ámbito de las bodegas vinícolas y sus operaciones internas, resulta especialmente relevante considerar diversas clases de problemas de planificación, programación y asignación de recursos. A continuación, se destacan algunos tipos de problemas cuya aplicación se ha explorado en la literatura y que pueden ser adaptados al contexto de

las líneas de producción de vino.

3.1.1. Job-Shop Scheduling

Este problema se centra en asignar un conjunto de trabajos compuestos por múltiples operaciones a un conjunto de máquinas o recursos, respetando las restricciones de capacidades de éstas máquinas así como las restricciones de precedencia entre las distintas tareas mientras se busca optimizar uno o más criterios o métricas.

El *Job Shop Scheduling* [6] se considera uno de los problemas fundamentales en la disciplina de investigación de operaciones y optimización combinatoria. Este se define en su forma clásica como el conjunto de tareas $\mathcal{T} = \{T_0, T_1, \dots, T_n\}$; donde T_0 y T_n se consideran trabajos ficticios con un tiempo de procesamiento cero; el conjunto de máquinas $\mathcal{P}$ y el conjunto de relaciones de precedencia $\mathcal{A}$; donde $(T_i, T_j) \in \mathcal{A}$ indica que la tarea T_i precede a la tarea T_j . Para cada máquina P_k , el conjunto $\mathcal{E}_k$ describe todas las tareas a ser ejecutadas en dicha máquina y que no pueden solaparse. Para cada tarea T_i , su tiempo de procesamiento p_i es fijo. El tiempo de inicio t_i para cada tarea T_i debe determinarse como parte de la solución [7]. El objetivo es minimizar el tiempo de finalización de la última tarea (makespan):

$$\text{mín } z = t_n \quad (3.1)$$

$$\text{s.t.} \quad (3.2)$$

$$t_j - t_i \geq p_i \quad \forall (T_i, T_j) \in \mathcal{A},$$

$$t_j - t_i \geq p_i \text{ or } t_i - t_j \geq p_j \quad \forall (T_i, T_j) \in \mathcal{E}_k, \forall P_k \in \mathcal{P}, \quad (3.3)$$

$$t_i \geq 0 \quad \forall T_i \in \mathcal{T} \quad (3.4)$$

Las restricciones (3.1) garantizan que la secuencia en la que se procesan las tareas dentro de cada trabajo sea la previamente definida. Por otra parte, las restricciones (3.2) aseguran que en cada máquina se ejecute solo una tarea a la vez, evitando solapamientos, mientras que la condición (3.3) establece que todas las tareas deben iniciar en un tiempo no negativo.

La Figura 3.1 ilustra una representación de una instancia clásica de un problema de job-shop con 3 máquinas ($P1, P2, P3$) y 3 trabajos ($J1, J2, J3$), donde en total se deben procesar 8 operaciones. Se considera que $J1$ debe procesarse en el orden $P1 \rightarrow P2 \rightarrow P3$, $J2$ sigue la secuencia $P3 \rightarrow P2$ y $J3$ con orden $P2 \rightarrow P1 \rightarrow P3$. Los nodos del grafo representan las tareas a realizar en cada máquina (por

ejemplo, el nodo 1 corresponde a la tarea de la máquina $P1$), los arcos dirigidos corresponden a las restricciones de precedencias de las tareas para cada trabajo y los arcos disyuntivos se dibujan entre las tareas que comparten la misma máquina. Los valores sobre los arcos corresponden a las unidades de tiempo de procesamiento entre las tareas.

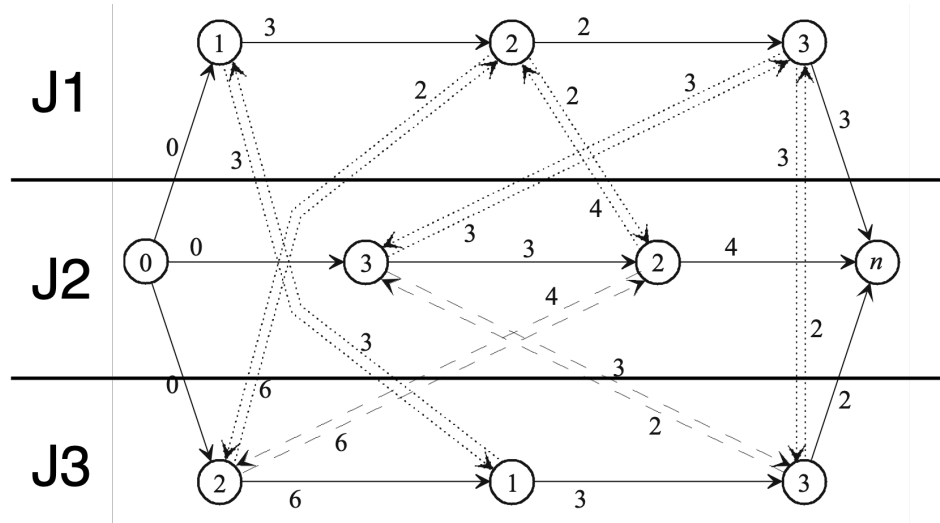


Figura 3.1: Ejemplo de Job-Shop Scheduling con 3 máquinas y 3 trabajos.

La Figura 3.2 muestra una solución factible al problema, donde se ha elegido para cada par disyuntivo, una dirección que garantiza que el grafo resultante sea acíclico y respete la secuencia de tareas de cada trabajo. En esta solución se observa que, por ejemplo, en la máquina $P1$ se procesa el trabajo $J1$ seguido de $J3$, mientras que en las máquinas $P2$ y $P3$ la secuencia es $J1$, $J2$ y $J3$. De esta forma, todas las tareas se ubican en un único camino crítico que cumpla con recorrer todos los nodos respetando las restricciones de precedencia, en este caso, el camino ejecuta todas las tareas de $J1$, después las tareas de $J2$ y finalmente las tareas de $J3$, dando así un makespan de 26 unidades de tiempo.

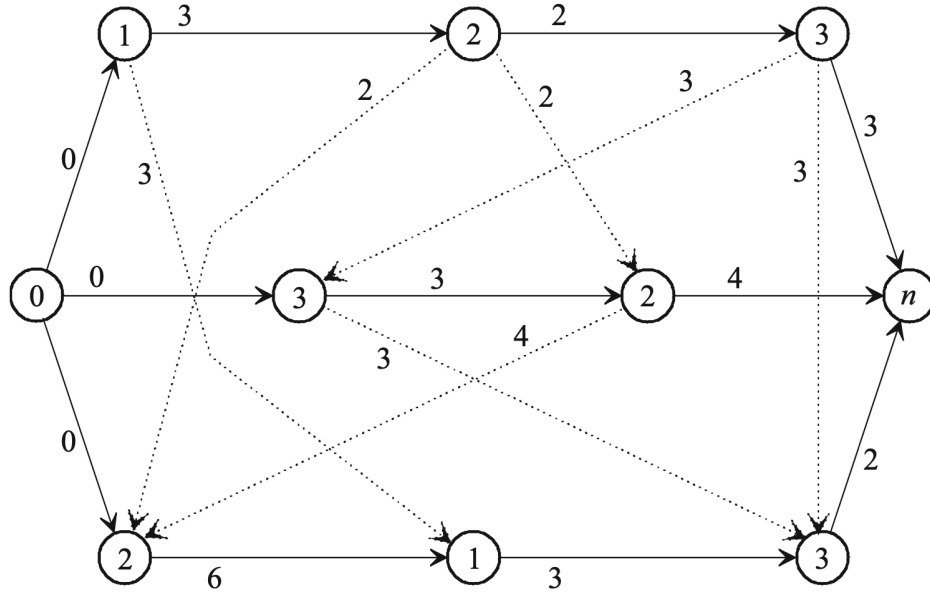


Figura 3.2: Solución al ejemplo de Job-Shop Scheduling.

Los primeros enfoques de solución para este problema están basados en métodos exactos, los cuales pudieron resolver instancias de tamaño reducido [8]. Debido a la complejidad intrínseca del problema, esta establecido en la literatura que corresponde a la clase de problemas NP-hard [9], por lo que las investigaciones más recientes adoptan enfoques en heurísticas y metaheurísticas para encontrar soluciones de alta calidad en tiempos de cómputo razonables [10]. Entre las técnicas más destacadas se encuentran algoritmos genéticos [11], búsqueda tabú [12], recocido simulado [13], búsqueda local iterada [14] y métodos híbridos que combinan estas aproximaciones [6].

La industria vitivinícola integra una cadena de valor con múltiples etapas secuenciales y recursos compartidos, desde la cosecha de la uva hasta el embotellado y distribución final del vino. Cada etapa (véase Capítulo 2.1), requiere un conjunto específico de máquinas, equipos y tanques con capacidades limitadas y condiciones particulares. Estas restricciones, sumadas a la necesidad de coordinación de disponibilidad de materias primas hacen que la programación de la producción en bodegas sea compleja y similar al problema de Job Shop Scheduling.

En el ámbito de la cosecha, [15] propone un modelo de programación lineal mixta que optimiza de forma simultánea los costos operativos y la calidad de las uvas recolectadas. Este modelo introduce una función de pérdida de calidad para medir el impacto de adelantar o retrasar la cosecha con respecto a la fecha óptima, integrando variables como el uso de maquinaria, transporte y ventanas de tiempo disponibles. Este enfoque captura la complejidad inherente a la planificación del rubro agrícola y

relacionados, en un entorno competitivo y con recursos limitados.

En [2] se propone un algoritmo genético para asignar de manera coordinada los camiones, depósitos de recepción y prensas, conforme al flujo operativo del sistema, con el fin de gestionar los recursos de manera dinámica. Cada solución generada por éste, representa una secuencia de decisiones sobre cuándo descargar uvas y en qué momento iniciar ciclos de prensado. En este caso, el problema es descrito como un Job-Shop Scheduling más restrictivo y cercano a un ambiente de producción real, considerando las capacidades de los camiones y máquinas en el algoritmo y permitiendo la solapación de tareas. El algoritmo utiliza dos operadores genéticos fundamentales para la evolución de las soluciones, mutación y cruce. El operador de mutación introduce variaciones aleatorias en las decisiones futuras, ya que modifica la solución únicamente en el segmento que aún no ha sido ejecutado. Estas variaciones incluyen la eliminación de una decisión, su desplazamiento a una posición cercana (dentro de un rango definido), la incorporación de una nueva decisión aleatoria o el intercambio entre dos decisiones. Por otro lado, el operador de cruce (o crossover) combina dos soluciones seleccionando un punto de corte aleatorio en el vector de decisiones y permutando las partes posteriores, generando así dos nuevas soluciones.

La Figura 3.3 muestra un ejemplo de solución válida para el problema, en donde se tienen dos camiones, cada uno con uvas de variedades distintas, un solo tanque de grano con capacidad de 10 toneladas y dos prensas, con capacidad de 10 toneladas cada una. La solución se representa mediante un vector de decisiones que se van ejecutando a lo largo del tiempo. En $t = 2$ se toma la primera decisión, iniciando la descarga de uvas desde el camión 1 hacia el tanque de grano. Como el tanque está disponible y el camión tiene uvas, la acción comienza. Dado el ritmo de descarga, esta acción finaliza en $t = 6$, momento en el que el camión 1 queda vacío. En $t = 7$ se ejecuta una nueva decisión, se envían las uvas acumuladas en el tanque de grano hacia la segunda prensa. La operación, limitada por la tasa de procesamiento del tanque, continúa hasta $t = 14$. En $t = 9$, se registra una decisión de comenzar la descarga de uvas del camión 2 hacia el mismo tanque de grano. Sin embargo, este camión transporta uvas de una variedad diferente a la que ya se está descargando, y dado que las uvas de distintas variedades no se pueden mezclar, esta decisión se marca como inválida (en la figura se representa con un símbolo de prohibición). Al ser inválida, su efecto se ignora durante la simulación de la solución. En $t = 12$, se toma la decisión de iniciar el ciclo de la segunda prensa. Al hacerlo, si la prensa tiene uvas disponibles, la acción comienza y, además, se impone que el tanque deje de enviar uvas hacia esa prensa.

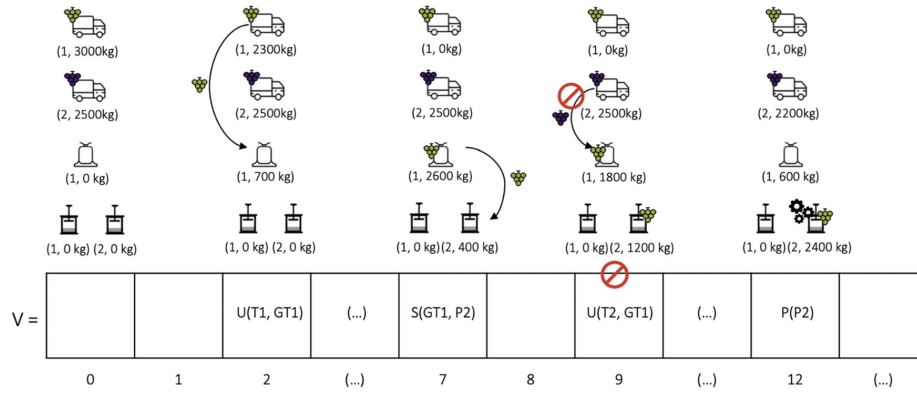


Figura 3.3: Programación de tareas de descarga de uva.

3.1.2. Line Balancing Problem

El *Line Balancing Problem* (LBP), o problema de balanceo de líneas de ensamble, se refiere al proceso de asignar tareas a estaciones de trabajo a lo largo de una línea de producción con el objetivo de minimizar el tiempo total de producción, equilibrar la carga de trabajo entre estaciones y/o minimizar el número de estaciones utilizadas [16, 17].

El estudio sistemático del balanceo de líneas comienza con [18], donde propone uno de los primeros enfoques de programación matemática para asignación de tareas a estaciones de trabajo. La literatura ha propuesto diversas clasificaciones de este problema, variando entre ellas su formulación y características específicas. La versión fundamental del problema es el *Simple Assembly Line Balancing Problem* (SALBP) [16] y se divide en dos sub problemas principales :

1. SALBP-1: Dado un número fijo de estaciones m y las restricciones de precedencia entre ellas se debe minimizar el tiempo de ciclo.
2. SALBP-2: Dado un tiempo de ciclo T_c se debe minimizar el número de estaciones necesarias

El *Mixed-Model Assembly Line Balancing Problem* (MALBP) surge cuando se requiere ensamblar más de un tipo de producto en la misma línea, aumentando la complejidad debido a las restricciones de precedencia distintas entre los modelos y el balanceo dependiente de las variantes de los productos [17]. En entornos industriales más complejos, se pueden plantear objetivos adicionales al problema clásico, como minimizar el costo de la mano de obra, minimizar la longitud física de la línea o distancia recorrida por los operarios, optimizar la ergonomía y seguridad laboral, o maximizar la eficiencia del proceso [19, 20].

En su forma más básica, el problema puede formularse como un problema de programación

entera mixta (MIP), siendo una de las representaciones más comunes la presentada en [18, 21]. Se considera un conjunto de tareas $N = 1, 2, \dots, n$ que deben ser asignadas a estaciones de trabajo, con un máximo de M estaciones disponibles. Cada tarea $i \in N$ tiene un tiempo de ejecución t_i determinado, y cada estación opera bajo un tiempo de ciclo máximo c . Además, se define un conjunto de precedencias $S \subseteq N \times N$, donde un par $(i, j) \in S$ indica que la tarea i debe ejecutarse antes que la tarea j , es decir, $i \prec j$. Se introducen variables binarias $x_{i,k}$, donde $x_{i,k} = 1$ si la tarea i se asigna a la estación k , y $x_{i,k} = 0$ en caso contrario. Asimismo, se definen variables binarias y_k , donde $y_k = 1$ si la estación k es utilizada en la solución, y $y_k = 0$ en caso contrario. El objetivo es minimizar la cantidad de estaciones utilizadas:

$$\begin{aligned}
\text{mín } z &= \sum_{k=1}^M y_k \\
\text{s.t.} \quad & \sum_{k=1}^M x_{i,k} = 1 && \forall i \in \{1, \dots, n\} \\
& \sum_{i=1}^n t_i x_{i,k} \leq c y_k && \forall k \in \{1, \dots, M\} \\
& \sum_{k=1}^M k x_{i,k} \leq \sum_{k=1}^M k x_{j,k} && \forall (i, j) \in S \\
& x_{i,k} \leq y_k && \forall i \in \{1, \dots, n\}, \forall k \in \{1, \dots, M\} \\
& x_{i,k} \in \{0, 1\}, \quad y_k \in \{0, 1\}.
\end{aligned} \tag{3.5}$$

La Figura 3.4 representa gráficamente una solución a una instancia del MALBP, donde cada nodo con un valor numérico indica el tiempo de ejecución t_i de una tarea y las flechas establecen las precedencias $i \prec j$ (conjunto S). Las áreas sombreadas corresponden a estaciones de trabajo a las cuales se asignan las tareas mediante las variables binarias $x_{i,k}$, asegurando que cada tarea se asigne a una única estación. La restricción que impone que la suma de los tiempos de las tareas asignadas a cada estación no supere el tiempo máximo de ciclo c , se refleja en la agrupación de nodos, ya que en cada zona la suma de los t_i es menor o igual a c .

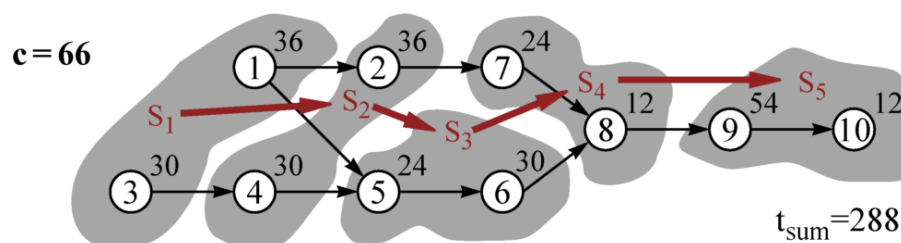


Figura 3.4: Ejemplo de MALBP.

Dadas las características del modelo, es de carácter NP-hard [22], lo que dificulta su resolución óptima para instancias de gran tamaño. A pesar de esto, se han trabajado métodos de resolución exacta utilizando técnicas de programación entera, branch and bounds y backtracking para encontrar la solución óptima [23]. Otras soluciones abarcan métodos de programación entera mixta [16] y heurísticas [20] como reglas de prioridad, métodos de escalada y búsquedas locales para encontrar soluciones factibles en tiempos reducidos. También se destacan metaheurísticas como algoritmos genéticos [24], búsqueda Tabú [12], algoritmo de colonia de hormigas [25] y enfoques híbridos que combinan métodos heurísticos y metaheurísticos [26].

3.2. Business Process Management

El Business Process Management (BPM), o *Gestión de Procesos de Negocio*, se define como una disciplina de gestión que combina las perspectivas del negocio y de la tecnología de información, con el objetivo de mejorar las operaciones y el desempeño organizacional [27]. En esencia, BPM constituye una filosofía de gestión respaldada por un conjunto de métodos, técnicas y herramientas para modelar, analizar, ejecutar y optimizar los procesos empresariales [28]. Dicho de otro modo, BPM involucra actividades como el diseño, la implementación, la medición y la configuración de procesos de negocio dentro de la organización [29]. El propósito fundamental es incrementar la eficacia y eficiencia de estos procesos, alineándolos con los objetivos estratégicos, lo que a su vez contribuye a un mejor rendimiento corporativo y mayor competitividad [27]. Gracias a este enfoque sistemático en los procesos, BPM se ha convertido en un factor clave para impulsar la innovación y la transformación organizacional, más allá de enfoques tradicionales centrados solo en la reducción de costos [27].

En [28] se muestra un panorama de la evolución de la disciplina, identificando temas centrales que han marcado el desarrollo de BPM, tales como los sistemas BPM, el modelado y diseño de procesos, la coordinación e interoperabilidad, la gestión de modelos, la minería de procesos y la incorporación de

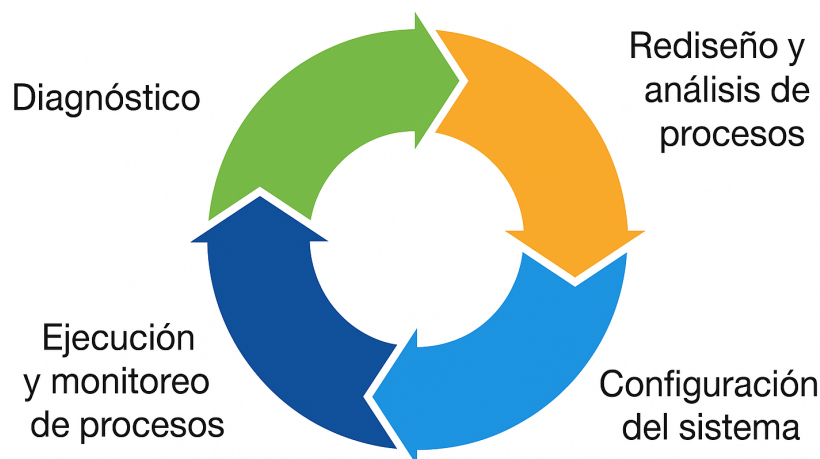


Figura 3.5: Ciclo de vida del BPM [29].

nuevas tecnologías. Estos temas reflejan cómo BPM ha ampliado su repertorio metodológico a lo largo del tiempo, consolidándose como una disciplina vigorosa y en expansión que integra múltiples enfoques para mejorar la gestión de procesos.

En [30] se analizó la producción científica desde 2021 en adelante, encontrando que los conceptos de *business process management*, *digital transformation* y *digitalization* aparecen estrechamente vinculados. Las relaciones entre estos tópicos subrayan la contribución teórica y práctica de nuevos métodos y modelos de BPM en el contexto de la transformación digital. La transformación digital se concibe como un proceso continuo, que involucra la reconfiguración de procesos de negocio mediante tecnologías digitales. En este contexto, BPM emerge como una pieza central para sostener nuevos modelos de negocio digitales, proporcionando un marco robusto sobre el cual apalancar iniciativas de digitalización. La investigación destaca además tendencias emergentes dentro de BPM asociadas a la transformación digital, incluyendo la gestión de datos administrativos, el modelado de procesos de negocio orientado a recursos y capacidades dinámicas, y la importancia de la agilidad organizacional. Todo ello sugiere que, en entornos empresariales altamente cambiantes, las prácticas de BPM siguen siendo esenciales para lograr organizaciones eficientes, escalables y adaptables, integrando estrechamente la gestión de procesos con la estrategia digital de la empresa.

Otro ámbito de interés reciente es la aplicación de Inteligencia Artificial (IA) en la gestión de procesos, dando lugar a enfoques de *BPM inteligente*. En [29] se realizó una revisión sistemática de la literatura sobre métodos de IA aplicados a BPM, identificando diversas técnicas que automatizan

tareas y apoyan la toma de decisiones en el ciclo de vida de los procesos (por ejemplo, en el diseño o la mejora de procesos). En los veintidós estudios analizados en dicha revisión, los autores encontraron métodos basados en machine learning y otras técnicas que ayudaron a los analistas a tomar decisiones más acertadas al rediseñar procesos. Entre las herramientas empleadas se destacan algoritmos de agrupamiento como K-means, redes Bayesianas para análisis predictivo, e incluso metaheurísticas de inteligencia de enjambre para optimizar operaciones complejas. Estos ejemplos ilustran cómo la IA se está incorporando en BPM para manejar grandes volúmenes de datos y tareas complejas en tiempo real, aumentando la capacidad de las organizaciones para automatizar procesos de forma inteligente y mejorar la eficiencia y flexibilidad de sus operaciones. La convergencia de BPM con técnicas de IA abre la puerta a sistemas de procesos de negocio más predictivos, adaptativos y proactivos, capaces de aprender y ajustar flujos de trabajo según patrones detectados en los datos.

Asimismo, la automatización robótica de procesos (Robotic Process Automation, RPA) se ha convertido en un complemento de BPM, con el enfoque de integrar robots de software dentro de la gestión de procesos. Por ejemplo, [31] propone una técnica para fusionar RPA con BPM de manera eficiente, sincronizando ambas tecnologías para lograr procesos empresariales altamente automatizados. La integración RPA-BPM busca automatizar las tareas repetitivas de un proceso con robots de software, al mismo tiempo que BPM proporciona la visión de conjunto y el control a largo plazo del proceso. Esta combinación ha mostrado potencial para reducir costos operativos (al eliminar actividades manuales rutinarias) y mejorar la rapidez y precisión de la ejecución de procesos.

El sector vitivinícola representa un caso particular donde la gestión de procesos adquiere relevancia para mejorar la competitividad y la sostenibilidad del negocio. En Chile, país con una importante industria del vino, se han documentado esfuerzos para modernizar la gestión de procesos en viñas y bodegas con el fin de enfrentar desafíos crecientes. En [32] se señala que las viñas chilenas afrontan una fuerte competencia global, además de dificultades como la volatilidad, altos costos energéticos y efectos del cambio climático. Su estudio reveló que, si bien la mayoría de las bodegas adopta estándares de calidad y cumplen con un código nacional de sustentabilidad, solo un 14 % de las empresas habían implementado algún tipo de Cuadro de Mando Integral (Balanced Scorecard) para gestionar su desempeño. Esta baja adopción indica una oportunidad importante para introducir herramientas de BPM más formales en el sector vitivinícola chileno, enfocadas en el monitoreo y mejora del desempeño de procesos con miras a la sostenibilidad y la creación de valor. De hecho, los autores proponen un modelo de Balanced Scorecard adaptado a las viñas chilenas como parte de un enfoque de BPM estratégico, incorporando dimensiones específicas de la industria, espaciales y temporales, para evaluar tanto indicadores financieros como no financieros. Este enfoque integrado permite alinear los procesos operativos (producción, logística y calidad) con los objetivos estratégicos de las bodegas, facilitando la

toma de decisiones basada en datos de proceso y la mejora continua.

Otro aspecto de la aplicación de BPM en el rubro vitivinícola es la gestión de la trazabilidad y la calidad a lo largo de la cadena de suministro del vino. La trazabilidad es fundamental para garantizar el origen de la uva, las condiciones de producción y la calidad del producto final, aspectos cada vez más valorados en los mercados. En [33] se desarrolla un modelo de referencia de procesos de trazabilidad específicamente orientado a la industria del vino. Este modelo, concebido bajo los principios de BPM, describe de manera estandarizada los procesos clave desde la producción en viñedos hasta la distribución, indicando cómo debería implementarse un sistema integral de trazabilidad del vino para satisfacer las necesidades de seguimiento eficiente del producto. Una contribución de dicho estudio es que el modelo incluye múltiples vistas de proceso (no solo el flujo de actividades, sino también los datos, los sistemas, los riesgos y la estructura organizacional relacionada). Además, adopta estándares internacionales como GS1 EPCIS para asegurar la compatibilidad de los datos de trazabilidad. Este enfoque integrado sirve como guía para las bodegas y sus socios logísticos, delineando las funciones que un sistema de trazabilidad debe cumplir y cómo interactúa con los diferentes actores de la cadena de valor del vino. Esto demuestra cómo las prácticas de BPM en el sector vitivinícola están abrazando las tecnologías emergentes para garantizar la transparencia, la seguridad y la sostenibilidad en la producción vitivinícola.

Capítulo 4

Diseño de la Solución

Este capítulo presenta el diseño de la solución tecnológica propuesta para abordar los desafíos operativos asociados al proceso de recepción de uva durante la vendimia. La solución propuesta se materializa en el desarrollo de una aplicación digital con arquitectura modular y una interfaz de usuario intuitiva, orientada a facilitar la supervisión y la toma de decisiones estratégicas informadas. El diseño de esta herramienta considera tanto los requerimientos funcionales del sistema como los criterios de usabilidad, escalabilidad y mantenibilidad, con el objetivo de integrarse adecuadamente a las dinámicas de trabajo existentes.

Además, se detallan los principios que guían el diseño de la solución, la estructura general del sistema y sus componentes principales. Asimismo, se justifica la elección de una arquitectura modular como enfoque que permite un desarrollo incremental, facilita la integración con otros sistemas y garantiza una mayor adaptabilidad ante cambios futuros.

4.1. Requisitos del Sistema

Para representar de forma precisa el proceso de recepción de uva, se definieron un conjunto de requerimientos técnicos y operacionales obtenidos a partir de visitas en terreno, entrevistas con el personal encargado del proceso de vendimia, análisis de datos históricos y referencias de casos de estudios similares [34, 35]. Debido a la ausencia de documentación técnica sistematizada, la información relevante fue recopilada directamente desde la experiencia de los operarios y supervisores de planta.

La caracterización del sistema se estructuró siguiendo un enfoque metodológico basado en etapas de definición, medición, análisis, mejora y control. A partir de esta estructura, se identificaron las

funcionalidades necesarias, las condiciones de entorno y las especificaciones de interfaz requeridas para el desarrollo de una herramienta digital orientada a la simulación y visualización del proceso productivo. Los parámetros extraídos permiten representar tanto el comportamiento individual de cada máquina como la interacción entre componentes dentro de las distintas líneas de producción.

4.1.1. **Requerimientos Funcionales**

Como primer paso en el diseño de la solución, se establecen los requerimientos funcionales que debe cumplir el sistema para responder adecuadamente a las necesidades operativas detectadas en el proceso de recepción de uva. Estos requerimientos definen las capacidades esenciales que la herramienta debe ofrecer a los usuarios, permitiendo tanto la visualización integral de los flujos de trabajo como la posibilidad de simular y ajustar escenarios operacionales. Además, se incorporan funciones orientadas a la personalización de parámetros y componentes, así como mecanismos de seguridad para proteger la integridad de la información. En la Tabla 4.1 se detallan los principales requerimientos funcionales identificados para el sistema propuesto.

Código	Requerimiento Funcional
RF1	El usuario debe ser capaz de visualizar íntegramente los flujos de la viña.
RF2	El usuario debe ser capaz de simular escenarios operacionales.
RF3	El sistema debe almacenar escenarios operacionales simulados previamente por el usuario.
RF4	El usuario debe ser capaz de cambiar los componentes de los flujos de recepción de la uva.
RF5	El usuario debe ser capaz de cambiar los parámetros de los componentes de los flujos.
RF6	El sistema debe poseer autenticación de usuarios para darle seguridad a la información.

Tabla 4.1: Requerimientos funcionales del sistema.

4.1.2. **Requerimientos de Interfaces**

La definición de los requerimientos de interfaces permite especificar las interacciones esperadas entre el usuario y el sistema. Estos requerimientos se describen mediante eventos que caracterizan las acciones posibles dentro de la aplicación, indicando el iniciador de cada evento, los parámetros necesarios y las respuestas que el sistema debe proporcionar. Esta especificación busca asegurar una correspondencia clara entre las funcionalidades del sistema y las operaciones que el usuario puede realizar durante su uso. En la Tabla 4.2, se presentan los eventos principales asociados a la interfaz de usuario de la herramienta propuesta.

Atributo	Iniciar Sesión	Cargar Simulación	Nueva Simulación	Editar Parámetro	Eliminar Simulación
Descripción	Al ingresar a la aplicación el usuario debe ingresar sus credenciales para autenticarse.	Al ingresar a la aplicación el usuario puede cargar una simulación diseñada previamente.	Al ingresar a la aplicación el usuario puede crear una nueva simulación.	El usuario puede editar los parámetros de los distintos componentes de la línea.	El usuario escoge una simulación hecha anteriormente y la elimina.
Iniciador	Usuario escribe sus credenciales.	Usuario elige la simulación a cargar.	Seleccionar botón para nueva simulación.	Usuario ingresa parámetro a editar.	Seleccionar simulación y luego botón para borrar.
Parámetro	Usuario y contraseña.	Simulación a cargar.	Escoger línea a simular.	Valor numérico del nuevo parámetro.	Simulación a borrar.
Respuesta	Acceso a la aplicación.	Carga de simulación seleccionada.	Se crea una simulación nueva en la línea escogida.	Actualización de la simulación.	Se elimina la simulación escogida.

Tabla 4.2: Requerimientos de interfaces del sistema.

4.1.3. Requerimientos de Ambiente

El desarrollo y la ejecución de sistemas digitales requieren condiciones específicas en términos de hardware y software, tanto para los entornos de desarrollo como para los entornos de producción. Esta sección detalla las características del ambiente utilizado durante la fase de desarrollo local de la herramienta, con el fin de establecer un marco de referencia técnico que permita replicar o adaptar el entorno según las necesidades de implementación futura.

Hardware de Desarrollo

El entorno de desarrollo local utilizado para la implementación de la herramienta se ejecutó sobre una estación de trabajo equipada con los siguientes componentes de hardware:

- **CPU:** AMD Ryzen 5 2600, de seis núcleos y 12 hilos.
- **RAM:** 8GB, suficiente para ejecutar los servicios de backend y frontend en paralelo, incluyendo el uso de contenedores, y bases de datos.

Estos recursos permitirán realizar pruebas locales de la aplicación bajo condiciones representativas, habilitando la ejecución de servicios interdependientes en un entorno controlado. Durante la etapa de desarrollo, se deben emplear contenedores para orquestar los servicios del sistema, lo que incrementa puntualmente el uso de memoria y procesamiento, especialmente en operaciones de reconstrucción de imágenes o sincronización de volúmenes. A pesar de estas exigencias, el hardware disponible mantendrá un comportamiento estable, sin generar cuellos de botella críticos. Cabe señalar que, si bien este entorno ofrece un margen adecuado para el desarrollo, se estima que los requerimientos mínimos para la ejecución del sistema en producción son considerablemente menores, particularmente si se opta por

políticas de despliegue distribuido o la utilización de servicios escalables en la nube.

4.2. Arquitectura del Sistema

En este apartado se describe la arquitectura general del sistema propuesto, especificando su estructura organizacional, los componentes principales y su forma de interacción. El diseño arquitectónico busca garantizar la modularidad, escalabilidad y mantenibilidad del sistema, permitiendo su adaptación a diferentes entornos de despliegue y facilitando su evolución futura. La solución se basa en un enfoque distribuido, orientado a servicios, que permite desacoplar las funcionalidades clave y promover una implementación flexible.

4.2.1. Diagrama de Contexto

El diseño propuesto se concibe como una aplicación desplegada en un entorno basado en servicios distribuidos, alojado en una infraestructura en la nube. En el diagrama de contexto, véase la Figura 4.1, se representa de manera abstracta la interacción entre los distintos actores del sistema y sus componentes principales. El sistema está compuesto por un servicio web que se comunica con componentes de almacenamiento y modelamiento de procesos. Esta disposición permite separar las responsabilidades funcionales y facilita la implementación de mecanismos de control, autenticación y persistencia de datos.

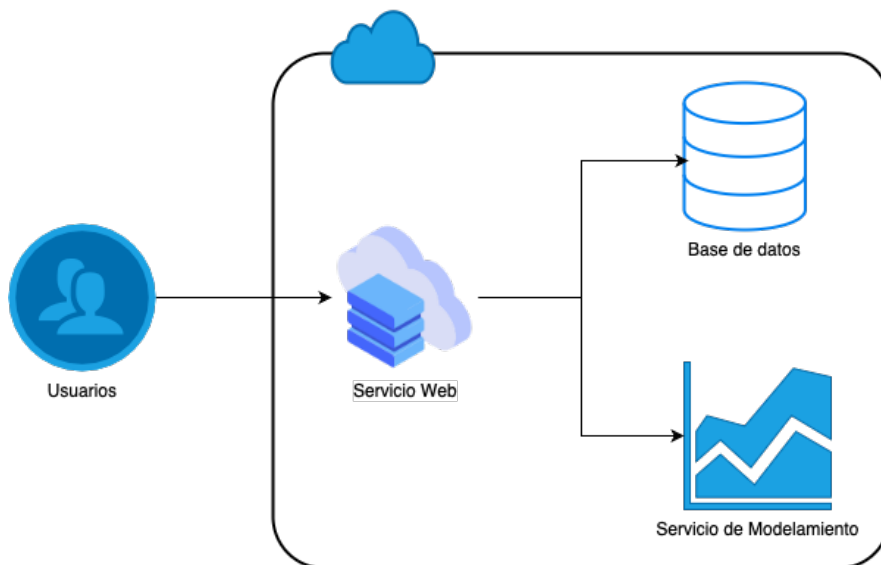


Figura 4.1: Esquema del sistema.

4.2.2. Diagrama de Arquitectura

El diseño arquitectónico del sistema se estructura en base a un enfoque de microservicios, lo que permite organizar la aplicación como un conjunto de módulos autónomos que colaboran entre sí a través de interfaces bien definidas. Cada módulo representa una funcionalidad específica del sistema, lo cual facilita la separación de responsabilidades, la reutilización de componentes y la independencia en el ciclo de vida de desarrollo e implementación.

En la Figura 4.2 se muestra un esquema general de la arquitectura interna propuesta. Los distintos módulos se encuentran desacoplados y expuestos mediante protocolos de comunicación estandarizados, lo cual permite su ejecución coordinada en entornos distribuidos o en infraestructura basada en contenedores. La interacción entre servicios se realiza a través de una red lógica que permite el intercambio de datos entre los distintos componentes del sistema, incluyendo la capa de presentación, los servicios de negocio y los mecanismos de persistencia.

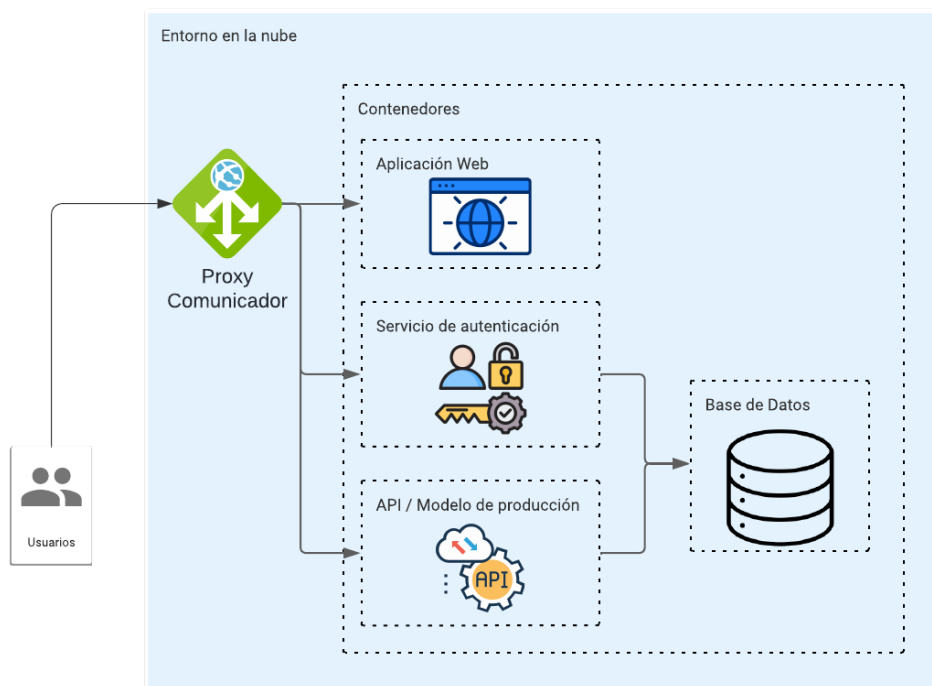


Figura 4.2: Diagrama de Arquitectura generalizado del sistema.

Este enfoque modular facilita la implementación de estrategias de escalabilidad horizontal, la actualización independiente de componentes, y la integración con herramientas de monitoreo, pruebas automatizadas y despliegue continuo. Los detalles específicos de las tecnologías empleadas en cada

módulo, así como la infraestructura utilizada para su implementación y orquestación, se detallan en secciones posteriores.

4.2.3. Descripción de Componentes

El sistema propuesto se compone de distintos elementos funcionales organizados en módulos que interactúan entre sí para permitir la operación integral de la herramienta. Cada componente cumple un rol específico dentro de la arquitectura general, abarcando desde la interacción con el usuario hasta el procesamiento de datos y la persistencia de información. La siguiente enumeración presenta una descripción general de los principales componentes involucrados en el sistema:

1. **Usuario:** El usuario puede acceder a la plataforma desde cualquier dispositivo que soporte navegación web, como lo pueden ser computadoras o teléfonos inteligentes.
2. **Entorno en la nube:** Todos los servicios necesarios para el funcionamiento del sistema serán alojados en un servidor dedicado del cliente. Otra opción viable es optar por un servicio cloud que permita asegurar la disponibilidad y escalabilidad de la herramienta en términos computacionales, evitando generar gastos operacionales para el cliente.
 - a) **Servicio Web (Módulo 1):** El apartado de presentación de interfaz al usuario corresponderá a un servicio web bajo protocolos seguros. Esta interfaz permitirá al usuario autenticado efectuar cambios de parámetros a las líneas existentes y mostrará los resultados del flujo según los modelos de flujo definidos.
 - b) **Servicio de modelamiento (Módulo 2):** Este microservicio es el responsable de realizar los cálculos dados por los modelos predefinidos, para luego entregar los resultados a la interfaz.
 - c) **Base de datos (Módulo 3):** Almacenamiento de datos necesarios por la interfaz como parámetros predefinidos de maquinarias conocidas y datos de autenticación de usuarios.

4.2.4. Matriz de requisitos funcionales y componentes

Con el objetivo de establecer una trazabilidad entre los requerimientos funcionales definidos previamente y los módulos del sistema que los satisfacen, se presenta la matriz de la Tabla 4.3. Esta matriz permite identificar de manera explícita qué componente del sistema está asociado a la implementación de cada funcionalidad, contribuyendo a una planificación coherente del desarrollo, pruebas y validación del sistema.

Componente	RF1	RF2	RF3	RF4	RF5	RF6
Usuario	X	X		X	X	
M1: Servicio Web	X	X		X	X	X
M2: Servicio de Modelamiento	X	X				
M3: Base de Datos		X	X			X

Tabla 4.3: Matriz de requisitos funcionales y componentes.

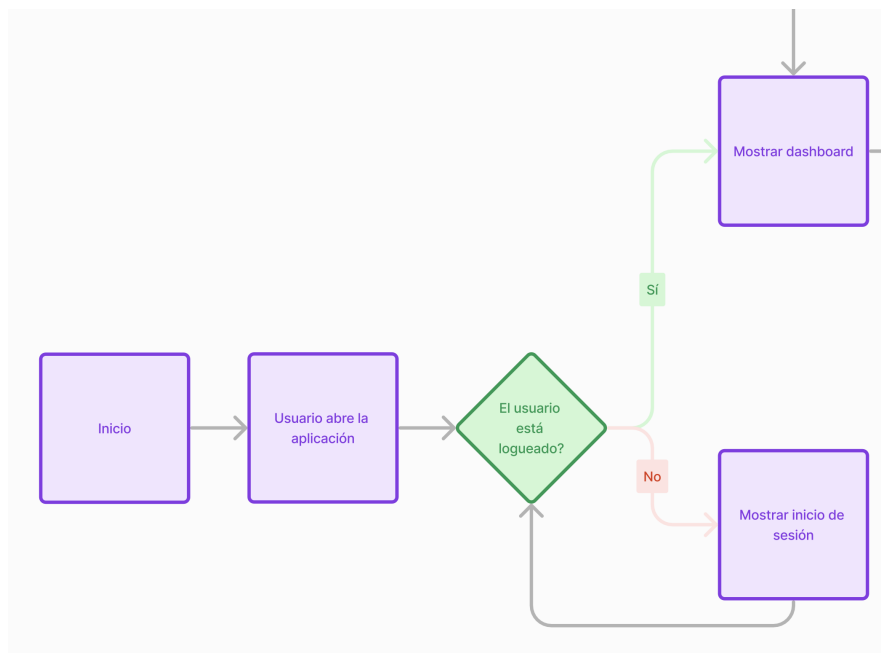
4.3. Diseño de interfaces

El diseño de la interfaz gráfica permite a los usuarios interactuar con las distintas funcionalidades del sistema a través de vistas específicas y un flujo de navegación definido. La organización de estas vistas considera las acciones principales que puede realizar un usuario, como iniciar sesión, configurar una simulación, revisar resultados anteriores o visualizar el estado actual de una simulación en curso. Para guiar la implementación, se elaboran *prototipos visuales* que representan la disposición y el comportamiento esperado de cada componente. A continuación, se expone el recorrido que realiza el usuario dentro de la aplicación y las características principales de cada vista.

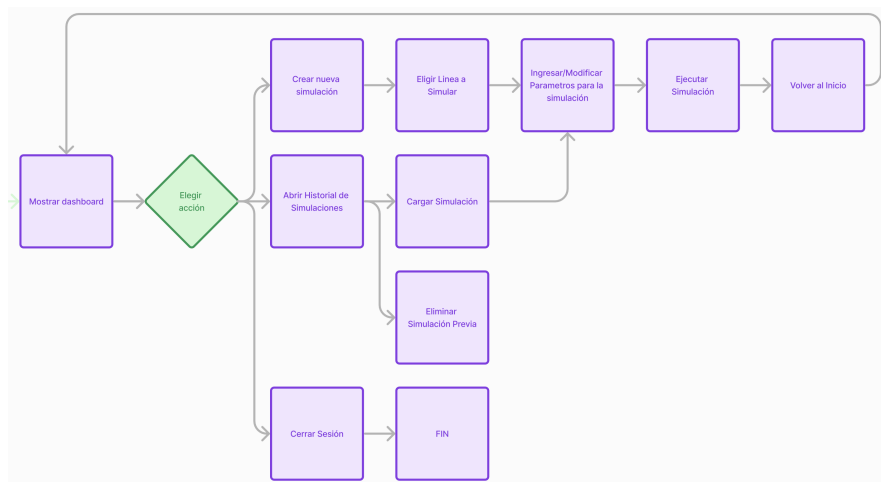
4.3.1. Modelo de navegación

El modelo de navegación de la aplicación inicia con un proceso de autenticación, véase la Figura 4.3a). En primer lugar, el sistema verifica si el usuario ya se encuentra autenticado; de ser así, es redirigido automáticamente hacia el dashboard principal. En caso contrario, se despliega una interfaz de inicio de sesión en la que el usuario debe proporcionar sus credenciales. Una vez que se autentica exitosamente, el sistema continúa el flujo habitual dirigiendo al usuario al dashboard, garantizando así un control de acceso eficiente y seguro.

Tras finalizar el proceso de autenticación, la aplicación ofrece al usuario un conjunto de acciones orientadas a la gestión de simulaciones, véase Figura 4.3b). El usuario tiene la posibilidad de crear una nueva simulación, seleccionar la línea de proceso que desea modelar, ingresar o ajustar parámetros, ejecutar la simulación correspondiente y finalmente retornar al menú inicial. De manera alternativa, la plataforma permite acceder al historial de simulaciones, desde donde el usuario puede cargar simulaciones previamente realizadas o eliminarlas. Asimismo, se incluye una opción específica para cerrar sesión, concluyendo así la interacción con el sistema de forma segura.



(a) Flujo de autenticación.



(b) Flujo de la herramienta.

Figura 4.3: Flujos de autenticación y de uso de la herramienta.

En la Figura 4.4 se muestra el flujo completo de la aplicación, uniendo la autenticación con el uso de la herramienta.

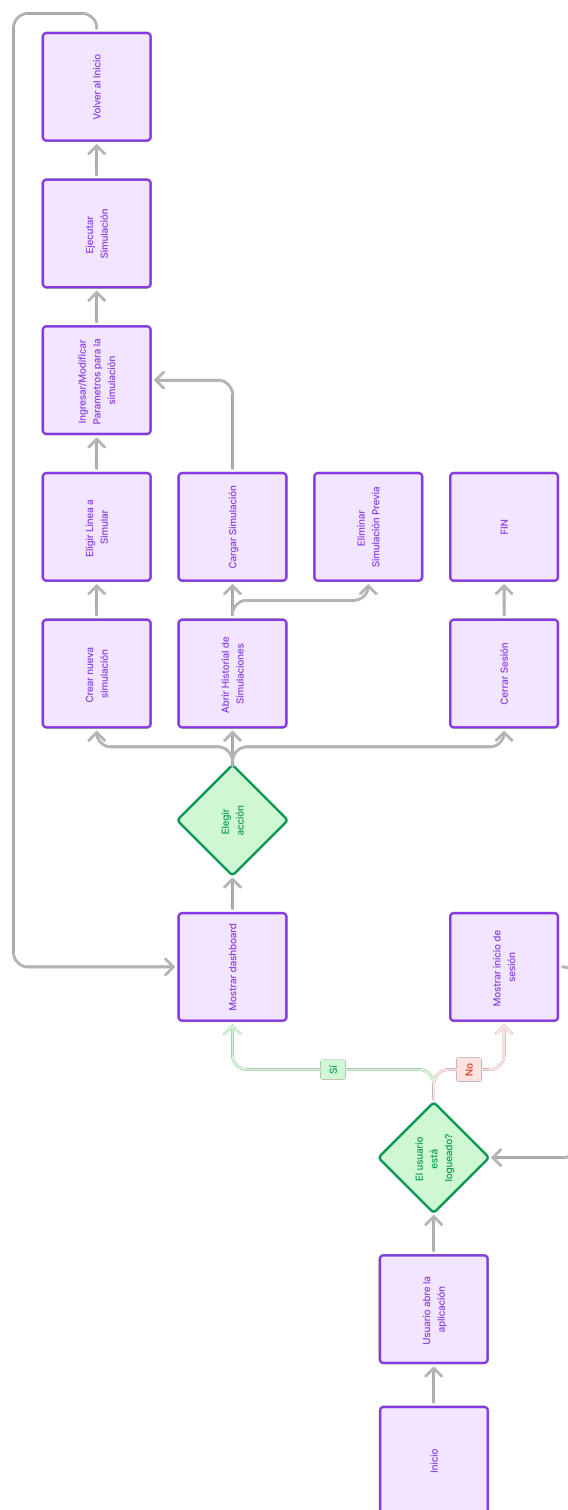


Figura 4.4: Flujo completo de la aplicación.

4.3.2. Prototipo de Interfaces Usuarias

Las interfaces usuarias fueron prototipadas en primera instancia creando un Mock-Up utilizando Figma [36]. A continuación, se describen las principales vistas:

- **Vista de Inicio de Sesión:** Permite al usuario ingresar su correo y contraseña para autenticarse en el sistema. Incluye opciones para recuperar la contraseña en caso de olvido. En la Figura 4.5 se puede ver la pantalla de autenticación para el ingreso a la herramienta, con las casillas respectivas para el ingreso de las credenciales.

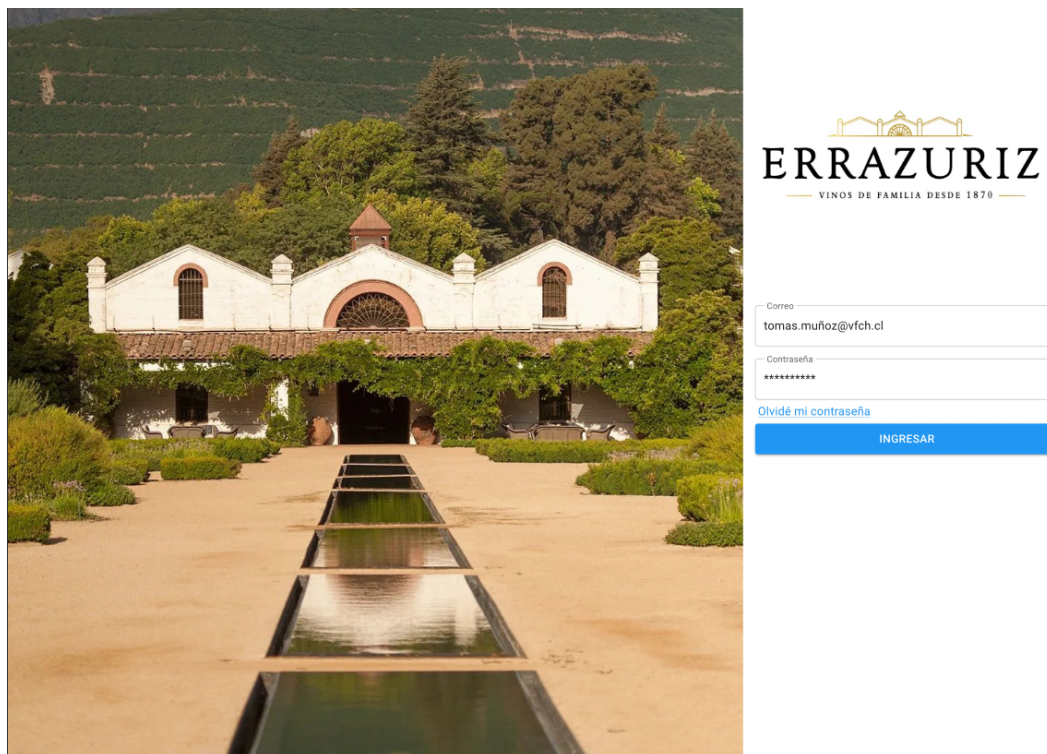


Figura 4.5: Mock-Up Inicio de Sesión de la aplicación.

- **Dashboard:** Una vez autenticado, el usuario accede a un panel principal donde puede ver un resumen de sus simulaciones recientes y opciones para crear nuevas simulaciones o modificar las existentes. También, proporciona una vista detallada de la simulación en curso, permitiendo al usuario pausar, reanudar, o detener la simulación, así como visualizar alertas y resultados intermedios. En la Figura 4.6 se muestra el dashboard de la aplicación, contando en el centro de la pantalla con la simulación en sí, en la que el usuario puede modificar los parámetros a simular, pausar/reanudar la simulación y obtener alertas significativas para la toma de decisiones.

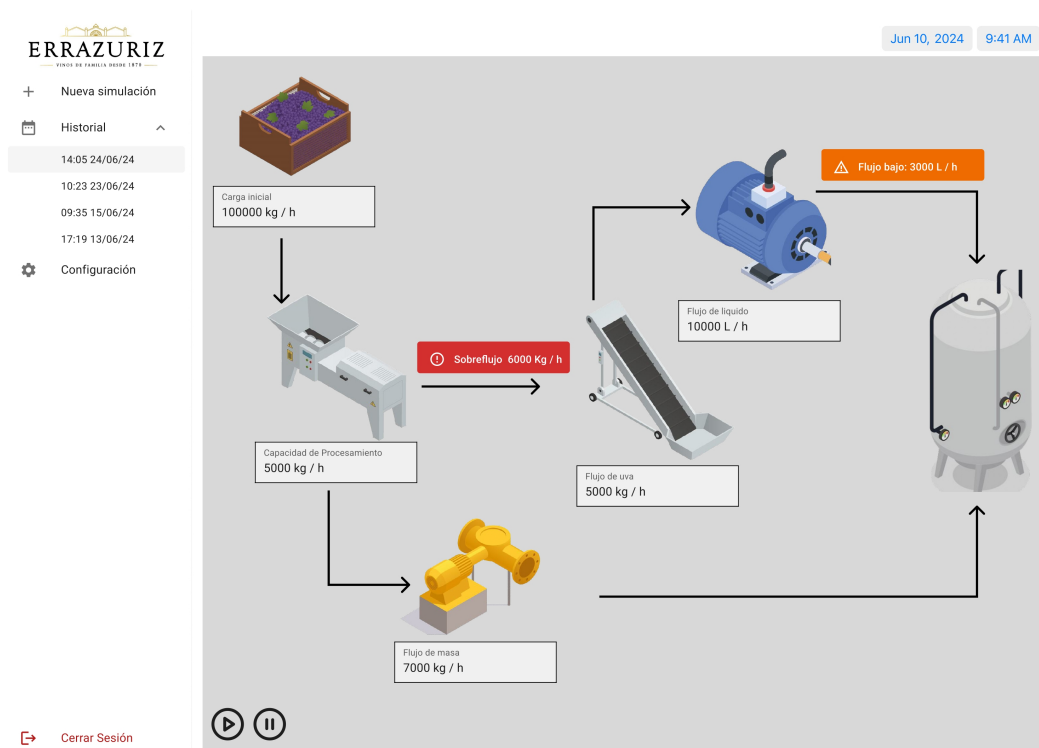
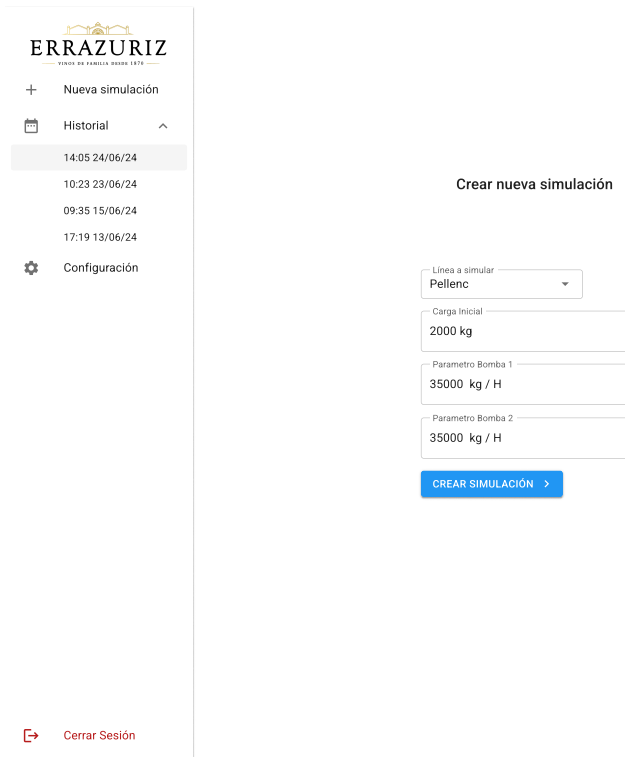


Figura 4.6: Mock-Up Dashboard de la aplicación.

- **Configuración de simulación:** En esta pantalla, el usuario puede ingresar o modificar parámetros específicos para la simulación, como la carga inicial, el flujo de líquido, y otros parámetros críticos. En la Figura 4.7 se ve la pantalla para la creación de una nueva simulación, donde el usuario puede elegir la línea a simular, y luego ingresar los parámetros de las máquinas a simular.



El mock-up muestra la interfaz de usuario para crear una nueva simulación. A la izquierda hay un menú lateral con el logo de 'ERRAZURIZ' y las opciones: '+ Nueva simulación', 'Historial' (con un icono de calendario y una flecha hacia arriba), una lista de fechas (14:05 24/06/24, 10:23 23/06/24, 09:35 15/06/24, 17:19 13/06/24) y 'Configuración' (con un icono de engranaje). En la parte inferior del menú está 'Cerrar Sesión' con un icono de salida. El área principal se titula 'Crear nueva simulación' y contiene los siguientes campos: 'Línea a simular' (menú desplegable con 'Pellenc' seleccionado), 'Carga Inicial' (campo de texto con '2000 kg'), 'Parametro Bomba 1' (campo de texto con '35000 kg / H') y 'Parametro Bomba 2' (campo de texto con '35000 kg / H'). Debajo de estos campos hay un botón azul que dice 'CREAR SIMULACIÓN' con una flecha hacia la derecha.

Figura 4.7: Mock-Up Formulario de nueva simulación.

- **Historial de simulaciones:** Muestra un listado de las simulaciones realizadas previamente con detalles como fecha y hora de ejecución, permitiendo al usuario revisar resultados pasados. Este historial se muestra en el costado izquierdo de la Figura 4.6 y la Figura 4.7.

El diseño busca ser intuitivo y facilitar la navegación y el uso de la aplicación, asegurando que los usuarios puedan interactuar de buena manera con las funcionalidades principales del sistema, principalmente con la simulación. Puntualmente, con respecto al diseño de interfaces, se ha realizado con una mirada ingenieril, desde la experiencia de desarrollo de software e interfaces usuarias.

Capítulo 5

Implementación y Despliegue de la Solución

En este capítulo se presenta la versión funcional de la herramienta de simulación de flujo de recepción de uva, incluyendo una descripción exhaustiva de sus componentes. Se aborda el entorno de desarrollo y de producción utilizado, la estructura de la base de datos, la interfaz de comunicación entre la base de datos y el front-end, el sistema de autenticación y las vistas principales de la herramienta. Cada apartado proporciona una visión detallada de cómo estos elementos trabajan en conjunto para ofrecer una experiencia de usuario eficiente y segura.

5.1. Caso de estudio

Para la implementación de la herramienta digital propuesta en este trabajo, se ha seleccionado la línea de producción denominada “Ícono” de Viña Errazuriz. Esta decisión se fundamenta en diversas razones técnicas y operativas detalladas en capítulos anteriores, que hacen de esta línea una candidata adecuada para la simulación y evaluación del desempeño del proceso.

La línea Ícono representa la configuración tecnológica más avanzada dentro de la viña, empleando equipos especializados de última generación para procesar uvas destinadas exclusivamente a vinos tintos de alta calidad. Su disposición en serie, integrada por máquinas sofisticadas de la marca Pellenc [37] importadas desde Francia, permite un control preciso y exhaustivo sobre variables críticas del proceso, tales como la velocidad de procesamiento, la separación eficaz del escobajo, la calidad del mosto obtenido y la detección automática de uvas defectuosas.

Además, esta línea se caracteriza por manejar un volumen moderado de uva con un alto valor comercial, lo que incrementa significativamente la relevancia económica y estratégica de identificar y gestionar eficientemente posibles cuellos de botella. Un retraso o falla en cualquier etapa del proceso puede resultar en pérdidas sustanciales tanto en términos de producción como en calidad final del vino.

Finalmente, debido a su avanzada tecnología y estructura modular, la línea Ícono ofrece condiciones ideales para implementar simulaciones computacionales precisas, que permitirán validar la efectividad de la herramienta propuesta en escenarios realistas y controlados. La complejidad técnica y la detallada instrumentación presentes en esta línea proporcionan una base sólida para evaluar la aplicabilidad general de la solución planteada y extrapolar los resultados obtenidos a otras líneas dentro de la misma viña o similares en otras instalaciones vitivinícolas.

5.2. Implementación de la Arquitectura

5.2.1. Entorno de Desarrollo Local

Durante la implementación de la herramienta de simulación se utilizó un entorno de desarrollo orientado a maximizar la productividad, facilitar la integración con herramientas de control de versiones y soportar múltiples lenguajes de programación. En este contexto, se empleó el editor de texto **Visual Studio Code** [38], una herramienta de código abierto ampliamente adoptada en la industria. Entre sus principales características se encuentran la depuración integrada, compatibilidad con una gran variedad de extensiones, soporte nativo para Git [39] y una interfaz liviana y multiplataforma.

Durante el proceso de selección se consideraron otras alternativas como **Sublime Text** [40] y **Atom** [41]. Sublime Text destaca por su velocidad y bajo consumo de recursos, sin embargo, no incluye de forma nativa funcionalidades clave como depuración o integración directa con sistemas de control de versiones. Atom, desarrollado originalmente por GitHub, ofrecía una experiencia altamente personalizable y orientada al desarrollo colaborativo, pero su discontinuación oficial en 2022 limitó su viabilidad a largo plazo [42].

También se evaluó **IntelliJ IDEA** [43], una opción robusta que proporciona herramientas avanzadas para análisis estático, refactorización y depuración. No obstante, su enfoque principal está orientado al desarrollo en entornos empresariales y en lenguajes como Java o Kotlin, lo que lo hace menos adecuado para proyectos centrados en tecnologías web ligeras como los empleados en esta simulación.

La elección de Visual Studio Code se fundamentó en su equilibrio entre rendimiento, extensibilidad

y facilidad de uso, convirtiéndolo en una solución eficaz para el desarrollo modular de esta herramienta.

5.2.2. Entorno DevOps en la Nube

El entorno de producción de la herramienta de simulación fue desplegado en una máquina virtual configurada en **DigitalOcean** [44], lo que permitió disponer de un entorno controlado, escalable y seguro para alojar tanto la API como la interfaz web. Esta configuración proporciona una infraestructura flexible que permite administrar el sistema operativo, los servicios de red y los recursos computacionales según los requerimientos específicos del sistema. En la planificación del entorno de despliegue se consideraron diferentes enfoques de infraestructura:

- **VPS (Servidor Privado Virtual):** como el ofrecido por DigitalOcean o Linode, permite acceso completo al sistema operativo, configuraciones personalizadas y control granular sobre los servicios desplegados.
- **Plataformas como Servicio (PaaS):** como Heroku o Google App Engine, simplifican el despliegue mediante abstracción del sistema operativo, pero limitan el control sobre configuraciones específicas del entorno.
- **Infraestructura basada en contenedores:** como AWS ECS, Kubernetes o Docker Swarm, ideal para arquitecturas complejas o distribuidas, pero requieren mayor inversión en configuración inicial y mantenimiento.

Dado el alcance de la herramienta y su arquitectura monolítica moderada, un entorno basado en VPS ofrecía el equilibrio ideal entre control, simplicidad de operación y costos operativos. Durante la evaluación de proveedores, se consideraron opciones como:

- **AWS EC2** [45]: ofrece escalabilidad y servicios avanzados, pero presenta mayor complejidad en la configuración y un modelo de precios menos predecible.
- **Google Compute Engine (GCE)** [46]: altamente integrable con otras soluciones de Google Cloud, aunque con una curva de aprendizaje más pronunciada.
- **Heroku** [47] y **Vercel** [48]: simplifican el despliegue para aplicaciones web, pero restringen el acceso a configuraciones del sistema operativo y requieren adaptaciones al modelo de despliegue propuesto.

DigitalOcean fue seleccionado por su modelo de servidor virtual (Droplets), que permite aprovisionar instancias con recursos dedicados, acceso root, configuraciones personalizadas y soporte para automatización mediante scripts. Además, ofrece una interfaz simplificada, tiempos de despliegue

reducidos y costos fijos por hora o por mes, lo que facilita la planificación del presupuesto. Las características específicas de esta configuración son las siguientes:

- **Máquina Virtual (Droplet):** La máquina virtual de Digital Ocean cuenta con 1 vCPU y 1 GB de memoria RAM, una configuración adecuada para manejar las operaciones de la herramienta en un entorno de baja a moderada carga de usuarios. Este recurso fue habilitado para albergar el código ejecutable (back-end), gestionar la base de datos y soportar la interfaz de usuario (front-end) de manera eficiente, además de tener la capacidad de escalar verticalmente en caso de que se estime conveniente.
- **Almacenamiento:** La droplet tiene 25 GB de almacenamiento, espacio necesario para almacenar datos de usuario, registros de simulación, y archivos de configuración necesarios.
- **Firewall:** El firewall está configurado para permitir únicamente el tráfico esencial, asegurando que solo los puertos necesarios para el funcionamiento de la herramienta estén abiertos:
 - **Puerto 80 (HTTP):** Permite el acceso a la herramienta mediante conexiones HTTP, facilitando el acceso a contenido no cifrado si se requiere para desarrollo o pruebas iniciales.
 - **Puerto 443 (HTTPS):** Habilitado para conexiones HTTPS, asegurando que los datos transmitidos entre el usuario y el servidor estén cifrados, lo que protege la privacidad y seguridad de los usuarios.
 - **Puerto 22 (SSH):** Reservado para el acceso SSH seguro, este puerto permite la gestión y administración remota de la máquina virtual, con el fin de realizar tareas de mantenimiento, actualizaciones y monitoreo de recursos.

5.2.3. Proxy Comunicador

Con el objetivo de garantizar la seguridad, accesibilidad y estabilidad operativa de la herramienta en el entorno de producción, se implementó una arquitectura basada en componentes específicos para la gestión del tráfico, el cifrado de datos y la administración del dominio. A continuación se detallan las tecnologías empleadas, junto con las alternativas evaluadas y los criterios que motivaron su selección:

- **Proxy inverso: Nginx.** Para gestionar las solicitudes entrantes, se configuró **Nginx** como proxy inverso en la máquina virtual [49]. Esta tecnología permite distribuir el tráfico hacia los servicios internos, implementar reglas de redireccionamiento, aplicar medidas de seguridad en las cabeceras HTTP y mejorar el rendimiento general del sistema. Durante el análisis previo, se consideraron alternativas como **Apache HTTP Server** [50] y **HAProxy** [51]. Apache, aunque ampliamente

utilizado, presenta una configuración más compleja y menor eficiencia al actuar exclusivamente como proxy inverso. HAProxy, por su parte, ofrece alto rendimiento para balanceo de carga, pero carece de la flexibilidad de configuración a nivel de contenido y rutas que Nginx proporciona, lo cual fue clave para los requisitos del sistema.

- **Certificación SSL: Let's Encrypt.** Para permitir conexiones seguras mediante HTTPS, se utilizó **Let's Encrypt** [52], una autoridad de certificación gratuita que emite certificados SSL/TLS válidos y reconocidos por los principales navegadores. Alternativas como **Cloudflare SSL** [53] y **Comodo SSL** [54] fueron consideradas. Cloudflare ofrece certificados como parte de su red CDN, pero su implementación requiere delegar la gestión del tráfico a sus servidores, lo que implicaba una complejidad adicional y dependencia externa. Comodo SSL, aunque robusto, requiere una suscripción paga y procesos manuales de renovación. Let's Encrypt fue seleccionada por su automatización con herramientas como **certbot**, su integración directa con Nginx y su confiabilidad general.
- **Sistema de nombres de dominio: Tech Domains.** Se utilizó el proveedor **Tech Domains** [55] para la gestión del DNS asociado al dominio de la herramienta. A través de este servicio, se configuraron registros **A** y **CNAME** que permiten la resolución rápida del dominio hacia la instancia de Digital Ocean. Como alternativas se evaluaron **GoDaddy** [56] y **Namecheap** [57]. GoDaddy ofrece una interfaz consolidada, pero su panel de configuración resulta menos ágil en comparación con otros proveedores, además de tener costos más elevados. Namecheap brinda un servicio competitivo, aunque con menos soporte para automatización vía API. Tech Domains fue seleccionada por su facilidad de uso, tiempos de propagación rápidos y su integración fluida con los servicios existentes.

5.2.4. Base de Datos

La elección del sistema de gestión de base de datos es un aspecto fundamental en el diseño de sistemas informáticos, ya que condiciona la forma en que se modelan, almacenan y consultan los datos a lo largo del ciclo de vida de la aplicación. En el caso de este desarrollo, se opta por un enfoque no relacional, seleccionando específicamente **MongoDB** [58] como sistema de base de datos.

Las bases de datos relacionales, como **PostgreSQL** [59] o **MySQL** [60], organizan la información en tablas con esquemas predefinidos y relaciones bien estructuradas entre entidades. Este enfoque es eficaz para sistemas con estructuras de datos estables y requerimientos estrictos de integridad referencial. Sin embargo, su rigidez estructural puede representar una limitación en contextos donde se requiere modificar con frecuencia la estructura de los datos, como ocurre en esta herramienta, donde el modelo

puede evolucionar para adaptarse a nuevas máquinas, parámetros o configuraciones técnicas definidas por el usuario.

Por otro lado, las bases de datos no relacionales o *NoSQL* se caracterizan por su flexibilidad en el esquema, escalabilidad horizontal y eficiencia en la manipulación de datos semi-estructurados o no estructurados. Este paradigma resulta particularmente útil en aplicaciones en constante evolución, donde la agilidad en el desarrollo y la capacidad de adaptarse a nuevas estructuras de datos sin migraciones costosas es un requisito esencial.

En el contexto específico de esta herramienta de simulación, la capacidad de trabajar con documentos de estructura variable permite modelar las distintas configuraciones de maquinaria, parámetros dinámicos y resultados de simulaciones sin necesidad de redefinir continuamente un esquema fijo. Esta característica reduce la complejidad asociada a los cambios estructurales y facilita la evolución funcional del sistema, en línea con metodologías de desarrollo ágil.

Entre las opciones de bases de datos no relacionales consideradas se incluyen:

- **MongoDB** [58], orientada a documentos y con un modelo flexible basado en JSON (BSON internamente), permite almacenar datos jerárquicos y consultar colecciones con gran eficiencia.
- **Apache Cassandra** [61], diseñada para manejar grandes volúmenes de datos distribuidos con alta disponibilidad y tolerancia a fallos. Su modelo de escritura es altamente eficiente, pero su curva de aprendizaje y su enfoque orientado a disponibilidad la hacen menos adecuada para sistemas con requerimientos dinámicos en estructura de datos.
- **Firebase Realtime Database** [62], orientada al desarrollo móvil y en tiempo real, permite sincronización inmediata entre cliente y servidor, pero presenta limitaciones en la estructura de consultas complejas y dependencia del ecosistema Google.

La elección de MongoDB se fundamenta en su flexibilidad de esquema, su amplia adopción en la industria, el soporte nativo para consultas complejas y agregaciones, así como su facilidad de integración con herramientas modernas de desarrollo. Además, ofrece herramientas de escalamiento horizontal, replicación y respaldo que permiten mantener la confiabilidad del sistema en entornos productivos.

5.2.5. API

El desarrollo de la API se diseñó considerando criterios de rendimiento, escalabilidad y facilidad de mantenimiento. La elección tecnológica abarcó tanto el lenguaje de programación como el framework para la implementación de servicios web. Entre los lenguajes evaluados para construir la API se

encuentran:

- **Python** [63]: ampliamente utilizado en ciencia de datos, automatización y desarrollo web, destaca por su sintaxis clara, comunidad activa y amplia variedad de librerías.
- **JavaScript (Node.js)** [64]: orientado a eventos, permite construir APIs asincrónicas con alto rendimiento, aunque su sistema de tipado dinámico puede dificultar el control de errores.
- **Java** [65]: robusto, compilado y con alta performance, pero presenta una curva de aprendizaje y tiempos de desarrollo mayores para proyectos medianos o ligeros.
- **Go** [66]: lenguaje compilado, eficiente y ligero, ideal para microservicios, pero con menor ecosistema en torno a desarrollo rápido de APIs.

Dado el enfoque modular y el ciclo iterativo del desarrollo, se optó por **Python** por su equilibrio entre facilidad de implementación, madurez del ecosistema y compatibilidad con herramientas de simulación y procesamiento de datos utilizados en el resto del sistema. Dentro del ecosistema Python, se analizaron los siguientes frameworks para el desarrollo de la API:

- **FastAPI** [67]: moderno, basado en ASGI, permite manejar solicitudes asincrónicas de forma nativa, soporta tipado estático (usando `pydantic`) y genera documentación automática a partir del código.
- **Flask** [68]: ligero y ampliamente adoptado, pero carece de soporte nativo para asincronía y requiere mayor configuración manual para validaciones y documentación.
- **Django REST Framework (DRF)** [69]: potente y extensible, ideal para aplicaciones con lógica de negocio compleja y administración incorporada, pero sobrecargado para APIs específicas o microservicios.

La elección de **FastAPI** responde a sus ventajas en términos de rendimiento y eficiencia en el manejo de solicitudes concurrentes, lo cual es crucial para una herramienta que puede realizar cálculos complejos durante la ejecución de simulaciones. Su integración con herramientas de tipado estático permite mayor robustez en tiempo de desarrollo y reducción de errores, además de facilitar la generación automática de documentación con OpenAPI. En la tabla 5.1 se detallan los endpoints de la API.

Método	Ruta	Recibe	Entrega	Función
GET	/lines	-	Lista de todas las líneas de producción	Obtener todas las líneas existentes.
GET	/simulations/{user}	ID de usuario	Lista de simulaciones (solo ID y fecha)	Obtener simulaciones de un usuario.
GET	/simulation/{id}	ID de simulación	Objeto Simulation completo	Obtener resultados de una simulación.
POST	/simulation	Parámetros de la simulación, ID de usuario, nombre para la simulación, tipo de uva (opcional)	Resultados de la simulación	Crear una simulación y retornar resultados.
DELETE	/simulation/{id}	ID de simulación	Mensaje de éxito o error	Borrar simulación de la base de datos.

Tabla 5.1: Descripción de los endpoints de la API

5.2.6. Autenticación

Para garantizar un acceso seguro y controlado a la herramienta de simulación, se implementó un sistema de autenticación basado en el uso de **JSON Web Tokens (JWT)** [70], integrado mediante el framework **NestJS** [71]. Esta combinación permite una autenticación robusta, sin estado y escalable, adecuada para gestionar múltiples usuarios simultáneos en un entorno productivo. Existen dos enfoques principales para implementar autenticación en aplicaciones web:

- **Autenticación basada en sesiones:** común en aplicaciones tradicionales, donde el servidor mantiene el estado de cada usuario mediante una sesión activa. Este enfoque requiere almacenamiento de sesiones en memoria o en una base de datos, lo que puede dificultar la escalabilidad en arquitecturas distribuidas.
- **Autenticación basada en tokens:** utiliza un token firmado (como JWT) que se almacena en el cliente y se reenvía con cada solicitud. Este enfoque elimina la necesidad de estado en el servidor y facilita la escalabilidad horizontal.

Dado el carácter modular y distribuido de la herramienta, se optó por la autenticación basada en tokens. Entre los mecanismos más comunes para autenticación sin estado se encuentran:

- **JWT (JSON Web Tokens)** [70]: permite almacenar de forma segura información codificada y firmada (como la identidad y permisos del usuario), verificable en cada solicitud sin necesidad de

consulta adicional.

- **OAuth2** [72]: estándar más complejo que permite delegar la autenticación entre servicios o aplicaciones de terceros. Es ideal para aplicaciones multi-cliente y de tipo SSO (Single Sign-On).
- **API Keys**: más simples de implementar, pero menos seguras al no ofrecer mecanismos integrados para expiración, firma o gestión de permisos.

JWT fue seleccionado por su equilibrio entre seguridad, flexibilidad y facilidad de integración con los controladores y middleware de NestJS.

Para implementar la autenticación, se utilizó **NestJS**, un framework de backend basado en TypeScript que permite estructurar aplicaciones modulares de forma clara. Entre las alternativas evaluadas se incluyeron:

- **Express** [73]: ampliamente utilizado y compatible con JWT, pero más básico en cuanto a estructura de proyecto y modularidad.
- **Koa** [74]: más minimalista y moderna, pero requiere mayor configuración manual.
- **Spring Boot (Java)** [75]: ofrece soluciones empresariales completas, aunque con mayor complejidad para proyectos ligeros y rápidos de desarrollar.

NestJS fue seleccionado por su integración nativa con bibliotecas de autenticación como `@nestjs/jwt`, su compatibilidad con middleware estándar de Express, y su arquitectura basada en decoradores, que permite una gestión clara de roles, guardias y control de acceso.

El proceso de autenticación implementado se resume en los siguientes pasos (véase Figura 5.1):

- El usuario envía sus credenciales (nombre de usuario y contraseña) al back-end de NestJS en el momento de inicio de sesión.
- NestJS verifica las credenciales del usuario con la base de datos. Si las credenciales son válidas, el servidor genera un token JWT firmado, que se envía al usuario como respuesta.
- El token JWT es almacenado en el cliente y se envía en cada solicitud que requiere autenticación, mediante el encabezado de autorización.
- NestJS valida el token en cada solicitud, permitiendo o denegando el acceso a recursos específicos en función de la validez del token y de los permisos del usuario.

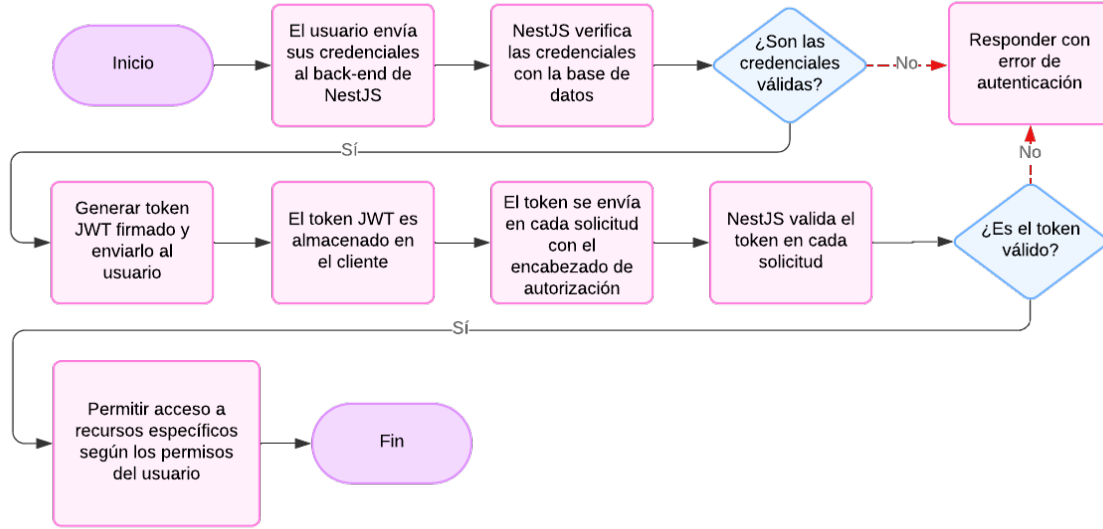


Figura 5.1: Flujo de autenticación con token JWT.

Este enfoque permite una autenticación sin estado (*stateless*), lo que facilita la escalabilidad del sistema y reduce la complejidad en la gestión de sesiones. Asimismo, mejora la seguridad al evitar exposiciones innecesarias de datos sensibles en el servidor y garantiza que solo usuarios autenticados puedan acceder a las funcionalidades críticas de la herramienta.

5.2.7. Motor de la Aplicación Web

El motor de la aplicación web, responsable de gestionar la interfaz y la experiencia del usuario, fue implementado utilizando el framework **Next.js** [76], una solución basada en React que combina renderizado del lado del cliente y del servidor (SSR, por sus siglas en inglés), generación estática de páginas (SSG) y capacidades híbridas. Esta elección respondió a necesidades específicas del sistema, tales como velocidad de carga, manejo eficiente del estado y facilidad de escalabilidad en el tiempo.

Durante el proceso de selección se evaluaron diversos frameworks modernos de frontend, considerando aspectos como rendimiento, modularidad, curva de aprendizaje, compatibilidad con servicios de backend y soporte comunitario. Entre las opciones consideradas se incluyeron:

- **Next.js** [76]: permite renderizado híbrido, generación de rutas automáticas, precarga inteligente de recursos, y optimización del rendimiento mediante carga bajo demanda.
- **Angular** [77]: framework completo mantenido por Google, con herramientas integradas para testing, validaciones, formularios reactivos y arquitectura en capas. Sin embargo, presenta una

curva de aprendizaje más pronunciada y mayor complejidad en la configuración inicial.

- **Vue.js** [78]: más simple de aprender, con una estructura flexible y una comunidad activa. No obstante, requiere mayor personalización y configuración para lograr características avanzadas de SSR o manejo de rutas.

Next.js fue elegido debido a su equilibrio entre rendimiento, flexibilidad y estructura clara basada en React. Su soporte para renderizado del lado del servidor permite mejorar el tiempo de carga percibido y optimizar el SEO de la herramienta, lo cual es relevante incluso en aplicaciones con componentes dinámicos.

La implementación del motor de la aplicación con Next.js permitió aprovechar varias ventajas clave:

- **Renderizado híbrido:** se pueden combinar páginas estáticas con páginas dinámicas.
- **Integración con backend:** la estructura de API Routes de Next.js facilita la conexión con servicios internos o externos cuando se requiere lógica del lado del servidor.
- **Modularidad y mantenibilidad:** su estructura basada en componentes y carpetas por convenciones promueve el desarrollo organizado y escalable.

En conjunto, la elección de Next.js como motor de la aplicación web se alinea con los objetivos de desarrollar una herramienta interactiva, adaptable y eficiente, mejorando tanto la experiencia del usuario final como la mantenibilidad del sistema.

5.3. Frontend

Las vistas del front-end fueron diseñadas para ofrecer una experiencia de usuario intuitiva y directa, permitiendo interactuar fácilmente con la herramienta de simulación. Cada vista ha sido creada con el objetivo de simplificar la navegación y proporcionar acceso rápido a las funciones principales, como la gestión de parámetros de simulación, la visualización de resultados y el control de acceso.

5.3.1. Inicio de sesión

La primera vista de la aplicación es la pantalla de inicio de sesión, donde los usuarios deben autenticarse para acceder a la herramienta de simulación. En esta vista, el usuario ingresa sus credenciales (correo electrónico y contraseña), que son validadas mediante el sistema de autenticación basado en JWT (véase Subsección 5.2.6). Esta capa de seguridad tiene como finalidad garantizar que solo los usuarios autorizados puedan acceder al sistema, protegiendo los datos y configuraciones de las simulaciones. En la Figura 5.2 se muestra esta vista en ejecución.

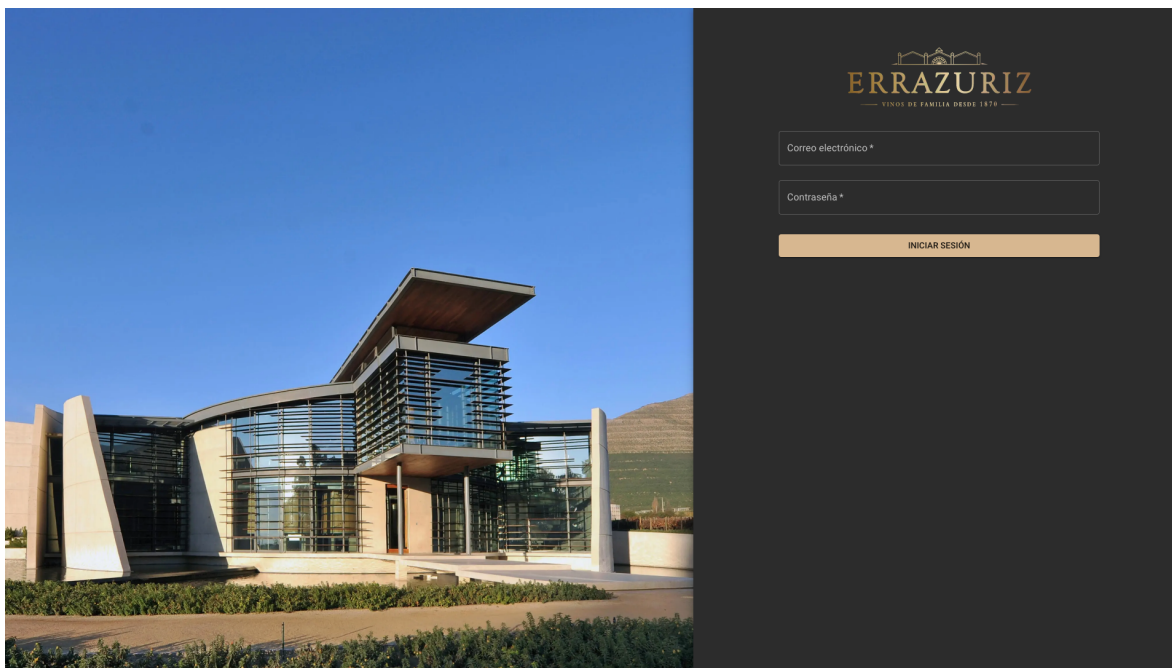


Figura 5.2: Vista de inicio de sesión

5.3.2. Dashboard

El Dashboard es la vista principal de la aplicación una vez que el usuario ha iniciado sesión. En esta vista, el usuario puede acceder a sus simulaciones recientes y gestionar o crear nuevas simulaciones. El dashboard ofrece un acceso rápido a las funcionalidades principales, permitiendo al usuario navegar de manera intuitiva entre las diferentes opciones de la herramienta. En la Figura 5.3 se muestra esta vista en ejecución.

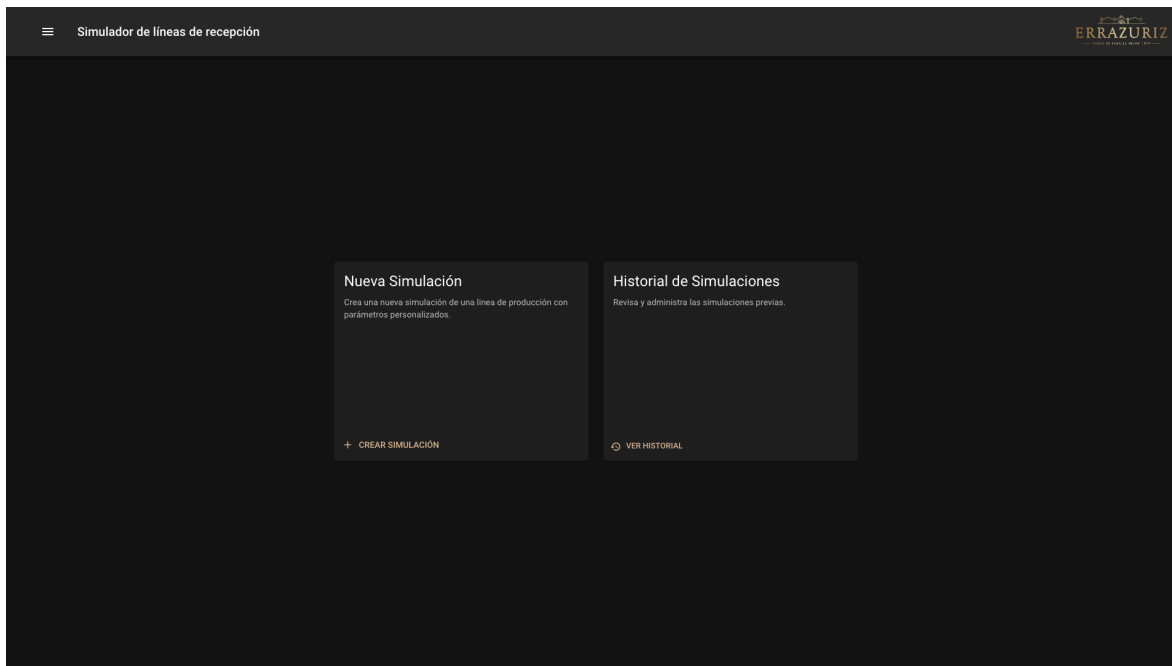
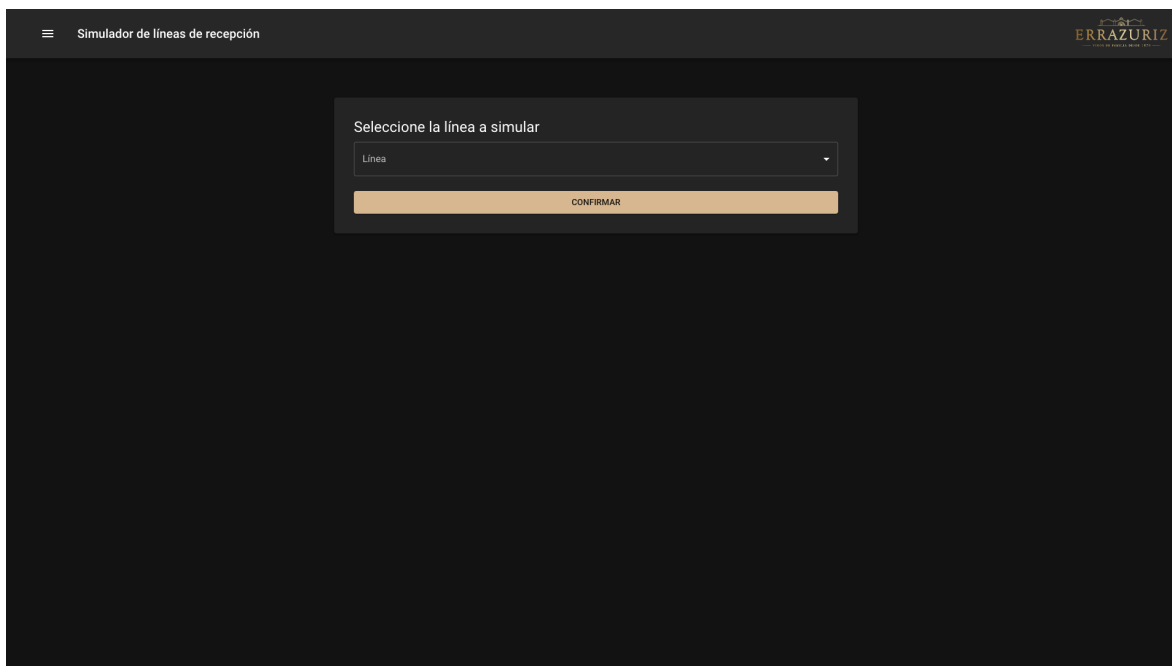


Figura 5.3: Vista del dashboard principal

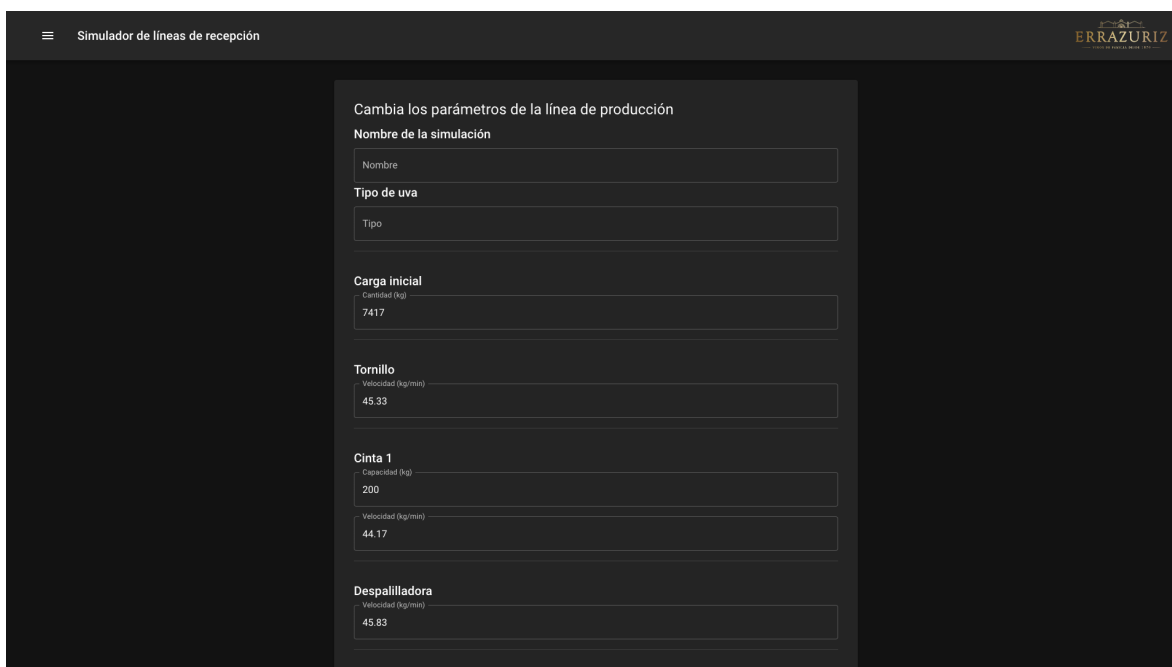
5.3.3. Formulario de nueva simulación

El formulario de nueva simulación permite al usuario configurar los parámetros de una simulación antes de su ejecución. En esta vista, el usuario puede ajustar valores específicos como el tipo de línea de producción, la velocidad de procesamiento de las máquinas y capacidad de estas según corresponda. Una vez configurados los parámetros, el usuario puede iniciar la simulación para obtener resultados. El formulario se compone de dos pasos, ilustrados en las Figuras 5.4a y 5.4b.



The screenshot shows a web application interface with a dark theme. At the top, there is a header bar with a hamburger menu icon on the left, the text 'Simulador de líneas de recepción' in the center, and the 'ERRAZURIZ' logo on the right. The main content area is a large dark rectangle. In the center of this area, there is a smaller, lighter gray rectangular box. Inside this box, the text 'Seleccione la línea a simular' is displayed above a dropdown menu. The dropdown menu has a label 'Línea' and a downward arrow. Below the dropdown menu is a wide, orange button with the text 'CONFIRMAR' in black capital letters.

(a) Selección de la línea.



The screenshot shows the same web application interface as in (a). The main content area now displays a configuration form. At the top of the form, the text 'Cambia los parámetros de la línea de producción' is shown. Below this, the form is organized into several sections, each with a title and input fields. The sections are: 'Nombre de la simulación' with a text input field containing 'Nombre'; 'Tipo de uva' with a text input field containing 'Tipo'; 'Carga inicial' with a text input field containing '7417' and a label 'Cantidad (kg)' above it; 'Tornillo' with a text input field containing '45.33' and a label 'Velocidad (kg/min)' above it; 'Cinta 1' with two text input fields, the first containing '200' (labeled 'Capacidad (kg)' above) and the second containing '44.17' (labeled 'Velocidad (kg/min)' above); and 'Despalilladora' with a text input field containing '45.83' and a label 'Velocidad (kg/min)' above it.

(b) Ingreso de variables de simulación.

Figura 5.4: Formulario de configuración de simulación.

5.3.4. Vista de simulación

La vista de simulación permite al usuario observar el flujo de uva en la línea de producción seleccionada, con los parámetros configurados previamente. En esta vista, se muestra una representación visual de cada máquina y su estado de funcionamiento, permitiendo al usuario monitorear la velocidad y acumulación de fruta para verificar posibles cuellos de botella o puntos críticos. En la Figura 5.5 se muestra la vista con los distintos bloques de la simulación, conectadas por tuberías con códigos de colores dependiendo de la velocidad de funcionamiento versus la velocidad nominal de salida de cada máquina (véase Tabla 5.2). También, dependiendo de la máquina, se muestra una barra de progreso que muestra el porcentaje de uso del acumulador, y si la simulación corresponde a un caso de falla, se colorea el bloque de la máquina responsable del cuello de botella.

Por ejemplo, en la Figura 5.5 se observa que, aunque la bomba ubicada al final de la línea opera a una velocidad de salida entre el 80 % y el 100 % de su velocidad nominal, la despalilladora presenta un rendimiento inferior al 50 % del valor especificado por el fabricante. En este escenario particular, el sistema identifica a la bomba como el componente responsable del cuello de botella en la producción.

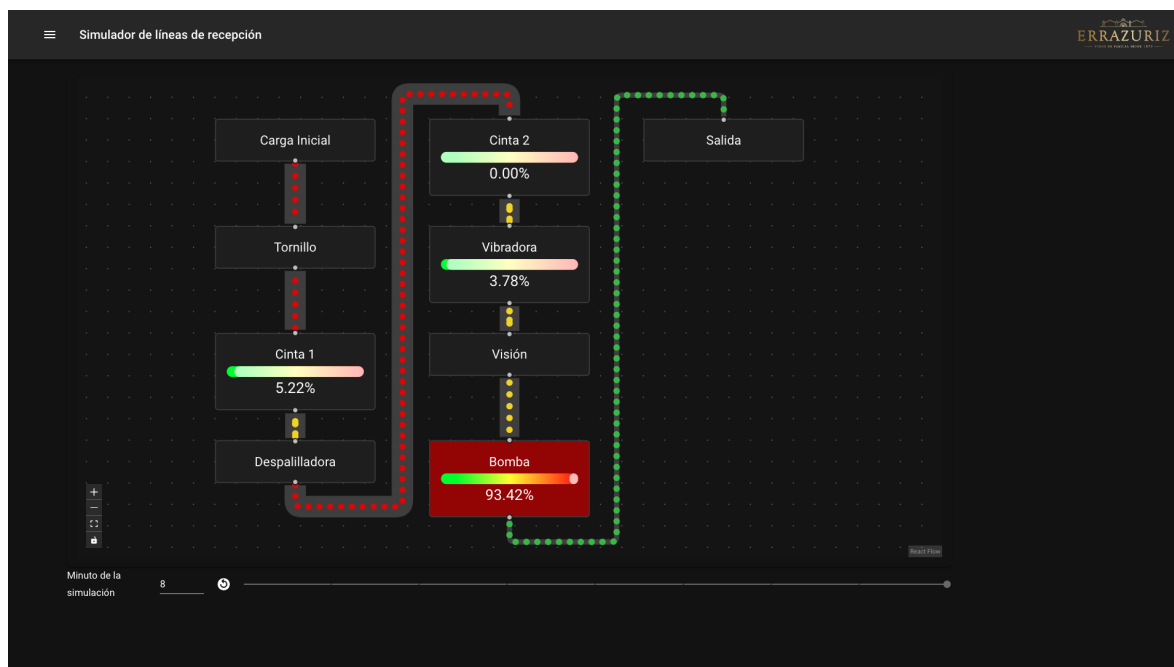


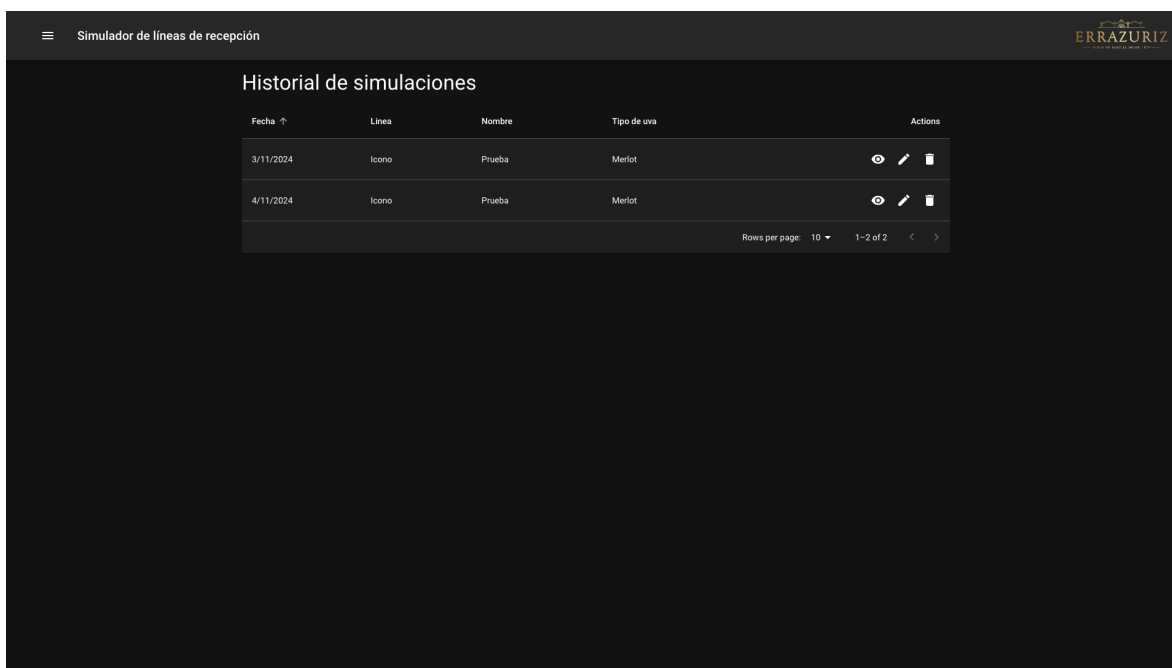
Figura 5.5: Vista de simulación en tiempo real

Color	Significado
Verde	La velocidad de funcionamiento de la máquina supera el 80 % de su velocidad nominal de salida. Representa un flujo óptimo y sin restricciones.
Amarillo	La velocidad de funcionamiento se encuentra entre el 50 % y el 80 % de la velocidad nominal. Indica una posible ralentización o acumulación moderada.
Rojo	La velocidad de funcionamiento es inferior al 50 % de la velocidad nominal. Señala un cuello de botella o punto crítico en la línea de producción.

Tabla 5.2: Códigos de colores en la vista de simulación y su interpretación

5.3.5. Historial de simulaciones

La vista de historial de simulaciones, véase Figura 5.6, presenta una tabla que permite al usuario acceder a una lista de todas las simulaciones realizadas previamente, junto con detalles claves como fecha, nombre de la simulación, línea y tipo de uva. Cada simulación en el historial puede ser visualizada, editada y eliminada según estime el usuario.



Historial de simulaciones				
Fecha ↑	Línea	Nombre	Tipo de uva	Actions
3/11/2024	Icono	Prueba	Merlot	👁️ ✎️ 🗑️
4/11/2024	Icono	Prueba	Merlot	👁️ ✎️ 🗑️

Rows per page: 10 1-2 of 2 < >

Figura 5.6: Tabla de historial de simulaciones.

5.4. Despliegue de la solución

Se detallan las actividades de verificación y validación realizadas para garantizar la calidad técnica y funcional del sistema, así como el plan de pruebas diseñado para asegurar su estabilidad antes de su implementación en producción. Adicionalmente, se expone la estrategia adoptada para gestionar los riesgos identificados durante el desarrollo, abordando tanto aspectos técnicos como de adopción por parte de los usuarios finales.

5.4.1. Verificación y Validación del Sistema

El proceso de verificación y validación constituyó una etapa fundamental para asegurar la calidad y funcionalidad del sistema desarrollado. Su propósito fue garantizar que los requisitos funcionales y no funcionales definidos durante las fases iniciales fueran cumplidos de manera íntegra.

La verificación se orientó a comprobar que el sistema fue construido conforme a las especificaciones técnicas, validando el correcto funcionamiento de cada componente, tanto de manera aislada como en conjunto. En contraste, la validación se enfocó en asegurar que la herramienta satisficiera las necesidades y expectativas de los usuarios finales. Ambos procesos se llevaron a cabo de manera iterativa a lo largo del ciclo de desarrollo, permitiendo identificar y corregir posibles desviaciones de manera oportuna.

Verificación

La verificación del sistema fue concebida como un proceso integral para asegurar que cada componente funcional opere de acuerdo con las especificaciones definidas. Esta etapa incluyó tanto pruebas automatizadas como herramientas de control de calidad en el flujo de desarrollo, abarcando tanto el backend como el frontend de la herramienta.

Para garantizar que la API respondiera correctamente a las solicitudes, se implementaron pruebas unitarias e integradas utilizando la herramienta **PyTest** [79]. Esta librería, ampliamente adoptada en el ecosistema Python, permite escribir pruebas de forma concisa y escalable, con soporte para fixtures, parametrización y detección automática de casos de prueba. Entre las alternativas consideradas se encuentran:

- **unittest** [80]: librería estándar de Python, robusta pero más verbosa y con menor flexibilidad en comparación con PyTest.

- **nose2** [81]: sucesor de la librería **nose**, ofrece características similares a PyTest pero con menor nivel de adopción y soporte.

La elección de PyTest se debió a su compatibilidad con herramientas de integración continua y su capacidad para estructurar pruebas complejas con menor sobrecarga sintáctica.

Para el código cliente, desarrollado con **Next.js**, se implementaron mecanismos de validación automática mediante herramientas de control de calidad y convenciones de desarrollo. En particular, se utilizó **Husky** [82] para definir *git hooks* que ejecutan scripts de validación antes de cada commit, y **Commitizen** [83] para estandarizar mensajes de confirmación de cambios según el formato **Conventional Commits**.

Estas herramientas contribuyen a mantener un historial de cambios claro y consistente, además de facilitar futuras automatizaciones como la generación de changelogs o la publicación automatizada de versiones. Alternativas evaluadas incluyeron:

- **Lint-staged** [84]: herramienta complementaria para validar archivos modificados, aunque menos integral en cuanto al control de versiones semántico.
- **Standard Version** [85]: automatiza versionado semántico, pero sin integración directa con formatos de commit previos al **push**.

La base de código frontend fue desarrollada utilizando **TypeScript**, un superset de JavaScript que incorpora tipado estático. Esta característica contribuyó significativamente a la detección temprana de errores relacionados con el uso incorrecto de variables o funciones, aumentando la robustez del sistema. Asimismo, el cumplimiento del estándar **ECMAScript 2024** [86] permitió asegurar la compatibilidad con modernas prácticas de desarrollo y navegadores actuales.

En conjunto, la verificación del sistema se implementó mediante una combinación de pruebas automatizadas y control de calidad continuo, con herramientas adaptadas a las características propias de cada módulo, asegurando así la fiabilidad, mantenibilidad y escalabilidad del sistema.

Validación

La validación se realizó en estrecha colaboración con los usuarios finales. A través de sesiones periódicas, se evaluó el funcionamiento y usabilidad de la herramienta, garantizando que cumpliera con los objetivos operacionales planteados. Este proceso incluyó la comparación de los resultados generados por la simulación con datos reales y la evaluación del desempeño en escenarios de uso cotidiano. Asimismo, se ajustaron funcionalidades de acuerdo a la retroalimentación proporcionada por

los usuarios, adoptando un enfoque iterativo que permitió adaptar la herramienta de manera progresiva a los requerimientos detectados.

5.4.2. Plan de Pruebas

El plan de pruebas se definió con el propósito de validar el correcto funcionamiento del sistema antes de su despliegue en producción. Para ello, se establecieron objetivos específicos orientados a cubrir tanto la integridad de las interfaces como el comportamiento de los servicios.

Objetivos

1. Verificar la correcta interacción entre los diferentes componentes del back-end.
2. Validar la funcionalidad de los endpoints de la API.
3. Asegurar un manejo adecuado de errores y excepciones.
4. Comprobar la integridad y consistencia de los datos procesados.

Estrategia

La estrategia de pruebas contempló los siguientes enfoques:

- **Pruebas unitarias:** Orientadas a validar de manera aislada cada unidad funcional del sistema.
- **Pruebas de integración:** Diseñadas para evaluar la interacción entre los módulos, incluyendo la comunicación entre el front-end, el back-end y la base de datos.

Herramientas

Para la ejecución de las pruebas se emplearon:

- **PyTest**, como framework principal para las pruebas unitarias e integradas.
- **Husky**, para la ejecución automática de validaciones previas a los commits.
- **Commitizen**, para estandarizar los mensajes de commit y garantizar buenas prácticas en el versionado.

Criterios de Aceptación

Un caso de prueba fue considerado exitoso si cumplía con los siguientes criterios técnicos y de calidad:

- Ejecución satisfactoria de todos los casos definidos sin errores ni fallos críticos.
- Respuestas de la API concordantes con los requisitos funcionales, incluyendo códigos de estado HTTP apropiados y tiempos de respuesta menores a 300 ms en el 95 % de las solicitudes, conforme a estándares de rendimiento en aplicaciones web [87].
- Cobertura de código superior al **90 %** en los módulos de backend validados con **PyTest**, incluyendo ramas de ejecución, condiciones límite y errores manejados [79, 88].
- Validación de integridad de datos: no se detectaron inconsistencias en registros de base de datos luego de la ejecución de pruebas integradas.
- Correcta finalización de flujos definidos en los **git hooks** de **Husky**, incluyendo revisión de estilo (**lint**), tipado estático con **TypeScript** y validación de convenciones de commit con **Commitizen**.
- Tasa de éxito en ejecución de pruebas automatizadas igual o superior al **98 %** en ambientes de desarrollo y staging.

Estos criterios permitieron establecer un umbral técnico verificable para asegurar la estabilidad funcional del sistema antes de su despliegue. Además, la definición de métricas cuantificables facilitó la integración del proceso de pruebas en los ciclos de integración y entrega continua.

5.4.3. Gestión de Riesgos

Durante el desarrollo y despliegue del sistema, se identificaron posibles riesgos que podrían afectar su correcto funcionamiento. Para cada uno de ellos se definieron acciones de mitigación, las cuales se detallan a continuación.

1. Datos inexactos o incompletos:

- *Evaluación:* Existía la posibilidad de que los datos utilizados no representaran fielmente la realidad.
- *Mitigación:* Se habilitó la parametrización de simulaciones para evaluar múltiples escenarios, restringiendo además la simulación de situaciones inviables.

2. Obsolescencia tecnológica:

- *Evaluación:* Las tecnologías empleadas podrían quedar obsoletas en el mediano plazo.
- *Mitigación:* La adopción de una arquitectura modular basada en contenedores facilita la actualización independiente de los componentes.

3. Adopción de la aplicación:

- *Evaluación:* Existía riesgo de baja adopción por parte de los usuarios finales.
- *Mitigación:* Se llevó a cabo un desarrollo iterativo con la participación activa del cliente, complementado con instancias de capacitación.

Este capítulo se detalla la construcción de la herramienta de simulación, abordando cada una de las decisiones técnicas que permitieron dar forma a una solución funcional y segura. Desde la elección del entorno de desarrollo hasta la configuración del sistema en la nube, se explican los distintos componentes que hacen posible su operación: la base de datos, la API, el motor de la aplicación web y el sistema de autenticación. También se describen las estrategias utilizadas para verificar su correcto funcionamiento, la validación junto a usuarios reales, y el aseguramiento de su calidad antes del despliegue. A través de este proceso, no solo se construye un sistema robusto, sino también uno que puede adaptarse con facilidad a las necesidades reales del proceso de recepción de uva.

Capítulo 6

Conclusiones y Perspectivas

En este trabajo se aborda de manera integral el diseño, implementación y validación de una herramienta digital orientada a la simulación de procesos en el contexto vitivinícola. A partir de una revisión de los requerimientos operacionales y técnicos del sistema, se define una arquitectura modular que facilita el desarrollo progresivo de sus componentes, garantizando la cohesión funcional entre las distintas partes del sistema. La implementación se lleva a cabo mediante tecnologías actuales y ampliamente utilizadas en la industria, lo que favorece tanto la mantenibilidad del código como la escalabilidad de la solución. La validación del sistema, a través de su aplicación en un caso de estudio representativo, permite verificar su funcionamiento en condiciones reales de uso y evaluar su capacidad para apoyar la toma de decisiones operativas. En conjunto, los resultados obtenidos evidencian la factibilidad técnica de la propuesta y su pertinencia frente a las necesidades prácticas del sector, constituyendo un aporte concreto al desarrollo de soluciones digitales aplicadas a contextos productivos específicos.

6.1. Conclusiones Generales

A lo largo del desarrollo de este proyecto, se cumplieron satisfactoriamente los objetivos propuestos, los cuales orientaron de forma estructurada todas las etapas del proyecto:

1. Se logró realizar un levantamiento completo, detallado y preciso de las siete líneas de producción presentes en Viña Errázuriz. Este trabajo incluyó la documentación técnica de cada máquina, su ubicación dentro del flujo productivo y sus principales características operacionales, como velocidades de salida, capacidades de acumulación y conexiones entre equipos. Esta información

fue sistematizada y estandarizada, sirviendo como base para alimentar el modelo computacional de simulación de manera coherente y confiable.

2. Se desarrolló un modelo matemático que representa el proceso como un flujo secuencial de operaciones industriales, considerando restricciones operativas como la capacidad de acumuladores, las velocidades nominales de salida y la posibilidad de rutas alternativas. El modelo permite detectar cuellos de botella de manera automatizada, simular condiciones diversas y evaluar con mayor objetividad el comportamiento de la línea frente a diferentes configuraciones operativas.
3. Se diseñó y construyó una aplicación web completamente funcional, desarrollada utilizando tecnologías como Next.js y FastAPI, que ofrece una interfaz gráfica amigable para que los usuarios configuren, ejecuten y analicen simulaciones personalizadas. La validación del sistema se realizó a través de pruebas técnicas exhaustivas y sesiones de trabajo con usuarios finales, donde se contrastaron los resultados simulados con datos históricos y se ajustaron los parámetros para mejorar la precisión del modelo y la experiencia de uso.
4. La herramienta digital desarrollada constituye un sistema funcional robusto que combina tecnologías web modernas con principios de modelamiento de procesos productivos, permitiendo a los usuarios simular, analizar y visualizar el comportamiento dinámico de una línea de producción de uva. La arquitectura modular, basada en componentes reutilizables y servicios RESTful, garantiza escalabilidad y facilita futuras adaptaciones o integraciones.
5. Desde la experiencia del usuario, la herramienta aporta claridad visual y operativa: permite identificar rápidamente zonas críticas o cuellos de botella gracias al uso de codificación cromática, gráficos dinámicos y representaciones esquemáticas de la línea. Esta visualización se complementa con una lógica de simulación eficiente y validaciones internas que aseguran la coherencia de los resultados.

No obstante, se identifican algunas áreas susceptibles de mejora. Actualmente, el modelo no considera elementos económicos como los costos de operación, consumo energético o mantenimiento de las máquinas, lo que limita la capacidad de realizar análisis de costo-beneficio. Además, la herramienta fue diseñada para ejecutarse en entornos locales, de red privada o acceso público bajo autorización utilizando credenciales, por lo que su uso en contextos de producción industrial requeriría mejoras en la seguridad (como el despliegue en servidores empresariales), monitoreo, respaldo de datos y rendimiento bajo carga.

6.2. Caso de estudio: Viña Errázuriz

La aplicación fue aplicada y validada utilizando como base la línea “Ícono” de Viña Errázuriz, la más tecnificada de la planta. Esta línea incorpora equipamiento moderno de origen europeo, altamente especializado, y opera bajo estándares de calidad exigentes para la elaboración de vinos de alta gama. Este contexto brindó un escenario ideal para probar la flexibilidad y adaptabilidad del modelo desarrollado.

La herramienta demostró su utilidad práctica al permitir simular distintos escenarios operacionales, evaluar el impacto de cambios en la velocidad de máquinas y detectar acumulaciones o sobrecargas en puntos específicos de la línea. Los operarios y supervisores de la viña participaron activamente en el proceso de validación, lo que permitió ajustar tanto la interfaz como los parámetros técnicos para alinearlos con la realidad operativa.

Sin embargo, se identificaron desafíos importantes relacionados con la disponibilidad, precisión y consistencia de los datos históricos de funcionamiento. En varios casos fue necesario trabajar con estimaciones, lo que introduce cierto margen de error en las simulaciones. Esta limitación evidencia la necesidad de avanzar hacia una integración directa con sistemas de adquisición de datos en planta, como SCADA o sensores IoT, lo cual representa una evolución lógica para futuras versiones.

6.3. Perspectivas y trabajo futuro

A partir del trabajo realizado, se identifican múltiples oportunidades para mejorar y expandir la herramienta desarrollada. Algunas de las principales líneas de trabajo futuro incluyen:

- **Integración con sistemas industriales:** Incorporar datos en tiempo real provenientes de sensores físicos, sistemas SCADA o plataformas de IoT permitiría representar condiciones actuales de operación, generar simulaciones más precisas y dinámicas, e integrar sistemas de alertas en caso de fallas críticas.
- **Optimización de parámetros:** Incluir algoritmos de optimización (basados en heurísticas, algoritmos genéticos o inteligencia artificial) que sugieran configuraciones óptimas de máquinas para minimizar el tiempo de proceso, evitar cuellos de botella o maximizar la eficiencia energética.
- **Análisis económico integrado:** Añadir un módulo que permita calcular indicadores económicos como costos operativos, retorno sobre inversión o ahorro potencial por sustitución de equipos, podría mejorar la herramienta como sistema de apoyo a la toma de decisiones estratégicas.

- **Aplicación a otras líneas e industrias:** El diseño modular permite replicar el modelo en otras líneas dentro de Viña Errázuriz o incluso adaptarlo a otros sectores industriales con procesos secuenciales, como agroindustrias, embotelladoras o plantas de alimentos.
- **Versión multiusuario y colaborativa:** Desarrollar una plataforma que permita a distintos usuarios compartir, comentar y guardar simulaciones facilitaría el trabajo en equipo, mejorando la trazabilidad de decisiones y la colaboración entre áreas.

En conclusión, el desarrollo de esta herramienta representa una contribución significativa al análisis y visualización del proceso de recepción de uva en entornos productivos reales. El sistema desarrollado cumple con el alcance inicial planteado, logrando integrar en una solución funcional los requerimientos definidos al comienzo del proyecto, tanto desde el punto de vista técnico como operativo. Si bien se reconoce que existen oportunidades de mejora, el proyecto constituye una base sólida y escalable sobre la cual se pueden construir nuevas funcionalidades, adaptaciones a diferentes realidades operativas y extensiones hacia contextos industriales más complejos.

Bibliografía

- [1] Decanter. *Decanter World Wine Awards 2024*. URL: https://awards.decanter.com/DWWA/2024/search/wines?_gl=1%2A1m9985p%2A_gcl_au%2AMTk2MTM5NDA1OS4xNzI4MjM5NTYx&competitionType=DWWA&country=Chile.
- [2] Davide Carneiro, Jorge Pereira y Eliana Costa e Silva. “Optimization of the grapes reception process”. En: *Neural Computing and Applications* 33.14 (jul. de 2021), págs. 8687-8707. ISSN: 1433-3058. DOI: 10.1007/s00521-020-05620-0. URL: <https://doi.org/10.1007/s00521-020-05620-0>.
- [3] Hugh Johnson. *Hugh Johnson’s story of wine*. Mitchell Beazley, 2004.
- [4] Sergi Maicas y José Juan Mateo. “Sustainability of Wine Production”. En: *Sustainability* 12.2 (ene. de 2020), pág. 559. ISSN: 2071-1050. DOI: 10.3390/su12020559. URL: <http://dx.doi.org/10.3390/su12020559>.
- [5] R. Bierbooms. “Performance analysis of production lines : discrete and continuous flow models”. English. Phd Thesis 2 (Research NOT TU/e / Graduation TU/e). Mathematics y Computer Science, 2012. ISBN: 978-90-5335-543-5. DOI: 10.6100/IR734088.
- [6] Michael Pinedo. *Scheduling: Theory, Algorithms, and Systems*. 4.^a ed. New York: Springer, 2012. ISBN: 978-1-4614-2360-1.
- [7] J. Blazewicz, K. H. Ecker, E. Pesch, G. Schmidt y J. Weglarz. *Handbook on Scheduling: From Theory to Applications*. Berlin, Heidelberg: Springer, 2007. ISBN: 978-3-540-28046-9.
- [8] A. S. Manne. “On the job-shop scheduling problem”. En: *Operations Research* 8.2 (1960), págs. 219-223. DOI: 10.1287/opre.8.2.219.
- [9] Michael R. Garey y David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-completeness*. San Francisco: W. H. Freeman, 1979. ISBN: 978-0-7167-1045-5.

- [10] Jesus Para, Javier Del Ser y Antonio J. Nebro. “Energy-Aware Multi-Objective Job Shop Scheduling Optimization with Metaheuristics in Manufacturing Industries: A Critical Survey, Results, and Perspectives”. En: *Applied Sciences* 12.3 (2022). ISSN: 2076-3417. DOI: 10.3390/app12031491. URL: <https://www.mdpi.com/2076-3417/12/3/1491>.
- [11] Mitsuo Gen, Runwei Cheng y Yasasuke Tsujimura. “A tutorial survey of job-shop scheduling problems using genetic algorithms—Part I: representation”. En: *Computers & Industrial Engineering* 30.4 (1996), págs. 983-997. DOI: 10.1016/0360-8352(95)00218-3.
- [12] Fred Glover. “Tabu search—Part I”. En: *ORSA Journal on Computing* 1.3 (1989), págs. 190-206. DOI: 10.1287/ijoc.1.3.190.
- [13] S. Kirkpatrick, C. D. Gelatt Jr. y M. P. Vecchi. “Optimization by simulated annealing”. En: *Science* 220.4598 (1983), págs. 671-680. DOI: 10.1126/science.220.4598.671.
- [14] H. R. Lourenço, O. Martin y T. Stützle. “Iterated local search: Framework and applications”. En: *Handbook of Metaheuristics*. Vol. 146. International Series in Operations Research & Management Science. Boston, MA: Springer, 2010, págs. 363-397. DOI: 10.1007/978-1-4419-1665-5_12.
- [15] Andrés Auger, Juan Carlos Ferrer, Sergio Maturana y Jorge Vera. *Simulation of the Grape Reception at a Winery*. 2003. URL: https://www.researchgate.net/publication/228715607_Simulation_of_the_Grape_Reception_at_a_Winery%7D.
- [16] Christian Becker y Armin Scholl. “A survey on problems and methods in generalized assembly line balancing”. En: *European Journal of Operational Research* 168.3 (2006), págs. 694-715.
- [17] Nils Boysen, Malte Flidner y Armin Scholl. “A classification of assembly line balancing problems”. En: *European Journal of Operational Research* 183.2 (2007), págs. 674-693.
- [18] Matthew E. Salveson. “The assembly line balancing problem”. En: *Journal of Industrial Engineering* 6.3 (1955), págs. 18-25.
- [19] Subir Ghosh y Robert J. Gagnon. “A comprehensive literature review and analysis of the design, balancing and scheduling of assembly systems”. En: *International Journal of Production Research* 27.4 (1989), págs. 637-670.
- [20] Imren Baybars. “A survey of exact algorithms for the simple assembly line balancing problem”. En: *Management Science* 32.8 (1986), págs. 909-932.
- [21] Armin Scholl. *Balancing and sequencing of assembly lines*. Physica-Verlag HD, 1999.
- [22] Armin Scholl y Christian Becker. “State-of-the-art exact and heuristic solution procedures for simple assembly line balancing”. En: *Production and Operations Management*. Ed. por Gregory C. Smith y Michael A. Buzacott. Springer, 2006, págs. 230-244.

- [23] E C Sewell y S H Jacobson. “A branch, bound, and remember algorithm for the simple assembly line balancing problem”. en. En: *INFORMS J. Comput.* 24.3 (ago. de 2012), págs. 433-442.
- [24] Mitsuo Gen, Runwei Cheng y Ling Lin. “Network models and optimization: Multiobjective genetic algorithm approach”. En: *Journal of Intelligent Manufacturing* 19.3 (2008), págs. 323-332.
- [25] Salih Akpinar y Gülçin M. Bayhan. “A hybrid genetic algorithm for mixed model assembly line balancing problem with parallel workstations and zoning constraints”. En: *Engineering Applications of Artificial Intelligence* 24.3 (2011), págs. 449-457.
- [26] Ronald L. Rardin y Reha Uzsoy. “Experimental evaluation of heuristic optimization algorithms: A tutorial”. En: *Journal of Heuristics* 7.3 (2001), págs. 261-304.
- [27] Jan vom Brocke, Lars Mathiassen y Michael Rosemann. “Business Process Management”. En: *Business & Information Systems Engineering* 6.4 (ago. de 2014), págs. 189-189. ISSN: 1867-0202. DOI: 10.1007/s12599-014-0330-8. URL: <https://doi.org/10.1007/s12599-014-0330-8>.
- [28] Hajo A. Reijers. “Business Process Management: The evolution of a discipline”. En: *Computers in Industry* 126 (2021), pág. 103404. ISSN: 0166-3615. DOI: <https://doi.org/10.1016/j.compind.2021.103404>. URL: <https://www.sciencedirect.com/science/article/pii/S0166361521000117>.
- [29] Poliana Gomes, Luiz Verçosa, Fagner Melo, Vinícius Silva, Carmelo Bastos Filho y Byron Bezerra. “Artificial Intelligence-Based Methods for Business Processes: A Systematic Literature Review”. En: *Applied Sciences* 12.5 (2022). ISSN: 2076-3417. DOI: 10.3390/app12052314. URL: <https://www.mdpi.com/2076-3417/12/5/2314>.
- [30] Bui Quang Truong, Anh Nguyen-Duc y Nguyen Thi Cam Van. “A Quantitative Review of the Research on Business Process Management in Digital Transformation: A Bibliometric Approach”. En: *Software* 2.3 (2023), págs. 377-399. ISSN: 2674-113X. DOI: 10.3390/software2030018. URL: <https://www.mdpi.com/2674-113X/2/3/18>.
- [31] Nurgul Nalgozhina, Abdul Razaque, Uskenbayeva Raissa y Joon Yoo. “Developing Robotic Process Automation to Efficiently Integrate Long-Term Business Process Management”. En: *Technologies* 11.6 (2023). ISSN: 2227-7080. DOI: 10.3390/technologies11060164. URL: <https://www.mdpi.com/2227-7080/11/6/164>.
- [32] Lionel Valenzuela Oyaneder y Sergio Maturana Valderrama and. “A new balanced scorecard approximation to enhance performance management systems of Chilean wineries”. En: *Journal of Wine Research* 27.1 (2016), págs. 1-18. DOI: 10.1080/09571264.2016.1141761. eprint: <https://doi.org/10.1080/09571264.2016.1141761>. URL: <https://doi.org/10.1080/09571264.2016.1141761>.

- [33] Sotiris P. Gayialis, Evripidis P. Kechagias, Georgios A. Papadopoulos y Nikolaos A. Panayiotou. “A Business Process Reference Model for the Development of a Wine Traceability System”. En: *Sustainability* 14.18 (2022). ISSN: 2071-1050. DOI: 10.3390/su141811687. URL: <https://www.mdpi.com/2071-1050/14/18/11687>.
- [34] Enrique Cañete-Carmona, Jin Chen, Carmen Martín y Bernardino Rubio. “Smart Winery: A real-time monitoring system for structural health and ullage in fino style wine casks”. En: *Sensors* 18.3 (2018), pág. 803. DOI: 10.3390/s18030803.
- [35] Andrés Auger, José-Carlos Ferrer, Sergio Maturana y Jorge Vera. *Simulation of the Grape Reception at a Winery*. Inf. téc. Informe técnico. Santiago, Chile: Escuela de Ingeniería, Pontificia Universidad Católica de Chile, 2003.
- [36] Figma. *Figma*. Accessed: 2025-05-01. URL: <https://www.figma.com/>.
- [37] PELLENC. *Fabricante de máquinas, equipos y herramientas con batería profesional para vitivinicultura, arboricultura y mantenimiento de espacios verdes — Pellenc*. Accedido en mayo de 2025. 2025. URL: <https://www.pellenc.com/es-cl/>.
- [38] Microsoft. *Visual Studio Code*. Accedido en mayo de 2025. 2024. URL: <https://code.visualstudio.com/>.
- [39] The Git Community. *Git - Distributed Version Control System*. <https://github.com/git/git>. Accedido en mayo de 2025. 2024.
- [40] Sublime HQ. *Sublime Text*. Accedido en mayo de 2025. 2024. URL: <https://www.sublimetext.com/>.
- [41] GitHub Inc. *Atom Editor*. Editor discontinuado en 2022. Accedido en mayo de 2025. 2022. URL: <https://github.blog/2022-06-08-sunsetting-atom/>.
- [42] GitHub. *Sunsetting Atom*. Accedido en mayo de 2025. 2022. URL: <https://github.blog/2022-06-08-sunsetting-atom/>.
- [43] JetBrains. *IntelliJ IDEA*. Accedido en mayo de 2025. 2024. URL: <https://www.jetbrains.com/idea/>.
- [44] DigitalOcean. *DigitalOcean*. Accessed: 2025-05-04. URL: <https://www.digitalocean.com>.
- [45] Amazon Web Services. *Amazon EC2 - Secure and resizable compute capacity*. Accedido en mayo de 2025. 2024. URL: <https://aws.amazon.com/ec2/>.
- [46] Google Cloud. *Google Compute Engine*. Accedido en mayo de 2025. 2024. URL: <https://cloud.google.com/compute>.

- [47] Salesforce. *Heroku - Cloud Application Platform*. Accedido en mayo de 2025. 2024. URL: <https://www.heroku.com/>.
- [48] Vercel Inc. *Vercel - Develop. Preview. Ship*. Accedido en mayo de 2025. 2024. URL: <https://vercel.com/>.
- [49] Inc. NGINX. *NGINX - High Performance Load Balancer, Web Server, & Reverse Proxy*. Accedido en mayo de 2025. 2024. URL: <https://www.nginx.com/>.
- [50] Apache Software Foundation. *Apache HTTP Server Project*. Accedido en mayo de 2025. 2024. URL: <https://httpd.apache.org/>.
- [51] HAProxy Technologies. *HAProxy - The Reliable, High Performance TCP/HTTP Load Balancer*. Accedido en mayo de 2025. 2024. URL: <https://www.haproxy.org/>.
- [52] Let's Encrypt. *Let's Encrypt - Free SSL/TLS Certificates*. Accedido en mayo de 2025. 2024. URL: <https://letsencrypt.org/>.
- [53] Cloudflare Inc. *SSL for Free by Cloudflare*. Accedido en mayo de 2025. 2024. URL: <https://www.cloudflare.com/ssl/>.
- [54] Comodo CA. *Comodo SSL Certificates*. Accedido en mayo de 2025. 2024. URL: <https://www.comodo.com/ssl/>.
- [55] Tech Domains. *Tech Domains – Domain Registration Services*. Accedido en mayo de 2025. 2024. URL: <https://www.techdomains.com/>.
- [56] GoDaddy Inc. *Domain Names, Websites, Hosting & Online Marketing Tools*. Accedido en mayo de 2025. 2024. URL: <https://www.godaddy.com/>.
- [57] Namecheap Inc. *Namecheap – Domain Name Registrar and Web Hosting*. Accedido en mayo de 2025. 2024. URL: <https://www.namecheap.com/>.
- [58] MongoDB Inc. *MongoDB*. Accedido en mayo de 2025. 2024. URL: <https://www.mongodb.com/>.
- [59] PostgreSQL Global Development Group. *PostgreSQL*. Accedido en mayo de 2025. 2024. URL: <https://www.postgresql.org/>.
- [60] Oracle Corporation. *MySQL*. Accedido en mayo de 2025. 2024. URL: <https://www.mysql.com/>.
- [61] Apache Software Foundation. *Apache Cassandra*. Accedido en mayo de 2025. 2024. URL: <https://cassandra.apache.org/>.
- [62] Google LLC. *Firebase Realtime Database*. Accedido en mayo de 2025. 2024. URL: <https://firebase.google.com/products/realtime-database>.
- [63] Python Software Foundation. *Python Programming Language*. Accedido en mayo de 2025. 2024. URL: <https://www.python.org/>.

- [64] OpenJS Foundation. *Node.js*. Accedido en mayo de 2025. 2024. URL: <https://nodejs.org/>.
- [65] Oracle Corporation. *Java Platform*. Accedido en mayo de 2025. 2024. URL: <https://www.oracle.com/java/>.
- [66] Google. *The Go Programming Language*. Accedido en mayo de 2025. 2024. URL: <https://go.dev/>.
- [67] Sebastián Ramírez. *FastAPI*. <https://fastapi.tiangolo.com>. URL: <https://github.com/fastapi/fastapi>.
- [68] Pallets Projects. *Flask Documentation*. Accedido en mayo de 2025. 2024. URL: <https://flask.palletsprojects.com/>.
- [69] Encode OSS Ltd. *Django REST framework*. Accedido en mayo de 2025. 2024. URL: <https://www.django-rest-framework.org/>.
- [70] Auth0. *JWT.IO - JSON Web Tokens*. Accedido en mayo de 2025. 2024. URL: <https://jwt.io/>.
- [71] NestJS. *NestJS - A Progressive Node.js Framework*. Accedido en mayo de 2025. 2024. URL: <https://nestjs.com/>.
- [72] OAuth Working Group. *The OAuth 2.0 Authorization Framework*. Accedido en mayo de 2025. 2024. URL: <https://oauth.net/2/>.
- [73] Express.js. *Express - Node.js web application framework*. Accedido en mayo de 2025. 2024. URL: <https://expressjs.com/>.
- [74] KOA. *Koa - Next generation web framework for Node.js*. Accedido en mayo de 2025. 2024. URL: <https://koajs.com/>.
- [75] Spring Team. *Spring Boot*. Accedido en mayo de 2025. 2024. URL: <https://spring.io/projects/spring-boot>.
- [76] Department of Computer, Faculty of Education, University of Zawia, Zawia, Libya, Hamed Hussen Ben kora, Mohamed Sasi Manita y College of Computer Technology, Zawia, Libya. "Modern front-end web architecture using React.Js and Next.Js". En: *Univ Zawia J Eng Sci Technol* 2.1 (ago. de 2024), págs. 1-13.
- [77] Google. *Angular*. Accedido en mayo de 2025. 2024. URL: <https://angular.io/>.
- [78] Evan You. *Vue.js - The Progressive JavaScript Framework*. Accedido en mayo de 2025. 2024. URL: <https://vuejs.org/>.
- [79] pytest-dev. *pytest - simple powerful testing with Python*. Accedido en mayo de 2025. 2024. URL: <https://docs.pytest.org/>.

- [80] Python Software Foundation. *unittest — Unit testing framework*. Accedido en mayo de 2025. 2024. URL: <https://docs.python.org/3/library/unittest.html>.
- [81] nose-devs. *nose2 - next generation of nicer testing for Python*. Accedido en mayo de 2025. 2024. URL: <https://docs.nose2.io/>.
- [82] Typicode. *Husky - Git hooks made easy*. Accedido en mayo de 2025. 2024. URL: <https://typicode.github.io/husky/>.
- [83] Commitizen Team. *Commitizen - Create consistent commit messages*. Accedido en mayo de 2025. 2024. URL: <https://commitizen-tools.github.io/commitizen/>.
- [84] OKonet. *lint-staged - Run linters on git staged files*. Accedido en mayo de 2025. 2024. URL: <https://github.com/okonet/lint-staged>.
- [85] Conventional Changelog. *standard-version*. Accedido en mayo de 2025. 2024. URL: <https://github.com/conventional-changelog/standard-version>.
- [86] Ecma International. *ECMAScript 2024 Language Specification*. Accedido en mayo de 2025. 2024. URL: <https://tc39.es/ecma262/>.
- [87] Google Developers. *Web Performance Fundamentals*. Accedido en mayo de 2025. 2024. URL: <https://web.dev/performance-scoring/>.
- [88] JetBrains. *Python Developers Survey 2023*. Accedido en mayo de 2025. 2024. URL: <https://www.jetbrains.com/lp/devecosystem-2023/python/>.