
# Sistema web para generar vectores simples para corte y grabado láser mediante agentes de IA

Chafuelan Gramal, Jefferson Stalyn

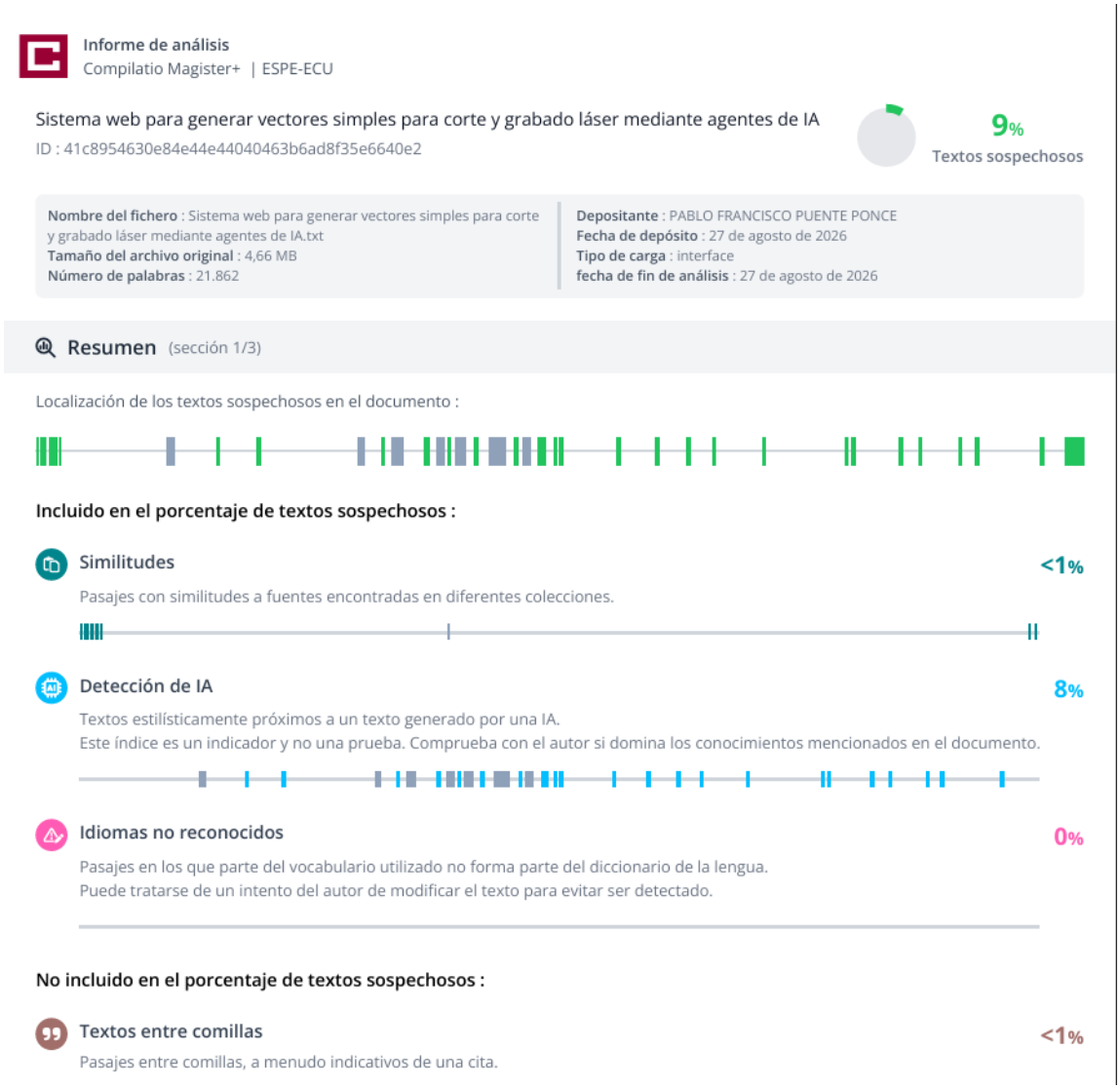
Departamento de Ciencias de la Computación

Carrera de Ingeniería en Tecnologías de la Información

Trabajo de integración curricular, previo a la obtención del título de Ingeniero en Tecnologías de la Información

Mgtr. Ing. Puente Ponce, Pablo Francisco

Santo Domingo, 31 de julio de 2026



Informe de análisis
Compilatio Magister+ | ESPE-ECU

Sistema web para generar vectores simples para corte y grabado láser mediante agentes de IA
ID : 41c8954630e84e44e44040463b6ad8f35e6640e2

9%
Textos sospechosos

**Nombre del fichero** : Sistema web para generar vectores simples para corte y grabado láser mediante agentes de IA.txt
**Tamaño del archivo original** : 4,66 MB
**Número de palabras** : 21.862

**Depositante** : PABLO FRANCISCO PUENTE PONCE
**Fecha de depósito** : 27 de agosto de 2026
**Tipo de carga** : interface
**fecha de fin de análisis** : 27 de agosto de 2026

Resumen (sección 1/3)

Localización de los textos sospechosos en el documento :

Incluido en el porcentaje de textos sospechosos :

Similitudes <1%
Pasajes con similitudes a fuentes encontradas en diferentes colecciones.

Detección de IA 8%
Textos estilísticamente próximos a un texto generado por una IA.
Este índice es un indicador y no una prueba. Comprueba con el autor si domina los conocimientos mencionados en el documento.

Idiomas no reconocidos 0%
Pasajes en los que parte del vocabulario utilizado no forma parte del diccionario de la lengua.
Puede tratarse de un intento del autor de modificar el texto para evitar ser detectado.

No incluido en el porcentaje de textos sospechosos :

Textos entre comillas <1%
Pasajes entre comillas, a menudo indicativos de una cita.

Validar únicamente en FirmaEC.
Firmado electrónicamente por:
PABLO FRANCISCO PUENTE PONCE

**Mgtr. Ing. Puente Ponce, Pablo Francisco**

**Director**

**Departamento de Ciencias de la Computación**

**Carrera de Ingeniería en Tecnologías de la Información**

## Certificación

Certifico que el trabajo de integración curricular: “Sistema web de generación de archivos vectoriales para corte láser mediante agentes de inteligencia artificial” fue realizado por el señor **Chafuelan Gramal, Jefferson Stalyn**, el mismo que cumple con los requisitos legales, teóricos, científicos, técnicos y metodológicos establecidos por la Universidad de las Fuerzas Armadas ESPE; habiéndose efectuado una revisión integral de forma y de fondo conforme a los lineamientos y formatos institucionales vigentes. Además, fue revisado y analizada en su totalidad por la herramienta de prevención y/o verificación de similitud de contenidos; razón por la cual me permito acreditar y autorizar para que se lo sustente públicamente.

**Santo Domingo, 31 de julio de 2026**

Validar únicamente en FirmaEC.
Firmado electrónicamente por:
PABLO FRANCISCO PUENTE PONCE

**Mgtr. Ing. Puente Ponce, Pablo Francisco**

**Director**

**C.C.: 1002771762**

**Departamento de Ciencias de la Computación**

**Carrera de Ingeniería en Tecnologías de la Información**

## Responsabilidad de Autoría

Yo, **Chafuelan Gramal, Jefferson Stalyn**, con cédula de ciudadanía n.° 1728036060, declaro que el contenido, ideas y criterios del trabajo de integración curricular: **Sistema web de generación de archivos vectoriales para corte láser mediante agentes de inteligencia artificial** es de mi autoría y responsabilidad, cumpliendo con los requisitos legales, teóricos, científicos, técnicos, y metodológicos establecidos por la Universidad de las Fuerzas Armadas ESPE, respetando los derechos intelectuales de terceros y referenciando las citas bibliográficas.

**Santo Domingo, 31 de julio de 2026**

**Chafuelan Gramal, Jefferson Stalyn**

**C.C.: 1728036060**

**Departamento de Ciencias de la Computación**

**Carrera de Ingeniería en Tecnologías de la Información**

## Autorización de Publicación

Yo, **Chafuelan Gramal, Jefferson Stalyn**, con cédula de ciudadanía n.º 1728036060, autorizo a la Universidad de las Fuerzas Armadas ESPE publicar el trabajo de integración curricular: **Título: Sistema web de generación de archivos vectoriales para corte láser mediante agentes de inteligencia artificial** en el Repositorio Institucional, cuyo contenido, ideas y criterios son de mi responsabilidad.

**Santo Domingo, 31 de julio de 2026**

**Chafuelan Gramal, Jefferson Stalyn**

**C.C.: 1728036060**

## Dedicatoria

Dedico este trabajo a mis padres, a mi familia y a todas las personas que me brindaron su apoyo en este arduo camino. Con la bendición de Dios, hoy estoy aquí demostrando que, con esfuerzo y dedicación, es posible alcanzar cualquier meta que nos propongamos.

## Agradecimiento

Al dar por concluido este trabajo y el esfuerzo dedicado a lo largo de estos años, expreso mi más profundo agradecimiento a mis padres por su apoyo incondicional en todo momento. A Dios, por darme la fortaleza para no rendirme ante cada obstáculo, y a mis docentes, quienes supieron guiarme y aconsejarme, e ir mucho más allá de su labor como educadores.

## Índice de contenidos

## Índice de tablas

## Índice de figuras

## Resumen

El presente trabajo se basó en el desarrollo de una aplicación web denominada Bezy, la cual está basada principalmente en el uso de agentes de Inteligencia artificial para automatizar el proceso de creación de diseños vectoriales a partir de peticiones en lenguaje natural. La propuesta del proyecto incluye la interfaz web, servicios de orquestación, un pipeline estructurado por los agentes especializados en interpretación, generación y validación, finalizando con la funcionalidad de conversión del archivo generado en SVG a DXF, correspondiente al formato establecido para trabajar en herramientas de software CAD. La investigación se centró en adoptar un enfoque mixto de alcance descriptivo, a su vez, se utilizó Scrum como marco de desarrollo iterativo, opción elegida gracias a su metodología ágil. La parte de la evaluación se establece en 90 casos de prueba, distribuidos equitativamente en distintos niveles como simple, intermedio y complejo, los cuales durante el proceso de evaluación resultó en un total de 121 ejecuciones, las cuales se las tomaron netamente como reintentos debido a distintos fallos durante la ejecución del modelo. El éxito técnico por ejecución se contabilizó en un 89,3%, mientras que los 90 casos obtuvieron una ficha técnica detallando los casos de éxito, así como sus medidas. Los resultados finales llegaron a cumplir con validaciones o reglas que se deben cumplir para el formato de diseños en 2D, tales como rutas cerradas, detección de curvas de Bézier y distinción entre capas de corte y grabado. Con esto se ha podido concluir que la arquitectura permite transformar solicitudes textuales en archivos vectoriales verificables y editables, manteniendo una trazabilidad observada para dicho proceso. No obstante, se debe recalcar que en la etapa final de fabricación, es necesario contar con una revisión humana para detectar posibles errores en el diseño que sean cometidos por el modelo basado en resultados probabilísticos.


**Palabras clave:** agentes de inteligencia artificial, corte láser, gráficos vectoriales, SVG, validación geométrica.

## Abstract

This work focused on the development of a web application called Bezy, which primarily uses artificial intelligence agents to automate the process of creating vector designs from natural language requests. The project proposal includes the web interface, orchestration services, and a pipeline structured by agents specializing in interpretation, generation, and validation. The pipeline culminates in the functionality to convert the generated SVG file to DXF, the format required for CAD software tools. The research employed a mixed-methods approach with a descriptive scope, utilizing Scrum as the iterative development framework, chosen for its agile methodology. The evaluation phase consisted of 90 test cases, evenly distributed across different levels of complexity (simple, intermediate, and complex). These tests resulted in a total of 121 executions, which were considered retries due to various errors encountered during model execution. The technical success rate per execution was 89.3%, and all 90 cases received a technical data sheet detailing the successes and their metrics. The final results met the validation requirements for 2D design formats, such as closed paths, Bézier curve detection, and distinction between cutting and engraving layers. Once the complexity levels were established, a 78.0% success rate per execution was achieved at the most complex level. This was also reflected in an increase in average latency, reaching peaks of 54.57 seconds. This allows us to conclude that the architecture transforms textual requests into verifiable and editable vector files, maintaining traceability throughout the process. However, it is important to emphasize that in the final manufacturing stage, a human review is necessary to detect potential design errors introduced by the probabilistic model.


**Keywords:** artificial intelligence agents, laser cutting, vector graphics, SVG, geometric validation.

# Capítulo I

## Introducción

La inteligencia artificial generativa y los grandes modelos de lenguaje ya no se limitan a teclear frases bonitas: empiezan a razonar sobre el espacio, a entender el mundo físico y a representarlo mediante código [1]. Y aquí entra un detalle práctico que cambia el juego: los gráficos vectoriales escalables (SVG) están hechos de primitivas geométricas —líneas, curvas, polígonos— descritas en XML. Eso significa que, en vez de “pintar” píxeles, un modelo puede generar directamente instrucciones vectoriales a partir de texto; en otras palabras, transformar una idea en una imagen interpretable por máquina sin pasar por el renderizado tradicional [2].

Ahora, podemos pensar el escenario de un taller de corte y grabado láser: antes de que la máquina haga su trabajo, alguien debe crear el archivo vectorial. Y ahí está el problema: esa tarea exige dominio de software CAD y técnicas que mucha gente no tiene. Resultado: emprendedores, estudiantes y aficionados quedan fuera del acceso a prototipado rápido y fabricación digital porque la curva de aprendizaje es demasiado empinada.

La propuesta de este trabajo es abordar justamente esa barrera mediante un sistema web que combine múltiples agentes de IA. ¿Por qué agentes múltiples? Porque dividir responsabilidades entre entidades autónomas funciona mejor para tareas complejas que un solo modelo tratando de hacerlo todo [3]. El motor de diseño de IA consta de tres componentes: el agente intérprete, que extrae las variables matemáticas del lenguaje natural del usuario; el generador, que construye el vector según un formato estricto; y el validador geométrico, que comprueba la validez del diseño. La herramienta recibe las instrucciones y luego utiliza Python para construir el SVG, que posteriormente se procesa de forma habitual. Este último componente es crucial, ya que la literatura científica confirma que los LLMs todavía fallan en los detalles finos o al intentar editar estructuras vectoriales complejas [2]. Para evitar fallos físicos en la producción, el validador calcula, por ejemplo, distancias euclidianas para garantizar que

las rutas geométricas estén completamente cerradas, condición indispensable para que la máquina láser pueda separar piezas del material sin errores.

El objetivo principal y central de la investigación propuesta es evidente, explícito y, a su vez, práctico para la empresa de ser algo que no se queda poco menos que en una reflexión y vislumbra algo que puede ser continuado: crear un sistema web automatizado que sea capaz de generar archivos vectoriales optimizados para el corte y el grabado láser a partir de la intervención de los agentes de la IA. Con respecto a las cuestiones metodológicas, se optó por un enfoque mixto de alcance descriptivo, orientado a caracterizar el comportamiento técnico del sistema y la correspondencia de los archivos generados con las solicitudes planteadas. Respecto a los aspectos de desarrollo, se adopta SCRUM para poder garantizar entregas e iteraciones constantes. La parte de implementación técnica se estructura según un modelo de cliente/servidor, con un frontend interactivo y un backend asíncrono desarrollado a través de Node.js y Python para la gestión de la comunicación de los modelos.

Estructuralmente, el documento se divide en cinco capítulos. En el primer capítulo se desarrolla la problemática del trabajo, la justificación del mismo y los objetivos propuestos. El segundo capítulo incluye los antecedentes y el marco teórico necesario. El tercer capítulo da cuenta de la metodología planteada, mientras que el cuarto capítulo se dedica a presentar el análisis de los resultados. Finalmente, el quinto capítulo da cuenta del producto del trabajo de investigación, las conclusiones obtenidas y las recomendaciones establecidas.

**Planteamiento del problema**

La creación y fabricación de las ejecuciones en cortadoras láser, conforma un proceso que requiere de una gran inventiva para poder idear líneas y la técnica del CAD, y conocimiento de las herramientas del programa que le corresponda al modelo de la cortadora láser, así como medidas y el tipo de material a utilizar. Por lo cual, al usuario normal que no tiene formación en el diseño vectorial, se le pone ante una barrera que lo limita a realizar sus propias ideas.

En la práctica, una persona que trabaje en el rubro de diseños o decoraciones, no cuenta con el dominio de herramientas como AutoCAD, Lightburn o RD works, y es que su estudio es bastante amplio, debido a que con estas herramientas no sirven simplemente para hacer figuras o adornos, sino que permiten hacer planos, convertir diseños de 2D a 3D, entre otras implementaciones que ofrecen. Con esto, este tipo de usuarios no tiene el conocimiento de lo que son capas, escalas, unidades, cierre de trayectorias, etc. Como consecuencia, están obligados a depender de profesionales que ofrezcan este tipo de servicios, esto normalmente aborda el tener que solicitar un archivo apto para fabricación y especificar parámetros, aumentando los costos y posibilidades de error o de obtener un resultado que puede no ser el esperado.

El problema no se limita a crear una imagen visualmente atractiva. Algunas personas prefieren crear archivos 2D específicamente para corte láser. Si desea crear uno, debe tener una trayectoria cerrada, sin segmentos duplicados, con la escala correcta, ubicado en el área de trabajo, con continuidad entre segmentos y compatible con el software de control. Un archivo que se ve bien en pantalla puede no funcionar debido a una trayectoria abierta, elementos fuera del área de trabajo, curvas sin convertir o problemas de unidades. Estos errores generan desperdicio de material, repeticiones del trabajo, pérdida de tiempo y resultados físicos defectuosos.

A pesar de que la inteligencia artificial generativa y los larguísimos modelos de lenguaje ya se están utilizando de manera progresiva, la gran mayoría de las soluciones existentes están enfocadas fundamentalmente a generar imágenes raster o ilustraciones visuales, y no necesariamente a generar ficheros vectoriales validados para la fabricación. Hablamos de que cuando un modelo genera SVG directamente, muchas veces genera un código plausible sintácticamente pero desprovisto de corrección geométrica, o válido para un flujo de corte o de grabado real. En este sentido, cuando se emplea IA generativa sin validación, se puede

trasladar de la fase de diseño manual un problema hacia la fase de corrección técnica posterior.

Ante esta situación, se hace necesaria la implementación de un sistema web que sirva para convertir instrucciones en lenguaje natural a archivos vectoriales para el corte y el grabado láser. Esto se hace necesario debido a que la solución necesita por un lado simplificar el diseño sin que deban hacerse muchas tareas manuales, pero por otro incorporar controles técnicos para asegurar que el resultado sea bueno. Entonces, se propone una arquitectura basada en agentes de inteligencia artificial donde un agente intérprete obtiene las variables y restricciones del prompt, un agente generador formula las instrucciones de construcción vectorial y el servicio Python produce el SVG que posteriormente revisa el validador geométrico antes de su conversión a DXF.

La problemática principal, por lo tanto, se sitúa precisamente en la brecha entre lo que quiere el usuario y poder conseguir un archivo vectorial que sea técnicamente factible de construir como un archivo utilizado por estas máquinas. Esta brecha es una combinación de tres dimensiones: la dimensión de la accesibilidad, porque no todas las personas que se quieren fabricar objetos han dominado la herramienta CAD; una dimensión técnica, porque deben estar bien definidas las reglas geométricas y los tipos de archivo que tienen que utilizarse a través de una máquina de este tipo; y una dimensión tecnológica, porque la construcción de modelos generativos en un entorno de fabricación digital requiere de orquestación, de validación y de control para poder conseguir resultados de calidad que se adapten a lo que el usuario quiere conseguir. Abordar esta brecha permitiría conseguir superar la brecha entre fabricación digital y la democratización del acceso para la fabricación digital, así como mejorar el flujo que hay antes de aplicar las herramientas que permiten utilizar máquinas láser.

## Formulación del problema o pregunta de investigación

A partir del planteamiento anterior, la investigación se orienta a responder la siguiente pregunta:

**¿Cómo desarrollar un sistema web basado en agentes de inteligencia artificial que genere y valide archivos vectoriales SVG y DXF, a partir de instrucciones en lenguaje natural, para facilitar la preparación de diseños destinados a procesos de corte y grabado láser?**

Partiendo de esta pregunta principal, tenemos algunas preguntas específicas que nos dan una idea de la definición del problema de investigación que se plantea solventar:

- ¿Cuáles son los parámetros técnicos que se deben extraer del lenguaje natural del usuario y que pueden pasar a un archivo vectorial, el cual pueda ser usado para corte o grabado láser?
- ¿Cómo debe estructurarse una cadena o pipeline de procesamiento multiagente?
- ¿Qué rol y ejecución tendría el modelo de interpretación, generación de vectores o la validación geométrica?
- ¿Qué métricas permiten evaluar la calidad del sistema en términos de validez geométrica, su tasa de éxito, número de reintento o el tiempo que le toma generar un archivo compatible con herramientas de software de corte láser?

## Objetivos

### *Objetivo general*

Desarrollar un sistema web basado en agentes de inteligencia artificial para la generación automática de archivos vectoriales optimizados para procesos de corte y grabado láser.

***Objetivos específicos***

- Analizar el estado del arte de la generación de gráficos vectoriales mediante IA y definir los requerimientos funcionales para la interpretación de comandos de lenguaje natural, Implementar la plataforma web y los agentes inteligentes capaces de transformar "prompts" de texto en geometrías vectoriales cerradas y escalables.
- Desarrollar el backend del sistema integrando la arquitectura de micro-agentes de IA y la interfaz de usuario frontend.

**Justificación**

La metodológica institucional establece que la justificación debe explicar los fundamentos sobre la relevancia de la investigación, sus efectos y contribuciones, desde una perspectiva teórica, práctica, social y metodológica [4]. Bajo esos criterios, el trabajo de Bezy es justificado porque rellena el espacio entre formular una idea en lenguaje natural y convertir el archivo en un formato técnico, con acceso a un software de diseño de computador y dificultades técnicas de conservar trayectorias, dimensiones, estructura vectorial, formatos y otros aspectos que se pueden analizar antes del proceso de corte y grabado.

Desde una perspectiva teórica, el trabajo presenta un conjunto de modelos de lenguaje, agentes de IA y gráficos SVG; la geometría y conversión DXF en un solo proceso continuo hasta la etapa de fabricación. La literatura advierte que los modelos generativos pueden producir estructuras vectoriales plausibles sin garantizar por sí solos la consistencia geométrica de los resultados [2]. Por esta razón, la propuesta no se limita a generar una representación visual, sino que incorpora interpretación estructurada, validación determinista, trazabilidad y diferenciación entre el estado técnico del pipeline y la correspondencia del archivo con el prompt.

Desde un punto de vista práctico, el prototipo facilita a quienes no poseen conocimientos avanzados de software CAD el diseño, la previsualización y la descarga de

archivos desde una interfaz web. Esto significa que pueden definir sus propios diseños, visualizar los resultados y obtener archivos SVG y DXF. La automatización de los procesos de interpretación, creación, verificación, almacenamiento y descarga, elimina las tareas innecesarias de preparación de archivos y otorga a los usuarios finales control sobre su documentación al contar con un aplicativo similar a otras pagina de IA generativa.

Desde una perspectiva social, el proyecto beneficia a estudiantes, emprendedores y pequeños talleres interesados en la creación de prototipos o la impresión de diseño digital, ya que a menudo no están familiarizados con el diseño vectorial. Para reducir las barreras de acceso a la innovación y las herramientas de desarrollo tecnológico, el proyecto se alinea con el Objetivo de Desarrollo Sostenible 9 [5].

Desde un punto de vista metodológico, la investigación propone tener un análisis en base a métricas y distinción entre los casos de prueba que se analizaron, la ejecución, trazabilidad del pipeline de agentes y el archivo final que deberá ser evaluado previamente antes de su proceso de fabricación. Se maneja la telemetría con la herramienta Langfuse, por otra parte, se tiene los registros que se envían a una tabla en especifico en la base de datos de supabase, además, la revisión cualitativa de los SVG permite un estudio diferente de la estabilidad operativa, la validez geométrica y la fidelidad a la solicitud. En base a las pruebas y trabajos realizados, el presente proyecto puede verse como un punto de inicio para futuros trabajos que estén enfocados en la misma área de diseño vectorial.

# Capítulo II

## Estado del arte

La investigación más reciente relacionada con la generación de gráficos SVG se ha centrado en disminuir la dependencia de herramientas de diseño especializadas y en convertir descripciones en lenguaje natural en archivos vectoriales editables. En esta línea, Chat2SVG combina modelos de lenguaje con modelos de difusión para generar plantillas SVG a partir de primitivas geométricas y posteriormente completar la complejidad de las rutas, con el propósito de mejorar la fidelidad visual, la regularidad de las formas y la alineación semántica respecto al prompt [6]. Este enfoque se relaciona con el presente trabajo porque también busca facilitar la generación de gráficos a usuarios que no poseen un dominio avanzado del software vectorial.

Otro trabajo relevante es LLM4SVG, que busca hacer frente a las limitaciones de los LLM para la generación de gráficos vectoriales complejos. Sus autores afirman que los modelos generales son capaces de generar primitivas simples, pero suelen ser incapaces de generar secuencias de rutas, atributos y relaciones de oclusión cuando la estructura SVG se vuelve más compleja. Para tratar de corregir este problema, proponen el uso de tokens semánticos y una arquitectura basada en módulos que sea capaz de separar las instrucciones vectoriales de los parámetros geométricos como ángulos o cálculos de geometría [7]. Este trabajo apoya la idea de una arquitectura con validación geométrica, ya que un archivo generado en formato SVG, a pesar de ser código capaz de ser interpretado por un modelo de IA, no asegura que su interpretación y conversión a DXF sea técnicamente correcto para el proceso de lineas continuas de corte.

Además de estas propuestas, también se han realizado intentos de utilizar la generación de SVG más allá de las formas básicas y en diseños más complejos. SVGCraft aprovecha el uso de LLM para producir la distribución de formas en el SVG y luego combina esto con mecanismos de difusión y atención para generar dibujos vectoriales más detallados [8]. Como complemento a esto, OmniSVG proporciona un modelo multimodal que discretiza los

comandos y coordenadas SVG, además de introducir el conjunto de datos MMSVG-2M, lo que resulta en un mejor rendimiento de generación de vectores condicionales [9]. Si bien estos trabajos logran grandes avances en la generación visual y multimodal, su principal interés radica en el diseño gráfico, no en la fabricación real mediante corte/grabado láser.

En el contexto de los modelos específicamente entrenados para texto a SVG, SVGen presenta su dataset SVG-1M y un generator SVG code from natural language model basado en anotaciones, razonamiento estilo cadena de pensamientos, curricular learning y reinforcement learning [10]. Su hincapié en la completitud estructural es interesante para nuestra investigación porque los archivos destinados a fabricación digital deben no sólo tener consistencia visual, sino que deben además tener rutas cerradas, dimensiones consistentes y ser conformes con los formatos industriales.

Como conclusión, el trabajo VectorEdits hace aparecer que la edición por instrucciones de SVG sigue siendo un problema abierto, hasta los modelos de más reciente ’state-of-the-art’ les cuesta dar soportes precisos o válidos a un SVG ya existente [11]. Este hecho hace totalmente justificable el incorporar en el sistema propuesto un validador geométrico. A diferencia de lo que han sido los antecedentes que hemos revisado, en la investigación que aquí se propone nos centramos en un flujo multiagente para corte y grabado láser, donde la generación de SVG se ve acompañada por la extracción de variables matemáticas, por estrictas reglas de formato y por la verificación automática de condiciones geométricas necesarias para la fabricación.

**Marco teórico**

En esta sección se reúnen todas las perspectivas necesarias para explicar el flujo lógico y proceso que sigue el pipeline, desde la captura del lenguaje natural hasta la posterior interpretación y transformación en un archivo vectorial técnicamente evaluable. En Bezy, los conceptos de modelos de lenguaje, agentes de inteligencia artificial, gráficos vectoriales, geometría computacional y trazabilidad, por mencionar algunos cuantos temas, no operan de

forma aislada, sino que se complementan para conformar un pipeline capaz de dar una representación de un SVG, verificar la geometría, convertirlo a un archivo DXF y registrar la evidencia del proceso.

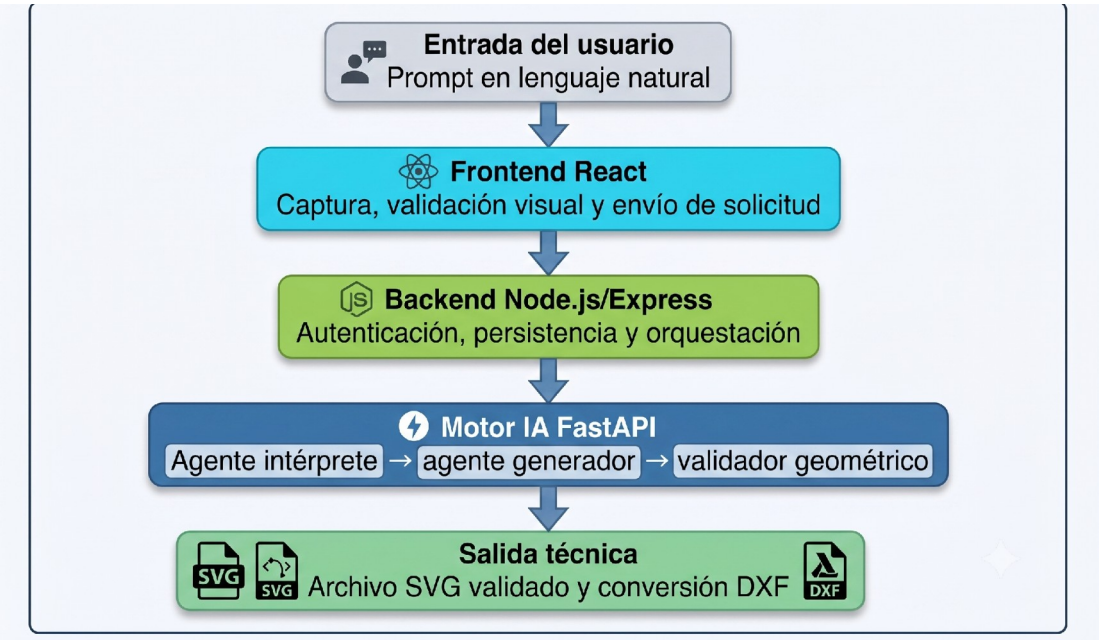
De acuerdo con la guía de elaboración del trabajo de integración curricular, esta sección no se limita a recopilar definiciones. Cada fundamento se relaciona con una decisión del sistema, un componente de la arquitectura o un indicador utilizado posteriormente en la evaluación. De esta manera, el marco teórico proporciona las explicaciones que permiten interpretar los resultados presentados en el Capítulo IV.

### *Fundamentos tecnológicos del sistema*

Un sistema orientado a fabricación digital requiere integrar interacción de usuario, procesamiento distribuido, inteligencia artificial, validación técnica, persistencia y observabilidad. En esta arquitectura se establece la separación de responsabilidades basándonos en los conceptos de micro servicios y la comunicación entre componentes. En el caso de Bezy, esto nos ha permitido evaluar el comportamiento probabilístico del modelo y tratar de establecer reglas estrictas para obtener resultados funcionales.

**Figura 1**

*Flujo conceptual del sistema Bezy para generación de archivos vectoriales*




Como se mencionó, Bezy cuenta con una estructura modular en donde las diferentes capas del proyecto permiten manejar posibles errores de manera más optimizada al separar los escenarios de fallos. Como se muestra en la Figura 1, la interfaz gráfica o denominado frontend recibe la petición del usuario, el backend procede a capturar las diferentes peticiones entrantes como autenticación, validación e ingreso de usuarios y de la misma forma la administración de los diseños y registros de archivos generados. Una ventaja de este enfoque es que el cambio de un modelo de lenguaje en alguno de los agentes no afectará a la interfaz ni al almacenamiento, sus tareas están completamente separadas.

**Tabla 1**

*Fundamentos tecnológicos y aplicación en Bezy*

| Capa | Tecnología | Implementación en Bezy | Evidencia técnica |
|---|---|---|---|
| Presentación | React y TypeScript | Implementa el chat, el visor SVG, el historial y las acciones de descarga. | Prompt enviado, vista previa y archivo seleccionado. |
| Orquestación web | Node.js y Express | Expone `/api/ai/generate-vector`, autentica la solicitud y coordina el flujo. | Estado del caso, respuesta HTTP y registro asociado. |
| Servicio especializado | FastAPI | Recibe parámetros estructurados y ejecuta el pipeline de agentes y validación. | Respuesta del servicio y reporte técnico. |
| Modelos de IA | DeepSeek V4 Pro | El modelo cumple funciones diferenciadas de interpretación y generación mediante prompts especializados; las instrucciones resultantes son procesadas por el servicio Python para construir la representación SVG. | Salidas por agente, tokens y latencia. |
| Persistencia | Supabase | Conserva diseños, métricas y rutas de los archivos SVG y DXF. | Registros relacionados y archivos finales. |
| Observabilidad | Langfuse | Registra trazas, llamadas, regeneraciones, tiempos, tokens y errores. | Telemetría de las 121 ejecuciones. |

*Nota.* Elaboración propia con base en la arquitectura implementada y en las fuentes de telemetría del sistema.

La Tabla 1 muestra que el valor del stack se encuentra en su integración. React y TypeScript materializan la interacción; Node.js y Express controlan el proceso; FastAPI expone el pipeline especializado; Supabase conserva los resultados; y Langfuse permite reconstruir el comportamiento interno de las ejecuciones. Esta relación de tecnologías enfocadas en distintas áreas, hace que su validez técnica sea considerable gracias a su manejo e interpretación a tiempos de respuesta, consumo de tokens y estado final de archivos generados.

***Modelos de lenguaje grandes y agentes de inteligencia artificial***

Los modelos de lenguaje grandes son sistemas que han sido entrenados con la finalidad de procesar secuencias, seguir instrucciones y producir contenido estructurado. Su aplicación en Bezy no se concentra simplemente en redactar texto, sino que lo toma y hace una interpretación de su estructura lingüística, a partir de ello, establece los parámetros que le

servirán al siguiente agente encargado de construir la representación enviada. El formato SVG resulta adecuado para esta tarea porque describe líneas, curvas, polígonos y atributos mediante sintaxis textual, si buscamos graficadores de SVG, veremos que cada propiedad puede ser editable mediante código, es lo que usan mucha tecnologías para crear iconos o archivos más optimizados que una imagen en otro formato. La literatura permite analizar y dar como válido este formato gracias a su vínculo entre lenguaje y representación visual [1].

La generación mediante un modelo de lenguaje es probabilística. Esto quiere decir que los resultados que genere el modelo, sin importar cual sea, nos dará diferentes resultados a pesar de tener el mismo input o petición por parte del usuario. Investigaciones recientes señalan dificultades para mantener relaciones geométricas, atributos y rutas correctamente formadas en gráficos vectoriales complejos [2], [11].En Bezy, este riesgo.se aborda y trata de solventarse separando la generación de la validación.Eso.se maneja mediante la generación primero de un Script en Python, con esto se tiene obtiene un mejor resultado al interpretar medidas, ángulos o peticiones con valores matemáticos.

La configuración evaluada utiliza el modelo Deepseek V4 Pro con funciones diferenciadas dentro del pipeline. El agente intérprete toma la petición del usuario y lo convierte en parámetros técnicos que serán enviados mediante un JSON, por otra parte, el agente generador recibirá la instrucción para programar en python construyendo así la representación vectorial. Esta configuración se mantuvo constante en las pruebas que se realizaron, por lo que los resultado obtenidos corresponden en gran medida a las capacidades del modelo usado y no a lo modelo de lenguaje en general, esto aplica de mejor manera en el intérprete con el modelo gratuito de Llama.

Un agente de inteligencia artificial puede definirse como una entidad que percibe información, toma decisiones y ejecuta acciones orientadas a un objetivo [12]. En sistemas basados en modelos de lenguaje, los agentes pueden especializarse, intercambiar resultados y utilizar herramientas para resolver tareas compuestas [13], [14]. Esta perspectiva se aplica en

Bezy mediante una secuencia de responsabilidades, en lugar de una única llamada encargada de interpretar, generar, validar y exportar simultáneamente.

**Tabla 2**

*Aplicación de los agentes y componentes del pipeline de Bezy*

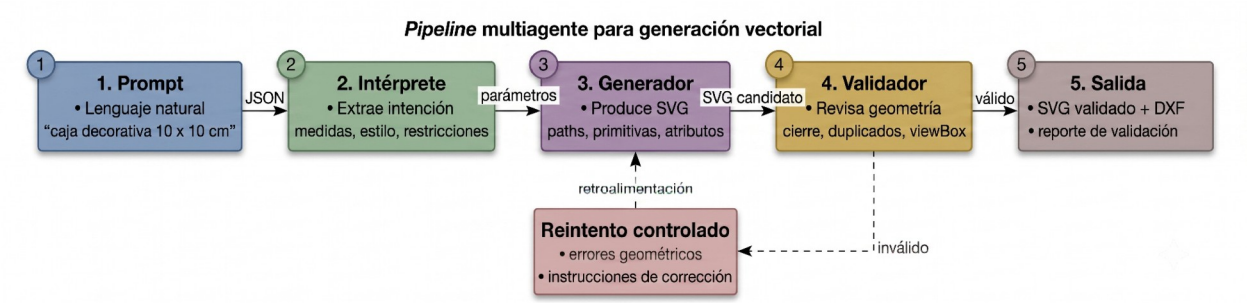
| Componente | Tecnología | Implementación en Bezy | Dato registrado |
|---|---|---|---|
| Intérprete | DeepSeek V4 Pro | Convierte el prompt en tema, dimensiones, unidades, estilo y restricciones estructuradas. | Tiempo del agente, tokens y parámetros interpretados. |
| Generador | DeepSeek V4 Pro | Formula instrucciones programáticas a partir del objeto recibido y de las reglas internas para construir el SVG. | Tiempo, tokens de salida, estado y regeneraciones. |
| Validador | Algoritmos geométricos | Analiza el SVG, calcula rutas cerradas y registra errores antes de aceptar el archivo. | Validez, rutas abiertas o cerradas y errores geométricos. |
| Conversor | Librería de conversión DXF | Transforma la geometría validada a entidades compatibles con aplicaciones CAD/CAM. | Existencia y ruta del archivo DXF. |
| Orquestador | Express, FastAPI y Langfuse | Coordina las etapas y conserva la relación entre caso, ejecución, llamadas y resultado. | Identificador de traza, latencia total y estado final. |

*Nota.* La configuración de modelos y componentes corresponde a la utilizada durante las 90 pruebas oficiales.

La Tabla 2 relaciona cada concepto con su implementación y con la evidencia que produce. La especialización permite identificar en qué etapa se concentra la latencia, qué componente produjo un error y si una regeneración logró recuperar la ejecución. Esta capacidad de observación se relaciona con los planteamientos multiagente, donde los roles definidos y la comunicación estructurada facilitan la resolución de tareas complejas [3].

**Figura 2**

*Secuencia lógica del pipeline multiagente aplicado a generación vectorial*




Veas en la figura 2. El hecho de que un generador produzca un archivo esto no significa que dicho archivo sea válido para su proceso de corte o grabado, al momento de hacerlo en una máquina. La validación actúa como una barrera entre las probabilidades del generador y la realidad del archivo. Si la respuesta no se puede procesar debido a errores en el código generado por Python, las rutas no están cerradas u otros errores que infrinjan las reglas geométricas, el sistema registrará un error y podría intentarlo de nuevo o generar un nuevo archivo.

La ingeniería de prompts es el proceso de diseñar el conjunto de instrucciones para el modelo. En Bezy, tiene dos niveles de uso. El primero se da cuando el usuario define las instrucciones. El segundo se da cuando se configuran los avisos internos que definen el formato de respuesta, las restricciones geométricas y los criterios que debe cumplir un agente. Se definieron 90 prompts oficiales antes de realizar las primeras pruebas, con lo que se pudo usar la complejidad como criterio de comparación durante la etapa de evaluación.

El contrato de datos complementa la ingeniería de prompts. El intérprete entrega campos predefinidos en lugar de una respuesta libre, como campos se tienen, el análisis que realizó, el modo seleccionado y las medidas, entre otros parámetros importantes. Esto permite verificar con typescript y pidentic que la información necesaria está presente antes de continuar dando error si alguno de estos no está por alguna razón. En Bezy, estos contratos reducen las

ambigüedades entre los diferentes servicios y facilitan la vinculación de cada parámetro interpretado con su traza, muy útil a la hora de usar herramientas como Langfuse.

La validación geométrica aplica criterios matemáticos a las trayectorias. Para comprobar el cierre de una ruta se utiliza la distancia euclidiana entre el punto inicial $P_0$ y el punto final $P_n$:

$$d\left(P_0, P_n\right)=\sqrt{\left(x_n-x_0\right)^2+\left(y_n-y_0\right)^2}\leq \varepsilon ,$$

donde $\varepsilon$ representa la tolerancia máxima aceptada. En Bezy, esta comprobación alimenta el porcentaje de rutas cerradas y el reporte de errores geométricos. Así, un concepto matemático se convierte en un indicador observable dentro de la matriz de evaluación.

***Gráficos vectoriales, geometría computacional y fabricación digital***

Los gráficos vectoriales se basan en entidades matemáticas, mientras que las imágenes rasterizadas se basan en valores de una matriz de píxeles. Esta diferencia es importante en el corte láser, ya que la máquina debe interpretar la trayectoria en términos de movimiento, no solo la imagen. En Bezy, los gráficos vectoriales permiten convertir las solicitudes del usuario en elementos editables, validables y exportables.

**Tabla 3**

*Comparación de representaciones e implementación vectorial en Bezy*

| Criterio | Representación rasterizada | SVG o DXF | Implementación en Bezy |
|---|---|---|---|
| Estructura | Matriz de píxeles sin rutas explícitas. | Primitivas, coordenadas y entidades geométricas. | El pipeline produce SVG como XML estructurado. |
| Previsualización | Se presenta como imagen plana. | El navegador renderiza directamente el SVG. | React muestra el resultado sin convertirlo previamente a mapa de bits. |
| Validación | No permite comprobar contornos mediante rutas. | Los elementos pueden analizarse individualmente. | El validador examina `path`, cierres y errores geométricos. |
| Edición y escala | La ampliación puede reducir la definición. | Las rutas conservan su estructura al escalarse. | El usuario obtiene un SVG editable y dimensionable. |
| Fabricación | Requiere vectorización para definir movimientos. | DXF representa entidades para CAD/CAM. | El conversor genera DXF desde el SVG previamente validado. |

*Nota.* Bezy utiliza SVG como representación intermedia validable y DXF como formato de intercambio técnico.

Bezy utiliza SVG como formato intermedio (porque se puede generar como texto, previsualizar en el navegador y validar) el cual puede permitir la definición de elementos como path o circle, además, DXF sirve como archivo de intercambio para enviar archivos a programas CAD/CAM. Los archivos se convierten entre ambos formatos mediante reglas programadas para evitar que se tome una segunda decisión generativa que pudiera alterar la geometría ya validada.

**Figura 3**

*Relación entre generación SVG, validación y fabricación mediante DXF*

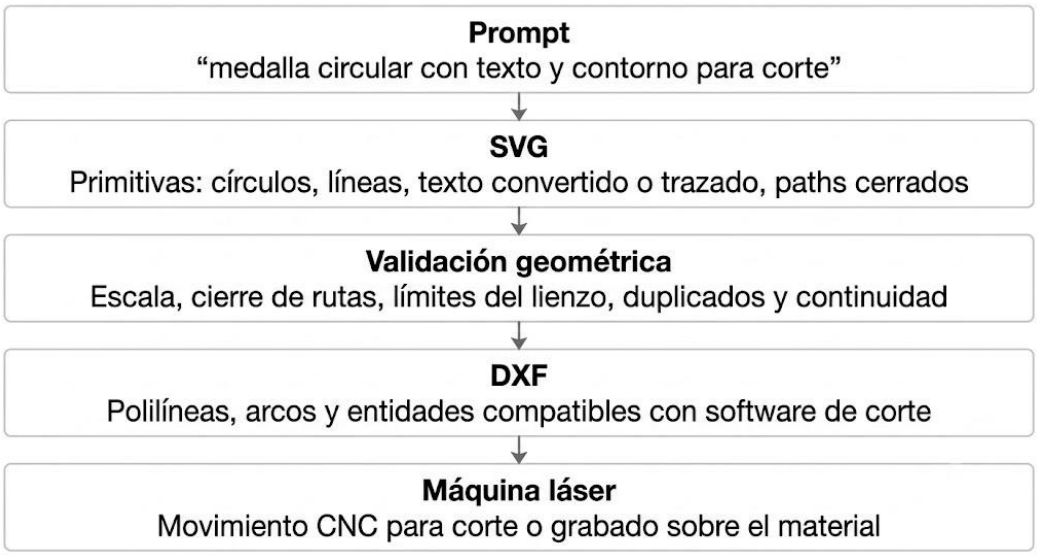



Figura 3 muestra un diagrama de flujo con la función de cada formato: SVG conecta la intención textual con la representación gráfica, el validador comprueba la integridad geométrica y DXF facilita la revisión en software técnico. De esta forma, un archivo se acepta en función de lo que se puede observar, no solo por su apariencia.

**Tabla 4**

*Criterios geométricos aplicados en Bezy*

| Criterio | Procedimiento implementado | Indicador registrado | Estado |
|---|---|---|---|
| Cierre de rutas | Calcula la distancia entre los puntos inicial y final con una tolerancia. | Porcentaje de rutas cerradas y trazos abiertos. | Implementado |
| Estructura SVG | Analiza si el documento puede procesarse y contiene geometría utilizable. | Campo `es_valido_svg`. | Implementado |
| Complejidad geométrica | Cuenta rutas, curvas Bézier, agujeros interiores y piezas sueltas. | Campos técnicos de la matriz de evaluación. | Implementado |
| Compatibilidad DXF | Convierte las entidades admitidas después de la validación del SVG. | Archivo DXF y ruta de almacenamiento. | Implementado |
| Límites del lienzo | Contrasta la geometría con el `viewBox` y las dimensiones solicitadas. | Dimensiones y advertencias del diseño. | Parcial |
| Operación láser | Clasifica las trayectorias de corte y grabado mediante grupos SVG y colores de trazo. | Grupos `corte`/`grabado` y capas DXF `CUT`/`ENGRAVE`. | Implementado |

*Nota.* El estado distingue las comprobaciones utilizadas en la evaluación de las capacidades todavía previstas.

La geometría computacional aporta los procedimientos para medir y comparar estas propiedades. En la evaluación de Bezy, los criterios que se toman en cuenta, tal como se observa en la tabla 4, se utilizaron para distinguir entre un archivo que a simple vista puede parecer correcto, pero que realmente, dentro de un software como RdworksV8 se debe validar que las líneas sean continuas y verificar los parámetros en base al color de la línea

La fabricación digital transforma la información proveniente, por ejemplo XML, en operaciones físicas que pueden ser trabajadas con el formato DXF. Por ejemplo, en corte laser (normalmente con lineas color negro) los contornos cerrados representan las piezas que deben separarse del material, mientras que los detalles interiores como figuras o imagenes de baja resolucion pueden asignarse a una operacion de grabado superficial. En el modo geometrico de Bezy, el agente generador organiza el contorno con una linea de corte de trazo negro y los elementos decorativos se los define con un id de grabado con color rojo para el trazo. Durante

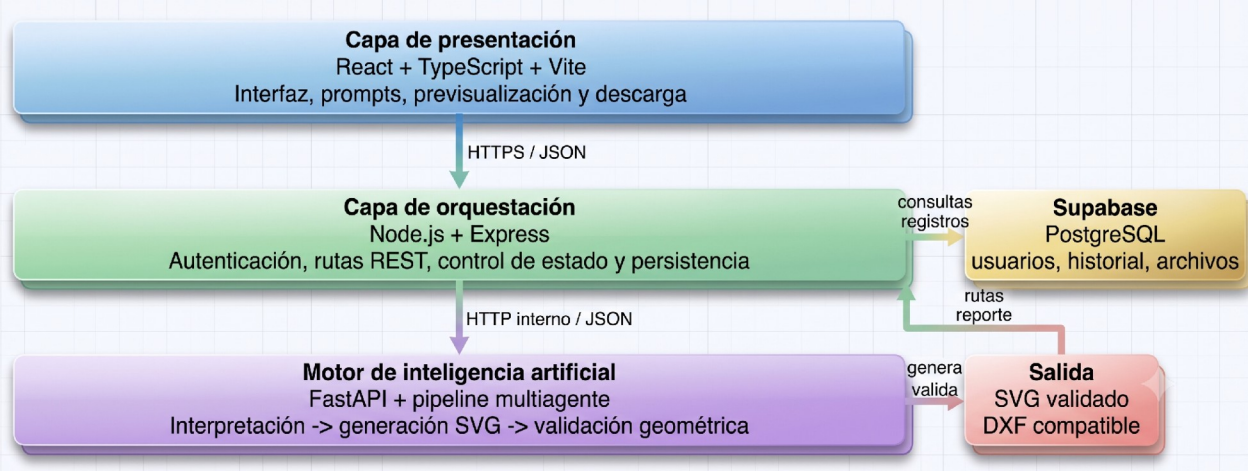
la conversión, el identificador del grupo actúa como criterio principal y el color del trazo funciona como mecanismo redundante para asignar las entidades a las capas DXF CUT y ENGRAVE. Esta clasificación conserva la intención de cada trayectoria, aunque el operador debe revisar el archivo y configurar posteriormente la potencia y velocidad correspondientes en el software de la máquina.

***Arquitectura de software, cliente-servidor y microservicios***

El modelo cliente/servidor implica que hay una diferenciación entre la interfaz que hace solicitudes y el servicio que las ejecuta. Los microservicios van aún más allá al dividir cada uno de los servicios en funciones independientes que se comunican a través de contratos predefinidos. En Bezy, empleamos estos principios para desconectar las capas de interfaz de usuario, lógica de negocio, procesamiento de inteligencia artificial, persistencia y observabilidad.

**Figura 4**

*Arquitectura lógica de Bezy organizada en capas y servicios especializados*




La Figura 4 muestra que el frontend React actúa como cliente y que el backend Node.js/Express centraliza autenticación, estados y enrutamiento. El servicio FastAPI ejecuta el pipeline especializado, mientras que Supabase conserva datos y archivos. Las comunicaciones

REST y los mensajes JSON permiten que estos componentes intercambien información sin compartir el mismo lenguaje de implementación.

**Tabla 5**

*Responsabilidades arquitectónicas y aplicación en Bezy*

| Componente | Tecnología | Implementación y comunicación | Evidencia producida |
|---|---|---|---|
| Frontend | React y TypeScript | Envía solicitudes JSON al backend y renderiza la respuesta SVG. | Prompt, parámetros e interacción visual. |
| API principal | Node.js y Express | Expone endpoints REST, autentica y comunica frontend, FastAPI y Supabase. | Estado HTTP, identificador y rutas de archivo. |
| Servicio de IA | FastAPI y Pydantic | Recibe esquemas tipados y ejecuta agentes, validación y conversión. | Resultado vectorial y reporte técnico. |
| Persistencia | Supabase | Relaciona diseños, métricas y archivos mediante identificadores. | Registros y almacenamiento de SVG/DXF. |
| Observabilidad | Langfuse | Instrumenta las llamadas de los agentes dentro de cada traza. | Latencia, tokens, regeneraciones y errores. |

*Nota.* Las evidencias producidas por cada componente se utilizaron para consolidar la matriz y la telemetría.

En una solicitud típica, el usuario envía un prompt desde el frontend; el backend crea el registro y remite los datos al servicio especializado; DeepSeek V4 Pro interpreta la solicitud y, mediante el rol generador, formula el resultado vectorial; el validador comprueba la geometría; el conversor produce el DXF cuando corresponde; y el backend actualiza el estado y devuelve las rutas de descarga. Paralelamente, Langfuse registra las llamadas y tiempos de los agentes.

Esta arquitectura se relaciona con la mantenibilidad porque permite sustituir un componente sin rediseñar todo el sistema; con la interoperabilidad porque los servicios utilizan HTTP y JSON; y con la observabilidad porque cada etapa produce evidencia. En la investigación, la observabilidad convirtió la arquitectura en una fuente de datos: permitió distinguir casos, ejecuciones, regeneraciones internas, errores y tiempo por agente.

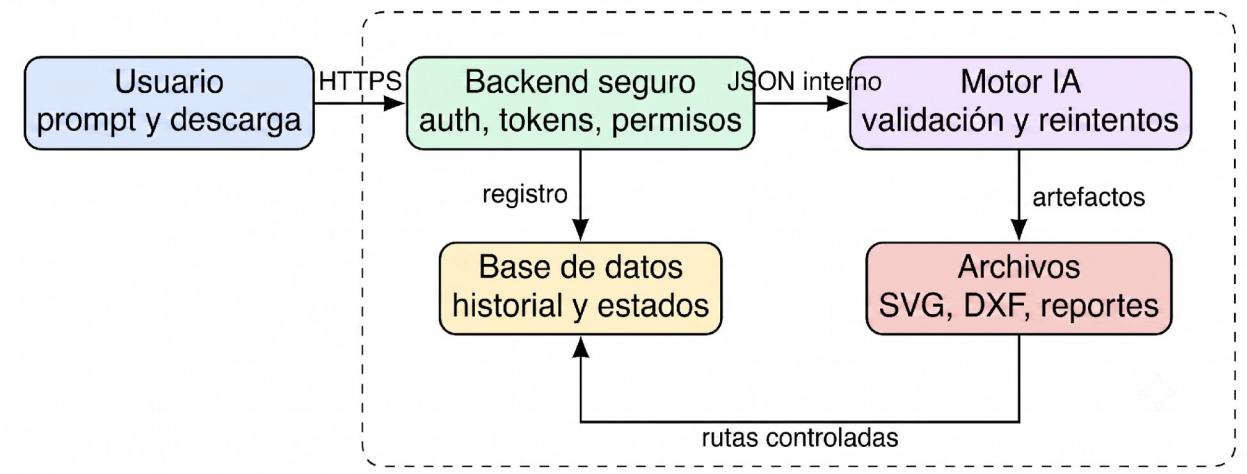
***Seguridad, privacidad, trazabilidad y normas aplicables***

La seguridad de la información se apoya en los principios de confidencialidad, integridad y disponibilidad. En Bezy, la confidencialidad se relaciona con la protección de credenciales, claves de API y tokens; la integridad incluye tanto la consistencia de los registros como la validez técnica de los archivos; y la disponibilidad se refleja en el manejo controlado de fallos para evitar que una excepción interrumpa toda la aplicación.

Las claves de los proveedores se mantienen en variables de entorno del lado servidor y no se exponen en el navegador. El backend controla el acceso y los estados de cada solicitud, mientras que la validación geométrica actúa como un control de integridad del archivo. Si una ejecución falla, el sistema conserva el error y la traza en lugar de presentar el resultado como válido.

**Figura 5**

*Controles de seguridad y trazabilidad aplicados al flujo de Bezy*




La Figura 5 muestra que los controles se distribuyen entre frontend, backend, servicio de inteligencia artificial y almacenamiento. A diferencia de las otras dos, esta distribución no concentra todas las credenciales y responsabilidades en una sola parte. La privacidad se garantiza mediante la minimización de datos (solo se conserva la información necesaria para autenticar al usuario) y la separación de los datos que permiten identificar al usuario (su

nombre, correo electrónico, etc.) de los datos utilizados para la evaluación (datos técnicos utilizados para evaluar la ejecución). Además, se trabaja con JSON web token para tener un periodo de tiempo en el que la cuenta puede estar abierta, luego de ese lapso se cierra, obligando a volver a iniciar sesión.

Se garantiza la trazabilidad, es decir, la posibilidad de reconstruir lo ocurrido durante una ejecución. En Bezy, esto incluye fechas, estado, latencia del tiempo de respuesta de cada agente, tokens, regeneraciones, errores, validación geométrica y archivos finales. En Langfuse, telemetría interna. En Supabase, información funcional y las rutas de los resultados. La posibilidad de combinar ambas fuentes permitió analizar las 90 pruebas oficiales sin mezclar un caso de prueba con sus intentos adicionales.

**Tabla 6**

*Normas de referencia, controles y alcance en Bezy*

| Norma o marco | Control aplicado en Bezy | Evidencia observable | Alcance |
|---|---|---|---|
| ISO/IEC 27002:2022 [15] | Autenticación, control de acceso, protección de credenciales y separación de secretos del frontend. | Configuración del servidor, tokens y estados de acceso. | Referencial |
| ISO/IEC 25010:2023 [16] | Adecuación funcional, eficiencia de desempeño, compatibilidad, fiabilidad, seguridad y mantenibilidad. | Requisitos, trazas, latencia, errores, archivos y resultados de las pruebas. | Referencial |
| Observabilidad | Instrumentación de agentes y registro del estado de cada ejecución. | Trazas, latencia, tokens, regeneraciones y errores. | Implementado |

*Nota.* Las normas se emplean como referencia de diseño; la tabla no representa una certificación formal del prototipo.

Estos datos no se utilizan como certificación formal ni como declaración de conformidad. La norma ISO/IEC 25010:2023 se utiliza para vincular los requisitos y las pruebas con las características de calidad del producto de software, y la norma ISO/IEC 27002:2022 se utiliza para seleccionar los controles de seguridad aplicables al prototipo. La observabilidad se

presenta como una práctica implementada en Bezy y no como una norma independiente. Adicionalmente, se mantiene la supervisión humana como salvaguarda metodológica, debido a que un archivo geométricamente válido puede no representar completamente la intención del usuario o puede requerir la comprobación de que cada trayectoria fue asignada a la operación láser adecuada.

***Metodología Scrum aplicada al desarrollo***

Scrum es un marco de trabajo ágil orientado al desarrollo iterativo e incremental de productos en contextos donde los requisitos, las decisiones técnicas y las prioridades pueden evolucionar durante la ejecución. En lugar de organizar el proyecto como una secuencia rígida de etapas cerradas, Scrum divide el trabajo en periodos de duración limitada denominados *Sprints*. Al finalizar cada Sprint se obtiene un incremento verificable, lo que permite revisar el avance, detectar problemas y ajustar las actividades posteriores [15].

La base de Scrum es el empirismo, entendido como la toma de decisiones a partir de la observación del trabajo realizado. Este enfoque se sostiene en tres pilares. La *transparencia* exige que las tareas, avances, dificultades y resultados sean visibles; la *inspección* implica revisar periódicamente el producto y el proceso; y la *adaptación* permite modificar prioridades o soluciones cuando la revisión evidencia desviaciones. En Bezy, estos pilares se relacionan con el seguimiento de incrementos funcionales, la revisión semanal del prototipo y la incorporación progresiva de mejoras en la arquitectura, la generación vectorial y la interfaz.

Scrum también promueve los valores de compromiso, foco, apertura, respeto y coraje. Su aplicación en el proyecto se refleja en la responsabilidad de completar los objetivos de cada iteración, concentrar el trabajo en el incremento planificado, comunicar limitaciones técnicas, considerar las recomendaciones del tutor y modificar decisiones cuando una alternativa no proporciona los resultados esperados. La sustitución del proveedor y del modelo utilizado para la generación, así como la incorporación posterior de mecanismos de observabilidad, constituyen ejemplos de adaptación técnica dentro del desarrollo de Bezy.

El marco define tres responsabilidades principales. El *Product Owner* prioriza las necesidades y procura maximizar el valor del producto; el *Scrum Master* facilita la aplicación del marco y contribuye a resolver impedimentos; y los *Developers* transforman los elementos priorizados en incrementos funcionales. Para Bezy, las responsabilidades de Product Owner y Scrum Master fueron asumidas por el Ing. Puente Ponce Pablo Francisco Mgtr, mientras que Jefferson Stalyn Chafuelan Gramal desempeñó la responsabilidad de Developer. Esta distribución corresponde a una adaptación académica con un único desarrollador y acompañamiento semanal del tutor.

Los principales artefactos de Scrum son el *Product Backlog*, el *Sprint Backlog* y el incremento. El Product Backlog reúne y prioriza las funcionalidades, mejoras y actividades necesarias; el Sprint Backlog selecciona los elementos que serán abordados durante una iteración; y el incremento representa el resultado integrado y verificable obtenido al finalizarla. En Bezy, estos artefactos permiten organizar elementos como la arquitectura inicial, autenticación, interfaz web, agentes de inteligencia artificial, generación SVG, conversión DXF, validación geométrica, almacenamiento, telemetría y pruebas.

Los eventos de Scrum proporcionan oportunidades de planificación, inspección y adaptación. La planificación del Sprint establece su objetivo y actividades; el seguimiento permite revisar el progreso; la revisión del Sprint examina el incremento junto con las partes interesadas; y la retrospectiva identifica mejoras para el siguiente ciclo. Debido a la naturaleza individual del desarrollo de Bezy, no se realizó una reunión diaria formal con un equipo. En su lugar, se mantuvo seguimiento personal de las actividades y se realizaron reuniones semanales con el tutor, orientadas principalmente a revisar avances, recibir recomendaciones y ajustar el proyecto y el documento de integración curricular.

**Tabla 7**

*Elementos de Scrum y aplicación conceptual en Bezy*

| Elemento | Función en Scrum | Aplicación en Bezy |
|---|---|---|
| Product Owner | Prioriza necesidades y valida el valor del incremento. | El tutor orientó prioridades y revisó semanalmente los avances funcionales y documentales. |
| Scrum Master | Facilita el proceso y ayuda a resolver impedimentos. | El tutor proporcionó recomendaciones metodológicas y técnicas durante el seguimiento. |
| Developer | Construye y verifica el incremento. | El estudiante desarrolló frontend, backend, agentes, validación, integración y pruebas. |
| Product Backlog | Reúne y prioriza el trabajo pendiente. | Se deriva de los objetivos, actividades y arquitectura aprobados en el Anexo 2, complementados con los requisitos del sistema. |
| Sprint Backlog | Define las actividades comprometidas para un Sprint. | Organiza tareas técnicas relacionadas con componentes y entregables específicos de Bezy. |
| Incremento | Resultado integrado y potencialmente verificable. | Corresponde a una versión funcional del sistema o a un componente integrado y comprobable. |
| Inspección y adaptación | Revisa el avance e incorpora mejoras. | Se realizó mediante seguimiento personal y reuniones semanales de revisión con el tutor. |

*Nota.* Scrum se aplicó de forma adaptada a un proyecto académico individual, conservando la planificación iterativa, los artefactos, la revisión de incrementos y la adaptación continua.

La aplicación de Scrum en Bezy permitió relacionar la evolución técnica con incrementos observables. Los elementos que ya se encontraban definidos en el perfil aprobado, como la arquitectura de agentes, la generación SVG, la validación geométrica, la conversión DXF y la interfaz web, sirvieron como base inicial del Product Backlog. Durante el desarrollo, estos elementos fueron refinados mediante decisiones de implementación, correcciones, mejoras visuales, mecanismos de seguridad, almacenamiento de resultados y registro de telemetría. La planificación detallada de roles, backlog, Sprints y criterios de aceptación se presenta posteriormente en el capítulo metodológico.

### *Marco conceptual*

El marco conceptual delimita el significado operativo de los términos empleados en la investigación. Las definiciones se presentan de forma breve para evitar repetir la fundamentación desarrollada en el marco teórico.

**Caso de prueba.** Prompt oficial definido antes de la evaluación y asignado a uno de los tres niveles de complejidad.

**Ejecución.** Intento individual de procesar un caso de prueba. Un caso puede producir varias ejecuciones cuando se repite la solicitud.

**Traza.** Registro cronológico de las llamadas, tiempos, tokens, estados y errores que componen una ejecución en Langfuse.

**Pipeline multiagente.** Secuencia coordinada de interpretación, generación, validación, conversión y registro utilizada por Bezy.

**Ingeniería de prompts.** Diseño de instrucciones externas e internas que orientan la interpretación del usuario y el formato de los resultados.

**SVG.** Representación vectorial textual utilizada como salida intermedia, previsualizable y verificable.

**DXF.** Formato de intercambio utilizado para abrir la geometría resultante en herramientas CAD/CAM compatibles.

**Validación geométrica.** Conjunto de comprobaciones deterministas aplicadas a rutas, continuidad, límites y estructura del archivo.

**Éxito técnico.** Finalización de una ejecución sin excepción y con una respuesta procesable. No equivale necesariamente a fidelidad visual o funcional.

**Regeneración interna.** Nueva llamada al agente generador dentro de una misma traza, realizada antes de finalizar la ejecución.

**Latencia.** Tiempo transcurrido desde el inicio del procesamiento hasta la obtención de la respuesta registrada.

**Tokens.** Unidades de procesamiento textual contabilizadas en las entradas y salidas de los modelos de lenguaje.

# Capítulo III

## Metodología

### *Enfoque de investigación*

La presente investigación se desarrolló mediante un enfoque mixto con predominio cuantitativo, debido a que combinó la medición objetiva del desempeño técnico de Bezy con la interpretación cualitativa de los archivos vectoriales generados. De acuerdo con la guía metodológica institucional, el enfoque mixto integra información cuantitativa y cualitativa para analizar de manera integral el objeto de estudio [4]. La combinación de ambos componentes permitió abordar la investigación mediante datos numéricos verificables y criterios descriptivos orientados a comprender las características y limitaciones de los resultados producidos por el sistema.

El componente cuantitativo comprendió las variables relacionadas con el funcionamiento del pipeline multiagente, entre ellas la latencia, el consumo de tokens, el tiempo de procesamiento de cada agente, el número de ejecuciones, los errores y las regeneraciones internas. Además, se consideraron las propiedades técnicas de los archivos SVG y DXF, como la validez, el tamaño, el número de trazados, el cierre de trazados, la composición geométrica y la compatibilidad de exportación. Estos factores permitieron clasificar y comparar el sistema según su complejidad para las pruebas.

La parte cualitativa consistió en evaluar la fidelidad del resultado del proceso a las instrucciones dadas en lenguaje natural. Se analizó si se conservaban los elementos solicitados, si el diseño era claro y reconocible, si se simplificaban o eliminaban detalles, si era apto para corte láser y si requería revisión humana. En esta etapa, pudimos diferenciar entre el éxito del proceso y la fidelidad del resultado a las instrucciones dadas.

Combinación de datos cualitativos y cuantitativos. Con el prototipo, pudimos probar su funcionamiento, su capacidad para cumplir con los requisitos técnicos y su adecuación a las restricciones semánticas, visuales y funcionales que deseábamos generar para los archivos.

No solo estábamos probando si el sistema podía generar los archivos, sino también si los archivos generados eran de buena calidad.

### *Tipo y diseño de investigación*

Se trata de un estudio descriptivo. Su objetivo es describir el funcionamiento del sistema Bezy y del flujo de trabajo multiagente en la producción de archivos vectoriales. Se estudia el comportamiento de las instrucciones en lenguaje natural al traducirlas a una representación vectorial, validarlas y prepararlas para su exportación en formatos SVG y DXF.

Por lo tanto, me resulta útil delimitar el sistema para tener una lista de pasos bien definida que describa lo que estoy probando y registrando (latencia, consumo de tokens, tiempo por agente, ejecuciones adicionales, regeneraciones internas, errores, etc.). Luego, reviso los archivos para comprobar su validez, sus dimensiones, su ubicación, si cierran la ruta, su composición, etc.

Esto significa que podemos observar cómo se comporta el prototipo ante solicitudes de diferentes formas y tamaños. Podemos identificar tendencias, diferencias y dónde falla el sistema, pero no asumimos relaciones de causa y efecto ni generalizamos los resultados fuera del entorno de prueba.

### *Requisitos del sistema*

Esto es lo que ofrecemos y los atributos de calidad que esperamos de Bezy. Los requisitos funcionales describen el comportamiento que esperamos de la interfaz, el backend y el microservicio de IA, mientras que los requisitos no funcionales describen las condiciones relacionadas con el rendimiento, la arquitectura, la compatibilidad, la usabilidad y la escalabilidad. Esta especificación proviene del documento de control de versiones técnicas del proyecto y se adaptó a los componentes del flujo de trabajo de generación de vectores.

**Tabla 8**

*Requisitos funcionales del sistema Bezy*

| Código | Módulo | Requisito funcional |
|---|---|---|
| RF-01 | Interfaz de usuario | El sistema debe proporcionar un campo de texto donde el usuario pueda ingresar, en lenguaje natural, la descripción del diseño vectorial que desea generar. |
| RF-02 | Interfaz de usuario | El sistema debe permitir seleccionar el espesor del material maderable y establecer las dimensiones máximas del diseño, expresadas mediante ancho y alto en milímetros. |
| RF-03 | Interfaz de usuario | El sistema debe renderizar una vista previa del diseño generado mediante un visor SVG antes de habilitar la descarga de los archivos. |
| RF-04 | Interfaz de usuario | El sistema debe permitir descargar el diseño validado en formato SVG para visualización o edición y en formato DXF para su utilización en software CAD/CAM. |
| RF-05 | Backend y orquestación | El backend debe recibir el prompt y los parámetros enviados desde el frontend y coordinar la comunicación asíncrona con el microservicio de inteligencia artificial desarrollado en Python. |
| RF-06 | Backend y base de datos | El sistema debe almacenar el identificador del usuario, el prompt original y la ubicación del archivo generado para conservar un historial de diseños y permitir consultas posteriores. |
| RF-07 | Inteligencia Artificial | El agente generador debe traducir la instrucción |

| | | |
|---|---|---|
| | | en lenguaje natural a código vectorial estructurado, respetando las dimensiones y proporciones solicitadas. |
| RF-08 | Inteligencia artificial | En el modo geométrico, el agente generador debe agrupar las líneas de corte mediante <g id="corte"> y trazo negro #000000, y las líneas de grabado mediante <g id="grabado"> y trazo rojo #FF0000, de modo que se conserven como capas CUT y ENGRAVE en el archivo DXF. |
| RF-09 | Validación geométrica | El validador debe analizar matemáticamente el código generado y comprobar que las trayectorias destinadas al corte sean rutas cerradas, de modo que el punto final coincida con el punto inicial dentro de la tolerancia definida. |
| RF-10 | Validación geométrica | Cuando se detecte una geometría abierta o inválida, el sistema debe solicitar una corrección automática al agente generador hasta alcanzar el número máximo de reintentos configurado antes de devolver un error al usuario. |
| RF-11 | Geometría de ensambles | El sistema debe procurar que las ranuras, dientes de unión u otros elementos de encastre correspondan al espesor del material indicado por el usuario. |

Los requisitos funcionales representan el recorrido principal de una solicitud dentro de Bezy: ingreso de la descripción, captura de parámetros físicos, procesamiento por el backend,

generación vectorial, validación geométrica, almacenamiento y descarga. Su organización por módulos permite relacionar cada comportamiento esperado con el componente responsable de implementarlo y facilita la elaboración posterior de casos de prueba.

**Tabla 9**

*Requisitos no funcionales del sistema Bezy*

| Código | Atributo | Requisito no funcional |
|---|---|---|
| RNF-01 | Rendimiento | El proceso de generación y validación vectorial no debe exceder un tiempo de espera de 45 segundos. Durante la ejecución, la interfaz debe mostrar indicadores de estado que informen al usuario sobre las etapas de interpretación y validación. |
| RNF-02 | Arquitectura | El procesamiento geométrico y la integración con la API del modelo de lenguaje deben ejecutarse en un servicio Python separado del servidor principal Node.js, evitando bloquear el ciclo de eventos del backend web. |
| RNF-03 | Compatibilidad | Los archivos DXF generados deben conservar la escala y utilizar entidades compatibles con versiones estándar del formato para permitir su importación en aplicaciones CAD/CAM como LightBurn o RDWorks. |
| RNF-04 | Usabilidad | La interfaz debe poder utilizarse sin conocimientos previos de software CAD y presentar un flujo de trabajo claro compuesto por tres acciones principales: ingresar la descripción, previsualizar el resultado y descargar los archivos. |
| RNF-05 | Escalabilidad | El sistema debe utilizar una API REST que permita que el motor de inteligencia artificial pueda ser consumido posteriormente por otras interfaces sin modificar el núcleo de generación y |

| Código | Atributo | Requisito no funcional |
|---|---|---|
| | | validación geométrica. |

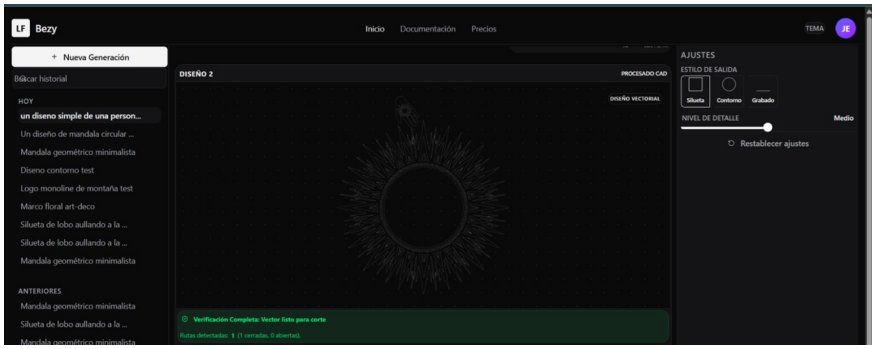
Estos requisitos constituyen una especificación de referencia para el desarrollo y la evaluación del prototipo. Su inclusión no implica que todos hayan alcanzado el mismo nivel de implementación. La diferenciación automática entre líneas de corte y grabado se implementó mediante grupos, colores de trazo y capas DXF; en cambio, la adaptación de los elementos de encastre al espesor del material se mantiene como una capacidad susceptible de perfeccionamiento y debe interpretarse junto con las limitaciones y recomendaciones presentadas en los capítulos posteriores.

***Interfaz web del prototipo***

La Figura 6 presenta la interfaz web implementada para la interacción con el sistema Bezy. La pantalla integra el historial de solicitudes, la previsualización del diseño y los controles de ajuste y descarga, de modo que el usuario pueda completar el flujo principal sin utilizar directamente herramientas CAD.

**Figura 6**

*Interfaz web de Bezy durante la visualización y validación de un diseño vectorial*




En el panel izquierdo se encuentra el historial de generaciones; el área central presenta el lienzo y la retroalimentación de la verificación geométrica; y el panel derecho reúne los ajustes del resultado. Esta organización corresponde a la implementación del prototipo y se

presenta en el capítulo metodológico porque describe el producto y su forma de interacción, mientras que su desempeño se analiza posteriormente mediante los resultados de las pruebas.

### *Metodología de desarrollo Scrum*

Desarrollamos Bezy utilizando Scrum desde el primer día. Dado que nuestra solución constaba de varios componentes interdependientes (interfaz web, backend, persistencia, agentes de IA, validación geométrica, conversión de archivos y observabilidad), nos pareció lógico desarrollarla en Sprints para obtener versiones que pudiéramos verificar, revisar el progreso y tomar decisiones de cambio sin tener que esperar a que todo estuviera terminado.

Se adaptó para un proyecto académico realizado por una sola persona. Por lo tanto, no se realizaban reuniones diarias de seguimiento. En su lugar, cada miembro del equipo supervisaba su progreso diariamente revisando tareas, archivos, cambios de código e incidencias abiertas. La revisión externa se realizaba en reuniones semanales con el asesor, generalmente los martes, donde revisábamos el progreso del sistema y la documentación, hacíamos recomendaciones y reajustábamos las prioridades. Esta adaptación mantuvo la esencia de Scrum (trabajo iterativo, backlog priorizado, incrementos, inspección y adaptación) sin las ceremonias ni los registros que no estaban documentados formalmente.

La planificación presentada en esta sección se consolidó a partir de tres fuentes: las actividades y la arquitectura aprobadas en el Anexo 2 antes del inicio del desarrollo, los requisitos funcionales y no funcionales definidos para Bezy, y los historiales documentados de los repositorios frontend y backend. Estos registros permiten vincular las iteraciones con los cambios reales (sin modificar las fechas de confirmación ni crear evidencia artificial) y muestran la división del trabajo en la Tabla 10.

#### Determinación de responsabilidades

Este es un desarrollo individual; el Ing. Puente Ponce Pablo Francisco Mgtr fue Product Owner y Scrum Master. Como Product Owner, definió las prioridades y el valor del incremento; como Scrum Master, facilitó el proceso, hizo sugerencias y eliminó obstáculos metodológicos.

**Tabla 10**

*Determinación de responsabilidades Scrum del proyecto Bezy*

| Responsabilidad | Integrante | Funciones dentro del proyecto |
|---|---|---|
| Product Owner | Msc. Pablo Francisco Puente Ponce | Orientar los requerimientos, establecer prioridades, revisar los incrementos y verificar su relación con los objetivos del trabajo de integración curricular. |
| Scrum Master | Msc. Pablo Francisco Puente Ponce | Facilitar el seguimiento semanal, formular recomendaciones, identificar impedimentos metodológicos y promover ajustes en el producto y el documento. |
| Developer | Jefferson Stalyn Chafuelan Gramal | Analizar, diseñar, implementar y probar frontend, backend, agentes de IA, validador, conversión SVG–DXF, persistencia, telemetría y documentación. |

*Nota.* Las responsabilidades de Product Owner y Scrum Master fueron asumidas por el tutor debido a la naturaleza académica e individual del proyecto.

**Eventos y seguimiento del trabajo**

Los eventos se ajustaron a la disponibilidad del equipo y al acompañamiento académico. La planificación se realizó al inicio de cada periodo; el seguimiento individual permitió controlar las actividades durante la iteración; las reuniones semanales funcionaron como espacios de inspección del incremento; y al finalizar cada Sprint se identificaron problemas, decisiones y mejoras para la siguiente etapa. La Tabla 11 resume esta adaptación.

**Tabla 11**

*Eventos Scrum adaptados al desarrollo de Bezy*

| Evento | Periodicidad aplicada | Aplicación en Bezy |
|---|---|---|
| Sprint Planning | Inicio de cada Sprint | Selección de elementos del Product Backlog, definición del objetivo, estimación de horas y establecimiento del entregable. |
| Seguimiento individual | Durante el Sprint | Revisión personal de tareas, cambios de código, archivos generados, errores e impedimentos pendientes. |
| Revisión de avances | Semanal, generalmente los martes | Presentación del incremento al tutor, revisión del proyecto y del documento, y recepción de recomendaciones. |
| Sprint Review | Cierre de la iteración | Verificación del entregable y de los criterios de aceptación antes de incorporar nuevas actividades. |
| Retrospectiva adaptada | Cierre de la iteración | Identificación individual de dificultades, decisiones técnicas y acciones de mejora para el siguiente Sprint. |

*Nota.* No se realizó una Daily Scrum formal, debido a que el desarrollo fue ejecutado por un único Developer.

### Planificación del Product Backlog

El Product Backlog se construyó a partir de los componentes previstos en el perfil aprobado y se complementó con las necesidades identificadas durante el desarrollo. Los elementos se priorizaron considerando dependencias técnicas: primero se establecieron requisitos y arquitectura; posteriormente se implementaron frontend, backend y agentes; luego se incorporaron validación, persistencia y experiencia de usuario; y finalmente se añadieron telemetría, seguridad, pruebas y documentación.

Las estimaciones se expresan en horas de trabajo y utilizan un promedio documental de seis horas por día hábil. La suma planificada es de 504 horas, distribuidas entre 84 días hábiles. Las horas representan esfuerzo aproximado de análisis, implementación, pruebas y documentación, y no un registro automatizado de tiempo.

**Tabla 12**

*Product Backlog consolidado del sistema Bezy*

| ID | Relación | Elemento del Product Backlog | Horas | Prioridad | Sprint |
|---|---|---|---|---|---|
| PB-01 | OE-01 | Revisar literatura sobre LLM, SVG, geometría computacional, agentes y fabricación digital. | 24 | alta | 1 |
| PB-02 | E-01 | Consolidar requisitos, alcance y arquitectura de agentes definidos en el Anexo 2. | 24 | alta | 1 |
| PB-03 | RNF-02 | Preparar entorno, estructura de trabajo, cronograma y planificación Scrum. | 12 | alta | 1 |
| PB-04 | RF-01-04 | Configurar el frontend React y TypeScript con una estructura modular inicial. | 16 | alta | 2 |
| PB-05 | RF-05 | Crear el proyecto en Node.js y Express junto con las rutas y controllers iniciales de autenticación. | 18 | alta | 2 |
| PB-06 | RF-06 | Integración de la base de datos con Supabase y la implementación de su verificación nativa. | 14 | alta | 2 |
| PB-07 | RNF-04 | Empezar con la parte de la interfaz de usuario con los componentes de historial, conversación y configuración. | 12 | media | 2 |
| PB-08 | RF-05 | Integrar el frontend y el backend mediante los services y endpoints de conversación y login. | 12 | alta | 3 |
| PB-09 | RF-07 | Implementar el agente intérprete con el modelo de llama 8b . | 18 | alta | 3 |
| PB-10 | RF-07 | Implementar el agente de IA con el modelo de DeepSeek V4 Pro para la generación. | 18 | alta | 3 |
| PB-11 | RF-02, RF-07 | Definir parámetros geométricos y prompts internos como reglas | 12 | alta | 3 |

| ID | Relación | Elemento del Product Backlog | Horas | Prioridad | Sprint |
|---|---|---|---|---|---|
| | | estrictas para la posterior generación. | | | |
| PB-12 | RF-03, RF-07 | Adoptar SVG como representación vectorial pre visualizable con ayuda de la biblioteca de Potrace en Python. | 12 | alta | 4 |
| PB-13 | RF-09-10 | Implementar controles de validación geométrica y control de rutas cerradas para que ayuden al resultado final en la etapa técnica. | 16 | alta | 4 |
| PB-14 | RF-04 | Implementar la funcionalidad de conversión de SVG a DXF con ayuda de los agentes. | 12 | alta | 4 |
| PB-15 | RF-06 | Incorporar funcionalidades adicionales como el historial de los diseños generados, conversaciones y rutas de descarga de los archivos en ambos formatos. & | 14 | alta | 4 |
| PB-16 | RF-01, RNF-04 | Transformar el componente de la sección de conversación para que funcione como un chat. | 14 | media | 5 |
| PB-17 | RF-02 | Incorporar métricas dinámicas en campos como las dimensiones del objeto generado. | 14 | alta | 5 |
| PB-18 | RF-07, RF-09 | Implementar el modelo encargado netamente a la validación y corrección de figuras geométricas básicas. | 16 | alta | 5 |
| PB-19 | RF-05, RNF-03 | Ejecutar pruebas de endpoints con métodos de POST y GET para poner a prueba específicamente la generación de archivos. | 10 | alta | 5 |
| PB-20 | RNF-04 | Mejorar progresivamente la UI con | 24 | media | 6 |

| ID | Relación | Elemento del Product Backlog | Horas | Prioridad | Sprint |
|---|---|---|---|---|---|
| | | componentes de la biblioteca de Shadcn en el frontend. | | | |
| PB-21 | RNF-04 | Aplicar mejoras de diseño responsivo, navegación y retroalimentación visual. | 18 | media | 6 |
| PB-22 | RNF-01 | Diseñar el esquema de métricas que se tomarán en cuenta para la evaluación de los resultados. | 12 | alta | 6 |
| PB-23 | RNF-01 | Registrar tiempos, métricas como líneas cerradas, dimensiones, número de reintentos dentro de la base de datos, de preferencia con un nueva tabla relacional. | 14 | alta | 7 |
| PB-24 | RNF-01 | tegrar Langfuse para ayudar al tema de registrar el tiempo de cada agente, mensajes en caso de errores y un control más automatizado para la evaluación de los resultados. | 16 | alta | 7 |
| PB-25 | RF-07 | Consolidar DeepInfra y la configuración del modelo de DeepSeek para los agentes del pipeline. | 16 | alta | 7 |
| PB-26 | AE-01 | Diseñar la matriz que contará con los 90 casos de prueba separados por su nivel de complejidad. | 8 | alta | 7 |
| PB-27 | RNF-05 | Incorporar JWT, rutas protegidas y manejo de expiración de sesión para asegurar la integridad y seguridad de cada cuenta. | 18 | alta | 8 |
| PB-28 | RF-06 | Asegurar el almacenamiento de los archivos de forma remota (supabase storage). | 12 | alta | 8 |

| ID | Relación | Elemento del Product Backlog | Horas | Prioridad | Sprint |
|---|---|---|---|---|---|
| PB-29 | RF-10 | Fortalecer el manejo de errores, respuestas incompletas y fallos en caso de la generación de un modelo. | 14 | alta | 8 |
| PB-30 | AE-01 | Preparar los casos oficiales a evaluar según la población e instrumento definido. | 10 | alta | 8 |
| PB-31 | AE-01 | Ejecutar las 90 pruebas oficiales y llevar un registro de las métricas de cada archivo generado, asi como los casos de errores o reintentos. | 20 | alta | 9 |
| PB-32 | AE-01 | Consolidar y analizar las métricas registradas tanto en Supabase como en Langfuse para el análisis cualitativo. | 14 | alta | 9 |
| PB-33 | RNF-04 | Incorporar ajustes finales en tema de lógica dentro de la página, rutas, elementos que funcionen con el estado de Zustand y mensajes al usuario. | 8 | media | 9 |
| PB-34 | OE-01-02, AE-01 | AE-01 & Consolidar e ir redactando los resultados y análisis dentro del documento, así como los archivos de registro de los casos de prueba realizados. | 12 | alta | 9 |

**Distribución de los Sprints**

El periodo planificado se extendió desde el 6 de abril hasta el 30 de julio de 2026. Se identificaron 84 días hábiles, equivalentes a 504 horas con el promedio de seis horas diarias. Para mantener iteraciones cercanas a nueve o diez días hábiles, el trabajo se distribuyó en nueve Sprints. Los tres primeros contienen 60 horas y los seis restantes 54 horas.

**Tabla 13**

*Distribución temporal de los Sprints del proyecto Bezy*

| Sprint | Inicio | Fin | Días hábiles | Horas | Backlog asignado |
|---|---|---|---|---|---|
| 1 | 06/04/2026 | 17/04/2026 | 10 | 60 | PB-01 a PB-03 |
| 2 | 20/04/2026 | 01/05/2026 | 10 | 60 | PB-04 a PB-07 |
| 3 | 04/05/2026 | 15/05/2026 | 10 | 60 | PB-08 a PB-11 |
| 4 | 16/05/2026 | 18/05/2026 | 9 | 54 | PB-12 a PB-15 |
| 5 | 29/05/2026 | 10/06/2026 | 9 | 54 | PB-16 a PB-19 |
| 6 | 11/06/2026 | 23/06/2026 | 9 | 54 | PB-20 a PB-22 |
| 7 | 24/06/2026 | 06/07/2026 | 9 | 54 | PB-23 a PB-26 |
| 8 | 07/07/2026 | 17/07/2026 | 9 | 54 | PB-27 a PB-30 |
| 9 | 18/07/2026 | 30/07/2026 | 9 | 54 | PB-31 a PB-34 |

*Nota.* La duración se calculó con días hábiles y un promedio documental de seis horas de trabajo diario.

### Planificación detallada de los Sprints

**Sprint 1: fundamentación, requisitos y arquitectura.** El primer Sprint estableció la base conceptual y técnica del proyecto. Debido a que el perfil había sido aprobado antes del inicio, se partió de la arquitectura de agentes, los formatos SVG y DXF y la necesidad de validación geométrica definidos en el Anexo 2. La iteración consolidó estos elementos como requisitos y actividades verificables.

**Tabla 14**

*Planificación del Sprint 1*

| PB | Actividades planificadas | Horas | Entregable |
|---|---|---|---|
| PB-01 | Revisión bibliográfica de LLM, SVG, agentes, validación y fabricación digital. | 24 | Base bibliográfica y estado del arte. |
| PB-02 | Consolidación de requisitos, alcance, componentes y flujo multiagente. | 24 | Especificación inicial y diagramas de arquitectura. |
| PB-03 | Organización del cronograma, entorno y planificación iterativa. | 12 | Estructura de trabajo y planificación Scrum. |

**Sprint 2: estructura inicial frontend, backend y persistencia.** El objetivo fue disponer de las bases ejecutables de la aplicación web, incluyendo autenticación, conexión con Supabase y una primera organización modular de la interfaz.

**Tabla 15**

*Planificación del Sprint 2*

| PB | Actividades planificadas | Horas | Entregable |
|---|---|---|---|
| PB-04 | Crear el proyecto React/TypeScript, layout y componentes iniciales. | 16 | Frontend ejecutable y modular. |
| PB-05 | Crear API Express, configuración TypeScript y rutas de autenticación. | 18 | Backend inicial operativo. |
| PB-06 | Configurar Supabase Auth, conexión y variables de entorno. | 14 | Registro e inicio de sesión conectados. |
| PB-07 | Construir historial, área de trabajo y panel de configuración inicial. | 12 | Primera interfaz navegable. |

**Sprint 3: integración de inteligencia artificial.** En el tercer sprint se hizo la conexión con el modelo de python para usar los agentes en base al pipeline definido y se establecieron los modelos de IA a usar, como principal tenemos el de DeepSeek V4 Pro con algunas reglas en formato de prompts para estructurar la salida y definir el modo que se debe usar.

**Tabla 16**

*Planificación del Sprint 3*

| PB | Actividades planificadas | Horas | Entregable |
|---|---|---|---|
| PB-08 | Crear y consumir el endpoint de generación vectorial. | 12 | Comunicación entre frontend, backend y servicio de IA. |
| PB-09 | Implementar interpretación estructurada con DeepSeek V4 Pro. | 18 | Parámetros técnicos extraídos del prompt. |
| PB-10 | Implementar la generación vectorial con DeepSeek V4 Pro. | 18 | Primera salida generada desde lenguaje natural. |
| PB-11 | Definir esquemas, restricciones geométricas y prompts internos. | 12 | Contratos de datos y reglas de generación. |

**Sprint 4: SVG, validación, DXF e historial.** Con el cuarto sprint se definió un flujo técnico pasando desde la idea inicial que es tomada como un prompt, hasta la generación de los archivos en los distintos formatos. Con el uso del formato SVG se pudo aplicar validaciones y reglas antes de que se genere el archivo DXF.

**Tabla 17**

*Planificación del Sprint 4*

| PB | Actividades planificadas | Horas | Entregable |
|---|---|---|---|
| PB-12 | Configurar SVG como salida directa, editable y previsualizable. | 12 | SVG renderizado en la interfaz. |
| PB-13 | Implementar cierre de rutas y reporte de validación. | 16 | Validador geométrico funcional. |
| PB-14 | Convertir entidades SVG válidas a DXF. | 12 | Archivo DXF descargable. |
| PB-15 | Almacenar archivos y organizar conversaciones e historial. | 14 | Persistencia y recuperación de resultados. |

**Sprint 5: interacción conversacional y refinamiento geométrico.** Esta iteración se orientó a mejorar el flujo de uso y a corregir comportamientos observados en geometrías curvas y parámetros dimensionales.

**Tabla 18**

*Planificación del Sprint 5*

| PB | Actividades planificadas | Horas | Entregable |
|---|---|---|---|
| PB-16 | Reemplazar el área estática por conversaciones organizadas en hilos. | 14 | Interfaz de chat con historial. |
| PB-17 | Incorporar dimensiones, detalle y parámetros del lienzo. | 14 | Solicitudes configurables. |
| PB-18 | Corregir generación y conversión de círculos, arcos y elipses. | 16 | Mayor compatibilidad geométrica. |
| PB-19 | Verificar endpoints, integración y apertura de archivos. | 10 | Flujo integrado sometido a pruebas. |

**Sprint 6: experiencia de usuario y preparación de métricas.** El Sprint 6 buscó mejorar la presentación del producto mediante componentes reutilizables y preparar los datos que posteriormente permitirían evaluar el rendimiento del pipeline.

**Tabla 19**

*Planificación del Sprint 6*

| PB | Actividades planificadas | Horas | Entregable |
|---|---|---|---|
| PB-20 | Integrar Shadcn UI y reorganizar los componentes visuales. | 24 | Interfaz moderna y modular. |
| PB-21 | Mejorar respuesta móvil, navegación, alertas y estados de carga. | 18 | Experiencia de usuario refinada. |
| PB-22 | Definir métricas, campos y estructura de evaluación. | 12 | Esquema de datos para las pruebas. |

**Sprint 7: observabilidad y consolidación del proveedor.** La séptima iteración incorporó la instrumentación necesaria para observar el comportamiento interno y consolidó la operación de DeepSeek V4 Pro mediante DeepInfra para los roles de interpretación y generación.

**Tabla 20**

*Planificación del Sprint 7*

| PB | Actividades planificadas | Horas | Entregable |
|---|---|---|---|
| PB-23 | Medir tiempos y registrar métricas en backend y Supabase. | 14 | Registros cuantitativos del proceso. |
| PB-24 | Instrumentar agentes y ejecuciones mediante Langfuse. | 16 | Trazas con tiempos, tokens y errores. |
| PB-25 | Consolidar DeepInfra y la configuración de DeepSeek V4 Pro. | 16 | Agentes configurados mediante el proveedor seleccionado. |
| PB-26 | Definir los 90 prompts y su clasificación por complejidad. | 8 | Matriz oficial de casos de prueba. |

**Sprint 8: seguridad, robustez y preparación de la evaluación.** El objetivo fue fortalecer el acceso a los recursos, controlar fallos del servicio de inteligencia artificial y preparar los instrumentos y condiciones de las pruebas oficiales.

**Tabla 21**

*Planificación del Sprint 8*

| PB | Actividades planificadas | Horas | Entregable |
|---|---|---|---|
| PB-27 | Implementar JWT, rutas protegidas y expiración de sesión. | 18 | Acceso autenticado y controlado. |
| PB-28 | Asegurar archivos e historial mediante almacenamiento remoto. | 12 | Recursos persistentes y recuperables. |
| PB-29 | Controlar errores del servicio, respuestas incompletas y reintentos. | 14 | Manejo de fallos y mensajes estructurados. |
| PB-30 | Preparar instrumentos, casos y procedimiento de recolección. | 10 | Entorno de evaluación reproducible. |

**Sprint 9: pruebas, análisis y consolidación final.** La última iteración se planificó para ejecutar los casos oficiales, consolidar la telemetría, aplicar ajustes finales y completar la documentación técnica y académica.

**Tabla 22**

*Planificación del Sprint 9*

| PB | Actividades planificadas | Horas | Entregable |
|---|---|---|---|
| PB-31 | Ejecutar 30 casos por cada nivel y conservar archivos y trazas. | 20 | Evidencia de las 90 pruebas oficiales. |
| PB-32 | Depurar, consolidar e interpretar métricas y resultados. | 14 | Tablas, gráficos y análisis estadístico. |
| PB-33 | Incorporar carga, retroalimentación y correcciones finales. | 8 | Interfaz final estabilizada. |
| PB-34 | Finalizar documento UIC, manuales y organización del proyecto. | 12 | Producto y documentación consolidados. |

**Definición de terminado**

Para considerar que un elemento del Sprint Backlog formaba parte del incremento, se estableció una definición de terminado adaptada al proyecto. Esta condición permitió evitar que una tarea se considerará completada únicamente por haber sido programada, sin integración, prueba o evidencia.

**Tabla 23**

*Definición de terminado aplicada a los incrementos de Bezy*

| Condición | Descripción aplicada |
|---|---|
| Implementación | La funcionalidad o componente se encuentra incorporado en la arquitectura correspondiente. |
| Integración | El cambio puede comunicarse con los componentes de los que depende sin impedir el flujo principal. |
| Verificación | Se ejecutó una comprobación tanto técnica como visual de los distintos casos de prueba realizados. |
| Manejo de errores | En base a los errores que se pudieron observar (generación de diseños), estos no afectan a los demás componentes dentro de la aplicación web. |
| Trazabilidad | Se obtuvieron distintos datos en base a las distintas métricas y también documentación tanto de las pruebas como del código. |
| Revisión | El avance continuo durante cada semana tuvo la capacidad de editar o ajustar distintos escenarios. |
| Documentación | Los resultados más relevantes fueron demostrados en |

| Condición | Descripción aplicada |
|---|---|
| | gráficas y tablas, por otra parte, el código y su lógica cuenta con archivos README. |

**Criterios de aceptación de los Sprints**

Los criterios de aceptación establecen las condiciones mínimas para considerar que el objetivo de una iteración fue atendido. No representan una certificación externa ni sustituyen las pruebas detalladas del sistema. Su finalidad es relacionar cada incremento con una comprobación observable y con los requisitos correspondientes.

**Tabla 24**

*Criterios de aceptación de los nueve Sprints de Bezy*

| Sprint | ID | Criterio | Verificación | Tipo |
|---|---|---|---|---|
| 1 | CA-01 | Los objetivos, actividades y alcance deben encontrarse delimitados. | Revisión del Anexo 2 y requisitos consolidados. | Metodológico |
| 1 | CA-02 | La arquitectura estará conformada por el frontend, backend, agentes y formatos o estructura de archivos dentro del proyecto. | Revisión de diagramas y especificación técnica. | Sistema |
| 1 | CA-03 | El trabajo debe estar separado en elementos en base a su nivel de prioridad. | Product Backlog y cronograma definidos. | Gestión |
| 2 | CA-04 | El frontend y el backend deben poder ser ejecutado en sus entornos de desarrollo, considerando la forma en que fueron creados. | Inicio local sin errores bloqueantes. | Sistema |
| 2 | CA-05 | El usuario podrá iniciar sesión, así como registrarse dentro de la aplicación web. | Prueba de autenticación conectada con | Funcional |

| Sprint | ID | Criterio | Verificación | Tipo |
|---|---|---|---|---|
| | | | Supabase. | |
| 2 | CA-06 | La interfaz que carga al inicio debe mostrar las áreas principales, dependiendo de los componentes que hayan sido implementados. | Inspección de layout, historial y configuración. | Usabilidad |
| 3 | CA-07 | El frontend debe poder comunicarse mediante algún service con el endpoint de generación en la parte del backen. | Petición y respuesta HTTP estructurada. | Funcional |
| 3 | CA-08 | Deepseek V4 Pro debe ser capaz de generar parámetros que puedan ser interpretados, como medidas, modo, etc. | Inspección del objeto estructurado. | Funcional |
| 3 | CA-09 | El agente generador debe poder producir una representación gráfica del diseño solicitado. | Visualización o lectura de la respuesta generada. | Funcional |
| 4 | CA-10 | El sistema por la parte del frontend deberá renderizar y mostrar en un div el SVG que haya recibido. | Vista previa visible en el navegador. | Funcional |
| 4 | CA-11 | El validador debe ser capaz de registrar errores geométricos encontrados durante su ejecución dentro del pipeline. | Reporte técnico asociado al resultado. | Funcional |
| 4 | CA-12 | El SVG que se haya generado y no tenga errores, podrá ser convertido a DXF y tener la función de descargarlo. | Archivo DXF disponible y procesable. | Compatibilidad |
| 4 | CA-13 | Los resultados deben guardarse y poder ser recuperados para consultas o para mostrar dentro de la aplicación. | Consulta de conversación y rutas de archivo. | Funcional |
| 5 | CA-14 | Los mensajes y resultados tendrán | Navegación entre | Usabilidad |

| Sprint | ID | Criterio | Verificación | Tipo |
|---|---|---|---|---|
| | | una estructura de conversación, estructurada a partir de la base de datos. | hilos almacenados. | |
| 5 | CA-15 | Las dimensiones o el modo deben ser enviados como parámetros en la respuesta del JSON. | Inspección de la petición y respuesta. | Funcional |
| 5 | CA-16 | Las formas curvas o figuras simples deben ser consideradas y validadas para que sean compatibles con el formato DXF. | Pruebas con círculos, arcos o elipses. | Compatibilidad |
| 6 | CA-17 | La interfaz del frontend con React debe utilizar componentes consistentes y que puedan ser reutilizables. | Inspección del sistema de componentes. | Mantenibilidad |
| 6 | CA-18 | La navegación y cambio de tamaño dentro del sitio debe acoplarse y ser responsivo. | Pruebas visuales de respuesta móvil. | Usabilidad |
| 6 | CA-19 | Los campos necesarios para evaluar el pipeline deben quedar bien definidos. | Revisión del esquema de métricas. | Investigación |
| 7 | CA-20 | Se debe poder visualizar el tiempo y estado de cada generación, ya sea nativamente o con herramientas de terceros. | Consulta de métricas en backend o Supabase. | Rendimiento |
| 7 | CA-21 | Se debe configurar Langfuse para que pueda mostrar trazas del pipeline en cada generación individual. | Inspección de ejecuciones instrumentadas. | Observabilidad |
| 7 | CA-22 | DeepSeek V4 Pro será usado para el agente generador principalmente, aunque puede | Ejecución completa con el proveedor | Funcional |

| Sprint | ID | Criterio | Verificación | Tipo |
|---|---|---|---|---|
| | | estar abierto a una sustitución del modelo. | configurado. | |
| 7 | CA-23 | Los 90 prompts deben estar definidos antes de su análisis y ejecución, así como su categorización. | Matriz con 30 casos por nivel. | Investigación |
| 8 | CA-24 | Las rutas que estén protegidas deben validar el token de acceso, así como el generado por Supabase. | Pruebas de acceso válido, expirado y ausente. | Seguridad |
| 8 | CA-25 | Los archivos y conversaciones deben tener un límite para no tener una sobrecarga en cada llamada o petición. | Consulta autenticada de historial y almacenamiento. | Seguridad |
| 8 | CA-26 | Los fallos en la generación del servicio de IA tendrán mensajes o retroalimentación para el cliente. | Simulación o registro de respuestas de error. | Fiabilidad |
| 8 | CA-27 | El procedimiento que se haga para la evaluación de los casos de prueba deben estar documentados. | Revisión de instrumentos y casos oficiales. | Investigación |
| 9 | CA-28 | Deben ejecutarse 90 casos oficiales, cada uno previamente formulado y colocado en su respectivo nivel de complejidad. | Registros de casos, archivos y trazas. | Investigación |
| 9 | CA-29 | Las métricas deben estar definidas en base a lo que se requiera analizar y sea necesario documentar. | Depuración y matriz final de análisis. | Investigación |
| 9 | CA-30 | Los resultados obtenidos deben estar basados en el cumplimiento de los objetivos. | Tablas, gráficos y discusión del Capítulo IV. | Metodológico |
| 9 | CA-31 | La versión final debe incluir la | Revisión del | Entregable |

| Sprint | ID | Criterio | Verificación | Tipo |
|---|---|---|---|---|
| | | documentación respectiva, así como las recomendaciones que se crean relevantes. | sistema y documento UIC. | |

Como se ha visto, la planificación basada en Scrum puede proporcionar una estructura para la comprensión del producto en su desarrollo, sin embargo, no sustituye a la metodología de investigación. Los sprints se encargan de llevar un control técnico de cada avance, mientras que la observación sistemática, tablas, telemetría y análisis cualitativo nos sirven para estudiar el comportamiento del sistema y sus respuestas. Esta distinción evita confundir el marco de gestión del desarrollo con el enfoque mixto y el alcance descriptivo de la investigación.

***Población y muestra***

La unidad de análisis estuvo constituida por cada caso de prueba oficial definido para evaluar Bezy, junto con las ejecuciones registradas y el archivo final asociado. Conforme a la guía metodológica institucional, este apartado debe delimitar con claridad qué elementos conforman la población, el procedimiento de selección y el alcance de los resultados [4]. En términos metodológicos, la población comprende la totalidad de unidades que comparten las características establecidas para el estudio y sobre las cuales se realiza el análisis [16].

La población de la investigación fue finita y estuvo conformada por el corpus completo de 90 prompts oficiales: 30 casos de complejidad simple, 30 de complejidad intermedia y 30 de complejidad compleja. Los casos fueron definidos antes de iniciar las pruebas y se diseñaron para representar las categorías geométricas y funcionales contempladas dentro del alcance del prototipo. Estos casos no forman parte de la población de muestra, ya que se trata de ajustes, no de una evaluación formal.

La muestra fue un censo (dado que la población era pequeña y estaba bien definida), por lo que coincidió con la población total y no se utilizó una fórmula de selección probabilística.

La muestra incluyó los 90 casos y se conservó un archivo SVG final para la evaluación técnica y cualitativa. Las 121 ejecuciones en Langfuse no constituyen una población diferente ni una ampliación de la muestra, sino los intentos resultantes del procesamiento de los mismos 90 casos cuando hubo repeticiones o regeneraciones. La cobertura del censo nos permite describir el comportamiento del corpus definido y comparar los tres niveles de complejidad sin omitir ningún caso. Sin embargo, los resultados corresponden a las notificaciones, modelos, configuración y arquitectura utilizados en Bezy, por lo que la interpretación se limita a estas condiciones y no pretende generalizar automáticamente el rendimiento a ninguna solicitud, modelo o proveedor.

***Técnica e instrumento***

Para el componente cuantitativo se empleó la técnica de observación sistemática, regulada o controlada, debido a que el comportamiento de Bezy fue examinado mediante casos de prueba ejecutados bajo condiciones previamente definidas. La técnica permitió registrar de manera objetiva tanto el desempeño del pipeline multiagente como las propiedades técnicas de los archivos finales, sin depender de apreciaciones visuales para determinar el estado de una ejecución, su consumo de recursos o el cumplimiento de las reglas geométricas implementadas.

Como instrumento cuantitativo se utilizó una **ficha digital integrada de evaluación técnica**. El instrumento reunió en una misma estructura de análisis dos dimensiones complementarias. La primera forma en que se obtuvo los datos necesarios, fue la telemetría del pipeline mediante la herramienta externa de Langfuse, la segunda consistió en las métricas netamente de los archivos generados, los cuales se fueron almacenando en la tabla metricas_evaluacion de la base de datos en Supabase. Cada herramienta describió distintas fuentes necesarias para la evaluación de los casos de prueba, Langfuse proporcionó datos como el tiempo de ejecución de cada modelo, casos de errores, y los parámetros dentro de las

respuestas JSON, mientras que Supabase, mediante funciones dentro del proyecto proporcionaba propiedades técnicas evaluando cada caso individual.

Por la parte de la dimensión de la telemetría: trace_id, execution_status, latencia, tokens y tiempo por agente, son algunas de las medidas recogidas dentro de Langfuse. Por otra parte, complementando las medidas necesarias, tenemos la cantidad de reintentos, rutas cerradas, curvas de Bézier, aplicación de búfer y las dimensiones. Estos datos que se obtienen en base a la estructura de la generación y finalización de archivos, se aplicó a los mencionados 90 casos de pruebas, divididos en 30 de un nivel denominado como simple, 30 intermedios y 30 complejos. Debido a las repeticiones, la dimensión de telemetría contenía 121 ejecuciones únicas, pero la dimensión técnica mantuvo un archivo final para cada uno de los 90 casos. Los registros exploratorios realizados durante las etapas iniciales del proyecto fueron excluidos antes del procesamiento para evitar que pruebas ajenas al corpus formal modificarán los indicadores.

**Tabla 25**

*Determinación de roles del proyecto*

| Dimensión | Unidad | Variables principales | Fuente y finalidad |
|---|---|---|---|
| Telemetría del pipeline | Ejecución o traza | Estado, latencia, tokens, tiempo por agente, modo, errores y regeneraciones internas. | Exportación de Langfuse para analizar rendimiento, estabilidad, consumo y comportamiento de las ejecuciones. |
| Evaluación técnica del archivo | Caso y archivo final | Validez, reintentos, rutas, trazos cerrados y abiertos, curvas Bézier, agujeros, piezas, *buffer*, dimensión y errores geométricos. | Exportación de `metricas_evaluacion` de Supabase para verificar la estructura y las propiedades técnicas del SVG final. |
| Identificación y relación | Caso de prueba | ID del caso, prompt, nivel, ID de traza, ID de registro, fecha y nombre del archivo. | Corpus oficial y registros del sistema para relacionar las 121 ejecuciones con los 90 resultados finales. |

Antes del análisis se verificaron identificadores, fechas, estados, latencias y consumos para evitar la contabilización duplicada de una misma traza. Las ejecuciones adicionales se asociaron con su caso original y se diferenció entre éxito por ejecución, éxito inicial y estado técnico final. Posteriormente, las variables fueron procesadas mediante estadística descriptiva, utilizando frecuencias, porcentajes, promedios, medianas, desviaciones estándar, percentiles, valores mínimos, valores máximos y correlaciones cuando correspondía.

Para el componente cualitativo se empleó la técnica de análisis de contenido mediante revisión visual estructurada. Esta técnica permitió interpretar los artefactos generados y valorar si cada SVG representaba la intención expresada en el prompt, mantenía claridad visual y presentaba una composición pertinente para su revisión previa a un proceso de fabricación digital. La evaluación cualitativa se aplicó a la totalidad de los 90 archivos finales, por lo que cada caso cuantitativo contó también con una valoración interpretativa.

Como instrumento se utilizó una **ficha digital de análisis cualitativo de archivos generados**. Cómo se creó una tabla dedicada a la obtención de los datos de cada generación, se tiene un identificador del caso o conversación, otras columnas como el nivel de complejidad, el prompt oficial, el nombre generado automáticamente para los archivos y demás métricas de apoyo con criterios para su posterior interpretación visual. Esta valoración cualitativa, se centró más en la comparación entre el prompt ingresado en el sistema y la representación visible final, tanto en el SVG como el DXF.

La ficha se fue organizando en un documento donde se tomaron en cuenta la interpretación de los datos registrados en Supabase, así como el análisis manual de la representación final, entre las categorías se tienen definidas como *adecuado*, *parcialmente adecuado* y *no adecuado*. Para que un resultado esté dentro de la categoría adecuado, debe conservar la solicitud principal y arrojar una representación comprensible y acorde a lo solicitado; parcialmente adecuado, el que representó la idea general, pero mostró señales de elementos adicionales o diferencias de escala o en su composición; y no adecuado al que no

fue capaz de representar con suficiente fidelidad la idea o petición del usuario, dando como resultado figuras incomprensibles o terminando en un error de generación por parte del modelo. Además, se incorporó un campo abierto para justificar la valoración y registrar la necesidad de revisión humana.

**Tabla 26**

*Criterios de la ficha de análisis cualitativo*

| Criterio | Descripción | Registro |
|---|---|---|
| Estado técnico automatizado | Incorpora como evidencia de apoyo la validez, rutas cerradas, errores y demás métricas registradas en Supabase. | Adecuado / parcialmente adecuado / no adecuado |
| Correspondencia prompt–resultado | Determina si el SVG representa la idea, forma o conjunto funcional solicitado. | Adecuado / parcialmente adecuado / no adecuado |
| Claridad visual | Valora si los elementos y piezas pueden reconocerse e interpretarse sin ambigüedad significativa. | Adecuado / parcialmente adecuado / no adecuado |
| Interpretación dimensional | Contrasta las dimensiones explícitas del prompt con la dimensión real registrada cuando este criterio resulta aplicable. | Adecuado / parcialmente adecuado / no adecuado / no aplica |
| Pertinencia para fabricación | Examina si la organización visible de las piezas y trazos permite considerar el archivo para una revisión previa a fabricación. | Adecuado / parcialmente adecuado / no adecuado |
| Valoración global | Integra la correspondencia, claridad, parámetros y pertinencia técnica observada en el caso. | Adecuado / parcialmente adecuado / no adecuado |
| Revisión humana | Registra si el archivo requiere comprobación adicional antes de considerarse satisfactorio. | Sí / preventiva |
| Observación cualitativa | Justifica la valoración mediante una descripción de aciertos, omisiones, elementos adicionales o problemas de composición. | Texto abierto |

El procedimiento cualitativo inició con la relación de cada prompt oficial con su archivo SVG final y su registro en Supabase. Posteriormente, los archivos fueron renderizados e inspeccionados visualmente, considerando conjuntamente la representación obtenida y las métricas técnicas disponibles. Los casos que tenían una representación algo ambigua tambien

pasaron por una revisión desde su estructura del tipo de archivo SVG. Para finalizar, se hizo una valoración para cada criterio y una observación de cada caso que se realizó.

***Validación y confiabilidad del instrumento***

Con el instrumento del análisis cuantitativo se evaluó cada caso de prueba en base a las variables y los objetivos que se evaluaron dentro del sistema. La separación que se hizo entre obtención de datos del pipeline y características de los diseños, sirvió para tener una mejor interpretación de cada ejecución, y como un plus, automatizar un proceso que hubiese sido innecesario realizar desde cero.

La confiabilidad de los datos no se apoya simplemente en datos que puedan considerarse como estáticos, sino que cada uno proviene de la generación individual, en donde funciones capturan estos valores en el proceso de generación de una petición.

En el componente cualitativo, se evaluó con un proceso prácticamente similar, aunque con criterios distintos, en donde entraron campos como la observación de los 90 casos, especialmente los archivos finales. Cada valoración se hacía en base al prompt inicial, y como campos relevantes se tenía el nombre del SVG, el registro de Supabase y una justificación que se realizaba textualmente. Los casos que se consideraban como ambiguos tenían que ser revisados nuevamente para comprobar que la estructura del archivo no cayera en un riesgo de clasificación errónea basada simplemente en una impresión inicial.

Como se ha mencionado, el análisis cualitativo se basó principalmente en una interpretación visual, y no debe tomarse como una medición automatizada o evaluación de satisfacción de usuarios, todas fueron evaluadas bajo los mismos criterios sin opiniones externas. Por esto, los resultados obtenidos por este tipo de metodología tiene el objetivo de complementar las métricas técnicas y explicar un poco menos técnico la validez geométrica, claridad visual y correspondencia con la petición.

### *Consideraciones éticas*

Para esta investigación, se tomó en cuenta la parte ética con respecto al uso responsable de la Inteligencia Artificial, el manejo de los datos y la integridad de la evaluación de un prototipo. Tomando en cuenta que Bezy es un sistema centralmente pensado para trabajar con un conjunto de agentes, separados en distintas tareas como la interpretación, generación y validación, este estudio se diseñó de forma que las capacidades de los modelos no sean tomadas como valores absolutos ni resultados infalibles, recordando que los resultados obtenidos son netamente probabilísticos. Por el contrario, la investigación reconoce estas limitaciones y cómo una interpretación de una petición ambigua e interpretación de instrucciones es un proceso que no elimina la necesidad de revisión humana, al menos antes de querer utilizar los archivos en un proceso de fabricación digital.

Al recopilar distintos datos, las pruebas se realizaron con el fin de obtener y demostrar la capacidad de la IA en este trabajo académico. Por lo que se puede establecer que la información sensible como datos personales o privados de cualquier usuario no serán usados con fines maliciosos y se tratará de salvaguardar su información. Los registros y toda información importante para el funcionamiento del sistema se almacena en Supabase, con el único propósito de contabilizar los registros, obtener información relevante para tener estadísticas y organizar los resultados de la evaluación técnica. Así pues, la investigación se limita a recopilar principalmente la información de los diseños y generación de archivos.

Teniendo en cuenta que todo gira alrededor de la IA y sus capacidades, es importante tener claro el papel de los modelos empleados. En el proyecto, los agentes de IA se usaron para el funcionamiento del sistema, no como una sustitución de la revisión humana en la parte técnica. Por este motivo, los resultados obtenidos de los casos evaluados se rigieron por validaciones geométricas y una revisión cualitativa con criterios predefinidos. Esta decisión se toma por el tema de la responsabilidad tecnológica, ya que un archivo que parezca correcto a simple vista no puede ser considerado estrictamente adecuado para corte láser, donde se

toman en cuenta características más referentes al diseño CAD. Como cualquier proyecto web, se toma muy en serio el tema de la seguridad, tanto del usuario como frente a posibles ataques por parte de terceros. El corte láser depende mucho del tipo o modelo de la máquina, y se debe tomar en cuenta que el tipo de archivo, así como los valores de parámetros del láser, pueden dañar el material si no se toman en cuenta los componentes anteriores. Por lo tanto, el sistema evalúa desde un punto de vista preventivo, priorizando la generación de diseños planos, sin poner por delante el diseño o su aspecto, ya que la generación debe constar de trayectorias cerradas y archivos compatibles. Aun así, la validación final no debe descartarse por completo y debe contar con una revisión humana. Antes de fabricar una pieza, el usuario debe verificar las dimensiones, el material, la potencia, la velocidad, la escala y la compatibilidad con el software de control láser correspondiente.

En cuanto a la propiedad intelectual y el uso del contenido generado, los casos de prueba se redactan como instrucciones genéricas, y no como una solicitud para copiar una marca registrada, un logotipo, un personaje o un diseño de terceros. Esto me permite centrarme en las capacidades del sistema, y no en copiar el trabajo de otros. Los archivos generados durante las pruebas del prototipo se utilizan únicamente con fines académicos.

Por último, los resultados se presentaron de forma objetiva para evitar influir en los lectores con mis opiniones. Las métricas reportadas se obtuvieron directamente de los registros técnicos del backend, el microservicio Python y la base de datos, mientras que los datos cualitativos se recopilaron según las siguientes reglas. Por lo tanto, los resultados no se manipularon para mostrar que el sistema es el más rápido, el mejor, el más rápido y el mejor, etc. En este prototipo se presentan las fortalezas y debilidades de Bezy para describirlo como una IA. Si bien es posible crear archivos vectoriales con IA, se necesitan mecanismos de control, validación y monitoreo para un uso responsable en la fabricación digital.

# Capítulo IV

## Análisis y discusión de los resultados

El presente capítulo expone los resultados obtenidos durante el desarrollo y la evaluación del sistema Bezy. En primer lugar, se describe la evolución del producto mediante los incrementos funcionales alcanzados en los nueve Sprints definidos en el Capítulo III. Posteriormente, se presentan los resultados cuantitativos y cualitativos de las pruebas oficiales, con el propósito de diferenciar el avance técnico del producto de la evaluación técnica de su comportamiento.

La presentación de los Sprints se concentra en los productos efectivamente integrados y en las evidencias que permiten comprobar cada incremento. Por esta razón, no se repiten las actividades planificadas ni las estimaciones del Sprint Backlog, sino que se explica qué componentes quedaron disponibles, cómo se relacionaron con el flujo general y qué base proporcionaron para las iteraciones posteriores.

## Resultados de los incrementos funcionales de Bezy

La ventaja de aplicar un desarrollo iterativo, es que una propuesta arquitectónica puede transformarse en una aplicación web capaz de recibir instrucciones en lenguaje natural, procesarlas mediante agentes potenciados por la IA, generar archivos vectoriales, controlar y validar la geometría de un diseño y obtener un formato especializado, así como la automatización de la trazabilidad de procesos específicos. Cada sprint puede evidenciar y documentar un incremento verificable, de tal forma que las funcionalidades planificadas y desarrolladas en una iteración fueran integradas y mejoradas o corregidas en los pasos siguientes.

### *Resultados del Sprint 1*

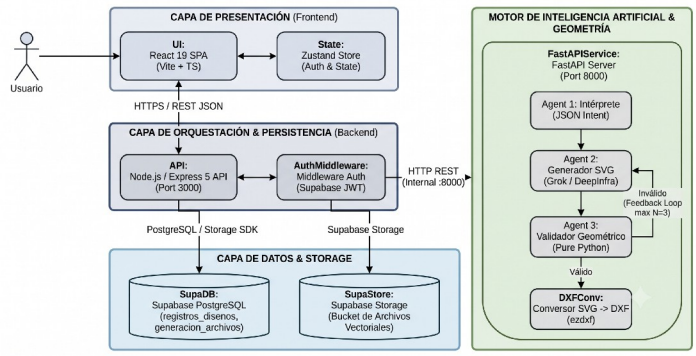
El sprint 1 tuvo el objetivo de sentar las bases y requisitos para el proyectos, entre las delimitaciones se tenía el definir el alcance y la arquitectura inicial. Como resultado, se delimitaron las responsabilidades tanto del frontend como el backend. El modelo de IA

encargado de la generación de los archivos también tuvo responsabilidades independientes, así como Supabase, que se encargó de definir la base de datos con una estructura relacional. La separación de cada funcionalidad permite establecer desde el principio la respuesta y responsabilidad del modelo, el cual se lo define como una herramienta para simplificar la tarea manual de crear diseños, sin dejar de lado el hecho de que sus resultados no deben ser usados directamente para fabricación.

También se organizaron los requisitos funcionales y no funcionales, el Product Backlog y la distribución temporal del trabajo. La arquitectura resultante definió una capa de presentación desarrollada con React y TypeScript, una capa de orquestación mediante Node.js y Express, una capa de datos apoyada en Supabase y un servicio especializado en FastAPI para coordinar los agentes y el procesamiento geométrico. La Figura 7 presenta la relación establecida entre estos componentes.

**Figura 7**

*Arquitectura consolidada del sistema Bezy en el Sprint 1*



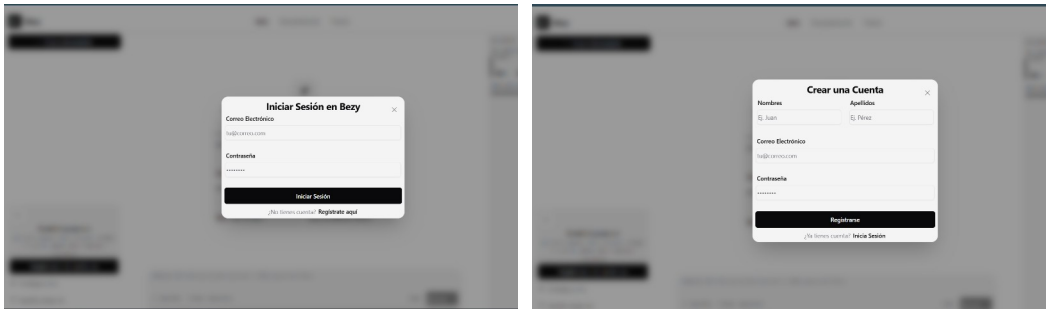

El incremento obtenido proporcionó una base técnica común para las siguientes iteraciones y permitió relacionar cada requisito con un componente específico. En consecuencia, el Sprint concluyó con el alcance delimitado, la arquitectura documentada y el trabajo distribuido en elementos verificables, cumpliendo las condiciones definidas para iniciar la implementación del sistema.

### *Resultados del Sprint 2*

En el sprint 2 el desarrollo se centró en construir las bases ejecutables de la aplicación. Con esto se obtuvo proyectos independientes de frontend y backend, cada uno con características de una arquitectura modular, donde se manejaron por separado componentes como services, rutas, componentes de la UI y controladores. En esta fase, la interfaz y la API se crearon simultáneamente, se evitó acoplar la capa de presentación y la lógica del sistema, por otra parte, Supabase facilitó la implementación de autenticación y persistencia de archivos. Los usuarios pueden crear una cuenta y confirmar su registro; luego, pueden iniciar sesión en el sistema y acceder al espacio de trabajo. El resultado se muestra en la Figura 8. Gracias a la autenticación, las conversaciones y los archivos generados se pueden vincular a un usuario específico, en lugar de mantener el resultado en el navegador solo durante la sesión.

**Figura 8**

*Interfaces de inicio de sesión y registro implementadas en el Sprint 2*




La conexión con Supabase también permitió definir las primeras relaciones entre usuarios, conversaciones, mensajes, diseños y registros técnicos. La estructura de datos se fue ampliando en iteraciones posteriores, pero desde este Sprint quedó disponible la base necesaria para autenticar usuarios y persistir información. El resultado fue una primera versión navegable con frontend, backend y autenticación conectados, sobre la cual se incorporó posteriormente el servicio de inteligencia artificial.

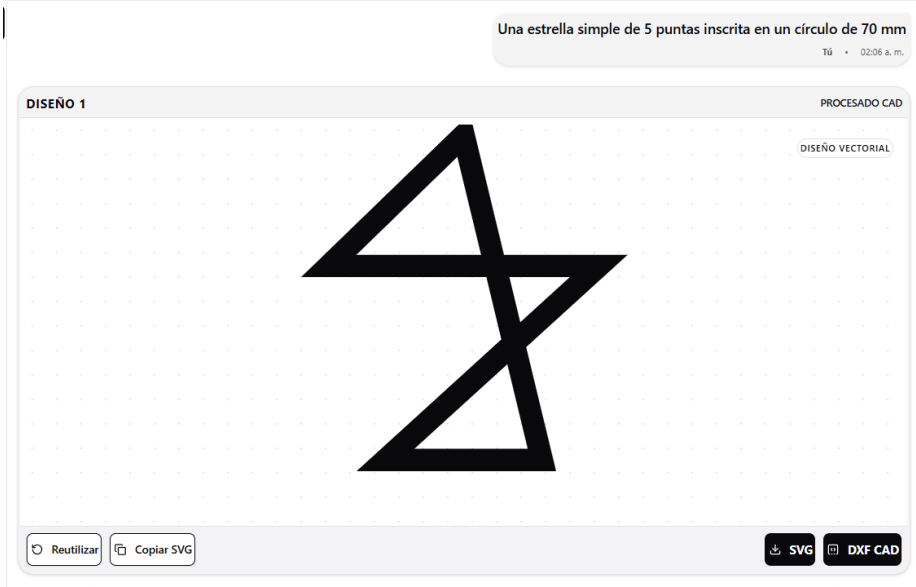
***Resultados del Sprint 3***

El Sprint 3 incorporó la inteligencia artificial al flujo de la aplicación. Se implementó la comunicación entre la interfaz web, la API de orquestación y el servicio especializado, permitiendo enviar un prompt desde el frontend y recibir una representación vectorial procesable. El agente intérprete se encargó de transformar la instrucción del usuario en parámetros estructurados, mientras que el agente generador utilizó dicha interpretación y las reglas internas para formular las instrucciones con las que el servicio Python produjo una primera salida en formato SVG.

Como mejora, se definió los esquemas de entrada y salida, así como restricciones o limitaciones en el contenido generado y también en los mensajes internos para evitar que se muestren respuestas no deseadas o con estructura de un error interno de ejecución. Se pudo evidenciar que en la integración inicial el sistema es capaz de procesar instrucciones en lenguaje natural y generar una representación visual en el navegador, específicamente con el formato SVG construido por el modelo. Aun así, las primeras salidas mostraron que una respuesta procesable no quiere decir que sea capaz de generar con precisión el formato o petición solicitada. La Figura 9 muestra uno de los primeros resultados obtenidos durante esta etapa.

**Figura 9**

*Primera salida SVG procesable obtenida durante la integración de los agentes*




Este resultado permitió verificar la comunicación entre los componentes y, al mismo tiempo, identificar la necesidad de incorporar validación geométrica, reglas más estrictas y mecanismos de refinamiento. Por tanto, el principal aporte del Sprint fue disponer de un pipeline funcional de extremo a extremo, aun cuando su salida todavía requería controles adicionales antes de considerarse adecuada para el objetivo del proyecto.

***Resultados del Sprint 4***

El sprint 4 estuvo definido para transformar el prototipo de generación en un pipeline integrado. El SVG que era construido por el agente comenzó a ser mostrado en el frontend y también se empezó a definir mediante el validador geométrico. Este modelo verificó la estructura del archivo, el cierre de trayectorias y presencia de errores que pudieran impedir el procesamiento posterior. Con la validación, se pudo separar la apariencia visual del cumplimiento de las condiciones geométricas mínimas definidas por el sistema y que son cruciales para la etapa de procesamiento.

Una vez que se podía obtener el archivo SVG, validando el diseño monolinea, se procedió a agregar la etapa de conversión al formato DXF, así como la descarga en ambos formatos, esto desde la interfaz del frontend. En este proceso, el módulo de conversión se encarga de recuperar la estructura del código SVG e identifica los grupos de corte y grabado, todo usando las librerías de ezdxf en el lenguaje de python. Para esto, fue necesario definir estrictamente por colores los trazos para CUT y ENGRAVE, una configuración distintiva debido al uso de la herramienta CAD usada (RDWorks V8). La Figura 10 muestra el archivo convertido, mientras que la Figura 11 muestra que este archivo se puede guardar en almacenamiento remoto para su posterior recuperación, en lugar de depender de una respuesta temporal generada durante la conversión.

**Figura 10**

*Descarga del archivo DXF generado por Bezy*

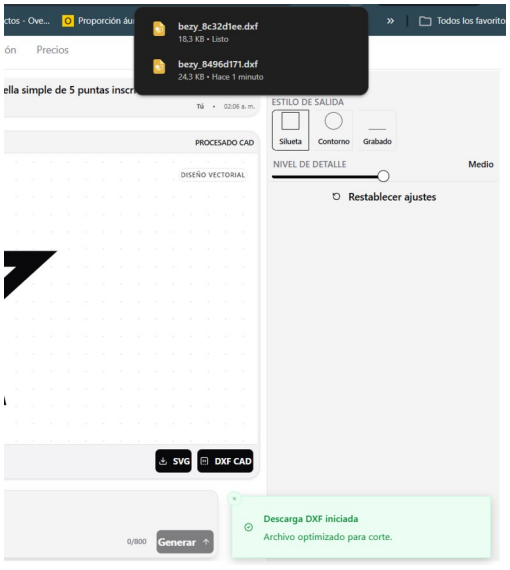
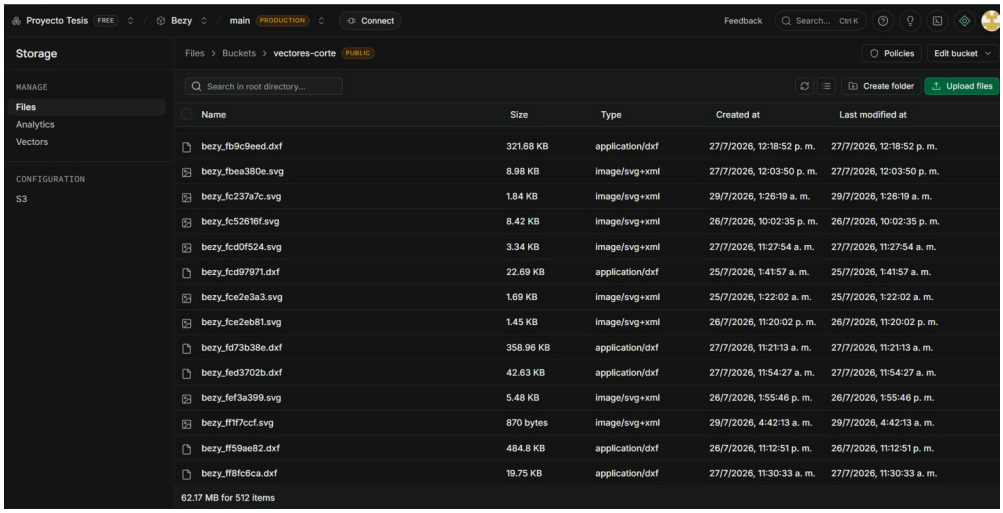



**Figura 11**

*Archivos SVG y DXF almacenados en Supabase Storage*




El último incremento de Sprint permitió informar al usuario sobre un archivo de datos y obtener un archivo descargable y persistente. La integración del historial y el almacenamiento permite a los usuarios establecer una relación entre ellos, la conversación y el resultado. Esta fue la primera versión técnicamente funcional del sistema y la base para mejorar la interacción conversacional y el manejo de geometrías específicas.

***Resultados del Sprint 5***

Con el Sprint 5 Se logró que la interfaz sea más interactiva y capaz de procesar mejor los parámetros geométricos. La interfaz ya no funciona como un formulario aislado, sino que organiza las solicitudes en una conversación. Cada hilo conserva la solicitud, así como la ruta del archivo en la base de datos, con esto se puede crear la vista previa del SVG en cada chat por parte del usuario e implementar un historial funcional basado en un ID de conversación, teniendo en cuenta la fecha del mismo.

También se ha añadido la funcionalidad de enviar como un parámetro las dimensiones, el nivel de detalle o el modo de generación. Estas variables no afectan realmente a la experiencia directa entre el usuario y la página, sino que sirven como una forma de tener claro los parámetros de un diseño, con el objetivo de tener claro las variables y ser capaz de identificar las variaciones entre las mismas. La Figura 12 muestra una conversación completa con su resultado, y la Figura 13 presenta la comprobación de un diseño generado con

parámetros dimensionales y visualizado posteriormente en el entorno de procesamiento vectorial.

**Figura 12**

*Conversación con prompt y resultado vectorial generado por Bezy*

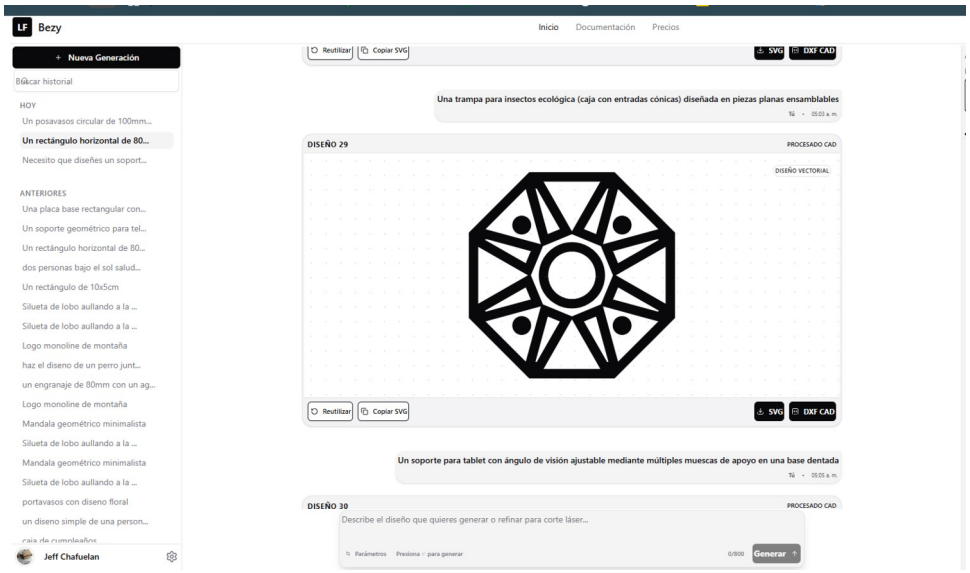
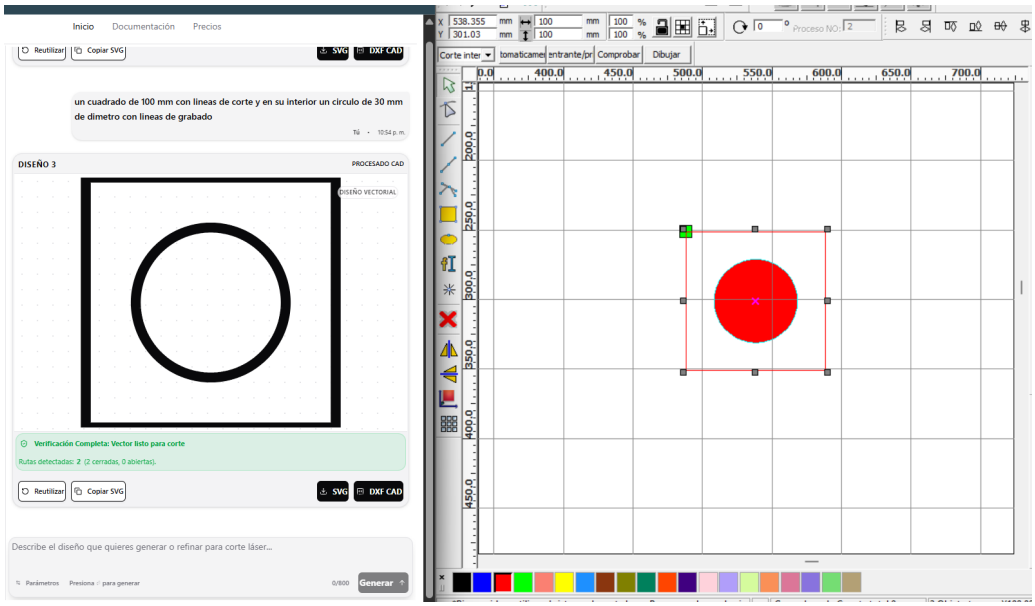



**Figura 13**

*Comprobación visual de un diseño generado con parámetros dimensionales*



Durante la iteración se aplicaron ajustes para mejorar el tratamiento de círculos, arcos, elipses y otras geometrías curvas durante la generación y conversión. Asimismo, se verificó la comunicación entre los endpoints involucrados y la apertura de los archivos finales. El resultado fue un flujo conversacional más comprensible y una mayor compatibilidad entre la solicitud, la vista previa y los formatos descargables.

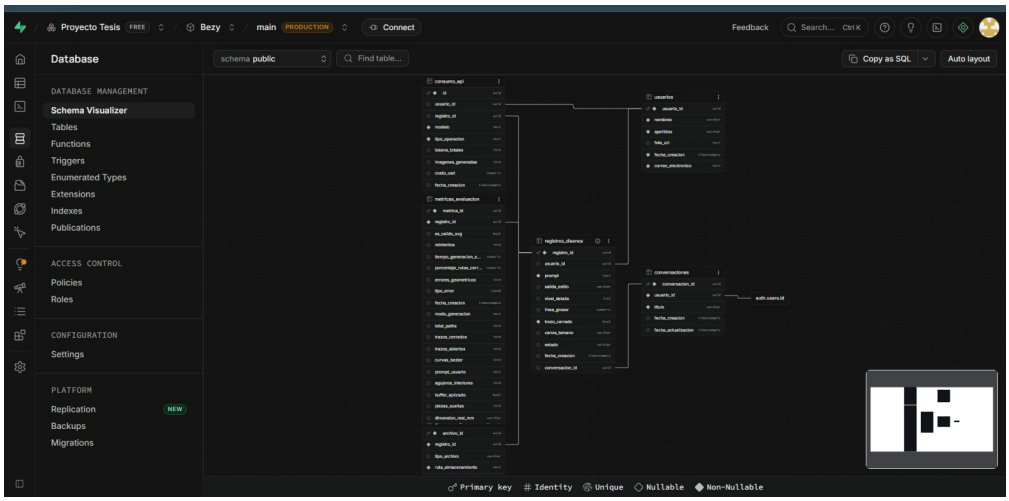
***Resultados del Sprint 6***

Durante el Sprint 6, se centró en mejorar la experiencia del usuario y el marco de evaluación. Reorganizamos los componentes visuales y se consideraron como reutilizables, se estandarizó los controles, los mensajes, las alertas y los estados de carga, y se ajustó la navegación para mantener una presentación coherente en las distintas áreas del sistema. Esto hizo que la lista de conversaciones, el área principal de generación, la vista previa y las acciones de descarga fueran más claras.

Simultáneamente, definí los campos necesarios para registrar el comportamiento del pipeline y las características técnicas de los archivos. Luego, extendí la estructura de Supabase para relacionar usuarios, conversaciones, mensajes, diseños, archivos y métricas. La Figura 14 muestra el modelo de relación que utilicé para organizar la persistencia y preparar la recopilación de datos para las siguientes pruebas.

**Figura 14**

*Modelo de datos de Supabase utilizado por el sistema Bezy*




El resultado del Sprint no fue solo un cambio estético. Definir componentes consistentes simplificó el mantenimiento de la interfaz, mientras que el esquema de métricas permitió comprender qué información conservar durante una generación. Así, la aplicación quedó preparada no solo para mostrar el estado del proceso al usuario, sino también para registrar evidencia técnica para la evaluación del sistema.

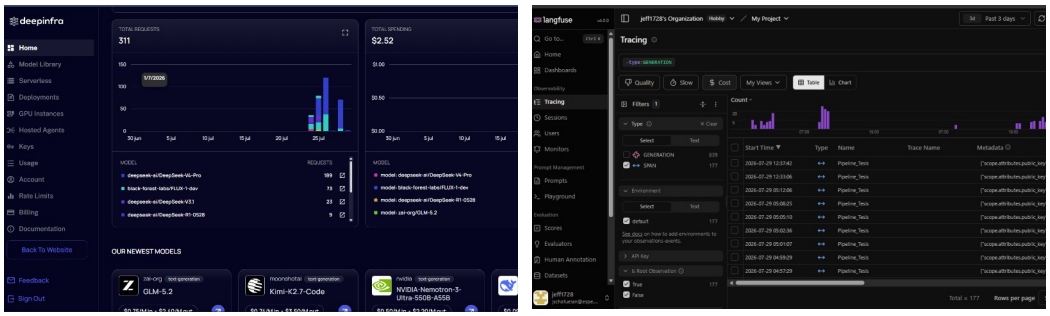
***Resultados del Sprint 7***

El sprint 7 se centró en implementar la observabilidad al pipeline específicamente, por otra parte se reemplazó el proveedor utilizado por el agente generador, netamente con la finalidad de probar un modelo más barato y probablemente más eficiente teniendo en cuenta la generación de código SVG y diseños en 2D. Este proceso de trazabilidad se hizo con la herramienta de Langfuse, permitiendo registrar medidas como estado, tiempo de ejecución, estructura de la petición JSON entre los modelos, y todo individualmente para cada generación, observando el pipeline completo. Esta información se usó principalmente para determinar el comportamiento de cada intento y distinguir el tiempo empleado por el intérprete, generador y validación, información sumamente relevante para el análisis de los casos de prueba.

Asimismo, los roles de interpretación y generación se consolidaron mediante DeepInfra con DeepSeek V4 Pro. La configuración mantuvo la separación de responsabilidades del pipeline mediante prompts especializados y permitió administrar el modelo sin modificar la interfaz utilizada por el usuario. Las Figuras 15 muestran los paneles empleados para observar el proveedor y las trazas generadas durante las ejecuciones.

**Figura 15**

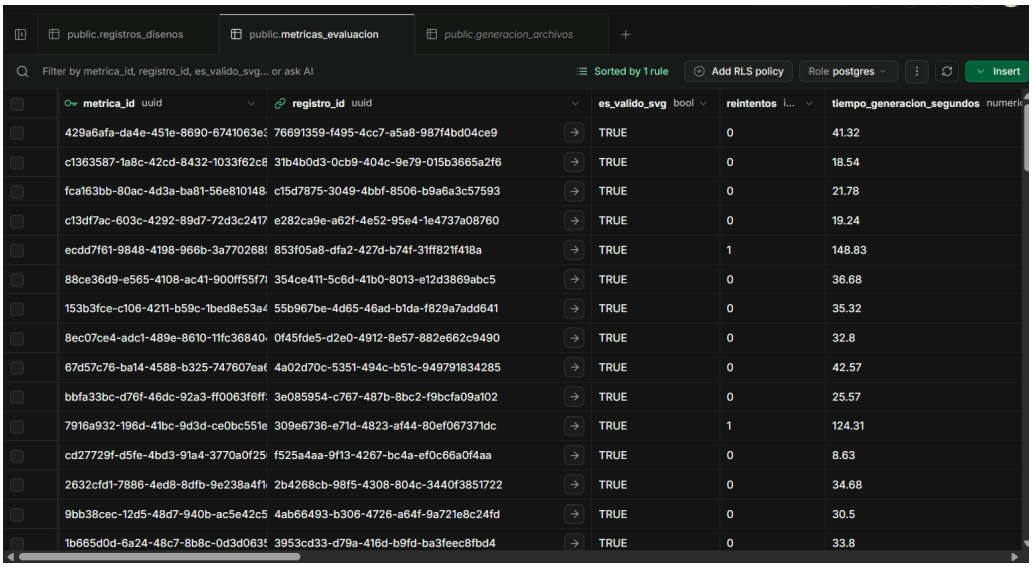
*Paneles de DeepInfra y Langfuse utilizados durante la observabilidad del pipeline*




Además de la telemetría, se definió el corpus oficial de 90 prompts, distribuido en 30 casos por cada nivel de complejidad. Los registros técnicos comenzaron a conservarse en Supabase para relacionar cada caso con su resultado, archivo y métricas geométricas, como se observa en la Figura 16. El incremento obtenido convirtió el pipeline en un proceso observable y proporcionó las fuentes de datos necesarias para medir posteriormente su desempeño.

**Figura 16**

*Registros técnicos conservados en la tabla de evaluación de Supabase*




***Resultados del Sprint 8***

Este Sprint 8 se centra en asegurar lo que hemos construido antes de que comiencen las pruebas reales. Esto incluye asegurar las rutas que requieren autenticación y agregar validación de tokens de acceso para evitar el acceso no autorizado a conversaciones, archivos y resultados. También gestionamos las sesiones perdidas o caducadas para evitar que las solicitudes no válidas lleguen a los servicios internos del sistema.

El flujo de trabajo de IA se ha mejorado para gestionar respuestas incompletas, errores de comunicación y resultados no procesables, de modo que los fallos se conviertan en respuestas controladas para evitar interrupciones inesperadas de la interfaz y permitir que el usuario sepa lo que está sucediendo.

La persistencia remota de conversaciones y archivos almacenados como registros en sus respectivas tablas complementa las medidas de mantener los recursos con el usuario autenticado, esto quiere decir que cada usuario podrá visualizar solo sus conversaciones asociadas, así como los archivos generados en las mismas.

Para terminar, se definieron y prepararon los instrumentos, los casos oficiales que iban a ser evaluados y el procedimiento de recolección. Se tuvo que llevar un registro lineal de cada ejecución, tanto con la trazabilidad de los agentes con ayuda de Langfuse, así como las

métricas almacenadas en la tabla de la base de datos. Para cada tipo de análisis se tuvo un registro técnico y un archivo en el que se evidenciara el proceso concluido. Como resultado, el sistema pudo reproducir bajo condiciones específicas cada ejecución, con acceso controlado, manejo u observación de fallos y peticiones previamente definidas.

***Resultados del Sprint 9***

El Sprint 9 se centró principalmente en la realización y documentación de las pruebas oficiales. Se procesaron detalladamente los 90 prompts, distribuidos uniformemente entre las tres categorías o niveles de complejidad. Considerando que los resultados están basados en procesos probabilísticos, se obtuvo algunos escenarios en donde se tuvo que hacer reintentos debido a fallos en la generación, dando como resultado una suma de 121 casos, contando reintentos, aunque dejando claro que estos no se lo toma como la suma total de los casos de prueba. Por esto, se estableció un usuario sin conversaciones anteriores, para asegurarse de no cometer errores en la contabilización de los 90 casos, que fueron definidos y establecidos para su evaluación.

Por una parte, la información arrojada de Langfuse determinó el rendimiento de los agentes, y por la parte de Supabase, se definieron los datos con relación a la generación estricta de los modelos en base a su archivo SVG. Esta integración permitió calcular tasas de éxito, latencia, consumo de tokens, tiempo por agente, repeticiones, errores y cumplimiento geométrico. De forma complementaria, los archivos finales fueron revisados cualitativamente para comparar su validez técnica con la correspondencia visual respecto del prompt. Los resultados estadísticos y cualitativos derivados de este Sprint se desarrollan en las secciones posteriores del presente capítulo.

Durante la iteración también se realizaron correcciones finales en relación con la interfaz, mensajes que aparecían de forma no adecuada para el usuario y organización de la documentación técnica basado en el análisis de los resultados de las pruebas. El incremento final fue una versión más pulida de Bezy capaz de autenticar usuarios de forma real,

administrar conversaciones en base a una estructura basada en chats, generar archivos vectorial, así como su almacenamiento en la nube, validacion de rutas, verificando el token del usuario logeado, conversión de resultados vectoriales a DXF y registro de la telemetría. Con este sprint se finalizó el desarrollo y se obtuvo la versión evaluada en la investigación.

## Resultados de la evaluación técnica y cualitativa

### *Estrategia de análisis y cobertura de las pruebas*

El análisis de resultados se desarrolló mediante la integración de evidencia cuantitativa y cualitativa, de acuerdo con el enfoque mixto con predominio cuantitativo definido para la investigación. La parte cuantitativa consistió en la estadística descriptiva de la telemetría registrada en Langfuse y las métricas técnicas de los archivos finales. La parte cualitativa consistió en una revisión del contenido de los diseños generados, evaluando si se correspondían con la solicitud, si eran visualmente claros y si podían cortarse con láser.

Se evaluaron 90 casos de prueba. Estos se definieron antes del inicio de las pruebas oficiales, se distribuyeron equitativamente entre 3 niveles de complejidad (30 casos de Nivel 1, 30 de Nivel 2 y 30 de Nivel 3) y, debido a un error (repeticiones), resultaron en 121 ejecuciones únicas (39 de Nivel 1, 41 de Nivel 2 y 41 de Nivel 3). Por tanto, en este capítulo se diferencia entre *caso de prueba*, que corresponde a uno de los 90 prompts oficiales, y *ejecución*, que representa cada intento registrado para procesar un caso.

Para interpretar el desempeño se utilizaron la tasa de éxito por ejecución, el éxito en el primer intento, el estado técnico final de cada caso, la latencia, el cumplimiento del límite de 45 segundos, el consumo de tokens, el tiempo por agente y el costo técnico de los fallos. Las capturas duplicadas de una misma traza no se contabilizaron como ejecuciones adicionales. Asimismo, los registros exploratorios generados antes de las pruebas oficiales no se incorporaron al análisis.

**Tabla 27**

*Cobertura de las pruebas oficiales por nivel de complejidad*

| Nivel | Casos evaluados | Ejecuciones únicas | Ejecuciones adicionales |
|---|---|---|---|
| Nivel 1 | 30 | 39 | 9 |
| Nivel 2 | 30 | 41 | 11 |
| Nivel 3 | 30 | 41 | 11 |
| Total | 90 | 121 | 31 |

La cobertura fue del 100% respecto de los prompts planificados. Las 31 ejecuciones adicionales permiten evaluar tanto la capacidad del resultado final, como también la estabilidad o casos especiales del pipeline donde el proceso arroje resultados que rompan con la media del total de casos evaluados, esto apoya a la idea de que la IA puede verse rebasada en casos donde el análisis tome mucho mas tiempo del estimado.

***Confiabilidad técnica y trazabilidad del pipeline***

De las 121 ejecuciones documentadas, 108 lograron finalizar correctamente, destacando que esto no equivale a que su resultado haya sido satisfactorio, sino que el modelo terminó su ejecución. Por otra parte, 13 terminaron con error, destacando que la tasa de éxito técnico por ejecución fue del 89.3%. Poniendo en perspectiva el primer intento de cada uno de los 90 casos, se puede establecer que 80 finalizaron correctamente, dando un porcentaje de éxito inicial del 88.9%. Después de las repeticiones documentadas, los 90 casos alcanzaron una ejecución final técnicamente exitosa.

**Tabla 28**

*Confiabilidad técnica del pipeline por nivel de complejidad*

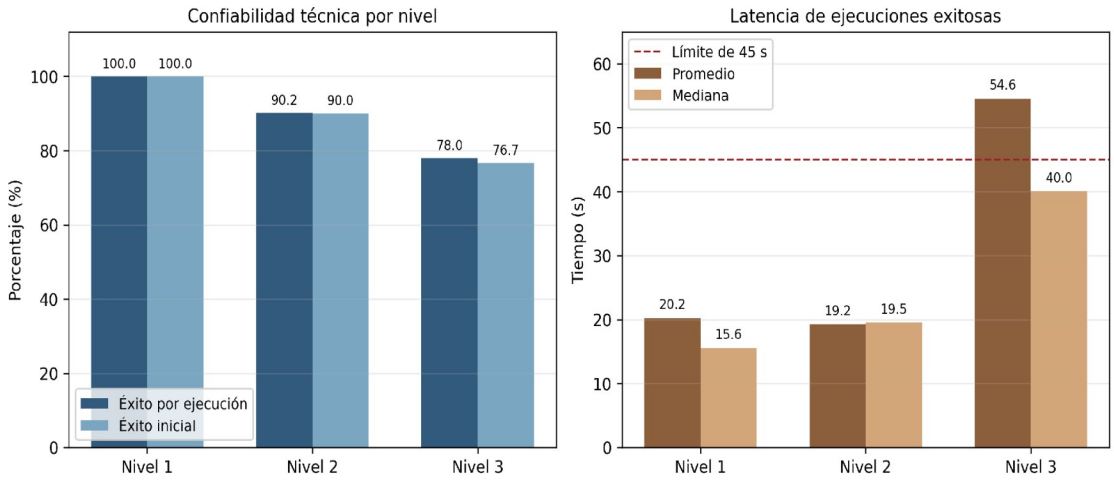
| Nivel | Éxitos | Fallos | Éxito por ejecución | Éxito inicial | Éxito final |
|---|---|---|---|---|---|
| Nivel 1 | 39 | 0 | 100% | 100% | 100% |

| Nivel | Éxitos | Fallos | Éxito por ejecución | Éxito inicial | Éxito final |
|---|---|---|---|---|---|
| Nivel 2 | 37 | 4 | 90,2% | 90% | 100% |
| Nivel 3 | 32 | 9 | 78% | 76,7% | 100% |
| Total | 108 | 13 | 89,3% | 88,9% | 100% |

La Tabla 28 muestra una reducción de estabilidad conforme aumenta la complejidad. El Nivel 1 no registró fallos, mientras que el Nivel 2 presentó cuatro y el Nivel 3 concentró nueve. El éxito final del 100% no debe interpretarse como ausencia de errores ni como cumplimiento semántico completo; significa que todos los casos lograron finalizar técnicamente después de los intentos documentados. Esta distinción es relevante debido a su posible interpretación errónea y recalcando que el esfuerzo requerido para alcanzarlo no se vea disminuido.

**Figura 17**

*Confiabilidad técnica y latencia por nivel de complejidad*




La trazabilidad pudo determinar que 27 casos tuvieron más de una ejecución: ocho por la parte del nivel 1, diez en el Nivel 2 y nueve en el Nivel 3. Además, se observaron 11 trazas con regeneración interna, en otras palabras, significa que el validador devuelve el resultado al

agente, solicitando un reintento con un máximo de 3 ejecuciones, diez de estos intentos provienen del Nivel 3. En este subconjunto, cinco ejecuciones pudieron finalizar correctamente y cinco terminaron con un error por parte del modelo generado y que el validador consideró como no adecuado, esto destaca que los reintentos no garantizan la recuperación del proceso o que el mismo sea satisfactorio.

***Latencia y cumplimiento del requisito temporal***

La latencia se calculó sobre las 108 ejecuciones técnicamente exitosas. Los niveles 1 y 2 presentaron tiempos promedio similares, de 20,25 y 19,25 segundos, respectivamente. En el Nivel 3, la media aumentó a 54,57 segundos y la mediana a 40,05 segundos. La diferencia entre ambas medidas evidencia la presencia de ejecuciones prolongadas que elevaron el promedio del grupo.

**Tabla 29**

*Latencia y cumplimiento temporal de las ejecuciones exitosas*

| Nivel | n | Promedio (s) | Mediana (s) | P95 (s) | Ejecuciones < 45s |
|---|---|---|---|---|---|
| Nivel 1 | 39 | 20,25 | 15,5 | 46,36 | 94,9% |
| Nivel 2 | 37 | 19,25 | 19,50 | 41,48 | 94,6% |
| Nivel 3 | 32 | 54,57 | 40,05 | 131,20 | 54,4% |
| **Total** | **108** | **30,08** | **–** | **–** | **84,3%** |

El requisito no funcional de respuesta máxima de 45 segundos se cumplió en 91 de las 108 ejecuciones exitosas, equivalentes al 84,3%. Esto se cumplió superando el 94% en los niveles 1 y 2, considerados como los niveles más aptos para el modelo. Sin embargo, en el Nivel 3 se descendió a 59,4%. En consecuencia, el requisito es recomendado para solicitudes simples e intermedias, pero no es capaz de satisfacer los parámetros ni requerimientos de diseños complejo, cuyo percentil 95 alcanzó 131,20 segundo, denotando un gran consumo de tokens, traducido a un gran tiempo de análisis por parte del modelo generador.

***Consumo de tokens y distribución del tiempo por agente***

El promedio de las ejecuciones exitosas está en un aproximado de 3 621 tokens en el Nivel 1, 3 586 en el Nivel 2 y 6 949 en el Nivel 3 respectivamente, recalcando que su análisis fue realizado en base al modelo de Deep Seek a través de la plataforma Deepinfra. Por tanto, el Nivel 3 utilizó aproximadamente 93,8% más tokens que el Nivel 2. La mediana del Nivel 3 fue de 5 795 tokens y su máximo alcanzó 15 378, lo que confirma una mayor variabilidad en las solicitudes de alta complejidad.

**Tabla 30**

*Consumo de tokens en las ejecuciones exitosas*

| Nivel | Promedio | Mediana | Desv. estándar | P95 | Máximo |
|---|---|---|---|---|---|
| Nivel 1 | 3 621 | 4 035 | 1 770 | 5 322 | 10.062 |
| Nivel 2 | 3 586 | 4 230 | 1 573 | 5.629 | 6 348 |
| Nivel 3 | 6 949 | 5 795 | 3 312 | 13 936 | 15 378 |

**Estimación del costo monetario del modelo de inteligencia artificial**

El consumo acumulado de DeepSeek V4 Pro dentro del desarrollo de Bezy comprendió 214 749 tokens de entrada, 487 473 tokens de salida y 80 384 tokens de entrada procesados mediante caché. De acuerdo con las tarifas publicadas por DeepInfra para este modelo, de USD 1,30 por millón de tokens de entrada, USD 2,60 por millón de tokens de salida y USD 0,10 por millón de tokens en caché, el costo se estimó mediante la siguiente expresión [17]:

$$C = \frac{(214\,749 \times 1{,}30) + (487\,473 \times 2{,}60) + (80\,384 \times 0{,}10)}{1\,000\,000}.$$

**Tabla 31**

*Estimación del consumo monetario acumulado de DeepSeek V4 Pro en Bezy*

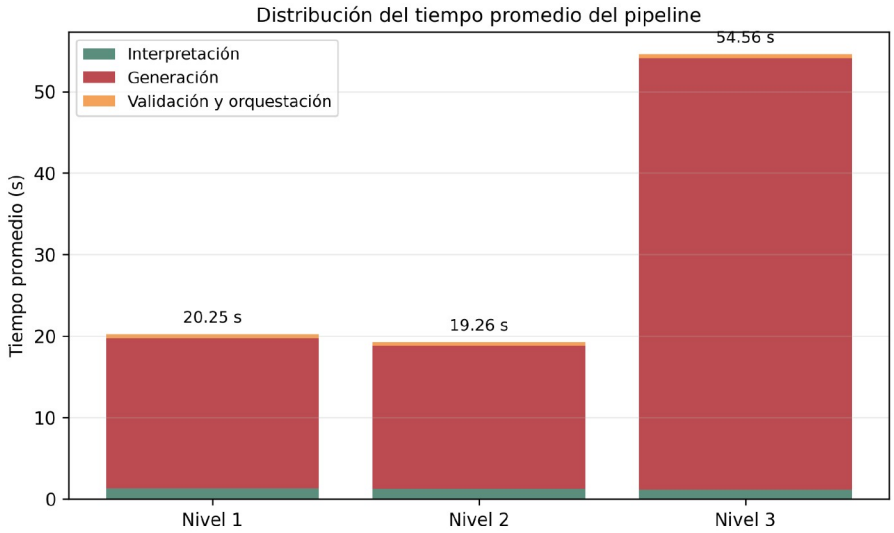
| Tipo de consumo | Tokens | Tarifa por millón (USD) | Costo estimado (USD) |
|---|---|---|---|
| Entrada | 214 749 | 1,30 | 0,2792 |
| Salida | 487 473 | 2,60 | 1,2674 |
| Entrada en caché | 80 384 | 0,10 | 0,0080 |
| **Total** | **782 606** | **–** | **1,5546** |

El resultado equivale aproximadamente a USD 1,55 cuando el redondeo se aplica al costo total calculado. Al sumar los costos individuales mostrados por el proveedor con dos decimales, USD 0,28, USD 1,27 y USD 0,01, se obtiene un valor acumulado de USD 1,56. Esta estimación representa el consumo observado del modelo en específico de DeepSeek V4 Pro durante el desarrollo de Bezy, mas no como el costo total de infraestructura ni como el costo exclusivo de las 90 pruebas oficiales, ya que se ha sumado el consumo total a partir del primer uso o prueba realizado con el modelo.

La distribución a través del pipeline muestra que la etapa en el agente de interpretación permanece estable alrededor de un segundo, considerablemente bajo debido al hecho de que su función es simplemente traducir la petición del usuario a algo más estructurado para pasarlo al siguiente modelo. En cambio la etapa más importante, que de igual forma concentró la mayor parte del tiempo de ejecución, es el modelo generador, concentrando el 91,0% del tiempo en el Nivel 1, 91,4% en el Nivel 2 y 97,1% en el Nivel 3. La validación y la sobrecarga de orquestación se mantuvieron por debajo de un segundo aproximadamente.

**Figura 18**

*Distribución del tiempo promedio del pipeline por nivel*




Este resultado identifica la etapa de generación como un cuello de botella operativo, debido a que su tiempo promedio es mucho más alto que el de los demás. Este alto consumo también se ve reflejado en el consumo de tokens de salida y latencia, que fue de 0,919 en el Nivel 1, 0,899 en el Nivel 2 y 0,948 en el Nivel 3. En términos descriptivos, las respuestas más extensas son equivalentes a mayor tiempo de razonamiento, especialmente en pruebas complejas observadas en el Nivel 3.

Al comparar únicamente las ejecuciones del modo geométrico, la latencia promedio aumentó de 25,03 segundos en el Nivel 2 a 57,76 segundos en el Nivel 3, un incremento de 130,8%. En el mismo contraste, el consumo promedio pasó de 4 534 a 7 321 tokens, equivalente a un aumento de 61,5%. Esto demuestra que la diferencia del Nivel 3 no se explica únicamente por el modo seleccionado, sino también por la mayor exigencia de las solicitudes evaluadas.

***Errores y costo técnico de los fallos***

Las 13 ejecuciones fallidas consumieron 1 058,83 segundos y 125 092 tokens. Estos valores representan aproximadamente el 24,6% del tiempo total observado y el 20,1% de los

tokens registrados en las 121 ejecuciones. El Nivel 3 concentró el 92,1% del tiempo perdido y el 88,4% de los tokens asociados con errores.

**Tabla 32**

*Costo técnico de las ejecuciones fallidas*

| Nivel | Fallos | Tiempo desperdiciado (s) | Tokens desperdiciados |
|---|---|---|---|
| Nivel 1 | 0 | 0,00 | 0 |
| Nivel 2 | 4 | 84,21 | 14 485 |
| Nivel 3 | 9 | 974,62 | 110 607 |
| **Total** | **13** | **1 058,83** | **125 092** |

Los errores observados correspondieron principalmente a respuestas incompletas, estructuras no procesables y fallos detectados durante el procesamiento del resultado. Los errores tempranos tuvieron un costo relativamente más reducido, mientras que los que pasaron más tiempo generando un diseño y terminaron en error consumieron considerablemente más tiempo y tokens. Este hallazgo evidencia que sin reglas que delimiten el tiempo de regeneración, los recursos como los son el consumo de tokens se verán en un consumo innecesario, por lo que se deben aplicar verificaciones o límites que detengan un posible exceso.

### *Validez técnica de los archivos finales*

La telemetría representa los valores en base al comportamiento del pipeline, esto se traduce en la trazabilidad completa de cada modelo y la comunicación entre ellos a través de una petición. Por otro lado, la matriz definida para la evaluación técnica de los archivos, definida en la base de datos, representa el análisis en el archivo final aceptado para cada uno de los 90 casos de pruebas. Esta diferencia explica que existan 13 ejecuciones fallidas y, al mismo tiempo, 90 resultados finales disponibles para validación geométrica.

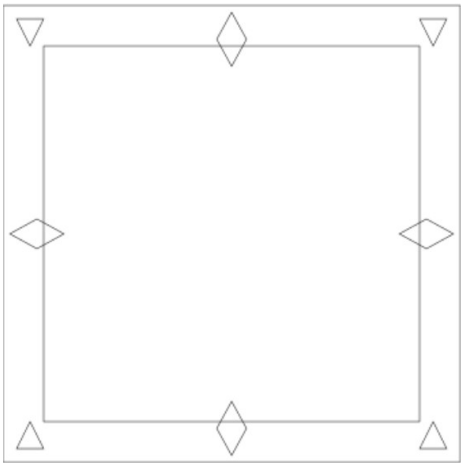
Los archivos finales cumplieron las condiciones geométricas mínimas establecidas por el instrumento: estructura SVG válida, cierre de trayectorias y ausencia de errores geométricos detectados. A pesar de esto, los indicadores verifican la integridad técnica y no representan un nivel de fidelidad semántica, calidad estética ni aptitud física en la etapa de fabricación, el cual es un proceso manual, aunque de menor complejidad comparado con el proceso de creación del diseño. Variables como las rutas, curvas de Bézier y uso del búfer complementaron la caracterización de cada diseño, pero no se establecen como valores de éxito, para eso se deben tomar en cuenta mas características independientes del diseño final.

***Interpretación cualitativa de los resultados***

La revisión del análisis cualitativo pudo contrastar la validez geométrica con la correspondencia visual y funcional de los archivos y sus formatos. En el Nivel 1, las solicitudes se resolvieron mediante formas básicas y reconocibles, así como una relación directa entre el prompt y el resultado, arrojando un porcentaje de satisfacción más alto. En el Nivel 2 se mantuvo la idea principal, aunque algunos diseños revelaron simplificaciones o resultados variados. En el Nivel 3 se observaron con mayor frecuencia diferencias entre la idea principal y la composición arrojada por el modelo, especialmente cuando el prompt exige múltiples piezas o piezas que deben ser concatenadas, subiendo la dificultad a que el modelo deba comprender un plano en 3D, algo en lo que se ha observado que los modelos no son muy buenos hasta la fecha.

**Figura 19**

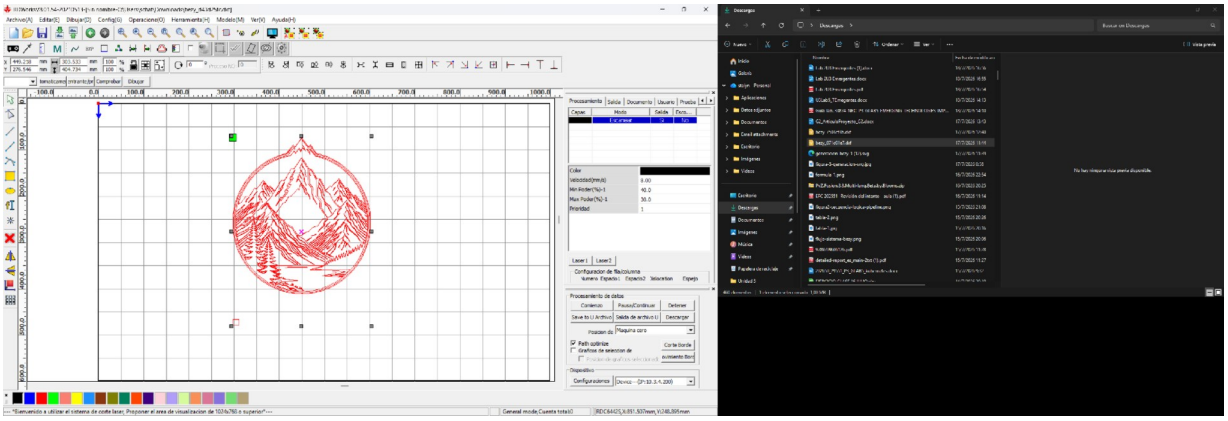
*Ejemplo de marco decorativo generado por Bezy*



El marco decorativo de la Figura 19 presenta una composición ordenada y cerrada, con líneas rectas y uso de figuras geométricas básicas, coherente con un diseño de contorno, un resultado que se podría considerar adecuado en base a la petición inicial. En contraste, los diseños que contengan texto o alta densidad de elementos pueden llegar a requerir una revisión adicional, puesto que sus formas (dependiendo de la fuente disponible) puedan tener contornos de líneas un poco más complicadas de definir entre las capas de CUT y ENGRAVE, aunque esto realmente sería un escenario más probable en figuras con un modo orgánico, en donde los diseños provienen originalmente de una imagen y no de un diseño vectorial.

**Figura 20**

*Diseño detallado de un paisaje montañoso visualizado en RDWorks*



La Figura 20 evidencia que un archivo puede ser geométricamente válido y, aun así, contener una elevada concentración de trayectorias. Aunque las entidades se encuentren

separadas en capas de corte y grabado, esta densidad incrementa la complejidad de revisión y puede reducir la conveniencia del diseño si no se simplifican detalles o se configuran adecuadamente los parámetros de procesamiento. Por esto, la calidad técnica y la validez al prompt deben ser analizados como dimensiones que se complementan para definir un resultado adecuado, sin embargo, no se debe confundir como una equivalencia entre las mismas.

La evidencia del análisis cualitativo pudo confirmar que el sistema es más estable en solicitudes orientadas a contornos y corte monolinea, así como peticiones estrictamente definidas en relación con parámetros como la definición de corte o dimensiones. Las solicitudes con texto, rellenos, sombreado o piezas solicitadas para ensamble demandan supervisión humana, puesto a su alto nivel de complejidad. Esta observación se complementa con los resultados y estadísticas obtenidas del Nivel 3, en el cual no solo representó una mayor latencia y consumo, sino que se evidenció una mayor dificultad para conservar la intención inicial del usuario, resultando en un diseño probablemente no válido o de igual forma terminando en un fallo.

***Discusión integrada de los resultados***

Los resultados dieron una perspectiva de que la arquitectura implementada y planificada permite transformar instrucciones en lenguaje natural en archivos tanto vectoriales como de formato basado en capas y diseños 2D. Sin embargo, la evidencia también demuestra que la complejidad afecta la estabilidad y calidad final: el éxito por ejecución descendió a un 78% en el Nivel 3, mientras que la latencia promedio aumentó de 20,25 a 54,57 segundos, determinando así que el alcance en el que todavía el modelo puede arrojar resultados aptos basados en la fidelidad de la petición inicial, es el Nivel 2.

La separación de responsabilidades entre interpretación, generación, validación y orquestación favoreció la observación detallada del proceso, en concordancia con los enfoques multiagente que distribuyen tareas complejas entre componentes especializados [3], [13]. A su

vez, la validación determinista permitió evitar que la aceptación del resultado dependiera únicamente de su apariencia. Este comportamiento coincide con la literatura que advierte que los modelos de lenguaje pueden producir estructuras vectoriales plausibles, pero requieren controles adicionales para mantener consistencia geométrica [2], [11].

**Tabla 34**

*Relación entre objetivos específicos y resultados obtenidos*

| Objetivo específico | Evidencia obtenida | Resultado interpretado |
|---|---|---|
| Analizar el estado del arte, definir los requerimientos funcionales e implementar la plataforma y los agentes para transformar prompts en geometrías vectoriales cerradas y escalables. | Revisión del estado del arte, requisitos consolidados,plataforma implementada y evaluación de 90 archivos finales. | El sistema integró interpretación, generación y validación, y produjo un archivo final técnicamente válido para cada caso oficial. |
| Desarrollar el backend integrando la arquitectura de micro-agentes de IA y la interfaz frontend. | Integración de React, Node.js, FastAPI, Supabase, Langfuse y los agentes evaluados mediante 121 ejecuciones únicas. | La arquitectura permitió coordinar solicitudes, procesar la generación vectorial, conservar archivos y registrar la trazabilidad del pipeline. |

Se puede deducir entonces que el prototipo alcanzó un estado técnico evidenciable en los 90 casos de pruebas, pero se necesitó repeticiones en escenarios donde en el primer intento no se obtuvo un resultado o terminó en error durante el proceso de generación. De igual manera, los costos aumentaron conforme aumentó la complejidad, un resultado esperado considerando que un mayor tiempo de ejecución equivale a más tokens quemados. Con esto claro, es preciso determinar que el aporte principal después de evaluar y obtener los valores finales de cada prueba, no definen o establecen que todos los intentos fueron perfectos, sino que ayudan a esclarecer que el sistema tiene protocolos ante posibles fallos, y aun así no es

ajeno a errores. Con esto se pudo registrar adecuadamente los escenarios en donde se obtuvieron complicaciones y donde se evidencia las variables que definen el comportamiento del modelo, ayudando a establecer los puntos donde se puede hacer mejoras en posibles trabajos futuros.

***Limitaciones del análisis y del sistema***

La telemetría enfocada en la obtención de datos de los agentes, vino de parte de la aplicación de Langfuse, y debido a su configuración, algunos tiempos superiores a un minuto se establecieron con un valor redondeado. Además, cuando se realizaba una repetición no se definió específicamente la causa de esta, lo que limita la separación exacta entre escenarios donde la repetición fue de forma manual o se debió a los reintentos internos del modelo validador.

La configuración y evaluación realizada usando el modelo de DeepSeek V4 Pro y de los prompts asignados a cada agente se establecieron de forma constante durante la fase de pruebas en los 90 casos. Como resultado, se determinó el comportamiento de la arquitectura, y los valores obtenidos no deben generalizarse en caso de que se reemplace el modelo, ni siquiera si se trata de un modelo diferente de la misma DeepSeek, ya que su consumo se ve reflejado en las capacidades del mismo, así como en temas como el proveedor utilizado. Se estimó un valor aproximado de USD 1,55 en la parte de tokens de entrada, salida y cache del modelo durante toda la etapa de desarrollo, sin embargo, este valor no representa el costo total del pipeline, porque no se usó este modelo en todos los agentes, además, no es establece valores de almacenamiento ni otros servicios que puedan definir el costo de infraestructura. Adicionalmente, se deja claro que los valores no corresponden estrictamente a las 90 pruebas, sino a toda la fase de desarrollo en el que se usó este modelo.

La validación técnica se centró en la estructura netamente del archivo, el cierre de las rutas y la ausencia de errores geométricos detectados, o se puede interpretar como reglas que deben cumplirse para la conversión al formato DXF. No se realizó una fabricación de los 90

casos de prueba, pero se realizó un par para poner a pruebas que se sigan las reglas de dimensiones, capas y continuidad de las líneas. La interpretación cualitativa se basa en un juicio técnico, y aunque se aplicaron criterios generales, puede que el análisis pase por alto especificaciones más subjetivas.

Con esto, Bezy asigna los colores a las capas CUT y ENGRAVE mediante los identificadores de grupo definidos en el SVG. Se debe tener en cuenta que la pertinencia de dicha clasificación radica en el agente intérprete y como definió cada elemento. Tomando esto en cuenta, los resultados finales no están absueltos de una revisión humana debido a su posible inconformidad con el diseño obtenido o posibles errores milimétricos que puedan observarse dentro del software de edición CAD.

# Capítulo V

## Conclusiones

El desarrollo del sistema Bezy permitió comprobar que es posible generar archivos vectoriales orientados a corte láser a partir de instrucciones en lenguaje natural mediante un pipeline apoyado en agentes de inteligencia artificial. La evaluación comprendió 90 casos oficiales y 121 ejecuciones únicas. El éxito técnico por ejecución fue de 89,3%, mientras que los 90 casos alcanzaron finalmente una ejecución técnicamente exitosa después de las repeticiones documentadas.

En cuanto a la automatización de la generación de vectores, la arquitectura final, compuesta por servicios de orquestación, agentes especializados, un validador geométrico y almacenamiento de métricas, permitió transformar la entrada del usuario en archivos técnicamente evaluables, rastrear el proceso y registrar la latencia, los tokens consumidos, el estado del proceso, la validez del SVG, el cierre de la ruta y los errores.

Respecto a la validación geométrica, los 90 archivos finales aceptados presentaron rutas cerradas y no registraron errores geométricos detectados por la herramienta. Esto no quiere decir que los 13 errores observados hayan sido eliminados, sino que los pasos de validación y repetición basados en reglas internas del modelo permitieron conservar el resultado final, que se puede considerar como condiciones técnicas que cumplen con un diseño de corta de monolinea.

Finalmente, se concluye que la diferenciación entre las capas de CUT y ENGRAVE se define mediante grupos y colores en el SVG, así como en el formato DXF. Esta funcionalidad facilita la preparación del archivo, aunque su utilidad se ve más aprovechada en el modo orgánico. Dich esto, cabe recalcar que esto no eliminar la revisión humana antes de la fabricación, especialmente cuando el diseño generado es propenso a tener varias líneas pegadas o figuras sobrepuestas, escenarios en los cuales la maquina cortar secciones donde debería ir un grabado, o donde simplemente el modelo no separó adecuadamente las piezas.

**Recomendaciones**

Se recomienda definir reglas más enfocadas a la parte que se establece las capas, esto especialmente en las solicitudes en donde se combinen contornos o detalles decorativos, así como piezas sueltas. Esta mejora debería definirse en el agente intérprete, y asegurarse que el agente generador sea capaz de definir dichas reglas, mejorando así la clasificación de propiedades y mejorando la precisión ante peticiones ambiguas.

Para mejoras o correcciones a futuro, se puede trabajar con formatos distintos del SVG, con el objetivo de ofrecer una configuración más personalizada, teniendo en cuenta el tipo de herramienta que se piensa usar, así como características de la máquina en la que se piensa trabajar. En particular, el formato .rld es uno de los más atractivos puesto que permite parametrizar valores de velocidad y potencia del láser, automatizando aún más el proceso de fabricación.

Se recomienda ampliar las parámetros de evaluación con nuevos casos de prueba, como ejemplos válidos se tomarían prompts con estructura de petición más ambigua, logotipos o patrones densos. Esto permitiría determinar de forma más amplia los límites del modelo, así como fortalecer el proceso de validación cualitativa.

Asimismo, uno de los puntos más relevantes que se deben tener en cuenta como recomendación, es la optimización de tiempo y recursos que utiliza el agente generador. Con esto se podría reducir el tiempo o incluso controlar casos atípicos en base a reglas estrictas como límites de consumo de tokens o tiempo que puede tardar una generación.

Finalmente, es recomendable ampliar el validador geométrico implementado mediante librerías de Python con el fin de incorporar verificaciones específicas para detalles en el tema de grabado o corte, dando así una validación técnica previa que incluya la continuidad geométrica y la coherencia de la operación asignada.

## Referencias

[1] M. Cai, Z. Huang, Y. Li, U. Ojha, H. Wang, and Y. J. Lee, “An investigation on LLMs’ visual understanding ability using SVG for image-text bridging.” 2023, [Online]. Available: https://arxiv.org/abs/2306.06094.

[2] B. Malashenko, I. Jarsky, and V. Efimova, “Leveraging large language models for scalable vector graphics processing: A review.” 2025, [Online]. Available: https://arxiv.org/abs/2503.04983.

[3] T. Guo et al., “Large language model based multi-agents: A survey of progress and challenges.” 2024, [Online]. Available: https://arxiv.org/abs/2402.01680.

[4] Universidad de las Fuerzas Armadas ESPE, Guía metodológica para la presentación de trabajos de integración curricular. Unidad de Desarrollo Educativo, 2026.

[5] United Nations, “Goal 9: Industry, innovation and infrastructure,” 2026. https://sdgs.un.org/goals/goal9 (accessed July 30, 2026).

[6] R. Wu, W. Su, and J. Liao, “Chat2SVG: Vector graphics generation with large language models and image diffusion models,” 2025.

[7] X. Xing, J. Hu, G. Liang, J. Zhang, D. Xu, and Q. Yu, “Empowering LLMs to understand and generate complex vector graphics,” in Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR), 2025, pp. 19487–19497.

[8] A. Banerjee, N. Mathur, J. Lladós, U. Pal, and A. Dutta, “SVGCraft: Beyond single object text-to-SVG synthesis with comprehensive canvas layout.” 2024, [Online]. Available: https://arxiv.org/abs/2404.00412.

[9] Y. Yang et al., “OmniSVG: A unified scalable vector graphics generation model.” 2025, [Online]. Available: https://arxiv.org/abs/2504.06263.

[10] F. Wang et al., “SVGen: Interpretable vector graphics generation with large language models.” 2025, [Online]. Available: https://arxiv.org/abs/2508.09168.

[11] J. Kuchař, M. Kadlčík, M. Spiegel, and M. Štefánik, “VectorEdits: A dataset and benchmark for instruction-based editing of vector graphics.” 2025, [Online]. Available: https://arxiv.org/abs/2506.15903.

[12] S. Russell and P. Norvig, Artificial intelligence: A modern approach, 2nd ed. Prentice Hall, 2003.

[13] Y. Cheng et al., “Exploring large language model based intelligent agents: Definitions, methods, and prospects.” 2024, [Online]. Available: https://arxiv.org/abs/2401.03428.

[14] L. Wang et al., “A survey on large language model based autonomous agents.” 2023, [Online]. Available: https://arxiv.org/abs/2308.11432.

[15] C. Drumond, “¿Qué es scrum? Guía para el marco de trabajo ágil,” 2026. https://www.atlassian.com/es/agile/scrum (accessed July 27, 2026).

[16] R. H. Sampieri, C. F. Collado, and M. del Pilar Baptista Lucio, Metodología de la investigación, 6th ed. McGraw-Hill Education, 2014.

[17] DeepInfra, “DeepSeek V4 Pro pricing and model information,” 2026. https://deepinfra.com/deepseek-ai/DeepSeek-V4-Pro (accessed July 30, 2026).