UNIVERSIDAD DE LA REPÚBLICA URUGUAY

FACULTAD DE INGENIERÍA

Inteligencia Artificial aplicada a la enseñanza

FinQuiz: Generación de cuestionarios estudiantiles con modelos de lenguaje

Informe de Proyecto de Grado presentado por

Facundo Busto, Luciano Caussade y Sebastián Reolon

en cumplimiento parcial de los requerimientos para la graduación de la carrera de Ingeniería en Computación de Facultad de Ingeniería de la Universidad de la República

Supervisores

Sandro Moscatelli
Omar Viera

Montevideo, 3 de agosto de 2025

Inteligencia Artificial aplicada a la enseñanza

FinQuiz: Generación de cuestionarios estudiantiles con modelos de lenguaje por Facundo Busto, Luciano Caussade y Sebastián Reolon tiene licencia CC Atribución 4.0.

Agradecimientos

Queremos agradecer a nuestros tutores, Omar Viera y Sandro Moscatelli, por acompañarnos en este proceso.

Al equipo de Programación 1, por su disposición y tiempo dedicado.

A la Universidad de la República, principalmente a la Facultad de Ingeniería, por estos años de formación. Y a todos los docentes que marcaron nuestra carrera.

A nuestros amigos, grandes acompañantes en este proceso. A Juanchi, Dalí, Admin, Salem y Diana, cuyo cariño fue tan importante en este proyecto.

A nuestras parejas, que fueron de gran apoyo en estos años de estudio: Agustín, Bernardo y Allison.

Y a nuestros familiares, en particular a nuestros padres, por el apoyo brindado a lo largo de esta etapa: Hugo, Andrea, Roald, Anahí, Stella y Alberto.

Resumen

El presente trabajo se enmarca en la investigación de la Inteligencia Artificial aplicada a la educación. Con este objetivo, se diseñó y desarrolló "FinQuiz", un sistema web concebido para asistir a los estudiantes en sus procesos de aprendizaje mediante la generación automática de cuestionarios de múltiple opción. Para la creación de dicho material de práctica, el sistema utiliza Grandes Modelos de Lenguaje (*Large Language Models* o LLMs), una de las tecnologías más avanzadas en el campo del procesamiento de lenguaje natural.

La arquitectura de la solución es de tipo cliente-servidor. La lógica central del sistema, encargada del procesamiento de datos y la interacción con el modelo de lenguaje, fue implementada utilizando el framework Ruby on Rails. Por su parte, la interfaz de usuario, con la que interactúan los estudiantes y docentes, fue desarrollada utilizando JavaScript y la biblioteca React.

FinQuiz constituye una propuesta concreta de innovación educativa, facilitando la generación automatizada de material de práctica ajustado a los contenidos de un curso. Esta herramienta promueve un aprendizaje más autónomo, ya que permite a los estudiantes ejercitar sus conocimientos de forma flexible y continua, sin depender exclusivamente de instancias presenciales o de la disponibilidad del cuerpo docente.

La evaluación del sistema arrojó resultados positivos. Las preguntas generadas son coherentes, relevantes y sólidas desde el punto de vista pedagógico. Se concluye, además, que la participación docente continúa siendo esencial, tanto en una etapa previa para definir y ajustar los contenidos teóricos utilizados por el sistema, como en una instancia posterior de validación del material generado, a fin de asegurar su calidad y pertinencia.

Este trabajo no solo aporta conocimiento a un área en constante desarrollo, sino que también sienta bases prometedoras para analizar el impacto de la Inteligencia Artificial en los procesos cognitivos de los estudiantes, y abre nuevas líneas de investigación sobre su integración en tecnologías educativas.

Palabras clave: Inteligencia Artificial, Educación, Modelos de Lenguaje, Generación Automática de Cuestionarios, Ingeniería en Computación, Proyecto de Grado

Índice general

Capítulo 1

Introducción

Este trabajo presenta el diseño, desarrollo y evaluación de *FinQuiz*, una herramienta que utiliza técnicas de Inteligencia Artificial con el objetivo de apoyar el proceso de aprendizaje de los estudiantes de Ingeniería en Computación de la UDELAR, mediante la generación automática de preguntas múltiple opción.

El propósito principal de su desarrollo es investigar el impacto de este tipo de herramientas en el proceso de aprendizaje. En particular, se busca contribuir positivamente a dicho proceso abordando uno de los principales desafíos que plantea la masificación de la enseñanza: la dificultad de ofrecer a cada estudiante un seguimiento personalizado que le permita identificar sus debilidades y avanzar en la comprensión de los contenidos del curso.

La estructura metodológica del trabajo comprende las siguientes etapas:

Investigación Inicial La primera etapa, fundamental para el desarrollo del proyecto, consistió en una investigación del estado del arte en dos dimensiones: por un lado, el campo general de la Inteligencia Artificial y por otro, su aplicación específica en contextos educativos. Durante esta fase se revisaron antecedentes relevantes, incluyendo la evolución histórica de ambas áreas, sus principales enfoques actuales y las aplicaciones más representativas. El resultado de esta etapa se encuentra en el documento "**Marco Teórico**", sus respectivos anexos y sus partes fundamentales resumidas en la sección 2.1.

En esta investigación se sientan las bases de las decisiones tomadas en las siguientes etapas.

Selección de la propuesta a desarrollar Partiendo de los resultados obtenidos durante la etapa de revisión del estado del arte, especialmente en lo referente a las posibles aplicaciones de la Inteligencia Artificial en el ámbito educativo, se procedió a seleccionar una propuesta concreta.

Para esta selección se consideraron distintos elementos tales como: las capacidades técnicas del equipo, las restricciones temporales del proyecto, la utilidad de la idea final, la posibilidad de medir los resultados obtenidos, entre otros. Es-

tos elementos se encuentran explicados en mayor detalle en la sección 2.2, donde son utilizados para analizar distintas alternativas destacando pros y contras.

De este proceso se concluye la elección de **FinQuiz**, una aplicación web diseñada para generar cuestionarios mediante grandes modelos de lenguaje. Su propósito es asistir al estudiante de Ingeniería en Computación de la FING en su proceso de aprendizaje, permitiéndole reforzar sus conocimientos de forma autónoma, sin requerir una intervención docente constante, y recopilando datos que permitan evaluar posteriormente el impacto de la herramienta en su aprendizaje.

Segunda etapa de investigación Definida la propuesta a desarrollar, se llevó a cabo una nueva etapa de investigación centrada en las necesidades específicas del sistema.

Esta investigación incluyó el estudio del formato de preguntas de múltiple opción y sus características particulares 2.3, así como el análisis de diversos enfoques para su generación automatizada y la revisión de sistemas existentes con objetivos similares 2.4.

Además, en esta etapa se sentaron las bases para la fase de experimentación, explorando distintas formas de interacción con modelos de lenguaje 4.1 y estableciendo criterios para evaluar la calidad de las preguntas generadas 4.2.

Implementación El proceso de implementación se llevó a cabo en paralelo con una etapa de experimentación. Mientras se desarrollaban distintas secciones del sistema y se definían sus funcionalidades, se probaron diversas formas de interacción con el modelo de lenguaje, se analizaron los resultados obtenidos de forma iterativa y se tomaron decisiones respecto al modelo más adecuado para utilizar.

El detalle de este proceso se presenta en los capítulos 3 y 4. Su resultado, definido en la sección 3.1, es el sistema desarrollado siguiendo el proceso descrito en la sección 3.2.

Análisis de Resultados Finalmente, el capítulo 5 presenta el análisis de los resultados obtenidos, ofreciendo un cierre al proyecto mediante su revisión crítica y contextualización. A partir de este análisis, y en relación con las etapas anteriores, se identifican oportunidades de mejora que incluyen tanto la implementación de ideas no abordadas como el refinamiento de algunos componentes del sistema que requieren ciertos ajustes. Las conclusiones buscan sintetizar los aprendizajes del proceso y ofrecer una interpretación general de los resultados alcanzados.

Capítulo 2

Revisión de Antecedentes

2.1. Resumen de Marco Teórico

En la siguiente sección se presentan las nociones teóricas que enmarcan el desarrollo del presente proyecto, haciendo énfasis en el estado del arte de la Inteligencia Artificial y en sus aplicaciones en el ámbito educativo. Es posible encontrar una versión extendida de este marco teórico en el documento "**Marco Teórico**".

El concepto "Inteligencia Artificial" (o IA) fue concebido en 1955 por John McCarthy (McCarthy, 2007) en el marco del *"Dartmouth Summer Research Project on Artificial Intelligence"*, evento que puede ser considerado como el acto fundacional de la IA como área de estudio. En este, se define la Inteligencia Artificial como:

"La ciencia e ingeniería de crear máquinas inteligentes, en particular programas de computadora inteligentes. Está relacionada con la tarea de utilizar computadoras para entender la inteligencia humana, aunque la IA no debe limitarse a métodos que sean biológicamente observables"

Esta definición plantea una idea de IA en base a sus aspiraciones pero no da una definición concreta de estas ni explica como alcanzarlas. El test de Turing (Turing, 1950) es una de las primeras propuestas que busca entender cuándo una máquina exhibe comportamientos inteligentes y afirma que esto se logra si, durante una interacción en lenguaje natural, un ser humano no es capaz de diferenciar si las respuestas provienen de una persona o de una computadora.

En concreto, es posible ver que para que una computadora pase esta prueba la misma debe tener las siguientes capacidades:

- **Aprendizaje Automático**, para adaptarse a nuevas circunstancias y lograr extrapolar patrones.
- **Procesamiento de Lenguaje Natural**, para interpretar las preguntas y comunicar sus respuestas.

- **Representación de Conocimiento**, para almacenar el conocimiento que posee.
- **Razonamiento Automatizado**, para en base al conocimiento almacenado llegar a nuevas conclusiones a la hora de dar respuestas.

Además si se considera un test de Turing completo, donde se incluye interacción física, se suman a estos requisitos de **Visión por Computadora** y aspectos de **Robótica**.

Estas seis disciplinas mencionadas conforman las áreas de estudio de la Inteligencia Artificial, lo cual respalda la validez de la teoría de Turing que sigue vigente más de cincuenta años después de haber sido concebida. En concreto, este apartado entra en detalle únicamente en las disciplinas que resultan relevantes para la aplicación desarrollada en este proyecto, dejando de lado algunas de estas ramas que fueron revisadas en profundidad en el documento "Marco Teórico".

2.1.1. Aprendizaje Automático

El aprendizaje automático, en inglés **Machine Learning (ML)**, se centra en el desarrollo de algoritmos y modelos que permitan a las computadoras "aprender" de los datos y tomar decisiones sin necesidad de ser programados explícitamente para cada tarea. En términos generales, el objetivo de una aplicación de *Machine Learning* es realizar una predicción mediante modelos matemáticos y algoritmos que permitan identificar patrones en los datos.

Un *dataset* es una colección de pares de datos (X, Y), donde X representa los vectores de características y Y los resultados deseados. A su vez, el dataset se divide en tres partes (Google, s.f.-b): un conjunto de entrenamiento, que se utiliza para entrenar el modelo a partir de los datos; un conjunto de validación, que permite evaluar y ajustar el modelo durante su desarrollo; y un conjunto de prueba, que se reserva para comprobar el rendimiento final del modelo con datos completamente nuevos.

El *tuning* de estos modelos matemáticos es el proceso mediante el cual, utilizando algoritmos de optimización, se busca minimizar la discrepancia entre los resultados obtenidos a partir del modelo y los resultados conocidos de los datos. El modelo resultante debe ser capaz de generalizar, es decir, de realizar predicciones precisas en nuevos datos que no fueron parte del conjunto de entrenamiento.

A continuación se presentan algunos conceptos y componentes claves del área de ML (Jung, 2022a):

- **Datos:** Son la base de cualquier problema de Aprendizaje Automático, dentro del conjunto de los datos podemos identificar a los *data points* individuales que los modelos de ML usan para aprender patrones. Estos *data points* incluyen *features* (entradas) y pueden incluir *labels* (salidas) conocidas dependiendo del modo de entrenamiento del modelo. Es interesante resaltar que la calidad y cantidad de los datos son cruciales, ya

que influyen directamente en la capacidad del modelo para "generalizar" correctamente.

- **Modelo:** Dada una aplicación de ML que genera *data points*, cada uno caracterizado por *features* $x \in X$ y *label* $y \in Y$, el principio de funcionamiento del modelo es aprender un mapa de hipótesis $h : X \to Y$ que cumpla $y \approx h(x)$ para cualquier *data point*. Es una representación matemática o estadística que intenta capturar los patrones en los datos y su implementación específica puede variar desde simples regresiones lineales hasta complejas redes neuronales profundas.

- **Pérdida:** Todos los métodos de ML utilizan un espacio de hipótesis que consiste en todos los mapas de predicción posibles. Para determinar cuál de estos mapas es el mejor, se utiliza una función de pérdida, que mide la discrepancia entre la *label* verdadera y la predicción. El objetivo es encontrar una hipótesis que minimice esta pérdida. La elección de la función de pérdida es crucial y depende de aspectos computacionales, estadísticos e interpretativos.

Clasificación

Dado que cualquier método de ML debe implementarse con recursos computacionales finitos, el mismo sólo podrá considerar un subconjunto de todos los posibles mapas de hipótesis, llamado espacio de hipótesis o modelo subyacente de un método de ML. Según la forma en que los métodos de ML evalúan la calidad de los diferentes mapas de hipótesis, se distinguen tres variantes principales de ML: Aprendizaje Supervisado, no supervisado y de refuerzo (en inglés *Supervised*, *Unsupervised* y *Reinforcement Learning*).

Aprendizaje Supervisado El **Aprendizaje Supervisado** (Jung, 2022b) se basa en datasets etiquetados (o *labeled datasets*) para entrenar los algoritmos que se utilizarán para realizar la predicción. Un *data point* se encuentra etiquetado (o *labeled*) si se conoce el valor correspondiente del *label*, valor que puede obtenerse de expertos humanos que asignan (etiquetan) estos valores a los datos. El Aprendizaje Supervisado busca una hipótesis que pueda replicar el proceso de anotación realizado por humanos y permita predecir la *label* basándose únicamente en las características (*features*) del *data point*.

Generalmente se divide en dos categorías: clasificación y regresión.

Clasificación Este tipo de enseñanza supervisada utiliza un algoritmo para asignar de manera precisa los datos de prueba a categorías específicas (Google Cloud, s.f.). Algunos algoritmos de clasificación comunes son los clasificadores lineales, *support vector machine* (SVM), los árboles de decisión, el k-vecinos más próximos y el *random forest*.

Regresión Regresión se utiliza para entender la relación entre dos o más variables, se enfoca en encontrar una relación matemática entre las variables de entrada y una variable de salida continua (Google Cloud, s.f.). Los algoritmos más utilizados dentro de regresión son regresión lineal, regresión polinómica y regresión logística.

Aprendizaje No Supervisado Algunos métodos de ML no requieren conocer el valor de la *label* de los datos y se conocen como métodos de **Aprendizaje No Supervisado** (Jung, 2022b). Estos métodos dependen únicamente de la estructura interna de los datos para formular una hipótesis efectiva, sin la necesidad de un experto que proporcione etiquetas para entrenar el modelo. Existen dos categorías principales dentro de Aprendizaje No Supervisado: ***clustering*** y ***feature learning***.

Clustering *Clustering* (IBM, s.f.) agrupa datos en subconjuntos o clústers, de manera que los puntos dentro de un mismo clúster son más similares entre sí que con los puntos de otros clústeres. El objetivo es descubrir patrones o estructuras subyacentes en los datos, permitiendo identificar segmentos o grupos en un conjunto de datos sin supervisión previa.

Feature Learning *Feature learning* (IBM, s.f.) automatiza la creación de representaciones útiles de los datos. En lugar de utilizar *features* predefinidas, estos métodos transforman los datos crudos en un nuevo conjunto de características que capturan mejor la información relevante. Esto permite un análisis más eficiente y facilita tareas como la reducción de dimensionalidad y la visualización de datos, mejorando el rendimiento de los algoritmos de Aprendizaje Automático.

Aprendizaje Por Refuerzo El **Aprendizaje Por Refuerzo** (Jung, 2022b) estudia aplicaciones donde las predicciones de una hipótesis influyen en la generación de datos futuros. En este tipo de aprendizaje automatizado, un agente autónomo aprende a tomar decisiones mediante prueba y error, sin la necesidad de datos etiquetados o supervisión directa. Similar al Aprendizaje No Supervisado, los métodos de Aprendizaje Por Refuerzo deben aprender una hipótesis sin acceso a datos etiquetados.

Una manera de dividir Aprendizaje Por Refuerzo es en ***online*** y ***offline***.

Online En este enfoque, el agente obtiene datos interactuando directamente con su entorno (Simonini, 2023). La información se recopila y procesa de manera iterativa a medida que el agente sigue interactuando con el entorno.

Offline Si un agente no tiene acceso directo a un entorno, puede aprender utilizando datos previamente registrados de dicho entorno (Simonini, 2023). Este método se ha vuelto cada vez más popular debido a las dificultades prácticas

asociadas con el entrenamiento de modelos a través de la interacción directa con entornos reales.

Estrategias de Aprendizaje Automático

A continuación se presentarán los algoritmos de Aprendizaje Automático relevantes para este trabajo. Específicamente, se hace foco en las redes neuronales y en la arquitectura de transformadores, conceptos que son esenciales para el entendimiento del proyecto.

Redes Neuronales Una **Red Neuronal**, o **Neural Network** (**NN**) en inglés, es un programa o modelo de *Machine Learning* que tiene como objetivo la toma de decisiones utilizando un proceso que busca imitar el funcionamiento cerebral, en particular, como las neuronas se conectan y comunican entre sí para llegar a conclusiones (IBM, s.f.-a).

Si bien las redes neuronales no son un algoritmo en sí, estas son una arquitectura fundamental en el campo del aprendizaje automático. Principalmente enmarcadas dentro del *Deep Learning*, las redes neuronales han demostrado ser particularmente eficaces en el manejo de grandes volúmenes de datos y en la identificación de patrones.

A continuación se explica, de manera resumida, el funcionamiento de las redes neuronales entrenadas mediante aprendizaje supervisado y con la estrategia de retropropagación. La información presentada se ha extraído del libro *Neural Networks and Deep Learning* de A. Nielsen (2015) (Nielsen, 2015) y se puede encontrar una versión extendida de esta explicación en el documento "Anexo del Marco Teórico: Redes Neuronales". Para esto, se presenta una estructura básica, similar a la de las **Redes Neuronales Prealimentadas** o ***Feed Forward Neural Networks*** en inglés, que sirve para definir conceptos generales sobre el funcionamiento de las mismas, los cuales son aplicables a otros tipos de redes neuronales.

Las **neuronas** son la unidad fundamental de una red neuronal artificial. Su objetivo es imitar conceptualmente el funcionamiento de las neuronas biológicas mediante un modelo matemático que puede variar dependiendo del tipo de red.

Las neuronas se agrupan en **capas**, existiendo una capa de entrada, una de salida y un conjunto de capas ocultas o "hidden layers". A su vez, la conexión entre neuronas tiene un **peso**, un valor positivo o negativo que se multiplica por el valor de salida de cada neurona y modifica los valores de salida de una capa en relación con los valores de entrada de la siguiente. Por otra parte, el ***bias*** es un valor utilizado para determinar cuándo una neurona debe activarse significativamente en función de los valores recibidos de la capa anterior.

Proceso de aprendizaje supervisado En esta parte, se presenta de forma general cómo se realiza el proceso de entrenamiento de una red neuronal. En particular, la siguiente sección hace foco en el proceso de aprendizaje super-

visado en una red neuronal utilizando *backpropagation*, que a pesar de no ser la única metodología aplicable, es una de las más extendidas.

El proceso consiste en la aplicación recursiva de un procedimiento por el cual los valores de pesos y *biases* son ajustados para lograr minimizar las diferencias entre los valores obtenidos y los esperados en un conjunto de datos de prueba.

El objetivo es lograr que luego de este proceso la red sea capaz de recibir cualquier entrada por fuera del conjunto utilizado para su entrenamiento y logre predecir correctamente la salida. Cabe destacar que la eficacia dependerá no solo del proceso de aprendizaje sino del *set* de entrenamiento utilizado.

Función Costo

Esta función se define como la distancia entre los valores obtenidos y los esperados para un conjunto de datos de prueba (Jung, 2022a). Para obtener una medida de la *performance* de la red es necesario no solo tener en cuenta este valor para una única entrada sino que se debe ver el promedio aplicado a todo el set de entrenamiento.

Minimización de la función costo

Una vez obtenida una función costo se busca encontrar el punto donde el costo sea mínimo, es decir que los valores obtenidos se acerquen lo más posible a los correctos (Jung, 2022a). Sin embargo, las variables de la función costo serán tantas como pesos y *biases* tenga la red por lo que la dimensionalidad puede dificultar hallar de forma exacta un mínimo.

En estos casos, una alternativa factible consiste en comenzar con una entrada variable e identificar en qué dirección se deben mover esas variables para hacer que ese valor de salida sea menor (encontrar un mínimo local). Si este mínimo local encontrado es el menor posible o no dependerá de los valores iniciales.

En un plano de varias dimensiones el problema se reduce a encontrar el vector que represente la dirección en que esta función decrece, es decir el **opuesto al gradiente**. El algoritmo para minimizar esta función puede resumirse en computar este gradiente, tomar un paso en la dirección opuesta y repetir el proceso hasta encontrar un mínimo local. A esto le llamamos ***gradient descent***. El procedimiento encargado de computar este gradiente eficientemente es llamado ***backpropagation***.

Backpropagation

Partiendo de un valor para las neuronas de una capa que compone la red en base a un ejemplo de entrada dado, existen tres valores que podemos modificar para alterar su resultado y acercarlo al esperado: los pesos, los *biases* y los resultados de las capas anteriores (IBM, 2025).

Los cambios en el ***bias*** son los más simples ya que su efecto en el valor de la neurona es lineal. Si el *bias* aumenta, también lo hace la activación de la neurona conectada y viceversa.

Cambios en los **pesos** implica el fortalecimiento o debilitamiento de la conexión entre dos neuronas. A mayor valor, mayor influencia tendrá sobre una neurona la activación de la neurona de la capa anterior a la que se encuentra conectada. Sin embargo esta relación no es tan lineal como en el caso anterior.

Cambios en **activación** de neuronas previas por su parte implican aumentar

o disminuir la activación de la neurona a la cual se conectan, dependiendo del peso que las une.

El proceso de *backpropagation* consta de aplicar el mismo procedimiento de minimización de manera recursiva desde la capa de salida hacia las capas anteriores, intentado hacer los cambios necesarios en la capa anterior para lograr los valores esperados en la siguiente.

Transformadores Los **Transformadores**, o ***Transformers***, son un tipo de arquitectura para modelos de *Deep Learning* que nacen como alternativa de los modelos convolucionales y recurrentes. Esta arquitectura fue presentada en "*Attention is All You Need*" (Vaswani y cols., 2017), que se trata de uno de los papers con más influencia en los desarrollos recientes dentro del mundo de la Inteligencia Artificial.

Al momento de ser planteada, la nueva arquitectura marcó un nuevo estado del arte en su objetivo inicial de traducción de lenguajes y desde entonces ha sido el motivo de grandes avances en todas las áreas de la Inteligencia Artificial.

La característica principal de estos modelos es que se basan en el concepto de **mecanismos de atención** (***attention mechanisms***) introducido en 2014 por Dzmitry Bahdanau, Kyunghyun Cho y Yoshua Bengio (Bahdanau, Cho, y Bengio, 2014). Estos mecanismos le dan a los transformadores una mayor capacidad a la hora de capturar dependencias en los datos de entrada que sus antepasados tecnológicos, ademas de hacerlos paralelizables y disminuir su tiempo de entrenamiento.

En el contexto de este documento, no resulta relevante entrar en los pormenores de la implementacion de los transformadores, el documento "Anexo del Marco Teórico: Attention Is All You Need" contiene una explicación en mayor detalle de las características técnicas de los mismos.

2.1.2. Procesamiento de lenguaje natural

El **Procesamiento del Lenguaje Natural** (PLN), en inglés **Natural Language Processing (NLP)**, es el área de la Inteligencia Artificial que tiene como objetivo lograr construir computadoras con la capacidad de manipular lenguaje humano, tanto escrito como hablado.

El procesamiento del lenguaje natural puede ser subdividido en dos grandes áreas, la Comprensión del Lenguaje Natural (**CLN**) y la Generación del Lenguaje Natural (**GLN**) (Khurana, Koli, Khatter, y Singh, 2022). La primera de ellas, CLN, se encarga de entender el lenguaje y analizarlo, extrayendo conceptos, reconociendo entidades y emociones. Por otro lado, GLN busca producir frases, sentencias y párrafos que tengan sentido.

Dada la amplitud del objetivo del campo del PLN esta es un área que, a lo largo de los años, ha intentado resolver una variedad de problemas, algunos de estos son:

- Traducción automática (*Machine translation*)

- Resumen automático (*Automatic summarization*)
- Análisis de discurso (*Discourse analysis*)
- Segmentación morfológica (*Morphological segmentation*)
- Reconocimiento de entidades nombradas (*Named entity recognition, NER*)
- Etiquetado de partes del discurso (*Part of speech tagging, POS tagging*)

Los objetivos centrales del PLN implican tanto interpretación como análisis y manipulación de datos de lenguaje natural con varios algoritmos, herramientas y métodos. Es esta variedad de factores que hacen que sea extremadamente complejo alcanzar todos los objetivos con un solo enfoque, por lo que el desarrollo e investigación del campo de PLN ha recibido la atención de varias áreas de la ciencia con distintas aproximaciones en los últimos años.

A comienzos de los 2000 se comenzó el trabajo en la aplicación de modelos de redes neuronales al procesamiento del lenguaje natural, teniendo como ejemplo a Bengio et al. (Bengio, Ducharme, Vincent, y Jauvin, 2000) que usando redes neuronales prealmientadas determinaron, dada una secuencia de palabras, la probabilidad de la siguiente.

Por otra parte, en 2008 Collobert et al. (Collobert y Weston, 2008) lograron aplicar *multitask learning*, un paradigma de aprendizaje que aprovecha de manera eficaz la información compartida entre tareas para abordarlas de manera simultánea, a una red neuronal convolucional que dada una sentencia aplica varias conceptos de PLN, como pueden ser *POS tagging* y *named entity recognition*.

Siguiendo con esta línea de desarrollos en diferentes dimensiones del PLN, en 2013 Mikolov et al. desarrollaron el modelo *word2vec* (Mikolov, Sutskever, Chen, Corrado, y Dean, 2013). Que es un acercamiento al concepto de los *word embeddings*, una representación vectorial de las palabras en un espacio vectorial de dimensión alta que permite a los modelos capturar de manera precisa la relación sintáctica y semántica entre palabras.

Como se puede ver en las investigaciones que se nombran anteriormente, el uso de redes neuronales ha cumplido un rol central en el desarrollo del procesamiento de lenguaje natural a lo largo de los últimos años.

Recientemente, el avance en el PLN se vio fuertemente influenciado por las grandes mejoras en el área del Aprendizaje Computacional, particularmente por el uso de arquitecturas de transformadores.

Modelos Grandes de Lenguaje Como una evolución natural de los modelos de lenguaje pre-entrenados aparecen los **Modelos Grandes de Lenguaje** o **Large Language Models (LLMs)** (Contributors, 2025), algunos ejemplos son BERT, GPT, LLAMA, PaLM, etc.

Estos modelos de lenguaje son capaces de comprender y generar texto de forma similar a un ser humano, lo que les permite traducir, resumir, responder

preguntas, ayudar en tareas creativas o de programación, entre otras capacidades. Estas aptitudes a la hora de manipular el lenguaje natural los han convertido en una parte fundamental de los sistemas de Inteligencia Artificial modernos.

Desde un punto de vista técnico, los LLMs son modelos de aprendizaje profundo basados en transformadores que tienen decenas o centenas de billones de parámetros y son pre-entrenados con enormes *datasets* de texto.

A grandes rasgos, el proceso de entrenamiento de estos modelos busca que logren predecir la siguiente palabra de una sentencia a partir del contexto. Este proceso se lleva a cabo utilizando corpus de texto masivos, logrando que estos adquieran conocimientos de gramática, semántica y relaciones conceptuales entre palabras mediante aprendizaje auto-supervisado y *zero-shot*. El resultado son modelos con capacidades de generación de lenguaje coherente y relevante al contexto (IBM, s.f.-b).

Generación Aumentada con Recuperación (Amazon Web Services, 2025)

Al referirnos a esta técnica, más conocida como ***RAG*** por sus siglas del inglés ***Retrieval Augmented Generation***, se hace referencia a un enfoque híbrido que enriquece las capacidades de un LLM combinándolo con sistemas de búsqueda de información externos. En lugar de depender únicamente del conocimiento almacenado en sus parámetros, uno de estos sistemas primero consulta una base de datos relevante para luego usar esos pasajes recuperados como contexto adicional al generar la respuesta.

En términos operativos, un sistema RAG consta de dos módulos principales: un buscador, o *retriever*, y un generador de texto. El buscador toma la consulta del usuario, la convierte en vectores y busca los pasajes más relevantes dentro de la base de conocimiento, devolviendo un conjunto de documentos relacionados. Luego, el generador, que suele ser un modelo de lenguaje de gran escala, recibe esos pasajes junto con la pregunta original y produce la respuesta final basándose en esta información enriquecida.

El uso principal de estas herramientas se da cuando es necesario incorporar durante la generación de una respuesta, información propia del problema a resolver que el modelo desconoce, guiar de forma más precisa las respuestas generadas o, simplemente, lograr una mayor eficiencia y ahorro de recursos al evitar el uso de contextos extensos e innecesarios.

Modelos de Razonamiento Dentro del mundo de los LLMs han ganado popularidad en el último tiempo los llamados **Modelos de Razonamiento** o ***Reasoning Models*** en inglés.

Los modelos de razonamiento son LLMs que difieren de los tradicionales en la forma en que procesan preguntas e instrucciones. Generan una cadena de razonamiento interna, es decir, un conjunto de pasos intermedios, antes de ofrecer una respuesta final. Gracias a este enfoque, son capaces de descomponer el problema en partes más pequeñas, explorar diferentes perspectiva, seleccionar los más adecuados, descartar los inválidos y, finalmente, elegir la mejor respuesta posible (Mahto, 2024).

Esta capacidad les permite analizar con mayor profundidad problemas analíticos y de múltiples etapas, a diferencia de un LLM generalista, que por defecto tiende a producir respuestas directas basadas en correlaciones textuales (Kelly, 2024).

2.1.3. Visión Computacional

La **Visión Computacional**, en inglés **Computer Vision**, es el campo de la Inteligencia Artificial especializado en el entrenamiento de computadoras para la identificación o generación de información en base a una fuente visual como puede ser un video o fotografía, por lo general mediante el uso de técnicas de Aprendizaje Automático y Redes Neuronales. ('IBM', s.f.)

Tiene como objetivo dotar a un sistema informático de la capacidad de observar y comprender lo observado como lo hace un ser humano para luego poder accionar en base a ello.

En el contexto de este documento es pertinente mencionar un problema especifico de esta área de la Inteligencia Artificial que será mencionado más adelante: el Reconocimiento Óptico de Caracteres.

El **Reconocimiento Óptico de Caracteres**, en ingles **Optical Character Recognition (OCR)**, es el proceso que se aplica a imágenes que contienen texto, ya sea impreso o manuscrito, para convertirla en un formato de texto que pueda ser consumido por una computadora ('Amazon', s.f.).

Esta tecnología es ampliamente utilizada como una forma de consumir datos de fuentes impresas, como pueden ser formularios, facturas o documentos, para que luego estos puedan ser utilizados en otros procesos computacionales, ya sea su edición, traducción, indexado, etc.

La primeras versiones de estas tecnologías precisaban ser entrenadas con imágenes de cada carácter y trabajan con una fuente de texto por vez. Actualmente existen sistemas capaces de funcionar con una variedad de fuentes, incluso texto manuscrito, y que son capaces de mantener información textual relevante, como pueden ser las imágenes o el formato de los documentos.

2.1.4. Inteligencia Artificial en educación

La aplicación de la IA en el ámbito educativo, conocida como **AIEd**, emerge rápidamente como un instrumento de suma importancia, abriendo paso a una nueva era de aprendizaje personalizado, mayor participación y mejores herramientas basadas en datos para enriquecer la experiencia de los estudiantes y profesores.

Uno de los informes del proyecto Nesta (Baker, Smith, y Anissa, 2019) identifica tres categorías principales de estas herramientas: aquellas orientadas al estudiante (por ejemplo, plataformas de aprendizaje adaptativo), orientadas al docente (por ejemplo, herramientas de evaluación automatizada o paneles avanzados para profesores) y una tercera categoría enfocada en la gestión del sistema educativo en su conjunto (por ejemplo, análisis de datos escolares para predecir desempeños).

A continuación se presentaran, de forma resumida, algunas de las áreas mas relevantes dentro del AIEd. En concreto, se hará foco en aquellas que sean relevantes a este proyecto, aunque es posible encontrar una investigación en más amplia en el documento "**Marco Teórico**".

Agentes Educativos

Los **Agentes Educativos**, en inglés **Educational Agents**, son sistemas educacionales con características humanas que tienen el objetivo de facilitar el aprendizaje, siendo capaces de expresarse mediante texto, gráficos, íconos, voz, animaciones, multimedia o incluso realidad virtual (Chou, Chan, y Lin, 2003).

Para clasificar estos agentes es posible utilizar su rol y función, por un lado se definen los **agentes pedagógicos** y por el otro se encuentran los **asistentes personales**.

Dos aplicaciones típicas de los agentes pedagógicos son los ITS (del inglés *Intelligent Tutoring System*) y los LCS (del inglés *Learning Companion System*).

Los ITS buscan simular el rol de un profesor que es experto en el dominio y actúa como tutor, tomando un rol más autoritario.

En cambio, los LCS se enfocan en actividades de aprendizaje colaborativo o competitivo como alternativas a la tutoría tradicional. En lugar de actuar como figuras de autoridad o expertos en un dominio, los agentes pedagógicos en los LCS se presentan como "compañeros de aprendizaje", cuya función principal es interactuar con los estudiantes en un rol de igualdad, fomentando un entorno más dinámico y participativo.

En cuanto a los **asistentes personales**, estos son diseñados para ofrecer a cada usuario, tanto docentes como estudiantes, información personalizada y relevante relacionada con las actividades de aprendizaje.

La implementación de agentes educativos se basa en la necesidad de personalizar la experiencia de aprendizaje para satisfacer las diversas necesidades de los estudiantes. Según Adriana Alexandru et al (Alexandru, Tirziu, Tudora, y Bica, 2015), esto implica diseñar sistemas adaptativos que combinen dos enfoques principales: uno donde el sistema ajusta dinámicamente el contenido según las características del estudiante, y otro que permite al usuario modificar su experiencia para recibir el orden más adecuado de los contenidos. Los agentes educativos, al integrar Inteligencia Artificial, operan como mediadores inteligentes que no solo ofrecen contenido relevante, sino que también promueven la interacción activa y la autonomía en el proceso de aprendizaje.

Sistemas Inteligentes de Instrucción Asistida por Computadora

Los **Sistemas Inteligentes de Instrucción Asistida por Computadora**, conocidos por su nombre en inglés *Intelligent Computer-assisted Instruction* (**ICAI**), son un tipo de agente educativo que representa una de las formas más investigadas de integrar la Inteligencia Artificial en el ámbito educativo (Duchastel y Imbeau, 1988). En este trabajo, se utilizarán indistintamente los términos **Sistemas de Tutoría Inteligente** o **ITS** por sus siglas en inglés

e ICAI tal como lo hace gran parte de la literatura especializada, sin embargo, cabe mencionar que algunos textos hacen distinciones sutiles entre ambos conceptos.

A diferencia de la enseñanza asistida por ordenador tradicional, **CAI** por sus siglas en inglés, los ITS no se limitan a complementar la labor del educador o a servir como una herramienta para consumir conocimientos de forma lineal, sino que su objetivo principal es replicar las capacidades de enseñanza humana. Entre sus principales ventajas se encuentran su flexibilidad y capacidad de adaptarse a las necesidades e intereses específicos de cada estudiante sin requerir una intervención humana directa, lo que en muchos casos las convierte en herramientas clave para la democratización de la enseñanza.

Las características de estos sistemas pueden variar, sin embargo desde la década de 1980 se identifican tres elementos fundamentales que han estado consistentemente presentes en la mayoría de ellos (Tan, Ji, y Jin, 2013). Estos elementos son:

- **Modelo de Datos:** Este es el responsable de almacenar y procesar la base de conocimiento. Su función no se limita a entender los conceptos que se desean enseñar, sino también a identificar las relaciones entre ellos y determinar cómo deben ser presentados para maximizar su efectividad.
- **Modelo del Estudiante:** Esta componente del sistema se encarga de modelar al estudiante y su proceso de aprendizaje. Debe detectar y registrar sus errores, aciertos, el conocimiento adquirido y los conceptos erróneos que pueda tener, con el objetivo de adaptar el proceso educativo a sus necesidades específicas.
- **Modelo del Profesor / Enseñanza:** El modelo del profesor simula las funciones de un docente al adaptarse al proceso de aprendizaje del estudiante. Este modelo utiliza la base de conocimiento a enseñar y los datos registrados por el modelo del estudiante para tomar decisiones informadas que orienten los pasos siguientes en el proceso. Para ello, debe diagnosticar la situación del estudiante, identificar sus puntos débiles y reconocer los conocimientos ya adquiridos. Este análisis es posible gracias al uso de capacidades inferenciales propias de la Inteligencia Artificial, lo que permite una personalización más efectiva en la enseñanza.

Si bien las implementaciones y los usos de estos sistemas son diversos y presentan cada uno sus características particulares, se han identificado a lo largo de los años ciertos desafíos comunes que marcaron su evolución.

A continuación se plantean parte de estos retos. Algunos fueron identificados décadas atrás pero siguen aún siendo relevantes (Dede, 1987), mientras que otros nacen de trabajos más cercanos (Paviotti, Rossi, y Zarka, 2012). En varios de ellos se ha avanzado en los últimos años, sin embargo, aún siguen siendo desafíos presentes.

- **Generalización:** Generalizar este tipo de sistemas para abarcar así problemas por fuera de un dominio acotado y bien definido ha sido siempre

un desafío. Como consecuencia, esto ha dificultado la incorporación de estos sistemas en un contexto educativo amplio donde se abarquen distintas áreas de conocimiento en conjunto. El acceso cada vez más simple a sistemas potentes y capaces de abarcar un dominio de conocimiento mayor se plantea como una posible solución a esta limitación. Sin embargo, obtener resultados satisfactorios se dificulta en sistemas tan masivos y el costo de estos puede ser elevado.

- **Barrera comunicacional:** Distintas herramientas han surgido a lo largo de estos años buscando mejorar la comunicación entre estos sistemas y el estudiante. Sin embargo, en particular en aquellas áreas técnicas como las matemáticas donde incorporar el lenguaje natural no es una tarea intuitiva, se siguen explorando a día de hoy distintas soluciones.

- **Falta de inteligencia emocional:** Otro de los desafíos destacados en los últimos años es la necesidad de dotar a estos sistemas de inteligencia emocional. Aunque este es un tema clave en el ámbito de los ITS, también representa un problema a resolver en el campo más amplio de los sistemas educativos aplicados, constituyendo un factor clave para mejorar su eficacia y aplicabilidad.

Modelado predictivo en educación

La analítica predictiva es el área de la analítica que se encarga de predecir eventos futuros a partir de analizar datos actuales e históricos, empleando técnicas de Minería de Datos e Inteligencia Artificial. Esta área ha crecido significativamente con el constante avance de los sistemas de *big data* y, en general, la aplicación de técnicas de análisis predictivo ayuda a diferentes organizaciones a ser más proactivas y anticipar tendencias o comportamientos basándose en los datos que recaban a lo largo del tiempo.

El análisis predictivo es una herramienta vastamente utilizada en el mundo de la educación, donde hay un claro interés en predecir el rendimiento de los estudiantes, el efecto de métodos de enseñanza específicos o incluso la predicción de factores organizativos como la retención de estudiantes. De hecho, existen varios productos comerciales que incorporan este tipo de tecnologías y compañías especializadas en desarrollarlas.

El modelado predictivo busca generar modelos que sean capaces de evaluar, clasificar y hasta predecir valores específicos para nuevos datos. Esto se logra bajo la suposición de que un conjunto de datos previamente conocidos (*training dataset*) puede ser digerido y utilizado para predecir el valor o categoría de nuevos datos a partir de ciertas variables seleccionadas (*features* o características), que no es más que el enfoque elegido por el aprendizaje automático para generalizar conocimiento.

En el dominio de la educación, los modelos predictivos son una parte esencial en la capacidad de reacción en tiempo real de las instituciones a las actitudes de sus estudiantes, logrando predecir situaciones puntuales y permitiendo a los

entes educativos plantear estrategias para intervenir y remediar problemas antes de que sucedan.

Sin embargo, el análisis predictivo no es una solución viable para cualquier situación. los problemas candidatos de estudio son aquellos en los que intervienen características medibles y cuantificables del sujeto de estudio (ya sean profesores o alumnos). Idealmente, debe ser fácil determinar un rango de valores deseables, debe ser posible afectar estas métricas con acciones concretas en la realidad y el set de datos que se utiliza debe ser lo suficientemente grande y variado como para asegurar que no haya sesgos y las predicciones sean correctas.

Una consideración importante a la hora de poner en práctica un modelo predictivo es la necesidad de similaridad entre los datos de entrenamiento del modelo y los datos sobre los que se intenta predecir. Es común en la educación utilizar modelos entrenados con datos de un período (año, semestre) y usarlo sobre la realidad del periodo siguiente, sin embargo si la realidad subyacente de los periodos es suficientemente distinta, ya sea porque cambia el programa de los cursos o la forma en la que son dictados, el modelo no va a lograr predecir correctamente contra los nuevos datos.

En esta última década han empezado a aparecer sistemas robustos que brindan herramientas de modelado predictivo a las instituciones educativas, tanto para datos en enseñanza como de aprendizaje. Este aumento de interés en el análisis predictivo también genera una necesidad de aumento de la calidad de la investigación alrededor de las tecnologías, las metodologías y el impacto que tiene su aplicación. Tanto Meenakumari (J Meenakumari, 2015), como Ferguson (Ferguson, 2012) discuten e identifican desafíos y oportunidades de mejora en el área, como pueden ser:

- **Consideraciones éticas:** El área de la analítica y sus investigadores suelen centrarse principalmente en el lado técnico de la recolección y la utilización de datos, dejando de lado cuestiones éticas relacionadas a la captura y uso de los mismos. Entonces, surgen cuestiones de protección y privacidad de datos, donde estudiantes y profesores no necesariamente son dueños de sus datos y desconocen quién puede estar teniendo acceso a estos sin su consentimiento.

- **Relación estudiante-docente:** El uso de modelos predictivos puede generar una simplificación de la relación que existe entre estudiantes y la enseñanza. Los modelos logran capturar la situación actual educativa de los estudiantes priorizando su suceso en términos de datos, pero ignoran la complejidad de la conexión entre estudiantes, profesores e instituciones educativas, corriendo el riesgo de generar rispideces. Las instituciones se enfrentan al reto de mantener sistemas de análisis que les permitan idear intervenciones valiosas y efectivas para sus estudiantes y tutores, que se deben alinear con necesidades pedagógicas que no siempre son tan visibles en un conjunto de datos.

- **Conexión con ciencias de la educación:** Investigadores del área de modelado predictivo aplicado a educación suelen dejar de lado las ciencias

cognitivas, la metacognición y las necesidades pedagógicas de los actores. Para optimizar la educación a través de datos es necesario tener un buen entendimiento de cómo sucede el aprendizaje, qué factores entran en juego y cómo puede ser acompañado, es por eso que la investigación de la analítica aplicada a ambientes educativos debe tender puentes con las ciencias de la educación.

- **Sesgos:** Existe una gran preocupación en cuanto a la posibilidad de que algunos modelos se vuelvan sesgados en contra de ciertas categorizaciones de personas, como puede ser la etnicidad o el género (Gándara, Anahideh, Ison, y Picchiarini, 2024). Por esto, ya existen ciertas conferencias (ACM, FAccT) y grupos de investigación que se encargan de mitigar este riesgo, buscando maneras de medir el sesgo (Tempelaar, Rienties, y Nguyen, 2020), el impacto de las decisiones de los usuarios sobre el mismo (W. Li, Brooks, y Schaub, 2019) y hasta medirlo en los métodos subyacentes para entender su impacto. Sin embargo, aún resta entender como la evidencia de sesgo debería afectar al uso de los modelos y la reacción que se debe tener ante evidencias del mismo.

Diseño de evaluaciones

Las evaluaciones forman una parte central e indispensable en las actividades pedagógicas, puesto que brindan a estudiantes y profesores información crucial a la hora de entender el estado del proceso educativo que se lleva a cabo. Los estudiantes pueden utilizar esta información para poder ajustar sus esfuerzos en base a resultados, mientras que las entidades educativas y los profesores pueden medir la efectividad de los recursos utilizados, reevaluar y ajustar las técnicas empleadas y hasta identificar grupos de estudiantes que necesiten seguimiento.

Según Popham (Popham, 2009) el término evaluación se define como *"una amplia variedad de técnicas utilizadas para recabar evidencia"*. Esta definición es interesante ya que muestra que no sólo instancias de pruebas formales, como cuestionarios o exámenes, son consideradas evaluaciones y que el objetivo de una evaluación no siempre es asignar una calificación.

McMillan (McMillan, 2007) define dos tipos de evaluaciones:

- Sumativas: Son aquellas evaluaciones que buscan medir el resultado de eventos educativos ya completados, como puede ser decidir sobre la promoción de un estudiante luego de un año escolar.

- Formativas: Son las evaluaciones que tienen como fin permitir a profesores y estudiantes entender cuál es la mejor estrategia educativa. Su objetivo es el de mejorar los procesos elegidos, estimulando cambios que ayuden al progreso de los mismos.

Dado que los procesos evaluativos son una parte fundamental en la formación estudiantil y afectan tan profundamente al proceso educativo, su diseño e implementación debe hacerse con mucho cuidado. Es por esto que las evaluaciones

son procedimientos altamente costosos, tanto a nivel de recursos humanos de los profesores como financieros para las instituciones (Divate y Salgaonkar, 2017). Este elevado costo hace que haya una motivación clara por intentar incluir a la tecnología en este proceso, desde la creación de preguntas hasta la construcción de pruebas o incluso la automatización completa del proceso de evaluación.

El problema de la **generación automática de preguntas** puede ser abordado de diferentes maneras y con diferentes algoritmos. Algunas propuestas logran crear preguntas utilizando información sintáctica del texto original, mientras que en sus versiones más sofisticadas y nuevas las preguntas son generadas en base a información semántica.

Los sistemas clásicos de generación de preguntas dependen fuertemente de reglas predefinidas, requiriendo configuraciones manuales y empleando grandes cantidades de tiempo por parte de los educadores. Además, al basarse en información sintáctica del material, su capacidad de adaptabilidad es reducida y desconoce de relaciones que pueden quedar escondidas en los textos empleados. En cambio, las aplicaciones más modernas que se basan en IA y LLMs logran subsanar algunas de estas limitaciones mencionadas, dando lugar a la creación de pruebas personalizadas y mejorando no solo la eficiencia a la hora de crear preguntas sino que también pueden llegar a mejorar los resultados y la calidad de las evaluaciones.

De manera genérica, las preguntas pueden ser divididas en 2 grupos:

- Preguntas objetivas, aquellas que le piden al evaluado elegir un respuesta entre una lista de alternativas o dar una respuesta breve de una palabra o una frase. Son preguntas con una única respuesta correcta, algunos ejemplos de estas son: Múltiple opción, verdadero-falso, *fill in the blanks*, etc.

- Preguntas subjetivas, aquellas en las que el evaluado tiene que componer una respuesta en sus propias palabras. Son preguntas donde no es directo saber si la respuesta es correcta y sus dos ejemplos más comunes son preguntas de respuesta corta (un párrafo) o de respuesta larga (varios párrafos).

A lo largo de los años, las generación de preguntas objetivas han sido el foco de atención de los investigadores, con principal interés en preguntas múltiple opción y *fill in the blanks* (*cloze* y *open-cloze*). De manera general, la creación de cualquiera de estos tipos de preguntas comienza con la selección de una frase informativa del texto de referencia a la cual se le omite una o más partes, desde ahí es simple definir cuál es la opción correcta o saber si la respuesta de una estudiante es correcta. Sin embargo, lo que resulta complejo es seleccionar qué parte del texto omitir y la generación de distracciones en las opciones ya que estas tareas requieren de una entendimiento más profundo del tema a tratar.

En cambio, en cuanto a preguntas subjetivas la investigación es más limitada y reciente, lo que se puede adjudicar a la complejidad del problema y la falta de herramientas que puedan ayudar a los investigadores en esta tarea. De hecho, no solo es complicada la generación de preguntas, ya que para corregirlas no solo

se requiere de la capacidad de entender la respuesta desde un punto de vista semántico sino que es necesario poder hacer una valoración de su contenido, lo cual hace de la corrección uno de los mayores desafíos.

Así como el proceso de evaluaciones es crucial en la salud de la educación, la automatización del mismo tiene sus posibles desventajas o desafíos (Swiecki y cols., 2022), algunos de ellos pueden ser:

- **Control del proceso evaluativo:** Un problema potencial que nace con el avance de las investigaciones en esta dirección es que aparece la posibilidad de que los profesores tengan cada vez menos control sobre el proceso evaluativo. Con la adopción de este tipo de tecnologías se podría empezar a confiar más en decisiones automatizadas que en opiniones formadas por profesionales, o incluso peor, los profesionales de la educación podrían empezar a confiar en sobremanera en herramientas automatizadas perdiendo parte de su capacidad de tomar decisiones informadas y con bases claras.

- **Distanciamiento de docentes:** La implantación de tecnologías que se hacen cargo del proceso de evaluación da lugar al posible distanciamiento de los profesores de este proceso, puesto que puede ser conveniente para todas las partes: A nivel institucional existe una clara reducción de costos y riesgos, los profesores podrían ver con buenos ojos evitar tener que calificar a estudiantes con los cuales forman una relación y conviven, mientras que los estudiantes podrían beneficiarse de no tener que sufrir la vulnerabilidad que genera conocer a sus evaluadores de primera mano y las posibles fricciones que nacen de esto. Sin embargo, es importante recordar que los actores que forman parte de la educación son personas con su propio historial socioeconómico y cultural, con una experiencia educativa y valores personales, que no necesariamente son cosas que es deseable eliminar de un proceso tan importante como lo es el de la evaluación

- **Consideraciones pedagógicas:** Es importante mantener el rol pedagógico de las evaluaciones, teniendo en mente que el aumento de la presencia de la Inteligencia Artificial no debería interferir con la utilización de la evaluación como herramienta didáctica. Como fue mencionado anteriormente, las evaluaciones pueden ser utilizadas para más que investigar si un cierto alumno aprendió qué conceptos, sino que pueden ser utilizadas para calibrar el proceso educativo, sin olvidar que pueden afectar directamente a la motivación de los alumnos.

- **Sesgos:** Las herramientas de Inteligencia Artificial no necesariamente son diseñadas, o entrenadas, pensando en todas las formas de enseñar y aprender posibles o en todos los posibles tipos de estudiantes que existen. Existe la posibilidad de que su utilización genere un sesgo en contra de ciertos tipos de estudiantes o de propuestas educativas, se podría pensar que cada alternativa que implemente IA también traería consigo una modalidad de aprendizaje preferida, un cierto esquema de prioridades en cuanto a conocimientos, un modelo disciplinario o cualquier característica intrínseca

de los datos con los cuales trabaja. Lo más interesante, es que estas "preferencias" propias de los sistemas de Inteligencia Artificial podrían llegar a afectar al sistema en el que son implantados, al ser inculcadas en sus estudiantes mediante las evaluaciones.

2.2. Elección de Implementación

Tras el análisis del estado del arte en el área de la Inteligencia Artificial y su aplicación en el ámbito educativo, esta sección se enfoca en el proceso de selección de una propuesta concreta con foco en el nivel educativo terciario y énfasis en la formación en ingeniería.

El proceso se divide en tres etapas:

Búsqueda de posibles ideas A partir de un análisis de las distintas áreas en la aplicación de la IA al campo educativo y nuestra experiencia como estudiantes universitarios, se identifican diversas alternativas que, en una primera evaluación subjetiva, aparentan ser implementables y ofrecer algún grado de valor, lo cual motiva su posterior análisis en profundidad.

Definición de los criterios y prioridades Una vez planteadas las distintas opciones, se establecen los criterios para evaluar su viabilidad, el valor que aportan y el grado de interés que generan, con el objetivo de articular los objetivos del proyecto, los intereses del equipo de trabajo y la factibilidad técnica de su implementación.

Si bien puede resultar contraintuitivo definir criterios de evaluación una vez identificadas las ideas iniciales, esta decisión tiene como propósito evitar restringir prematuramente las opciones. En un campo tan amplio, propuestas que en un principio parecen poco factibles o acordes al proyecto pueden revelarse viables tras una investigación más profunda o ser útiles para conjugar una idea final.

Selección final Definido el conjunto de ideas y los criterios de evaluación, se procede a elaborar una propuesta final, fundamentando debidamente su elección. Paralelamente, se justifican las razones por las cuales se descartan las demás alternativas y se identifican aquellas que podrían resultar de interés para trabajos futuros.

2.2.1. Búsqueda de posibles ideas

Luego de una larga investigación acerca del estado del arte de la Inteligencia Artificial y en particular de su uso en el campo educativo, cuyos hallazgos se encuentran resumidos en la sección anterior y en mayor detalle en el Marco Teórico adjunto a este informe, cada integrante inició un proceso de reflexión sobre posibles propuestas. Este proceso concluyó con una puesta en común, de la cual surgieron las siguientes ideas preliminares.

Asistente para la búsqueda de material

A partir de dudas planteadas por los estudiantes o el uso de palabras clave, se propone el desarrollo de un sistema capaz de sugerir material relacionado, utilizando como fuente los recursos disponibles en el curso (EVA, libros, OpenFing, entre otros).

Variaciones de la idea original proponen la posibilidad de implementar distintas modalidades de interacción, como chatbots o motores de búsqueda, y se podría considerar fuentes externas, como la web, para enriquecer recomendaciones.

En base a la investicación planteada en el marco teórico, se podría decir que se trata de un **Agente Educativo**, en particular un **asistente personal**, enfocado en el estudiante y que desempeñaría un papel importante en el proceso de aprendizaje sin intervenir de una forma tan directa en el mismo.

Índice automático en vídeos OpenFing

Partiendo de la necesidad de un acceso ágil a la información dentro de un OpenFing y enmarcada nuevamente en el campo de los **asistentes personales**, esta propuesta busca identificar los distintos temas tratados y secciones de una clase grabada de forma automática.

Esto permitiría un acceso más eficiente al material, sin necesidad de trabajo extra por parte de aquellos que generan o editan el contenido.

Evaluación automatizada post OpenFing

En búsqueda de otras formas de nutrir la plataforma OpenFing y la experiencia que ofrece, la propuesta tiene como eje la generación automatizada de una evaluación al momento de terminar de ver una clase, lo que podría ser acompañado de una recomendación de puntos de la clase a repasar y sus respectivos minutos.

En este caso podemos hablar de una idea que abarca los campos de **Diseño de Evaluaciónes** y **Tutoría Inteligente**, puesto que no solo genera evaluaciones para el estudiante sino que de un modo un tanto simplificado intenta guiar al mismo en su proceso de aprendizaje mediante recomendaciones.

Diseño de cuestionarios

Esta propuesta puede entenderse como dos propuestas distintas según su fin, viéndola como una ayuda para la generación de cuestionarios para un profesor o como una herramienta para la generación de autoevaluaciones por parte del estudiante.

En ambos casos, la idea consiste en la posibilidad de seleccionar un tema o conjunto de ellos y generar automáticamente una evaluación en base al material del curso.

En el caso de usarse como método de autoevaluación, puede verse mejorado con un mecanismo que detecte dificultades del estudiante al responder y que

pueda o bien guiar el proceso de práctica o sugerir material a revisar. Debido a estas características podemos clasificar también a esta idea como un sistema híbrido que comparte características de **Diseño de Evaluaciónes** y **Tutoría Inteligente**.

Análisis de la calidad de las evaluaciones

Ante la necesidad de una estandarización de las evaluaciones en muchos de los cursos impartidos en la facultad, se propone el desarrollo de un sistema que permita evaluar distintos aspectos como la dificultad, disponibilidad del material evaluado en el EVA del curso, apego a un formato o similitud con evaluaciones anteriores.

Esta herramienta, la cual se enmarca en el campo del **Diseño de Evaluaciónes**, podría ser utilizada por profesores y encargados del curso para iterar sobre evaluaciones y recibir retroalimentación antes de su implementación, pudiendo agilizar el proceso y evitar problemas.

Predicción de rendimiento en un curso

Explorando ideas en el campo del **Modelado Predictivo** en educación, se consideró una propuesta basada en el uso de información acerca de la trayectoria académica del estudiante y rendimientos históricos en un curso, para evaluar cuál tema representará un mayor desafío. Esto permitiría a los profesores poner énfasis en aquellos temas donde los estudiantes pudieran tener mayores dificultades.

Recomendación de materias en base a disponibilidad y escolaridad

Otra idea relacionada al **Modelado Predictivo**, es la de utilizar datos históricos de escolaridades en una carrera para entrenar un modelo capaz de, en base a una escolaridad, recomendar el mejor camino académico.

Esto permitiría al estudiante tomar decisiones más informadas acerca de las materias en base a su trayectoria y puede ser acompañado con el sistema de previaturas para evitar opciones inviables.

Predicción de rendimiento en una carrera

Enfocado al área académica y administrativa de la facultad, pero con posibilidades para dar al estudiante información importante acerca de su rendimiento esperable, la propuesta busca utilizar el **Modelado Predictivo** para, en base a información de resultados y trayectoria académica de un estudiante, evaluar al comienzo de su carrera cuál es el rendimiento esperable y donde podría encontrar mayores dificultades.

Estos datos podrían permitir una mejor planificación de la trayectoria de estudiantes particulares, así como generaciones enteras, a la vez que ajustar expectativas para evitar frustraciones.

2.2.2. Definición de los criterios y prioridades

Luego de identificar las ideas posibles, el siguiente paso es definir criterios de selección que tengan en cuenta viabilidad, utilidad y alineación con los objetivos del proyecto. Los mismos se encuentran detallados a continuación:

Impacto en el desarrollo cognitivo del estudiante Mejorar el proceso cognitivo de los estudiantes es uno de los objetivos centrales del proyecto. Por ello, se busca que la propuesta seleccionada provea un mecanismo claro a la hora de facilitar el aprendizaje, a través de la mejora en la adquisición de conocimientos, el desarrollo de habilidades específicas o la simplificación del proceso de enseñanza.

Evaluación del impacto Alineado con el criterio anterior, se busca que la idea seleccionada permita establecer un enfoque claro para recoger y analizar aquellos datos necesarios para medir su impacto en el desarrollo cognitivo del estudiante y, por consiguiente, su eficacia.

Enfoque en el estudiante A partir de los puntos anteriores y de los objetivos iniciales, es evidente la necesidad en analizar el impacto en la cognición de los estudiantes terciarios de ingeniería. En función de esto, priorizamos aquellas ideas que, al responder a las necesidades específicas de estos estudiantes, resulten en un impacto en su desarrollo más directo y evaluable.

Acceso a los recursos Se busca priorizar el simple acceso a los recursos necesarios, es decir, que la herramienta pueda utilizar información y recursos que sean fácilmente accesibles. Esto abarca distintos factores, como la estandarización de aquellos datos utilizados, su interpretabilidad, la disponibilidad técnica de las fuentes y los permisos necesarios para su uso. Este objetivo no solo busca que el producto sea versátil y adaptable a distintas realidades, sino que vela por el cumplimiento de los plazos establecidos y la eliminación de barreras para la obtención de aquellos insumos necesarios.

Manejo de información sensible Tanto por razones éticas como prácticas, a la hora de la elección se priorizan aquellas ideas que requieran tratar la menor cantidad de datos sensibles. Esto busca abordar con responsabilidad el manejo de datos de los estudiantes, considerando su privacidad, la seguridad y el cumplimiento de normativas legales aplicables.

Implicaciones éticas y pedagógicas Aunque guarda relación con la gestión de datos personales, este criterio abarca un espectro más amplio de consideraciones éticas. Priorizaremos ideas que respeten la privacidad de los estudiantes, sean transparentes en cómo procesan y utilizan sus datos, y eviten incurrir en

sesgos, discriminación o exclusión. Asimismo, se busca que las soluciones fomenten un aprendizaje motivador y saludable para los estudiantes, evitando enfoques que generen presiones innecesarias o dinámicas tóxicas.

Escalabilidad y adaptabilidad Se priorizaran propuestas que puedan abarcar distintos contextos, áreas de conocimiento o tipos de estudiantes más allá de los inicialmente considerados, como aquellos fuera del ámbito de la ingeniería. También evalúa la facilidad con la que la herramienta podría manejar un aumento en la cantidad de datos o usuarios, considerando la eficiencia en el uso de recursos y la integración con nuevas fuentes de información.
En cuanto a la adaptabilidad, también se pretende prorizar la capacidad para ajustarse a cambios en el entorno, como modificaciones en la currícula, incorporación de nuevas tecnologías o la variación en las necesidades de los usuarios.

Pertinencia desde la experiencia Como criterio final pero de gran importancia en el proceso de selección, se encuentra nuestro interés por abordar dificultades vividas en nuestra trayectoria como estudiantes de ingeniería. Creemos que esto nos permitirá entender de mejor manera las necesidades al momento de realizar la implementación y simplificará enormemente la toma de decisiones.

2.2.3. Definición de la idea seleccionada

Definidos los criterios y elaborada una lista preliminar de ideas, se dio inicio al proceso de selección. Es importante señalar que varias de las opciones descartadas, a pesar de no cumplir con algunos de los criterios, despertaron gran interés en el grupo y serán consideradas como posible trabajo futuro 5.1.1.

Ideas Descartadas

Predicción de rendimiento en un curso/carrera Si bien se evaluaron ambas ideas, ciertos factores llevaron a descartarlas. Desde el punto de vista de su planteamiento y evaluación, dado que no se centra directamente en el estudiante ni en su proceso de aprendizaje, resulta difícil determinar sus efectos y establecer criterios claros para evaluarlos.

Por otro lado, como ocurre con toda propuesta basada en modelos predictivos, ambas ideas requieren una cantidad considerable de datos, muchos de ellos sensibles al tratarse de información estudiantil. El manejo de los mismos plantea dificultades técnicas, éticas y la necesidad de tener en cuenta aspectos legales.

A esto se suma que, si bien un modelo de este tipo puede reutilizarse para distintos estudiantes una vez entrenado, lo que lo vuelve escalable, su desempeño depende fuertemente de los datos utilizados durante su entrenamiento, los cuales pueden quedar rápidamente obsoletos ante cambios de contexto. Por ejemplo, si un modelo fuese entrenado con datos correspondientes al curso “Cálculo 1”, al implementarse una nueva versión del curso, como “Cálculo DIVV”, las modificaciones de la currícula harían que los rendimientos no fuesen extrapolables y por tanto el modelo quedara obsoleto.

Recomendación de materias en base a disponibilidad y escolaridad Al igual que la predicción de rendimientos, se trata de una idea basada en el modelado predictivo y por tanto, requiere gran cantidad de información para su implementación.

Contar con datos anonimizados de escolaridades pasadas podría facilitar su manejo, quitando el carácter sensible de los mismos. Sin embargo, el acceso sigue siendo una dificultad, en particular con los plazos manejados, pues requiere la colaboración, autorización y anonimización por parte de la facultad. A su vez, al basarse en un conjunto de materias dadas y un programa para la carrera, la solución será susceptible ante cambios de currícula, incluso pudiendo perder valor por completo.

Por otro lado, aunque no menos importante, al afectar de forma tangencial las decisiones del estudiante, se vuelve difícil pensar cual será su impacto directo y por tanto qué datos se deberían evaluar para valorar su funcionamiento.

A pesar de todo ello, esta idea es de las que mayor interés despertó en nosotros y creemos que podría haber aportado a nuestro pasaje por la enseñanza terciaria.

Análisis de la calidad de las evaluaciones La razón principal por la cual descartar esta propuesta es su enfoque en el docente, con un impacto colateral en el estudiante y sin forma directa de medir los efectos en su cognición.

A su vez, desde nuestra experiencia, nos es difícil evaluar los resultados que pudieran obtenerse, ya que no hemos estado expuestos a esta problemática ni interiorizados con los pormenores de la generacion de evaluaciones oficiales.

Idea Seleccionada - FinQuiz

Al seleccionar la idea final, observamos que muchas de las propuestas cumplen de manera similar con los criterios definidos y resultan igualmente interesantes de implementar. Por ello, se desarrolló una propuesta que integra elementos de varias de las ideas originales y que puede ser acotada a una realidad concreta o ajustarse según sea necesario para alcanzar un resultado dentro de los tiempos previstos.

FinQuiz, es una aplicación web diseñada para generar cuestionarios múltiple opción que permitan al estudiante repasar uno o varios temas de un curso, con una intervención mínima por parte del docente. Estos son corregidos de forma automática, siendo capaz el sistema de sugerir material para afianzar aquellos conocimientos menos firmes.

Como resultado, se conjuga elementos del **Diseño de Evaluaciónes** a la hora de generar cuestionarios, con características propias de sistemas de **Tutoría Inteligente** que agregan una capa de complejidad a los mismos mediante la recomendación de material para el repaso.

Razones para la selección A continuación se mencionan algunas de las razones por las que nos decantamos por esta elección.

Enfoque el Estudiante Se encuentra enfocada en el estudiante, en particular al estudiante de la Facultad de Ingeniería de la UDELAR, y le aporta valor al mismo de forma directa, haciendo más simple y guiado su proceso de aprendizaje.

Medición de resultados Al tener relación directa con el proceso de aprendizaje del estudiante, permite recabar aquellos datos necesarios para evaluar su funcionamiento. Los resultados en los cuestionarios, personas que utilizaron la herramienta, entre otras estadísticas, pueden ser luego comparados con porcentajes de aprobación o resultados en evaluaciónes, y de ese modo obtener un aproximado de su eficacia.

Relevancia personal Ha sido uno de los puntos con mayor impacto al momento de la decisión, ya que como estudiantes en el momento final de sus carreras, quisimos destacar una herramienta que nos hubiese sido útil a lo largo de la misma. Esto debido a que muchas veces no se cuenta con autoevaluaciones, ni formas de realizar un proceso autoguiado de estudio que pueda adaptarse a los tiempos del estudiante y permitan afianzar paso a paso lo aprendido antes de enfrentarse a evaluaciones más completas.

Facilidad de acceso a los recursos La herramienta busca, en principio, hacer uso de todo aquel material ya disponible para el curso o que pueda ser proporcionado por sus docentes. Esto es una ventaja para el estudiante, que puede sacar el mayor provecho posible de los materiales disponibles, y para el sistema que no requiere de datos difíciles de acceder o procesar. A su vez, al no utilizar información personal, excepto por las credenciales de registro, el sistema no se enfrenta a dificultades agregadas sobre el manejo de datos sensibles.

Desarrollo Paralelizable Se trata de un sistema modular, compuesto por secciones independientes, lo que ofrece varias ventajas.

En primer lugar, permite el desarrollo paralelo, acelerando este proceso y permitiendo enfocar los esfuerzos en aquellas secciones que resulten más valiosas. Por otro lado, facilita el mantenimiento y la adaptabilidad, permitiendo intercambiar secciones sin que ello implique un gran retrabajo en el resto de la aplicación. Esto contribuye a mantener el sistema actualizado, algo especialmente importante en un campo tan dinámico y en constante evolución.

Crecimiento futuro Posibilita futuras iteraciones, ya que permite la anexión de otros sistemas que puedan nutrir los ya propuestos o mejorar cada una de sus secciones.

Extensible y adaptable Es extensible y adaptable, ya que busca que la menor cantidad de trabajo sea necesária a la hora de adaptar el sistema a nuevas materias o cambios de currícula, dependiendo en gran medida del material correspondiente

Consideraciones A continuación se mencionan algunas consideraciones tomadas al momento de plantear la propuesta y que han impactado su desarrollo.

- El objetivo final es ayudar al proceso de aprendizaje del estudiante, por lo tanto, no se hará foco en la seguridad de los cuestionarios, pues su uso no es el de evaluación formal y buscar atajos para responderlos de forma correcta no aporta valor al estudiante de ningún modo.
- Se trata de un sistema independiente, que no requiere integración con otros sistemas de la facutlad ni esta pensado para la comunicación con ellos. Esto para acotar su función y generar un ambiente controlado que permita experimentar y analizar sus resultados.
- La elección de implementar una aplicación web, a la vez de facilitar la creación rápida de un Producto Mínimo Viable o *MVP* escalable y de fácil acceso, busca una similitud con otras plataformas ya usadas por los estudiantes como EVA y OpenFing. A su vez, en una primera instancia se pondrá foco en la experiencia de escritorio, ya que en base a nuestras experiencias personales y el acceso que permite de forma gratuita la facultad mediante sus computadoras, creemos se adecúa de mejor manera a una instancia de estudio.

2.3. Definición de Preguntas Múltiple Opción

Las evaluaciones generadas por el sistema a implementar estarán compuestas de preguntas múltiple opción (o *MCQ* por su nombre en inglés).

La facilidad a la hora de corregir preguntas MCQ hace que resulten particularmente atractivas a la hora de evaluar grandes grupos de estudiantes. También, la velocidad con la que un estudiante puede llegar a responder este tipo de evaluaciones hace que sean buenas a la hora de evaluar un temario amplio. A su vez, según (Begum, 2012) las preguntas de múltiple opción bien formuladas logran ser más objetivas que las preguntas abiertas, ya sea con respuestas largas o cortas, lo cual da lugar a generar evaluaciones más confiables e imparciales que permiten entender el rendimiento de los estudiantes de una manera simple y directa.

A continuación se presentan los conceptos teóricos relacionados necesarios para orientar las decisiones de diseño e implementación del sistema.

El objetivo de profundizar en la definición y los atributos que hacen valiosas a este tipo de preguntas en el proceso de aprendizaje es establecer bases conceptuales claras. Estas podrán ser transmitidas de forma efectiva a un LLM, permitiendo guiar el proceso de generación y asegurar la calidad de los resultados obtenidos.

De acuerdo con (Centre for Teaching Excellence, s.f.), una pregunta múltiple opción se compone de tres partes: un **enunciado** (o *stem*) que presenta la pregunta a responder y un conjunto de posibles respuestas, dentro de las cuales se debe encontrar al menos una **respuesta correcta** (clave o *key*) y un número de respuestas posibles pero incorrectas llamadas **distractores**.

2.3.1. Definición extendida

Con esto en mente, existen varios lineamientos (Brigham Young University Testing Center, s.f.), (Al-Rukban, 2006), (Haladyna, Downing, y Rodriguez, 2002) de diferentes entes que intentan estandarizar, o al menos guiar, el proceso de diseño de este tipo de preguntas y así garantizar la fiabilidad de las evaluaciones. Es a partir de estos que se formula la siguiente definición extendida de MCQ.

Una pregunta de múltiple opción es un formato de evaluación estructurado utilizado para medir el conocimiento, la comprensión y las habilidades de toma de decisiones de un estudiante en un determinado tema, dado un contexto específico sobre lo que se espera que el estudiante sepa. Como fue mencionado anteriormente, cada una de estas consta de tres componentes: **Enunciado**, **clave** y **distractores**.

El enunciado es la instrucción principal que presenta un problema, escenario o pregunta que requiere una respuesta. Debe cumplir con las siguientes características:

- **Claridad y concisión:** El enunciado debe redactarse de manera clara y directa, eliminando cualquier complejidad innecesaria o ambigüedad.
- **Completo y autosuficiente:** El enunciado debe proporcionar suficiente información para que el estudiante comprenda la pregunta sin necesidad de revisar las opciones de respuesta primero.
- **Centrado en un único problema:** El enunciado debe evaluar un solo concepto u objetivo de aprendizaje, evitando interpretaciones múltiples.
- **Evitar preguntas negativas cuando sea posible:** Las preguntas deben formularse en términos positivos en lugar de negativos, evitando preguntas del tipo de "¿Cuál de las siguientes opciones NO es correcta?".

La unión de los distractores y las claves forman el conjunto de opciones de una pregunta, estas deben cumplir con las siguientes propiedades:

- **Distractores plausibles:** Los distractores deben parecer razonables, coherentes y plausibles para un estudiante que no entiende del todo el tema.
- **Distractores disjuntos:** Los distractores no deben superponerse a la respuesta correcta. Cada distractor debe ser diferente a los otros y deben reflejar un error diferente sobre el tema.
- **Largo similar de distractores:** Los distractores deben ser concisos y tener una longitud similar a la respuesta correcta.
- **Evitar claves ambiguas:** El lenguaje de la clave debe ser claro y adecuado al nivel del estudiante, sin dificultar innecesariamente la comprensión.

- **Evitar claves más largas o detalladas que el resto de las opciones:** Las respuestas correctas no deben ser significativamente más extensas o específicas que el resto de las opciones, ya que esto puede dar pistas involuntarias.
- **No introducir nueva información en las claves:** Todo el contenido necesario para responder la pregunta debe estar presente en el enunciado.
- **No usar opciones que hagan referencias a otras opciones:** Respuestas del tipo "Todas las anteriores", "Las opción B y C son correctas" o "Ninguna de las anteriores" no alientan a los estudiantes a pensar.
- **Consistencia sintáctica y gramatical:** Todas las opciones deben seguir la misma estructura gramatical y utilizar el mismo tiempo verbal para evitar dar pistas no intencionales al estudiante.

Esta definición brinda lineamientos y restricciones claras sobre cada una de las partes de una pregunta, sentando las bases mínimas de consistencia, claridad y completitud que estas deben cumplir para evitar ambigüedades y maximizar su utilidad como herramientas de aprendizaje. Sin embargo, aun existen distintas variables que no han sido tomadas en cuenta como pueden ser la cantidad de distractores, la cantidad de opciones correctas o el orden de las opciones.

2.3.2. Consideraciones

Las decisiones sobre estas variables pueden tener un impacto significativo en la eficacia de las preguntas. Es por esto que a continuación se discute el efecto de las posibles alternativas para luego, en la sección 3.1.3, poder tomar decisiones informadas y llegar a una definición aplicada que será utilizada para la implementación posterior.

Cantidad de respuestas correctas

En cuanto a la cantidad de opciones correctas existen dos posibles alternativas: preguntas con una única opción correcta, llamadas de única mejor respuesta (según la literatura, *single-best-answer* o *SBA*); o varias opciones correctas, dando lugar a preguntas de múltiple opción con múltiple selección (*multiple-choice-multiple-response* o *MCMR*).

En este caso es directo notar que la alternativa con una única opción correcta es más simple de corregir y puntuar que su versión con múltiples opciones. Las preguntas MCMR necesitan de una estrategia más compleja de puntajes y penalizaciones parciales, mientras que las SBA tienen una forma binaria y predecible de corrección, lo que hace que sean más fáciles de escribir, requieran un menor esfuerzo de corrección a gran escala y den menor lugar a confusiones por parte de los estudiantes.

De acuerdo con (Olsho y cols., 2021), las preguntas de selección múltiple presentan la ventaja de permitir, para un mismo enunciado, evaluar varios caminos

de razonamiento posibles y correctos, dando más información sobre el proceso de aprendizaje de los estudiantes. Sin embargo, también en ese mismo estudio se plantea la posibilidad de que este tipo de preguntas no sean tan valiosas para evaluaciones plenamente formativas o con pocas consecuencias, ya que los estudiantes suelen incurrir en más riesgos al seleccionar una mayor cantidad de opciones, conducta que se ve agravada cuando las evaluaciones son tomadas en modalidad online.

Por otra parte, (Neumann, Simmrodt, Bader, Opitz, y Gergs, 2023) directamente comparan evaluaciones de SBA y MCMR en el ámbito de la farmacología a lo largo de un año completo, viendo que los estudiantes que reciben cuestionarios con preguntas MCMR tienen un mejor rendimiento en pruebas subsiguientes. A partir de esto, se puede concluir que la complejidad conceptual de las pruebas de selección múltiple hace que aumente la capacidad de retención de conocimiento de los estudiantes, por lo cual este estudio propone las MCMR como una buena opción para evaluaciones formativas.

Por último, es interesante notar que (Lin y Singh, 2016) examinan la efectividad de preguntas de respuesta libre contra preguntas MCQ de respuesta única, llegando a la conclusión de que preguntas de múltiple opción bien formuladas pueden ser tan efectivas como preguntas de desarrollo abiertas, manteniendo la facilidad para su corrección y posterior análisis cuantitativo.

Cantidad de opciones

La cantidad de opciones óptimas para preguntas de múltiple opción ha sido foco de la discusión de los investigadores desde hace ya varios años, ejemplos de esto son (Budescu y Nevo, 1985), (Rodriguez, 2005), (Raymond, Stevens, y Bucak, 2018) o (Norooziasl y cols., 2021)

En este caso la discusión es algo más simple, ya que parece haber un consenso claro que favorece a las preguntas MCQ con tres opciones por sobre el resto. Hay varias razones para llegar a esta conclusión, en particular, (Raymond y cols., 2018) plantea que en preguntas con más de tres opciones algunos distractores resultan redundantes y no funcionales, siendo seleccionados por menos del 5 % del total de estudiantes que son evaluados.

Los estudios mencionados anteriormente apuntan a que las preguntas MCQ con tres opciones son tan válidas y confiables como lo son las de cuatro y cinco opciones, logrando alcanzar los mismos niveles de dificultad pero requiriendo un menor esfuerzo a la hora de crearlas y presentando una menor cantidad de distracciones innecesarias a la hora de ser resueltas.

Orden de opciones

Otro punto sobre el cual existe discusión a la hora de la generación de preguntas MCQ es el orden de las opciones y el posicionamiento de la respuesta correcta dentro del conjunto de las mismas. Una mala estrategia de ordenamiento de opciones puede ser contraproducente, tanto complejizando innecesariamente

una evaluación como simplificándola al seguir patrones predecibles que ofrecen pistas a los estudiantes.

En concreto, (Lions y cols., 2022) mencionan varios aspectos interesantes de este tema, como puede ser la posición de las respuestas correctas dentro de las opciones. Basándose en el hecho de que las preguntas MCQ son propensas a que los estudiantes intenten adivinar la respuesta correcta sin basarse en conocimiento, es posible identificar ciertos patrones de los estudiantes respecto del posicionamiento de la misma que pueden afectar a la tasa de acierto de una pregunta.

Existe un fenómeno llamado *middle bias* por el cual los estudiantes inconscientemente no seleccionan las opciones de los extremos de una lista de opciones, lo cual hace que las preguntas tengan una mayor cantidad de respuestas correctas si la opción correcta se encuentra entre medio de los distractores. También, ha sido comprobado que la dificultad de una pregunta disminuye si la opción correcta es presentada como la primera opción.

El mismo estudio también discute el efecto del ordenamiento de las opciones para evaluaciones de múltiple opción, donde existen lineamientos que sugieren que las opciones deben ser ordenadas de manera lógica. De cualquier manera, no existe un consenso claro sobre qué es un ordenamiento lógico, ya sea numéricamente, alfabéticamente, de manera ascendente o descendente.

La idea tras dar las opciones en un orden lógico tiene dos motivos principales. En primer lugar, el ordenamiento de las opciones pasa a ser un proceso mecanizado que evita que las respuestas correctas terminen en posiciones determinadas dentro de la lista de opciones, generando patrones aleatorios a lo largo de las evaluaciones. En segundo lugar, son varios los estudios (Moreno, Martínez, y Muñiz, 2006), (Roelofs, 2019) que afirman que dar las opciones en un orden inconsistente puede agregar fuentes de complejidad innecesaria a las preguntas, ya que aumenta la carga cognitiva para los estudiantes y los distrae del objetivo real de la misma.

Por otro lado, existe una aversión por parte de los estudiantes, a lo largo de una evaluación completa, a responder varias veces seguidas con la misma opción o terminar con una lista "desbalanceada" de respuestas, por lo tanto, una prueba que siga una distribución atípica de las respuestas correctas suele ser interpretada como más compleja.

Es por esto que se plantean dos opciones para posicionar las respuestas correctas a lo largo de una evaluación: *Key balancing*, que consiste en distribuir de manera equitativa la ubicación de las respuestas correctas a lo largo de la evaluación; y *key randomization* que implica asignar la respuesta correcta a una posición aleatoria en cada ítem, sin seguir un patrón sistemático. En particular, las evaluaciones balanceadas fallan ante suposiciones equilibradas, donde un alumno, al notar que las respuestas siguen algún tipo de patrón puede llegar a acertar la respuesta correcta sin realmente tener conocimiento sobre el tema evaluado.

2.4. Revisión de antecedentes

En esta sección se busca entender el panorama actual del área de la generación de preguntas, haciendo foco en la generación de preguntas de múltiple opción, con el objetivo de entender cuál es el punto de partida de este proyecto y qué producto es alcanzable mediante la comparación con resultados de herramientas que atacan problemas similares.

También, es valioso analizar cuál es el impacto de ciertas decisiones tomadas por otros sistemas semejantes sobre la calidad de las preguntas que generan. De esta manera, se cuenta con más herramientas a la hora de entender el impacto de las decisiones tomadas durante la etapa de implementación.

2.4.1. Sistemas

En primer lugar, vale la pena analizar los resultados que surgen a partir de probar distintos sistemas existentes que se encargan de generar preguntas, o directamente evaluaciones, a partir de un material de referencia.

Estos sistemas resuelven integralmente el problema de la generación de preguntas: desde la exposición de una página web que permita subir el contenido, pasando por la creación de uno o mas cuestionarios, hasta la presentación de los resultados y su posible exportación.

El proceso de prueba de estos sistemas consistió en utilizar la interfaz web de los mismos para, dado un contenido fijo, generar evaluaciones y asi poder analizar, en base a nuestro propio criterio, la calidad de las preguntas generadas. Este proceso se hizo enfocándose en la dificultad y la relevancia de las preguntas generadas, además de observar el sitio en sí y su usabilidad.

En concreto, los sistemas en los que se enfoca esta revisión son:

- *quizify.xyz*(Rotimi Best, 2025): Es el más simple de los sistemas que fueron probados, simplemente toma un texto como entrada y devuelve las preguntas en diferentes formatos. En base a la pruebas hechas, es posible afirmar que la usabilidad del sitio es simple y logra generar preguntas aceptables, aunque tal vez demasiado sencillas.

- *Quizbot* (Capstone Edu LLC, 2025): Es un sistema más completo y con más opciones además de generación de preguntas. Para generarlas toma como entrada el material de referencia en texto o pdf y otras variables como pueden ser la dificultad esperada de las preguntas, a qué nivel educativo están destinadas o el idioma esperado de salida. A partir de esto, nuestras pruebas nos permiten decir que las preguntas generadas son relativamente complejas y con explicaciones para cada una de las opciones.

- *Quizgecko* (Quizgecko, 2025): Es una aplicación web que toma como entrada el material de referencia en texto o en formato pdf y, utilizando IA, genera una guía y material de ayuda para el estudio del mismo. Además de crear una evaluación con preguntas de múltiple opción, también genera un medidor de progreso de estudio, tarjetas de estudio y hasta un

podcast para facilitar al estudiante el consumo del material. En cuanto a las preguntas generadas en el proceso de pruebas empleado, se vio que las preguntas fueron relevantes al tema dado y con un nivel de complejidad aceptable.

Cabe destacar que, a pesar de que todos los sistemas mencionados anteriormente generan preguntas múltiple opción, no son totalmente comparables entre sí ya que tienen objetivos distintos, algunos más enfocados en el proceso de generación en si y otros en el estudiante y en su proceso cognitivo. Sin embargo, este primer análisis permite confirmar la viabilidad de la generación de preguntas múltiple opción a partir de contenido de referencia y por sobre todo analizar el funcionamiento de sistemas existentes y las ideas que estos implementan.

2.4.2. Antecedentes técnicos

La investigación anteriormente presentada permite ver los resultados obtenidos por soluciones existentes, sin embargo no es posible entrar en detalle en cuanto al aspecto técnico del proceso de generación de preguntas empleado por estas soluciones, dado que los mismos no lo permiten. Por esto, también resulta interesante la revisión de antecedentes desde un punto de vista puramente técnico.

La generación automática de preguntas es una tarea en desarrollo desde hace décadas, y a lo largo del tiempo, se han propuesto diversas soluciones que han evolucionado conforme avanza la tecnología.

Inicialmente, los sistemas convencionales se basaban en reglas heurísticas para convertir un texto descriptivo en preguntas relacionadas, ya sea mediante enfoques sintácticos, semánticos o utilizando plantillas.

Con la aparición de las redes neuronales y el mejor manejo de los grandes volúmenes de datos aparecieron diversas soluciones, donde fueron predominantes los modelos que seguían el framework *seq2seq*, que a partir de un codificador generaban una representación interemedia de los datos de entrada para luego usar un decodificador y generar preguntas a partir de la representación generada.

Los modelos de lenguaje pre-entrenados, como pueden ser T5 (Raffel y cols., 2019) o Bart (Lewis y cols., 2019), representaron un avance significativo respecto de las soluciones basadas en redes neuronales más simples, para finalmente dar lugar al uso de LLMs modernos.

Sin embargo, el avance en esta área no depende únicamente del desarrollo tecnológico, sino que también existen diferentes estratégias para diferentes dominios que permiten mejorar la calidad de los resultados obtenidos. Como resultado de la investigación hecha con el fin de interiorizarse con los pormenores de esta área, se presenta a continuación una selección de estudios recientes que tocan el tema de la generación de preguntas utilizando diferentes enfoques que nos resultaron interesantes.

Question Generation Using Sequence-to-Sequence Model with Semantic Role Labels

El primero de los ejemplos a analizar es *Question Generation Using Sequence-to-Sequence Model with Semantic Role Labels* (Naeiji, An, Davoudi, Delpisheh, y Alzghool, 2023), que propone un método de generación de preguntas que busca combinar los beneficios de los modelos basados en reglas lingüísticas y aquellos basados en *seq2seq*.

En concreto, esta solución utiliza herramientas de etiquetado de roles semánticos, propio de los sistemas basados en reglas, para capturar la infromación lingüística de las sentencias de entradas y sus posibles puntos de vista. Con esto se logra que el modelo de generación de preguntas *seq2seq* trabaje únicamente con entradas y salidas etiquetadas semánticamente, alcanzando así algunas ventajas de ambos tipos de solución a la misma vez.

A nivel técnico, esta solución utiliza tanto BART como T5 como modelos *seq2seq* basados en transformadores, los cuales pasan por un proceso de *fine-tuning* con diferentes datasets de generación de preguntas, como pueden ser *SQuAD2.0* (Rajpurkar, Zhang, Lopyrev, y Liang, 2016) o *NewsQA* (Trischler y cols., 2017). Además, se utiliza un modelo BERT pre-entrenado para el proceso de etiquetado semántico.

Finalmente, en base a un proceso objetivo de comparación, se logra concluir que esta estrategia de generación de preguntas es superior a los modelos basados en reglas, a los modelos de *seq2seq* originales e incluso tuvo mejores puntuaciones de precisión y recuperación que otros modelos de generación de preguntas avanzados basados en redes neuronales.

Planning First, Question Second: An LLM-Guided Method for Controllable Question Generation

Planning First, Question Second: An LLM-Guided Method for Controllable Question Generation (K. Li y Zhang, 2024) introduce la idea de un sistema *Planning First, Question Second o PFQS* guiado por un LLM (Llama 2), que genera un plan de respuesta que luego es utilizado como valor de entrada para un modelo de generación de preguntas.

Este proceso en dos pasos permite aumentar el control que se tiene sobre el contenido y la dificultad de los resultados, dos subáreas de la generación de preguntas relativamente nuevas y poco exploradas. Además, es un enfoque ampliamente adapatable permitiendo cambiar los modelo de generación de preguntas o de lenguaje independientemente de ser necesario.

El diferencial de PFQS con respecto a otro metodos de generación yace en que utiliza un LLM para generar los planes de respuesta, de los cuales se extraen las respuestas candidatas, sin la necesidad de tener que entrenar un modelo extra para hacerlo. El plan generado por el LLM luego extiende el prompt que recibe el modelo de generación de preguntas, usualmente un modelo *seq2seq* como puede ser BART, y aumenta su capacidad de generación. De hecho, parte de la comprobación empírica de este estudio es utilizar BART-large con y sin PFQS

viendo un aumento notable en la calidad de los resultados obtenidos.

A Hybrid Approach for Automatic Question Generation from Program Codes

En este estudio (Alshboul y Baksa-Varga, 2024) construyen un modelo que logra conceptualizar y generar preguntas sobre programación con Python a partir de código. También, se busca extraer los conceptos de los extractos de código y asociarlo con contenidos del curso para poder así asegurar la relevancia de las preguntas obtenidas.

En este proceso se utiliza QuestGenAI (Goutham, 2020), un framework *open-source* para generación de preguntas, y ontologías para estandarizar la representación de conocimiento y conceptualizar la lógica de los programas que se utilizan como dato de entrada.

Este enfoque, que utiliza tanto IA como conocimiento semántico, logra generar preguntas apropiadas a partir de fragmentos de código, disminuyendo la necesidad de poder computacional haciendo uso de ontologías y así mejorando la capacidad de escalabilidad de este sistema. Cabe destacar que el preprocesamiento semántico del código comienza a fallar en la medida en la que este se vuelve mas complejo.

A Comparative Study of AI-Generated (GPT-4) and Human-crafted MCQs in Programming Education

A Comparative Study of AI-Generated (GPT-4) and Human-crafted MCQs in Programming Education (Doughty y cols., 2023) describe un sistema basado en GPT-4 que automatiza la generación de preguntas múltiple opción para estudiantes de nivel terciarios de cursos introductorios de programación que utilizan Python.

Lo novedoso de su enfoque yace en que estos resultados fueron logrados basándose no tanto en el material de la asignatura, sino en una visión amplia de alto nivel del contenido del curso y agregando en detalle los objetivos de aprendizaje por tema, logrando así que cada pregunta se alinee mejor con cada uno de estos objetivos mientras se mantienen en el marco del curso. Esto es una gran diferencia en comparación con otros sistemas que toman una sección del texto del curso y generan una única pregunta, logrando un paralelismo directo entre preguntas y texto pero no entre preguntas y objetivos de aprendizaje.

A grandes rasgos, lograron concluir, luego de comparar sus resultados con un banco de preguntas generadas por humanos, que las preguntas generadas por el sistema desarrollado son tan claras y su código es tan correcto desde un punto de vista lógico y sintáctico como el de los humanos. Además, afirman que las preguntas generadas por el sistema se ajustan mejor a los objetivos de aprendizaje que las generadas por profesores. Sin embargo, es más común que tengan más de una solución correcta por error y que la calidad de sus distractores sea peor, todo dentro de los parámetros de lo aceptable.

MCQGen: A Large Language Model-Driven MCQ Generator for Personalized Learning

Otro estudio notable y que vale la pena referenciar es *MCQGen: A Large Language Model-Driven MCQ Generator for Personalized Learning* (Hang, Wei Tan, y Yu, 2024), un sistema basado en GPT-4 que, utilizando técnicas avanzadas de *prompt engineering* y *retrieval augmented generation (RAG)*, genera cuestionarios múltiple opción que se adaptan a las necesidades de los estudiantes a partir de los contenidos de un curso.

MCQgen es un sistema con un diferencial técnico respecto de otros sistemas con el mismo objetivo, donde cada uno de estos agregados por encima del LLM tienen un objetivo concreto. En particular, la utilización de RAG apunta a mejorar las capacidades de generación mediante la incorporación de conocimiento de una base de datos externa que contiene ejemplos de preguntas hechas por alumnos e instructores. Por otro lado, utiliza dos técnicas de prompt engineering: *chain-of-thought*, para obligar al LLM a emular un razonamiento similar al de un humano durante la creación de preguntas, y *self-refine*, para que el mismo LLM re-evalúe a cada paso la calidad del contenido que genera y así ajustar diferentes parámetros.

Finalmente, implantando MCQGen en un curso de iniciación de la programación se concluyó que el mismo mejora el rendimiento de los estudiantes a la vez que facilita su experiencia educativa.

2.4.3. Conclusiones

Este relevo de antecedentes nos permite entender el panórama actual del área de generación de preguntas.

Como una primera observación, es posible confirmar la viabilidad de la construcción de sistemas de generación de preguntas y la existencia de diferentes enfoques para lograrlo.

Mirando al problema planteado desde el lado técnico, este estudio revela que hay una gran variedad de posibles encares y tecnologías que pueden ser útiles a la hora de idear un sistema que lo resuelva. Por otro lado, también podemos afirmar que estas tendencias están en constante desarrollo y sus capacidades cambian muy rápidamente, haciendo de la elección de una estrategia un desafío en sí mismo.

En concreto, las soluciones que alcanzan mejores resultados suelen utilizar LLMs, modelos *seq2seq* basados en transformadores o una combinación de ellos como modelos de generación de preguntas, donde cada uno de estos enfoques tiene sus ventajas y desventajas, haciéndolos mas o menos convenientes en ciertas circunstancias.

Específicamente, los modelos *seq2seq* permiten encarar la construcción del sistema planteado con una perspectiva de más bajo nivel, necesitando grandes volúmenes de datos para su entrenamiento pero permitiendo un desarrollo medible y cuantificable.

Por otro lado, un enfoque basado en LLMs es, objetivamente, más avanzado

desde un punto de vista tecnológico y su pre-entrenamiento extenso hace que la solución no necesite de un corpus de entrenamiento para funcionar. A su vez, parece importante notar que dada la naturaleza del problema a resolver no es trivial el proceso de obtención de métricas que ayuden con el proceso de evaluación de los resultados obtenidos.

Como un punto relacionado, de acuerdo a lo mencionado por (Guo, Liao, Li, y Chua, 2024) existe una clara tendencia en el área de generación de preguntas a favor de los LLMs, los cuales son cada vez más utilizados y siguen sacando ventaja en materia de resultados cuando son comparados contra sus antepasados tecnológicos.

Además, dado lo novedoso de esta tecnología, los enfoques basados en LLMs siguen siendo aún un espacio algo inexplorado de la investigación, con lo cual parece interesante aportar en esta dirección. Más aún teniendo en cuenta la vertiginosidad con la que los LLMs han sido desarrollados en los últimos años y la gran cantidad de nuevas opciones que han aparecido con nuevas capacidades, mayor poder de procesamiento, mayor cantidad de parámetros o capacidad de razonamiento.

Mirando los antecedentes mostrados, también es posible identificar que existen ciertas estratégias que mejoran el resultado de los sistemas de generación de preguntas, como puede ser el *fine-tuning* a partir de un corpus de entrenamiento, la aplicación de un RAG, o, desde un punto de vista menos técnico, diferentes formas de preprocesar las entradas de los modelos. Estas también son variables que vale la pena tener en mente a la hora de decidir sobre la implementación del proyecto.

Como último punto a destacar, este proceso de revisión permite afirmar que estos enfoques discutidos han sido probados para la generación de preguntas en ámbitos educativos relacionado con la enseñanza de programación y han obtenidos buenos resultados. En concreto, también puede ser interesante destacar que no todas las asignaturas llegan al mismo producto, principalmente debido a la complejidad de la manipulación de datos que no son necesariamente textuales, como pueden ser gráficos en asignaturas de matemática o física.

Capítulo 3

FinQuiz: Generación de cuestionarios estudiantiles con modelos de lenguaje

3.1. Definición del Sistema

A partir de la idea seleccionada 2.2 y la información recabada durante el proceso de revisión de antecedentes 2, se procedió a definir y desarrollar una aplicación web denominada FinQuiz. Una descripción detallada de esta se presenta en esta sección.

3.1.1. Problema a Resolver

La masificación estudiantil representa un desafío persistente en distintos niveles educativos y contextos de enseñanza. Esta situación impacta tanto a los educadores, que pueden encontrar dificultades a la hora de atender la creciente demanda de estudiantes, como a los propios alumnos, quienes se enfrentan a un sistema con recursos limitados para brindarles una atención personalizada.

Uno de los problemas que esta situación trae consigo, es la falta de herramientas de autoevaluación accesibles, personalizadas y adaptables que acompañen el proceso de estudio de los estudiantes. Esta carencia se vuelve especialmente problemática en aquellas asignaturas donde la práctica constante es clave, ya que los estudiantes suelen depender de un conjunto limitado de ejercicios predefinidos y enfrentan la imposibilidad de acceder a nuevos ejercicios generados por un docente según sus necesidades puntuales.

FinQuiz busca abordar este problema permitiendo al estudiante generar cuestionarios múltiple opción enfocados en aquellos temas del curso que desea practicar, utilizando para ello las capacidades de un gran modelo de lenguaje (LLM) y ofreciendo, en base a los resultados, indicaciones de dónde repasar aquellos temas más débiles.

A la hora del desarrollo, cabe mencionar que con el objetivo de enfocar el desarrollo y obtener los mejores resultados posibles, se decidió limitar el alcance inicial del sistema a un único curso: Programación 1. Esta elección se fundamenta en la disponibilidad de material y en las características propias de la materia. Entre ellas se destacan la presencia de contenido práctico basado en código y la afinidad con las capacidades de la Inteligencia Artificial para generar preguntas a partir de dicho contenido, tal como se vio en el análisis de sistemas existentes 2.4.

Esta solución ofrece beneficios tanto para el estudiante como para el docente. Por un lado, el estudiante adquiere la capacidad de guiar su propio proceso de repaso y autoevaluación, ganando autonomía y complementando el material ya disponible. Por otro, el docente dispone de una herramienta que le permite optimizar su tiempo, al reducir la carga asociada a la generación manual de ejercicios, y enfocarse en formas más estratégicas de acompañamiento pedagógico.

Finalmente, la herramienta representa una oportunidad valiosa para evaluar el impacto que los sistemas basados en Inteligencia Artificial pueden tener en los procesos de aprendizaje. Este propósito se alinea con uno de los objetivos iniciales del proyecto, impulsado por los tutores, en un contexto donde la irrupción de estas tecnologías en la educación despierta un interés y debate cada vez mayores.

3.1.2. Descripción Técnica

Para el desarrollo de FinQuiz se optó por una aplicación web, con el objetivo de ofrecer una experiencia simple desde computadoras de escritorio, accesible tanto desde los hogares de los estudiantes como desde los equipos disponibles en la facultad.

Con esta premisa, se eligió **Ruby on Rails** (Ruby on Rails Core Team, 2025) para el desarrollo del backend, donde reside la lógica de negocio de la aplicación. Para el frontend, es decir, la interfaz visual de la herramienta, se seleccionó la biblioteca de JavaScript **React** (Meta Open Source, 2024).

Para la capa de persistencia de datos, se utilizó **PostgreSQL**, un sistema de gestión de bases de datos relacional y de código abierto (The PostgreSQL Global Development Group, 2025).

La Inteligencia Artificial cumple un rol central en la lógica de negocio de la herramienta, ya que permite generar cuestionarios personalizados con un mínimo esfuerzo por parte de los docentes. Sin este componente, la herramienta se limitaría a ser una interfaz más para crear cuestionarios. A la hora de incorporar estas tecnologías en nuestro sistema, se optó por integrar un gran modelo de lenguaje (LLM) a través de una API, una decisión que se detalla en el documento "Anexo: Decisiones Sobre el Modelo de Lenguaje", en la sección "Uso como Servicio frente a Implementación Local". En concreto, el modelo seleccionado, **Gemini 2.5 Pro** (Kavukcuoglu, 2025), fue elegido en función de su facilidad de integración, sus capacidades técnicas y su estructura de costos.

Todas estas decisiones técnicas responden a un análisis exhaustivo de diversas alternativas, con el objetivo de equilibrar factores como la facilidad de

desarrollo y cumplimiento de plazos, la creación de un sistema seguro y estable, y la optimización de los costos. Este análisis, junto con un mayor detalle de las tecnologías empleadas, se presenta en la sección 3.3.

3.1.3. Diseño de las preguntas múltiple opción

Más allá de los aspectos técnicos del sistema desarrollado, resulta fundamental profundizar en su componente central: las preguntas de opción múltiple que conforman los cuestionarios destinados a apoyar el aprendizaje del estudiante. Para ello, es necesario precisar qué entendemos por este tipo de preguntas y cuáles son las características deseables para maximizar su valor didáctico.

Previamente, en la sección 2.3, se presentó una definición detallada de las preguntas múltiple opción (MCQ), junto con sus componentes fundamentales, buenas prácticas y distintos enfoques de diseño posibles discutidos por diversos autores. Esa base conceptual permite comprender qué características hacen que una pregunta sea clara, válida, justa y útil en contextos educativos.

Como parte del proceso de definición del sistema, uno de los objetivos principales es traducir dichas bases teóricas en decisiones prácticas que orienten la generación automática de preguntas. En este sentido, se definió una estructura concreta para las preguntas múltiple opción utilizadas por el sistema, tomando en cuenta no solo las recomendaciones conceptuales, sino también las restricciones técnicas, los objetivos pedagógicos y la necesidad de mantener consistencia y calidad.

A continuación, se detallan los elementos que conforman esta estructura, junto con las decisiones tomadas en cada caso y su justificación.

Cantidad de respuestas correctas: única clave

Se optó por utilizar preguntas con una única opción correcta (SBA, por sus siglas en inglés, *Single Best Answer*). Esta decisión responde, en primer lugar, a la simplicidad que ofrece este formato a la hora de su corrección: el proceso de validación de respuestas es binario y predecible, lo cual facilita tanto la generación automática como la posterior evaluación a gran escala.

Esta elección también se ajusta al contexto particular del sistema orientado inicialmente al curso de Programación 1, decisión explicada en mayor detalle anteriormente, donde algunas preguntas, como las de tipo *correct output*, tienen una única salida esperada derivada de la ejecución de un fragmento de código. En estos casos, presentar múltiples respuestas correctas no solo sería innecesario, sino que podría inducir ambigüedad y generar confusión en la interpretación de la pregunta.

Si bien existen otros tipos de preguntas en el sistema, como las de tipo *fill in the blanks* o teóricas, en los que sería técnicamente posible considerar más de una opción correcta, se optó por mantener una única clave también en estos casos. Esta decisión se basa tanto en la claridad que ofrece al estudiante como en la simplificación de los procesos internos del sistema: desde la interacción

con el modelo de lenguaje para generar preguntas, hasta la definición de la arquitectura y la lógica de corrección.

Además, estudios como el de Lin y Singh (Lin y Singh, 2016) muestran que, en términos de efectividad en la evaluación del aprendizaje, no hay una correlación directa entre la cantidad de respuestas correctas y la calidad de la pregunta. En consecuencia, dado que admitir múltiples claves por pregunta implicaría una mayor complejidad técnica sin aportar mejoras significativas en la calidad evaluativa, se decidió adoptar un formato con una única clave.

Cantidad de opciones: tres opciones por pregunta

Como se vio en la sección 2.3, existe un consenso entre autores donde se concluyen que el uso de preguntas de múltiple opción con tres alternativas es mejor por sobre aquellas con cuatro o cinco. Diversos estudios señalan que, en muchos casos, los distractores adicionales tienden a ser poco funcionales, redundantes o directamente irrelevantes desde el punto de vista evaluativo.

En particular, Raymond et al. (Raymond y cols., 2018) muestran que, en preguntas con más de tres opciones, algunos distractores son seleccionados por menos del 5 % de los estudiantes, lo que indica que no cumplen su función de generar una elección plausible en aquellos alumnos que no dominan el contenido. Además, reducir el número de opciones facilita el proceso de generación automática por parte del modelo, ya que se requiere producir menos contenido con restricciones de calidad estrictas.

En base a estos puntos, se adoptó un formato de tres opciones por pregunta: una respuesta correcta (clave) y dos distractores.

Orden de las opciones: aleatorio

En relación al orden de las opciones y la posición de la respuesta correcta dentro de cada pregunta, se adoptó una estrategia de desordenamiento aleatorio (*Key Randomization*). Esto implica que, para cada pregunta generada, las opciones se presentan en un orden aleatorio, sin seguir ningún patrón sistemático en la ubicación de la clave.

Esta decisión se apoya en evidencia que desaconseja el uso de técnicas como el *Key Balancing*, es decir, distribuir de forma uniforme las claves en distintas posiciones a lo largo de una evaluación. Por ejemplo, Lions et al. (Lions y cols., 2022) argumentan que el *Key Balancing* puede ser explotado por estudiantes mediante estrategias de adivinanza balanceada. Si perciben una concentración de respuestas correctas en una opción (por ejemplo, muchas respuestas "A"), pueden asumir que las siguientes probablemente sean "B" o "C", introduciendo sesgos no deseados.

Además de sus ventajas pedagógicas, *Key Randomization* resultó un procedimiento sencillo de automatizar en el sistema, lo que lo convirtió en una opción adecuada tanto desde el punto de vista técnico como educativo.

3.1.4. Roles y Entidades

A la hora de comprender el sistema implementado, es necesario identificar sus componentes fundamentales. Estos corresponden a una serie de elementos conceptuales que representan aspectos clave de la realidad, y que han sido modelados en el *backend* de la aplicación y estructurados en la base de datos. Comprender estos componentes permite establecer una relación clara entre la lógica del código y el dominio educativo que se busca representar.

A continuación, se presenta una descripción detallada de todos aquellos elementos representados en el sistema:

Curso

En el contexto universitario, y en particular en la carrera de ingeniería de la Facultad de Ingeniería de la UDELAR, el estudiante se enfrenta a diversas asignaturas a lo largo de su formación, cada una con su propia currícula y desafíos específicos. Con el objetivo de representar esta realidad, el sistema incorpora el concepto de **Curso**, el cual busca modelar cada una de estas materias dentro de la aplicación.

A pesar de haberse enfocado inicialmente en Programación 1 como única asignatura, el sistema ha sido diseñado de manera flexible, requiriendo muy pocas modificaciones para adaptarse a distintos cursos, elemento planteado como trabajo futuro 5.1.6, cada uno con su respectivo contenido. Esto permite que los estudiantes puedan generar cuestionarios personalizados para repasar y evaluarse en las diversas materias que enfrenten a lo largo de su trayectoria académica.

Unidad

La gran mayoría de materias a las que un estudiante se enfrenta a lo largo de su trayectoria académica presentan sus contenidos separados en secciones o pequeños grupos. Estos grupos, algunas veces definidos por las semanas en las que se imparten ciertos temas y otras por características en común entre el contenido, son representados en nuestro sistema bajo el nombre de **Unidad**.

Cada unidad está asociada a un curso y se presenta al estudiante con un título y una descripción, definidos inicialmente por un responsable de la asignatura.

En el caso concreto de Programación 1, durante el proceso de desarrollo se realizó una división en unidades basadas en similitudes temáticas y en el material disponible.

Tema

Cada unidad busca impartir un conjunto de conocimientos relacionados con un elemento en concreto de la misma. Este conjunto se modela en el sistema mediante la entidad denominada **Tema**, la cual se encuentra asociada exclusivamente a una única unidad.

El Tema constituye uno de los elementos centrales del sistema, ya que la información que representa sirve como insumo principal para la generación de cuestionarios, y define una unidad básica de conocimiento a ser evaluada.

Cada tema se encuentra compuesto por:

- Un **título** que lo identifica tanto para estudiantes como responsables de la materia.
- Una **descripción corta** la cual es utilizada para que el estudiante pueda identificar fácilmente el tema y relacionarlo con el contenido dado en clase.
- Una **descripción extensa** que define en detalle el contenido del tema. Esta es utilizada a la hora de generar una pregunta formando parte de la entrada del *LLM* 3.4.
- **Notas** que tienen el objetivo de aclarar ciertos elementos del tema para evitar errores a la hora de generar preguntas o guiar de forma más detallada el proceso.
- Una selección de **tipos de preguntas soportadas**. Según las características del tema, este puede ser más apropiado para preguntas de corte teórico, para aquellas de análisis de código, para las de seleccionar la salida correcta de un programa o incluso para llenar espacios vacíos en cierto código. Un tema puede soportar uno mas de una de estas opciones.
- Un conjunto de **Temas Previos** que indican que otros temas ya deberían saberse a la hora estudiar este. Esta información es utilizada para complejizar las preguntas generadas pudiendo agregarles contenido de temas ya dados 3.4.

De todos estos elementos, solo título y descripción corta son visibles para el estudiante, siendo el resto utilizados únicamente para la generación de cuestionarios.

Además de esto, un tema se encuentra relacionado a un conjunto de materiales o "Learning Aid" el cual es utilizado para sugerir elementos de repaso al estudiante.

En principio, se espera que estos datos sean proporcionados por un responsable del curso. Sin embargo, se ha experimentado con su generación automática a partir del material disponible, lo cual se plantea como una línea de trabajo futuro (véase sección 5.1.4)

Pregunta

Las **preguntas** son el elemento fundamental del sistema, ya que son el medio por el cual un estudiante practicará y evaluará sus conocimientos.

El enunciado de cada pregunta se genera automáticamente mediante la interacción con un gran modelo de lenguaje 3.4, a partir de los datos asociados al tema correspondiente. Para su generación, el sistema considera el título, la

descripción, las notas, el tipo de preguntas soportadas y los temas previos vinculados.

Es importante destacar que cada pregunta está asociada a un único tema y busca centrarse exclusivamente en él. Esto permite, a través de las respuestas del estudiante, identificar con claridad en qué aspectos aún necesita mejorar y en cuáles ha consolidado su aprendizaje.

Cada pregunta cuenta con un puntaje que puede ser modificado a partir de la interacción de los estudiantes, ya sea mediante votos positivos o reportes. Este puntaje permite evaluar la calidad de la pregunta, facilitando su posterior reutilización (como se plantea en el trabajo futuro 5.1.9) o descartándola en caso de contener errores. De reportarse, la pregunta será marcada de modo que no sea tenida en cuenta para las estadísticas.

Opciones Cada pregunta tiene tres **opciones**, de las que únicamente una es correcta y el resto son simples distractores.

Cada opción se genera mediante la interacción con un gran modelo de lenguaje 3.4, utilizando para su generación la pregunta previamente generada y los datos asociados al tema correspondiente.

Estas opciones, contienen un campo de texto que contiene la opción en sí misma, un indicador de si se trata de la respuesta correcta y otro campo de texto que presenta su explicación similar a otros sistemas analizados como (Capstone Edu LLC, 2025). Esta explicación puede mostrar por qué la opción es correcta, así como por qué no lo es.

Cuestionario

Un **Cuestionario** es un conjunto de preguntas generadas a partir de distintos temas, seleccionados por el estudiante al elegir las unidades a evaluar.

Cada cuestionario queda asociado a las unidades seleccionadas, al usuario que lo generó y a las preguntas generadas. En esta última relación se registra también la opción seleccionada por el estudiante en cada pregunta, lo que permite determinar si su respuesta fue correcta y el resultado final.

El cuestionario representa el punto de interacción entre el estudiante y las preguntas, tanto desde el punto de vista del modelado del sistema como de su uso en la aplicación. Además, constituye un elemento clave, ya que simboliza una sesión concreta de estudio o autoevaluación, a la cual el estudiante podrá recurrir posteriormente para analizar sus resultados y observar su progreso.

Usuario

Con el objetivo de mantener un registro de los cuestionarios resueltos, permitir la administración del curso y, sobre todo, evaluar la efectividad de la aplicación, es necesario que el sistema modele a los usuarios.

Cada usuario posee un nombre, nickname, correo electrónico y una contraseña que le permite acceder al sitio. A su vez, se relaciona con cursos, teniendo un curso seleccionado en concreto que será el que vea al iniciar sesión.

Por último, cada usuario cuenta con un **Rol** definido, el cual lo identifica como **estudiante** o **profesor**. A modo de simplificar el modelado, se optó por que los roles sean excluyentes, por lo que si un docente fuera estudiante, debería serlo con dos usuarios distintos. Si bien no fue contemplado, pequeños cambios permitirían la incorporación de usuarios con múltiples roles.

Profesor Los profesores, si bien se busca que tengan que dedicar la menor cantidad de tiempo posible a la herramienta, cumplen un rol clave en dos aspectos fundamentales. Por un lado, son responsables de la creación y el mantenimiento de los contenidos del curso en la plataforma. Por otro, tienen acceso exclusivo a los reportes generados, que les permiten evaluar la eficacia del sistema y obtener información valiosa para enriquecer su tarea pedagógica.

Estudiante El estudiante es el eje central de la aplicación. Tiene acceso al contenido del curso, puede generar cuestionarios en base a las unidades seleccionadas y revisar sus resultados luego de completarlos. Se espera que sea el tipo de usuario con mayor actividad dentro del sistema, y es el único al que pueden estar asociados los cuestionarios generados.

Un diagrama general de las entidades mencionadas y sus relaciones puede encontrarse en la figura 3.1.

3.1.5. Funcionalidades

El sistema cuenta con distintas funcionalidades, las cuales, excluyendo el inicio de sesión, se encuentran divididas entre aquellas destinadas al estudiante y aquellas propias del docente.

En cuanto a las funciones básicas como el inicio de sesión, es importante señalar que la creación y edición de usuarios no fueron contempladas dentro del alcance del desarrollo actual. Esta decisión responde a la intención de enfocar los esfuerzos en aquellas funcionalidades de mayor valor, dejando como trabajo futuro la incorporación de estas tareas administrativas, cuya implementación resulta sencilla.

Estudiante

El estudiante, una vez inicia sesión, puede acceder a las siguientes funcionalidades:

Información del Curso Lo primero con lo que se encuentra el estudiante al ingresar al sistema es una página que presenta la información general del curso del que forma parte. En ella se muestra el título del curso, su descripción, las unidades que lo componen organizadas en orden, una breve descripción de cada unidad y los temas asociados a cada una, acompañados de una definición concisa.

Esta vista permite al estudiante obtener una visión general del curso, explorar su estructura y decidir qué aspectos desea repasar.

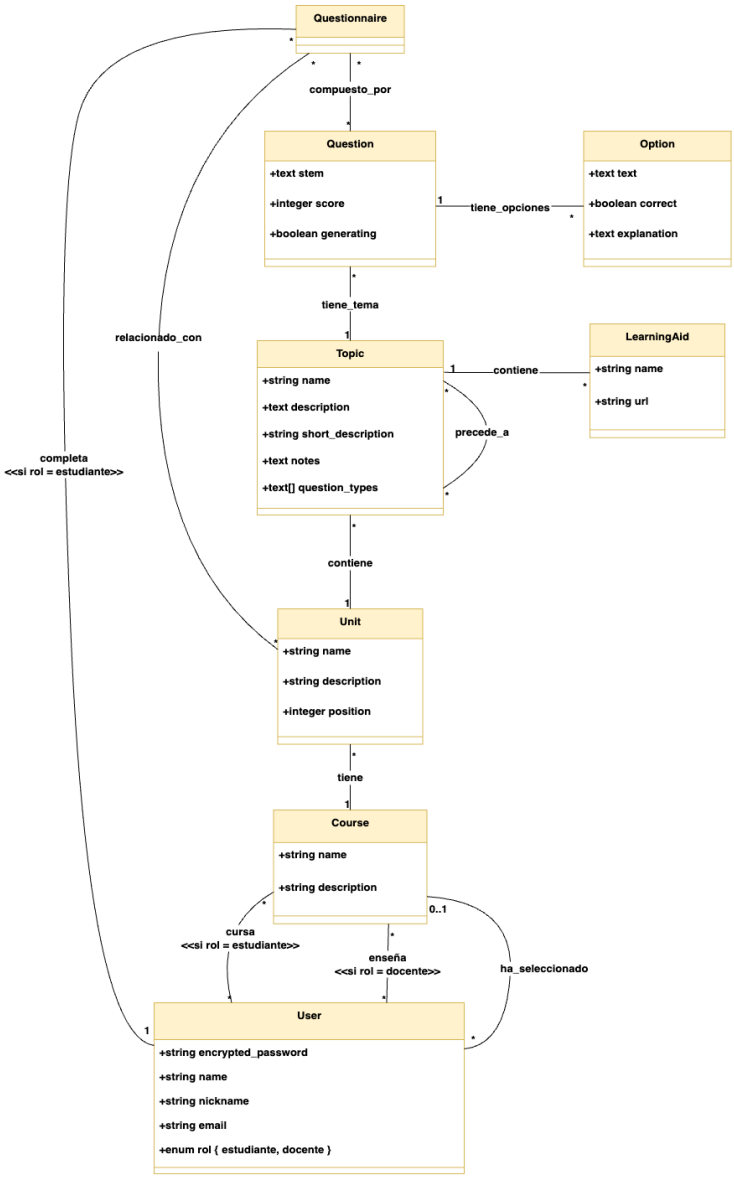

Figura 3.1: Diagrama UML del sistema

Creación de Cuestionario Una vez identificadas las unidades que quiere evaluar, el estudiante es capaz de hacer su selección y presionar un botón para iniciar el proceso de generación del cuestionario.

Este proceso se realiza de forma asíncrona y puede demorar algunos minutos, ya que implica la interacción con un modelo de lenguaje de gran escala. Duran-

te la espera, el sistema presenta una pantalla de carga con pequeños consejos sobre programación en Pascal, pensados para aprovechar el tiempo y reforzar conocimientos.

Cuestionario Dentro del cuestionario, el estudiante se enfrenta a cada pregunta generada junto con sus respectivas opciones.

Una vez que responde, puede acceder a una explicación que justifica la corrección o el error, permitiéndole comprender el razonamiento adecuado en caso de haber fallado. Luego, puede avanzar a la siguiente pregunta, repitiendo este proceso hasta completar el cuestionario.

Además de responder, el estudiante tiene la posibilidad de evaluar cada pregunta: puede reportarla si encuentra errores que justifiquen descartarla, o resaltar su utilidad para el aprendizaje.

Al finalizar el cuestionario, el estudiante accede a un análisis detallado de sus resultados. Puede repasar las preguntas y respuestas, identificar los errores cometidos y visualizar qué temas evaluados presentan mayores dificultades. Además, el sistema le permite enfocarse en esos temas que necesita reforzar, brindándole acceso directo al material relacionado de forma simple.

Mis Cuestionarios En la sección "Mis Cuestionarios", el estudiante tendrá un resumen del trabajo realizado durante su uso de la aplicación.

Desde allí, el estudiante puede visualizar cuántos cuestionarios ha completado, acceder a cuestionarios anteriores observando su rendimiento en ellos, y conocer qué temas le han resultado más desafiantes y en los que ha tenido mejor rendimiento en base a su rendimiento acumulado. A partir de los temas que requieren refuerzo, el sistema ofrece acceso directo al material necesario para su repaso.

Profesor

El profesor, una vez inicia sesión, puede acceder a las siguientes funcionalidades:

Administración del curso El profesor ejerce el rol de administrador del curso, por lo que, una vez iniciada la sesión, accede a una página similar a la del estudiante, con la diferencia de que puede editar la información allí presentada.

Tiene la capacidad de crear, modificar y eliminar tanto los datos generales del curso como los contenidos de sus unidades y temas. En estos últimos, puede editar tanto la información visible para el estudiante como aquella utilizada para la generación automática de preguntas, lo que le otorga control sobre este proceso.

Reportes Al acceder a la sección de **reportes**, el docente puede obtener un conjunto de datos obtenidos en base al uso de la aplicación por parte de los estudiantes. El objetivo de los mismos es poder trazar luego una relación entre

el uso del sistema y el rendimiento de cada estudiante, la efectividad de la herramienta y que aspectos del curso son los que generan mayor problemas a la hora de repasar.

Se cuenta con las siguientes estadísticas generales de uso:

- Promedio de cuestionarios realizados por estudiante.
- Tasa general de aciertos, es decir el resultado promedio de los cuestionarios realizados.
- Temas con mejor y peor desempeño de los evaluados.
- Cantidad de cuestionarios que han completado al menos un cuestionario.

Y también se cuenta con estadísticas por estudiante, para evaluar su desempeño en específico. En estas se puede ver:

- Total de cuestionarios completados.
- Tasa de respuestas correctas.
- Temas en los que cuenta con mejor rendimiento.
- Temas donde presenta el peor rendimiento.
- Cantidad de respuestas correctas.
- Cantidad de respuestas incorrectas.
- Desempeño en los últimos cinco cuestionarios completados.

Exportación de preguntas En la sección de reportes, el profesor cuenta con un botón que le permite exportar las preguntas generadas en formato CSV. Esta funcionalidad facilita su reutilización en otras instancias y permite evaluar su calidad para realizar las modificaciones necesarias.

3.2. Gestión del proceso de diseño e implementación

El desarrollo del proyecto implicó una importante dimensión organizativa con el objetivo de establecer un enfoque de trabajo ordenado, colaborativo y flexible. Esto permitió abordar de manera eficiente las distintas etapas del proyecto: la investigación inicial, la definición de requerimientos y particularmente el diseño del sistema, la implementación, las validaciones con docentes y las iteraciones finales.

3.2.1. Métodología de trabajo

Metodología ágil

A lo largo del proyecto se adoptó un enfoque ágil basado en Kanban, una metodología de gestión centrada en la mejora continua mediante tableros visuales. En lugar de planificar todo por adelantado, Kanban (Atlassian, s.f.) permite que las tareas se "arrastren" progresivamente desde una lista de pendientes, llamada *backlog*, hacia la finalización, promoviendo la mejora incremental. Este método facilitó la coordinación del equipo, permitiéndonos adaptar las prioridades según el avance real del trabajo. Esto resultó particularmente útil durante las etapas de investigación, en las que las tareas presentaban gran diversidad en cuanto a tipo y duración, por lo que la flexibilidad que provee este enfoque fue clave.

Como espacio central de organización se utilizó un tablero en Notion (Notion, s.f.). En este definimos y clasificamos todas las tareas en forma de tickets, organizados en columnas según su estado:

- *Blocked*: tareas que no pueden avanzar por alguna dependencia o problema externo.
- *To Do*: tickets pendientes aún no iniciados.
- *In Progress*: tareas en curso.
- *In Review*: ticket que fueron completados pero requieren revisión por parte de otro integrante del equipo.
- *Done*: tareas terminadas y validadas.

Para mejorar el proceso, cada ticket cuenta a su vez con etiquetas para indicar su categoría: investigación, desarrollo y documentación.

Ceremonias

Como parte de la metodología establecida, se definieron una serie de reuniones con el fin de fijar una comunicación regular entre el equipo, y en ocasiones, con los tutores. Estas ceremonias fueron las siguientes:

- *Stand-ups*: encuentros breves (generalmente cada dos o tres días) donde cada integrante compartía qué hizo, qué va a hacer y si tenía algún bloqueo. Esta rutina fue importante para detectar a tiempo los problemas y mantener el ritmo de trabajo.
- *Grooming*: reuniones en las que definimos y refinamos las tareas, discutiendo su prioridad, complejidad y criterios de aceptación. El objetivo de estas reuniones es asegurarnos que las tareas estén bien definidas antes de comenzar.
- Reuniones con docentes: también realizamos encuentros periódicos con los tutores del proyecto para validar decisiones técnicas y obtener retroalimentación sobre lo hecho.

Iteración y prototipado

Desde el inicio del proyecto decidimos trabajar de forma iterativa, construyendo versiones parciales del sistema que pudiéramos probar y mejorar en ciclos cortos. Este enfoque se aplicó también a las primeras fases de investigación y documentación, donde se dieron iteraciones hasta lograr el entregable final.

Además, durante las primeras semanas de la etapa de implementación, nos enfocamos en desarrollar un prototipo simple que permitiera probar la integración con un modelo de lenguaje para la generación de preguntas. Luego, fuimos incorporando nuevas funcionalidades por etapas: autenticación y roles de usuario, visualización y edición de cursos, creación de cuestionarios y reportes para los docentes. Cada iteración fue guiada por una planificación en la que se priorizaron las tareas y analizaron las dependencias entre ellas.

3.2.2. Flujo de trabajo

El desarrollo del sistema estuvo precedido por un proceso de investigación técnica y diseño conceptual que sentó las bases para la etapa de desarrollo.

Investigación

Inicialmente, realizamos una revisión del estado del arte en el área de *Question Generation*. Estas instancias de investigación también fueron organizadas mediante tickets, al igual que las tareas de desarrollo. Cada investigación derivó en documentos que buscaron sintetizar los hallazgos, y luego funcionaron como insumo tanto para la toma de decisiones técnicas como para la redacción de este informe.

A partir de esta revisión, se identificaron distintos enfoques posibles para la generación de preguntas, como el *fine-tuning* de modelos, el uso de *datasets* específicos o el *prompt engineering* de los cuales se hablara en detalle en los capítulos de desarrollo y experimentación.

Diseño y toma de decisiones

Luego de analizar el material generado y realizar la puesta en común, se tomó la decisión de optar por un sistema de generación de preguntas basado en *prompt engineering*, enfoque detallado en la sección 3.4.

Con este enfoque definido, se procedió a diseñar la arquitectura general del sistema, incluyendo tanto la estructura técnica como los principales flujos de interacción.

A **nivel técnico**, seleccionamos las tecnologías principales: una API en Ruby on Rails (Ruby on Rails Core Team, 2025) para el backend, una interfaz en React (Meta Open Source, 2024) para el frontend, y el uso de un modelo de lenguaje de gran escala a través de una API externa para la generación de preguntas. Estas decisiones se encuentran detalladas en la sección 3.3.

Durante esta etapa también definimos las **entidades** fundamentales que debían ser representadas en el sistema: cursos, unidades, temas, usuarios, cuestionarios y preguntas. Esta definición fue clave para estructurar la base del sistema y guiar tanto el diseño visual como la lógica de negocio.

Una vez establecida la arquitectura y modelado el dominio, se realizó el **diseño de interfaz** del sitio web utilizando la herramienta Figma (Figma Inc., 2025). Estos diseños nos sirvieron como referencia directa durante la implementación.

Desarrollo

Ya en la etapa de desarrollo, utilizamos Git (GitHub, 2024) como sistema de control de versiones, con una convención de nombres para las ramas basada en el tipo de tarea: *feature/[id-del-ticket]* para nuevas funcionalidades, y *fix/[id-del-ticket]* para correcciones de errores. Además, cada tarea era revisada mediante pull requests que eran evaluadas por el resto del equipo. Esta instancia de revisión de código nos permitió asegurar buenas prácticas y detectar errores.

Estas definiciónes técnicas se pueden encontrar con mayor detalle en la sección 3.3.

3.3. Definiciones Técnicas

Para llevar a cabo este proyecto, además de los procesos mencionados y las herramientas utilizadas, fue necesario definir un stack tecnológico para su desarrollo. A continuación, se presenta el mismo, junto al por qué de su elección.

3.3.1. Control de Versiones

Parte fundamental del proceso de desarrollo consiste en el manejo de las distintas iteraciones del código. Este manejo debe priorizar la paralelización del trabajo y el registro de los cambios, todo ello con el objetivo de agilizar y simplificar el proceso de desarrollo.

Teniendo estas necesidades, se optó por un sistema de control de versiones distribuido, es decir, aquel donde cada desarrollador cuenta con una copia local del proyecto y el servidor se encarga de registrar y permitir manejar las diferencias entre las distintas ramas (versiones del código) (GitLab, 2023). De las distintas opciones disponibles dentro de este tipo de sistemas, elegimos utilizar Git, en particular mediante la plataforma GitHub que permite de forma simple el seguimiento y administrción del código, el proceso de revisión del mismo, compartir el trabajo fácilmente y la colaboración entre los integrantes (GitHub, 2024).

La decisiíon de esta plataforma ante otras opciones con funcionalidades similares basadas en Git como GitLab (GitLab Inc., 2025) o Azure Repos (Microsoft Corporation, 2025) se basa principalmente en la familiaridad de los integrantes del equipo con la herramienta.

3.3.2. Base de Datos y sistema de gestión

La base de datos es un componente esencial de cualquier sistema informático que requiera la persistencia de sus datos. En nuestro sistema, esto se hace fundamental no solo para el manejo de usuarios y almacenamiento de los datos necesarios para generar preguntas, sino que es la clave para generación de reportes y análisis de su eficiencia.

Nuestros datos cuentan con una estructura clara, tipos definidos previamente, relaciones claras entre las distintas entidades y es posible que sean necesarias consultas complejas al momento de generar reportes. Es por esto que se utilizó una base de datos relacional, la cual se adapta de mejor manera a las características planteadas (Amazon Web Services, 2024).

Para el manejo de esta base de datos relacional, es necesario el uso de un gestor de bases de datos. La herramienta elegida es PostgreSQL, es un gestor de código abierto con 35 años de desarrollo activo, confiable, robusto y con el cual todos los integrantes del grupo tenemos experiencia trabajando (The PostgreSQL Global Development Group, 2025).

Por último y debido al uso de Ruby on Rails, decisión tecnológicas explicada más adelante, la aplicación hará uso de **ActiveRecord** (Ruby on Rails, 2024) como ***Object-Relational Mapping*** (***ORM***), lo cual permite interactuar con la base de datos utilizando objetos de Ruby, facilitando la implementación, el mantenimiento del código y la interacción entre la base de datos y el resto del sistema.

3.3.3. Cliente - Servidor

Desarrollar una aplicación web implica adoptar una arquitectura cliente-servidor. Esto requiere definir qué tecnologías se utilizarán tanto para el desarrollo del cliente como del servidor, así como la estrategia de comunicación que permitirá su interacción.

Es importante destacar, que a fines prácticos, cliente y servidor son dos aplicaciones independientes, cada una con su propio repositorio en GitHub y teniendo necesidades tecnológicas diferentes. Sin embargo, al interactuar forman en conjunto el sistema desarrollado.

Arquitectura REST

Para la comunicación entre ambas partes del sistema, optamos por una arquitectura REST, la cual presenta varias ventajas para el desarrollo de sistemas distribuidos. Algunas de ellas son:

- Interoperabilidad: Al utilizar HTTP como protocolo subyacente y formatos estándar como JSON, permite fácilmente comunicar servidor y cliente implementados con distintas tecnologías manteniendo la independencia entre ambos.

- Reusabilidad y mantenimiento más sencillo: Cada recurso tiene una URL única, la cual puede ser utilizada por diferentes clientes siempre y cuando respeten el formato definido, permitiendo futuras expansiones e integraciones. Esto permitiría por ejemplo, sin mayor dificultad, el consumo de las preguntas creadas para su uso externo o el desarrollo de una aplicación mobile.

- Escalabilidad: Al estar diseñado para funcionar bien en sistemas distribuidos, el uso de REST permite escalar horizontalmente (por ejemplo, con balanceadores de carga y microservicios) sin modificar la arquitectura central.

- Simplicidad y uso de estándares web: Aprovecha los métodos HTTP estándar (GET, POST, PUT y DELETE.), lo cual simplifica el diseño y la comprensión del sistema.

Servidor - Ruby on Rails

Como tecnología para implementar el backend optamos por utilizar Ruby (Ruby Programming Language Core Team, 2025) en su última versión (Ruby 3.4.1), en particular su framework más popular Ruby on Rails (Ruby on Rails Core Team, 2025) (*RoR*) en su versión 8, dado que presenta varias ventajas tanto técnicas como prácticas. A continuación se listan algunos de los puntos tenidos en cuenta a la hora de realizar la elección:

- Agilidad del desarrollo: RoR adopta el principio de *convención sobre configuración* (Hansson, 2025), que promueve el uso de convenciones establecidas en lugar de configuraciones explícitas. Esto puede ser complejo en un primer acercamiento pues requiere aprender estas convenciones, sin embargo, con la experiencia se traduce en una aceleración significativa del desarrollo. Cabe destacar que todos los integrantes del equipo cuentan con experiencia previa en el lenguaje, lo cual permite aprovechar plenamente sus beneficios.

- Comunidad activa: Con más de veinte años, RoR cuenta con una comunidad amplia y consolidada. Esto se traduce en abundante documentación, foros, tutoriales, y soluciones previamente desarrolladas para problemas comunes, elementos claves para garantizar un desarrollo agil manteniendo la calidad del sistema.

- Arquitectura clara y mantenible (MVC): RoR promueve una estructura basada en el patrón Modelo-Vista-Controlador (MVC), lo que favorece el mantener el código organizado, facilitando la separación de responsabilidades y mejorando la mantenibilidad del sistema a largo plazo.

- Tecnología moderna: El uso de Rails 8, ultima versión de Rails salida a principios de 2025, asegura que el proyecto se apoya en una tecnología actualizada, lo cual implica compatibilidad con los estándares más recientes

del desarrollo web, mejoras en seguridad, rendimiento y buenas prácticas modernas.

- Ecosistema maduro: Rails se beneficia de un ecosistema muy rico de librerías, conocidas como *gemas*, que permiten añadir funcionalidades complejas con mínima configuración. Esta disponibilidad reduce significativamente los tiempos de desarrollo y permite concentrar los esfuerzos del equipo en la implementación de la idea principal del proyecto, en lugar de reinventar soluciones ya establecidas.

Dentro de este ecosistema maduro que Ruby y en particular RoR ofrece, hemos decidido usar un conjunto de librerías (*gemas*) para incorporar funcionalidades elementales de forma eficiente. A continuación se mencionan las más importantes y la razón de su elección.

Devise Token Auth Dada la necesidad de guardar información relacionada a cada usuario, con el fin de contrastarla posteriormente con sus resultados académicos y así evaluar la eficacia de la aplicación, se vuelve fundamental contar con un sistema de autenticación robusto y eficiente.

Si bien existen diversas alternativas para implementar un sistema con estas características, optamos por utilizar **Devise** (Heartcombo, 2009), en conjunto con **Devise Token Auth** (Hurley, 2014), por ser una de las soluciones más consolidadas y con mayor respaldo dentro de la comunidad de desarrollo en Ruby on Rails.

Estas gemas, utilizadas de forma complementaria, ofrecen un manejo completo y seguro del ciclo de autenticación a través de una API.

Para esto se hace uso de un sistema basado en tokens, los cuales son enviados a través de los headers de cada interacción, y permiten precindir del uso de cookies. Esta modalidad se adapta tanto a nuestras necesidades actuales como a una posible expansióón futura hacia aplicaciones móviles o frontends desacoplados.

CanCanCan Otra de las gemas utilizadas es **CanCanCan** (CanCanCommunity, 2015). Ella nos permite un manejo de la autorización dentro de la aplicación, pudiendo mantener dos o más roles separados, en principio estudiante y profesor, y permitir al profesor acceso a aquella información sobre el curso o el estudiante que sea necesario sin comprometer el funcionamiento o la privacidad de los datos.

Sidekiq Teniendo en mente una aplicación que pueda escalar y la necesidad de ejecutar procesos o interactuar con sistemas que puedan demorar en ejecutar, como es el caso de algunos LLMs, es que se vuelve necesario el uso de un procesador de tareas en segundo plano. Esto permite ejecutar ciertas tareas de forma asíncrona, sin bloquear el hilo de ejecución principal, y por tanto, mejorando la experiencia del usuario.

Con este objetivo es que surge la posibilidad de utilizar **Sidekiq** (Perham, 2012), un procesador de tareas asincrónicas gratuito de alto rendimiento que permite una gran flexibilidad gracias a sus distintas configuraciones y garantiza la fiabilidad con su manejo de errores y reintentos.

RSpec En todo sistema, es importante garantizar la calidad del código y minimizar la aparición de errores. Para esto, resulta esencial contar con un conjunto de tests que validen el correcto funcionamiento de la aplicación.

Si bien no se adoptó un proceso de **desarrollo guiado por pruebas**, sí se decidió enfatizar la creación de pruebas unitarias y funcionales, las cuales abarcan tanto la lógica de negocio a nivel de modelo como la validación de los distintos endpoints definidos en los controladores.

Para implementar estos tests, se optó una de las gemas más utilizadas, con mayor comunidad y documentación en el ecosistema de RoR llamada RSpec (RSpec Team, 2025). Esta ofrece un ecosistema maduro y amplio, con gran cantidad de gemas que complementan su funcionamiento y simplifican su uso, una muy simple integración con Rails y por sobre todo una sintaxis legible y expresiva que permite su fácil comprensión y mantenimiento.

Cliente - React

Como tecnología para implementar nuestro frontend optamos por utilizar React (Meta Open Source, 2024) en su última versión estable (React 19), publicada en diciembre de 2024. Sumado a la experiencia que los miembros del equipo contamos con esta librería basada en JavaScript, existen una cantidad de ventajas que esta ofrece que la hacen ideal para el uso pensado, a continuación se listan algunas de ellas:

- Arquitectura basada en componentes: React permite mantener una arquitectura basada en componentes, los cuales permiten no solo mantener un código limpio y una estructura ordenada, sino que reutilizar la mayor cantidad de código posible, disminuyendo errores y acelerando el proceso de desarrollo.

- Documentación y comunidad activa: Al tratarse de una de las librerías más utilizadas para el desarrollo de interfaces de usuario, esta cuenta con una amplia comunidad y años de documentación y desarrollo a sus espaldas.

- Interfaz rápida y responsiva: React utiliza una tecnología denominada *Virtual DOM*, que le permite identificar de forma eficiente qué partes de la interfaz requieren ser actualizadas, evitando así la necesidad de recargar completamente la página. A su vez, el uso de herramientas como los *hooks* facilita el manejo del estado de forma declarativa, permitiendo que los cambios en los datos se reflejen automáticamente en la interfaz. Esta combinación de rendimiento y reactividad contribuye al desarrollo de aplicaciones web más ágiles, interactivas y fáciles de mantener.

- Ecosistema amplio, modular: React al igual que Rails, cuenta con una gran cantidad de bibliotecas y herramientas complementarias que permiten incorporar funcionalidades de forma eficiente y experimentar con distintas integraciones de forma sencilla.

Haciendo uso de este ecosistema amplio y modular, y con el objetivo de desarrollar una aplicación que se encuentre al día con las últimas tendencias en la industria, se mencionan a continuación algunas incorporaciones hechas por sobre una aplicación React por defecto.

Typescript En lugar de utilizar JavaScript puro para desarrollar el proyecto con React, se optó por el uso de **TypeScript** (Microsoft, 2024) , un lenguaje de programación fuertemente tipado que se transpila a JavaScript, lo que permite ejecutarlo en cualquier entorno donde JavaScript sea compatible.

Esta elección aporta mayor seguridad en tiempo de desarrollo y mejora la mantenibilidad del código a lo largo del proyecto, ya que los tipos actúan como documentación implícita, y permiten detectar errores al momento de desarrollar.

Tailwind Para agilizar el proceso de desarrollo y la definición de estilos, se optó por utilizar **Tailwind** (Tailwind Labs, 2024), un framework CSS del tipo *utility-first*, es decir, que ofrece una serie de clases utilitarias predefinidas que puedes usar directamente en la página en lugar de definir reglas css personalizadas.

Por sobre Tailwind, a su vez, con el objetivo de agilizar aún mas el desarrollo y mantener una linea de diseño coherente a lo largo de la página, se utiliza **Shadcn** (Shadcn, 2024) una librería de componentes de UI predefinidos, fácilmente utilizables y construidos cumpliendo estándares de accesibilidad.

Diagrama de la Aplicación

La figura 3.2 muestra a grandes rasgos un diagrama de la arquitectura adoptada con sus respectivas tecnologías. Cada una de sus componentes tiene sus complejidades y estructura interna, se hará énfasis en ellas de ser necesario en las distintas secciones de este informe.

3.3.4. Modelo de Lenguaje

Una parte esencial del stack tecnológico es el modelo de lenguaje, encargado del procesamiento del material y la generación de preguntas en base al mismo. Esta sección describe tanto las alternativas disponibles como las posibilidades de integración con dichos modelos.

Integración del Modelo

Actualmente, el panorama de los modelos de lenguaje es diverso y en constante evolución, con numerosas variantes y propuestas emergentes de manera

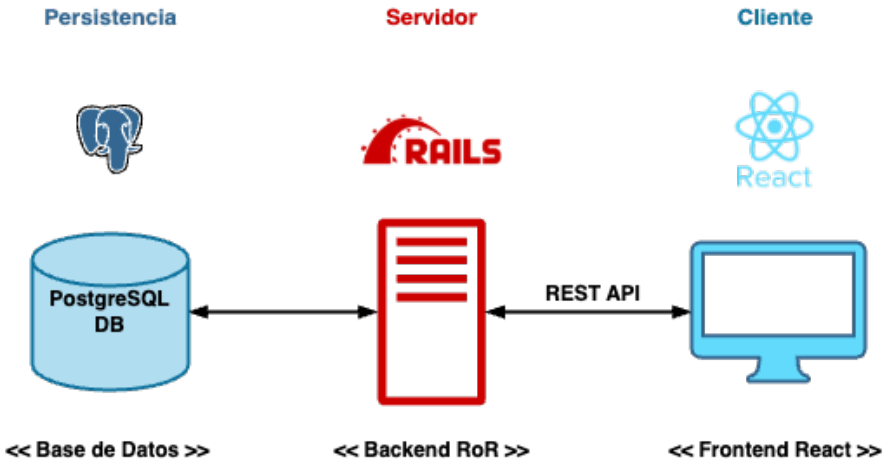

Figura 3.2: Diagrama general de la aplicación

continua. Sin embargo, al momento de considerar su integración en un sistema, las posibilidades se agrupan en dos enfoques principales:

Alojamiento de un modelo propio Consiste en desplegar un modelo de lenguaje de gran escala en servidores disponibles o alquilados. Esta opción permite un mayor control sobre el modelo y sus datos, pero requiere utilizar modelos de código abierto y afrontar limitaciones técnicas en base a los equipos disponibles.

Modelo como servicio Es la opción más extendida actualmente. Consiste en acceder a un modelo de lenguaje a través de una API proporcionada por un proveedor externo como OpenAI, Google, Anthropic, entre otros. Esta modalidad simplifica la integración con la aplicación y permite acceder a un rango más amplio de modelos avanzados sin la necesidad de infraestructura propia; sin embargo, implica depender de un tercero que definirá precios y políticas de manejo de datos.

Cada una de estas alternativas ofrece sus ventajas y desventajas; en base a ellas y luego de realizar un análisis profundo que se encuentra detallado en el documento “Anexo: Decisiones Sobre el Modelo de Lenguaje”, en la sección “Uso como Servicio frente a Implementación Local”, se optó por utilizar un modelo de lenguaje como servicio. Esta decisión, si bien sacrifica el control sobre los recursos y los datos manejados, presenta una mayor viabilidad en términos de costos, facilidad de incorporación a nuestro sistema y posibilidades de explorar distintos modelos. Además, no representa un riesgo significativo, dado que los datos manejados no son de carácter personal.

Elección del Modelo

Con esto en mente, es necesario definir qué modelo en concreto se utilizará, una decisión compleja debido a la gran cantidad de alternativas existentes. A la hora de tomar esta decisión, se ponderaron distintos elementos con el objetivo de lograr un sistema moderno, flexible y accesible, proceso que se encuentra detallado en el documento "Anexo: Decisiones Sobre el Modelo de Lenguaje", en la sección "Elección final del modelo: Google Gemini 2.5 Pro".

Rendimiento cercano al estado de arte Luego de experimentar con diferentes modelos de distintas capacidades técnicas, obteniendo resultados que no alcanzaban la precisión o coherencia requerida, se optó por buscar un modelo cercano al estado del arte posible. Para ello se realizó una investigación extensa de distintos benchmarks, pruebas mediante las cuales los modelos de lenguaje son evaluados, permitiendo su comparación. De este modo se descartaron aquellos que se encontraban lejos de los primeros puestos, con la idea de conseguir un sistema lo más eficaz posible.

Costos, desarrollo y experimentación Otro elemento clave a tener en cuenta fue la facilidad de integrar el modelo seleccionado a nuestro sistema. Una vez definido el uso de APIs para la comunicación, se revisó las distintas formas de interacción con la misma y las herramientas disponibles para esta tarea. Modelos que ofrecieran una forma de comunicación fácilmente intercambiable con otros fueron ponderados positivamente debido a las posibilidades de experimentación y adaptabilidad que esto representa. También lo fueron aquellos que presentan menores costos, a modo de reducir los costos de desarrollo y funcionamiento del sistema final.

A partir del análisis realizado en mayo de 2025 y considerando la buena conjunción de los factores evaluados, se optó por utilizar el modelo ofrecido por Google: Gemini 2.5 Pro (Kavukcuoglu, 2025). Este ofrece resultados a la par de los mejores modelos de lenguaje a la fecha en las pruebas analizadas, cuenta con una API similar a la de otros proveedores como OpenAI, lo que facilita su intercambiabilidad, y dispone de un entorno de prueba gratuito durante los primeros tres meses de desarrollo.

3.4. Interacción con modelo de lenguaje de gran escala

Para lograr la generación de cuestionarios, y en concreto, la generación de preguntas de múltiple opción, el sistema se apoya en un **modelo de lenguaje a gran escala** o **LLM** (del inglés, ***Large Language Model***). A partir del contenido del curso e información relacionada al mismo y su forma de enseñarse, este modelo busca generar sus componentes claves: enunciado, respuesta correcta y distractores.

La interacción con este modelo se realiza mediante instrucciones, en este caso textuales, denominadas ***prompts***, diseñadas para guiar al modelo a realizar una tarea específica. Un *prompt* puede adoptar la forma de una pregunta, una instrucción, una descripción del contexto o una combinación de estos elementos, y su correcta formulación es fundamental para obtener respuestas relevantes por parte del modelo.

Este capítulo describe en detalle cómo se integra el modelo de lenguaje en el flujo de la aplicación, qué servicios fueron desarrollados para facilitar esta interacción, y cómo se estructura la comunicación entre el sistema y el modelo. Asimismo, se exponen las estrategias de diseño utilizadas para construir *prompts* efectivos, y las decisiones tomadas para asegurar la calidad de las preguntas generadas.

3.4.1. Enfoque de la interacción

En el diseño del sistema, la interacción con el modelo de lenguaje tiene un propósito específico: generar preguntas de múltiple opción una a una. Aunque el modelo recibe un encuadre general sobre el contexto y el propósito de las preguntas, cada invocación está destinada a la creación de una única pregunta, lo cual permite mantener un mayor control sobre el proceso de generación.

Cada pregunta contiene en principio: un tema asociado, una breve descripción que delimita el contenido sobre el cual debe tratar, y la definición de un estilo de pregunta, que define el formato que se quiere utilizar para la evaluación. La construcción del cuestionario como entidad completa es una tarea separada que no involucra directamente al modelo de lenguaje.

Cada pregunta generada debe contener:

- Un enunciado, o *stem*, que plantea el problema a resolver.
- Una única respuesta correcta. Acompañada de una breve explicación de su correctitud y razonamiento a seguir.
- Dos distractores (opciones incorrectas), cada uno con su propia explicación que justifica por qué es un distractor plausible pero incorrecto.

La interacción con el modelo ocurre de forma bajo demanda: cuando el estudiante solicita la creación de un nuevo cuestionario, el sistema inicia la generación de las preguntas de manera secuencial basándose en las unidades escogidas. Además, con el objetivo de reducir el tiempo de espera percibido, ni bien se completa la primera pregunta, esta se presenta al estudiante, permitiéndole comenzar con la resolución mientras el resto del cuestionario se continúa generando en segundo plano.

3.4.2. Arquitectura de servicios

En aplicaciones Ruby on Rails, es común encapsular lógica de negocio compleja dentro de *service objects*. Estos son objetos Ruby simples (en inglés *Plain*

Old Ruby Objects, o POROs) diseñados para ejecutar una acción bien definida dentro del dominio de la aplicación. (Gilani, 2018)

Siguiendo este enfoque, nuestro sistema implementa una arquitectura basada en servicios, que interactúan entre sí y con el modelo de lenguaje a través de llamadas independientes. El objetivo de esta división en servicios es mantener separadas las distintas responsabilidades del proceso de generación de preguntas, promoviendo la modularidad y facilitando la reutilización de código, testing y trazabilidad de los componentes.
Los servicios que conforman esta arquitectura son:

- **QuestionGenerationService - Servicio de generación de enunciado y respuesta correcta**: este servicio, a través del LLM, es responsable de la creación del enunciado de la pregunta, la respuesta correcta y la explicación de la misma.
- **DistractorsGenerationService - Servicio de generación de distractores**: una vez generado el núcleo de la pregunta, este servicio se encarga de solicitar al modelo un conjunto de distractores plausibles. Cada distractor se acompaña de una explicación que justifica por qué esa opción es razonable pero incorrecta.
- **MCQGenerationService - Servicio de composición de pregunta completa**: este componente actúa como orquestador. Su función es coordinar las invocaciones a los dos servicios anteriores y consolidar sus respuestas en una única estructura que representa una pregunta de múltiple opción.

Estos servicios se ejecutan desde el backend del sistema, el cual se comunica con el LLM a través de la API de Gemini. En particular, esta comunicación se efectúa mediante la gema "ruby-openai", que simplifica la construcción y envío de requests desde Ruby, y permite abstraerse de muchos detalles de bajo nivel relacionados con la comunicación HTTPS, manejo de errores y formateo de requests. Esta librería, si bien está diseñada principalmente para trabajar con los modelos de OpenAI, también permite interactuar con modelos de otros proveedores. Esta capacidad fue aprovechada durante la etapa de experimentación para probar distintos modelos de lenguaje, entre ellos Gemini.

3.4.3. Mecanismo de interacción

En esta sección se describe en detalle cómo se estructura la interacción con el modelo de lenguaje, incluyendo la información que se le proporciona, la construcción de los *prompts* y las estrategias empleadas para guiar la generación.

Información transmitida al modelo de lenguaje

Al momento de la generación, el modelo debe ser contextualizado con información relevante para asegurar una salida de calidad, alineada con el curso y las unidades seleccionadas para el cuestionario.

Dentro de las *prompts* utilizadas, la información se organiza de la siguiente manera:

- **Contexto del curso**: en esta sección se brinda un contexto amplio del curso compuesto por su descripción, sus objetivos pedagógicos y algunas observaciones concretas que deben tenerse en cuenta al momento de generar pregunta, por ejemplo notación de Free Pascal.

  La información utilizada corresponde al curso *Programación 1* y fue recopilada a partir de fuentes públicas, conversaciones con los docentes y un proceso iterativo de experimentación. No obstante, gracias a la arquitectura modular del *prompt*, sería posible adaptar dinámicamente este componente a otros cursos con modificaciones mínimas.

  El contexto del curso utilizado y su formato puede encontrarse en detalle en el documento "Anexo: Prompt para la Generación del Enunciado y Clave" junto al resto de componentes de la *prompt* inicial.

- **Definición del formato de evaluación (MCQ)**: con el objetivo de garantizar la calidad de las preguntas generadas, se describe detalladamente qué es una pregunta de múltiple opción, especificando sus componentes, junto con lineamientos claros para redactar el enunciado, su respuesta única y sus dos distractores.

  Esta información se encuentra distribuida entre las distintas *prompts* utilizadas ya que guía tanto el proceso de generación del *stem* y pregunta correcta tal como se puede ver en el documento "Anexo: Prompt para la Generación del Enunciado y Clave", como de sus distractores detallado en el documento "Anexo: Prompt para la Generación de Distractores".

- **Tema de la pregunta**: cada pregunta generada se enfoca en un único tema, el cual es tomado de las unidades seleccionadas por el usuario a la hora de generar un cuestionario. Esta limitación tiene dos objetivos: por un lado, acotar el conocimiento general del modelo y evitar que incorpore elementos irrelevantes o avanzados fuera del alcance del curso, y por otro, evaluar un único tema por vez permite al estudiante identificar fácilmente sus debilidades y como mejorarlas.

  Como parte de este proceso de generación guiada, también se incorporan los temas marcados como previos al tema principal de la pregunta. Estos se utilizan exclusivamente para complejizar la formulación, sin que ello implique un cambio en los aspectos a evaluar. Con esta estrategia se buscó resolver una de las principales limitaciones detectadas en las primeras etapas del desarrollo: la tendencia del modelo a generar preguntas demasiado simples por falta de contexto adicional.

  Teniendo en cuenta estos aspectos, y siguiendo el mecanismo detallado en el documento "Anexo: Prompt para la Generación del Enunciado y Clave", se utiliza la descripción del tema, las notas asociadas y los temas previos con el objetivo de conseguir una pregunta suficientemente compleja pero enfocada en evaluar el tema definido.

- **Tipo de pregunta**: define el formato de evaluación deseado, guiando la forma que debe adoptar el enunciado y las opciones. Cada tema cuenta con un conjunto de tipos de pregunta compatibles, que pueden ser configurados por el docente. Al generar una pregunta, se selecciona uno de estos tipos de forma aleatoria y su descripción se incorpora en el *prompt*.

- **Formato esperado de salida**: se indica que la respuesta debe estructurarse como un diccionario en formato JSON, con campos bien definidos que permitan su posterior procesamiento de forma sencilla.

  Esta exigencia se refuerza gracias a que la API de Gemini permite definir el formato de salida del modelo mediante el parámetro *response_format*. Este parámetro especifica la estructura que debe tener la respuesta generada por el modelo. En nuestro caso, utilizamos el valor "{ type: :json_object }", lo que indica que el modelo debe devolver una respuesta estructurada como un objeto JSON y rechazar automáticamente aquellas respuestas que no lo cumplan.

Entre todos los datos enviados al modelo de lenguaje, el **Tema de la Pregunta** y el **Tipo de Pregunta** son los únicos que pueden variar entre invocaciones. Esto se debe a que representan aspectos intrínsecos de cada pregunta particular, en contraste con el resto de los datos que forman parte del contexto general y permanecen constantes.

Cabe destacar que, si bien estos datos fijos están pensados para el curso de *Programación 1*, la estructura modular actual permitiría su definición dinámica en el futuro para extender el sistema a otras materias.

Estrategia de acotación temática mediante *prompt* A diferencia de enfoques como *Retrieval-Augmented Generation* (RAG), que optimizan las respuestas del modelo accediendo a una base de conocimientos externa antes de generar la salida, nuestro sistema se basa únicamente en información previamente estructurada que se incorpora directamente al *prompt*. En particular, a la hora de crear una pregunta, se toma uno de los temas pertenecientes a las unidades que el estudiante seleccionó y su información es incorporada en un campo denominado **Tema de la Pregunta**.

Optamos por este enfoque por varias razones:

- **Conocimiento previo del modelo**: los LLMs ya cuentan con una base de conocimiento sólida, construida a partir de grandes volúmenes de datos públicos disponibles durante su entrenamiento. En particular, poseen nociones teóricas y prácticas sobre lenguajes de programación como Pascal, lo que permite generar preguntas sin necesidad de entrenar el modelo o "alimentarlo" con gran cantidad de documentación específica.

- **Simplicidad del sistema**: al no utilizar mecanismos de recuperación dinámica, se evita la necesidad de mantener una base de conocimientos externa, diseñar un sistema de embeddings o ajustar mecanismos de búsqueda. Sin embargo, sí es necesario garantizar que se cuenta con una

definición actualizada de cada tema para construir el prompt, ya que esta información actúa como contexto fijo durante la generación. Esto implica un esfuerzo de mantenimiento menor, pero no nulo.

- **Capacidad contextual de los LLMs**: si bien en nuestro caso no incluimos como *prompt* toda la documentación o material del curso, los modelos actuales permiten hacerlo (Codingscape, 2025). Por ejemplo, Gemini 2.5 cuenta con una ventana de contexto de hasta 1 millón de *tokens* (Kavukcuoglu, 2025), lo que permite prescindir de esquemas de recuperación dinámica como RAG pues el modelo puede razonar directamente sobre grandes conjuntos de información incluidos en el *prompt*.

Construcción del prompt

La interacción con el modelo se basa en diversas técnicas de ***Prompt Engineering***, mediante las cuales se busca estructurar cuidadosamente la entrada del modelo para lograr orientarlo de mejor forma y obtener respuestas alineadas.

Cada servicio, tanto el de generación de enunciado y clave como el de creación de distractores, implementa su propia estrategia. No obstante, antes de analizar en detalle las mismas, es importante describir ciertos mecanismos comunes que comparten ambos servicios al interactuar con el modelo.

Historial de mensajes La API de Gemini facilita la comunicación con el LLM a través del método ***chat***, que no solo permite enviar un mensaje actual, sino que también incluir un historial de mensajes previos. Esta capacidad resulta vital para mantener el contexto conversacional a lo largo de todo el proceso de generación de una pregunta.

Con este mecanismo, los dos servicios que intervienen en la construcción de la pregunta pueden compartir un mismo historial, facilitando que el modelo conserve memoria tanto sobre lo ya generado como sobre el contexto dado en previas instrucciones, y pueda así continuar con base en ello.

Roles de los mensajes La API permite también definir el rol de la entidad que está creando el mensaje. Se utilizan tres tipos:

- *System*: usualmente el mensaje inicial que define al modelo su rol y la manera por la cual este debe responder.
- *User*: indica que el mensaje es mandado por un usuario real y debe usarse para representar mensajes generados por usuarios.
- *Assistant*: indica que el mensaje es generado por el modelo y se utiliza para insertar mensajes del modelo en la conversación.

Utilizamos el rol *system* para mandar el siguiente mensaje: *"Eres un asistente para estudiantes de Ingeniería en Computación cursando Programación 1. Tu tarea es crear preguntas de múltiple opción de alta calidad que sirvan como*

herramienta de estudio y autoevaluación informal para que los alumnos refuercen su comprensión de los temas.”.

Para el resto de interacciones, se utilizo el rol *user* para indicar los mensajes enviados y assistant para guardar las respuestas a los mismos.

De esta manera, se construye una conversación progresiva con el LLM, donde cada mensaje se apoya en el contexto de los anteriores.

Auto-refiniamiento y Cadena de Razonamiento: estrategia de *prompting* para la generación del enunciado y respuesta correcta *QuestionGenerationService*, servicio que se encarga de la generación del enunciado y la respuesta correcta, utiliza con este objetivo una mezcla de dos técnicas: *Self-Refine* (Madaan y cols., 2023) y *Chain-of-Thought (CoT)* (Zhang, Zhang, Li, y Smola, 2022).

Self-Refine refiere a una estrategia basada en solicitar al modelo que analice y critique su propio razonamiento, permitiéndole realizar mejoras y correcciones en el proceso.

Chain-of-Thought (CoT) por su parte, es una técnica usada en modelos de lenguaje para mejorar su capacidad de resolver problemas complejos, que consiste en guiar al modelo a que produzca una serie de pasos intermedios antes de llegar a una respuesta final, simulando un proceso de razonamiento más similar al humano.

Inicialmente se exploró, con el objetivo de implementar el proceso de *Self-Refine*, la posibilidad de hacer uso de *Prompt Chaining*. Esta estrategia consiste en encadenar múltiples *prompts* de forma secuencial, donde la salida de un paso alimenta el siguiente. En este caso la *prompt* inicial, a partir de los datos anteriormente definidos, generaría un enunciado y su respuesta correcta. Luego, esa salida se utiliza como entrada para una segunda prompt, cuya función es analizar críticamente la pregunta generada y proponer mejoras para corregir y aumentar la complejidad del contenido.

Los resultados obtenidos al utilizar múltiples *prompts* para este objetivo no demostraron mejoras significativas en la calidad de la generación, además de incrementar considerablemente la complejidad del sistema. Por esta razón, se optó por implementar una estrategia de *Self-Refine* dentro de una única *prompt*, incorporando al final de la misma una guía específica para el procesamiento y análisis de la respuesta:

```
# **Proceso de Razonamiento**

Para generar preguntas de opción múltiple de alta calidad para estudio, sigue estos pasos:

1. **Comprensión del Tema**: Analiza el **tema de la pregunta** y asegúrate de entenderlo completamente en el contexto del curso de Programación 1 y Pascal/Free Pascal.
2. **Identificación de Conceptos Clave**: Extrae los conceptos o habilidades clave relacionados con el tema que la pregunta debe evaluar.
```

```
3.  **Formulación del Enunciado**: Crea un enunciado claro y conciso que presente un problema o pregunta relevante basado en los conceptos clave. Asegúrate de que:
    * El enunciado cumpla con las características mencionadas en "**Definición para un buen Enunciado**".
    * Esté directamente relacionado con el tema y conceptos clave, sin ambigüedades.
    * Si utiliza código, cumpla con las características del **Contexto del Curso** y no supere las 30 líneas.
    * Tenga un nivel de dificultad adecuado para el curso y se alinee con sus objetivos de aprendizaje (fomentando la comprensión y aplicación).
    * Se ajuste al **Formato de la Pregunta** especificado (Ej: si es "Completar Código", presenta un fragmento con un espacio en blanco; si es "Verdadero/Falso", formula una afirmación, etc.).
4.  **Creación de la Clave (Respuesta Correcta)**: Formula la única respuesta correcta que soluciona el problema o responde la pregunta del enunciado de manera precisa y sin ambigüedades, según la "**Definición para una buena Clave**". Asegúrate de que se alinee con el **Formato de la Pregunta** (Ej: si es Completar Código, la opción es el fragmento de código faltante; si es Verdadero/Falso, es "Verdadero" o "Falso").
5.  **Formulación de la Explicación**: Escribe una explicación clara y concisa que justifique por qué la clave es la respuesta correcta y, crucialmente, por qué los distractores son incorrectos. Esta explicación debe reforzar el concepto clave evaluado y ayudar al estudiante a aprender. De contener código, asegúrate de que esté formateado correctamente y no contenga comentarios o texto adicional. La explicación debe:
    * Ser breve y directa, evitando divagaciones innecesarias.
    * Incluir ejemplos concretos si es necesario para ilustrar el concepto.
    * No contener errores gramaticales o tipográficos.
    * Estar escrita en español, utilizando un lenguaje claro y accesible.
6.  **Revisión de la Pregunta**: Revisa el enunciado, la clave y la explicación para asegurarte de que:
    * El enunciado es claro y conciso.
    * La clave es la única respuesta correcta.
    * La explicación es precisa y relevante.
    * No hay errores gramaticales o tipográficos.
    * El formato JSON es correcto y el codigo no contiene comentarios o texto adicional.
7.  **Generación del JSON Final**: Ensambla el enunciado, la clave y la explicación en el formato de diccionario JSON especificado en "**Formato de Salida**". Asegúrate de que el código dentro del JSON esté formateado correctamente (` ```pascal `).
```

```
8. **Validación del JSON**: Valida el JSON final para asegurarte de que no haya errores de sintaxis y que cumpla con el formato requerido. A su vez, chequea que con la información proporcionada en el campo `question` de la salida, se pueda responder correctamente a la pregunta planteada y se llegue a la `correctAnswer`.
```

El formato de salida final solicitado al utilizar esta promt es en formato JSON y contiene el enunciado, la respuesta y la explicación generadas. La estructura es la siguiente:

```
{
  "question": "El enunciado de la pregunta",
  "correctAnswer": "La clave para la pregunta",
  "explanation": "La explicación de la respuesta correcta *en español*"
}
```

Cadena de Razonamiento y Prompt Chaining: estrategia para la generación de distractores Una vez finalizada la generación del enunciado y la respuesta correcta, el siguiente paso es la creación de distractores. El *DistractorsGenerationService* se encarga de esta tarea, utilizando nuevamente una estrategia basada en técnicas de *Chain-of-Thought (CoT)* (Zhang y cols., 2022) con distinto fin que en el paso anterior. Esta estrategia de *prompt engineering* guía al modelo a razonar de manera explícita y secuencial, generando pasos intermedios que llevan a una respuesta o decisión más elaborada.

En particular, nuestro servicio se desarrolla en cuatro etapas principales:

- **Generación de candidatos**: se solicita al modelo que proponga diez distractores plausibles.
- **Evaluación paso a paso**: aprovechando la técnica de Chain-of-Thought, se le pide al modelo que analice detalladamente cada distractor generado según tres criterios:
  - Plausibilidad: el distractor debe parecer una respuesta válida para alguien con una comprensión incompleta del tema.
  - Diferenciación: debe diferenciarse claramente de la respuesta correcta y de los otros distractores, evitando redundancias.
  - Longitud: debe tener una extensión comparable a la de la respuesta correcta y entre el resto de distractores.
- **Filtrado**: con base en la evaluación anterior, se descartan aquellos distractores que son claramente erróneos o son del estilo "Todas las anteriores" o "Ninguna de las anteriores".
- **Selección final**: el modelo elige los dos mejores distractores entre los restantes.

En este caso, debido a la clara modularidad de cada iteración y la posibilidad de modificar sus componentes de manera independiente, se decidió utilizar *Prompt Chaining* para dividir el proceso en múltiples prompts, en lugar de concentrar todos los pasos en una sola entrada.

Por otra parte, es importante mencionar que existen distintas estrategias a la hora de implementar un mecanismo de CoT. *Manual-CoT* y *Auto-CoT* (Zhang y cols., 2022) por ejemplo, incorporan demostraciones (ya sea creadas por humanos o generadas automáticamente) de cuál es el proceso de razonamiento esperado. *Zero-Shot-CoT* (Zhang y cols., 2022) por su parte, es una estrategia más simple donde se indica al modelo que debe seguir un razonamiento escalonado (utilizando un simple enunciado como "Piensa en esto paso a paso") para posteriormente extraer la respuesta. En nuestro caso, optamos por una estrategia más simple y generalizable, similar a *Zero-Shot-CoT* ya que se encuentra basada únicamente en instrucciones textuales, pero haciendo una guía más específica del razonamiento esperado, dividiendo al mismo en distintas iteraciones o *prompts*.

La figura 3.3 ilustra las distintas *prompts* utilizadas en este proceso, las cuales pueden ser encontradas en el anexo documento "Anexo: Prompt para la Generación de Distractores".

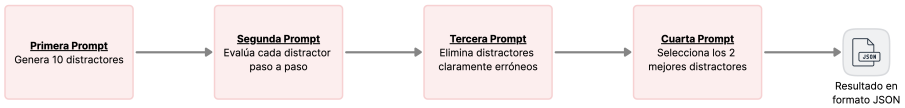

Figura 3.3: Diagrama estrategia *Prompt-Chaining* y *Chain-of-Thought* para generación de distractores

El resultado final de esta etapa se estructura en un objeto JSON con una única clave "distractors", cuyo valor es un arreglo de dos objetos. Cada objeto representa un distractor y contiene dos claves: "text" (el contenido del distractor) y "evaluation" (el análisis generado por el modelo). La estructura es la siguiente:

```
{
  "distractors": [
    {
      "text": "Distractor final 1",
      "evaluation": "Comentario sobre el distractor final 1"
    },
    {
      "text": "Distractor final 2",
      "evaluation": "Comentario sobre el distractor final 2"
    }
```

```
11    ]
12  }
```

Capítulo 4

Experimentación

4.1. Exploración inicial de estrategias con modelos de lenguaje

Antes de tener completamente definido el flujo del sistema y la forma en que interactuaríamos con el modelo de lenguaje de gran escala, realizamos una etapa de experimentación para explorar distintos enfoques y comprender mejor las capacidades y limitaciones de los mismos. En particular, buscamos entender cómo estructurar la interacción con el modelo para lograr una generación efectiva de preguntas, es decir, cómo introducir el contexto, los temas de cada unidad, el tipo de pregunta a generar, entre otros elementos de configuración. Además, también evaluamos técnicas de recuperación aumentada (RAG), con el objetivo de mejorar la relevancia y precisión del contenido generado a partir de documentos específicos.

Asimismo, buscábamos conocer mejor los modelos disponibles, evaluar sus requerimientos de cómputo para ejecutarlos localmente, y explorar las APIs existentes para integrarlos a nuestro sistema.

4.1.1. Pruebas con modelos autoalojados mediante Ollama

El primer enfoque consistió en ejecutar modelos de lenguaje en entorno local utilizando **Ollama**, una plataforma que permite la ejecución directa de estos modelos en computadoras personales (Tahir, 2025), simplificando considerablemente el proceso de puesta en marcha.

Se probaron los modelos **LLaMA 3.2** (Meta AI, 2024) y **Deepseek-R1** (DeepSeek AI, 2025), que ofrecen variantes con relativamente pocos parámetros (por ejemplo, 3B, 7B o 14B), lo que los hacía aptos para su ejecución en computadoras personales. Estos parámetros son los valores numéricos que el modelo ajusta durante su entrenamiento y que determinan su capacidad para representar patrones del lenguaje. Cuantos más parámetros tiene un modelo, mayor es

su capacidad para comprender y generar texto complejo, pero también mayores son sus requerimientos computacionales, especialmente en términos de memoria y procesamiento. Deepseek-R1, por ejemplo, también cuenta con versiones mucho más grandes, incluso de hasta 671 billones de parámetros, cuya ejecución local resulta inviable. El documento "Anexo: Decisiones Sobre el Modelo de Lenguaje" profundiza en los aspectos técnicos y los recursos computacionales necesarios para ejecutar modelos de lenguaje de gran escala.

Si bien Ollama facilita la instalación y ejecución, el rendimiento depende de las especificaciones de hardware de cada computadora personal. En la práctica, procurábamos realizar las pruebas más exigentes en el equipo con mayores recursos disponibles, lo cual nos permitió ejecutar modelos de hasta 32 billones de parámetros.

Estas pruebas nos brindaron un primer acercamiento al rendimiento de los LLMs en entornos locales, sin depender de servicios ni conectividad externa. Nos ayudaron a comprender sus capacidades reales bajo limitaciones de hardware, y a detectar fenómenos relevantes como las **alucinaciones**: situaciones en las que el modelo genera información falsa o imprecisa con aparente seguridad. Esto nos permitió tomar conciencia de la limitación de los enfoques puramente locales: la imposibilidad de acceder a modelos con mayores capacidades de razonamiento. A partir de esto, comprendimos la necesidad de explorar alternativas que nos permitieran utilizar esos tipos de modelos.

4.1.2. Enfoque de generación aumentada con recuperación de información

Exploramos también el uso de **RAG** (del inglés, *Retrieval-Augmented Generation*). Este enfoque consiste en combinar un modelo de lenguaje con un sistema de recuperación de información que le proporciona fragmentos relevantes antes de generar la respuesta (Amazon Web Services, s.f.). A través de esto, construimos un prototipo que nos permitía evaluar si un modelo podía generar preguntas de múltiple opción basándose en materiales del curso, como PDFs, apuntes o textos específicos.

Los resultados iniciales mostraron que la calidad de las preguntas generadas dependía fuertemente del material proporcionado al sistema. Si bien el modelo podía generar preguntas basadas en los textos de los PDFs o apuntes, esta dependencia provocaba que las preguntas fueran muy similares entre sí, repetitivas y con poca variación conceptual.

No obstante, luego de investigar más en profundidad, descubrimos que el enfoque *RAG* no era estrictamente necesario para nuestro caso, ya que los LLMs modernos (incluso sin *fine-tuning*) tienen un conocimiento general profundo sobre el dominio de la programación. Esto, sumado al hecho de que los modelos actuales soportan **ventanas de contexto muy amplias** (Vellum, 2025), nos llevó a optar por pasar la información directamente dentro del prompt, sin necesidad de implementar un enfoque *RAG*.

Por ejemplo, en los modelos Gemini, un *token* equivale a cuatro caracteres, y cien *tokens* equivalen a entre sesenta y ochenta palabras (en inglés) (Google AI

Modelo	Ventana de contexto (tokens)
LLaMA 4 Scout	10.000.000
LLaMA 4 Maverick	10.000.000
Gemini 2.5 Flash	1.000.000
Gemini 2.5 Pro	1.000.000
GPT-4.1	1.000.000
Claude 4 Opus	200.000
Claude 4 Sonnet	200.000

Tabla 4.1: Comparativa de ventanas de contexto por modelo

for Developers, s.f.). En base a esto y a la información de la tabla 4.1, la ventana de contexto de 1 millón de tokens equivale a aproximadamente 4 millones de caracteres o entre 600.000 y 800.000 palabras.

Esto implica que es posible incluir directamente en el *prompt* una cantidad considerable de contenido, como apuntes, textos completos de una unidad o incluso múltiples documentos, sin requerir un sistema adicional de recuperación. De este modo, aprovechamos al máximo el conocimiento interno del modelo y su capacidad para razonar sobre un contexto extenso, simplificando la arquitectura general del sistema.

4.1.3. Pruebas con APIs externas

Posteriormente avanzamos hacia la experimentación con APIs de terceros, principalmente de OpenAI y Google. En esta etapa probamos distintos modelos de la serie GPT de OpenAI, así como también los modelos *Gemini 2.5 Flash* y *Gemini 2.5 Pro* de Google. En particular, la variante *Flash* está orientada a tareas ligeras y respuestas rápidas, mientras que *Pro* fue diseñada para abordar tareas complejas con mayores capacidades de razonamiento (Kavukcuoglu, 2025). Estas diferencias en los modelos fueron comparadas de forma práctica, realizando pruebas con ambos para evaluar cómo impactaban sus características en la generación de preguntas.

Los modelos de OpenAI fueron utilizados en una primera instancia, donde construimos una versión preliminar de la interfaz que comunicaría nuestro servidor de Ruby on Rails con los modelos GPT. Compramos créditos por un valor de USD 5 para realizar pruebas iniciales. Una vez que la interfaz estuvo operativa y validamos su funcionamiento, incorporamos la posibilidad de variar el modelo utilizado, lo que nos permitió extender las pruebas a los modelos *Gemini 2.5 Flash* y *Gemini 2.5 Pro* mencionados anteriormente.

Posteriormente, el enfoque se centró en la **experimentación iterativa de *prompts***. Tras cada diseño, se evaluaban los resultados obtenidos, se introducían ajustes pertinentes y se repetía el proceso. Este enfoque cíclico nos permitió refinar los resultados y entender mejor qué tipo de instrucciones producían mejores preguntas. Además, como parte de este proceso, el sistema que construi-

mos nos permitía modificar fácilmente el modelo utilizado, lo cual introducía una nueva variable en el ciclo de refinamiento. Esta flexibilidad nos permitió ajustar los resultados no solo a partir de los *prompts*, sino también eligiendo el modelo que mejor se adaptara al objetivo.

Más adelante, como parte de esta misma exploración, nos enfocamos específicamente en estudiar estrategias de prompting propuestas en papers académicos sobre generación automática de preguntas múltiple opción con LLMs. Esta revisión nos llevó a testear enfoques como: *Chain of Thought*, *Prompt Chaining*, *Self-Refine*, entre otros.

Muchas de estas estrategias fueron posteriormente incorporadas en la versión final del *prompt* de nuestro sistema. El capítulo 3.4 profundiza en el análisis de estos enfoques, junto con la explicación de los fundamentos teóricos detrás de cada estrategia, referencias a los artículos consultados y citas relevantes que guiaron su implementación.

4.2. Evaluación de correctitud y calidad de las preguntas generadas

Como parte de la etapa de experimentación y, en particular, del mencionado proceso de refinamiento iterativo de *prompts*, incorporamos una instancia de evaluación sistemática de la **correctitud** y **calidad** de las preguntas generadas por el modelo. Esta evaluación tiene como objetivo identificar errores en las preguntas generadas, detectar oportunidades de mejora y orientar ajustes en el diseño de *prompts*.

4.2.1. Correctitud

Para los fines de esta evaluación, entendemos la correctitud como un criterio objetivo, enfocado en la ausencia de errores e inconsistencias lógicas, conceptuales o técnicas que invaliden la pregunta. Algunos de los aspectos analizados son:

- **Fragmento de código válido y coherente:** en aquellas preguntas que incluyen código, se analiza que el mismo sea sintácticamente correcto y que compile exitosamente.
- **Respuesta correcta verificada:** se valida que la opción marcada como correcta sea efectivamente la única respuesta correcta, ya sea mediante ejecución del código (cuando corresponde) o razonamiento conceptual.
- **Distractores no ambiguos:** se revisa que las opciones incorrectas no se solapen entre sí ni con la correcta.
- **Claridad en el enunciado:** se comprueba que el planteo esté libre de errores gramaticales o ambigüedades, y que la consigna esté formulada de forma precisa.

4.2.2. Calidad

A diferencia de la correctitud que se enfoca en la validez técnica, en el marco de nuestro análisis, la calidad apunta a la capacidad que tiene una pregunta para evaluar de forma efectiva la comprensión del estudiante, dentro del marco del tema correspondiente. Algunos de los aspectos que analizamos incluyen:

- **Nivel de dificultad apropiado**: buscamos que la pregunta no sea ni demasiado trivial ni excesivamente compleja, sino que represente un desafío razonable acorde al contenido a evaluar.
- **Pertinencia temática**: verificamos que la pregunta esté alineada con el tema seleccionado, evitando que se introduzcan conceptos no vistos.
- **Buenas prácticas de preguntas múltiple opción**: como se menciona en la sección 2.3, buscamos que la pregunta generada cumpla con las características que constituyen una pregunta múltiple opción bien formulada.

4.2.3. Proceso manual de evaluación y ajustes posteriores

Existen formas sistemáticas y automatizadas de evaluar la calidad de las preguntas, como el cálculo del **Índice de discriminación** o el **Índice de dificultad** (University of Washington, s.f.), que se basan en el análisis de respuestas de los estudiantes una vez que interactúan con las preguntas dentro de la plataforma. Sin embargo, estas métricas requieren datos empíricos provenientes del uso real del sistema y, por lo tanto, no pueden ser aplicadas en esta etapa de experimentación. Estos mecanismos y su utilidad serán considerados como parte de los análisis posteriores al despliegue de la plataforma y se describen en detalle en la sección 5.1.7.

Asimismo, no encontramos en nuestra investigación herramientas, algoritmos o librerías que permitan evaluar automáticamente la calidad de una pregunta de múltiple opción tomando únicamente como entrada su enunciado y opciones. De existir este tipo de herramienta, podría integrarse directamente en las etapas de desarrollo para asistir en la validación automatizada de las preguntas generadas.

En cuanto a la correctitud, sí se consideró la posibilidad de desarrollar un sistema que compile automáticamente el código incluido en la pregunta y compare su salida con la opción marcada como correcta. No obstante, esta estrategia solo es relevante para asignaturas relacionadas con la programación, donde el contenido puede ser compilado y ejecutado. Además, este enfoque sería aplicable únicamente a un subconjunto específico de preguntas: aquellas que contienen un fragmento de código en su enunciado y cuya consigna sea del tipo *"correct output"*. En preguntas conceptuales u orientadas a razonamiento teórico, donde las opciones no reflejan directamente la ejecución del código, esta estrategia no sería válida. Esta posibilidad también es analizada en la sección de trabajo a futuro 5.1.7.

Por este conjunto de razones es que la evaluación se realiza de forma manual y se incorpora como un paso adicional en el ciclo iterativo de mejora. Cada vez

que se modifica el *prompt*, se generan múltiples preguntas nuevas y se analizan sus niveles de correctitud y calidad según los criterios mencionados. De esta forma se busca observar el impacto de los cambios e identificar patrones de error.

Cabe aclarar que, al tratarse de un proceso manual y no sistematizado, no se puede garantizar completamente la objetividad ni la consistencia de las evaluaciones. Aun así, este enfoque exploratorio resulta útil para detectar tendencias generales y tomar decisiones informadas durante el proceso de desarrollo.

El proceso nos permitió, por ejemplo, identificar que el fragmento de código ocasionalmente contenía demasiados comentarios explicativos que facilitaban en exceso la comprensión de la pregunta. En respuesta, agregamos al *prompt* una instrucción explícita para evitar comentarios innecesarios o demasiado descriptivos.

También detectamos que algunas preguntas contenían fragmentos de código muy fáciles de comprender y donde la respuesta correcta se hacía evidente. Para abordar esto, aplicamos estrategias de *prompting* como *Chain of Thought* y *Self-Refine*, que fueron explicadas en detalle en la sección 3.4, y ayudan al modelo a separar el razonamiento del resultado final. Además, profundizamos las descripciones temáticas que recibe el modelo para favorecer la generación de preguntas más sofisticadas y alineadas al contenido, incluyendo asimismo breves descripciones de los temas previos del curso, lo que instruye al modelo a combinar conceptos y generar preguntas más integradoras.

En ciertos casos, el modelo generaba preguntas con fallas al trabajar con funciones como *ord()*, produciendo conversiones incorrectas de caracteres a enteros. Para mitigar este problema, incorporamos directamente en el *prompt* una tabla ASCII básica, con el fin de reducir ambigüedades en este tipo de conversiones.

Estos ejemplos reflejan cómo la evaluación manual, aunque limitada, ha sido fundamental para detectar limitaciones del modelo y guiar mejoras efectivas en la formulación de los *prompts*. A futuro, este proceso de detección de errores se verá complementado con la funcionalidad en la plataforma que permite a los propios estudiantes reportar preguntas, facilitando un ciclo de mejora continua.

4.3. Análisis Final de Correctitud y Calidad de las Preguntas Generadas

Una vez concluido el proceso iterativo de ajuste y refinamiento de la generación, se llevó a cabo una instancia final de revisión centrada en las preguntas generadas con la versión definitiva del sistema. Esta incorpora toda la información disponible para cada tema, así como la iteración más reciente de las *prompts* desarrolladas a lo largo del proyecto.

El análisis contempla tanto la perspectiva del equipo de desarrolladores, basada en los conocimientos adquiridos durante el proceso de investigación, como la del equipo docente de *Programación 1*, que aporta una mirada experta desde el enfoque pedagógico y disciplinar, gracias a su experiencia en los contenidos

del curso y en la formulación de este tipo de preguntas.

Es importante destacar que, si bien este proceso tiene como base las preguntas generadas para el curso *Programación 1*, se busca identificar fortalezas y puntos de mejora generales del sistema independientes del contenido del curso utilizado.

4.3.1. Fortalezas

A partir del análisis conjunto del equipo de desarrollo y del equipo docente, se identificaron una serie de fortalezas que reflejan tanto la solidez de las estrategias implementadas como el potencial del sistema para generar preguntas múltiple opción de calidad.

A continuación se presentan algunos de los puntos que más destacan del sistema de generación desarrollado:

Correctitud y Precisión Técnica

A la hora de evaluar una pregunta, un aspecto fundamental para garantizar su utilidad pedagógica es evitar errores que puedan inducir al estudiante a construir conocimientos incorrectos o generar confusión.

Durante el proceso de iteración, se identificaron múltiples casos en los que, al generar una pregunta y su respectiva respuesta, se detectaban errores conceptuales o problemas de compilación, especialmente en ejercicios que incluían fragmentos de código. Con el objetivo de reducir este tipo de errores, se incorporó a la *prompt* información explícita sobre reglas sintácticas del lenguaje Pascal, así como la inclusión de notas orientativas por tema. Estas notas permiten destacar elementos clave a considerar al generar preguntas sobre determinados temas, mejorando así la precisión y coherencia de los ítems generados. Gracias a estas incorporaciones fue posible resolver, por ejemplo, la generación de código sin punto final, lo cual impedía su compilación, así como la interpretación incorrecta de funciones como *ord([LETRA])*. En este último caso, se añadió una tabla ASCII en las notas asociadas a los temas correspondientes, lo que permitió al sistema generar preguntas más precisas y correctas.

Como resultado de estas mejoras, se obtuvo un sistema robusto en el que no se detectaron errores significativos de correctitud, ni por parte del equipo de desarrollo ni del equipo docente. El código generado es sintácticamente correcto, compilable y no presenta respuestas incorrectas. Además, y quizás lo más relevante, se lograron identificar estrategias simples y efectivas para abordar posibles problemas en caso de que surjan cambios de contexto, ya sea mediante la incorporación de nuevos temas o su adaptación a otros cursos.

Claridad y Redacción

Así como contar con preguntas correctas es un aspecto fundamental para ofrecer una herramienta efectiva de autoevaluación y estudio, también lo es que dichas preguntas estén bien redactadas y sean claras. La ambigüedad, las

confusiones o las interpretaciones erróneas pueden comprometer el proceso de aprendizaje y reducir la utilidad pedagógica del sistema. Con este objetivo en mente, el proceso de construcción de *prompts* incluyó, entre otros datos, la definición de atributos clave que una pregunta debe cumplir para considerarse clara.

El resultado de estos esfuerzos es la generación de preguntas que no dan lugar a dobles interpretaciones. En los casos en que se incluye una negación, esta se resalta explícitamente para evitar confusiones, como se observa en la figura 4.1. Además, se evita la inclusión de elementos innecesarios que puedan entorpecer el proceso de razonamiento, tanto en la redacción como en el código presentado.

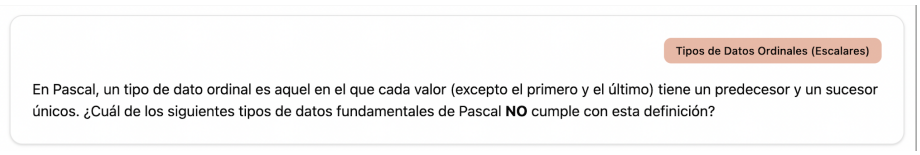

Figura 4.1: Ejemplo de pregunta correctamente negada

Creatividad y Variedad de Enfoques

Un elemento importante al momento de completar múltiples cuestionarios es la presencia de enfoques variados y preguntas creativas que no solo desafíen al estudiante, sino que también lo motiven a continuar con el proceso de aprendizaje.

En este sentido, se destaca la capacidad del sistema para generar preguntas con un alto grado de creatividad. En lugar de limitarse a reproducir los ejemplos incluidos en la definición del tema, el sistema propone distintos enfoques y, en algunos casos, aplica conceptos abstractos a situaciones reales, lo que enriquece la experiencia de estudio y motiva al estudiante que ve sus conocimientos aplicados a un problema real. A continuación se presenta una sección de código generada como ejemplo de esto, donde la pregunta generada simula ser parte de un videojuego:

```
1  program Selector;
2  var
3    puntuacion: integer;
4    activo: boolean;
5  begin
6    puntuacion := 75;
7    activo := false;
8
9    if puntuacion > 50 then
10     if activo then
```

```
      writeln('Nivel superado')
    else
      writeln('Jugador inactivo');
end.
```

Si bien algunos temas específicos presentaron cierta repetitividad al generar múltiples cuestionarios con un mismo tipo de pregunta, esta situación mejoró considerablemente al incorporar otros tipos de pregunta a la generación. Aunque se trataba del mismo contenido, las nuevas preguntas requerían diferentes formas de razonamiento y apuntaban a otros niveles de comprensión.

La incorporación de temas previos, por su parte, también contribuyó a aportar mayor complejidad y diversidad a las preguntas generadas. Si bien no desvía el foco del tema central, permite al modelo hacer uso de conceptos ya abordados, lo que, en el caso de contenidos más simples, amplía considerablemente las posibilidades a la hora de generar ejercicios.

Calidad General

En términos generales, los ejercicios generados fueron clasificados tanto por el equipo de desarrollo como por el docente como buenos ejercicios.

Entre los aspectos destacados que respaldan esta valoración, se encuentra particularmente la calidad de las explicaciones generadas. Estas presentan una estructura clara y completa; durante el uso del sistema, resultaron fundamentales para generar una sensación de aprendizaje. En muchos casos, permitieron comprender los conceptos sin necesidad de recurrir al material teórico.

A modo ilustrativo, se muestra a continuación una explicación generada por el sistema correspondiente a una respuesta correcta:

```
1. **Cálculo de la intersección `(set1 * set3)`**: La intersección ()
↪  devuelve los elementos comunes a ambos conjuntos.
   - `set1` es `[1, 3, 5, 7, 9]`
   - `set3` es `[0, 1, 2, 3]`
   - Los elementos comunes son `1` y `3`. El resultado de esta
   ↪  operación es el conjunto `[1, 3]`.
2. **Cálculo de la diferencia `(set2 - set3)`**: La diferencia ()
↪  devuelve los elementos del primer conjunto que no están en el
↪  segundo.
   - `set2` es `[0, 2, 4, 6, 8]`
   - `set3` es `[0, 1, 2, 3]`
   - Los elementos en `set2` que no están
   ↪  en `set3` son `4`, `6` y `8`. El resultado es el conjunto `[4,
   ↪  6, 8]`.
3. **Cálculo de la unión `(...) + (...)` y la salida final**: La unión
↪  (`+`) combina los resultados de las operaciones anteriores.
   - `[1, 3] + [4, 6, 8]` da como resultado el conjunto `[1, 3, 4, 6,
   ↪  8]`.
```

```
11    - El bucle `for` itera de 0 a 9 y utiliza el operador `in` para
      ↪  verificar si cada número pertenece al conjunto `resultado`. Si
      ↪  pertenece, lo imprime seguido de un espacio.
12    - Por lo tanto, la salida final es `1 3 4 6 8` .
```

Sin embargo, el análisis docente reveló que, en algunos ejercicios, las explicaciones no se alineaban del todo con el razonamiento que se buscaba fomentar. Este aspecto representa una oportunidad de mejora luego tratada, especialmente en términos de lograr un mayor control sobre la forma en que se justifican las respuestas.

Por último, un punto clave a destacar es que, tanto en la redacción de las preguntas como en sus opciones y en las explicaciones correspondientes, resulta difícil percibir que fueron generadas por un modelo de lenguaje sin intervención humana, siempre que no presenten errores notorios en su construcción.

Todos estos elementos, considerados en conjunto, convierten al sistema en una herramienta valiosa para el repaso y el aprendizaje. Las explicaciones guían de forma efectiva el proceso de comprensión, y las preguntas bien formuladas se asemejan a aquellas elaboradas por docentes. En este sentido, un estudiante de grado puede tener una experiencia de estudio enriquecedora y adecuada para el curso de Programación 1, aunque el sistema es lo suficientemente flexible como para proyectar resultados igual de prometedores en cursos similares.

4.3.2. Debilidades

Si bien el análisis final, que combina nuestras conclusiones con las del equipo docente de Programación 1, evidencia un desempeño general sólido por parte del sistema, también identificamos ciertas limitaciones y áreas con potencial de mejora.

A continuación se presentan los principales hallazgos que representan debilidades o problemas del sistema generativo:

Fallas esporádicas en la estructuración del enunciado

Uno de los problemas identificados es que, en un pequeño número de casos, el modelo genera preguntas cuyo enunciado hace referencia a un fragmento de código que no está presente. Esta omisión, que solo afecta a los tipos de pregunta que requieren incluir código en el enunciado, invalida la consigna planteada y dificulta completamente su resolución. Si bien el problema se observó con mayor frecuencia en temas más complejos del curso de Programación 1, es razonable suponer que podría presentarse también en otras materias que aborden contenidos técnicos de similar complejidad, especialmente aquellas que requieran representar elementos no textuales como fragmentos de programa.

Por otra parte, también se detectaron casos esporádicos en los que el modelo incluye, dentro del campo correspondiente al enunciado, no solo la consigna sino también las opciones de respuesta. Esto contradice la arquitectura definida para la generación, que establece dos etapas diferenciadas: primero, la creación del

enunciado junto con la opción correcta; luego, la incorporación de los distractores. Además, va en contra del formato JSON previsto, que exige separar el enunciado y las opciones en campos distintos. Como resultado, se obtiene una salida redundante y confusa, en la que las opciones aparecen dos veces: una embebida dentro del valor de la clave *"question"*, y otra bajo la clave correspondiente a las opciones.

Simplicidad ocasional y patrones repetitivos

En algunos casos observamos que la dificultad de las preguntas generadas resulta relativamente baja. Si bien esto es esperable en temas más simples e introductorios, también se identificaron algunas situaciones en las que, incluso ante contenidos más complejos, el enunciado guía sutilmente al estudiante o incluye pistas que facilitan la resolución. Esta situación también fue señalada por el equipo docente, aunque no como un aspecto necesariamente negativo, ya que consideran valioso contar con algunas preguntas simples.

Asociado a esto, al analizar un conjunto amplio de preguntas generadas para un mismo tema, se observa una tendencia del sistema a repetir ciertos patrones o enfoques para evaluar un concepto. Es decir, el modelo tiende a recurrir reiteradamente a ciertas formulaciones que resultan efectivas, en lugar de explorar distintas formas de abordar el mismo contenido.

Por otro lado, también detectamos que algunos distractores se vuelven fácilmente descartables tras cierto uso continuado del sistema. Opciones como "El programa tiene un error de compilación" o "El programa entra en bucle y falla" rara vez aparecen como la respuesta correcta. Aunque esta situación se detectó en el contexto específico del curso Programación 1, podría replicarse en otras materias donde se utilicen distractores de estructura similar.

Esta última situación, que reduce progresivamente la efectividad de ciertos distractores, sugiere que para algunos temas y tipos de pregunta, podría ser beneficioso incorporar en la *prompt* de generación, una instrucción explícita para evitar estas opciones fácilmente descartables.

Desajustes en el enfoque del curso

El equipo docente también reveló que algunas preguntas generadas, si bien son técnicamente correctas, no se alinean con las convenciones o el enfoque pedagógico específico del curso.

Por ejemplo, señalaron casos puntuales donde se generaron preguntas que utilizan la función *ord()* de Pascal asumiendo que el estudiante memorizó la tabla ASCII, preguntas que modifican las variables de control dentro de un bucle *for*, o preguntas que proponen métodos de resolución que no se enseñan explícitamente. Algunas de estas "reglas no escritas" no están detalladas en el material teórico del curso, por lo que el responsable de este sería el encargado de explicitarlas.

Esta desconexión también se reflejó en algunas explicaciones asociadas a las respuestas, que en ocasiones seguían pasos de razonamiento poco pertinentes o

distintos a los que fomentan los docentes. Se plantea como trabajo futuro en 5.1.5 la posibilidad de definir de forma más estricta y explícita los objetivos de aprendizaje asociados a cada tema evaluado. Esto permitiría al sistema alinearse mejor con las intenciones pedagógicas del curso y evitar casos en los que se generen preguntas que, si bien son técnicamente válidas y utilizan herramientas vistas en clase, abordan aspectos que no forman parte de los aprendizajes que los docentes esperan promover.

Capítulo 5

Conclusiones y Trabajo Futuro

5.1. Trabajo Futuro

Esta sección presenta los elementos que, durante el proceso de investigación y desarrollo, identificamos como oportunidades para trabajos futuros. Incluyen tanto posibles mejoras al sistema actual como ideas que van más allá de la herramienta implementada.

Si bien todas despertaron nuestro interés, no fueron llevadas a cabo por exceder el alcance del proyecto o requerir más tiempo del disponible.

5.1.1. Ideas descartadas

Si bien en la sección 2.2 se justifica la elección de FinQuiz como aplicación a desarrollar, también se consideraron propuestas alternativas que fueron descartadas por diferentes motivos.

Luego de haber trabajado en este proyecto y haber profundizado en el área de la Inteligencia Artificial aplicada a la educación, alguna de estas ideas destacan como posibles soluciones que pueden ser de gran valor tanto para estudiantes como profesores.

Posibles mejoras a la plataforma OpenFing Se plantearon dos ideas que podrían aumentar, utilizando Inteligencia Artificial, las capacidades de la plataforma OpenFing, en particular: Indizar los videos de las clases de forma automática y generar evaluaciones automatizadas en base a los contenidos de las clases.

El proceso de indexación automática de los videos de OpenFing implica reconocer los temas tratados y las diferentes secciones de una clase grabada, para darle al estudiante más facilidad a la hora de navegar entre los temas dentro un

curso y sus videos. Esta idea resulta interesante y mejoraría, a nuestro parecer, de gran manera la experiencia de los estudiantes al utilizar OpenFing.

Por otro lado, la generación de evaluaciones automatizadas en base al contenido de una clase puede tener un gran impacto sobre la utilización de la plataforma. Para los estudiantes puede servir como una forma de afianzar los conocimientos e identificar debilidades al terminar una clase, mientras que para los profesores ofrece una forma de descubrir falencias en las explicaciones o identificar qué forma de presentar cierto contenido tiene mejores resultados. Cabe destacar que esta herramienta atacaría alguna de las mismas necesidades que resuelve FinQuiz, por lo que este podría servir como ayuda para una posible investigación e implementación.

La mayor dificultad técnica de estas ideas es la necesidad de procesar no solo el audio y la imágen de los videos, sino que el procesamiento debe poder combinar estas dos fuentes de información para lograr capturar todo el contenido de las clases. Es interesante mencionar que una vez resuelto el desafío de la obtención de los contenidos de una clase de OpenFing, FinQuiz puede llegar a ser de gran utilidad a la hora de la generación automática de evaluaciones.

En particular, en el marco del desarrollo de este proyecto se hicieron algunas pruebas con respecto al proceso de extracción de la información de las clases de OpenFing. Para esto se utilizaron herramientas de Inteligencia Artificial para generar transcripciones de las mismas, proceso que es discutido en mas detalle en el documento "Anexo: Transcripción de OpenFing".

Aplicaciones de modelado predictivo También, se descartaron varias ideas en torno al modelado predictivo de la trayectoria de los estudiantes. La que resalta como un posible trabajo a futuro es la recomendación de materias en base a la escolaridad.

Esta es una herramienta que busca darle a los estudiantes la capacidad de tomar decisiones de manera informada sobre su camino académico, así poder aumentar su rendimiento basándose en los conocimientos que hayan adquirido hasta el momento.

El entrenamiento de un modelo de este tipo necesita de grandes volúmenes de datos críticos y personales de los estudiantes, a los cuales no es posible acceder fácilmente. Sin embargo, es un instrumento que puede ser de gran ayuda para todas las partes involucradas en el proceso educativo y optimizaría el proceso de acompañamiento de los estudiantes.

Análisis de calidad de evaluaciones Luego de haber profundizado en conceptos de diseño de evaluaciones durante la implementación de FinQuiz, parece valioso el desarrollo de un sistema que permita determinar la calidad de una evaluación propuesta.

Una aplicación de este estilo podría ayudar en el proceso de generación de pruebas y su estandrización, dando a los profesores la capacidad de iterar sobre evaluaciones antes de implementarlas, recibiendo correcciones y así evitando problemas a futuro.

Cabe destacar que esta idea de implementación no enfoca al estudiante y a su proceso de estudio, sino que se enfoca en darle a los profesores herramientas para mejorar el proceso de generación de evaluaciones. Con esta herramienta se busca facilitar este proceso que puede resultar costoso a nivel de tiempo y recursos.

5.1.2. Experimentación con nuevas formas de interacción con modelos de lenguaje

Una línea de trabajo futuro consiste en explorar enfoques alternativos de interacción con modelos de lenguaje, tales como Generación Aumentada con Recuperación (RAG), *Fine-Tuning* y el uso de *Modelos de Lenguaje Pequeños (SLMs)* (Johnson, 2025). Estos métodos no reemplazan la arquitectura general del sistema, sino que podrían integrarse a ella para optimizar la generación de contenido, abordando desafíos como la especificidad temática, la latencia y el uso eficiente de recursos.

La implementación con RAG podría refinar cómo se utiliza el material del curso. Con este enfoque, el sistema podría funcionar en dos pasos: primero, un módulo de búsqueda analizaría la consulta, por ejemplo, una solicitud para generar una pregunta múltiple opción sobre un tema determinado, y buscaría en una base de conocimiento, como el libro del curso, los fragmentos más relevantes y los recuperaría. Luego, un módulo generador recibiría estos fragmentos junto con la consulta original para construir la prompt y consultarle al modelo de lenguaje. Este enfoque podría mejorar la eficiencia y ahorrar recursos al utilizar contextos más precisos, en lugar de descripciones de temas extensas.

Otro enfoque posible es especializar al modelo de lenguaje mediante estrategias de *fine-tuning*. El objetivo sería entrenar al modelo para que domine no solo el contenido, sino también el formato y la estructura específica de una pregunta de múltiple opción.

Se podrían explorar dos niveles de especialización:

- *Fine-tuning* para la tarea de generación de MCQs: entrenar un modelo con un dataset diverso de preguntas de múltiple opción de diferentes dominios. Esto lo convertiría en un generador "experto" de MCQs, independientemente de la materia.

- *Fine-tuning* por materia: se podría entrenar un modelo distinto para cada materia. Un modelo afinado con datos de Programación 1, por ejemplo, no solo sería bueno generando MCQs, sino que también asimilaría el estilo, los formatos y los errores conceptuales comunes de esa asignatura.

El principal desafío de esta técnica sigue siendo la necesidad de un dataset de entrenamiento grande y de alta calidad.

Finalmente, la aplicación de enfoques como RAG o *fine-tuning* podrían facilitar el camino para reemplazar el modelo de lenguaje actual por un Modelo de Lenguaje Pequeño (SLM). Al especializar el modelo o proporcionarle contextos más precisos, se reduce la necesidad de contar con un modelo de gran

escala para obtener resultados de calidad. Los SLMs, por ser más livianos y eficientes, permiten una generación más rápida y con menor costo computacional. Esto contribuiría directamente a mejorar uno de los principales puntos débiles identificados en el sistema: la demora en la generación de preguntas.

5.1.3. Recomendación Inteligente de Fragmentos de Clase

Un elemento clave de FinQuiz es su capacidad para fomentar la mejora continua del estudiante, brindando herramientas de repaso focalizado ante aquellos temas que se evidencian como menos afianzados a partir de los resultados obtenidos en los cuestionarios.

Actualmente, esto se materializa en la identificación de temas menos consolidados a partir de los cuestionarios resueltos, seguida de la recomendación de material relacionado. Entre los recursos sugeridos se incluyen artículos, textos y clases grabadas disponibles en la plataforma OpenFing. Sin embargo, en muchos casos, una clase puede durar una o varias horas y contener varios conceptos, lo que dificulta encontrar con precisión el tema que se desea repasar.

Para dar solución a este problema, se plantea la posibilidad de incorporar un sistema de procesamiento de las clases, utilizando mecanismos similares a los ya experimentados y detallados en el documento "Anexo: Transcripción de OpenFing". Dicho procesamiento permitiría obtener la transcripción en texto de cada clase y, potencialmente, identificar en qué secciones se aborda cada concepto o tema. De este modo, sería posible analizar los videos asociados a un determinado tema, desglosarlos y ofrecer una recomendación más precisa y personalizada sobre qué fragmento visualizar para el repaso.

5.1.4. Generación automática de temas

Nuestro sistema requiere la existencia de una descripción detallada de los temas que componen cada unidad a la hora de generar preguntas. Estos temas son utilizados como insumo del modelo de lenguaje, ya que su descripción es enviada en cada *prompt* al generar un *stem* y respuesta correcta.

Actualmente el sistema cuenta con un proceso manual para la carga de estos temas, por el cual el profesor responsable del curso es capaz de generar la cantidad de temas necesarios para cada unidad. Sin embargo, el uso de la Inteligencia Artificial puede extenderse para facilitar esta tediosa tarea. Esta idea se encontraba dentro de nuestras intenciones iniciales al plantear la implementación, pero no pudo llegar a verse plasmada en el resultado final, por cuestiones de tiempos.

La incorporación de un mecanismo automatizado capaz de agilizar esta tarea en la aplicación tiene como objetivo ofrecer al responsable del curso una herramienta para asociar material didáctico al momento de cargar una unidad. Para ello, podrá adjuntar un archivo o indicar una fuente vinculada. Dicho material será procesado de manera automatizada, utilizando un modelo de lenguaje que analiza el contenido, identifica y extrae los temas clave. Una vez finalizado el

proceso, el material quedará asociado a los temas identificados, lo que permitirá realizar recomendaciones futuras sobre dónde estudiarlos.

A pesar de no poder ejecutar originalmente la idea, el uso de un modelo de lenguaje para esta tarea fue evaluado de forma independiente al sistema durante el transcurso de este proyecto. Los resultados obtenidos, que pueden consultarse en detalle en el documento "Anexo: Reconocimiento de Temas en Base a Material", fueron satisfactorios y se utilizaron durante el proceso de desarrollo. Su aplicación permitió alimentar al sistema con un corpus de temas bien definidos, lo que facilitó la generación de preguntas de calidad.

A partir de los resultados obtenidos en esta experimentación, se puede esperar un desempeño favorable de esta adición. No obstante, consideramos que la supervisión de un docente o de alguien con conocimiento profundo del curso sigue siendo fundamental para garantizar una correcta segmentación de los temas y evitar posibles errores.

5.1.5. Incorporación de objetivos de aprendizaje en los temas de un curso

En el análisis de calidad realizado, y en particular a partir del feedback brindado por el equipo docente, un aspecto especialmente relevante que surgió es la desconexión de algunas preguntas generadas con el enfoque pedagógico esperado. Si bien estas preguntas eran técnicamente correctas y se apoyaban en contenidos vistos en clase, no necesariamente promovían los aprendizajes que el equipo docente busca fomentar.

En este sentido, se plantea como línea de trabajo futuro la incorporación de los objetivos de aprendizaje en las *prompts* utilizadas para interactuar con el modelo de lenguaje. El propósito de esta integración es que tanto las preguntas como sus explicaciones se alineen de forma más precisa con las intenciones pedagógicas del curso.

Cabe destacar que estos objetivos de aprendizaje suelen corresponder a criterios pedagógicos que no siempre se encuentran documentados, sino que forman parte del conocimiento del equipo docente. Esto implica la necesidad de desarrollar un mecanismo que permita a los responsables del curso definirlos manualmente dentro del sistema.

La idea de incorporar objetivos de aprendizaje en el proceso de generación de preguntas no es novedosa, y ha sido explorada en investigaciones previas sobre generación automática de preguntas. Por ejemplo, en el trabajo de Doughty et al. (Doughty y cols., 2023), se integran estos objetivos como una etapa clave dentro del pipeline de generación.

5.1.6. Expansión a múltiples materias

Si bien inicialmente el sistema está desarrollado poniendo el foco en el curso de *Programación 1*, su arquitectura fue diseñada desde el inicio con la escalabilidad en mente como se refleja en la sección 3.1, contemplando su aplicación en

diversas materias dentro de una misma carrera, en múltiples carreras o incluso en contextos educativos por fuera del ámbito universitario.

El sistema ya cuenta con una relación N:N entre usuarios y cursos, lo que permite tanto a estudiantes como a docentes estar asociados a múltiples materias. Resta implementar sin embargo un mecanismo que permita al usuario seleccionar activamente en qué curso desea trabajar. A partir de esa elección, la aplicación mantendría su funcionamiento habitual, cargando únicamente las unidades, temas y cuestionarios correspondientes al curso seleccionado.

Gracias a la estructura jerárquica ya existente de curso, unidad, tema y cuestionario, la adaptación visual y funcional sería mínima. Cada curso tendría su propio conjunto de unidades, temas, cuestionarios y docentes encargados de su administración.

En cuanto al contenido en sí mismo, esta expansión se complementa de manera natural con el trabajo futuro propuesto en la sección 5.1.4, ya que los profesores podrían cargar materiales específicos de su materia (como apuntes o bibliografía), que serían analizados automáticamente para identificar y estructurar los temas clave del curso.

Otro cambio necesario es la adaptación del *prompt* utilizado para interactuar con el modelo de lenguaje al contexto de cada materia. Si bien el sistema ya permite variar dinámicamente los temas incluidos en el *prompt*, actualmente se utiliza una introducción fija basada en la descripción del curso de Programación 1. Además, a lo largo del *prompt* se incluyen detalles específicos de esta materia, como restricciones sobre la longitud máxima de los fragmentos de código tanto en el enunciado como en las opciones, indicaciones sobre cómo debe marcarse correctamente el código en el *output* (por ejemplo, respetando ciertas convenciones de formato o estructura JSON), y la forma en que debe estructurarse una pregunta de opción múltiple en este contexto. Estos aspectos, si bien son adecuados para una materia de programación, no serían aplicables en otras disciplinas. Por lo tanto, sería necesario implementar una lógica que permita construir *prompts* dinámicos según el curso seleccionado, adaptando no solo la descripción inicial, sino también las instrucciones detalladas que guían la generación de contenido para cada dominio específico.

Este trabajo futuro, finalmente, también abriría nuevas oportunidades para el análisis transversal del rendimiento de los estudiantes, al permitir la comparación entre materias, el seguimiento de trayectorias individuales en diferentes cursos, y la identificación de patrones de aprendizaje a mayor escala.

5.1.7. Evaluación automática de la calidad y correctitud de las preguntas generadas

En la etapa de experimentación se implementó un proceso sistemático para evaluar manualmente la calidad y correctitud de las preguntas generadas por el sistema. Dicho proceso se describió en detalle en la sección correspondiente 4.2, y fue fundamental para el refinamiento de la entrada recibida por el modelo de lenguaje.

Si bien este enfoque podría seguir utilizándose durante la puesta en producción del sistema, su naturaleza manual limita su escalabilidad y no permite aprovechar completamente la disponibilidad de datos reales de estudiantes interactuando con el sistema. Por esta razón, una de las líneas de trabajo futuro es desarrollar mecanismos automáticos que permitan evaluar la calidad y correctitud de las preguntas generadas en contexto real de uso, de forma continua y con la menor intervención humana posible.

Es importante aclarar que este tipo de análisis automático no reemplaza el juicio académico ni la revisión cualitativa realizada por docentes y expertos en la materia, sino que actúa como un complemento. Es un enfoque que facilita la detección sistemática de preguntas problemáticas y, potencialmente, podría automatizar parcialmente el proceso de depuración, eliminando del banco aquellas preguntas que con regularidad presentan bajo rendimiento.

A continuación, se describe con mayor profundidad una propuesta de verificación automática, así como también posibles estrategias para evaluar la calidad de las preguntas a partir del análisis de datos reales.

Evaluación de correctitud

Se propone la incorporación de un módulo que verifique automáticamente la correctitud de aquellas preguntas generadas que contengan fragmentos de código. Aunque técnicamente factible desde etapas tempranas, se optó por no incluirla en el sistema dado que el proceso manual fue suficiente para validar la correctitud del contenido generado durante el desarrollo. Parte de dicho proceso manual consiste en compilar, ejecutar y comparar el resultado con la solución esperada; la propuesta actual busca automatizar esta tarea.

El objetivo de este módulo es compilar y ejecutar el código presente en el enunciado, y comparar el resultado de dicha ejecución con la opción marcada como respuesta correcta. Si el resultado coincide, la pregunta se considera válida; en caso contrario, se descarta y se solicita al modelo que genere una nueva.

Este enfoque tiene como propósito evitar la persistencia en el sistema de preguntas con errores técnicos, los cuales podrían afectar negativamente la experiencia del estudiante o introducir confusión.

Actualmente, el sistema incluye una funcionalidad que permite a los usuarios reportar preguntas incorrectas. El módulo que se propone tiene el potencial de disminuir significativamente la cantidad de preguntas defectuosas que llegan a los usuarios, y por lo tanto reducir la necesidad de recurrir a dicho mecanismo de reporte. A su vez, los reportes de preguntas con código que logren pasar esta validación automatizada podrían servir como indicadores de fallos en el módulo de evaluación, permitiendo así monitorear y mejorar su precisión de forma continua.

Cabe destacar que este método es aplicable únicamente a materias relacionadas a la programación y a preguntas generadas cuyo enunciado contiene código y cuya consigna implica predecir su salida o completar código faltante (*fill-in-the-blanks*). No resulta adecuado para preguntas de tipo conceptual o teórico, en las que la comprensión del código no se refleja directamente en su ejecución.

La implementación de este módulo también presenta ciertos desafíos técnicos. Por un lado, implicaría un incremento en el tiempo de procesamiento necesario para completar la generación de una pregunta, dado que sería necesario compilar el código con un compilador de Pascal y ejecutar el resultado antes de persistir la pregunta en la base de datos. Por otro lado, existe la posibilidad de que el código generado contenga bucles infinitos u otros comportamientos no deseados, lo cual podría provocar bloqueos o demoras excesivas durante la verificación. Para mitigar este riesgo, se debería implementar un límite de tiempo de ejecución que permita abortar la evaluación si se excede un umbral definido.

Finalmente, en caso de que una pregunta sea considerada incorrecta, el sistema intentaría generar automáticamente una nueva. Esto implica la necesidad de definir también un número máximo de intentos consecutivos para evitar bucles infinitos de generación en los que el modelo no logre producir una pregunta válida.

Evaluación de calidad

Una forma de abordar la evaluación automática de la calidad de las preguntas es a través del **análisis de ítems**, utilizando métricas estadísticas para interpretar el rendimiento de las preguntas a partir de las respuestas de los estudiantes.

La idea central es que, a partir del desempeño en los cuestionarios, se puedan aplicar métricas que permitan detectar preguntas mal formuladas, poco útiles para evaluar la comprensión o con distractores ineficaces. Para que estas métricas sean aplicables, es necesario que cada pregunta sea respondida por múltiples estudiantes en distintas instancias. Esto implica reutilizar preguntas generadas previamente, o mantener un banco estable, ya que sin una muestra suficiente de respuestas por pregunta no es posible realizar análisis estadísticos significativos. Este mecanismo se encuentra planteado como línea de desarrollo futura en la sección 5.1.9.

El trabajo futuro en esta área se centraría en calcular e interpretar automáticamente dos métricas importantes del análisis de ítems (Assess.com, 2024):

- **Índice de Dificultad**: permite medir qué tan fácil o difícil resulta una pregunta para los estudiantes.
- **Índice de Discriminación**: permite evaluar si la pregunta distingue correctamente entre los estudiantes con alto y bajo rendimiento general.

La elección de estas métricas se basa en los análisis de Chiavaroli y Familari (Chiavaroli y Familari, 2011), quienes demuestran particularmente que el índice de discriminación es una herramienta esencial para validar exámenes. Su estudio resalta que, a diferencia de solo observar el porcentaje de respuestas correctas, el análisis de este índice permite identificar errores en la puntuación, detectar preguntas defectuosas y confirmar la validez de preguntas desafiantes.

Para una descripción detallada de los fundamentos teóricos, el cálculo e interpretación de estas métricas, véase el documento "Anexo: Fundamentos del Análisis de Ítems".

5.1.8. Ampliación de reportes docentes

Actualmente, el sistema incluye una página de reportes destinada a docentes que permite visualizar estadísticas tanto a nivel general del curso como por estudiante en concreto.

Si bien esta información ofrece una visión general útil, se plantea como trabajo futuro la ampliación de este módulo con estadísticas más detalladas y orientadas a apoyar la toma de decisiones de los profesores. El objetivo de esta ampliación no es solo brindar más datos, sino fortalecer la capacidad del sistema para evaluar con mayor precisión qué tan útil resulta la aplicación para el aprendizaje de los estudiantes. En particular, busca generar evidencia que permita discernir si el uso de la plataforma contribuye efectivamente a una mejora en los niveles de comprensión y capacidad cognitiva de los estudiantes, tanto a nivel individual como grupal.

A continuación, se presentan algunas propuestas de ampliación basadas en nuestro análisis técnico. Sin embargo, consideramos fundamental recoger directamente la opinión y experiencia de los docentes, ya que incorporar sus comentarios permitirá ajustar y mejorar estas ideas, asegurando que respondan a sus necesidades reales.

Una posible extensión es la incorporación del **análisis de ítems** mencionado en 5.1.7, incluyendo los índices de dificultad y de discriminación, calculados a partir de los resultados reales obtenidos por los estudiantes.

Otras posibles mejoras incluyen:

- **Evolución por tema**: permite observar la evolución del rendimiento de los estudiantes a lo largo del curso, segmentada por temas específicos, para identificar en qué momentos ciertos conceptos generaron mayores dificultades.
- **Evolución de aciertos por estudiantes**: representa el porcentaje de respuestas correctas de cada estudiante a lo largo del tiempo, facilitando la detección de patrones de mejora, estancamientos o retrocesos.
- **Nivel de dificultad alcanzado**: calcula la dificultad promedio de las preguntas que un estudiante responde correctamente (basado en el índice de dificultad), para evaluar si el estudiante está consolidando conocimientos básicos o si ya domina conceptos avanzados.
- **Comparación de resultados según uso de la plataforma**: contrasta el desempeño entre grupos de estudiantes que utilizan activamente FinQuiz y aquellos que no, para medir el impacto directo del uso de la herramienta sobre el aprendizaje.

La incorporación de estos indicadores mejoraría la capacidad del sistema para reflejar el proceso de aprendizaje. Por un lado, permitiría analizar la calidad y pertinencia de las preguntas generadas, identificando cuáles funcionan como se espera y cuáles generan confusión. Por otro lado, facilitaría la identificación de tendencias y patrones de progreso o dificultad a nivel grupal e individual, lo que permite la detección temprana de problemas y la personalización de la enseñanza.

Para potenciar la validez de estas métricas, especialmente el análisis comparativo, se podría incorporar un mecanismo para importar resultados de evaluaciones externas. Cargar datos de exámenes o parciales oficiales (por ejemplo, mediante un archivo en formato CSV) permitiría cruzar el desempeño dentro de la aplicación con resultados obtenidos en instancias de evaluación formales. Esto ofrecería evidencia más robusta y concluyente sobre la utilidad de la plataforma como herramienta de apoyo al desarrollo cognitivo de los estudiantes.

5.1.9. Creación de banco de preguntas y generación asincrónica

Actualmente, la generación de un cuestionario es un proceso sincrónico y único para cada estudiante, lo que significa que las preguntas se crean bajo demanda y no se reutilizan, ya sea entre distintos usuarios o para un mismo usuario. Este enfoque, si bien garantiza variedad, presenta dos problemas principales: impacta en los tiempos de carga que el estudiante percibe al solicitar un cuestionario y no capitaliza el valor de las preguntas que han demostrado ser de alta calidad.

Para abordar estos puntos, se propone el desarrollo de un sistema enfocado en la **reutilización de preguntas** y la **optimización del proceso de generación**.

El objetivo es construir un banco de preguntas validadas que puedan ser seleccionadas para conformar nuevos cuestionarios, siempre respetando la asociación entre la pregunta y su tema correspondiente, y verificando que no hayan sido previamente respondidas por el estudiante. El criterio para que una pregunta sea considerada "exitosa" y, por tanto, candidata a ser reutilizada, se basaría en una lógica que combine la retroalimentación de los usuarios con métricas objetivas de calidad:

- **Valoraciones de estudiantes**: se utilizaría el sistema de valoración positiva ya existente para medir la aceptación de una pregunta. Sería necesario definir un umbral de valoraciones positivas para que una pregunta ingrese al banco de reutilizables.

- **Gestión de reportes**: se refinaría el sistema de reportes. En lugar de invalidar una pregunta con un solo reporte, se podría establecer una política más robusta, como definir el "peso" de un reporte negativo o requerir múltiples reportes para descartar una pregunta.

- **Criterios estadísticos**: se incorporarían el índice de dificultad y el índice de discriminación, mencionados en 5.1.7, como una capa de validación objetiva. Una pregunta ideal tendría una dificultad media y una alta capacidad de discriminación.

La implementación de este banco de preguntas mejoraría la experiencia de usuario al reducir significativamente los tiempos de espera, que actualmente representan una de las debilidades más importantes del sistema. Esta herramienta permitiría la creación inmediata de cuestionarios con preguntas ya validadas, no solo agilizando el acceso, sino también aumentando la calidad general del contenido.

Para complementar esta propuesta y asegurar que este banco de preguntas se mantenga poblado con contenido nuevo y relevante, se propone complementar la estrategia con un mecanismo de **generación asincrónica**. En lugar de invocar al LLM únicamente a demanda del estudiante, el sistema generaría preguntas en segundo plano, poblando el banco durante períodos de inactividad. Se definiría un tamaño máximo para este conjunto de preguntas para mantener el proceso eficiente, y dicho límite podría estar condicionado además por controles que revisen la diversidad y originalidad de las preguntas generadas, evitando así saturar el banco con variantes demasiado similares o redundantes.

Por último, es importante destacar que esta estrategia de generación asincrónica no solo optimiza los tiempos de respuesta, sino que también presenta una oportunidad de reducción de costos operativos, ya que permitiría realizar llamadas a las APIs de los modelos de lenguaje en horarios de baja demanda, que suelen tener tarifas más económicas (Google, s.f.-a).

5.1.10. Despliegue en un entorno real de uso

Uno de los pasos fundamentales para habilitar el uso de la herramienta por parte de estudiantes y docentes es su despliegue en un entorno de producción. Hasta el momento, el sistema se ha ejecutado en entornos locales de desarrollo, lo que permitió su implementación, prueba y validación. Sin embargo, para que la plataforma pueda ser utilizada efectivamente en contextos educativos reales, es necesario realizar su publicación en servidores remotos y configurar correctamente los entornos correspondientes.

Para ello, es necesario definir un proveedor de infraestructura para cada uno de los tres componentes principales del sistema: aplicación backend (Ruby on Rails), aplicación frontend (React) y base de datos (PostgreSQL)

Estos componentes pueden ser desplegados en proveedores distintos, ya que no existe una restricción que obligue a utilizar una única plataforma. Lo que sí es necesario es configurar correctamente las variables de entorno en cada servidor, de modo que puedan comunicarse entre sí como ya lo hacen en el entorno de desarrollo local.

A la hora de elegir estos proveedores, existen dos posibilidades. Por un lado, utilizar servicios en la nube comerciales como Google Cloud, AWS o Heroku, lo cual implica costos variables según el uso. Por otro lado, una alternativa viable

y sin costo económico es desplegar el sistema en los servidores de la UDELAR, *ClusterUY*.

Con respecto al servicio que provee la comunicación con el modelo de lenguaje, se requeriría asumir los costos asociados a las llamadas a la API. Durante el desarrollo de este proyecto, se utilizaron créditos gratuitos provistos por el programa de prueba de Google Cloud (300 USD por 90 días), lo cual permitió experimentar con recursos reales sin incurrir en gastos. Sin embargo, en un entorno de producción, este beneficio dejaría de estar disponible y se deberían asumir los costos correspondientes, especialmente si se prevé una mayor carga de uso por parte de múltiples usuarios concurrentes.

Finalmente, una vez desplegada la aplicación, se podrá avanzar con nuevas etapas del proyecto que dependen directamente de esta instancia: iniciar el proceso de inscripción de usuarios, configuración de cursos, y cargar los materiales correspondientes.

Tanto los flujos de manejo de usuarios, como el de creación de curso no fueron expuestos a los usuarios de la aplicación, una decisión que fue tomada con el objetivo de priorizar la implementación de otras partes de la misma.

Dicho esto, la iteración actual del sistema opta por un enfoque lo menos restrictivo posible, dando lugar a una implementación rápida de estas interacciones una vez se establezca el diseño de las mismas.

Los usuarios deben ser creados utilizando la herramienta por consola de la aplicación backend, asignando a cada uno un email único, una contraseña, un nombre, un *nickname* y un rol. Además, dada la característica del sistema, resulta simple agregar restricciones a este proceso, por lo que seria posible restringir el acceso a usuarios con correos académicos o a una lista de usuarios habilitados.

Cabe destacar que, a pesar de que hoy en día el manejo de usuario debe ser hecho directamente desde el backend, gracias a las librerías utilizadas para la autenticación (Heartcombo, 2009) (Hurley, 2014), el único paso necesario para poder exponer estas funcionalidades en la aplicación es la implementación de sus vistas, dado que la lógica ya se encuentra implementada.

Por otra parte, utilizando la misma herramienta por consola de la aplicación backend es posible crear cursos dado su nombre y descripción. Con la implementación actual, los usuarios pueden pertenecer a más de un curso y todos son asignados al curso de Programación 1 por defecto. A futuro, una vez se decida cuál es la interacción deseada, puede ser necesario implementar vistas que expongan a los usuarios las acciones definidas en torno al manejo de los cursos.

5.2. Conclusiones

En las siguientes líneas se exponen las conclusiones generales del presente trabajo, integrando consideraciones sobre la etapa de investigación y el desarrollo del sistema. El objetivo es ofrecer una reflexión crítica sobre los resultados obtenidos y el valor potencial de la herramienta propuesta.

Aporte al campo de estudio

A lo largo del desarrollo de este trabajo, al adentrarnos en el campo de la Inteligencia Artificial aplicada a la educación, y en particular a la generación de cuestionarios, fue posible confirmar una impresión inicial: se trata de un ámbito profundamente dinámico, en el que los avances se suceden a gran velocidad y el estado del arte varía año a año, volviendo obsoletas incluso investigaciones relativamente recientes.

En un escenario tan cambiante, donde incluso resulta difícil anticipar cuán vigentes podrán mantenerse estas conclusiones a lo largo del tiempo, la realización de un trabajo de investigación y desarrollo que busca aplicar los avances más recientes a un entorno educativo tan específico representa, en sí misma, una contribución valiosa.

Teniendo esto en cuenta, es importante destacar que el trabajo realizado puede servir como punto de partida para nuevas exploraciones en el campo de la Inteligencia Artificial aplicada a la educación, ya sea en vistas a mejorar el sistema propuesto o a desarrollar otros enfoques alternativos.

Consideraciones generales

Antes de analizar los resultados del sistema desarrollado, es necesario realizar una aclaración importante: las conclusiones que puedan tomarse se basan en la experiencia obtenida con el curso de *Programación 1*. A pesar de esto, varios elementos son independientes de la asignatura seleccionada.

Teniendo esto en cuenta, es importante señalar que, aunque el sistema fue aplicado inicialmente a una única materia, los resultados obtenidos demuestran la viabilidad de incorporar este tipo de herramientas en un entorno educativo real aplicando técnicas similares a otros cursos. No obstante, la bibliografía consultada advierte que podrían surgir dificultades al trasladar estos enfoques a ciertas áreas del conocimiento, como las disciplinas matemáticas. Aún queda por investigar si estas limitaciones siguen vigentes o si podrían superarse mediante ajustes metodológicos o mediante los avances técnicos más recientes.

Exploración de Modelos de Lenguaje de Última Generación

Desde el inicio de la implementación, se optó por utilizar modelos de lenguaje de alta capacidad con amplios contextos, evitando técnicas como la incorporación de *RAG* o *fine-tuning*, con el objetivo de experimentar puramente con las capacidades de los más potentes grandes modelos de lenguaje disponibles. Una vez finalizada la implementación, puesta a prueba la herramienta y analizados los resultados obtenidos, es posible realizar una serie de valoraciones respecto a la decisión adoptada.

Por un lado, este enfoque, centrado en aprovechar la capacidad del modelo y su amplia ventana de contexto, permite desacoplar la lógica del sistema del modelo en sí mismo, lo que facilita un intercambio ágil y fomenta la experimentación y la mejora continua. Esto también facilita la adaptación del sistema a

nuevas materias o contextos, lo cual era uno de los objetivos principales, requiriendo únicamente un ajuste de las *prompts* utilizadas y un proceso posterior de refinamiento, sin necesidad de llevar a cabo un entrenamiento adicional ni la generación de un corpus de conocimiento extenso.

En cuanto a los resultados obtenidos en la generación de cuestionarios, se observa que la calidad de las preguntas generadas demuestra que este tipo de modelos, incluso sin grandes adaptaciones, es capaz de producir resultados satisfactorios, requiriendo además menos trabajo manual y ofreciendo mayor flexibilidad en su aplicación. A su vez, el rápido avance tecnológico en este campo permite anticipar que, en el futuro cercano, estas capacidades se incrementarán aún más, mejorando la eficacia y los resultados posibles.

A pesar de estos aspectos positivos, la adopción de este enfoque también presenta ciertas desventajas. No obstante, es importante considerar que, en su mayoría, estas limitaciones podrían volverse obsoletas o reducir significativamente su impacto a medida que continúan los avances técnicos y emergen nuevos modelos.

Uno de los principales inconvenientes asociados al uso de modelos tan potentes es el tiempo de respuesta. El procesamiento de una gran cantidad de variables, combinado con las limitaciones del *hardware* actualmente disponible, puede llevar a demoras que resultan incompatibles con un uso en tiempo real. En el contexto de nuestro sistema, esto afecta directamente la experiencia del estudiante, quien podría verse obligado a esperar varios minutos para que su cuestionario sea generado.

Si bien es probable que estas demoras disminuyan con los avances técnicos futuros, el problema ha sido tenido en cuenta durante la creación y análisis del sistema. Como parte de los trabajos a futuro, se ha planteado un plan de mejora que incluye distintas alternativas para mitigar este problema, como se detalla en las secciones 5.1.9 y 5.1.2.

Estas mejoras no solo apuntan a mitigar los tiempos de respuesta, sino que también abordan otro de los principales desafíos asociados al uso de modelos de lenguaje de alta capacidad: el costo económico. Si bien el desarrollo del sistema fue posible gracias a programas de prueba que permitieron acceder gratuitamente al modelo seleccionado, en un entorno real de uso, los costos podrían ser significativos, especialmente en contextos con presupuestos limitados.

Por último, a esto se suman ciertas dificultades inherentes al uso de modelos de lenguaje de gran escala, independientemente de su potencia o características particulares. Una de las más notorias durante el desarrollo fue la limitada capacidad para comprender el proceso de razonamiento que realiza el modelo. Esta opacidad complejiza la tarea de realizar ajustes puntuales sobre las salidas generadas, ya que no siempre resulta evidente qué modificaciones en la entrada producirán el efecto deseado.

Resultados

En cuanto a los resultados obtenidos, luego de múltiples análisis realizados tanto durante el desarrollo como tras su culminación, incluyendo una revisión

conjunta con parte del equipo docente de Programación 1, se observan resultados muy positivos. Esto es especialmente destacable considerando que el sistema desarrollado corresponde a un MVP (Producto Mínimo Viable, por sus siglas en inglés), con amplio margen aún para futuras mejoras.

Desde una perspectiva de estudiantes en el momento final de su carrera, vemos en el sistema una herramienta útil que podría haber nutrido nuestro proceso de aprendizaje ampliamente. Las preguntas generadas, salvo por algunas situaciones puntuales ya identificadas, son mayoritariamente correctas. Además, presentan una calidad destacable, con explicaciones claras que contribuyen a que el estudiante comprenda mejor el tema evaluado. A su vez, la disponibilidad de material complementario favorece el repaso, brindando acceso rápido y sencillo a los contenidos relevantes.

Sin bien los resultados son alentadores, es fundamental señalar que la verdadera efectividad del sistema solo podrá evaluarse de manera concluyente en un contexto de uso real. La herramienta incorpora mecanismos que permiten analizar su impacto en el aprendizaje del estudiante mediante un gran abanico de reportes, en la medida en que lo posibilita un sistema de evaluación cuantitativa. Queda por comprobar su desempeño en funcionamiento, contrastando los resultados de los estudiantes que utilicen la herramienta con los de un grupo de control, a fin de validar empíricamente la hipótesis sobre su efectividad.

Reflexiones finales

Luego de alcanzar un desempeño satisfactorio en el sistema desarrollado, explorar el estado del arte en inteligencia artificial aplicada a la educación y experimentar con el uso de grandes modelos de lenguaje en este contexto, es posible plantear algunas reflexiones relevantes.

La intervención docente, incluso en sistemas altamente automatizados como el desarrollado, continúa siendo un factor clave para alcanzar buenos resultados. Aunque el sistema sea capaz de generar cuestionarios de alta calidad, el camino hacia esa calidad, desde el diseño y evaluación hasta la configuración óptima, requiere conocimientos que van más allá de lo técnico y documentado. La perspectiva pedagógica, la comprensión de los objetivos de enseñanza y la experiencia del equipo docente son elementos esenciales que hacen que su participación siga siendo indispensable en el funcionamiento y la mejora del sistema. Sin embargo, una vez alcanzada una configuración adecuada y comprobada su eficacia, el sistema tiene el potencial de liberar a los docentes de parte del tiempo destinado al diseño de autoevaluaciones o ejercicios prácticos. Esto permitiría optimizar el uso del tiempo disponible y focalizar los esfuerzos en otros objetivos pedagógicos de mayor valor agregado.

El sistema, con sus fortalezas y aspectos por mejorar, reviste una importancia significativa en sí mismo. Representa una prueba concreta de la viabilidad de incorporar este tipo de tecnologías en el contexto educativo de la Facultad de Ingeniería, abriendo la posibilidad de modernizar las herramientas disponibles en la carrera. Además de brindar apoyo directo a los estudiantes en su proceso formativo y generar datos valiosos para la institución, el sistema deja entre-

ver una posible estrategia para abordar uno de los desafíos estructurales más relevantes: la masificación estudiantil.

Como línea de análisis futura, que fue considerada pero requiere una profundización mayor por parte de especialistas en la materia, se destaca la necesidad de revisar aspectos vinculados a la protección de los datos utilizados, la elección de emplear un modelo de lenguaje externo y sus posibles implicancias en un contexto académico público, así como también las consideraciones éticas asociadas al uso de este tipo de sistemas y las potenciales consecuencias medioambientales que su implementación pueda generar.

Referencias

Alexandru, A., Tirziu, E., Tudora, E., y Bica, O. (2015, 03). Enhanced education by using intelligent agents in multi-agent adaptive e-learning systems. *Studies in Informatics and Control*, *Vol 24*, pp. 13-22.

Alshboul, J., y Baksa-Varga, E. (2024). A hybrid approach for automatic question generation from program codes. *International Journal of Advanced Computer Science and Applications*, *15*(1). Descargado de http://dx.doi.org/10.14569/IJACSA.2024.0150102 doi: 10.14569/IJACSA.2024.0150102

Al-Rukban, M. O. (2006). Guidelines for the construction of multiple choice questions tests. *Journal of Family & Community Medicine*, *13*(3), 125–133. Descargado de https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3410060/ (Accessed: 2025-06-13) doi: 10.4103/2230-8229.97543

'Amazon'. (s.f.). *What is ocr?* https://aws.amazon.com/what-is/ocr/. ([Accessed 18-07-2025])

Amazon Web Services. (s.f.). *What is retrieval-augmented generation?* https://aws.amazon.com/what-is/retrieval-augmented-generation/. (Consultado el 14 de junio de 2025)

Amazon Web Services. (2024). *Bases de datos relacionales frente a los no relacionales: diferencia entre tipos de bases de datos.* Descargado de https://aws.amazon.com/es/compare/the-difference-between-relational-and-non-relational-databases/ (Accedido el 21 de abril de 2025)

Amazon Web Services. (2025). *What is retrieval-augmented generation?* https://aws.amazon.com/what-is/retrieval-augmented-generation. (Accessed: 2025-07-18)

Assess.com. (2024). *Análisis y estadísticas de ítems.* Descargado de https://assess.com/analisis-y-estadisticas-de-items/ (Consultado el 6 de julio de 2025)

Atlassian. (s.f.). *Tableros kanban.* https://www.atlassian.com/es/agile/kanban/boards. Descargado de https://www.atlassian.com/es/agile/kanban/boards (Accedido el 6 de mayo de 2025)

Bahdanau, D., Cho, K., y Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. *CoRR*, *abs/1409.0473*.

Baker, T., Smith, L., y Anissa, N. (2019). *Educ-ai-tion rebooted? exploring the future of artificial intelligence in schools and colleges.* https://www

.nesta.org.uk/report/education-rebooted/. (Accessed: 2024-12-09)

Begum, T. (2012, noviembre). A guideline on developing effective multiple choice questions and construction of single best answer format. *Journal of Bangladesh College of Physicians and Surgeons*, *30*(3), 159–166. Descargado de http://dx.doi.org/10.3329/jbcps.v30i3.12466 doi: 10.3329/jbcps.v30i3.12466

Bengio, Ducharme, Vincent, y Jauvin. (2000). A neural probabilistic language mode. *CrossRef Listing of Deleted DOIs*, *1*. Descargado de http://dx.doi.org/10.1162/153244303322533223 doi: 10.1162/153244303322533223

Brigham Young University Testing Center. (s.f.). *14 rules for writing multiple-choice questions.* Descargado de https://testing.byu.edu/handbooks/14%20Rules%20for%20Writing%20Multiple-Choice%20Questions.pdf (Accessed: 2025-06-11)

Budescu, D. V., y Nevo, B. (1985, septiembre). Optimal number of options: An investigation of the assumption of proportionality. *Journal of Educational Measurement*, *22*(3), 183–196. Descargado de http://dx.doi.org/10.1111/j.1745-3984.1985.tb01057.x doi: 10.1111/j.1745-3984.1985.tb01057.x

CanCanCommunity. (2015). *Cancancan - authorization gem for ruby on rails.* https://www.rubydoc.info/github/CanCanCommunity/cancancan. (Accedido el 3 de mayo de 2025)

Capstone Edu LLC. (2025). *Quizbot.ai: Ai-powered quiz question generator.* https://quizbot.ai/. (Accedido el 13 de junio de 2025)

Centre for Teaching Excellence. (s.f.). *Designing multiple-choice questions.* Descargado 2025-06-11, de https://uwaterloo.ca/centre-for-teaching-excellence/catalogs/tip-sheets/designing-multiple-choice-questions

Chiavaroli, N., y Familari, M. (2011). When majority doesn't rule: The use of discrimination indices to improve the quality of mcqs. *Bioscience Education*, *17*(1), 1–7. Descargado de https://doi.org/10.3108/beej.17.8 (Consultado el 6 de julio de 2025) doi: 10.3108/beej.17.8

Chou, C.-Y., Chan, T.-W., y Lin, C.-J. (2003, 04). Redefining the learning companion: The past, present, and future of educational agents. *Computers & Education*, *40*, 255-269. doi: 10.1016/S0360-1315(02)00130-6

Codingscape. (2025, marzo). *Llms with largest context windows.* Descargado de https://codingscape.com/blog/llms-with-largest-context-windows (Last updated: March 2025. Accessed: 2025-05-10)

Collobert, R., y Weston, J. (2008). A unified architecture for natural language processing: deep neural networks with multitask learning. En *Proceedings of the 25th international conference on machine learning - icml '08* (p. 160–167). ACM Press. Descargado de http://dx.doi.org/10.1145/1390156.1390177 doi: 10.1145/1390156.1390177

Contributors, A. L. (2025). *Awesome llm.* https://github.com/Hannibal046/Awesome-LLM. (Último acceso: 24 de marzo de 2025)

Dede, C. J. e. a. (1987). *Intelligent computer-assisted instruction: A review and assessment of icai research and its potential for education* (Inf. Téc.). ": Educational Technology Center, Cambridge, MA.

DeepSeek AI. (2025, 28 de mayo). *Deepseek-r1 .* https://api-docs.deepseek.com/zh-cn/news/news250528. (Publicado en la documentación oficial de DeepSeek, 28 de mayo de 2025)

Divate, M., y Salgaonkar, A. (2017, noviembre). Automatic question generation approaches and evaluation techniques. *Current Science*, *113*(09), 1683. Descargado de http://dx.doi.org/10.18520/cs/v113/i09/1683-1691 doi: 10.18520/cs/v113/i09/1683-1691

Doughty, J., Wan, Z., Bompelli, A., Qayum, J., Wang, T., Zhang, J., ... Sakr, M. (2023). A comparative study of ai-generated (gpt-4) and human-crafted mcqs in programming education. Descargado de https://arxiv.org/abs/2312.03173 doi: 10.48550/ARXIV.2312.03173

Duchastel, P., y Imbeau, J. (1988). Intelligent computer-assisted instruction (icai): Flexible learning through better student-computer interaction. *Journal of Information Technology*, *3*(2).

Ferguson, R. (2012). Learning analytics: drivers, developments and challenges. *International Journal of Technology Enhanced Learning*, *4*(5/6), 304. Descargado de http://dx.doi.org/10.1504/IJTEL.2012.051816 doi: 10.1504/ijtel.2012.051816

Figma Inc. (2025). *¿Qué es Figma?* https://help.figma.com/hc/es-419/articles/14563969806359--Qu%C3%A9-es-Figma. (Accedido el 12 de junio de 2025)

Gilani, A. S. (2018). *Rails service objects: A comprehensive guide.* https://www.toptal.com/ruby-on-rails/rails-service-objects-tutorial. (Consultado el 6 de mayo de 2025)

GitHub. (2024). *Acerca de github y git.* Descargado de https://docs.github.com/es/get-started/start-your-journey/about-github-and-git (Accedido el 19 de abril de 2025)

GitLab. (2023). *¿que es un sistema de control de versiones distribuido?* Descargado de https://about.gitlab.com/es/topics/version-control/benefits-distributed-version-control-system/ (Accedido el 19 de abril de 2025)

GitLab Inc. (2025). *Gitlab: The devsecops platform.* Descargado de https://about.gitlab.com/ (Accedido el 21 de abril de 2025)

Google. (s.f.-a). *Batch mode — gemini api.* https://ai.google.dev/gemini-api/docs/batch-mode. (Accedido el 13 de julio de 2025)

Google. (s.f.-b). *Datasets: Dividing the original dataset — Machine Learning — Google for Developers — developers.google.com.* https://developers.google.com/machine-learning/crash-course/overfitting/dividing-datasets. ([Accessed 19-07-2025])

Google AI for Developers. (s.f.). *Understand and count tokens.* https://ai.google.dev/gemini-api/docs/tokens?authuser=1&lang=python. (Consultado: 8 de junio de 2025)

Google Cloud. (s.f.). *What is supervised learning.* https://cloud.google.com/discover/what-is-supervised-learning. (Consultado el 11 de agosto de 2024)

Goutham, R. (2020). *Questgen.ai: Question generation using state-of-the-art natural language processing algorithms.* Descargado de https://github.com/ramsrigouthamg/Questgen.ai (Accessed: 2025-05-12)

Guo, S., Liao, L., Li, C., y Chua, T.-S. (2024). *A survey on neural question generation: Methods, applications, and prospects.* arXiv. Descargado de https://arxiv.org/abs/2402.18267 doi: 10.48550/ARXIV.2402.18267

Gándara, D., Anahideh, H., Ison, M. P., y Picchiarini, L. (2024, enero). Inside the black box: Detecting and mitigating algorithmic bias across racialized groups in college student-success prediction. *AERA Open*, *10*. Descargado de http://dx.doi.org/10.1177/23328584241258741 doi: 10.1177/23328584241258741

Haladyna, T. M., Downing, S. M., y Rodriguez, M. C. (2002, julio). A review of multiple-choice item-writing guidelines for classroom assessment. *Applied Measurement in Education*, *15*(3), 309–333. Descargado de http://dx.doi.org/10.1207/s15324818ame1503_5 doi: 10.1207/s15324818ame1503_5

Hang, C. N., Wei Tan, C., y Yu, P.-D. (2024). Mcqgen: A large language model-driven mcq generator for personalized learning. *IEEE Access*, *12*, 102261–102273. Descargado de http://dx.doi.org/10.1109/ACCESS.2024.3420709 doi: 10.1109/access.2024.3420709

Hansson, D. H. (2025). *The rails doctrine: Convention over configuration.* https://rubyonrails.org/doctrine#convention-over-configuration. (Accedido el 1 de mayo de 2025)

Heartcombo. (2009). *devise.* https://github.com/heartcombo/devise. (Accedido el 3 de mayo de 2025)

Hurley, L. D. (2014). *devise_token_auth.* https://github.com/lynndylanhurley/devise_token_auth. (Accedido el 3 de mayo de 2025)

IBM. (s.f.-a). *Neural networks.* Descargado de https://www.ibm.com/topics/neural-networks

IBM. (s.f.-b). *What Are Large Language Models (LLMs)?* https://www.ibm.com/think/topics/large-language-models. ([Accessed 16-07-2025])

'IBM'. (s.f.). *What is computer vision?* https://www.ibm.com/topics/computer-vision. (Accessed: 2024-9-28)

IBM. (s.f.). *What is unsupervised learning?* https://www.ibm.com/topics/unsupervised-learning. (Consultado el 16 de setiembre de 2024)

IBM. (2025). *What is backpropagation?* https://www.ibm.com/think/topics/backpropagation. (Consultado: 27 de julio de 2025)

J Meenakumari, J. M. K. (2015, May). Learning analytics and its challenges in education sector: A survey. *An Architectural Framework for Workload Demand Prediction in Scalable Federated Clouds*, *ICCTAC2015*(2), 6-10. Descargado de /proceedings/icctac2015/number2/20925-2012/

Johnson, J. j. (2025, 22 de February). *Small language models (slm): A comprehensive overview.* Hugging Face Blog. Descargado de https://huggingface.co/blog/jjokah/small-language-model

Jung, A. (2022a). Components of ml. En *Machine learning: The basics* (pp. 19–56). Singapore: Springer. Descargado de https://doi.org/10.1007/978-981-16-8193-6_2 doi: 10.1007/978-981-16-8193-6_2

Jung, A. (2022b). Introduction. En *Machine learning: The basics* (pp. 1–17). Singapore: Springer. Descargado de https://doi.org/10.1007/978-981-16-8193-6_1 doi: 10.1007/978-981-16-8193-6_1

Kavukcuoglu, K. (2025, marzo). *Gemini 2.5: Our most intelligent ai model.* Descargado de https://blog.google/technology/google-deepmind/gemini-model-thinking-updates-march-2025/ (Accessed: 2025-05-10)

Kelly, C. (2024, 15 de septiembre). *Chain of thought prompting (cot).* https://humanloop.com/blog/chain-of-thought-prompting/. (Publicado el 15 de septiembre de 2024. Accedido: 18 de julio de 2025)

Khurana, D., Koli, A., Khatter, K., y Singh, S. (2022). Natural language processing: state of the art, current trends and challenges. *Machine Learning and Knowledge Extraction*. (Corrected publication 2022. © The Author(s), under exclusive licence to Springer Science+Business Media, LLC, part of Springer Nature) doi: 10.1007/s11042-022-13428-4

Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., ... Zettlemoyer, L. (2019). *Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension.* arXiv. Descargado de https://arxiv.org/abs/1910.13461 doi: 10.48550/ARXIV.1910.13461

Li, K., y Zhang, Y. (2024). Planning first, question second: An llm-guided method for controllable question generation. En *Findings of the association for computational linguistics acl 2024* (p. 4715–4729). Association for Computational Linguistics. Descargado de http://dx.doi.org/10.18653/v1/2024.findings-acl.280 doi: 10.18653/v1/2024.findings-acl.280

Li, W., Brooks, C., y Schaub, F. (2019, marzo). The impact of student opt-out on educational predictive models. En *Proceedings of the 9th international conference on learning analytics and knowledge* (p. 411–420). ACM. Descargado de http://dx.doi.org/10.1145/3303772.3303809 doi: 10.1145/3303772.3303809

Lin, S.-Y., y Singh, C. (2016). Can multiple-choice questions simulate free-response questions? Descargado de https://arxiv.org/abs/1603.07910 doi: 10.48550/ARXIV.1603.07910

Lions, S., Monsalve, C., Dartnell, P., Blanco, M. P., Ortega, G., y Lemarié, J. (2022, abril). Does the response options placement provide clues to the correct answers in multiple-choice tests? a systematic review. *Applied Measurement in Education*, *35*(2), 133–152. Descargado de http://dx.doi.org/10.1080/08957347.2022.2067539 doi: 10.1080/08957347.2022.2067539

Madaan, A., Tandon, N., Gupta, P., Hallinan, S., Gao, L., Wiegreffe, S., ... Clark, P. (2023). SELF-REFINE: Iterative Refinement with Self-Feedback. *arXiv preprint arXiv:2303.17651*. Descargado de https://arxiv.org/abs/2303.17651

Mahto, A. (2024, March). *How reasoning models are transforming logical ai thinking.* Descargado de https://techcommunity.microsoft.com/blog/azuredevcommunityblog/how-reasoning-models-are-transforming-logical-ai-thinking/4373194 (Microsoft Tech Community)

McCarthy, J. (2007). *What is artificial intelligence?* Descargado de http://www-formal.stanford.edu/jmc/whatisai.pdf

McMillan, J. H. (Ed.). (2007). *Formative classroom assessment.* New York, NY: Teachers' College Press.

Meta AI. (2024). *Llama 3.2: Revolutionizing edge ai and vision with open, customizable models.* https://ai.meta.com/blog/llama-3-2-connect-2024-vision-edge-mobile-devices/. (Publicado en el blog de Meta AI, 25 de septiembre de 2024)

Meta Open Source. (2024). *React – una biblioteca de javascript para construir interfaces de usuario.* Sitio web oficial. Descargado de https://es.react.dev/ (Consultado el 3 de mayo de 2025)

Microsoft. (2024). *Typescript: Typed javascript at any scale.* Sitio web oficial. Descargado de https://www.typescriptlang.org/ (Consultado el 3 de mayo de 2025)

Microsoft Corporation. (2025). *Azure repos – git repositories — microsoft azure.* Descargado de https://azure.microsoft.com/en-us/products/devops/repos (Accedido el 21 de abril de 2025)

Mikolov, T., Sutskever, I., Chen, K., Corrado, G., y Dean, J. (2013). *Distributed representations of words and phrases and their compositionality.* Descargado de http://arxiv.org/abs/1310.4546 ([Accessed 06-10-2024])

Moreno, R., Martínez, R. J., y Muñiz, J. (2006, enero). New guidelines for developing multiple-choice items. *Methodology*, *2*(2), 65–72. Descargado de http://dx.doi.org/10.1027/1614-2241.2.2.65 doi: 10.1027/1614-2241.2.2.65

Naeiji, A., An, A., Davoudi, H., Delpisheh, M., y Alzghool, M. (2023). Question generation using sequence-to-sequence model with semantic role labels. En *Proceedings of the 17th conference of the european chapter of the association for computational linguistics* (p. 2830–2842). Association for Computational Linguistics. Descargado de http://dx.doi.org/10.18653/v1/2023.eacl-main.207 doi: 10.18653/v1/2023.eacl-main.207

Neumann, J., Simmrodt, S., Bader, B., Opitz, B., y Gergs, U. (2023, marzo). Direct comparison of online tests using single-choice items or multiple-select items in pharmacology over one year. *American Journal of Educational Research*, *11*(3), 125–132. Descargado de http://dx.doi.org/10.12691/education-11-3-4 doi: 10.12691/education-11-3-4

Nielsen, A. (2015). *Neural networks and deep learning.* Determination Press.

Norooziasl, S., Fazli, B., Shojaei, H., Yazdi, M. S. G., Haghighat Shoar, S. M., Sobhani, G., ... Mousavi Bazaz, N. (2021). The op-

timal number of choices in multiple-choice tests: A systematic review. *Medical Education Bulletin*, *2*(3), 253–260. Descargado de https://www.medicaleducation-bulletin.ir/article_139233_00d0a422c0d1bf0041863579d5bd358f.pdf (Accessed: 2025-06-13) doi: 10.22034/meb.2021.311998.1031

Notion. (s.f.). *Tableros (boards).* https://www.notion.com/es/help/boards. Descargado de https://www.notion.com/es/help/boards (Accedido el 6 de mayo de 2025)

Olsho, A., Smith, T., Brahmia, S. W., Eaton, P., Zimmerman, C., y Boudreaux, A. (2021). *Online test administration results in students selecting more responses to multiple-choice-multiple-response items.* arXiv. Descargado de https://arxiv.org/abs/2106.02137 doi: 10.48550/ARXIV.2106.02137

Paviotti, G., Rossi, P. G., y Zarka. (2012). Intelligent tutoring systems: an overview. *Pensa Multimedia*, 1-176.

Perham, M. (2012). *Sidekiq - simple, efficient background processing for ruby.* https://sidekiq.org/. (Accedido el 3 de mayo de 2025)

Popham, W. J. (2009, enero). Assessment literacy for teachers: Faddish or fundamental? *Theory Into Practice*, *48*(1), 4–11. Descargado de http://dx.doi.org/10.1080/00405840802577536 doi: 10.1080/00405840802577536

Quizgecko. (2025). *Quizgecko: Ai-powered quiz flashcard generator.* https://quizgecko.com/. (Accedido el 13 de junio de 2025)

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., ... Liu, P. J. (2019). *Exploring the limits of transfer learning with a unified text-to-text transformer.* arXiv. Descargado de https://arxiv.org/abs/1910.10683 doi: 10.48550/ARXIV.1910.10683

Rajpurkar, P., Zhang, J., Lopyrev, K., y Liang, P. (2016). Squad: 100, 000+ questions for machine comprehension of text. En *Proceedings of the 2016 conference on empirical methods in natural language processing.* Association for Computational Linguistics. Descargado de http://dx.doi.org/10.18653/v1/D16-1264 doi: 10.18653/v1/d16-1264

Raymond, M. R., Stevens, C., y Bucak, S. D. (2018, octubre). The optimal number of options for multiple-choice questions on high-stakes tests: application of a revised index for detecting nonfunctional distractors. *Advances in Health Sciences Education*, *24*(1), 141–150. Descargado de http://dx.doi.org/10.1007/s10459-018-9855-9 doi: 10.1007/s10459-018-9855-9

Rodriguez, M. C. (2005, junio). Three options are optimal for multiple-choice items: A meta-analysis of 80 years of research. *Educational Measurement: Issues and Practice*, *24*(2), 3–13. Descargado de http://dx.doi.org/10.1111/j.1745-3992.2005.00006.x doi: 10.1111/j.1745-3992.2005.00006.x

Roelofs, E. (2019). A framework for improving the accessibility of assessment tasks. En *Theoretical and practical advances in computer-based educational measurement* (p. 21–45). Springer International Publishing. Descargado

de http://dx.doi.org/10.1007/978-3-030-18480-3_2 doi: 10.1007/978-3-030-18480-3_2

Rotimi Best. (2025). *Quizify: an ai quiz generator powered by vercel ai sdk.* https://github.com/rotimi-best/quizify. (Accedido el 13 de junio de 2025)

RSpec Team. (2025). *Rspec: Behaviour driven development for ruby.* https://rspec.info/. (Accedido el 3 de mayo de 2025)

Ruby on Rails. (2024). *Active record basics — ruby on rails guides.* Descargado de https://guides.rubyonrails.org/active_record_basics.html (Accedido el 9 de mayo de 2025)

Ruby on Rails Core Team. (2025). *Ruby on Rails: Compress the complexity of modern web apps.* https://rubyonrails.org/. (Accedido el 12 de junio de 2025)

Ruby Programming Language Core Team. (2025). *Official Ruby Programming Language Website.* https://www.ruby-lang.org/en/. (Accedido el 12 de junio de 2025)

Shadcn. (2024). *shadcn/ui: Componentes accesibles y modulares para react.* Sitio web oficial. Descargado de https://ui.shadcn.com/ (Consultado el 3 de mayo de 2025)

Simonini, T. (2023). *Offline vs. online reinforcement learning.* https://huggingface.co/learn/deep-rl-course/en/unitbonus3/offline-online. (Consultado el 18 de setiembre de 2024)

Swiecki, Z., Khosravi, H., Chen, G., Martinez-Maldonado, R., Lodge, J. M., Milligan, S., ... Gašević, D. (2022). Assessment in the age of artificial intelligence. *Computers and Education: Artificial Intelligence*, *3*, 100075. Descargado de http://dx.doi.org/10.1016/j.caeai.2022.100075 doi: 10.1016/j.caeai.2022.100075

Tahir. (2025). *What is ollama? running large language models locally.* Descargado de https://medium.com/@tahirbalarabe2/what-is-ollama-running-large-language-models-locally-e917ca40defe (Accedido: 2025-06-08)

Tailwind Labs. (2024). *Tailwind css: Un framework de css utilitario.* Sitio web oficial. Descargado de https://tailwindcss.com/ (Consultado el 3 de mayo de 2025)

Tan, D.-P., Ji, S.-M., y Jin, M.-S. (2013). Intelligent computer-aided instruction modeling and a method to optimize study strategies for parallel robot instruction. *IEEE Transactions on Education*, *56*(3), 268-273. doi: 10.1109/TE.2012.2212707

Tempelaar, D., Rienties, B., y Nguyen, Q. (2020, junio). Subjective data, objective data and the role of bias in predictive modelling: Lessons from a dispositional learning analytics application. *PLOS ONE*, *15*(6), e0233977. Descargado de http://dx.doi.org/10.1371/journal.pone.0233977 doi: 10.1371/journal.pone.0233977

The PostgreSQL Global Development Group. (2025). *Postgresql: The world's most advanced open source relational database.* Descargado de https://www.postgresql.org/ (Accedido el 21 de abril de 2025)

Trischler, A., Wang, T., Yuan, X., Harris, J., Sordoni, A., Bachman, P., y Suleman, K. (2017, agosto). NewsQA: A machine comprehension dataset. En P. Blunsom y cols. (Eds.), *Proceedings of the 2nd workshop on representation learning for NLP* (pp. 191–200). Vancouver, Canada: Association for Computational Linguistics. Descargado de https://aclanthology.org/W17-2623/ doi: 10.18653/v1/W17-2623

Turing, A. M. (1950). Computing machinery and intelligence. *Mind*, *59*(236), 433–460. Descargado de http://www.jstor.org/stable/2251299

University of Washington. (s.f.). *Understanding Item Analyses.* Descargado 2025-07-05, de https://www.washington.edu/assessment/scanning-scoring/scoring/reports/item-analysis/ (Accedido el 5 de julio de 2025)

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... et al., I. P. (2017). Attention is all you need. En *Advances in neural information processing systems (neurips)* (Vol. 30).

Vellum. (2025). *Llm leaderboard.* Descargado de https://www.vellum.ai/llm-leaderboard (Accedido el 31 de mayo de 2025)

Zhang, Z., Zhang, A., Li, M., y Smola, A. (2022). Automatic chain of thought prompting in large language models. *arXiv preprint arXiv:2210.03493v1*. Descargado de https://arxiv.org/abs/2210.03493v1