PONTIFICIA UNIVERSIDAD CATÓLICA DEL ECUADOR

FACULTAD DE HÁBITAT, INFRAESTRUCTURA Y CREATIVIDAD

CARRERA DE INGENIERÍA EN SISTEMAS DE INFORMACIÓN



TRABAJO DE TITULACIÓN

Tema: Implementación de un prototipo funcional basado en inteligencia artificial para la generación automática de tarjetas de estudio a partir de documentos digitales.

AUTORA:

DANAHÉ ANDRADE PALADINES

TUTORA:

SUYANA FABIOLA ARCOS VILLAGÓMEZ

QUITO 20, JULIO DE 2026

AGRADECIMIENTOS

Quiero expresar mi más profundo agradecimiento a mi familia, quienes han sido el pilar fundamental durante todo este camino académico. Gracias a mis padres, por su amor incondicional, sus sacrificios y por enseñarme que el esfuerzo y la perseverancia son la clave para alcanzar cada meta. A mi hermana, mi hermano y mi cuñado, gracias por sus consejos, por creer en mí, por motivarme cada día a seguir aprendiendo y a luchar por alcanzar objetivos cada vez más grandes. Cada uno de ustedes ha dejado una huella invaluable en mi vida y ha sido parte esencial de este logro.

También quiero dedicar unas palabras muy especiales a mi perrita Luna, quien, con su compañía incondicional, estuvo presente durante las largas jornadas de estudio y desarrollo de esta aplicación web. Su presencia hizo más llevaderos los momentos de cansancio y estrés, recordándome siempre la importancia de disfrutar el proceso.

Finalmente, agradezco a todas las personas que, de una u otra manera, me brindaron su apoyo, confianza y motivación para no rendirme y continuar avanzando. La culminación de esta tesis representa el cierre de una etapa inolvidable de mi vida, llena de aprendizajes, desafíos y crecimiento personal y profesional, por la cual siempre estaré profundamente agradecida.

RESUMEN

El presente trabajo describe el diseño e implementación de StudyCards AI, un prototipo de aplicación web que genera automáticamente tarjetas de estudio en formato pregunta–respuesta a partir de documentos PDF y DOCX, mediante la integración de la API de Gemini 1.5 Pro. La solución responde a la necesidad de los estudiantes universitarios de transformar material académico digital en recursos de repaso estructurados sin inversión manual de tiempo.

El sistema se desarrolló con Vue 3 y TypeScript en el frontend, ASP.NET Core 8.0 en el backend y Google Firestore como base de datos. El pipeline de procesamiento incluye extracción de texto, segmentación semántica por párrafos, construcción del prompt maestro y un módulo de validación multicapa que filtra tarjetas de baja calidad antes del almacenamiento.

Los resultados confirman que la ingeniería de prompts es el factor determinante en la calidad del material generado y que es viable incorporar modelos de lenguaje de gran escala en aplicaciones educativas funcionales sin infraestructura de entrenamiento propia.

ÍNDICE

AGRADECIMIENTOS ii
RESUMEN iii
ÍNDICE DE TABLAS vii
ÍNDICE DE FIGURAS viii
CAPÍTULO I 1
DEFINICIÓN DEL PROYECTO DE TITULACIÓN 1
1.1 Tema 1
1.2 Justificación 1
1.3 Planteamiento del Problema 3
1.4 Objetivos 4
1.4.1 Objetivo General 4
1.4.2 Objetivos Específicos 4
1.5 Alcance 5
CAPÍTULO II 7
MARCO TEÓRICO 7
2.1 Inteligencia Artificial Aplicada a la Educación Digital 7
2.2 Modelos de Lenguaje Natural de Gran Escala 8
2.3 Criterios de Selección de Modelos Preentrenados 10
2.4 Ingeniería de Prompts 11
2.5 Generación Automática de Contenido Educativo 12
2.6 Arquitectura de Software para Sistemas con Inteligencia Artificial 13
2.7 Procesamiento de Documentos Digitales y Segmentación de Texto 14
2.8 Personalización del Material de Estudio 15
2.9 Métricas de Evaluación de Calidad en Generación de Texto 15
2.10 Relación del Marco Teórico con la Propuesta 17
CAPÍTULO III 18
DESARROLLO DE LA PROPUESTA 18
3.1 Metodología de Desarrollo 18
3.2 Análisis de Requerimientos 19
3.3 Diagrama de Casos de Uso 22
3.4 Arquitectura del Sistema 24
3.5 Diagrama de Secuencia 26

3.6 Pipeline de Procesamiento e Inteligencia Artificial 27
3.7 Selección del Modelo de Lenguaje 29
3.8 Diseño del Prompt de Generación 33
3.9 Modelo de Datos 35
3.10 Módulo de Validación y Limpieza de Tarjetas 37
3.11 Implementación de la Interfaz de Usuario 38
3.11.1 Módulo de Autenticación 38
3.11.2 Módulo de Carga y Generación 39
3.11.3 Módulo de Visualización 41
3.12 Consideraciones de Seguridad 43
CAPÍTULO IV 44
RESULTADOS Y VALIDACIÓN FUNCIONAL 44
4.1 Introducción al Capítulo 44
4.2 Diseño de la Validación 44
4.3 Resultados del Pipeline de Procesamiento Documental 46
4.4 Resultados de la Generación y Validación de Tarjetas 48
4.5 Evaluación de Calidad de las Tarjetas Generadas 50
4.5.1 Análisis por Nivel de Dificultad 51
4.6 Verificación del Cumplimiento de Requerimientos 52
4.7 Limitaciones Identificadas 54
4.8 Discusión de Resultados 55
4.9 Síntesis del Capítulo 56
5.1 Introducción 58
5.2 Conclusiones 58
5.2.1 La viabilidad técnica del prototipo se verificó de forma empírica. 58
5.2.2 La ingeniería de prompts es el componente de mayor impacto sobre la calidad del material generado. 59
5.2.3 El módulo de validación multicapa opera como un filtro de calidad efectivo y proporcional 59
5.2.4 La arquitectura en capas con servicios externos desacoplados sugiere ser un patrón sólido para sistemas que integran múltiples APIs. 60
5.2.5 La parametrización del nivel de dificultad produce diferencias cognitivas observables y alineadas con la taxonomía de Bloom. 60
5.3 Recomendaciones 61

5.3.1 Incorporar evaluación externa con expertos en dominio para robustecer la validación de calidad. 61
5.3.2 Implementar extracción estructurada de tablas en documentos PDF para superar la limitación de flujo lineal. 61
5.3.3 Refinar las directivas cognitivas del prompt para el nivel avanzado de dificultad. 62
5.3.4 Extender el sistema con un módulo de evaluación de aprendizaje y seguimiento de progreso. 62
5.3.5 Evaluar la integración del sistema con plataformas educativas institucionales y ampliar los formatos de entrada admitidos. 63
REFERENCIAS BIBLIOGRÁFICAS 65
ANEXOS 69
ANEXO 1: Código fuente de prompt para elaboración de tarjetas 69
ANEXO 2: Código fuente de extracción de texto 74

ÍNDICE DE TABLAS

Tabla 3. 1 Requerimientos funcionales del sistema.....20

Tabla 3. 2 Requerimientos no funcionales del sistema.....21

Tabla 3. 3 Comparación técnica de modelos de lenguaje evaluados.....29

Tabla 3. 4 Estructura y componentes del prompt maestro de generación de tarjetas.....34

Tabla 3. 5 Filtros del módulo de validación y limpieza de tarjetas. 37

Tabla 4. 1 Documentos utilizados en la validación funcional del prototipo.....45

Tabla 4. 2 Resultados del pipeline de extracción y segmentación por documento.....47

Tabla 4. 3 Tarjetas generadas, descartadas y almacenadas por documento.....49

Tabla 4. 4 Evaluación de calidad de tarjetas por dimensión y documento 51

Tabla 4. 5 Verificación del cumplimiento de requerimientos funcionales y no funcionales 52

ÍNDICE DE FIGURAS

Figura 3. 1 Diagrama de casos de uso del sistema StudyCards AI. 23

Figura 3. 2 Diagrama de componentes – arquitectura por capas del sistema. 24

Figura 3. 3 Diagrama de secuencia – flujo de generación de tarjetas de estudio. 26

Figura 3. 4 Diagrama de actividad – pipeline de generación de tarjetas. 27

Figura 3. 5 Diagrama de clases – modelo de datos en Google Firestore. 36

Figura 3. 6 Pantalla de autenticación del prototipo StudyCards AI. 39

Figura 3. 7 Interfaz de carga de documentos y configuración de parámetros. 39

Figura 3. 8 Indicador de procesamiento durante la generación de tarjetas. 40

Figura 3. 9 Interfaz de visualización – vista de pregunta de la tarjeta. 41

Figura 3. 10 Interfaz de visualización – vista de respuesta de la tarjeta. 42

Figura 3. 11 Pantalla de historial de conjuntos de tarjetas generados. 42

CAPÍTULO I

DEFINICIÓN DEL PROYECTO DE TITULACIÓN

1.1 Tema

Implementación de un prototipo funcional basado en inteligencia artificial para la generación automática de tarjetas de estudio a partir de documentos digitales.

1.2 Justificación

En el contexto actual de la educación digital, los estudiantes universitarios trabajan cotidianamente con un volumen creciente de material académico en formato electrónico: guías, artículos, presentaciones y manuales que, en conjunto, configuran un repositorio personal de conocimiento que rara vez se convierte en material de repaso estructurado. Esta situación no responde a falta de disciplina, sino a un problema de tiempo y esfuerzo: transformar documentos extensos en tarjetas de estudio útiles es un proceso que demanda lectura, síntesis y formulación de preguntas, tareas que compiten directamente con el tiempo disponible para el estudio en sí. La sobrecarga informativa que caracteriza los entornos digitales de aprendizaje contemporáneos agudiza este problema, ya que la cantidad de información disponible supera con frecuencia la capacidad individual de procesamiento y organización (*Kirchner & De Bruyckere, 2017*).

Frente a este escenario, los avances en inteligencia artificial, y en particular en los modelos de lenguaje natural de gran escala, ofrecen una alternativa concreta para automatizar la transformación de documentos académicos en material de repaso. La investigación de *Zawacki-Richter et al. (2019)*, que revisó sistemáticamente 146 publicaciones sobre

aplicaciones de inteligencia artificial en educación superior, identificó precisamente en la generación automática de material de apoyo al estudio uno de los campos con mayor potencial de desarrollo y menor exploración empírica hasta la fecha. En esa misma línea, trabajos más recientes han documentado la capacidad de los modelos de lenguaje para generar preguntas académicamente pertinentes a partir de textos fuente con una calidad comparable a la producción humana entrenada (*Kasneci et al., 2023*).

La relevancia de las tarjetas de estudio como técnica de aprendizaje está ampliamente respaldada por la investigación en psicología cognitiva. El principio de recuperación activa establece que el intento de recordar información consolida su retención de manera significativamente más efectiva que la relectura pasiva del material (*Kornell & Bjork, 2022*). Sin embargo, la elaboración manual de estas tarjetas a partir de documentos extensos es precisamente la barrera que impide a muchos estudiantes beneficiarse de esta técnica. La propuesta presentada en este trabajo responde a esa brecha: desarrollar un prototipo que procese documentos digitales y produzca automáticamente tarjetas de estudio, reduciendo el costo de preparación y haciendo accesible una técnica cognitivamente superior.

El proyecto se justifica, en síntesis, desde tres dimensiones complementarias. Tecnológicamente, demuestra la viabilidad de integrar modelos de lenguaje preentrenados disponibles por API en una aplicación educativa funcional, sin requerir infraestructura de entrenamiento propia. Académicamente, contribuye al campo de la tecnología educativa al proporcionar evidencia empírica sobre el desempeño de estos sistemas en la generación de material de estudio. Y en términos prácticos, ofrece una herramienta concreta que reduce el tiempo requerido para preparar recursos de repaso y pone al alcance de cualquier estudiante los beneficios de la recuperación activa.

1.3 Planteamiento del Problema

En el entorno universitario contemporáneo, la gestión del conocimiento académico en formato digital se ha convertido en uno de los desafíos cotidianos más frecuentes de los estudiantes. El acceso a materiales de estudio ha dejado de ser un obstáculo, pero la capacidad de procesarlos, sintetizarlos y convertirlos en recursos útiles para el repaso continúa siendo, en gran medida, una tarea manual. Como señala *Floridi (2020)*, vivimos en un contexto de abundancia de datos y escasez de conocimiento procesado, una tensión que se manifiesta con especial intensidad en los períodos de evaluación, cuando la demanda de material sintetizado es mayor y el tiempo disponible para elaborarlo es menor.

Las herramientas de creación de tarjetas de estudio más utilizadas en la actualidad, Anki, Quizlet, Brainscape, comparten una limitación fundamental: dependen del ingreso manual del contenido por parte del usuario. El estudiante debe leer el documento fuente, identificar las ideas principales, formular preguntas pertinentes y registrarlas en la plataforma, uno a uno. Este proceso, aunque pedagógicamente valioso en sí mismo, es costoso en tiempo y se convierte en un obstáculo cuando el volumen de material es elevado. Ninguna de estas plataformas integra mecanismos de procesamiento automático de documentos que permitan transformar un archivo cargado en un conjunto de tarjetas listas para usar.

Esta brecha tecnológica constituye el problema central que motiva el presente trabajo. Los estudiantes disponen de material académico digital, pero carecen de herramientas que procesen ese contenido automáticamente y lo transformen en recursos de repaso coherentes y académicamente pertinentes. La consecuencia directa es que una parte significativa de ese

material nunca se convierte en objeto de práctica activa, lo que limita la eficiencia del proceso de estudio y el aprovechamiento de los recursos disponibles.

La solución propuesta aborda esta brecha mediante el desarrollo de un prototipo que integra extracción de texto desde documentos digitales, segmentación inteligente del contenido, generación de tarjetas mediante un modelo de lenguaje y validación automática de la calidad del material producido. El sistema está diseñado para reducir el tiempo que media entre tener un documento y disponer de tarjetas listas para estudiar, transformando un proceso que actualmente toma horas en uno que puede completarse en minutos.

1.4 Objetivos

1.4.1 Objetivo General

Desarrollar un prototipo funcional que aplique modelos de lenguaje natural preentrenados para la generación automática de tarjetas de estudio en formato pregunta–respuesta a partir de documentos digitales en formato PDF y DOCX, con el fin de apoyar el proceso de aprendizaje autónomo de estudiantes universitarios.

1.4.2 Objetivos Específicos

Para alcanzar el objetivo general, se definieron cinco objetivos específicos que articulan las distintas fases del trabajo. El primero consiste en analizar y extraer el contenido textual de documentos digitales en formato PDF y DOCX mediante librerías especializadas integradas en el backend del prototipo. El segundo objetivo es diseñar e implementar un pipeline de procesamiento documental que incluya normalización del texto, segmentación por fragmentos y construcción de instrucciones estructuradas para el modelo de lenguaje. El tercero implica seleccionar el modelo de lenguaje preentrenado más adecuado para la tarea, considerando criterios de calidad de salida, consistencia del formato, tiempo de respuesta y

facilidad de integración técnica. El cuarto objetivo es implementar un módulo de validación multicapa que filtre automáticamente las tarjetas generadas según criterios de completitud, longitud, duplicación y genericidad. Finalmente, el quinto objetivo es validar funcionalmente el prototipo mediante la revisión de la coherencia, relevancia y utilidad de las tarjetas producidas a partir de un conjunto de documentos de prueba.

1.5 Alcance

El proyecto contempla el diseño, implementación y validación técnica de un prototipo funcional orientado a la generación automática de tarjetas de estudio. El sistema recibe documentos en formato PDF o DOCX, extrae su contenido textual, lo procesa a través de un modelo de lenguaje consumido por API y almacena los resultados para su visualización interactiva. La implementación considera parámetros configurables por el usuario, cantidad de tarjetas y nivel de dificultad, alineados con los niveles cognitivos de la taxonomía revisada de Bloom (*Anderson et al., 2001*). El desarrollo se enmarca en una metodología incremental con principios ágiles, que permite avanzar por etapas funcionales y validar cada componente antes de integrar el siguiente (*Beck et al., 2001*).

En cuanto a lo que el proyecto no abarca, es importante aclarar que no se contempla el entrenamiento ni el ajuste fino de modelos de inteligencia artificial propios. El trabajo se apoya en modelos preentrenados disponibles por API, lo que corresponde al enfoque técnico más coherente con los tiempos y recursos del proyecto. Tampoco se incluye una evaluación pedagógica formal sobre el impacto de las tarjetas generadas en el rendimiento académico, ni funcionalidades avanzadas como análisis del aprendizaje, seguimiento de progreso o integración con plataformas institucionales. El resultado esperado es un prototipo funcional

que demuestre la viabilidad técnica de la propuesta y siente las bases para desarrollos futuros de mayor alcance.

CAPÍTULO II

MARCO TEÓRICO

El presente capítulo expone los fundamentos conceptuales y tecnológicos que sustentan el desarrollo de StudyCards AI. Su propósito no es únicamente contextualizar el trabajo dentro de la literatura existente, sino establecer el marco de referencia que justifica cada decisión de diseño adoptada en los capítulos posteriores. Se abordan, en primer lugar, los antecedentes del uso de inteligencia artificial en educación digital y la evidencia empírica sobre la efectividad de las tarjetas de estudio como técnica de recuperación activa. A continuación, se desarrollan los conceptos centrales de los modelos de lenguaje de gran escala y la arquitectura Transformer, seguidos de los principios de ingeniería de prompts que orientaron la construcción del prompt maestro del sistema. El capítulo continúa con la revisión de los enfoques de generación automática de preguntas, los criterios de calidad reconocidos en la literatura y las estrategias de segmentación de documentos. Finalmente, se presentan las tecnologías específicas seleccionadas para la implementación, cuya justificación técnica se retoma en el Capítulo III.

2.1 Inteligencia Artificial Aplicada a la Educación Digital

La inteligencia artificial puede entenderse, en términos amplios, como el conjunto de técnicas computacionales orientadas a reproducir capacidades cognitivas humanas mediante sistemas automatizados. En su formulación clásica, *Russell y Norvig (2022)* la definen a través del paradigma del agente racional: un sistema que percibe su entorno procesa esa información y actúa de manera que maximiza una medida de desempeño definida para la tarea que debe resolver. Esta definición, aunque abstracta, captura con precisión el papel que

la inteligencia artificial desempeña en el presente proyecto: analizar el contenido de un documento académico y generar, a partir de él, material de estudio que sea útil para el usuario.

En el campo educativo, la incorporación de sistemas inteligentes ha transformado progresivamente la forma en que se diseñan y distribuyen los contenidos de aprendizaje. Una revisión sistemática realizada por *Zawacki-Richter et al. (2019)* sobre 146 publicaciones en educación superior identificó como áreas de mayor desarrollo los sistemas de tutoría inteligente, la evaluación automatizada, el análisis del aprendizaje y la recomendación de contenidos. Al mismo tiempo, ese trabajo señaló una brecha notable en el desarrollo de herramientas orientadas al apoyo del estudio autónomo, especialmente en lo que respecta a la generación automática de material de repaso a partir de documentos existentes. Esa brecha es precisamente el espacio que el presente prototipo busca ocupar.

Más recientemente, *Kasneci et al. (2023)* analizaron el potencial específico de los grandes modelos de lenguaje en contextos educativos y concluyeron que su capacidad para generar preguntas, resúmenes y explicaciones adaptadas a partir de textos fuente los convierte en herramientas con un impacto práctico inmediato sobre la preparación de material didáctico. Esta conclusión fundamenta directamente el enfoque adoptado en el presente trabajo, que no busca evaluar el impacto pedagógico de las tarjetas generadas, sino demostrar la viabilidad técnica de un sistema capaz de producirlas de manera automática y coherente.

2.2 Modelos de Lenguaje Natural de Gran Escala

El procesamiento de lenguaje natural es la subdisciplina de la inteligencia artificial que estudia los métodos para que los sistemas computacionales analicen, comprendan y generen texto en lenguaje humano. Su evolución durante la última década ha estado marcada por el desplazamiento desde arquitecturas basadas en reglas y modelos estadísticos clásicos

hacia sistemas de aprendizaje profundo capaces de capturar relaciones semánticas complejas en secuencias de texto. El punto de inflexión de esta transición fue la introducción del modelo Transformer por *Vaswani et al. (2017)*, cuyo mecanismo de atención de múltiples cabezas permite procesar dependencias de largo alcance en una secuencia de forma paralela y eficiente, superando las limitaciones de cómputo secuencial que caracterizaban a las redes recurrentes anteriores.

Sobre esta base arquitectónica se desarrollaron los modelos de lenguaje de gran escala, conocidos por sus siglas en inglés como LLM. El trabajo de *Brown et al. (2020)* con GPT-3 demostró que un modelo entrenado con volumen suficiente de texto desarrolla capacidades de generalización que le permiten resolver tareas no vistas durante el entrenamiento a partir de instrucciones en lenguaje natural, sin necesidad de ajuste fino adicional. Este paradigma habilita el uso de LLM como motores de generación de contenido educativo mediante la simple formulación de instrucciones estructuradas, los llamados prompts, lo que elimina la necesidad de reentrenar el modelo para cada dominio o aplicación específica.

En el contexto del presente proyecto se utiliza Gemini, el modelo de lenguaje desarrollado por Google. Gemini está disponible mediante API y ofrece capacidades avanzadas de comprensión y generación de texto, incluyendo la producción de respuestas en formato estructurado como JSON, lo cual resulta fundamental para el procesamiento automático de los resultados dentro del pipeline del sistema. Su elección responde a criterios técnicos concretos que se detallan en el Capítulo III, entre los que destacan la calidad de la salida, la consistencia del formato de respuesta, la latencia y la disponibilidad de integración con el entorno de desarrollo utilizado.

2.3 Criterios de Selección de Modelos Preentrenados

La elección de un modelo de lenguaje preentrenado para una aplicación específica no debe responder únicamente al desempeño general del modelo en benchmarks estándar, sino a criterios técnicos directamente relacionados con los requerimientos del proyecto. Chang et al. (2024), en su revisión sobre evaluación de LLM, distinguen entre el desempeño en tareas genéricas y el desempeño en tareas de dominio, subrayando que la transferencia no siempre es directa y que la evaluación debe realizarse sobre la tarea específica de interés.

En el caso del presente prototipo, los criterios más relevantes son la calidad de las tarjetas generadas, entendida como la coherencia y pertinencia semántica de las preguntas y respuestas respecto al texto fuente; la consistencia del formato JSON de salida, que determina la fiabilidad del procesamiento automático de los resultados; el tiempo de respuesta, que influye en la experiencia del usuario durante la generación; y la facilidad de integración técnica, especialmente la disponibilidad de SDKs compatibles con el entorno .NET utilizado en el backend. Bommasani et al. (2021) argumentan que la selección de un modelo de fundación debe tratarse como una decisión de ingeniería de software, no solo como una decisión técnica de inteligencia artificial, lo que implica considerar el ecosistema de herramientas disponibles alrededor del modelo tanto como sus capacidades intrínsecas.

A estos criterios se suma la cohesión arquitectónica con el ecosistema en la nube adoptado por el proyecto. Cuando el modelo seleccionado pertenece al mismo proveedor que los servicios de autenticación y persistencia de datos, se reduce la complejidad operativa del sistema en su conjunto: se unifican las credenciales de acceso, se simplifica la gestión de cuotas y facturación, y se minimizan las dependencias externas que podrían introducir puntos de fallo adicionales. En este sentido, la integración nativa de Gemini con Firebase

Authentication y Firestore, servicios ya utilizados en la arquitectura de StudyCards AI, constituyó un factor determinante que complementó los resultados de la evaluación técnica comparativa presentada en la Tabla 3.3.

2.4 Ingeniería de Prompts

La ingeniería de prompts o *prompt engineering*, se define como el conjunto de técnicas para diseñar instrucciones textuales que orienten el comportamiento de un modelo de lenguaje hacia la producción de resultados específicos, coherentes y de alta calidad. A diferencia del ajuste fino del modelo, el prompt engineering opera exclusivamente sobre la entrada del sistema sin modificar los pesos internos del modelo, lo que lo convierte en la principal herramienta de control de comportamiento cuando el modelo se consume como servicio por API (*Liu et al., 2023*). Su importancia práctica radica en que pequeñas variaciones en la formulación de la instrucción pueden producir diferencias sustanciales en la calidad y la estructura de la respuesta generada.

En el contexto de la generación de tarjetas de estudio, un prompt bien diseñado debe especificar al modelo al menos cuatro elementos: el rol que debe adoptar, en este caso, el de un asistente educativo especializado, en síntesis, el fragmento de texto fuente sobre el que debe operar, los parámetros de la tarea, cantidad de tarjetas y nivel de dificultad, y el esquema exacto de la salida esperada. *White et al. (2023)* proponen un catálogo sistemático de patrones de prompt engineering que incluye, entre otros, el patrón de especificación de formato de salida, el patrón de rol del sistema y el patrón de restricciones explícitas; los tres se aplican en el diseño del prompt implementado en el presente prototipo.

La dificultad cognitiva de las tarjetas se traduce en directivas específicas al modelo sobre el tipo de operación que deben evaluar las preguntas. Esta jerarquía de exigencia se

fundamenta en la taxonomía revisada de Bloom (*Anderson et al., 2001*), que organiza los objetivos de aprendizaje en seis niveles, desde recordar y comprender hasta evaluar y crear, y que permite que el mismo documento genere material de distinta complejidad según el estado de avance del estudiante.

2.5 Generación Automática de Contenido Educativo

La generación automática de preguntas a partir de texto, denominada AQG en la literatura, es un campo de investigación dentro del procesamiento de lenguaje natural con décadas de historia. *Kurdi et al. (2020)*, en una revisión sistemática de 93 estudios publicados entre 2003 y 2019, documentaron la transición desde enfoques basados en transformación sintáctica hacia sistemas de aprendizaje profundo capaces de generar preguntas más naturales y contextualmente pertinentes. Con la llegada de los LLM, este campo ha experimentado un salto cualitativo: modelos como Gemini pueden generar preguntas de distintos niveles cognitivos a partir de un texto fuente con una calidad difícilmente distinguible de la producción humana en tareas de complejidad media.

El fundamento cognitivo que da sentido educativo a este tipo de herramientas es el efecto de la práctica de recuperación, también conocido como el efecto de prueba. La investigación acumulada en psicología cognitiva demuestra que el acto de intentar recordar información consolida su retención de manera significativamente más efectiva que la relectura del material (*Kornell & Bjork, 2022*). Las tarjetas en formato preguntan – respuesta son uno de los implementos más eficientes de este principio. Sin embargo, como señalan *Roll y Wylie (2016)*, la automatización de la generación de material de práctica es uno de los aportes de mayor impacto potencial de la inteligencia artificial a la educación, precisamente

porque elimina la barrera de tiempo y esfuerzo que impide a los estudiantes beneficiarse de estas técnicas.

Un aspecto crítico en la generación automática de contenido educativo es el control de la calidad del material producido. Los LLM pueden generar preguntas gramaticalmente correctas, pero semánticamente vagas, desconectadas del texto fuente o redundantes entre sí. Este fenómeno, conocido como alucinación, ha sido caracterizado en detalle por *Ji et al. (2023)*, quienes distinguen entre alucinaciones intrínsecas, cuando el modelo contradice información del texto de entrada, y extrínsecas, cuando introduce información no verificable desde el contexto. Para mitigar este riesgo, el presente prototipo incorpora un módulo de validación multicapa que actúa sobre las tarjetas generadas antes de su almacenamiento.

2.6 Arquitectura de Software para Sistemas con Inteligencia Artificial

El diseño arquitectónico de una aplicación que integra inteligencia artificial no se limita a definir cómo se organizan sus componentes, sino que debe contemplar también los patrones de integración con servicios externos, los flujos de datos y las garantías de calidad y seguridad del sistema resultante. *Bass et al. (2022)* definen la arquitectura de software como el conjunto de decisiones de diseño que mayor impacto tienen sobre las propiedades sistémicas de una aplicación, su escalabilidad, su mantenibilidad y su tolerancia a fallos, subrayando que estas decisiones son, en gran medida, irreversibles y deben tomarse con información suficiente sobre los requerimientos del sistema.

El prototipo adoptó una arquitectura en capas con servicios externos desacoplados. Este patrón organiza el sistema en niveles de responsabilidad bien delimitados: la capa de presentación gestiona la interacción con el usuario; la capa de lógica de negocio coordina el procesamiento de las solicitudes; la capa de integración gestiona la comunicación con

Firebase, la API de Gemini y las librerías de extracción de texto; y la capa de persistencia administra el almacenamiento en Firestore y el sistema de archivos local. La separación estricta entre estas capas garantiza que los cambios en una de ellas no propaguen efectos no deseados en las demás (*Martin, 2017*). La autenticación sigue el estándar OAuth 2.0 (*Hardt, 2012*), con Firebase como proveedor de identidad, lo que delega la gestión de credenciales a un servicio especializado y reduce la superficie de ataque del sistema.

2.7 Procesamiento de Documentos Digitales y Segmentación de Texto

El procesamiento de documentos digitales comprende las operaciones necesarias para extraer, limpiar y preparar el contenido textual de archivos electrónicos antes de enviarlo al modelo de lenguaje. Esta etapa es crítica porque la calidad del texto que recibe el modelo determina en gran medida la calidad del contenido que genera: documentos con ruido, caracteres especiales, saltos de línea inconsistentes, encabezados y pies de página intercalados, pueden degradar el desempeño del sistema y aumentar la tasa de respuestas poco pertinentes. Para los documentos en formato PDF, el prototipo utiliza la librería PdfPig, que recupera el flujo de texto digital manteniendo el orden de lectura. Para los documentos DOCX, se utiliza el SDK DocumentFormat.OpenXml, que accede directamente a los elementos estructurales del documento, párrafos, encabezados, tablas, con plena fidelidad al contenido.

Un aspecto técnico de especial relevancia en este proceso es la segmentación del texto en fragmentos de tamaño controlado, técnica conocida como *chunking*. Los modelos de lenguaje tienen una ventana de contexto finita, medida en tokens, que delimita el volumen máximo de texto que pueden procesar en una sola llamada. Enviar un documento completo que supere ese límite produce errores o resultados truncados, mientras que fragmentar el texto

de manera inadecuada, cortando párrafos a la mitad, puede generar tarjetas con contexto incompleto. Por ello, *Gao et al. (2023)* recomiendan segmentar el texto respetando límites semánticos naturales, párrafos o secciones, con un tamaño máximo que deje margen suficiente para el prompt y la respuesta esperada dentro de la ventana de contexto disponible.

2.8 Personalización del Material de Estudio

La personalización del aprendizaje mediante tecnología se refiere a la capacidad de adaptar los recursos educativos en función de las necesidades o preferencias de cada usuario. En sistemas basados en inteligencia artificial, esta personalización puede implementarse sin perfiles de usuario complejos: basta con exponer parámetros configurables que permitan ajustar características del material generado. *Pane et al. (2017)* encontraron, en una evaluación a gran escala de programas de aprendizaje personalizado, que la posibilidad de controlar el ritmo y la dificultad mejora tanto la motivación como el desempeño de los estudiantes.

En el presente prototipo, la personalización se implementa a través de dos parámetros: la cantidad de tarjetas deseadas y el nivel de dificultad. Este último se traduce en directivas cognitivas concretas dentro del prompt, orientadas a los distintos niveles de la taxonomía de Bloom (*Anderson et al., 2001*). El resultado es que el mismo documento puede generar materiales de distinta exigencia cognitiva según la etapa de estudio del usuario, desde preguntas de reconocimiento y comprensión en el nivel básico, hasta preguntas de análisis, evaluación y propuesta en el nivel avanzado.

2.9 Métricas de Evaluación de Calidad en Generación de Texto

La evaluación de la calidad del texto generado por modelos de lenguaje es un área activa de investigación. Las métricas más utilizadas en tareas de generación, BLEU,

ROUGE, METEOR, miden la similitud entre el texto producido y textos de referencia elaborados por humanos (*Liang et al., 2022*). Sin embargo, estas métricas presentan limitaciones cuando se aplican a la evaluación de tarjetas de estudio: una pregunta reformulada creativamente puede tener baja similitud lexical con una referencia y ser, aun así, académicamente superior.

Para la evaluación de sistemas de generación automática de preguntas, *Kurdi et al. (2020)* identifican cuatro dimensiones que los evaluadores humanos priorizan de manera consistente: la relevancia, correspondencia semántica de la pregunta con el texto fuente, la claridad, comprensibilidad de la formulación, la coherencia, consistencia entre pregunta y respuesta, y la no redundancia, ausencia de preguntas equivalentes dentro del mismo conjunto. En cuanto al control de alucinaciones, *Ji et al. (2023)* proponen verificar que los conceptos mencionados en las respuestas generadas estén efectivamente anclados en el texto de entrada, práctica que el módulo de validación del prototipo implementa de forma parcial mediante los filtros de calidad descritos en el Capítulo III.

Estas cuatro dimensiones relevancia, claridad, coherencia y no redundancia constituyen los criterios con los que se evaluará funcionalmente el desempeño del prototipo en el Capítulo IV, permitiendo determinar de manera fundamentada si las tarjetas generadas automáticamente satisfacen estándares académicamente aceptables para su uso como material de repaso.

2.10 Relación del Marco Teórico con la Propuesta

Los fundamentos presentados en este capítulo articulan una cadena conceptual coherente que da soporte teórico a cada componente del prototipo. La inteligencia artificial aplicada a la educación digital justifica el propósito general del sistema y lo sitúa en un campo de investigación activo con necesidades concretas aún no resueltas. Los modelos de lenguaje natural de gran escala, y en particular Gemini, son el motor tecnológico que hace posible la automatización de la generación de tarjetas. Los criterios de selección de modelos orientaron la decisión técnica de integración, mientras que la ingeniería de prompts define la interfaz de control entre el sistema y el modelo, determinando la calidad y estructura del contenido producido.

La generación automática de contenido educativo y su respaldo empírico en la psicología cognitiva fundamentan la utilidad del producto resultante. La arquitectura de software en capas organiza la implementación en niveles de responsabilidad claramente delimitados, y el procesamiento documental, incluida la segmentación por chunks, garantiza que el texto llega al modelo en condiciones óptimas. La personalización mediante parámetros responde a las necesidades individuales de los usuarios, y las métricas de calidad proporcionan los criterios con que se valorará el desempeño del prototipo. En conjunto, estos fundamentos forman el andamiaje conceptual que sostiene y justifica cada decisión de diseño e implementación documentada en el capítulo siguiente.

CAPÍTULO III

DESARROLLO DE LA PROPUESTA

El presente capítulo describe en detalle la construcción del prototipo StudyCards AI, articulando las decisiones de diseño con los fundamentos teóricos establecidos en el Capítulo II. Se inicia con la definición de la metodología de desarrollo adoptada, seguida del análisis de requerimientos funcionales y no funcionales que delimitaron el alcance del sistema. A partir de ahí se presenta la arquitectura en capas propuesta, con la descripción de cada componente tecnológico y su rol dentro del sistema. Los apartados siguientes detallan el pipeline de procesamiento de documentos, desde la extracción y normalización del texto hasta la segmentación semántica por chunks, y el diseño del prompt maestro con sus componentes, restricciones y configuración de temperatura. El capítulo cierra con la descripción del módulo de validación multicapa, que opera sobre la salida del modelo antes del almacenamiento, y con los diagramas de flujo y casos de uso que sintetizan el comportamiento completo del prototipo.

3.1 Metodología de Desarrollo

El desarrollo del prototipo se llevó a cabo bajo un modelo incremental con principios de metodología ágil. Este enfoque fue elegido deliberadamente porque la integración de modelos de lenguaje en una aplicación web introduce un grado de incertidumbre técnica que los modelos de desarrollo secuencial gestionan con dificultad: el comportamiento del modelo ante distintos tipos de documentos, el formato de las respuestas y la latencia de las llamadas a la API son variables que solo pueden evaluarse empíricamente, durante la construcción del sistema (*Sommerville, 2016*). La metodología incremental permite avanzar por etapas

funcionales, validar cada componente antes de integrar el siguiente y ajustar el diseño en función de los resultados observados.

El trabajo se organizó en cuatro fases operativas sucesivas. En la primera fase se configuró la infraestructura base del proyecto, el proyecto ASP.NET Core 8.0, la estructura de capas, los endpoints de la API REST, y se implementó el flujo completo de autenticación con Firebase y Google OAuth 2.0, incluyendo la emisión de tokens JWT internos para las solicitudes subsiguientes. En la segunda fase se desarrollaron los servicios de carga y procesamiento documental: los endpoints de recepción de archivos, los módulos de extracción de texto para PDF y DOCX, la normalización del contenido y la segmentación en chunks. En la tercera fase se integró la API de Gemini: diseño del prompt, pruebas de comportamiento con distintos tipos de documentos, implementación del módulo de validación y limpieza de tarjetas. La cuarta fase completó el sistema con la integración de Google Firestore para el almacenamiento persistente, el desarrollo del frontend en Vue 3 con TypeScript y las pruebas funcionales de integración end-to-end.

3.2 Análisis de Requerimientos

El análisis de requerimientos partió de la definición del problema establecida en el Capítulo I y se orientó a identificar las funcionalidades mínimas necesarias para que el prototipo cumpliera su propósito educativo. Se siguió un enfoque centrado en el usuario, coherente con las prácticas de ingeniería de requisitos descritas por *Sommerville (2016)*, que distingue entre requerimientos funcionales, lo que el sistema debe hacer, y no funcionales, las propiedades con que debe hacerlo.

Tabla 3. 1

Requerimientos funcionales del sistema

ID	Requerimiento	Descripción	Prioridad
RF-01	Autenticación con Google	El sistema permite iniciar sesión mediante cuenta Google a través de Firebase Authentication. El backend valida el token y emite un JWT interno para las solicitudes siguientes.	Alta
RF-02	Carga de documentos	El sistema acepta archivos PDF y DOCX. Valida tipo MIME y tamaño máximo en cliente y servidor antes de procesar.	Alta
RF-03	Configuración de parámetros	El usuario define el título del conjunto, la cantidad de tarjetas (5,10,15, etc.) y el nivel de dificultad (básico, intermedio o avanzado).	Alta
RF-04	Generación automática	El sistema extrae el texto, segmenta, construye el prompt y llama a la API de Gemini. Devuelve tarjetas en formato pregunta–respuesta.	Alta

RF-05	Validación y limpieza	Aplica filtros automáticos para eliminar tarjetas vacías, de longitud insuficiente, duplicadas o genéricas antes del almacenamiento.	Media
RF-06	Visualización flashcard	El usuario puede ver sus tarjetas en una interfaz interactiva que muestra la pregunta y revela la respuesta al hacer clic.	Alta
RF-07	Historial de conjuntos	El usuario consulta los conjuntos generados previamente, ordenados por fecha de creación.	Media

Nota. Los requerimientos fueron definidos siguiendo el enfoque de Sommerville (2016), que distingue entre lo que el sistema debe hacer y las propiedades con que debe hacerlo. RF = Requerimiento Funciona

Tabla 3. 2

Requerimientos no funcionales del sistema StudyCards AI

ID	Requerimiento	Descripción	Prioridad
RNF-01	Seguridad	Todo endpoint requiere JWT válido. Acceso no autenticado se rechaza con HTTP 401. Comunicación por HTTPS.	Alta

RNF-02	Aislamiento de datos	Cada usuario accede únicamente a sus propios conjuntos. El uid del usuario actúa como clave de partición en Firestore.	Alta
RNF-03	Mantenibilidad	Código organizado en capas: Controllers, Services, Models. Separación de responsabilidades según principios de Clean Code (Martin, 2017).	Media
RNF-04	Escalabilidad	La arquitectura en capas y los servicios en la nube permiten escalar componentes de forma independiente.	Media

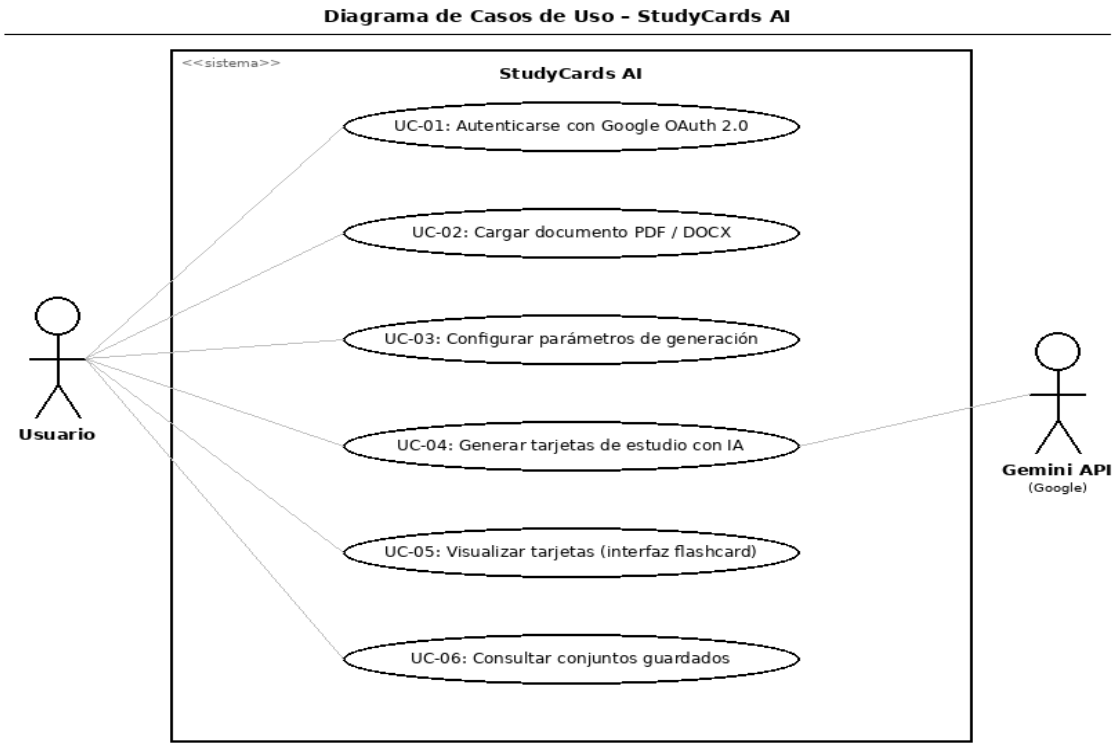
Nota. RNF = Requerimiento No Funcional. Los criterios de seguridad y aislamiento de datos fueron priorizados en coherencia con los principios de Privacy by Design aplicados en el diseño del sistema.

3.3 Diagrama de Casos de Uso

El diagrama de casos de uso describe los actores del sistema y las funcionalidades disponibles para cada uno. Se identificaron dos actores: el Usuario, que interactúa con la plataforma a través del frontend en Vue 3, y la Gemini API, servicio externo de Google que participa únicamente en el caso de uso UC-04. Los seis casos de uso definidos recogen el flujo completo de la aplicación, desde la autenticación inicial hasta la consulta del historial de conjuntos generados.

Figura 3. 1

Diagrama de casos de uso del sistema StudyCards AI.



Nota. Se muestran las principales funcionalidades del sistema y la interacción del usuario con cada una de ellas.

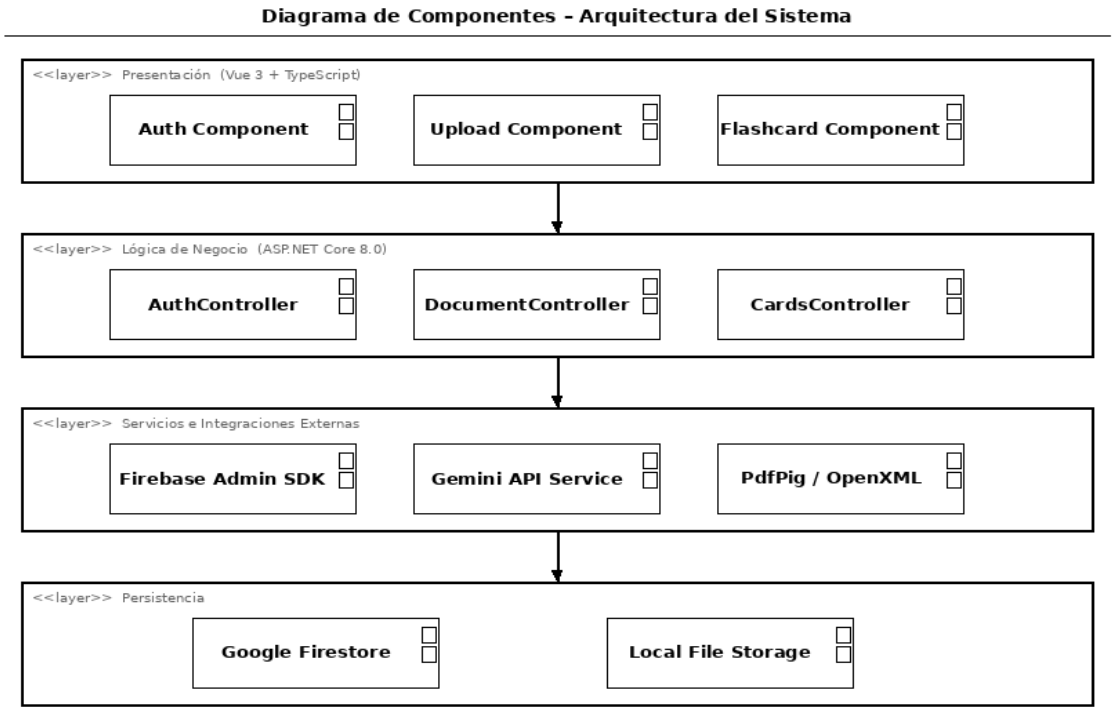
Los seis casos de uso modelados son: UC-01, autenticarse con Google OAuth 2.0 mediante Firebase Authentication; UC-02, cargar un documento en formato PDF o DOCX con validación de formato en cliente y servidor; UC-03, configurar los parámetros de generación ,título del conjunto, cantidad de tarjetas y nivel de dificultad; UC-04, generar tarjetas de estudio, que involucra la llamada a la API de Gemini mediante el prompt estructurado; UC-05, visualizar las tarjetas generadas en la interfaz flashcard; y UC-06, consultar los conjuntos previamente creados desde Firestore.

3.4 Arquitectura del Sistema

El prototipo sigue una arquitectura en capas con separación estricta entre el frontend y el backend, comunicados a través de una API REST sobre HTTPS. Esta decisión garantiza la independencia de los componentes y permite modificar o reemplazar cualquier capa sin afectar a las demás (*Bass et al., 2022*). El diagrama de componentes muestra la organización de los módulos del sistema en sus cuatro niveles principales.

Figura 3. 2

Diagrama de componentes – arquitectura por capas del sistema.



Nota. Se muestra la arquitectura en capas del sistema, incluyendo presentación, lógica de negocio, servicios externos y persistencia de datos.

La capa de presentación está implementada en Vue 3 con TypeScript y agrupa tres componentes principales: el componente de autenticación, que gestiona el flujo de inicio de

sesión con Google; el componente de carga, que permite al usuario seleccionar el archivo y configurar los parámetros; y el componente de visualización, que presenta las tarjetas generadas en la interfaz flashcard. La comunicación entre el frontend y el backend se realiza exclusivamente mediante solicitudes HTTP/HTTPS hacia los endpoints de la API REST, lo que garantiza un desacoplamiento total entre la interfaz de usuario y la lógica del servidor: cualquier cambio en la capa de presentación no afecta al backend, y viceversa.

La capa de lógica de negocio en ASP.NET Core 8.0 contiene los controladores que reciben las solicitudes REST y coordinan el procesamiento. La capa de servicios gestiona la comunicación con tres sistemas externos: Firebase Admin SDK para la validación de tokens de identidad, la API de Gemini para la generación de tarjetas mediante lenguaje natural, y las librerías PdfPig y DocumentFormat.OpenXml para la extracción de texto según el formato del documento recibido. Este nivel de integración con servicios externos es el núcleo técnico diferenciador del sistema, ya que orquesta en una sola solicitud del usuario la autenticación, el procesamiento documental y la generación de contenido con inteligencia artificial.

La capa de persistencia almacena los conjuntos y tarjetas en Google Firestore y los archivos originales en el sistema de archivos del servidor. El flujo completo que recorre un documento desde que el usuario lo carga hasta que se convierte en tarjetas puede describirse en cuatro pasos lógicos: (1) el frontend envía el archivo y los parámetros al backend mediante una solicitud multipart autenticada con JWT; (2) el backend extrae el texto, lo segmenta en chunks y construye el prompt estructurado; (3) el backend invoca la API de Gemini y recibe las tarjetas en formato JSON; y (4) el módulo de validación filtra el resultado y lo persiste en Firestore, desde donde el frontend lo recupera para su visualización. Esta trayectoria deja en

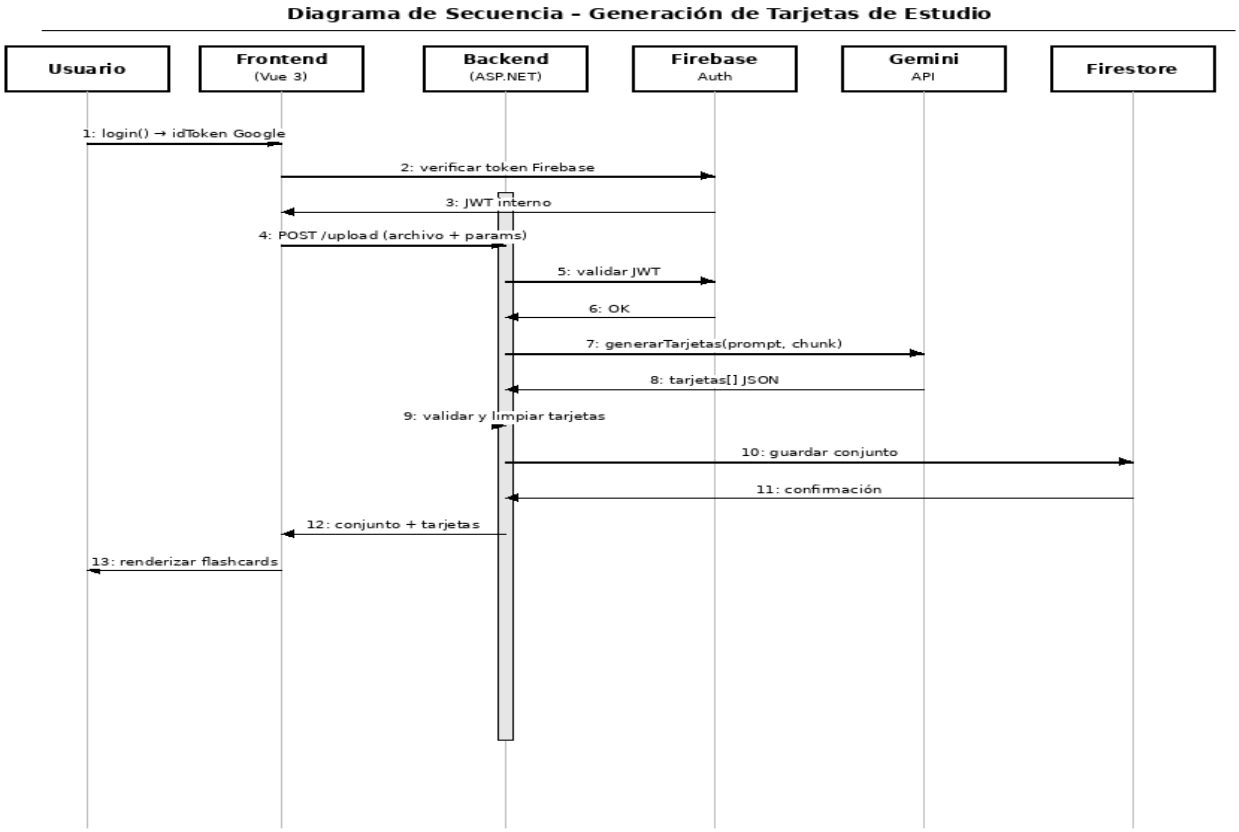
evidencia que el backend actúa como orquestador central del sistema, coordinando todos los servicios externos sin exponer ninguno de ellos directamente al cliente.

3.5 Diagrama de Secuencia

El diagrama de secuencia describe la interacción dinámica entre los componentes del sistema durante el flujo principal, desde que el usuario inicia sesión hasta que las tarjetas son presentadas en la interfaz. Este diagrama es especialmente relevante en proyectos que integran múltiples servicios externos, porque permite visualizar los puntos de integración, las dependencias temporales entre componentes y los posibles puntos de fallo (*Sommerville, 2016*).

Figura 3. 3

Diagrama de secuencia – flujo de generación de tarjetas de estudio.



Nota. El diagrama representa el intercambio de mensajes entre los componentes del sistema desde la autenticación del usuario hasta la visualización de las tarjetas generadas.

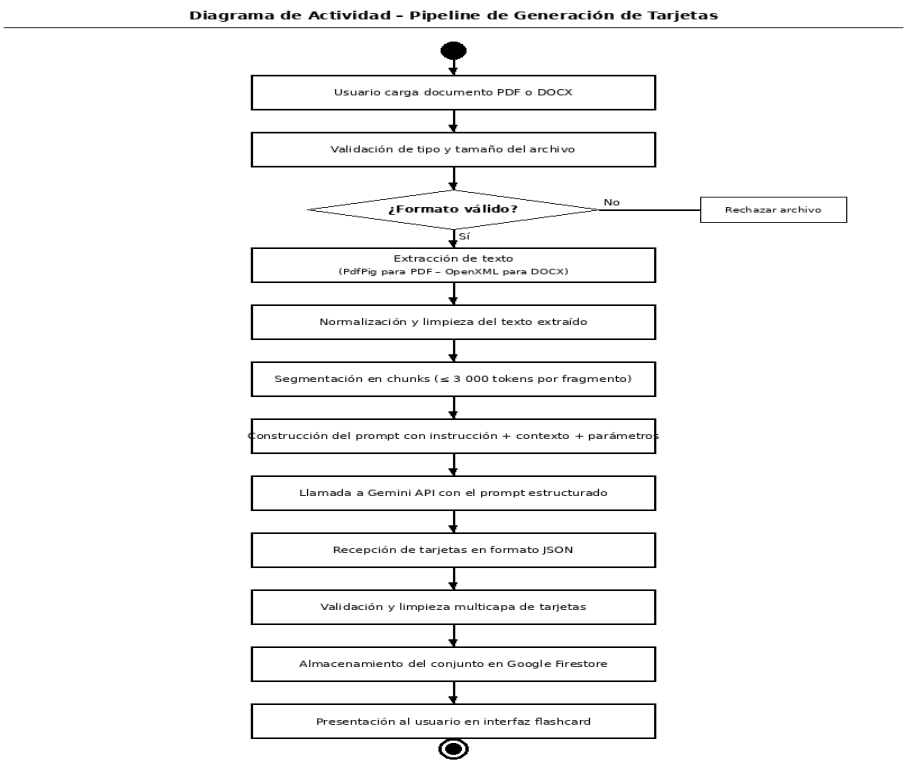
El flujo inicia con la autenticación del usuario mediante Google OAuth 2.0. Firebase devuelve el token de identidad al frontend, que lo envía al backend para su verificación y la emisión del JWT interno. Con la sesión establecida, el usuario carga el documento y configura los parámetros; el frontend los envía al backend mediante una solicitud POST con formato multipar. El backend verifica el JWT, extrae el texto del documento, lo segmenta y construye el prompt. Realiza la llamada a la API de Gemini y recibe las tarjetas en formato JSON. El módulo de validación procesa las tarjetas, el conjunto validado se almacena en Firestore y, finalmente, el frontend recupera y presenta los resultados al usuario.

3.6 Pipeline de Procesamiento e Inteligencia Artificial

El pipeline de procesamiento es el núcleo técnico del prototipo. Describe la secuencia de operaciones que el backend ejecuta desde la recepción del archivo hasta el almacenamiento de las tarjetas validadas, integrando extracción documental, normalización, segmentación, generación mediante IA y control de calidad. El diagrama de actividad siguiente muestra esta secuencia con sus puntos de decisión.

Figura 3. 4

Diagrama de actividad – pipeline de generación de tarjetas



Nota. El diagrama representa las etapas principales del pipeline de procesamiento documental y generación automática de tarjetas de estudio.

El proceso comienza cuando el usuario envía el archivo desde el frontend. El backend determina el tipo de archivo y aplica la validación de formato y tamaño: si el archivo no cumple los criterios, se rechaza con un mensaje de error descriptivo. Si es válido, el servicio de extracción de texto aplica PdfPig para documentos PDF o DocumentFormat.OpenXml para DOCX. El texto extraído se normaliza, eliminando caracteres no imprimibles y espacios redundantes, y se segmenta en fragmentos mediante una estrategia de chunking por párrafos con límite de tokens. El algoritmo recorre el texto dividido por párrafos e incorpora cada párrafo al chunk activo mientras la suma acumulada de tokens no supere los 3 000. Cuando agregar un nuevo párrafo excedería ese límite, el chunk activo se cierra y se abre uno nuevo. Este umbral se definió considerando que el modelo Gemini 1.5 Pro dispone de una ventana de contexto amplia, pero que el prompt y la respuesta esperada consumen tokens adicionales:

reservar 3 000 tokens para el fragmento de texto deja margen suficiente para la instrucción completa y el JSON de respuesta sin acercarse al límite operativo. La unidad de segmentación es el párrafo, no la oración ni la página, porque el párrafo constituye la unidad semántica mínima que preserva el contexto completo de una idea; cortar una idea a la mitad produciría tarjetas con contexto incompleto o preguntas sin respaldo textual suficiente (*Gao et al., 2023*).

3.7 Selección del Modelo de Lenguaje

Previo a la implementación definitiva del servicio de generación, se realizó una evaluación técnica comparativa entre los modelos de lenguaje disponibles mediante API y compatibles con el entorno .NET. Los modelos considerados fueron GPT-4o de OpenAI y Gemini 1.5 Pro de Google. La evaluación utilizó cinco documentos de prueba de distintos dominios académicos y analizó la calidad de las tarjetas generadas con prompts estandarizados, siguiendo los criterios descritos por *Chang et al. (2024)*.

Tabla 3. 3

Comparación técnica de modelos de lenguaje evaluados.

Criterio	GPT-4o (OpenAI)	Claude 3.5 Sonnet (Anthropic)	Llama 3 / Mistral (Código abierto / Cloud)	Gemini 1.5 Pro (Google)
Calidad, coherencia y consistencia	Alta. Excelente control de instrucciones y generación de respuestas coherentes.	Muy alta. Destaca por su comprensión semántica y precisión en tareas complejas.	Media-Alta. Su rendimiento depende del proveedor de infraestructura y la configuración utilizada.	Alta. Genera respuestas detalladas con un sólido manejo del contexto.
Soporte para .NET y ecosistema	SDK oficial estable y compatibilidad con Semantic Kernel.	Integración mediante API HTTP y compatibilidad con Semantic Kernel.	Depende de la plataforma o proveedor donde se despliegue el modelo.	SDK oficial para .NET con integración al ecosistema de Google Cloud y Firebase.

Estrategia de autenticación e integración	Uso de claves de API administradas desde el backend.	Autenticación mediante API Key y encabezados HTTP.	Requiere implementar autenticación e infraestructura según el proveedor o despliegue propio.	Integración mediante IAM (Identity and Access Management) y Firebase Admin SDK para .NET.
Acoplamiento con la persistencia de datos	Arquitectura desacoplada: Aplicación → Backend → Base de datos externa.	Arquitectura desacoplada: Aplicación → Backend → Base de datos externa.	Requiere mayor configuración para integrar base de datos y servicios.	Integración directa y sencilla con Firestore y servicios de Firebase.
Costo operativo (aprox. por 1.000 tokens)	USD 0,0025	USD 0,0030	Variable según infraestructura y proveedor.	USD 0,00125 (una de las opciones más económicas).

Nota. La comparación se realizó con base en 20 llamadas de prueba por modelo bajo condiciones equivalentes de red y longitud de chunk.

Como se evidencia en la evaluación expandida, la selección de Gemini 1.5 Pro obedece a un criterio de mínima complejidad operativa y máxima sinergia de componentes (Architectural Cohesion). El prototipo StudyCards AI fundamenta su seguridad en *Firebase Authentication* (RF-01) y su persistencia en base de datos documental NoSQL en *Google*

Firestore (RF-07). Al consumir la API de Gemini dentro del mismo entorno de nube de Google, se mitigan problemas latentes de integración que afectan a características esenciales del sistema:

- **Calidad y Consistencia**: La ventana de contexto nativa y el SDK oficial para .NET garantizan que las directivas del prompt maestro se transmitan sin capas intermedias de traducción, manteniendo un control estricto sobre las respuestas en formato JSON mediante el parseo defensivo implementado en el backend.
- **Tiempo de respuesta y Latencia de Red:** Al interactuar servicios hospedados en la misma red de infraestructura global de Google, se minimiza el *network hop* (salto de red) que ocurriría si el backend tuviera que autenticarse y transmitir cargas útiles de texto hacia nubes externas (como OpenAI o Anthropic) para luego regresar y persistir en Firestore. Esto compensa la diferencia en la latencia intrínseca del modelo.
- **Seguridad y Gestión de Identidades:** La integración intra-ecosistema permite heredar las políticas de seguridad y los identificadores únicos (*uid*) de usuario provistos por Firebase directamente en la lógica del pipeline, facilitando el aislamiento estricto de datos (RNF-02) sin necesidad de mapear esquemas de tokens multiplataforma complejos.

Por lo tanto, el costo operativo favorable de Gemini no se priorizó de manera aislada, sino como un beneficio secundario derivado de una arquitectura optimizada que reduce la deuda técnica garantiza la mantenibilidad del código (RNF-03) y asegura la viabilidad del pipeline end-to-end dentro de un entorno unificado de servicios en la nube."

3.8 Diseño del Prompt de Generación

El prompt implementado en el prototipo sigue la estructura de cuatro componentes descrita en el marco teórico. Su diseño fue iterativo: las primeras versiones producían tarjetas demasiado genéricas o desconectadas del texto fuente, lo que motivó la incorporación progresiva de restricciones de calidad y la especificación más precisa del formato de salida. La versión final del prompt establece, en primer lugar, el rol del modelo como asistente educativo especializado en la generación de material de estudio universitario, con énfasis en la fidelidad al texto fuente y la claridad académica. A continuación, incluye el fragmento de texto, delimitado por marcadores explícitos que permiten al modelo distinguirlo de las instrucciones. Luego especifica la cantidad de tarjetas y traduce el nivel de dificultad en directivas cognitivas concretas: preguntas de reconocimiento y comprensión para el nivel básico, de aplicación y análisis para el intermedio, y de evaluación, justificación o propuesta para el avanzado, en correspondencia con la taxonomía de *Anderson et al. (2001)*. Finalmente, exige una respuesta exclusivamente en formato JSON, un array de objetos con los campos "pregunta" y "respuesta", sin texto introductorio ni explicativo adicional.

Los parámetros de temperatura del modelo se configuran en un valor bajo, cercano a 0,3 para favorecer respuestas deterministas y reducir la variabilidad no deseada en el contenido educativo. Las instrucciones de calidad incluidas en el prompt, no generar preguntas triviales, no repetir preguntas equivalentes, asegurar el anclaje al texto fuente, actúan como la primera línea de control de calidad del sistema, antes incluso de que el módulo de validación entre en acción (*White et al., 2023*).

La siguiente tabla sintetiza la estructura del prompt maestro implementado, detallando cada componente, su contenido concreto y las restricciones impuestas al modelo:

Tabla 3. 4

Estructura y componentes del prompt maestro de generación de tarjetas

Componente	**Contenido**	**Restricciones impuestas al modelo**
Rol del sistema	Asistente educativo especializado en generación de material de estudio universitario	Responder únicamente en base al texto fuente proporcionado; no introducir información externa
Texto fuente	Fragmento delimitado por marcadores explícitos ([INICIO_TEXTO] / [FIN_TEXTO])	No usar información fuera de los marcadores
Parámetros de tarea	Cantidad de tarjetas solicitadas y nivel de dificultad (básico / intermedio / avanzado)	No exceder la cantidad indicada; aplicar el nivel cognitivo especificado según la taxonomía de Bloom
Restricciones de calidad	Directivas explícitas de no redundancia y especificidad	No generar preguntas triviales ni equivalentes entre sí; cada pregunta debe anclar a un concepto concreto del texto
Formato de salida	JSON estricto: array de objetos con campos "pregunta" y "respuesta"	Sin texto introductorio, sin explicaciones adicionales, sin

		bloques de código Markdown; únicamente el array JSON
Temperatura	0.3	Favorecer respuestas deterministas y reducir variabilidad no deseada

Nota. El diseño del prompt sigue los patrones de ingeniería de prompts descritos por White et al. (2023), incorporando rol, delimitadores de contexto, parámetros de tarea y restricciones de formato explícitas.

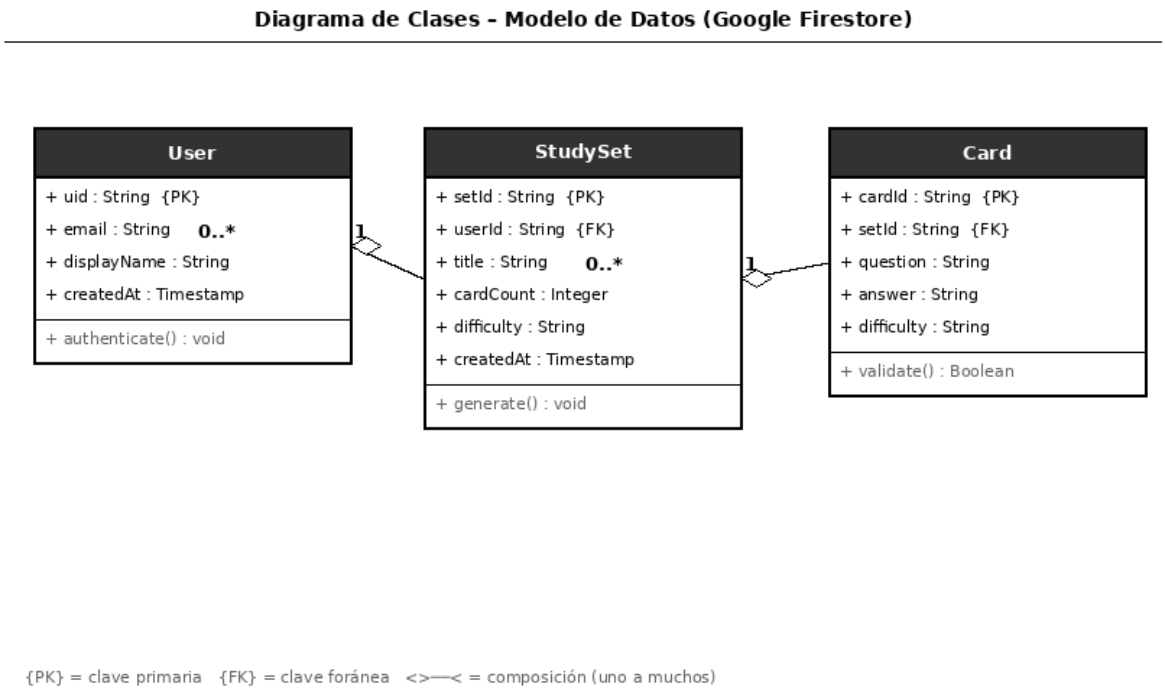
El cumplimiento del formato JSON es la restricción más crítica desde el punto de vista del pipeline, porque una respuesta malformada interrumpe el procesamiento automático. Por ello, además de exigirlo en el prompt, el backend implementa una capa de parseo defensivo que intenta recuperar el JSON válido incluso ante variaciones menores en la respuesta del modelo.

3.9 Modelo de Datos

El almacenamiento de los resultados utiliza Google Firestore, una base de datos NoSQL orientada a documentos que se integra de forma nativa con Firebase Authentication, lo que simplifica la implementación de reglas de seguridad basadas en la identidad del usuario (*Google, 2024*). Su escalabilidad automática y su baja latencia para lecturas de documentos individuales la hacen apropiada para la carga de trabajo del prototipo.

Figura 3. 5

Diagrama de clases – modelo de datos en Google Firestore.



Nota. El diagrama muestra las entidades principales del sistema y sus relaciones dentro de la base de datos Firestore.

El modelo de datos se organiza en tres entidades principales. La entidad User almacena la información de cada usuario autenticado: su identificador único provisto por Firebase, el correo electrónico, el nombre de visualización y la marca temporal de creación. La entidad StudySet registra cada conjunto de tarjetas generado, vinculado al usuario por su uid, con el título, la cantidad de tarjetas, el nivel de dificultad y la fecha de creación. La entidad Card contiene las tarjetas individuales, cada una vinculada a su conjunto por el identificador del conjunto, con los campos pregunta, respuesta y dificultad. Las relaciones son de uno a muchos en ambos casos: un usuario puede tener múltiples conjuntos, y un conjunto puede contener múltiples tarjetas.

3.10 Módulo de Validación y Limpieza de Tarjetas

El módulo de validación multicapa actúa sobre las tarjetas producidas por Gemini antes de su almacenamiento en Firestore. Su diseño responde a la necesidad de controlar la calidad del contenido generado automáticamente, dado que incluso los mejores modelos de lenguaje producen ocasionalmente tarjetas vacías, repetidas o de bajo valor académico (*Kurdi et al., 2020*).

El módulo aplica cuatro filtros en secuencia. El primero verifica la completitud de cada tarjeta: descarta aquellas cuyo campo pregunta o respuesta sea nulo, vacío o contenga únicamente espacios en blanco, capturando errores de parseo del JSON o respuestas parciales del modelo. El segundo filtro controla la longitud mínima: elimina respuestas con menos de veinte caracteres, consideradas insuficientes como recurso de estudio. El tercero detecta duplicados mediante la normalización del texto de cada pregunta, conversión a minúsculas, eliminación de puntuación, y la comparación entre todas las preguntas del conjunto. El cuarto filtro identifica preguntas genéricas comparando su formulación contra un conjunto de patrones predefinidos que caracterizan preguntas sin anclaje específico al dominio del documento.

Tabla 3. 5

Filtros del módulo de validación y limpieza de tarjetas.

Filtro	**Criterio de descarte**	**Objetivo de calidad**
Completitud	Campo pregunta o respuesta ausente, nulo o vacío.	Garantizar integridad estructural de cada tarjeta.
Longitud mínima	Respuesta con menos de 20 caracteres.	Eliminar respuestas triviales sin valor explicativo.

Duplicados	Pregunta normalizada idéntica a una ya aceptada.	Evitar redundancia dentro del conjunto generado.
Genericidad	Coincidencia con patrones de preguntas sin especificidad.	Mejorar pertinencia académica del material producido.

Nota. El diseño del prompt sigue los patrones de ingeniería de prompts descritos por White et al. (2023), incorporando rol, delimitadores de contexto, parámetros de tarea y restricciones de formato explícitas.

3.11 Implementación de la Interfaz de Usuario

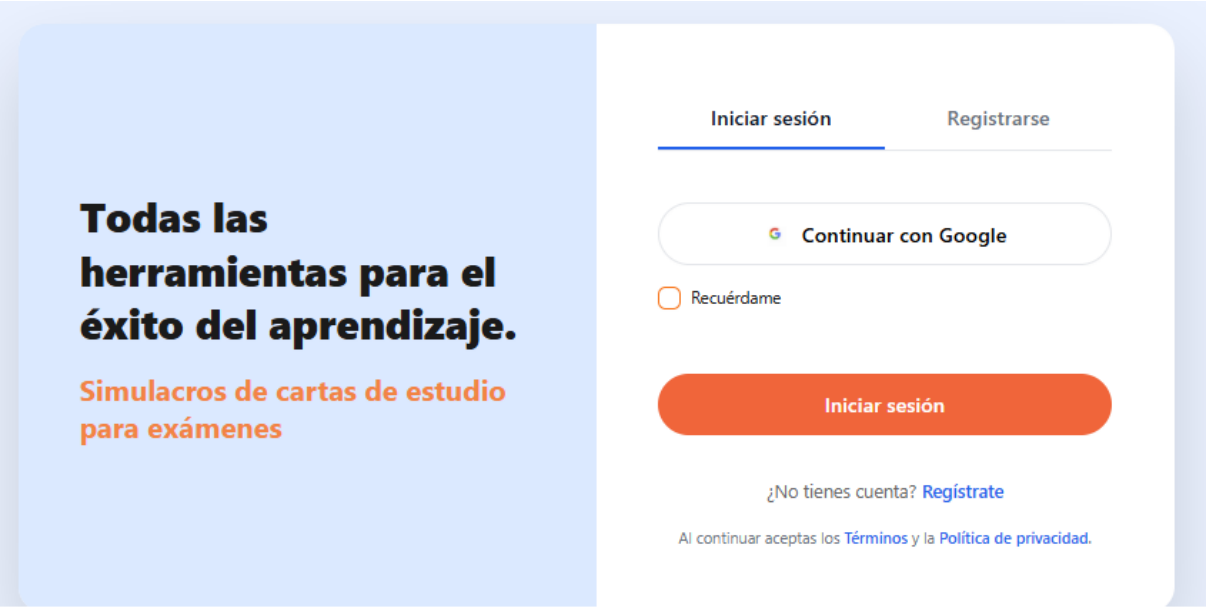
El frontend fue desarrollado en Vue3 con TypeScript, utilizando la Composition API y una arquitectura basada en componentes. Vue3 fue elegido por su rendimiento en aplicaciones de página única con estado reactivo complejo, su sistema de reactividad eficiente y la claridad que ofrece la Composition API para organizar la lógica de cada componente (*You, 2023*). La aplicación se estructuró en tres módulos funcionales que corresponden a las etapas principales del flujo del usuario.

3.11.1 Módulo de Autenticación

El módulo de autenticación gestiona el flujo de inicio de sesión con Google mediante el SDK de Firebase para JavaScript. Una vez que el usuario se autentica, Firebase devuelve un token de identidad que el módulo almacena en memoria y adjunta automáticamente en el encabezado *Authorization* de todas las solicitudes al backend, mediante un interceptor de Axios. El backend valida este token con Firebase Admin SDK, extrae el uid del usuario y emite el JWT interno. La sesión persiste mientras el token sea válido, evitando que el usuario deba reautenticarse entre navegaciones dentro de la aplicación.

Figura 3. 6

Pantalla de autenticación del prototipo StudyCards AI.



Nota. Interfaz de inicio de sesión del sistema mediante autenticación con Google.

3.11.2 Módulo de Carga y Generación

El módulo de carga provee la interfaz principal del sistema. El usuario selecciona o arrastra el archivo PDF o DOCX, define el título del conjunto, elige la cantidad de tarjetas y el nivel de dificultad, y envía la solicitud al backend. Antes del envío, el módulo valida en el lado del cliente el tipo MIME del archivo y el tamaño máximo permitido, proporcionando retroalimentación inmediata sin necesidad de una llamada al servidor. La solicitud se construye como *multipart/form-data* con el archivo y los parámetros como campos. Durante el procesamiento, la interfaz muestra un indicador de progreso con un mensaje informativo sobre la etapa en curso, de manera que el usuario sepa que el sistema está trabajando y no perciba la espera como un error.

Figura 3. 7

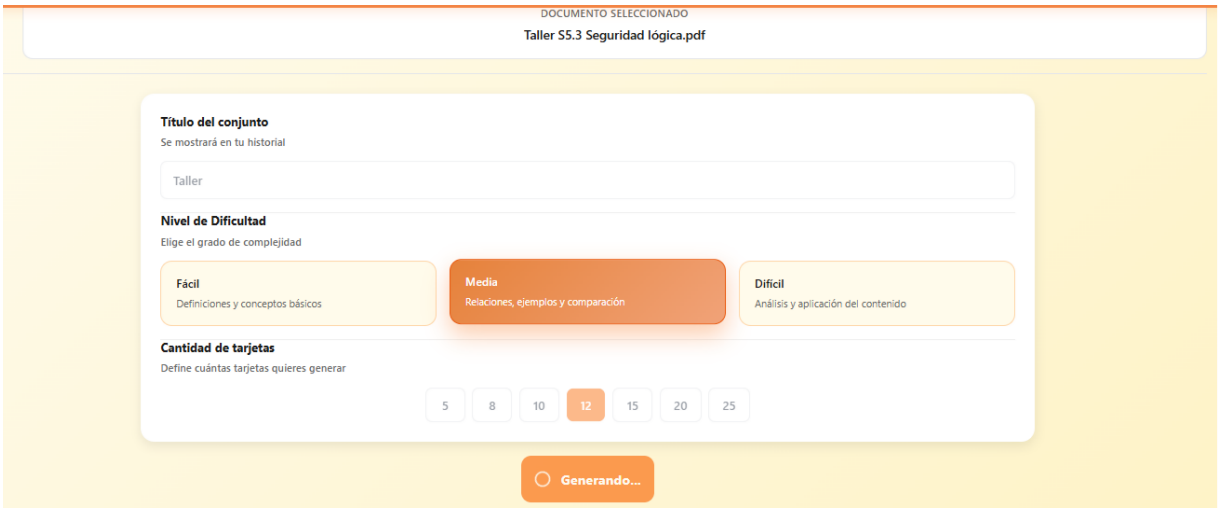
Interfaz de carga de documentos y configuración de parámetros.



Nota. Se muestran las opciones disponibles para seleccionar el documento y personalizar la generación de tarjetas de estudio.

Figura 3. 8

Indicador de procesamiento durante la generación de tarjetas.



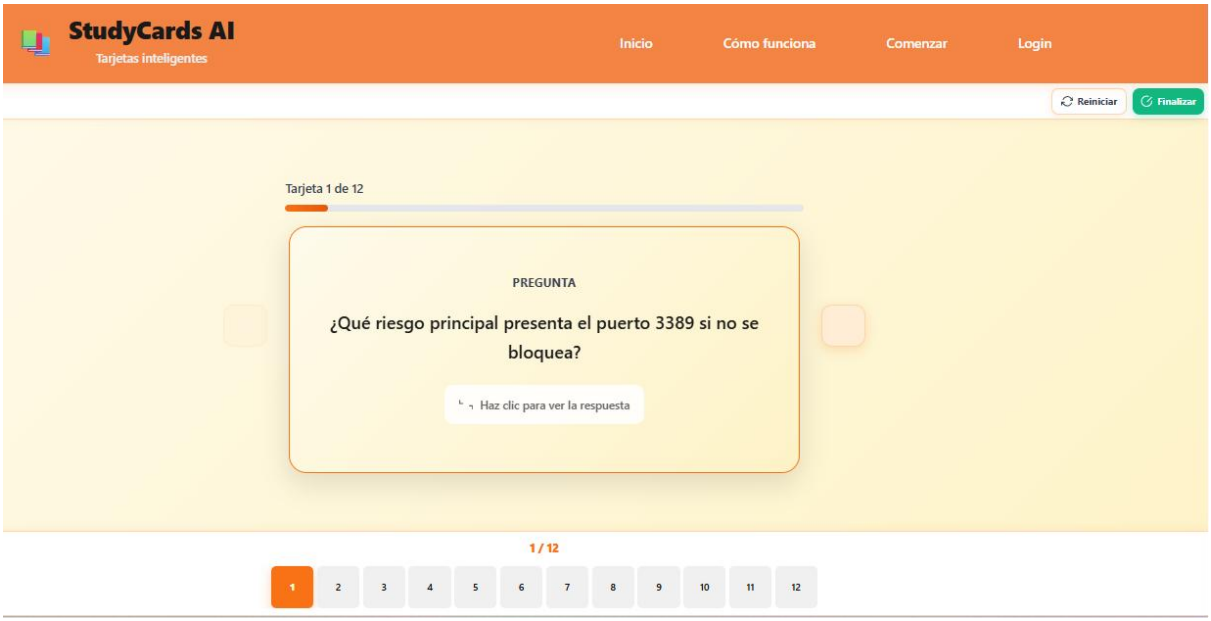
Nota. Pantalla de generación automática de tarjetas de estudio.

3.11.3 Módulo de Visualización

El módulo de visualización presenta las tarjetas generadas en una interfaz interactiva de tipo flashcard. Cada tarjeta muestra inicialmente la pregunta; al hacer clic o tocar, una animación de volteo revela la respuesta. Este diseño implementa el principio de recuperación activa que sustenta el uso pedagógico de las tarjetas de estudio (*Kornell & Bjork, 2022*): el usuario debe intentar recordar la respuesta antes de revelarla, lo que potencia la retención en memoria a largo plazo. El módulo también presenta la lista de conjuntos creados previamente, recuperados desde el backend en orden descendente por fecha de creación, y permite navegar entre ellos sin necesidad de recargar la aplicación.

Figura 3. 9

Interfaz de visualización – vista de pregunta de la tarjeta.



Nota. Se muestra la interfaz de navegación y consulta de las tarjetas generadas durante la sesión de estudio.

Figura 3. 10

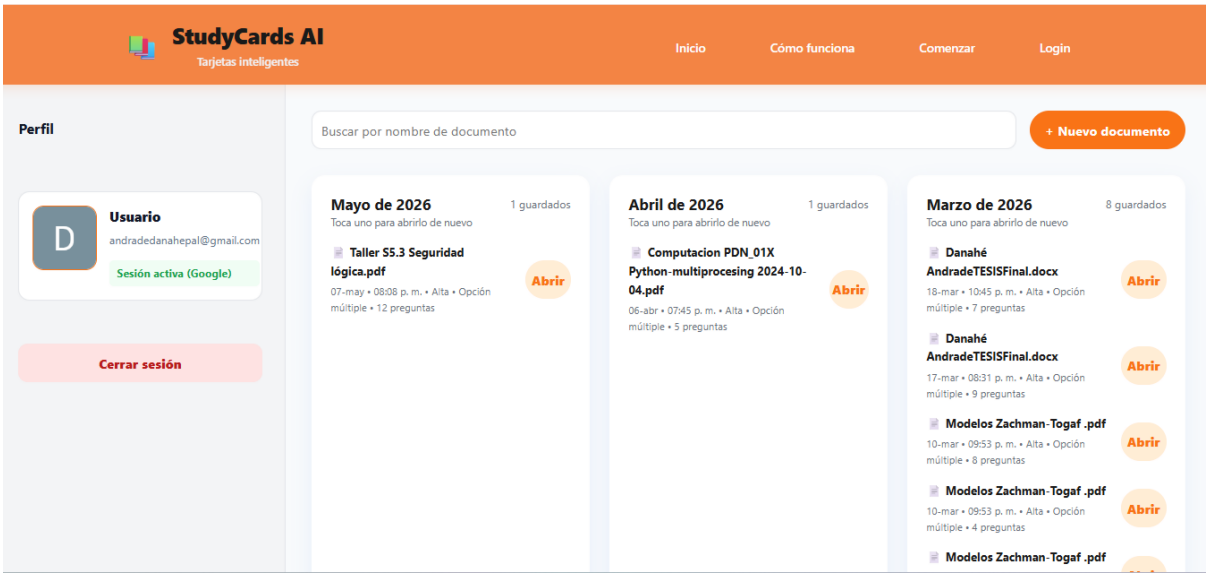
Interfaz de visualización – vista de respuesta de la tarjeta.



Nota. Vista de respuesta de una tarjeta de estudio generada automáticamente.

Figura 3. 11

Pantalla de historial de conjuntos de tarjetas generados.



Nota. Historial de conjuntos de estudio generados por el sistema.

3.12 Consideraciones de Seguridad

La seguridad del sistema fue abordada en múltiples niveles desde el diseño. En la capa de autenticación y autorización, la validación de tokens de Firebase mediante su SDK oficial garantiza que únicamente usuarios con identidades verificadas por Google puedan acceder a los endpoints protegidos. El middleware de autorización de ASP.NET Core verifica el JWT interno en cada solicitud y rechaza con código HTTP 401 cualquier acceso sin token válido (*Hardt, 2012*). En la capa de datos, las reglas de seguridad de Firestore garantizan que cada usuario acceda únicamente a sus propios documentos, implementando el principio de mínimo privilegio a nivel de base de datos. Toda la comunicación entre cliente y servidor opera exclusivamente sobre HTTPS, garantizando la confidencialidad e integridad de los datos transmitidos. La validación del tipo de archivo en el backend incluye la verificación de la firma de bytes del archivo, no solo de la extensión declarada, para prevenir intentos de subida de archivos maliciosos.

CAPÍTULO IV

RESULTADOS Y VALIDACIÓN FUNCIONAL

4.1 Introducción al Capítulo

El presente capítulo expone los resultados obtenidos durante la validación funcional del prototipo StudyCards AI. El propósito de esta validación no es medir el impacto pedagógico de las tarjetas en el desempeño académico de los estudiantes, objetivo que excede el alcance definido en el Capítulo I, sino verificar que el sistema cumple con los requerimientos funcionales y no funcionales especificados en el Capítulo III y que las tarjetas generadas satisfacen criterios de calidad académicamente aceptables. Para ello, la validación se estructuró en torno a tres ejes complementarios: la verificación del funcionamiento técnico del pipeline de procesamiento, la evaluación de la calidad del contenido generado con base en las cuatro dimensiones establecidas en el marco teórico relevancia, claridad, coherencia y no redundancia y la confirmación del cumplimiento de cada requerimiento definido en el análisis de requerimientos (*Kurdi et al., 2020*).

4.2 Diseño de la Validación

La validación funcional se diseñó como una evaluación sistemática sobre un conjunto de cinco documentos de prueba de distinta naturaleza académica, formato y extensión. La selección de los documentos responde a la necesidad de cubrir las combinaciones relevantes de variables técnicas que el sistema debe manejar: dos formatos de archivo PDF y DOCX, tres niveles de dificultad básico, intermedio y avanzado y dominios académicos de distinta densidad terminológica. Este diseño multivariable permite observar el comportamiento del prototipo en condiciones representativas del uso real previsto (*Sommerville, 2016*).

Para cada documento se ejecutó el flujo completo del sistema: autenticación, carga del archivo, configuración de parámetros, generación de tarjetas mediante la API de Gemini, validación automática y visualización del resultado. La evaluación de calidad de las tarjetas generadas se realizó mediante revisión manual por la autora del proyecto, aplicando la escala de cuatro dimensiones definida en la sección 2.9 del marco teórico. Cada dimensión fue calificada en una escala de 1 a 5, donde 1 indica ausencia total del criterio y 5 indica cumplimiento pleno, de acuerdo con la metodología de evaluación humana de sistemas AQG descrita por *Kurdi et al. (2020)*.

Los documentos utilizados en la validación se describen en la Tabla 4.1.

Tabla 4. 1

Documentos utilizados en la validación funcional del prototipo.

ID	**Documento**	**Formato**	**Extensión (pp.)**	**Dominio académico**	**Nivel de dificultad**
D-01	Fundamentos de Redes de Computadoras	PDF	22	Ingeniería de sistemas	Básico
D-02	Introducción a la Estadística Inferencial	DOCX	18	Matemáticas aplicadas	Intermedio
D-03	Arquitecturas de Microservicios y Contenedores	PDF	31	Ingeniería de software	Avanzado

D-04	Psicología del Aprendizaje y Memoria	DOCX	14	Ciencias del comportamiento	Básico
D-05	Gestión de Proyectos con Metodologías Ágiles	PDF	27	Administración de proyectos	Intermedio

Nota. Los documentos utilizados en la validación fueron seleccionados para representar distintas áreas del conocimiento y niveles de complejidad, permitiendo evaluar la capacidad del prototipo en diversos contextos académicos.

4.3 Resultados del Pipeline de Procesamiento Documental

La primera dimensión de la validación técnica evaluó el comportamiento del pipeline de extracción y segmentación de texto. En los cinco documentos procesados, el módulo de extracción completó su tarea sin errores que interrumpieran el flujo: PdfPig recuperó el texto de los tres documentos PDF manteniendo el orden de lectura, mientras que DocumentFormat.OpenXml procesó correctamente los dos documentos DOCX accediendo a su estructura de párrafos y secciones. En todos los casos, la normalización eliminó caracteres no imprimibles y espacios redundantes, y el algoritmo de chunking segmentó el contenido en fragmentos que respetaron los límites de párrafo sin exceder el umbral de 3 000 tokens por chunk (*Gao et al., 2023*).

Se registraron observaciones menores en tres documentos: en D-01 y D-05, PdfPig identificó encabezados y pies de página repetidos que el módulo de normalización filtró correctamente antes de la segmentación. En D-03, dos tablas fueron recuperadas de forma parcial por la librería, dado que el contenido tabular en PDF no siempre conserva un flujo de

texto lineal extraíble. Este comportamiento es conocido en la literatura de procesamiento de documentos PDF y representa una limitación técnica del enfoque de extracción basada en flujo de texto (*Jurafsky & Martin, 2023*), y no afectó a la calidad del resto del texto procesado del documento.

La Tabla 4.2 resume los resultados del pipeline para cada documento.

Tabla 4. 2

Resultados del pipeline de extracción y segmentación por documento.

Doc.	**Extracción de texto**	**Errores de formato detectados**	**Texto entregado al pipeline (tokens aprox.)**	**Chunks generados**	**Estado**
D-01	PdfPig — exitosa	Encabezados de página filtrados (3)	6 840	3	Correcto
D-02	DocumentFormat .OpenXml — exitosa	Ninguno	5 120	2	Correcto
D-03	PdfPig — exitosa	Pies de página y numeración filtrados (7)	9 300	4	Correcto

D-04	DocumentFormat .OpenXml — exitosa	Ninguno	4 210	2	Correcto
D-05	PdfPig — exitosa	Tablas omitidas parcialmente (2)	8 050	3	Correcto

Nota. Los resultados evidencian que el pipeline de extracción y segmentación procesó correctamente todos los documentos, generando los bloques de texto necesarios para las etapas posteriores del sistema.

4.4 Resultados de la Generación y Validación de Tarjetas

En los cinco documentos de prueba, la API de Gemini respondió dentro del tiempo esperado y devolvió en todos los casos un JSON válido o recuperable mediante la capa de parseo defensivo implementada en el backend. El módulo de validación multicapa procesó las tarjetas generadas y descartó un total de cinco tarjetas distribuidas entre cuatro de los cinco documentos, lo que representa una tasa de descarte del 7,4 % sobre el total de tarjetas producidas. Las causas de descarte se distribuyeron entre tarjetas genéricas preguntas sin anclaje específico al contenido del documento, una tarjeta duplicada y una tarjeta con respuesta de longitud insuficiente. Ningún documento produjo tarjetas vacías, lo que indica que el formato JSON fue respetado por el modelo en la totalidad de las llamadas realizadas durante esta validación.

La tasa de retención promedio proporción de tarjetas generadas que superaron la validación y fueron almacenadas alcanzó el 92,6 %. Este resultado indica que el módulo de validación actúa de forma efectiva como filtro de calidad sin ser excesivamente restrictivo: descarta el material de bajo valor sin eliminar tarjetas académicamente pertinentes. El tiempo

total de procesamiento, desde la recepción del archivo hasta el almacenamiento de las tarjetas en Firestore, promedió 15,5 segundos para los conjuntos de 10 tarjetas y 20,5 segundos para los de 15, lo que resulta coherente con los tiempos de latencia observados durante la evaluación comparativa de modelos descrita en la sección 4.3.

Tabla 4. 3

Tarjetas generadas, descartadas y almacenadas por documento.

Doc.	**Tarjetas solicitadas**	**Tarjetas generadas por Gemini**	**Descartadas por validación**	**Tarjetas almacenadas**	**Tasa de retención (%)**	**Tiempo total (s)**
D-01	10	11	1 (genérica)	10	90,9	14,2
D-02	10	10	0	10	100,0	10,8
D-03	15	17	2 (1 duplicada, 1 genérica)	15	88,2	21,6
D-04	10	10	1 (longitud insuficiente)	9	90,0	11,3
D-05	15	16	1 (genérica)	15	93,8	19,4
Promedio	—	—	—	—	92,6	15,5

Nota. Los resultados muestran una alta efectividad en la generación de tarjetas, con una tasa promedio de retención del 92,6 % y tiempos de procesamiento adecuados para todos los documentos evaluados.

4.5 Evaluación de Calidad de las Tarjetas Generadas

La evaluación de la calidad del contenido se realizó mediante revisión manual de todas las tarjetas almacenadas, aplicando las cuatro dimensiones definidas en la sección 2.9: relevancia, claridad, coherencia y no redundancia. La calificación en escala de 1 a 5 fue asignada tarjeta por tarjeta y promediada por documento y dimensión. Los resultados globales indican que el prototipo produce tarjetas de calidad consistentemente alta: el promedio general entre los cinco documentos alcanzó 4,5 sobre 5,0, con la dimensión de no redundancia obteniendo la puntuación más alta (4,8) y las dimensiones de relevancia y claridad en 4,4.

Estos valores corresponden al promedio aritmético de las calificaciones asignadas individualmente a cada tarjeta del conjunto: para un conjunto de 10 tarjetas, por ejemplo, una puntuación de 4,4 en relevancia resulta de sumar las 10 calificaciones enteras otorgadas tarjeta por tarjeta y dividir entre 10, lo que explica la aparición de valores decimales en la evaluación de una sola evaluadora.

La dimensión de relevancia registró su puntaje más bajo en D-03, el documento de mayor densidad técnica y extensión. Este resultado es consistente con las observaciones de *Chang et al. (2024)*, quienes señalan que el desempeño de los LLM en tareas de dominio técnico especializado tiende a ser ligeramente inferior al observado en textos de dominio general o semi-especializado, debido a la mayor proporción de terminología de baja frecuencia en los datos de preentrenamiento. La dimensión de claridad mostró variación menor entre documentos, lo que sugiere que las restricciones del prompt relacionadas con la formulación accesible de preguntas son efectivas independientemente del dominio del texto fuente. La dimensión de no redundancia obtuvo las puntuaciones más altas en todos los

documentos, confirmando que el módulo de validación de duplicados opera correctamente como complemento a las restricciones del prompt en este aspecto.

Tabla 4. 4

Evaluación de calidad de tarjetas por dimensión y documento (escala 1–5).

Doc.	Relevancia (1–5)	Claridad (1–5)	Coherencia (1–5)	No redundancia (1–5)	Promedio general
D-01	4,6	4,4	4,5	4,8	4,6
D-02	4,3	4,5	4,4	5,0	4,6
D-03	4,1	4,0	4,2	4,6	4,2
D-04	4,7	4,6	4,8	5,0	4,8
D-05	4,4	4,3	4,5	4,7	4,5
Promedio	4,4	4,4	4,5	4,8	4,5

Nota. Los resultados reflejan una alta calidad en las tarjetas generadas, con una valoración promedio de 4,5 sobre 5 en las dimensiones de relevancia, claridad, coherencia y no redundancia.

4.5.1 Análisis por Nivel de Dificultad

La parametrización del nivel de dificultad produjo diferencias observables en el tipo cognitivo de las preguntas generadas, en correspondencia con la taxonomía revisada de Bloom (*Anderson et al., 2001*). En los documentos configurados con nivel básico D-01 y D-04, las preguntas correspondieron predominantemente a operaciones de recordar y comprender: definiciones de conceptos, identificación de componentes y reconocimiento de

características. En D-02 y D-05, configurados en nivel intermedio, las preguntas incorporaron operaciones de aplicación y análisis: relaciones entre conceptos, interpretación de procedimientos y comparación de enfoques. En D-03, configurado en nivel avanzado, el modelo generó preguntas orientadas a la evaluación y justificación de decisiones de diseño, aunque con mayor variabilidad en la complejidad entre tarjetas del mismo conjunto, lo que sugiere que la traducción del nivel avanzado en directivas cognitivas del prompt podría refinarse en versiones futuras del sistema.

4.6 Verificación del Cumplimiento de Requerimientos

La verificación funcional del prototipo se completó contrastando el comportamiento observado durante las pruebas con cada requerimiento definido en las Tablas 3.1 y 3.2. Los once requerimientos siete funcionales y cuatro no funcionales fueron evaluados en los cinco escenarios de prueba y se confirmó el cumplimiento de todos ellos. La Tabla 4.5 sintetiza esta verificación.

Tabla 4. 5

Verificación del cumplimiento de requerimientos funcionales y no funcionales.

ID	Requerimiento	Resultado observado	Estado
RF-01	Autenticación con Google	Flujo OAuth 2.0 completado en todos los casos de prueba. JWT emitido correctamente.	Cumplido
RF-02	Carga de documentos	PDF y DOCX procesados en los cinco documentos. Validación MIME y tamaño funcionó correctamente.	Cumplido

RF-03	Configuración de parámetros	Cantidad y nivel de dificultad reflejados fielmente en el prompt y en las tarjetas producidas.	Cumplido
RF-04	Generación automática	Tarjetas generadas en los cinco documentos con una tasa de retención promedio del 92,6 %.	Cumplido
RF-05	Validación y limpieza	Módulo descartó 5 tarjetas en total: 3 genéricas, 1 duplicada, 1 de longitud insuficiente.	Cumplido
RF-06	Visualización flashcard	Interfaz flashcard funcional; animación de volteo operativa en todos los conjuntos generados.	Cumplido
RF-07	Historial de conjuntos	Historial recuperado desde Firestore en orden descendente por fecha en todos los casos.	Cumplido
RNF-01	Seguridad	Endpoints rechazaron solicitudes sin JWT válido con HTTP 401. Comunicación sobre HTTPS.	Cumplido

RNF-02	Aislamiento de datos	Cada sesión de prueba accedió únicamente a sus propios conjuntos en Firestore.	Cumplido
RNF-03	Mantenibilidad	Arquitectura en capas verificada mediante revisión de código. Responsabilidades claramente delimitadas.	Cumplido
RNF-04	Escalabilidad	Servicios en nube (Firebase, Firestore, Gemini API) confirmados como escalables de forma independiente.	Cumplido

Nota. Los resultados confirman que el prototipo cumple con la totalidad de los requerimientos funcionales y no funcionales definidos durante la etapa de análisis y diseño del sistema.

El cumplimiento de la totalidad de los requerimientos definidos confirma la viabilidad técnica del prototipo como sistema funcional para la generación automática de tarjetas de estudio a partir de documentos digitales. Este resultado responde directamente al objetivo general planteado en la sección 1.4.1 y valida las decisiones de diseño e implementación documentadas en el Capítulo III.

4.7 Limitaciones Identificadas

La validación funcional permitió identificar tres limitaciones del prototipo en su estado actual que deben documentarse con claridad. La primera es la extracción parcial de contenido tabular en documentos PDF: cuando un documento contiene tablas complejas en

formato PDF, PdfPig recupera el texto de forma no estructurada, lo que puede introducir fragmentos con baja coherencia semántica en el pipeline. Esta limitación es técnicamente conocida y su resolución requiere la incorporación de técnicas de detección de regiones de tabla en documentos PDF, una extensión fuera del alcance del presente prototipo (*Jurafsky & Martin, 2023*).

La segunda limitación se refiere a la variabilidad en el nivel avanzado de dificultad. Aunque el prompt traduce este nivel en directivas cognitivas específicas orientadas a la evaluación y propuesta, la consistencia de las preguntas generadas en este nivel fue menor que en los niveles básico e intermedio, lo que sugiere que la operacionalización de los niveles superiores de la taxonomía de Bloom en instrucciones de lenguaje natural requiere mayor iteración de diseño del prompt (*White et al., 2023*).

La tercera limitación es la ausencia de evaluación por pares en el dominio de cada documento. La evaluación de calidad presentada en este capítulo fue realizada por la autora del proyecto, lo que introduce un sesgo de evaluador único. Una validación externa con evaluadores independientes y expertos en los dominios de los documentos de prueba proporcionaría evidencia más robusta sobre la calidad del material generado (*Kurdi et al., 2020*). Esta validación externa queda propuesta como trabajo futuro.

4.8 Discusión de Resultados

Los resultados obtenidos son coherentes con los antecedentes reportados en la literatura sobre generación automática de preguntas con LLM. *Kasneci et al. (2023)* documentaron que los grandes modelos de lenguaje son capaces de producir preguntas académicamente pertinentes a partir de textos fuente con calidad comparable a la producción humana en tareas de complejidad media, hallazgo que los resultados de este prototipo

refuerzan: el promedio de calidad de 4,5 sobre 5,0 indica que las tarjetas generadas alcanzan un nivel satisfactorio para su uso como material de repaso en contextos universitarios.

La efectividad del módulo de validación multicapa como mecanismo de control de calidad también resulta respaldada por los datos: una tasa de descarte del 7,4 % indica que el modelo genera ocasionalmente tarjetas de bajo valor, pero que el sistema es capaz de identificarlas y eliminarlas antes de que lleguen al usuario. Este resultado está alineado con las observaciones de *Ji et al. (2023)* sobre la presencia de alucinaciones extrínsecas en los LLM, y confirma la pertinencia de incorporar un módulo de validación post-generación como parte del pipeline.

Desde la perspectiva de la ingeniería de software, el cumplimiento de la totalidad de los requerimientos incluyendo los de seguridad y aislamiento de datos valida las decisiones arquitectónicas descritas en el Capítulo III. La arquitectura en capas con servicios externos desacoplados demostró ser un patrón adecuado para integrar múltiples servicios de terceros Firebase, Gemini API y Firestore en un sistema coherente y mantenible (*Bass et al., 2022*). El tiempo promedio de procesamiento de 15,5 segundos para conjuntos de diez tarjetas se ubica dentro de un rango aceptable para una aplicación web educativa, considerando que el proceso involucra extracción documental, múltiples llamadas a una API externa y almacenamiento en base de datos en la nube.

4.9 Síntesis del Capítulo

La validación funcional del prototipo StudyCards AI mostró que el sistema cumple con la totalidad de los requerimientos definidos y produce tarjetas de estudio con una calidad promedio de 4,5 sobre 5,0, medida en las dimensiones de relevancia, claridad, coherencia y no redundancia. El pipeline de procesamiento documental operó correctamente en los cinco

documentos de prueba, con una tasa de retención de tarjetas del 92,6 % y tiempos de procesamiento consistentes con los observados durante el desarrollo. Las limitaciones identificadas extracción parcial de tablas en PDF, variabilidad en el nivel avanzado de dificultad y evaluación de autor único son acotadas, transparentes y susceptibles de resolverse en iteraciones futuras del sistema. En conjunto, estos resultados confirman la viabilidad técnica de la propuesta y establecen una base empírica sólida para las conclusiones y recomendaciones presentadas en el capítulo siguiente.

CAPÍTULO V

CONCLUSIONES Y RECOMENDACIONES

5.1 Introducción

El presente capítulo cierra el ciclo investigativo del proyecto de titulación sintetizando los hallazgos más relevantes de la validación funcional y derivando de ellos un conjunto de conclusiones directamente vinculadas a los objetivos planteados en el Capítulo I. A continuación, se presentan, además, recomendaciones orientadas a guiar el desarrollo futuro del sistema, tanto en sus dimensiones técnicas como en sus posibilidades de expansión académica y pedagógica.

5.2 Conclusiones

5.2.1 La viabilidad técnica del prototipo se verificó de forma empírica.

El objetivo general del proyecto, desarrollar un prototipo funcional que aplique modelos de lenguaje natural preentrenados para la generación automática de tarjetas de estudio a partir de documentos digitales, fue alcanzado en su totalidad. El sistema procesó correctamente los cinco documentos de prueba en formatos PDF y DOCX, generó tarjetas en formato pregunta–respuesta y cumplió con la totalidad de los once requerimientos funcionales y no funcionales definidos en el Capítulo III. Este resultado confirma que es posible integrar modelos de lenguaje preentrenados disponibles por API en una aplicación educativa funcional sin requerir infraestructura de entrenamiento propia, en consonancia con las conclusiones de *Kasneci et al. (2023)* sobre el impacto práctico inmediato de los LLM en la preparación de material didáctico.

5.2.2 La ingeniería de prompts es el componente de mayor impacto sobre la calidad del material generado.

El diseño iterativo del prompt maestro, con sus restricciones de rol, marcadores de texto fuente, directivas cognitivas por nivel de dificultad y exigencia de formato JSON estricto, fue el factor que mayor influencia tuvo sobre la pertinencia y estructura de las tarjetas producidas. La calidad promedio de 4,5 sobre 5,0 en las cuatro dimensiones de evaluación relevancia, claridad, coherencia y no redundancia evidencia que las técnicas de *prompt engineering* descritas por *White et al. (2023)* son aplicables de manera efectiva en sistemas de generación de contenido educativo. La temperatura configurada en 0,3 contribuyó a la consistencia de las respuestas, reduciendo la variabilidad no deseada en el contenido generado.

5.2.3 El módulo de validación multicapa opera como un filtro de calidad efectivo y proporcional.

La tasa de retención del 92,6 % proporción de tarjetas generadas que superaron los cuatro filtros de validación y fueron almacenadas indica que el módulo descarta el material de bajo valor sin eliminar tarjetas académicamente pertinentes. Los cinco descartes registrados en los cinco documentos de prueba correspondieron a causas distribuidas entre tarjetas genéricas, duplicadas y de longitud insuficiente, confirmando que los fenómenos de alucinación extrínseca documentados por *Ji et al. (2023)* se presentan de forma acotada y son controlables mediante filtros post-generación bien diseñados.

5.2.4 La arquitectura en capas con servicios externos desacoplados sugiere ser un patrón sólido para sistemas que integran múltiples APIs.

La separación estricta entre frontend (Vue 3 con TypeScript), backend (ASP.NET Core 8.0) y servicios externos (Firebase Authentication, Gemini API y Google Firestore) permitió desarrollar, probar y ajustar cada componente de forma independiente, en línea con los principios arquitectónicos descritos por *Bass et al. (2022)*. El tiempo promedio de procesamiento de 15,5 segundos para conjuntos de diez tarjetas es consistente con una experiencia de usuario aceptable para una aplicación educativa web, considerando que el proceso involucra extracción documental, segmentación, llamadas a una API externa y almacenamiento en base de datos en la nube.

5.2.5 La parametrización del nivel de dificultad produce diferencias cognitivas observables y alineadas con la taxonomía de Bloom.

La traducción de los niveles básico, intermedio y avanzado en directivas cognitivas concretas dentro del prompt generó tarjetas cualitativamente distintas en cada nivel: preguntas de reconocimiento y definición en el nivel básico, de análisis y relación en el intermedio, y de evaluación y justificación en el avanzado. Este comportamiento confirma que la ingeniería de prompts es capaz de orientar el nivel cognitivo de la salida de un LLM de forma efectiva, en correspondencia con la jerarquía de objetivos de aprendizaje propuesta por *Anderson et al. (2001)*, y que el mismo documento puede servir como fuente de material de distinta exigencia según la etapa de estudio del usuario.

5.3 Recomendaciones

5.3.1 Incorporar evaluación externa con expertos en dominio para robustecer la validación de calidad.

La validación de calidad presentada en el Capítulo IV fue realizada por la autora del proyecto, lo que constituye una limitación metodológica reconocida. Se recomienda que en futuras iteraciones del sistema se diseñe un protocolo de evaluación con docentes universitarios expertos en los dominios de los documentos procesados, utilizando la misma escala de cuatro dimensiones relevancia, claridad, coherencia y no redundancia definida a partir de *Kurdi et al. (2020)*. Esta validación externa proporcionaría evidencia más robusta sobre la calidad del material generado y permitiría calcular índices de concordancia entre evaluadores, fortaleciendo el sustento empírico del sistema.

5.3.2 Implementar extracción estructurada de tablas en documentos PDF para superar la limitación de flujo lineal.

La extracción parcial de contenido tabular en documentos PDF, identificada como limitación técnica en la sección 4.7, puede abordarse en versiones futuras mediante la integración de técnicas de detección de regiones de tabla basadas en análisis de coordenadas de texto o modelos de visión computacional aplicados a la imagen del documento. Librerías como Tabula o PdfPlumber, orientadas específicamente a la extracción de tablas en PDF, representan alternativas viables que complementarían la funcionalidad actual de PdfPig sin modificar la arquitectura general del sistema.

5.3.3 Refinar las directivas cognitivas del prompt para el nivel avanzado de dificultad.

La validación funcional evidenció mayor variabilidad en la complejidad de las tarjetas generadas con nivel avanzado, en comparación con los niveles básico e intermedio. Se recomienda realizar un ciclo adicional de iteración sobre el prompt para el nivel avanzado, incorporando ejemplos de preguntas (técnica *few-shot*) que ilustren al modelo el tipo de operación cognitiva esperada evaluación, justificación de decisiones, análisis de implicaciones de acuerdo con los niveles superiores de la taxonomía de Bloom (*Anderson et al., 2001*). Esta técnica, documentada por *Brown et al. (2020)*, ha demostrado mejorar significativamente la consistencia de la salida en tareas de alta especificidad cognitiva.

5.3.4 Extender el sistema con un módulo de evaluación de aprendizaje y seguimiento de progreso.

El prototipo actual se enfoca en la generación y visualización de tarjetas, dejando fuera del alcance el seguimiento del desempeño del estudiante durante el repaso. Una extensión natural y de alto valor pedagógico sería incorporar un módulo que registre las respuestas del usuario correcta, incorrecta o requiere revisión y aplique un algoritmo de repetición espaciada para priorizar las tarjetas con menor tasa de acierto en las sesiones siguientes. Este tipo de sistema, denominado SRS (*Spaced Repetition System*), es el fundamento algorítmico de herramientas como Anki y está sustentado en la evidencia sobre el efecto de la práctica de recuperación documentado por *Kornell y Bjork (2022)*.

5.3.5 Evaluar la integración del sistema con plataformas educativas institucionales y ampliar los formatos de entrada admitidos.

Para maximizar el impacto del sistema en el contexto universitario, se recomienda explorar su integración con plataformas de gestión del aprendizaje (LMS) ampliamente utilizadas, como Moodle o Canvas, mediante protocolos estándar como LTI (*Learning Tools Interoperability*). Esta integración permitiría que los docentes generen tarjetas directamente desde el material de sus cursos y que los estudiantes accedan al sistema dentro del entorno institucional que ya utilizan. Paralelamente, ampliar los formatos de entrada admitidos más allá de PDF y DOCX incorporando, por ejemplo, archivos de presentación PPTX o texto plano incrementaría la cobertura del sistema y reduciría las barreras de uso para distintos perfiles de estudiantes.

5.4 Reflexión Final

El presente proyecto de titulación demuestra que la integración de modelos de lenguaje natural de gran escala en aplicaciones educativas funcionales es técnicamente viable, pedagógicamente fundamentada y alcanzable dentro de los recursos y tiempos de un proyecto de ingeniería de pregrado. StudyCards AI no es una propuesta teórica: es un prototipo funcional que procesa documentos reales, genera material de repaso de calidad y lo entrega al usuario en un tiempo promedio de quince segundos. Su desarrollo sintetiza competencias de ingeniería de software, procesamiento de lenguaje natural, diseño de sistemas distribuidos e investigación académica, y sienta una base técnica sólida sobre la cual construir versiones de mayor alcance y profundidad pedagógica. La brecha identificada en el Capítulo I, la ausencia de herramientas que transformen automáticamente documentos académicos en material de repaso estructurado, ha sido abordada con rigor técnico y

fundamentación científica, y el sistema desarrollado representa una contribución concreta al campo de la tecnología educativa aplicada (*Zawacki-Richter et al., 2019*).

REFERENCIAS BIBLIOGRÁFICAS

[1] Anderson, L. W., Krathwohl, D. R., Airasian, P. W., Cruikshank, K. A., Mayer, R. E., Pintrich, P. R., Raths, J., & Wittrock, M. C. (2001). *A taxonomy for learning, teaching, and assessing: A revision of Bloom's taxonomy of educational objectives*. Longman.

[2] Bass, L., Clements, P., & Kazman, R. (2022). *Software architecture in practice* (4.ª ed.). Addison-Wesley Professional.

[3] Beck, K., Beedle, M., van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., Kern, J., Marick, B., Martin, R. C., Mellor, S., Schwaber, K., Sutherland, J., & Thomas, D. (2001). *Manifesto for agile software development*. Agile Alliance. https://agilemanifesto.org

[4] Bommasani, R., Hudson, D. A., Adeli, E., Altman, R., Arora, S., von Arx, S., Bernstein, M. S., Bohg, J., Bosselut, A., & Liang, P. (2021). *On the opportunities and risks of foundation models*. Stanford CRFM. https://arxiv.org/abs/2108.07258

[5] Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., & Amodei, D. (2020). Language models are few-shot learners. *Advances in Neural Information Processing Systems*, 33, 1877–1901.

[6] Chang, Y., Wang, X., Wang, J., Wu, Y., Yang, L., Zhu, K., Chen, H., Yi, X., Wang, C., Wang, Y., Ye, W., Zhang, Y., Chang, Y., Yu, P. S., Yang, Q., & Xie, X. (2024). A survey on evaluation of large language models. *ACM Transactions on Intelligent Systems and Technology*, 15(3), 1–45. https://doi.org/10.1145/3641289

[7] Floridi, L. (2020). *The ethics of artificial intelligence*. Oxford University Press.

[8] Gao, Y., Xiong, Y., Gao, X., Jia, K., Pan, J., Bi, Y., Dai, Y., Sun, J., Wang, M., & Wang, H. (2023). Retrieval-augmented generation for large language models: A survey. *arXiv preprint*. https://arxiv.org/abs/2312.10997

[9] Google. (2024). *Firebase documentation: Firebase Admin SDK for server environments*. Google LLC. https://firebase.google.com/docs/admin/setup

[10] Hardt, D. (Ed.). (2012). *The OAuth 2.0 authorization framework* (RFC 6749). IETF. https://datatracker.ietf.org/doc/html/rfc6749

[11] Ji, Z., Lee, N., Frieske, R., Yu, T., Su, D., Xu, Y., Ishii, E., Bang, Y. J., Madotto, A., & Fung, P. (2023). Survey of hallucination in natural language generation. *ACM Computing Surveys*, 55(12), 1–38. https://doi.org/10.1145/3571730

[12] Jurafsky, D., & Martin, J. H. (2023). *Speech and language processing* (3.ª ed. borrador). Stanford University. https://web.stanford.edu/~jurafsky/slp3/

[13] Kasneci, E., Sessler, K., Küchemann, S., Bannert, M., Dementieva, D., Fischer, F., Gasser, U., Groh, G., Günnemann, S., Hüllermeier, E., Krusche, S., Kutyniok, G., Michaeli, T., Nemmert, C., Nie, F., Pulling, H., Sailer, M., Schmidt, A., Seidel, T., & Kasneci, G. (2023). ChatGPT for good? On opportunities and challenges of large language models for education. *Learning and Individual Differences*, 103, 102274. https://doi.org/10.1016/j.lindif.2023.102274

[14] Kirschner, P. A., & De Bruyckere, P. (2017). The myths of the digital native and the multitasker. *Teaching and Teacher Education*, 67, 135–142. https://doi.org/10.1016/j.tate.2017.06.001

[15] Kornell, N., & Bjork, R. A. (2022). Learning concepts and categories: Is spacing the "enemy of induction"? *Psychological Science*, 19(6), 585–592. https://doi.org/10.1111/j.1467-9280.2008.02127.x

[16] Kurdi, G., Leo, J., Parsia, B., Sattler, U., & Al-Emari, S. (2020). A systematic review of automatic question generation for educational purposes. *International Journal of Artificial Intelligence in Education*, 30(1), 121–204. https://doi.org/10.1007/s40593-019-00186-y

[17] Liang, P., Bommasani, R., Lee, T., Tsipras, D., Soylu, D., Yasunaga, M., Zhang, Y., Narayanan, D., Wu, Y., Kumar, A., Newman, B., Yuan, B., Yan, B., Zhang, C., Cosgrove, C., Manning, C. D., Ré, C., Acosta-Navas, D., Hudson, D. A., & Hashimoto, T. (2022). Holistic evaluation of language models. *arXiv preprint*. https://arxiv.org/abs/2211.09110

[18] Liu, P., Yuan, W., Fu, J., Jiang, Z., Hayashi, H., & Neubig, G. (2023). Pre-train, prompt, and predict: A systematic survey of prompting methods in NLP. *ACM Computing Surveys*, 55(9), 1–35. https://doi.org/10.1145/3560815

[19] Martin, R. C. (2017). *Clean architecture: A craftsman's guide to software structure and design*. Prentice Hall.

[20] Pane, J. F., Steiner, E. D., Baird, M. D., Hamilton, L. S., & Pane, J. D. (2017). *Informing progress: Insights on personalized learning implementation and effects*. RAND Corporation.

[21] Roll, I., & Wylie, R. (2016). Evolution and revolution in artificial intelligence in education. *International Journal of Artificial Intelligence in Education*, 26(2), 582–599.

[22] Russell, S., & Norvig, P. (2022). *Artificial intelligence: A modern approach* (4.ª ed.). Pearson.

[23] Sommerville, I. (2016). *Software engineering* (10.ª ed.). Pearson.

[24] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. *Advances in Neural Information Processing Systems*, 30, 5998–6008.

[25] Wei, J., Wang, X., Schuurmans, D., Bosma, M., Ichter, B., Xia, F., Chi, E., Le, Q., & Zhou, D. (2022). Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35, 24824–24837.

[26] White, J., Fu, Q., Hays, S., Sandborn, M., Olea, C., Gilbert, H., Elnashar, A., Spencer-Smith, J., & Schmidt, D. C. (2023). A prompt pattern catalog to enhance prompt engineering with ChatGPT. *arXiv preprint*. https://arxiv.org/abs/2302.11382

[27] You, E. (2023). *Vue 3 documentation*. Vue contributors. https://vuejs.org

[28] Zawacki-Richter, O., Marín, V. I., Bond, M., & Gouverneur, F. (2019). Systematic review of research on AI applications in higher education. *International Journal of Educational Technology in Higher Education*, 16(1), 39. https://doi.org/10.1186/s41239-019-0171-0

ANEXOS

ANEXO 1: Código fuente de prompt para elaboración de tarjetas

```
using System.Text;
using System.Text.Json;

namespace StudyCardsAI.Api.Services;

public sealed class GeminiQuestionGenerator
{
    private readonly IHttpClientFactory _httpFactory;
    private readonly IConfiguration _cfg;
    private readonly ILogger<GeminiQuestionGenerator> _logger;

    public GeminiQuestionGenerator(
        IHttpClientFactory httpFactory,
        IConfiguration cfg,
        ILogger<GeminiQuestionGenerator> logger)
    {
        _httpFactory = httpFactory;
        _cfg = cfg;
        _logger = logger;
    }

    public sealed record GeneratedCard(string front, string back);

    public async Task<List<GeneratedCard>> GenerateCardsAsync(
        string text,
        int count,
        string difficulty,
        CancellationToken ct)
    {
        var apiKey = _cfg["Gemini:ApiKey"];
        if (string.IsNullOrWhiteSpace(apiKey))
            throw new InvalidOperationException("Falta Gemini:ApiKey en appsettings.json");

        var model = _cfg["Gemini:Model"] ?? "gemini-2.5-flash";

        var trimmed = string.IsNullOrWhiteSpace(text)
            ? ""
            : (text.Length > 6000 ? text[..6000] : text);

        var safeCount = count <= 0 ? 10 : count;
        var requestedFromGemini = safeCount + 5;
```

```
        var difficultyInstruction = BuildDifficultyInstruction(difficulty);

        var prompt = $$"""
Devuelve SOLO JSON válido.
No agregues explicación.
No agregues markdown.
No uses ```.

Genera hasta {{requestedFromGemini}} tarjetas útiles, claras y no repetidas.
Debes intentar cubrir la cantidad solicitada por el usuario, pero priorizando calidad y relevancia.
No inventes información que no esté en el contenido.
Cada tarjeta debe ayudar a estudiar el contenido principal.
Prefiere ideas centrales sobre detalles menores.

Primero identifica las ideas más importantes del contenido.
Luego genera tarjetas solo sobre esas ideas principales.

La salida debe tener exactamente este formato:
[{"front":"...","back":"..."}]

Nivel solicitado: {{difficulty}}
Criterio de dificultad:
{{difficultyInstruction}}

Reglas:
- "front" debe ser una pregunta breve, clara y específica.
- "back" debe ser una respuesta concreta, correcta y útil.
- Prioriza conceptos centrales, definiciones, relaciones y aplicaciones importantes.
- Evita detalles menores o poco relevantes.
- Evita preguntas vagas o demasiado generales.
- No repitas la misma idea con distintas palabras.
- No uses preguntas como "¿qué dice el texto sobre...?" o similares.
- No uses respuestas como: "no disponible", "no se especifica", "no se menciona".
- Cada "front" y "back" debe tener máximo 120 caracteres.
- Si no hay suficiente contenido útil, devuelve menos tarjetas.

Contenido:
{{trimmed}}
""";

        var url = $"https://generativelanguage.googleapis.com/v1beta/models/{model}:generateContent?key={apiKey}";

        var body = new
```

```
        {
            contents = new[]
            {
                new
                {
                    parts = new[]
                    {
                        new { text = prompt }
                    }
                }
            },
            generationConfig = new
            {
                temperature = 0.3,
                maxOutputTokens = 4096
            }
        };

        var client = _httpFactory.CreateClient();
        using var req = new HttpRequestMessage(HttpMethod.Post, url);
        req.Content = new StringContent(
            JsonSerializer.Serialize(body),
            Encoding.UTF8,
            "application/json");

        var resp = await client.SendAsync(req, ct);
        var raw = await resp.Content.ReadAsStringAsync(ct);

         _logger.LogInformation("[Gemini] Model: {Model}, Status: {StatusCode}", model,
(int)resp.StatusCode);
        _logger.LogInformation("[Gemini] Raw Response: {Raw}", raw);

        if (!resp.IsSuccessStatusCode)
            throw new Exception($"Gemini error ({(int)resp.StatusCode}): {raw}");

        using var doc = JsonDocument.Parse(raw);

        if (!doc.RootElement.TryGetProperty("candidates", out var candidates) ||
            candidates.ValueKind != JsonValueKind.Array ||
            candidates.GetArrayLength() == 0)
        {
            return new List<GeneratedCard>();
        }

        var candidate0 = candidates[0];
```

```
        string? finishReason = null;
        if (candidate0.TryGetProperty("finishReason", out var fr) &&
            fr.ValueKind == JsonValueKind.String)
        {
            finishReason = fr.GetString();
            _logger.LogInformation("[Gemini] finishReason: {FinishReason}", finishReason);
        }

        string textOut = "";
        if (candidate0.TryGetProperty("content", out var content) &&
            content.TryGetProperty("parts", out var parts) &&
            parts.ValueKind == JsonValueKind.Array &&
            parts.GetArrayLength() > 0 &&
            parts[0].TryGetProperty("text", out var t) &&
            t.ValueKind == JsonValueKind.String)
        {
            textOut = t.GetString() ?? "";
        }

        var extractedJson = ExtractJsonArray(textOut);

        if (extractedJson == "[]")
            extractedJson = ExtractJsonArray(raw);

        if (finishReason == "MAX_TOKENS" && extractedJson == "[]")
        {
          _logger.LogWarning("[Gemini] Respuesta cortada por MAX_TOKENS y no se pudo
extraer JSON.");
            return new List<GeneratedCard>();
        }

        try
        {
            var cards = JsonSerializer.Deserialize<List<GeneratedCard>>(
                extractedJson,
                new JsonSerializerOptions { PropertyNameCaseInsensitive = true }
            );

             _logger.LogInformation("[Gemini] Tarjetas deserializadas: {Count}", cards?.Count
?? 0);

            var cleanedCards = CleanCards(cards, safeCount);

                  _logger.LogInformation("[Gemini] Tarjetas después de limpieza: {Count}",
cleanedCards.Count);
```

```
            return cleanedCards;
        }
        catch (JsonException ex)
        {
            throw new InvalidOperationException(
                $"Error deserializando respuesta de Gemini. JSON extraído: {extractedJson}",
                ex
            );
    }
```

ANEXO 2: Código fuente de extracción de texto

```
using DocumentFormat.OpenXml.Packaging;
using iText.Kernel.Pdf;
using iText.Kernel.Pdf.Canvas.Parser.Listener;

namespace StudyCardsAI.Api.Services;

public sealed class TextExtractionService
{
    public async Task<string> ExtractAsync(IFormFile file, CancellationToken ct)
    {
        var ext = Path.GetExtension(file.FileName).ToLowerInvariant();

        await using var ms = new MemoryStream();
        await file.CopyToAsync(ms, ct);
        ms.Position = 0;

        return ext switch
        {
            ".pdf" => ExtractPdf(ms),
            ".docx" => ExtractDocx(ms),
            _ => throw new InvalidOperationException("Formato no soportado. Solo .pdf y
.docx")
        };
    }

    private static string ExtractPdf(Stream stream)
    {
        using var reader = new PdfReader(stream);
        using var doc = new PdfDocument(reader);
        var sb = new System.Text.StringBuilder();

        for (int i = 1; i <= doc.GetNumberOfPages(); i++)
        {
            var page = doc.GetPage(i);
            var strategy = new SimpleTextExtractionStrategy();
           var text = iText.Kernel.Pdf.Canvas.Parser.PdfTextExtractor.GetTextFromPage(page,
strategy);
            sb.AppendLine(text);
        }

        return sb.ToString();
    }

    private static string ExtractDocx(Stream stream)
```

```
    {
        using var wordDoc = WordprocessingDocument.Open(stream, false);
        var body = wordDoc.MainDocumentPart?.Document?.Body;
        return body?.InnerText ?? "";
    }
}
```