

UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA
DEPARTAMENTO DE ELECTRÓNICA E INFORMÁTICA
CONCEPCIÓN - CHILE



**“DESARROLLO DEL BACK-END DE UNA PLATAFORMA
EDUCATIVA GAMIFICADA COMO APOYO A PROCESOS
DE AUTORREGULACIÓN DEL APRENDIZAJE EN
ESTUDIANTES DE SEXTO BÁSICO: MONITOREO DEL
PROPIO DESEMPEÑO Y AUTORREFLEXIÓN”**

DIEGO ANTONIO PILGRIN BALCÁZAR

**MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERO EN INFORMÁTICA**

Profesor Guía: JOSÉ LUIS CARRASCO SÁEZ

Agosto – 2026



CONSTANCIA DE VALIDACIÓN Y CONFIDENCIALIDAD DE
MONOGRAFÍA A REPOSITORIO ACADÉMICO

1.- IDENTIFICACIÓN DEL TRABAJO ACADÉMICO

Tipo de monografía (marcar una opción): ☒ Memoria o trabajo de título ☐ Tesis de Postgrado

Título del trabajo: DESARROLLO DEL BACK-END DE UNA PLATAFORMA EDUCATIVA GAMIFICADA COMO APOYO A PROCESOS DE AUTORREGULACIÓN DEL APRENDIZAJE EN ESTUDIANTES DE SEXTO BÁSICO: MONITOREO DEL PROPIO DESEMPEÑO Y AUTORREFLEXIÓN

Nombre del candidato(a): Diego Antonio Pilgrin Balcázar

Carrera / Grado: Ingeniería en Informática

Campus: Sede Concepción Departamento: Electrónica e Informática

2.- VALIDACIÓN DEL PROFESOR GUÍA/DIRECTOR DE TESIS

Yo, José Luis Carrasco Sáez, en mi calidad de profesor(a) guía/director(a) del trabajo académico mencionado anteriormente **DEJO CONSTANCIA** que:

- He revisado esta versión del documento y corresponde a la versión final aprobada del trabajo.
- El trabajo cumple con los requisitos académicos y de formato establecidos por la institución.

3.- EVALUACIÓN DE CONFIDENCIALIDAD POR PROPIEDAD INDUSTRIAL (marcar una opción)

☒ El trabajo **NO contiene** información que amerite confidencialidad y puede ser publicado de inmediato en repositorio con acceso abierto.

☐ El trabajo **CONTIENE** información con potenciales implicancias de propiedad industrial o intelectual y requiere un período de confidencialidad (**embargo**) por (marcar una opción):

☐ 6 meses ☐ 12 meses ☐ 2 años ☐ 3 años ☐ 5 años ☐ 10 años

Fundamentación de la necesidad de confidencialidad (obligatorio si se solicita embargo):

4.- FIRMAS

Profesor(a) guía o director(a) de memoria o tesis:

Fecha: 22/10/2026

Firma:

Estudiante o Candidato(a):

Fecha: 21/10/2026

Firma:

Este formulario debe ser insertado como página 2 de la memoria o tesis, completado y firmado por estudiante y profesor(a) antes de la entrega en portal PRISMA de Biblioteca USM.

DEDICATORIA

A mis padres, por ser el pilar fundamental en mi vida y por enseñarme que con esfuerzo y perseverancia no existen metas inalcanzables. Este logro es, en gran parte, el fruto del amor y apoyo incondicional que me brindaron en cada etapa de mi formación.

A mi hermano por siempre motivarme a seguir adelante y darme su ejemplo de perseverancia.

A mi primo que a la par estábamos en la misma carrera y etapas de la vida, agradecido por ese sentimiento hermano que él y yo conocemos.

A mi tío que siempre me ofreció de su ayuda, Ingeniero en Informática con experiencia y mi referente.

A mi abuela siempre preocupada de que me vaya super bien y constantemente manifestar esa preocupación de madre.

A mí que un día pensé que no lo lograría, pero como todo en la vida con constancia y dedicación todo es posible.

Finalmente, a todas aquellas personas que, de una u otra forma, contribuyeron a mi crecimiento personal y profesional, permitiéndome culminar con éxito esta importante etapa de mi vida.

AGRADECIMIENTOS

A los docentes responsables de la formación técnica, teórica y profesional entregada en la Universidad Técnica Federico Santa María, me llevo valiosas experiencias junto a ellos, agradecer su compromiso y dedicación con los estudiantes.

A mi equipo de trabajo, por su dedicación, compromiso y esfuerzo por el proyecto. Cada uno con su grano de arena hicimos de esta instancia posible.

A los estudiantes quienes son protagonistas en este proyecto enfocado en la educación y su futuro.

A quienes de alguna u otra forma fueron de apoyo directo o indirecto para que el proyecto se hiciera posible.

RESUMEN

Esta memoria presenta el desarrollo del Back-End de Aurora, soporte tecnológico de una plataforma educativa gamificada orientada a estudiantes de sexto básico. La solución sustenta funcionalidades dirigidas a apoyar procesos específicos de autorregulación del aprendizaje, principalmente el monitoreo del propio desempeño y la autorreflexión, e incorpora el registro emocional y la gamificación como elementos complementarios. El trabajo corresponde a un subproyecto dentro de una iniciativa grupal y delimita su alcance a la infraestructura lógica y de persistencia encargada de articular las interfaces móvil y web, la base de datos y los servicios externos utilizados por la plataforma.

El Back-End fue implementado mediante FastAPI bajo una arquitectura monolítica modular. La solución integra una API REST, endpoints especializados para la aplicación móvil y vistas web renderizadas mediante Jinja2. La persistencia se gestiona mediante SQLAlchemy y MySQL, a partir de modelos relacionados con usuarios, roles, actividades educativas, evaluaciones, registros emocionales, desempeño, gamificación e interacciones con servicios de inteligencia artificial.

Entre los principales procesos implementados se encuentran la autenticación web mediante sesiones firmadas, la verificación de acceso móvil, el registro emocional previo y posterior a las actividades, el cálculo de puntajes, el seguimiento del progreso y el desbloqueo progresivo de unidades. También se implementó la asignación de Glims de acuerdo con la mejora respecto del mejor desempeño anterior del estudiante. Asimismo, el Back-End se integró con los componentes de inteligencia artificial desarrollados en el proyecto grupal, permitiendo gestionar y persistir la información utilizada por el chatbot, el análisis emocional y los resúmenes destinados al dashboard docente.

El funcionamiento de la solución se validó mediante pruebas funcionales y técnicas ejecutadas en un entorno local controlado. Estas pruebas permitieron verificar las principales reglas de negocio, las operaciones de persistencia y la comunicación entre las interfaces cliente, el servidor y la base de datos. La validación realizada se limitó al funcionamiento del Back-End y no contempló la medición de efectos sobre la autorregulación o el desempeño académico de los estudiantes.

Como resultado, se obtuvo un prototipo funcional e integrado capaz de centralizar las reglas de negocio y sostener los principales flujos de Aurora. El aporte de esta memoria consiste en proporcionar la base tecnológica que vincula los componentes educativos, emocionales y de gamificación de la plataforma desde la perspectiva del Back-End. Finalmente, se identificaron oportunidades de mejora relacionadas con la seguridad de las credenciales, la autenticación móvil, el almacenamiento de archivos, el despliegue productivo y la futura validación pedagógica de la solución.

Palabras clave—Back-End; Autorregulación del aprendizaje; FastAPI; Gamificación; Monitoreo del desempeño; Persistencia de datos.

ABSTRACT

Abstract—This thesis presents the development of Aurora’s Back-End, which provides the technological support for a gamified educational platform aimed at sixth-grade students. The solution supports functionalities intended to assist specific self-regulated learning processes, primarily the monitoring of students’ own performance and self-reflection, while incorporating emotional tracking and gamification as complementary elements. This work constitutes a subproject within a group initiative and limits its scope to the logical and persistence infrastructure responsible for connecting the mobile and web interfaces, the database, and the external services used by the platform.

The Back-End was implemented using FastAPI under a modular monolithic architecture. The solution integrates a REST API, specialized endpoints for the mobile application, and server-side rendered web views using Jinja2. Data persistence is managed through SQLAlchemy and MySQL, using models related to users, roles, educational activities, assessments, emotional records, performance, gamification, and interactions with artificial intelligence services.

The main implemented processes include web authentication through signed sessions, mobile access verification, emotional state recording before and after activities, score calculation, progress tracking, and the progressive unlocking of units. The allocation of Glims according to improvements over each student’s previous best performance was also implemented. Furthermore, the Back-End was integrated with the artificial intelligence components developed as part of the group project, enabling the management and persistence of the information used by the chatbot, emotional analysis, and summaries displayed on the teacher dashboard.

The solution was validated through functional and technical tests conducted in a controlled local environment. These tests verified the main business rules, persistence operations, and communication among the client interfaces, the server, and the database. The validation was limited to the operation of the Back-End and did not include measuring its effects on students’ self-regulation or academic performance.

As a result, a functional and integrated prototype was obtained, capable of centralizing business rules and supporting Aurora’s main workflows. The contribution of this thesis lies in providing the technological foundation that connects the platform’s educational, emotional, and gamification components from the Back-End perspective. Finally, opportunities for improvement were identified regarding credential security, mobile authentication, file storage, production deployment, and the future pedagogical validation of the solution.

Keywords—Back-End; Self-regulated learning; FastAPI; Gamification; Performance monitoring; Data persistence.

GLOSARIO

API: Application Programming Interface. Conjunto de reglas y puntos de acceso que permite la comunicación y el intercambio de datos entre diferentes componentes o aplicaciones de software.

Autorreflexión: Fase posterior a la ejecución de una actividad en la cual el estudiante evalúa sus resultados, formula juicios sobre su desempeño y genera reacciones que pueden influir en experiencias de aprendizaje posteriores.

Autorregulación del aprendizaje: Proceso cíclico mediante el cual el estudiante participa activamente en su aprendizaje a través de la anticipación, la ejecución y la autorreflexión, orientando sus acciones hacia el cumplimiento de objetivos académicos.

Back-End: Componente de una aplicación encargado de procesar las solicitudes de los clientes, aplicar las reglas de negocio, gestionar la persistencia de datos e integrar los servicios necesarios para el funcionamiento del sistema.

Endpoint: Punto de acceso específico de una API, asociado normalmente con una URL y un método HTTP, mediante el cual una aplicación cliente puede consultar o modificar un recurso, o ejecutar una operación del servidor.

FastAPI: Framework web para Python orientado al desarrollo de APIs. Utiliza anotaciones de tipo para apoyar la validación de datos y la generación automática de documentación, y permite implementar operaciones tanto síncronas como asíncronas.

Gamificación: Uso de elementos y mecánicas propios del diseño de juegos en contextos no lúdicos, como el educativo, con el propósito de apoyar la motivación y la participación de los usuarios.

Glims: Unidad virtual de recompensa utilizada en Aurora. Se asigna al estudiante cuando mejora su mejor desempeño anterior y puede emplearse para adquirir elementos disponibles dentro de la plataforma.

IA: Inteligencia artificial. Área de la informática dedicada al desarrollo de sistemas capaces de realizar tareas asociadas con capacidades humanas, como la generación de respuestas, el análisis de información y la identificación de patrones.

Jinja2: Motor de plantillas para Python utilizado para generar contenido HTML dinámico en el servidor a partir de datos proporcionados por la aplicación.

JSON: JavaScript Object Notation. Formato de texto ligero utilizado para representar e intercambiar datos estructurados entre aplicaciones, como un servidor y una interfaz web o móvil.

Monitoreo del propio desempeño: Proceso mediante el cual el estudiante observa y revisa sus resultados, avances y dificultades durante el desarrollo de actividades de aprendizaje.

MySQL: Sistema de gestión de bases de datos relacionales utilizado por Aurora para almacenar y administrar de manera persistente la información de la plataforma.

OpenAPI: Especificación estandarizada para describir la estructura, las operaciones, los parámetros y las respuestas de una API basada en HTTP mediante un formato legible por personas y herramientas de software.

ORM: Object-Relational Mapping. Técnica de programación que permite representar las tablas y relaciones de una base de datos mediante clases y objetos, facilitando la consulta y persistencia de información desde el código de la aplicación.

Pydantic: Librería de Python utilizada para definir estructuras y validar datos mediante anotaciones de tipo, permitiendo comprobar que la información recibida cumpla con los formatos esperados por la aplicación.

RBAC: Role-Based Access Control. Modelo de autorización en el que los permisos de acceso a las funcionalidades o recursos del sistema se determinan de acuerdo con los roles asignados a cada usuario.

REST: Representational State Transfer. Estilo arquitectónico utilizado para diseñar servicios web alrededor de recursos, empleando operaciones estandarizadas del protocolo HTTP para acceder a ellos y modificarlos.

Scrum: Marco de trabajo ágil e iterativo orientado a la gestión de proyectos complejos y a la entrega incremental de valor mediante ciclos breves de trabajo denominados sprints.

Sesión firmada: Mecanismo utilizado para mantener el estado de autenticación de un usuario mediante datos protegidos por una firma criptográfica que permite detectar modificaciones no autorizadas. La firma garantiza la autenticidad de los datos, pero no implica necesariamente su cifrado.

SQLAlchemy: Biblioteca de Python que proporciona herramientas para trabajar con bases de datos SQL e incluye un ORM para representar y manipular tablas, relaciones y registros mediante clases y objetos.

Swagger UI: Interfaz web interactiva que presenta la documentación de una API a partir de su especificación OpenAPI y permite consultar sus endpoints, parámetros, esquemas y respuestas, además de ejecutar solicitudes de prueba.

UTP: Unidad Técnico-Pedagógica. Área de los establecimientos educacionales encargada de planificar, coordinar y supervisar los procesos pedagógicos.

INDICE DE CONTENIDOS

RESUMEN	5
<i>ABSTRACT</i>	6
GLOSARIO	7
INDICE DE CONTENIDOS	9
INTRODUCCIÓN	13
CAPÍTULO 1: DEFINICIÓN DEL PROBLEMA	14
1.1 CONTEXTO GENERAL	14
1.2 SITUACIÓN ACTUAL	15
1.3 ACTORES INVOLUCRADOS	15
1.4 DIFICULTADES ACTUALES	17
1.5 CONSECUENCIAS DE NO INTERVENIR	17
1.6 ANÁLISIS DE LA COMPETENCIA	18
1.7 ALCANCE DEL SUBPROYECTO	19
1.8 PROBLEMA ESPECÍFICO	20
1.9 OBJETIVO GENERAL DEL SUBPROYECTO	20
1.10 OBJETIVOS ESPECÍFICOS	20
CAPÍTULO 2: MARCO CONCEPTUAL	21
2.1 Metodologías de desarrollo de software	21
2.2 Arquitectura Back-End	22
2.2.1 Arquitectura por capas	22
2.2.2 Modularidad y desacoplamiento	22
2.2.3 API REST y contratos de comunicación	23
2.2.4 Separación de responsabilidades	24
2.2.5 Aplicación en Aurora	24
2.3 FastAPI	25
2.3.1 Fundamentos de FastAPI	25
2.3.2 Ventajas técnicas	25
2.3.3 Comparación con otras alternativas	26
2.3.4 Aplicación en Aurora	26
2.4 Persistencia de datos	27
2.4.1 Bases de datos relacionales	27
2.4.2 ORM	27

2.4.3 SQLAlchemy	28
2.4.4 Relaciones e integridad	28
2.4.5 Aplicación en Aurora	28
2.5 Seguridad en sistemas Back-End	29
2.5.1 Autenticación, sesiones y autorización	29
2.5.2 Almacenamiento seguro de contraseñas	30
2.5.3 Roles y control de acceso	30
2.5.4 Validación de datos	31
2.5.5 Protección de endpoints	31
2.6 Inteligencia Artificial aplicada a la educación	32
2.6.1 IA generativa	32
2.6.2 Retroalimentación y apoyo contextual	32
2.6.3 Posibilidades y limitaciones	33
2.6.4 Aplicación en Aurora	33
2.7 Autorregulación escolar	34
2.7.1 Fundamentos teóricos	34
2.7.2 Autorregulación y aprendizaje autónomo	35
2.7.3 Aplicación en Aurora	35
2.8 Gamificación	36
2.8.1 Motivación y participación	37
2.8.2 Recompensas, progreso y engagement	37
2.8.3 Aplicación en Aurora	38
2.9 Cierre del capítulo	39
CAPÍTULO 3: PROPUESTA DE SOLUCIÓN	40
3.1 Introducción	40
3.2 Metodología de desarrollo aplicada	40
3.3 Arquitectura general del Back-End	41
3.4 Arquitectura lógica y separación de responsabilidades	43
3.4.1 Capa de rutas o presentación	43
3.4.2 Capa de servicios y funciones auxiliares	44
3.4.3 Capa de persistencia	44
3.5 Entorno de ejecución y stack tecnológico	45
3.6 Diseño de la base de datos	46
3.6.1 Decisiones principales de modelamiento	47
3.6.2 Integridad y restricciones	48

3.6.3 Uso de SQLAlchemy.....	48
3.6.4 Deuda técnica en la persistencia.....	49
3.7 Flujo de autenticación y control de acceso	49
3.7.1 Autenticación web mediante sesión firmada	49
3.7.2 Mecanismo de acceso móvil.....	50
3.7.3 Limitaciones de seguridad actuales.....	50
3.8 Flujo de autorregulación y evaluación.....	50
3.9 Algoritmo de cálculo de puntaje	53
3.9.1 Preguntas de alternativa	53
3.9.2 Preguntas de desarrollo	53
3.9.3 Preguntas de clasificación y minijuegos	54
3.10 Desbloqueo progresivo de unidades	54
3.11 Gamificación y economía de Glims	55
3.12 Integración del Back-End con el módulo de inteligencia artificial	56
3.12.1 Integración con el chatbot.....	57
3.12.2 Integración con el registro emocional.....	57
3.12.3 Integración con el dashboard docente.....	57
3.13 Gestión de archivos y seguridad de rutas.....	57
3.14 Gestión de sesiones de base de datos y consistencia transaccional	58
3.15 Documentación automática mediante Swagger y OpenAPI	59
3.16 Mantenibilidad, capacidad de evolución y limitaciones actuales	59
3.17 Cierre del capítulo	61
CAPÍTULO 4: VALIDACIÓN Y RESULTADOS DEL BACK-END DE AURORA	62
4.1 Estrategia de validación	62
4.2 Antecedente diagnóstico	65
4.3 Validación exploratoria en contexto escolar	65
4.4 Presentación en feria tecnológica	67
4.5 Validación funcional del Back-End	69
4.5.1 Autenticación y control de acceso.....	69
4.5.2 Navegación académica	72
4.5.3 Gestión de materiales didácticos	74
4.6 Validación del flujo de autorregulación y reglas de negocio	76
4.6.1 Registro emocional previo.....	77
4.6.2 Finalización de la evaluación y cálculo del puntaje	79
4.6.3 Actualización del desempeño y asignación de Glims.....	81

4.6.4 Prevención de recompensas duplicadas	83
4.6.5 Desbloqueo progresivo de unidades	86
4.6.6 Registro emocional posterior	88
4.7 Validación de integraciones externas y manejo de errores	89
4.7.1 Manejo de indisponibilidad del servicio de inteligencia artificial	90
4.7.2 Validación de rutas de descarga.....	92
4.7.3 Manejo de solicitudes a entidades inexistentes.....	94
CAPÍTULO 5: CONCLUSIONES Y RECOMENDACIONES	96
5.1 Discusión de resultados y cumplimiento de objetivos.....	96
5.2 Contribuciones e impacto del trabajo realizado	101
5.3 Alcances, limitaciones y aprendizajes.....	103
5.4 Recomendaciones y trabajo futuro	105
REFERENCIAS BIBLIOGRÁFICAS	109
Anexo A. MATRIZ CONSOLIDADA DE CASOS DE PRUEBA	111
Anexo B. EVIDENCIAS COMPLEMENTARIAS DE VALIDACIÓN	115
B.1 Objetivo y alcance	115
B.2 Verificación de la comunicación entre FastAPI y MySQL	115
B.3 Evidencias de navegación académica	116
B.4 Estado anterior al registro del Pre-check	119
B.5 Estado anterior a la finalización de la evaluación	119
B.6 Condiciones del desbloqueo progresivo	120
B.7 Estado anterior a la interacción con el chatbot	121
Anexo C. INVENTARIO COMPLETO DE EVIDENCIAS	122
C.1 Objetivo y alcance.....	122
C.2 Inventario consolidado.....	122
C.3 Consideraciones de trazabilidad	125

INTRODUCCIÓN

Durante las últimas décadas, la educación ha respondido a desafíos que trascienden la transmisión de conocimientos disciplinares. Entre ellos se encuentra la autorregulación del aprendizaje, concebida por Zimmerman (2002) como un proceso cíclico de anticipación, ejecución y autorreflexión. En este ciclo, el monitoreo del propio desempeño aporta información para la autorreflexión.

El apoyo a estos procesos en el entorno escolar requiere el acompañamiento de docentes, apoderados y equipos de apoyo, además de mecanismos para registrar el progreso, relacionar el desempeño con el estado emocional de los estudiantes y dar continuidad a las actividades realizadas. Esta información puede favorecer una observación más sistemática de la experiencia de aprendizaje.

Desde la Ingeniería en Informática, las tecnologías educativas permiten estructurar actividades, conservar información histórica y facilitar la interacción entre actores del entorno escolar. No obstante, una plataforma digital no produce por sí misma una mejora en la autorregulación del aprendizaje, sino que debe actuar como un apoyo dentro de un proceso pedagógico más amplio.

En este contexto surge Aurora, un proyecto grupal que desarrolla una plataforma educativa gamificada para estudiantes de sexto básico. Sus subproyectos integran actividades de aprendizaje, registros emocionales, seguimiento del desempeño e inteligencia artificial, mientras la gamificación actúa como apoyo motivacional. La plataforma busca apoyar el monitoreo del propio desempeño y la autorreflexión, sin sustituir la función pedagógica del docente ni abarcar toda la autorregulación del aprendizaje.

La memoria aborda el diseño y la implementación del Back-End de Aurora. Este subproyecto comprende la infraestructura encargada de recibir solicitudes, validar información, ejecutar reglas de negocio, gestionar la persistencia y conectar las interfaces web y móvil con servicios externos de la plataforma.

La solución se implementó mediante FastAPI bajo una arquitectura monolítica modular, utilizando SQLAlchemy y MySQL para la persistencia de datos. Entre los procesos soportados se encuentran la gestión de usuarios, las actividades educativas, el registro emocional, el cálculo y seguimiento del desempeño y la gamificación. El Back-End proporciona mecanismos de comunicación y persistencia para integrarse con componentes de inteligencia artificial desarrollados por otro subproyecto del equipo.

El documento se organiza en cinco capítulos. El primero presenta el problema y el alcance del subproyecto; el segundo desarrolla el marco conceptual; el tercero describe la solución; el cuarto aborda su validación técnica y funcional; y el quinto expone las conclusiones, limitaciones y recomendaciones futuras.

CAPÍTULO 1: DEFINICIÓN DEL PROBLEMA

1.1 CONTEXTO GENERAL

La autorregulación del aprendizaje corresponde a un proceso mediante el cual los estudiantes participan activamente en su aprendizaje, orientando sus pensamientos, emociones y acciones hacia objetivos académicos. Zimmerman (2002) representa este proceso mediante un modelo cíclico compuesto por las fases de anticipación, ejecución y autorreflexión. Estas fases comprenden, entre otros procesos, el establecimiento de metas, la planificación de estrategias, el monitoreo del propio desempeño, la autoevaluación y las reacciones posteriores a los resultados obtenidos.

En el contexto educativo chileno, las Bases Curriculares de primero a sexto básico reconocen la autonomía como parte de la formación de los estudiantes y establecen Objetivos de Aprendizaje Transversales asociados con el desarrollo personal y la adquisición progresiva de responsabilidades (Ministerio de Educación, 2012). Este antecedente permite justificar la relevancia educativa de apoyar gradualmente la observación y evaluación del propio aprendizaje, pero no demuestra que todos los estudiantes presenten dificultades de autorregulación ni que estas puedan resolverse exclusivamente mediante una herramienta tecnológica.

Aurora se orienta específicamente a estudiantes de sexto básico. Considerando la edad oficial de ingreso al sistema escolar y una trayectoria educativa regular, este nivel suele comprender estudiantes de aproximadamente 11 y 12 años. Dicho rango se encuentra dentro de la adolescencia temprana, etapa situada por UNICEF (2011) entre los 10 y 14 años y caracterizada por transformaciones físicas, cognitivas, sociales y emocionales. Las trayectorias individuales pueden variar, por lo que esta clasificación constituye una referencia general y no una caracterización uniforme de todos los estudiantes.

La selección de sexto básico también posee pertinencia dentro del contexto educativo chileno. En 2024, los estudiantes de este nivel rindieron las evaluaciones Simce de Lectura y Matemática, además de los Cuestionarios de Calidad y Contexto de la Educación (Agencia de Calidad de la Educación, 2024). Este antecedente permite reconocer la relevancia nacional del nivel y la importancia de disponer de información sobre sus procesos educativos. Sin embargo, Aurora no fue diseñada como una herramienta de preparación para el Simce ni fue validada mediante sus resultados.

En este marco surge Aurora, un proyecto grupal que integra actividades educativas, seguimiento del desempeño, registros emocionales, gamificación e inteligencia artificial. Su alcance pedagógico no comprende la autorregulación del aprendizaje en toda su amplitud. La plataforma busca apoyar principalmente el monitoreo del propio desempeño y la autorreflexión, incorporando la conciencia emocional como elemento complementario y la gamificación como apoyo motivacional para sostener la participación.

1.2 SITUACIÓN ACTUAL

El Currículum Nacional plantea la autonomía como un propósito progresivo de la Educación Básica y reconoce la importancia de que los estudiantes participen activamente en su formación. Asimismo, la normativa chilena define la evaluación como un conjunto de acciones orientadas a obtener e interpretar información sobre el aprendizaje, con el propósito de adoptar decisiones y retroalimentar los procesos de enseñanza (Ministerio de Educación, 2018).

Desde esta perspectiva, registrar resultados, conservar información histórica y entregar retroalimentación puede contribuir a que estudiantes y docentes dispongan de antecedentes para observar el progreso. No obstante, la disponibilidad de información no produce automáticamente autorregulación. De acuerdo con el modelo de Zimmerman (2002), estos procesos requieren la participación activa del estudiante y se relacionan con factores personales, conductuales y contextuales.

Por esta razón, el punto de partida de Aurora no debe interpretarse como la existencia demostrada de una carencia generalizada de autorregulación en el sistema escolar chileno. La evidencia local disponible corresponde a un levantamiento exploratorio sobre la percepción de utilidad y la experiencia previa con aplicaciones de estudio. Dicho instrumento no evaluó directamente la capacidad de establecer metas, planificar estrategias, gestionar el tiempo, mantener la concentración o regular el comportamiento.

El problema abordado consiste, por una parte, en explorar el uso de una herramienta digital que permita apoyar el monitoreo del desempeño y la reflexión posterior a las actividades. Por otra, desde la Ingeniería en Informática, se identifica la necesidad de contar con una infraestructura capaz de mantener y relacionar la información generada por los componentes educativos, emocionales y de gamificación de Aurora.

La plataforma actúa, por tanto, como un medio tecnológico de apoyo y no como sustituto del acompañamiento pedagógico. La interpretación de los resultados, la orientación del estudiante y la adaptación de las estrategias educativas continúan dependiendo del criterio de docentes y demás actores responsables del proceso formativo.

1.3 ACTORES INVOLUCRADOS

Los principales usuarios considerados por Aurora son los estudiantes de sexto básico. Mediante la aplicación móvil pueden desarrollar actividades, responder evaluaciones, registrar su estado emocional antes y después de una sesión, consultar resultados y observar su progreso. Estas funcionalidades se relacionan principalmente con el monitoreo del propio desempeño y la reflexión posterior a la actividad, sin incorporar de manera directa la definición de metas personales ni la planificación de estrategias de estudio.

Los docentes constituyen otro actor relevante, debido a que la plataforma web les permite acceder a información relacionada con las actividades, el desempeño y los registros emocionales de los estudiantes. Esta información puede apoyar el seguimiento

pedagógico, pero los reportes generados por Aurora no corresponden a diagnósticos psicológicos ni reemplazan la evaluación profesional del docente.

Los equipos de Unidad Técnico-Pedagógica se consideran actores institucionales potencialmente interesados en el seguimiento de los procesos educativos. Sin embargo, el prototipo no fue validado específicamente con integrantes de UTP. Por esta razón, no se atribuyen a estos equipos necesidades o valoraciones que no hayan sido comprobadas mediante el levantamiento realizado.

También participan los integrantes de los distintos subproyectos de Aurora, responsables de la aplicación móvil, la plataforma web, los componentes de inteligencia artificial, el contenido y la infraestructura Back-End. La operación integrada de la solución depende de la comunicación entre estos componentes y del intercambio coherente de información.

Como parte del levantamiento inicial se aplicó una encuesta exploratoria a 40 estudiantes para conocer su percepción sobre aplicaciones orientadas al estudio. Los resultados relevantes se presentan en la Figura 1.1.

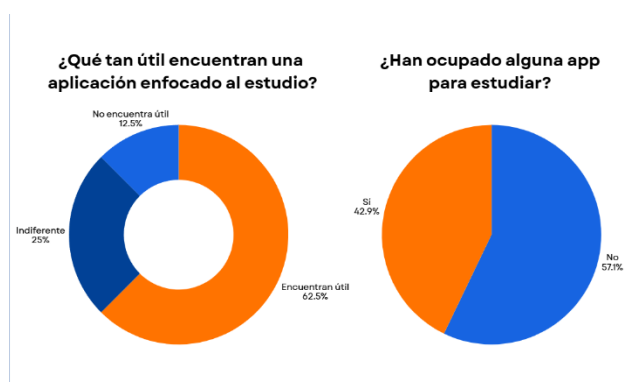


Figura 1.1. Resultados de encuesta diagnóstica aplicada a estudiantes sobre percepción de utilidad y uso previo de aplicaciones de estudio. Fuente: Elaboración propia a partir de formulario aplicado mediante Google Forms.

Figura 1.1. Resultados de encuesta diagnóstica aplicada a estudiantes sobre percepción de utilidad y uso previo de aplicaciones de estudio. Fuente: Elaboración propia a partir de un formulario aplicado mediante Google Forms.

En la pregunta relacionada con la utilidad percibida, el 62,5 % de los 40 participantes consideró útil una aplicación enfocada en el estudio, el 25 % manifestó una posición indiferente y el 12,5 % no la consideró útil. Estos resultados muestran una disposición mayoritariamente favorable dentro del grupo consultado, pero no permiten concluir que una aplicación mejore la autorregulación ni que sus resultados puedan generalizarse a la totalidad de los estudiantes de sexto básico.

1.4 DIFICULTADES ACTUALES

A partir de la evidencia disponible no resulta posible afirmar que todos los estudiantes presentan problemas de concentración, motivación, planificación o gestión del tiempo. El diagnóstico aplicado tampoco permite atribuir estas dificultades al conjunto del sistema educativo chileno. Por esta razón, la problemática se delimita a los procesos que Aurora efectivamente puede apoyar y a las necesidades técnicas que surgen de su implementación.

Desde la perspectiva educativa, una de las dificultades abordadas consiste en presentar de manera organizada información sobre las actividades, los puntajes y el progreso del estudiante. Dentro del modelo de Zimmerman (2002), la observación del propio desempeño forma parte de la fase de ejecución y proporciona antecedentes que pueden utilizarse posteriormente en la autorreflexión. Aurora apoya parcialmente estos procesos al registrar resultados, comparar desempeños y entregar retroalimentación, pero no enseña directamente estrategias de estudio ni incorpora mecanismos para establecer metas personales.

Desde la perspectiva técnica, la aplicación móvil, la plataforma web, la base de datos y los servicios externos requieren acceder a información compartida. Sin una infraestructura común, las reglas relacionadas con evaluaciones, puntajes, progreso, registros emocionales y recompensas podrían quedar distribuidas entre diferentes componentes, dificultando la consistencia de los datos y la trazabilidad de las operaciones.

También resulta necesario relacionar información de distinta naturaleza dentro de un mismo flujo. Una sesión educativa puede incluir la identificación del estudiante, el registro emocional previo, las respuestas entregadas, el cálculo del puntaje, la actualización del progreso, la asignación de recompensas y el registro emocional posterior. La coordinación de estas operaciones constituye una dificultad de ingeniería que debe resolverse en el servidor.

1.5 CONSECUENCIAS DE NO INTERVENIR

Si Aurora no contara con una infraestructura Back-End común, sus interfaces no tendrían de un mecanismo centralizado para consultar y modificar la información. Esto podría provocar duplicación de reglas, diferencias en el tratamiento de los datos y dificultades para conservar el estado de los usuarios entre distintas sesiones.

La ausencia de persistencia impediría mantener un historial consistente de respuestas, resultados, estados emocionales, progreso y recompensas. Como consecuencia, la plataforma no podría presentar al estudiante una visión acumulada de su desempeño ni proporcionar al docente información consolidada para su seguimiento.

Asimismo, procesos como la mejora respecto del mejor puntaje anterior, la asignación de Glims y el desbloqueo progresivo de unidades requieren comparar información actual con registros previamente almacenados. Sin un componente encargado de gestionar estas reglas, su ejecución dependería de cada interfaz cliente y aumentaría el riesgo de comportamientos inconsistentes.

Estas consecuencias se limitan al funcionamiento de Aurora y a su capacidad de proporcionar información de apoyo. No se plantea que la ausencia de la plataforma provoque por sí misma problemas académicos, emocionales o profesionales, ni que su implementación garantice una mejora en la autorregulación del aprendizaje.

1.6 ANÁLISIS DE LA COMPETENCIA

Para contextualizar la propuesta se realizó una revisión documental de la información pública disponible en los sitios oficiales de Khan Academy, Smartick, Colegium, Napsis y Lirmi. Esta revisión tuvo un carácter exploratorio y no correspondió a una evaluación experimental de usabilidad, efectividad o impacto educativo.

Khan Academy proporciona videos educativos, ejercicios prácticos, recursos personalizados y paneles que permiten a estudiantes y docentes consultar información sobre el avance en diferentes áreas de aprendizaje (Khan Academy, s. f.). Smartick se concentra principalmente en el aprendizaje personalizado de Matemática mediante sesiones breves, adaptación de ejercicios, retroalimentación inmediata, reportes de progreso y elementos gamificados (Smartick, s. f.).

Colegium, Napsis y Lirmi presentan un enfoque principalmente relacionado con la gestión escolar y académica. Colegium incluye funcionalidades para gestionar asistencia, calificaciones, finanzas y comunicación institucional (Colegium, s. f.). Napsis ofrece herramientas como libro de clases digital, comunicación con apoderados, gestión pedagógica y acceso a información académica (Napsis, s. f.). Lirmi incorpora evaluación, control escolar, registro de asistencia y calificaciones, además de herramientas de comunicación y gestión institucional (Lirmi, s. f.).

La revisión permite reconocer que algunas de estas plataformas incluyen seguimiento del progreso, personalización, recompensas o gestión académica. Por lo tanto, tales funcionalidades no deben presentarse individualmente como exclusivas de Aurora. La particularidad del prototipo se encuentra en la articulación de un flujo que combina el registro emocional previo y posterior, las actividades educativas, la comparación con el mejor desempeño anterior, la asignación de Glims, el desbloqueo progresivo y la integración con componentes de inteligencia artificial.

Dentro de las plataformas seleccionadas y de acuerdo con la información pública revisada, no se identificó una descripción equivalente de esta combinación específica. Esta observación debe entenderse como una inferencia limitada a la muestra analizada y no como prueba de que no exista otra plataforma con características similares.

La diferenciación de Aurora tampoco constituye evidencia de superioridad ni demuestra un mayor impacto educativo. Desde la perspectiva del subproyecto, su valor se encuentra en el desafío técnico de relacionar usuarios, actividades, resultados, registros emocionales, recompensas y servicios externos dentro de una infraestructura Back-End común.

1.7 ALCANCE DEL SUBPROYECTO

Aurora corresponde a un proyecto grupal compuesto por subproyectos responsables de las interfaces, el contenido educativo, la experiencia de usuario, los componentes de inteligencia artificial y la infraestructura lógica. La presente memoria se concentra en el desarrollo del Back-End.

El alcance comprende la implementación del servidor encargado de recibir solicitudes desde la aplicación móvil y la interfaz web, validar la estructura de la información, ejecutar reglas de negocio, gestionar la persistencia y coordinar la comunicación con servicios externos. La solución combina una API REST, endpoints especializados para la aplicación móvil y vistas web renderizadas en el servidor.

Dentro del subproyecto se incluyen los siguientes procesos:

- gestión de personas, roles y contexto institucional;
- autenticación web mediante sesiones firmadas;
- verificación de acceso para la aplicación móvil;
- administración de unidades, sesiones, preguntas y alternativas;
- registro emocional previo y posterior a las actividades;
- cálculo de puntajes y actualización del desempeño;
- seguimiento del progreso y desbloqueo progresivo de unidades;
- administración de Glims y compra de elementos;
- integración técnica con los componentes de inteligencia artificial;
- recepción y recuperación de archivos dentro del entorno implementado; y
- generación automática de documentación de la API mediante OpenAPI y Swagger UI.

El subproyecto no comprende el desarrollo completo de la aplicación móvil en Flutter, el diseño visual de todas las interfaces, la creación del contenido pedagógico ni la implementación interna de los componentes de inteligencia artificial. Tampoco incluye una preparación integral para producción, debido a que permanecen aspectos de seguridad, autenticación móvil, almacenamiento de archivos, despliegue y operación que requieren fortalecimiento.

Desde la perspectiva educativa, el alcance se limita al soporte tecnológico de funcionalidades relacionadas principalmente con el monitoreo del desempeño y la autorreflexión. Aurora no incorpora mecanismos completos para establecer metas personales, planificar estrategias, organizar el tiempo o regular el entorno de estudio. En consecuencia, no cubre la autorregulación del aprendizaje en toda la amplitud propuesta por Zimmerman.

Las pruebas realizadas corresponden a una validación técnica y funcional del Back-End. No se evaluó experimentalmente la usabilidad de la plataforma con estudiantes ni su efecto sobre la autorregulación, la motivación, el bienestar emocional, el rendimiento académico o los resultados Simce.

1.8 PROBLEMA ESPECÍFICO

La operación de Aurora requiere una infraestructura Back-End capaz de coordinar los componentes que participan en la plataforma. La aplicación móvil y la interfaz web necesitan acceder a información compartida, ejecutar las mismas reglas de negocio y conservar el estado de los usuarios entre distintas interacciones.

El problema específico abordado corresponde a la necesidad de desarrollar una estructura lógica y persistente que permita articular personas, roles, establecimientos, actividades educativas, evaluaciones, registros emocionales, desempeño, recompensas e interacciones con servicios de inteligencia artificial.

Esta necesidad presenta diferentes desafíos de ingeniería. El servidor debe atender clientes con mecanismos de comunicación distintos, validar los datos recibidos, mantener relaciones coherentes en la base de datos y centralizar las reglas compartidas. También debe coordinar operaciones como el cálculo de puntajes, la comparación con resultados anteriores, la asignación de Glims y el desbloqueo de unidades.

La integración de registros emocionales e inteligencia artificial requiere, además, delimitar el uso de la información generada. Las respuestas y los resúmenes proporcionados por estos componentes se utilizan como apoyo dentro de la plataforma y no como diagnósticos psicológicos ni como sustitutos del criterio docente.

Desde esta perspectiva, el subproyecto no busca resolver autónomamente las dificultades educativas de los estudiantes. Su propósito es proporcionar la infraestructura tecnológica necesaria para que los componentes de Aurora operen de manera integrada y sostengan funcionalidades orientadas al monitoreo del propio desempeño y la autorreflexión.

1.9 OBJETIVO GENERAL DEL SUBPROYECTO

Desarrollar la infraestructura Back-End de Aurora como soporte tecnológico de una plataforma educativa gamificada para el monitoreo del propio desempeño y la autorreflexión de estudiantes de sexto básico.

1.10 OBJETIVOS ESPECÍFICOS

1. Diseñar una arquitectura Back-End modular con separación de responsabilidades entre servicios, lógica de negocio, persistencia de datos e integración con las interfaces cliente.
2. Implementar los componentes Back-End para autenticación, gestión académica, evaluación y seguimiento del desempeño, registro emocional, gamificación e integración con servicios de inteligencia artificial.
3. Validar el funcionamiento de los principales flujos del Back-End mediante pruebas funcionales y técnicas en un entorno local controlado.

CAPÍTULO 2: MARCO CONCEPTUAL

El desarrollo del Back-End de Aurora requiere una base conceptual que permita comprender las decisiones metodológicas, arquitectónicas y tecnológicas adoptadas durante el subproyecto. Por esta razón, el capítulo presenta los fundamentos relacionados con la metodología de desarrollo, la arquitectura del servidor, la construcción de APIs, la persistencia de datos, la seguridad y la integración de servicios de inteligencia artificial.

La revisión no se limita a enumerar las herramientas utilizadas. Cada concepto se relaciona con su aplicación concreta en Aurora, distinguiendo entre los fundamentos generales, las decisiones efectivamente implementadas y las limitaciones identificadas. Esta distinción resulta necesaria para mantener el alcance de la memoria en el Back-End y evitar atribuir al prototipo características que no fueron desarrolladas o validadas.

2.1 Metodologías de desarrollo de software

Las metodologías de desarrollo permiten organizar las actividades necesarias para construir, integrar y revisar un sistema de software. En proyectos cuyos requerimientos evolucionan durante la implementación, los enfoques iterativos permiten dividir el trabajo en incrementos y ajustar la planificación a partir de los avances y dificultades observados.

Scrum es un marco de trabajo utilizado para abordar problemas complejos mediante ciclos iterativos. Se fundamenta en los principios de transparencia, inspección y adaptación, e incorpora periodos breves de trabajo denominados sprints, al término de los cuales se obtiene y revisa un incremento del producto (Schwaber & Sutherland, 2020).

De acuerdo con la metodología declarada para el proyecto grupal, el desarrollo de Aurora se organizó mediante iteraciones y funcionalidades progresivas. Esta organización resultó pertinente debido a que la plataforma fue construida por diferentes subproyectos responsables de las interfaces, el contenido educativo, los componentes de inteligencia artificial y la infraestructura Back-End.

En el caso del Back-End, el trabajo se dividió en procesos como la gestión de usuarios, las actividades educativas, las evaluaciones, los registros emocionales, el seguimiento del desempeño, la gamificación y la integración con servicios externos. El desarrollo gradual permitió coordinar estos procesos con las necesidades de las interfaces web y móvil.

La metodología también se relacionó con la organización modular del código. Las funcionalidades fueron incorporadas mediante routers, modelos y módulos auxiliares asociados con distintos dominios del sistema. No obstante, esta memoria no evalúa comparativamente la eficacia de Scrum ni mide su efecto sobre la productividad del equipo; su descripción se limita al marco utilizado para organizar el desarrollo.

2.2 Arquitectura Back-End

El Back-End corresponde al componente encargado de recibir solicitudes, validar información, ejecutar reglas de negocio, administrar la persistencia y proporcionar servicios a las interfaces cliente. En Aurora, esta función incluye el procesamiento de información relacionada con personas, roles, actividades educativas, evaluaciones, desempeño, registros emocionales y recompensas.

El diseño de un Back-End puede apoyarse en principios como la modularidad, la separación de responsabilidades y la organización por capas. Estos principios buscan distribuir las funciones del sistema entre componentes identificables, facilitando la comprensión y modificación del código.

En Aurora, la arquitectura adquiere especial relevancia porque un mismo servidor debe sostener la aplicación móvil desarrollada en Flutter y la plataforma web renderizada mediante Jinja2. Además, expone endpoints HTTP para consultar y modificar recursos. Por consiguiente, el Back-End debe mantener reglas coherentes para las distintas formas de interacción con la plataforma.

2.2.1 Arquitectura por capas

La arquitectura por capas organiza un sistema en niveles con responsabilidades diferenciadas. Una división habitual distingue entre la recepción y presentación de las solicitudes, la aplicación de reglas de negocio y el acceso a los datos.

La capa de presentación recibe las solicitudes y construye las respuestas entregadas a los clientes. La lógica de negocio procesa las reglas propias del dominio, como el cálculo de puntajes, la actualización del desempeño o el desbloqueo de unidades. La persistencia, por su parte, permite consultar y modificar la información almacenada en la base de datos.

Esta separación busca evitar que una misma unidad de código concentre la recepción de la solicitud, las validaciones, los cálculos y todas las operaciones sobre la base de datos. Sin embargo, la existencia de routers, modelos y funciones auxiliares no demuestra por sí sola una arquitectura por capas estricta. Para considerarla como tal, las dependencias y responsabilidades deben mantenerse separadas de manera consistente.

En Aurora se observa una separación parcial de estas responsabilidades. Los routers coordinan las solicitudes, los modelos representan la información persistente y determinados módulos auxiliares concentran reglas compartidas. No obstante, algunas rutas también ejecutan directamente consultas o reglas de negocio. Por ello, la implementación se describe con mayor precisión como un monolito modular con separación parcial por responsabilidades.

2.2.2 Modularidad y desacoplamiento

La modularidad consiste en dividir un sistema en unidades funcionales relacionadas con responsabilidades específicas. En un Back-End puede materializarse mediante routers, modelos, esquemas, servicios y funciones auxiliares.

El desacoplamiento busca reducir las dependencias innecesarias entre estas unidades. Cuando una regla de negocio se encuentra aislada de la ruta que recibe la solicitud, puede reutilizarse desde diferentes puntos del sistema y modificarse sin alterar todos sus consumidores.

En Aurora, los routers agrupan operaciones correspondientes a dominios como personas, academia, emociones, autenticación, tienda, minijuegos, dashboard docente e inteligencia artificial. Esta estructura permite localizar las rutas asociadas con cada área y evita concentrar todos los endpoints en un único archivo.

La modularidad implementada no implica que los componentes sean servicios independientes. Todos los routers, modelos, vistas y funciones auxiliares se ejecutan dentro de un mismo proceso FastAPI y comparten la conexión con la base de datos. En consecuencia, Aurora corresponde a una arquitectura monolítica modular y no a una arquitectura de microservicios.

2.2.3 API REST y contratos de comunicación

REST es un estilo arquitectónico para sistemas distribuidos basado en recursos, representaciones y una interfaz uniforme. En servicios HTTP, suele materializarse mediante URLs asociadas con recursos y operaciones ejecutadas con métodos como GET, POST, PUT, PATCH y DELETE (Fielding, 2000).

Una API define los puntos de acceso mediante los cuales un cliente se comunica con el servidor. Cada endpoint establece qué información recibe, qué operación realiza y qué respuesta puede entregar. Estos contratos son relevantes cuando diferentes interfaces dependen de un mismo Back-End, porque reducen la ambigüedad en el intercambio de datos.

OpenAPI proporciona una especificación estandarizada para describir APIs HTTP. Mediante ella pueden documentarse rutas, parámetros, cuerpos de solicitud, esquemas y respuestas. FastAPI genera esta especificación a partir de las rutas y modelos declarados en la aplicación, y la presenta mediante interfaces como Swagger UI (OpenAPI Initiative, 2024).

En Aurora se implementaron endpoints HTTP orientados a recursos y operaciones de la plataforma. La aplicación móvil consume rutas especializadas bajo el prefijo correspondiente al módulo móvil, mientras otros routers proporcionan operaciones sobre personas, actividades, emociones y elementos académicos.

No todas las interacciones del sistema corresponden estrictamente a una API REST. La plataforma web también utiliza formularios, sesiones y vistas HTML renderizadas desde el servidor. Por tanto, resulta más preciso señalar que Aurora combina una API REST con rutas web renderizadas en el servidor, sin afirmar que todo el Back-End cumple de manera estricta todas las restricciones arquitectónicas definidas por REST.

2.2.4 Separación de responsabilidades

La separación de responsabilidades busca asignar funciones diferentes a componentes distintos del sistema. En un Back-End, este principio permite diferenciar la recepción de solicitudes, la validación de datos, la aplicación de reglas de negocio y la persistencia.

Por ejemplo, el cálculo del puntaje de una evaluación puede mantenerse separado de la ruta que recibe las respuestas. De esta manera, la regla puede reutilizarse y revisarse sin depender completamente del mecanismo mediante el cual el cliente envía la información.

En Aurora, esta separación se materializa parcialmente mediante routers especializados, modelos de persistencia y módulos auxiliares. Entre estos últimos se encuentran archivos como `helpers_evaluacion.py`, `helpers_resultados.py` y `helpers_web.py`, que reúnen operaciones relacionadas con evaluaciones, resultados y construcción del contexto utilizado por las vistas web.

Esta organización contribuye a disminuir la duplicación entre rutas. Sin embargo, no todas las operaciones siguen una separación uniforme, debido a que algunos routers todavía contienen consultas y reglas propias del dominio. Esta situación se reconoce como parte de la implementación real y como una oportunidad de mejora de la estructura interna.

2.2.5 Aplicación en Aurora

El Back-End de Aurora se implementó como un monolito modular ejecutado en un único proceso FastAPI. Dentro de este proceso coexisten tres tipos principales de rutas: endpoints JSON de propósito general, endpoints especializados para la aplicación móvil y rutas web que generan vistas HTML mediante Jinja2.

Estas rutas no representan tres clientes independientes. Las interfaces principales son la aplicación móvil y la plataforma web, mientras los endpoints constituyen mecanismos de comunicación proporcionados por el servidor.

La decisión de utilizar un único proceso se ajusta al carácter de prototipo académico de la solución y permite compartir modelos, reglas de negocio y conexiones con la base de datos. No se realizó una comparación experimental con una arquitectura distribuida ni se midieron diferencias de rendimiento o mantenibilidad frente a microservicios.

Los routers separan los dominios funcionales, mientras determinados helpers concentran operaciones reutilizadas por diferentes flujos. Esta organización sostiene procesos como el cálculo de puntajes, la actualización del desempeño, el desbloqueo de unidades, la asignación de Glims y la preparación de información para las vistas web.

En síntesis, la arquitectura de Aurora busca centralizar las reglas compartidas por las interfaces web y móvil dentro de una estructura modular. Su finalidad no es distribuir el sistema entre servicios independientes, sino proporcionar una base comprensible y extensible para integrar los componentes del proyecto.

2.3 FastAPI

FastAPI es un framework web para Python orientado principalmente a la construcción de APIs. Utiliza anotaciones de tipo para interpretar los datos esperados por las rutas, apoyar su validación y generar documentación basada en OpenAPI. Su funcionamiento se construye sobre componentes como Starlette y Pydantic (FastAPI, s. f.).

FastAPI implementa el estándar ASGI, que define una interfaz entre aplicaciones web Python y servidores compatibles. Para ejecutar una aplicación FastAPI se requiere un servidor ASGI, como Uvicorn. El framework permite definir operaciones síncronas y asíncronas, aunque el uso de ASGI no significa que todas las operaciones de una aplicación sean automáticamente asíncronas.

2.3.1 Fundamentos de FastAPI

Uno de los fundamentos de FastAPI es el uso de anotaciones de tipo de Python. Estas anotaciones permiten declarar los parámetros, cuerpos y estructuras de datos esperados por un endpoint. FastAPI puede utilizar esa información para comprobar solicitudes y generar esquemas de documentación.

La validación de estructuras complejas se apoya en Pydantic. Mediante modelos se pueden declarar campos obligatorios, tipos y restricciones. Si una solicitud no cumple con la estructura declarada, el framework genera una respuesta de error antes de ejecutar la operación asociada.

Esta validación comprueba que los datos se ajusten a la estructura definida, pero no reemplaza las reglas de negocio. Por ejemplo, un identificador puede tener el tipo correcto y aun así referirse a un registro inexistente, o un puntaje numérico puede encontrarse fuera de las condiciones válidas para una actividad. Estas comprobaciones deben ser realizadas por la aplicación.

FastAPI genera una especificación OpenAPI a partir de las rutas registradas. Esta especificación puede consultarse mediante Swagger UI, facilitando la inspección y ejecución de solicitudes durante el desarrollo.

2.3.2 Ventajas técnicas

Entre las capacidades de FastAPI se encuentran la declaración de rutas, la validación estructural de datos, la inyección de dependencias, la generación de documentación y la compatibilidad con middleware ASGI.

La documentación automática resultó especialmente útil para Aurora, porque permitió inspeccionar los endpoints y ejecutar solicitudes durante el desarrollo y la validación funcional. Sin embargo, la existencia de documentación automática no garantiza por sí sola que todos los contratos estén completos; su precisión depende de que los parámetros, modelos y respuestas hayan sido declarados adecuadamente.

FastAPI también permite combinar respuestas JSON con plantillas y middleware. Esta flexibilidad permitió integrar en un mismo servidor los endpoints consumidos por la aplicación móvil y las vistas web renderizadas con Jinja2.

Aunque la documentación oficial destaca el rendimiento del framework, en Aurora no se realizaron pruebas comparativas de carga. Por esta razón, la elección de FastAPI no debe justificarse mediante resultados de rendimiento obtenidos por el subproyecto, sino mediante su adecuación funcional, su documentación automática y su compatibilidad con la estructura requerida.

2.3.3 Comparación con otras alternativas

Dentro del ecosistema Python existen alternativas como Django y Flask. Django proporciona un conjunto integrado de herramientas para desarrollar aplicaciones web, incluyendo un ORM, mecanismos de autenticación y un panel administrativo. Flask ofrece un núcleo más reducido y permite incorporar funcionalidades adicionales mediante extensiones.

FastAPI se diferencia por integrar directamente anotaciones de tipo, validación con Pydantic y generación de una especificación OpenAPI. Estas características resultaban coherentes con la necesidad de construir endpoints que pudieran ser inspeccionados y consumidos por diferentes interfaces.

Esta comparación tiene un carácter conceptual y no corresponde a una evaluación experimental de los tres frameworks. No se implementaron versiones equivalentes de Aurora con Django y Flask ni se compararon métricas de rendimiento, seguridad o mantenibilidad. En consecuencia, la selección de FastAPI se justifica por su ajuste a los requerimientos del subproyecto y no como demostración de superioridad general.

2.3.4 Aplicación en Aurora

Aurora utiliza FastAPI 0.110.2 como framework principal del Back-End y Uvicorn 0.29.0 como servidor ASGI. Ambas versiones se encuentran declaradas entre las dependencias del proyecto.

FastAPI permite registrar los routers que componen el servidor y exponer las rutas desde una misma aplicación. A través de estos routers se atienden operaciones relacionadas con autenticación web, acceso móvil, personas, actividades educativas, evaluaciones, emociones, desempeño, gamificación e integración con servicios externos.

La documentación OpenAPI y Swagger UI se utilizó para consultar y probar los endpoints disponibles. Su aplicación se concentra principalmente en las rutas HTTP documentables; las vistas web renderizadas mediante Jinja2 forman parte del mismo servidor, pero siguen un flujo basado en solicitudes, sesiones, formularios y respuestas HTML.

FastAPI también proporciona el entorno de integración para los componentes relacionados con inteligencia artificial. El servidor recibe la información, prepara las solicitudes dirigidas al servicio externo y conserva las respuestas requeridas por los flujos de Aurora.

2.4 Persistencia de datos

La persistencia permite conservar información más allá de la ejecución inmediata de una aplicación. En Aurora, resulta necesaria para almacenar usuarios, roles, establecimientos, actividades, evaluaciones, respuestas, resultados, registros emocionales, progreso y recompensas.

La persistencia también permite relacionar eventos ocurridos en diferentes momentos. Por ejemplo, la asignación de Glims depende de comparar el resultado actual con el mejor desempeño anterior del estudiante. Del mismo modo, el seguimiento del progreso requiere conservar las actividades completadas y las unidades disponibles.

Estas funciones no demuestran por sí solas una mejora del aprendizaje. Su aporte consiste en mantener información histórica y proporcionar los datos requeridos por las funcionalidades de monitoreo del propio desempeño y autorreflexión implementadas en la plataforma.

2.4.1 Bases de datos relacionales

Una base de datos relacional organiza la información mediante tablas compuestas por filas y columnas. Las relaciones entre tablas pueden representarse mediante claves primarias y foráneas, mientras las restricciones permiten establecer condiciones sobre los valores almacenados.

Este modelo resulta adecuado cuando el dominio presenta entidades y relaciones estructuradas. En Aurora existen asociaciones entre personas, roles, establecimientos, unidades, actividades, preguntas, respuestas, emociones, resultados y recompensas. Una base relacional permite representar estas dependencias y realizar consultas que combinen la información correspondiente.

MySQL es un sistema de gestión de bases de datos relacionales que admite consultas SQL, índices, restricciones y transacciones. En tablas que utilizan un motor transaccional compatible, como InnoDB, pueden aplicarse claves foráneas y operaciones de confirmación o reversión (Oracle, s. f.).

La utilización de una base relacional no garantiza automáticamente la calidad de los datos. La integridad depende de las restricciones definidas en el esquema, de las transacciones empleadas y de las validaciones efectuadas por la aplicación.

2.4.2 ORM

El mapeo objeto-relacional, conocido como ORM, permite representar tablas y relaciones mediante clases y objetos del lenguaje de programación. Su propósito es proporcionar una abstracción para consultar y modificar registros sin escribir manualmente todas las sentencias SQL.

Un ORM puede facilitar la reutilización de modelos y la definición de relaciones. No obstante, no reemplaza el conocimiento del modelo relacional ni elimina la necesidad de revisar las consultas generadas. Las estrategias de carga, el manejo de sesiones y los

límites transaccionales continúan influyendo en el comportamiento y rendimiento del sistema.

Por esta razón, el ORM debe entenderse como una herramienta de acceso y representación, no como una garantía automática de integridad, eficiencia o seguridad.

2.4.3 SQLAlchemy

SQLAlchemy es una biblioteca de Python que proporciona herramientas para construir y ejecutar operaciones SQL, además de un ORM para relacionar clases con tablas. Permite definir modelos declarativos, relaciones, sesiones y transacciones (SQLAlchemy, s. f.).

El ORM de SQLAlchemy permite recuperar registros como objetos y representar asociaciones entre entidades. También admite consultas explícitas cuando una operación requiere mayor control sobre filtros, uniones, bloqueos o actualizaciones.

Las sesiones de SQLAlchemy administran la interacción de los objetos con la base de datos dentro de una unidad de trabajo. La aplicación debe determinar cuándo confirmar una transacción, cuándo revertirla ante un error y cuándo cerrar la sesión. Por consiguiente, el uso de SQLAlchemy no elimina la responsabilidad de gestionar correctamente las operaciones transaccionales.

2.4.4 Relaciones e integridad

La integridad de los datos se relaciona con el mantenimiento de información válida y coherente. En una base de datos relacional puede apoyarse mediante claves primarias, claves foráneas, restricciones de unicidad, dominios de valores y transacciones.

Las claves primarias identifican los registros, mientras las claves foráneas representan dependencias entre tablas. Estas restricciones pueden impedir, por ejemplo, que un registro relacionado haga referencia a una entidad inexistente, siempre que se encuentren correctamente declaradas y habilitadas.

Las transacciones permiten agrupar operaciones que deben confirmarse como una unidad. Si una parte del proceso falla, la aplicación puede revertir los cambios pendientes para evitar escrituras parciales. Esto resulta relevante en operaciones de Aurora que modifican el desempeño, el progreso o las recompensas.

La integridad no depende exclusivamente de la base de datos. Algunas condiciones corresponden a reglas de negocio y deben verificarse en el Back-End. Entre ellas se encuentran la elegibilidad para recibir Glims, la comparación con el mejor puntaje anterior y las condiciones necesarias para desbloquear unidades.

2.4.5 Aplicación en Aurora

Aurora utiliza MySQL como sistema de gestión de bases de datos, SQLAlchemy 2.0.30 para el mapeo objeto-relacional y PyMySQL 1.1.0 como controlador de conexión. Estas dependencias permiten que el Back-End consulte y modifique la información desde Python.

El RUT se utiliza como identificador principal de las personas y participa en las relaciones con los perfiles y roles del sistema. La tabla PersonaRol utiliza una clave compuesta por RUT, RolId y Colegioid, lo que permite representar la asignación de una persona a un rol dentro de un establecimiento.

El esquema también utiliza claves foráneas y valores restringidos para representar relaciones y limitar determinados estados. Estas restricciones complementan las validaciones realizadas por la aplicación, aunque no sustituyen las reglas de negocio ejecutadas en el servidor.

Algunas relaciones definidas mediante SQLAlchemy emplean carga anticipada, como la estrategia joined. Esta opción permite recuperar ciertos objetos relacionados junto con la consulta principal. Su uso puede reducir accesos posteriores en los flujos que requieren preguntas y alternativas, pero en el subproyecto no se realizaron mediciones comparativas de rendimiento entre estrategias de carga.

SQLAlchemy también participa en operaciones que requieren control transaccional. La actualización del desempeño y la asignación de recompensas deben considerar registros previos, mientras la compra de elementos requiere verificar y modificar el saldo disponible de manera coherente. El comportamiento de estos procesos se describe con mayor detalle en el Capítulo 3.

2.5 Seguridad en sistemas Back-End

La seguridad de un Back-End educativo resulta especialmente relevante cuando se procesan datos personales, académicos y emocionales de estudiantes. Entre sus dimensiones se encuentran la autenticación, la autorización, el almacenamiento de credenciales, la validación de entradas y la protección de las operaciones expuestas por el servidor.

En este marco debe distinguirse entre los mecanismos recomendados por las buenas prácticas y aquellos efectivamente implementados en el prototipo. Aurora incorpora medidas parciales, pero conserva limitaciones que impiden presentar el sistema como una solución preparada integralmente para producción.

2.5.1 Autenticación, sesiones y autorización

La autenticación permite verificar la identidad declarada por un usuario. La autorización determina las acciones o recursos a los que puede acceder después de autenticarse. Ambos procesos se relacionan, pero cumplen funciones diferentes.

Las aplicaciones web pueden mantener el estado de autenticación mediante sesiones. En este mecanismo, el servidor o una cookie asociada conserva información que permite reconocer al usuario entre solicitudes. Cuando una cookie se encuentra firmada, la firma permite detectar modificaciones no autorizadas, pero no implica necesariamente que su contenido esté cifrado.

En Aurora, la autenticación web valida el RUT y la contraseña registrados para la persona, comprueba su asociación con un rol y conserva información de la sesión mediante

SessionMiddleware. Este componente utiliza sesiones basadas en cookies firmadas. La información de la cookie puede ser legible, pero una modificación invalida su firma.

El acceso móvil utiliza un mecanismo diferente. El endpoint POST /mobile/login comprueba la combinación de RUT y establecimiento y devuelve información del usuario. Este flujo no emite JWT ni otro token para autenticar las solicitudes posteriores. Por esta razón, debe describirse como una verificación de acceso básica y no como un sistema completo de autenticación móvil.

JWT no se utilizó en Aurora y, por consiguiente, no corresponde presentarlo como una tecnología implementada. La incorporación futura de un mecanismo de autenticación móvil basado en tokens, junto con expiración y validación de credenciales, puede plantearse como una recomendación en el Capítulo 5.

2.5.2 Almacenamiento seguro de contraseñas

Las contraseñas no deben almacenarse en texto plano ni mediante cifrado reversible. Las buenas prácticas recomiendan utilizar funciones específicas para derivación de contraseñas, acompañadas de sal y parámetros de costo que dificulten los intentos automatizados. OWASP recomienda Argon2id como primera opción y mantiene alternativas como scrypt o bcrypt según las condiciones del sistema (OWASP Foundation, s. f.).

El almacenamiento mediante hash permite verificar una contraseña sin conservar directamente su contenido original. Aunque no elimina todos los riesgos, reduce el impacto de una exposición de la base de datos frente al almacenamiento en texto plano.

En el prototipo de Aurora, las contraseñas se almacenan y comparan en texto plano. Por lo tanto, no corresponde afirmar que se implementó hashing, cifrado o almacenamiento seguro de credenciales. Esta condición constituye una limitación de seguridad y debe identificarse como una mejora prioritaria antes de considerar un despliegue productivo.

2.5.3 Roles y control de acceso

El control de acceso basado en roles, o RBAC, es un modelo en el que los permisos se asignan a roles y los usuarios reciben autorizaciones mediante su asociación con dichos roles (Ferraiolo et al., 2001).

En una plataforma educativa, este modelo permite diferenciar las operaciones disponibles para estudiantes, docentes y otros actores. Sin embargo, la existencia de una tabla de roles no demuestra por sí sola una implementación completa de RBAC. Para ello se requiere una política de permisos aplicada de forma consistente sobre los recursos protegidos.

Aurora modela los roles mediante entidades como Rol y PersonaRol. La clave compuesta de PersonaRol relaciona una persona, un rol y un establecimiento. Esta estructura permite representar el contexto institucional en el cual una persona desempeña un rol.

En la interfaz web, el proceso de inicio de sesión recupera el rol del usuario, almacena información en la sesión y utiliza estos datos para determinar determinadas vistas o redirecciones. Esto constituye un control basado en roles, pero no se verificó una matriz completa de permisos aplicada uniformemente a todos los endpoints.

Por esta razón, resulta más preciso señalar que Aurora incorpora diferenciación de acceso mediante roles, sin afirmar que implementa integralmente un modelo RBAC sobre todas las operaciones del Back-End.

2.5.4 Validación de datos

La validación de datos comprueba que la información recibida cumpla con la estructura, el tipo y las condiciones esperadas antes de ser utilizada. Esta validación puede efectuarse en diferentes niveles: modelos de entrada, lógica de negocio y restricciones de la base de datos.

FastAPI y Pydantic permiten validar los parámetros y cuerpos declarados mediante anotaciones de tipo y modelos. Cuando una entrada no cumple con el esquema correspondiente, la solicitud puede rechazarse antes de ejecutar la operación.

No todas las entradas de Aurora utilizan modelos Pydantic de manera uniforme. El Back-End también recibe parámetros directos y formularios web, por lo que la cobertura de validación depende de cada ruta. En consecuencia, no corresponde afirmar que toda la información recibida por el servidor se valida automáticamente.

Las reglas de negocio complementan la validación estructural. Por ejemplo, el servidor debe comprobar la existencia de registros relacionados, las condiciones de avance, los resultados anteriores y la validez funcional de las operaciones. La base de datos añade restricciones sobre claves y determinados dominios de valores.

2.5.5 Protección de endpoints

La protección de endpoints busca impedir que usuarios no autorizados accedan a operaciones o información restringida. Para ello se requiere autenticar al usuario, determinar sus permisos y comprobar su relación con el recurso solicitado.

En Aurora, la protección presenta diferencias entre los distintos grupos de rutas. Las vistas web pueden consultar los datos almacenados en la sesión firmada para verificar la existencia de un usuario y su rol. El flujo móvil, en cambio, no utiliza tokens que permitan autenticar de forma consistente las solicitudes posteriores al inicio de sesión.

Tampoco se verificó una política uniforme de autorización aplicada a todos los endpoints generales. Por esta razón, la protección debe describirse como parcial y acorde con el estado de prototipo del sistema.

La versión anterior mencionaba mecanismos de protección contra manipulación de rutas y *path traversal* en la gestión de archivos. Dado que no se cuenta con evidencia suficiente para asegurar que este control se aplica sistemáticamente, dicha afirmación se elimina. La

validación de tipos, nombres, rutas y permisos de archivos debe considerarse dentro de las mejoras necesarias para un eventual entorno productivo.

2.6 Inteligencia Artificial aplicada a la educación

La inteligencia artificial puede utilizarse en educación para generar respuestas, procesar información y apoyar determinadas tareas de retroalimentación. La IA generativa amplía estas posibilidades mediante modelos capaces de producir contenido a partir de instrucciones y datos de entrada.

Su incorporación en una plataforma educativa no implica que el sistema comprenda al estudiante ni que sus respuestas sean necesariamente correctas. La UNESCO advierte que el uso educativo de IA generativa requiere protección de datos, supervisión humana, adecuación a la edad y una definición clara de sus propósitos pedagógicos (UNESCO, 2023).

En Aurora, la inteligencia artificial se utiliza como un componente de apoyo. Sus salidas no corresponden a diagnósticos psicológicos ni reemplazan la interpretación y responsabilidad pedagógica del docente.

2.6.1 IA generativa

La IA generativa comprende modelos capaces de producir contenido nuevo, como texto, imágenes o audio, a partir de los patrones aprendidos durante su entrenamiento y de las instrucciones proporcionadas por el usuario.

Los modelos de lenguaje pueden generar respuestas conversacionales, resúmenes y explicaciones. Sin embargo, también pueden producir información incorrecta, incompleta, sesgada o inadecuada para el contexto. Por esta razón, sus resultados no deben asumirse automáticamente como hechos comprobados.

Desde la perspectiva del Back-End, la integración con un modelo externo requiere controlar las credenciales de acceso, preparar las instrucciones, delimitar la información enviada y procesar las respuestas. La comunicación a través del servidor evita que la clave del proveedor de IA deba incorporarse directamente en la interfaz cliente.

Esta intermediación también permite registrar las interacciones necesarias para el funcionamiento de Aurora. No obstante, la persistencia de datos no garantiza por sí sola una trazabilidad completa: esta depende de qué entradas, respuestas, fechas y relaciones sean efectivamente almacenadas.

2.6.2 Retroalimentación y apoyo contextual

La personalización del aprendizaje supone adaptar contenidos, estrategias o actividades a las características y necesidades del estudiante. Esta definición es más amplia que las funcionalidades implementadas en Aurora.

Aurora no modifica automáticamente el currículo ni selecciona contenidos mediante un modelo adaptativo. La plataforma utiliza información individual, como comentarios y registros emocionales, para generar determinadas respuestas y resúmenes

contextualizados. Por esta razón, resulta más preciso hablar de retroalimentación y apoyo contextual que de personalización integral del aprendizaje.

El Back-End participa recuperando los datos requeridos, construyendo la solicitud destinada al servicio de IA y conservando las respuestas utilizadas posteriormente por la plataforma.

Los resultados generados a partir de comentarios emocionales deben interpretarse como texto de apoyo. No corresponden a evaluaciones clínicas, diagnósticos psicológicos ni recomendaciones profesionales sobre la salud mental del estudiante.

2.6.3 Posibilidades y limitaciones

La IA generativa permite producir respuestas y resúmenes en periodos breves, además de adaptar la redacción a las instrucciones definidas por la aplicación. Estas capacidades pueden utilizarse para apoyar interacciones educativas y sintetizar información.

No obstante, sus resultados dependen del modelo, de los datos de entrada y de las instrucciones utilizadas. También pueden presentar errores, sesgos o contenido poco adecuado para la edad del usuario. Cuando se procesan datos de menores y registros emocionales, deben considerarse especialmente la privacidad, la minimización de datos y la supervisión humana.

No se realizaron pruebas pedagógicas que permitan afirmar que las funciones de IA de Aurora mejoraron el aprendizaje, la autorregulación, el acompañamiento percibido o el trabajo docente. La validación del subproyecto se limitó al funcionamiento técnico de la integración y de los flujos asociados.

Por ello, la IA debe presentarse como un servicio externo integrado por el Back-End y no como una solución autónoma capaz de interpretar de manera infalible el estado del estudiante.

2.6.4 Aplicación en Aurora

La integración con OpenAI se implementó mediante la biblioteca openai 1.47.1 y el modelo GPT-4o-mini. Desde el alcance del subproyecto, el Back-End gestiona la comunicación con el servicio externo y la persistencia requerida por tres funciones de la plataforma.

La primera función corresponde al chatbot de Aurora. El servidor recibe el mensaje del estudiante, prepara la solicitud destinada al modelo y devuelve la respuesta generada. Las interacciones se almacenan mediante la entidad RegistroIA, permitiendo recuperar el historial conservado por la aplicación. El comportamiento del chatbot depende de las instrucciones definidas en el Back-End y de la respuesta proporcionada por el servicio externo.

La segunda función corresponde al análisis emocional individual ejecutado durante el post-check. El flujo utiliza la información emocional y el comentario ingresado por el estudiante para solicitar un texto de retroalimentación. La respuesta se almacena junto

con la información correspondiente al registro emocional. Su propósito es apoyar la reflexión posterior a la actividad y no emitir un diagnóstico.

La tercera función genera un resumen para el dashboard docente a partir del historial emocional disponible para un estudiante. Este resumen presenta una síntesis generada por el modelo y debe ser interpretado por el docente como información complementaria, no como una evaluación profesional o clínica.

Estas funciones demuestran la integración técnica entre el Back-End, la base de datos y el servicio de OpenAI. Sin embargo, no permiten concluir que el modelo mejora la autorregulación o el desempeño académico de los estudiantes, debido a que dichos efectos no fueron evaluados.

2.7 Autorregulación escolar

La autorregulación del aprendizaje se refiere a los procesos mediante los cuales el estudiante participa activamente en la planificación, supervisión y evaluación de su propio aprendizaje. No consiste únicamente en dedicar más tiempo al estudio, sino en establecer propósitos, seleccionar estrategias, monitorear el desempeño y ajustar la conducta de acuerdo con los resultados obtenidos.

Desde la perspectiva sociocognitiva, Zimmerman describe la autorregulación como un proceso cíclico compuesto por tres fases: anticipación, ejecución y autorreflexión (Zimmerman, 2000). Durante la anticipación, el estudiante analiza la tarea, establece metas y selecciona estrategias. En la fase de ejecución, aplica dichas estrategias, mantiene el esfuerzo y supervisa su desempeño. Finalmente, en la autorreflexión, evalúa los resultados, interpreta sus causas y considera ajustes para futuras situaciones de aprendizaje.

Estas fases no corresponden necesariamente a una secuencia rígida y aislada. La reflexión sobre un resultado puede influir en las metas, expectativas y estrategias utilizadas en una actividad posterior. Por ello, la autorregulación se comprende como un ciclo en el que las experiencias previas intervienen en las decisiones futuras del estudiante.

2.7.1 Fundamentos teóricos

La autorregulación comprende componentes cognitivos, metacognitivos, motivacionales, conductuales y emocionales. Los componentes cognitivos se relacionan con las estrategias utilizadas para comprender, organizar y aplicar información. Los componentes metacognitivos permiten observar el propio proceso de aprendizaje y reconocer si una estrategia está produciendo el resultado esperado.

La dimensión motivacional interviene en la disposición para iniciar una actividad, sostener el esfuerzo y persistir frente a las dificultades. Los componentes conductuales y emocionales se relacionan con la forma en que el estudiante organiza sus acciones y reconoce la influencia que su estado interno puede ejercer sobre el aprendizaje.

Autores como Zimmerman, Pintrich y Schunk han señalado que los procesos autorregulatorios pueden desarrollarse mediante orientación, práctica, observación, retroalimentación y oportunidades para reflexionar sobre el propio desempeño (Pintrich, 2000; Zimmerman, 2002; Schunk & Zimmerman, 2012). El apoyo educativo resulta especialmente relevante cuando los estudiantes todavía están desarrollando hábitos de autonomía y estrategias para supervisar su aprendizaje.

El registro de experiencias también puede apoyar determinados procesos autorregulatorios. Contar con información sobre el estado previo del estudiante, la actividad realizada, el resultado obtenido y su estado posterior puede facilitar la identificación de cambios o patrones. Sin embargo, la existencia de estos registros no demuestra por sí sola que el estudiante analice la información, establezca metas o modifique autónomamente sus estrategias.

2.7.2 Autorregulación y aprendizaje autónomo

El aprendizaje autónomo requiere que el estudiante asuma un papel activo en las decisiones relacionadas con su aprendizaje. Esto no significa que deba trabajar sin acompañamiento, sino que participe progresivamente en la definición de propósitos, selección de estrategias, supervisión de avances y evaluación de resultados.

En contextos escolares, la autonomía puede desarrollarse mediante estructuras de apoyo que orienten estas decisiones y reduzcan gradualmente la asistencia externa. La retroalimentación, el modelamiento y las oportunidades de autoevaluación pueden contribuir al desarrollo de estos procesos (Zimmerman, 2002).

La tecnología puede proporcionar algunas de estas estructuras mediante secuencias de actividades, retroalimentación, registros históricos y representaciones del progreso. Sin embargo, una plataforma no promueve automáticamente la autonomía o la autorregulación por el solo hecho de digitalizar actividades. Para apoyar estos procesos debe permitir que el estudiante utilice la información disponible para comprender su desempeño y orientar decisiones posteriores.

Desde la perspectiva del Back-End, esto requiere conservar información producida en distintos momentos del proceso, como registros emocionales, respuestas, resultados y avances. La infraestructura puede organizar y proporcionar estos datos, pero la interpretación y utilización consciente de la información continúan dependiendo del estudiante, de las interfaces y del acompañamiento pedagógico.

2.7.3 Aplicación en Aurora

Aurora incorpora funcionalidades relacionadas con algunos procesos de la autorregulación mediante un flujo que articula registros emocionales, actividades educativas, resultados y progreso. Desde el Back-End, este flujo se materializa mediante el almacenamiento del precheck emocional, la ejecución de evaluaciones o minijuegos, el cálculo del puntaje, la actualización del desempeño y el registro del postcheck emocional.

Estos componentes pueden relacionarse parcialmente con el modelo cíclico de Zimmerman, pero no constituyen una implementación completa de sus fases. El precheck

permite que el estudiante registre cómo se siente antes de la actividad, lo que puede favorecer una conciencia inicial de su estado. Sin embargo, no incorpora por sí mismo procesos centrales de la fase de anticipación, como establecer metas académicas o seleccionar estrategias de aprendizaje.



Figura 2.1. Ciclo de autorregulación del aprendizaje.

Fuente: Elaboración propia basada en Zimmerman (2000).

La resolución de la actividad y el registro de sus resultados se relacionan con la fase de ejecución y con el monitoreo del desempeño. El postcheck, junto con la visualización del resultado y la retroalimentación disponible, proporciona una oportunidad para reconocer el estado posterior a la actividad y reflexionar sobre la experiencia. No obstante, el sistema no comprueba que el estudiante interprete las causas de su resultado ni que ajuste sus estrategias futuras.

El Back-End también aplica condiciones de desbloqueo progresivo. Para habilitar determinadas unidades posteriores, comprueba que las sesiones requeridas de la unidad anterior cuenten con registros emocionales previo y posterior, además del desempeño correspondiente. Esta regla asegura la completitud técnica del flujo definido por Aurora y evita que el avance dependa únicamente de responder una actividad.

Por consiguiente, Aurora proporciona infraestructura para el monitoreo del propio desempeño, el registro de estados emocionales y determinadas instancias de autorreflexión. No implementa la autorregulación del aprendizaje en toda su amplitud ni permite concluir que los estudiantes hayan desarrollado esta capacidad. La evaluación realizada en el subproyecto se concentra en el funcionamiento técnico de estos componentes y no en sus efectos pedagógicos.

2.8 Gamificación

La gamificación consiste en utilizar elementos propios del diseño de juegos en contextos que no corresponden propiamente a juegos (Deterding et al., 2011). En educación, estos elementos pueden incluir puntos, niveles, desafíos, indicadores de progreso, recompensas o elementos de personalización.

Su incorporación busca influir en la experiencia y participación del usuario, pero sus resultados dependen del diseño y del contexto en que se aplica. La presencia de puntos o recompensas no garantiza un aumento de la motivación ni una mejora del aprendizaje. Para adquirir valor educativo, las mecánicas deben relacionarse de manera coherente con las actividades y objetivos pedagógicos.

La gamificación tampoco debe confundirse con el aprendizaje basado en juegos. En este último, el juego constituye el medio principal mediante el cual se desarrolla la experiencia de aprendizaje. En la gamificación, en cambio, se incorporan determinados elementos de diseño de juegos dentro de una actividad que mantiene un propósito principal diferente.

2.8.1 Motivación y participación

La motivación influye en la disposición del estudiante para iniciar una actividad, mantener el esfuerzo y persistir frente a una dificultad. Desde la teoría de la autodeterminación, Ryan y Deci identifican la autonomía, la competencia y la relación como necesidades psicológicas vinculadas con formas de motivación de mayor calidad (Ryan & Deci, 2000).

En una plataforma educativa, la percepción de competencia puede apoyarse mediante desafíos comprensibles, retroalimentación y señales de progreso. La autonomía requiere oportunidades reales para tomar decisiones significativas, mientras la relación se vincula con la percepción de apoyo o conexión con otras personas.

Los elementos de gamificación pueden contribuir a estas dimensiones cuando su diseño es coherente con ellas. Sin embargo, también existe el riesgo de que las recompensas se conviertan en el propósito principal de la actividad. En ese caso, el estudiante podría concentrarse en obtener puntos o elementos virtuales sin prestar atención al aprendizaje o a la reflexión sobre su desempeño.

En el contexto de Aurora, las recompensas se relacionan con el resultado de las actividades y con la mejora respecto de resultados anteriores. Esta relación busca orientar la participación hacia el progreso individual. Sin embargo, el subproyecto no evaluó si este mecanismo modificó efectivamente la motivación, la participación o los hábitos de estudio de los estudiantes.

2.8.2 Recompensas, progreso y engagement

Las recompensas constituyen uno de los elementos más visibles de la gamificación. Pueden representarse mediante puntos, monedas virtuales, insignias, niveles o elementos de personalización. Su función dentro del sistema consiste en comunicar un logro o proporcionar acceso a nuevas posibilidades.

Los indicadores de progreso permiten representar la continuidad entre diferentes actividades. Mediante ellos, el usuario puede reconocer qué acciones ha completado y cuáles permanecen disponibles o bloqueadas. Esta representación puede favorecer la comprensión de la trayectoria propuesta por la plataforma, aunque su efecto sobre la conducta debe evaluarse y no asumirse a partir de su sola implementación.

El compromiso o involucramiento describe el grado de participación que una persona mantiene con una actividad. En educación, una mayor frecuencia de interacción no equivale necesariamente a un aprendizaje más profundo. Por esta razón, el análisis de una solución gamificada debe diferenciar entre el funcionamiento de sus mecánicas, la participación observable y sus posibles efectos pedagógicos.

Desde el Back-End, las recompensas y el progreso requieren reglas consistentes. El servidor debe determinar cuándo corresponde entregar una recompensa, conservar los resultados anteriores y evitar que una misma acción produzca beneficios duplicados o inconsistentes.

2.8.3 Aplicación en Aurora

En Aurora, la gamificación se implementa principalmente mediante los Glims, una moneda virtual asociada con el desempeño obtenido en las actividades. Los estudiantes pueden utilizarla para adquirir elementos disponibles en la tienda de la plataforma.

La asignación de Glims considera el mejor puntaje registrado anteriormente. La cantidad entregada corresponde a la diferencia positiva entre el puntaje nuevo y la mejor marca previa:

$$\Delta\text{Glims} = \max(0, \text{puntaje nuevo} - \text{mejor puntaje previo})$$

De esta forma, si el estudiante obtiene un resultado igual o inferior a su mejor puntaje anterior, no recibe Glims adicionales. Si supera esa marca, recibe únicamente la diferencia correspondiente a la mejora. Esta regla reduce la posibilidad de acumular recompensas mediante la repetición mecánica de una actividad sin mejorar el resultado.

El Back-End conserva los resultados necesarios para realizar esta comparación y coordina la actualización del desempeño con la asignación de la recompensa. De esta manera, los Glims no dependen exclusivamente de que una actividad haya sido enviada, sino de la variación respecto del mejor desempeño almacenado.

La tienda incorpora reglas adicionales. Antes de completar una compra, el servidor comprueba que el estudiante disponga de Glims suficientes y que no posea previamente el elemento solicitado. La operación utiliza bloqueo de fila mediante SELECT ... FOR UPDATE, lo que permite mantener la coherencia del saldo ante solicitudes de compra concurrentes.

El desbloqueo progresivo de unidades también proporciona una representación del avance dentro de la plataforma. No obstante, su función principal en el Back-End consiste en comprobar la finalización de las condiciones definidas para las sesiones anteriores.

Estas decisiones demuestran que Aurora implementa reglas de negocio para administrar progreso, recompensas y compras de manera coherente. Sin embargo, no permiten concluir que la gamificación haya aumentado la motivación, el compromiso o el aprendizaje, debido a que estos efectos no fueron evaluados con estudiantes.

2.9 Cierre del capítulo

Este capítulo presentó los fundamentos conceptuales relacionados con el desarrollo del Back-End de Aurora. Se abordaron la metodología de desarrollo, la arquitectura del servidor, FastAPI, la persistencia relacional, la seguridad, la integración de inteligencia artificial, la autorregulación del aprendizaje y la gamificación.

El análisis distinguió entre los fundamentos generales, las funciones efectivamente implementadas y las limitaciones del prototipo. Esta separación permitió evitar que las capacidades potenciales de las tecnologías se presentaran como resultados comprobados del subproyecto.

Desde la perspectiva educativa, Aurora proporciona soporte tecnológico para el monitoreo del propio desempeño, el registro emocional y determinadas instancias de autorreflexión. Estas funcionalidades se relacionan con algunos procesos de la autorregulación, pero no representan su implementación completa ni demuestran efectos sobre el aprendizaje, la motivación o el bienestar de los estudiantes.

Desde la perspectiva técnica, el Back-End centraliza la persistencia, las reglas de evaluación, el progreso, las recompensas y la comunicación con servicios externos. El capítulo siguiente presenta la forma en que estos fundamentos se materializaron en la arquitectura implementada, el modelo de datos y los principales flujos de negocio que sostienen la interacción entre la aplicación móvil, la plataforma web y los servicios externos.

CAPÍTULO 3: PROPUESTA DE SOLUCIÓN

3.1 Introducción

Sobre la base del marco conceptual establecido en el capítulo anterior, el presente capítulo describe la solución de ingeniería implementada en el subproyecto Back-End de Aurora. Su propósito es explicar cómo los principios de modularidad, separación de responsabilidades, persistencia relacional, control de acceso y documentación de servicios se materializaron en una aplicación funcional.

Aurora corresponde a un proyecto grupal compuesto por distintos componentes tecnológicos y pedagógicos. En este contexto, el alcance de la presente memoria no comprende el desarrollo completo de la plataforma, sino específicamente la infraestructura Back-End que permite articular la aplicación móvil, la plataforma web, la base de datos y los servicios externos utilizados por el sistema.

El Back-End actúa como punto central de coordinación. Recibe las solicitudes generadas por los clientes, valida la información de entrada, ejecuta reglas de negocio, administra la persistencia y construye las respuestas correspondientes. Entre los procesos soportados se encuentran la gestión de usuarios y roles, el desarrollo de sesiones educativas, el registro emocional, el cálculo de desempeño, el desbloqueo progresivo de unidades, la economía de Glims y la integración con servicios de inteligencia artificial.

La solución se implementó mediante un único servidor FastAPI organizado modularmente. Esta aplicación atiende simultáneamente endpoints REST, rutas especializadas para la aplicación móvil y vistas web renderizadas del lado del servidor mediante Jinja2. La persistencia se gestiona con SQLAlchemy y MySQL, mientras que la integración con GPT-4o-mini se realiza desde el Back-End para evitar que los clientes accedan directamente al proveedor externo.

El capítulo comienza describiendo la metodología aplicada y la arquitectura general. Posteriormente, profundiza en la separación lógica de responsabilidades, el stack tecnológico, el modelo de datos, los mecanismos de acceso y los principales flujos de negocio. Finalmente, se analizan la consistencia transaccional, la documentación de la API, la mantenibilidad y las limitaciones reconocidas de la implementación.

3.2 Metodología de desarrollo aplicada

El desarrollo de Aurora se organizó mediante un enfoque iterativo basado en Scrum. Los fundamentos de este marco de trabajo fueron presentados en el Capítulo 2; por ello, esta sección se concentra en su aplicación dentro del subproyecto Back-End.

La naturaleza grupal de Aurora hizo necesario coordinar componentes que evolucionaban de manera paralela. La aplicación móvil requería endpoints capaces de intercambiar información en formato JSON, mientras que la plataforma web necesitaba rutas para renderizar páginas HTML, administrar formularios y mantener el contexto del usuario mediante sesiones. Ambos clientes dependían, además, de un modelo de datos y de reglas de negocio comunes.

El trabajo iterativo permitió desarrollar estas capacidades de manera incremental. En lugar de construir cada módulo de forma completamente aislada y posponer la integración hasta las etapas finales, se habilitaron progresivamente flujos funcionales completos. Una funcionalidad no se consideraba integrada únicamente por disponer de un endpoint, sino cuando el cliente podía enviar información, el servidor validarla, la base de datos persistirla y el flujo continuar hacia la operación siguiente.

La priorización se orientó inicialmente hacia los componentes de los cuales dependían los demás módulos. La configuración de la aplicación, la conexión con la base de datos, los modelos principales y las rutas de acceso precedieron a funcionalidades de mayor nivel, como el cálculo de puntajes, el desbloqueo de unidades, la economía de Glims y la generación de respuestas mediante inteligencia artificial.

Esta forma de trabajo redujo el riesgo de detectar incompatibilidades tardías entre los subproyectos. Los contratos de comunicación pudieron ajustarse durante el desarrollo, mientras que las reglas centrales permanecieron bajo responsabilidad del Back-End. La metodología aplicada aportó así capacidad de adaptación y facilitó la integración gradual entre las interfaces y la infraestructura del servidor.

3.3 Arquitectura general del Back-End

La solución implementada corresponde a un monolito modular construido sobre FastAPI. El sistema se ejecuta como una única aplicación de servidor, pero organiza internamente sus responsabilidades mediante routers, modelos, esquemas, funciones auxiliares y componentes de integración diferenciados.

Esta clasificación permite distinguir la arquitectura adoptada de una solución basada en microservicios. Aurora no divide sus funcionalidades en procesos independientes desplegados por separado. En cambio, mantiene una única unidad de ejecución, dentro de la cual los dominios funcionales se separan mediante módulos de código.

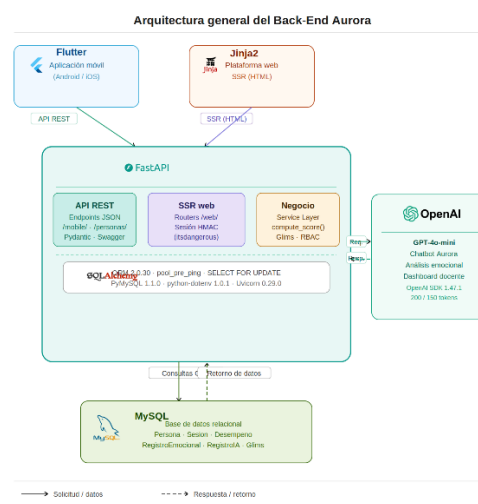


Figura 3.1. Arquitectura general del Back-End de Aurora. Fuente: elaboración propia.

Como se observa en la Figura 3.1, el servidor se comunica con dos interfaces de usuario principales. La aplicación móvil desarrollada en Flutter consume endpoints especializados que intercambian estructuras JSON. La plataforma web, por su parte, utiliza rutas renderizadas del lado del servidor mediante Jinja2 y mantiene el estado de navegación por medio de sesiones.

Además de estas interfaces, el Back-End expone endpoints REST para operaciones de consulta y administración de entidades. De esta manera, una misma aplicación FastAPI atiende tres canales de comunicación, resumidos en la Tabla 3.1

Tabla 3.1. Canales de acceso atendidos por el Back-End de Aurora.

Canal	Organización Principal	Formato de respuesta	Propósito
API REST general	Prefijos como /personas, /academia y /emociones	JSON	Operaciones CRUD y consulta de recursos
Plataforma web	Routers ubicados en routers/web/	HTML renderizado con Jinja2	Interacción web mediante formularios y sesiones
Aplicación móvil	Prefijo /mobile/	JSON	Integración con la aplicación Flutter

Fuente: elaboración propia.

El núcleo FastAPI concentra la recepción de solicitudes, la validación de datos y la coordinación de operaciones. Las reglas reutilizables se ubican en módulos auxiliares, mientras que SQLAlchemy actúa como intermediario entre la aplicación y MySQL.

La integración con OpenAI se mantiene como una dependencia externa controlada desde el servidor. El Back-End coordina el intercambio de información con el módulo de inteligencia artificial y gestiona la presentación o persistencia de sus resultados. Esta decisión evita exponer directamente las credenciales del proveedor en los clientes y mantiene la trazabilidad de las interacciones dentro de la aplicación.

El uso de un monolito modular fue coherente con el alcance académico de Aurora. Una arquitectura distribuida habría incorporado necesidades adicionales de comunicación entre servicios, despliegue, monitoreo, seguridad y tolerancia a fallos que no eran indispensables para implementar los flujos definidos. La estructura seleccionada reduce la complejidad operacional, pero mantiene una separación interna que favorece el mantenimiento y la evolución del código.

3.4 Arquitectura lógica y separación de responsabilidades

La organización lógica del Back-End distingue tres responsabilidades principales: recepción de solicitudes, aplicación de reglas de negocio y persistencia. Aunque estas responsabilidades se ejecutan dentro de una misma aplicación, su separación reduce el acoplamiento entre los endpoints y los procedimientos internos.

La estructura no constituye una arquitectura por capas completamente estricta, puesto que algunos routers todavía realizan consultas y coordinaciones específicas. Sin embargo, la extracción de las reglas más relevantes hacia funciones auxiliares representa una mejora respecto de concentrar todo el comportamiento dentro de las rutas.

3.4.1 Capa de rutas o presentación

La capa de rutas constituye el punto de entrada de las solicitudes HTTP. Los routers de FastAPI definen las direcciones disponibles, los métodos aceptados, los parámetros requeridos y el formato general de las respuestas.

Dentro de esta capa coexisten tres categorías de rutas. La primera corresponde a endpoints REST para operaciones de creación, consulta, modificación y eliminación de entidades. La segunda agrupa las rutas destinadas a la aplicación móvil bajo el prefijo `/mobile/`. La tercera se encuentra en los routers web y entrega páginas HTML construidas mediante plantillas Jinja2.

La función principal de esta capa consiste en coordinar la solicitud. Una ruta recibe la información, comprueba sus condiciones básicas, obtiene una sesión de base de datos cuando corresponde, invoca la lógica requerida y construye la respuesta HTTP.

Los endpoints JSON emplean modelos Pydantic para representar y validar las estructuras de entrada. Este mecanismo permite detectar campos ausentes, tipos incorrectos o formatos incompatibles antes de ejecutar las operaciones de negocio. Sin embargo, la validación de esquema no reemplaza las comprobaciones funcionales. Una solicitud puede ser sintácticamente válida y, aun así, representar una operación no autorizada o inconsistente con el estado del usuario.

La organización mediante routers permite separar dominios como personas, academia, emociones, evaluaciones, tienda, minijuegos, clientes móviles y vistas web. En total, la aplicación registra más de veinte routers, lo que refleja la extensión funcional alcanzada y justifica la necesidad de mantener una estructura modular.

3.4.2 Capa de servicios y funciones auxiliares

Las reglas reutilizables de mayor relevancia se concentran principalmente en los módulos ubicados en `core/helpers/`. Entre ellos se encuentran `helpers_evaluacion.py`, `helpers_resultados.py` y `helpers_web.py`.

`helpers_evaluacion.py` contiene operaciones asociadas al desarrollo de las actividades educativas. En este módulo se ubican procedimientos vinculados con el cálculo de puntajes, la comprobación del avance y el desbloqueo de unidades.

`helpers_resultados.py` concentra la actualización del desempeño y la asignación de Glims. Su responsabilidad incluye coordinar operaciones que deben mantenerse consistentes, como el registro del mejor puntaje y la recompensa asociada a una mejora.

`helpers_web.py` proporciona funciones de apoyo para las rutas renderizadas con Jinja2. Entre sus responsabilidades se encuentran la construcción de contextos para plantillas, la obtención de elementos visuales activos y el tratamiento seguro de rutas utilizadas para acceder a archivos.

Esta organización se aproxima al patrón *Service Layer*, aunque no se implementa mediante clases de servicio independientes. Su principal aporte consiste en impedir que reglas como el cálculo de resultados o la actualización de puntos queden duplicadas en múltiples endpoints.

Las rutas continúan coordinando el flujo, pero delegan las operaciones reutilizables a funciones especializadas. De este modo, un ajuste en el cálculo de puntaje puede realizarse dentro del módulo de evaluación sin modificar la forma en que el cliente envía la solicitud.

3.4.3 Capa de persistencia

La capa de persistencia se encuentra compuesta por los modelos SQLAlchemy, la configuración del motor de base de datos y la administración de sesiones.

SQLAlchemy representa las tablas mediante clases de Python y permite expresar relaciones entre entidades a través de atributos del modelo. La mayoría de las operaciones se realiza utilizando el ORM, evitando construir manualmente consultas SQL en cada endpoint.

Las rutas no mantienen conexiones permanentes. Cada operación que requiere datos recibe una sesión a través de la dependencia `get_db()`. Esta sesión permanece activa durante el procesamiento correspondiente y posteriormente se cierra, incluso cuando se produce una excepción.

MySQL conserva la información estructurada del sistema, incluyendo personas, roles, contenidos educativos, resultados, registros emocionales, interacciones con inteligencia artificial y elementos de gamificación. La capa de negocio es responsable de decidir qué

operaciones deben realizarse, mientras que la capa de persistencia materializa dichas decisiones en consultas y transacciones.

3.5 Entorno de ejecución y stack tecnológico

El Back-End se implementó en Python utilizando tecnologías seleccionadas en función de los requerimientos de comunicación, persistencia, renderizado web e integración externa. La Tabla 3.2 resume sus principales componentes.

Tabla 3.2. Tecnologías utilizadas en el Back-End de Aurora.

Tecnología	Versión	Función dentro de la solución
FastAPI	0.110.2	Framework principal para rutas REST, móviles y web
SQLAlchemy	2.0.30	Mapeo objeto-relacional y acceso a datos
MySQL	8.0	Sistema gestor de base de datos relacional
PyMySQL	1.1.0	Controlador de conexión entre SQLAlchemy y MySQL
Uvicorn	0.29.0	Servidor ASGI utilizado para ejecutar FastAPI
Jinja2	3.1.3	Renderizado de páginas HTML del lado del servidor
OpenAI SDK	1.47.1	Comunicación con el modelo GPT-4o-mini
Python-dotenv	1.0.1	Carga de variables de configuración
SessionMiddleware	Starlette	Gestión de sesiones web mediante cookies firmadas
itsdangerous	2.2.0	Soporte para firma e integridad de sesiones
Python-multipart	0.0.9	Procesamiento de formularios y archivos
Pydantic	2.7.1	Definición y validación de esquemas de datos

Fuente: elaboración propia.

FastAPI fue seleccionado como framework principal debido a que permite combinar endpoints JSON, dependencias, validación de esquemas, sesiones y renderizado web dentro de una misma aplicación. Su organización mediante routers resultó adecuada para separar los distintos dominios funcionales.

Uvicorn actúa como servidor ASGI y proporciona el entorno de ejecución de FastAPI. Aunque ASGI permite trabajar con operaciones asíncronas, la presencia del estándar no implica que cada operación interna sea automáticamente no bloqueante. Parte del acceso a datos se realiza mediante sesiones síncronas de SQLAlchemy, por lo que la capacidad efectiva depende también del comportamiento de estas operaciones.

SQLAlchemy proporciona la capa ORM utilizada para representar entidades y ejecutar consultas. PyMySQL funciona como controlador de comunicación con MySQL. Esta combinación permite mantener la lógica de acceso en Python sin perder la capacidad de utilizar operaciones relacionales específicas, como bloqueos de filas.

Jinja2 se incorporó para las interfaces web renderizadas desde el servidor. Esto permite que las rutas preparen un contexto y entreguen directamente una página HTML, evitando la necesidad de desarrollar un cliente web completamente independiente para cada funcionalidad.

python-dotenv permite separar determinados valores de configuración del código fuente. python-multipart habilita el procesamiento de formularios y archivos enviados mediante HTTP. Finalmente, el SDK de OpenAI encapsula la comunicación con GPT-4o-mini para los casos de uso definidos en Aurora.

3.6 Diseño de la base de datos

La base de datos relacional constituye el mecanismo principal de persistencia de Aurora. Su diseño representa tanto la estructura institucional del sistema como los flujos educativos, emocionales y de gamificación.

Entre las entidades principales se encuentran Persona, PersonaRol, Estudiante, Profesor, Apoderado, Colegio, Curso, Unidad, Sesion, Pregunta, Alternativa, Desempeno, RegistroEmocional, RegistroIA y las estructuras asociadas a Glms.

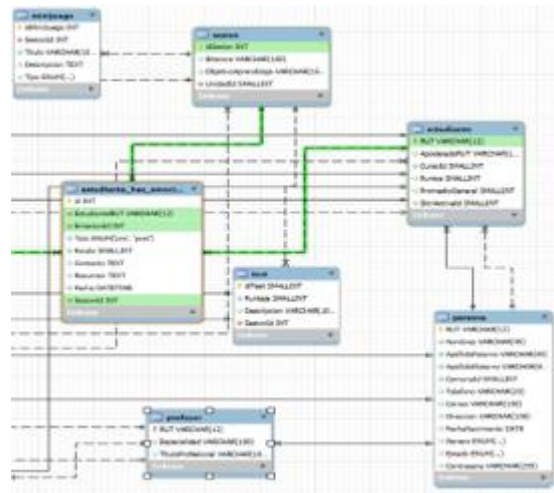


Figura 3.2. Extracto del modelo entidad-relación utilizado por el Back-End de Aurora. Fuente: elaboración propia.

La Figura 3.2 presenta las principales entidades y asociaciones administradas por el Back-End. Su estructura permite relacionar a los actores del sistema con su contexto institucional, las actividades educativas y los resultados generados durante la interacción con la plataforma.

3.6.1 Decisiones principales de modelamiento

La entidad Persona concentra los atributos comunes de los usuarios. A partir de ella se relacionan las tablas específicas de estudiante, profesor y apoderado. Esta especialización evita duplicar los datos generales de una misma persona en múltiples representaciones del sistema.

El RUT se utiliza como clave natural en las entidades personales y se almacena como una cadena de caracteres. Esta decisión facilita la identificación dentro del contexto chileno y permite relacionar directamente los distintos perfiles. Como contrapartida, exige mantener criterios uniformes de formato, validación y normalización.

La relación entre personas y roles no se representa mediante un único campo en Persona. Para ello se utiliza PersonaRol, cuya clave compuesta considera RUT, RolId y ColegioId. Esta estructura permite asociar a una persona con un rol específico dentro de un establecimiento y evita reducir todo el modelo de autorización a una categoría global.

Las entidades Unidad, Sesion, Pregunta y Alternativa representan la estructura del contenido educativo. Desempeno almacena el resultado de un estudiante para una determinada sesión, mientras que RegistroEmocional conserva las observaciones previas y posteriores a la actividad.

RegistroIA mantiene las interacciones vinculadas al chatbot, proporcionando trazabilidad sobre los mensajes intercambiados. Las estructuras relacionadas con Glims permiten administrar el saldo del estudiante y la adquisición de elementos cosméticos.

3.6.2 Integridad y restricciones

Las claves primarias identifican de manera inequívoca los registros, mientras que las claves foráneas mantienen las asociaciones entre entidades dependientes. Por ejemplo, un desempeño debe vincularse con un estudiante y una sesión existentes.

La clave compuesta de PersonaRol impide repetir una misma combinación de persona, rol y establecimiento. De esta manera, parte de la consistencia se establece directamente en la estructura persistente y no depende únicamente de comprobaciones realizadas en Python.

Los enumerados restringen atributos cuyo dominio se encuentra previamente definido. Entre ellos se consideran valores asociados con género, estado, tipo de pregunta y tipo de emoción. El uso de enumeraciones disminuye la posibilidad de representar un mismo concepto mediante valores escritos de manera diferente.

Sin embargo, la integridad referencial no resulta suficiente para validar todos los procesos. La base de datos puede garantizar que un registro apunte a una sesión existente, pero la lógica de negocio debe comprobar adicionalmente que el estudiante pueda acceder a ella, que el ciclo previo se encuentre completo y que una recompensa no haya sido asignada anteriormente.

3.6.3 Uso de SQLAlchemy

Los modelos declarativos de SQLAlchemy representan las tablas como clases y permiten acceder a sus relaciones desde Python. Esto reduce la cantidad de consultas SQL repetidas y mantiene una correspondencia reconocible entre las entidades del dominio y el esquema relacional.

En distintos modelos se utiliza synonym, por ejemplo mediante expresiones equivalentes a `rut = synonym("RUT")`. Este mecanismo permite conservar compatibilidad entre el nombre real de una columna y el atributo utilizado por la aplicación.

Determinadas relaciones utilizan la estrategia `lazy="joined"`. Esta configuración recupera en una misma operación registros que habitualmente se requieren juntos, como un test, sus preguntas y las alternativas correspondientes.

La carga anticipada reduce consultas adicionales durante el desarrollo de una actividad. No obstante, su aplicación debe mantenerse limitada a relaciones donde los datos asociados sean utilizados de manera frecuente, puesto que una carga indiscriminada también puede recuperar información innecesaria.

3.6.4 Deuda técnica en la persistencia

Las estructuras MiniJuego y MiniJuegoItem se consultan mediante SQL directo, sin encontrarse completamente integradas al conjunto de modelos ORM.

Esta solución permitió incorporar la funcionalidad durante el periodo de desarrollo, pero genera una diferencia respecto del patrón utilizado en el resto de la aplicación. Las consultas, relaciones y validaciones de los minijuegos quedan separadas del modelo principal, lo que aumenta el esfuerzo de mantenimiento.

El uso de SQL directo no constituye por sí mismo una práctica incorrecta. Puede resultar apropiado para consultas específicas o para operaciones donde se requiere control detallado. En este caso, la deuda técnica se encuentra en la ausencia de una representación consistente del dominio.

Una evolución posterior debería incorporar modelos ORM equivalentes, definir sus relaciones con sesiones y actividades, y encapsular sus consultas en un módulo específico. Esta mejora conservaría el comportamiento actual, pero reduciría la dispersión de responsabilidades.

3.7 Flujo de autenticación y control de acceso

La solución implementa mecanismos diferentes para la plataforma web y la aplicación móvil. Esta separación responde a que ambos clientes utilizan patrones de interacción distintos, pero también produce diferentes niveles de protección que deben describirse con precisión.

3.7.1 Autenticación web mediante sesión firmada

El ingreso a la plataforma web comienza mediante una solicitud POST /login. El servidor recibe el RUT y la contraseña, consulta la información de Persona y comprueba la relación correspondiente en PersonaRol.

Cuando los datos coinciden, se almacenan en la sesión valores como el RUT y el rol. Las solicitudes posteriores pueden consultar esta información para determinar si el usuario inició sesión y qué tipo de vistas u operaciones puede utilizar.

El estado de la sesión se administra mediante SessionMiddleware de Starlette. La información se mantiene en una cookie firmada con apoyo de itsdangerous. La firma permite detectar alteraciones efectuadas por el cliente.

Es importante diferenciar firma y cifrado. La sesión firmada protege la integridad de la información, pero no significa que su contenido se encuentre cifrado. Por este motivo, la cookie no debe utilizarse para almacenar contraseñas ni información confidencial innecesaria.

Después de autenticar al usuario, el sistema realiza una redirección según el rol asociado. Este comportamiento permite distinguir las experiencias de estudiante y profesor desde el servidor, sin depender únicamente de que la interfaz oculte determinadas opciones.

3.7.2 Mecanismo de acceso móvil

La aplicación móvil utiliza el endpoint POST /mobile/login. Esta operación valida la combinación de RUT y Colegioid en PersonaRol. Si la relación existe, el servidor devuelve información como usuario_rut y colegio_id.

Este mecanismo permite identificar al usuario y su contexto institucional para habilitar la integración inicial con Flutter. Sin embargo, no emite un token firmado ni solicita una credencial secreta equivalente a la utilizada en la web.

Por esta razón, debe interpretarse como una verificación básica de identidad y pertenencia institucional, no como un mecanismo completo de autenticación apto para un entorno productivo.

3.7.3 Limitaciones de seguridad actuales

La principal limitación corresponde al tratamiento de las contraseñas. En la versión implementada, la comparación se realiza directamente y no se utiliza un algoritmo de hashing adaptativo como bcrypt o Argon2. En consecuencia, no corresponde afirmar que las credenciales se encuentran cifradas o almacenadas de manera robusta.

La segunda limitación es la ausencia de JWT u otro mecanismo equivalente para la aplicación móvil. Después del ingreso, las solicitudes no incorporan un token criptográficamente verificable que demuestre la autenticación del usuario.

Asimismo, la configuración de CORS utiliza allow_origins=["*"], condición apropiada únicamente durante etapas de desarrollo controladas. Para un despliegue real debería restringirse el origen de las solicitudes a los dominios o aplicaciones autorizados.

Estas limitaciones no impiden demostrar los flujos funcionales del prototipo, pero delimitan su preparación para un ambiente productivo. El fortalecimiento del acceso mediante hashing, tokens, expiración de sesiones, restricción de orígenes y comunicación HTTPS se mantiene como trabajo futuro prioritario.

3.8 Flujo de autorregulación y evaluación

Uno de los procesos centrales implementados por el Back-End corresponde al flujo que relaciona el registro emocional con el desarrollo de una actividad educativa.

La secuencia contempla un registro previo, la presentación de preguntas, el cálculo del desempeño y un registro emocional posterior. Desde el punto de vista del servidor, cada etapa representa una operación distinta que debe conservar el contexto de la sesión y validar la continuidad del flujo.

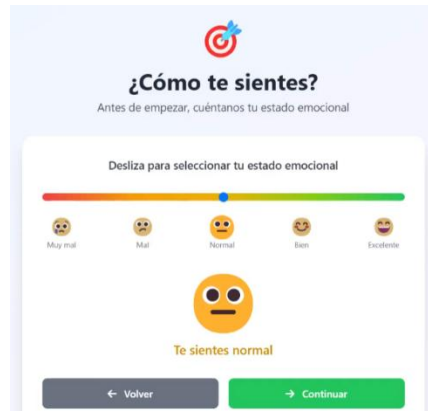


Figura 3.3. Registro emocional previo al desarrollo de una sesión de aprendizaje. Fuente: elaboración propia.

La Figura 3.3 corresponde a la interfaz utilizada antes del inicio de la actividad. Su relevancia para el Back-End se encuentra en la solicitud que genera: el valor seleccionado debe asociarse con el estudiante, la sesión educativa y el tipo de registro emocional previo.

El flujo implementado se desarrolla mediante las siguientes operaciones:

1. GET /desafio/precheck/{sesion_id} presenta el formulario emocional inicial.
2. POST /desafio/precheck/{sesion_id} valida la entrada y crea un RegistroEmocional de tipo pre.
3. GET /desafio/test/{sesion_id}?i=N recupera la pregunta ubicada en la posición indicada.
4. POST /desafio/test/{sesion_id} recibe la respuesta y la incorpora temporalmente a la sesión HTTP.
5. GET /desafio/test/finish/{sesion_id} recupera las respuestas, calcula el puntaje, actualiza el desempeño y asigna los Glims correspondientes.
6. GET /desafio/postcheck/{sesion_id} presenta el formulario emocional posterior.
7. POST /desafio/postcheck/{sesion_id} registra el estado final y ejecuta el análisis emocional asociado.



Figura 3.4. Ejemplos de preguntas procesadas durante el flujo de evaluación: (a) pregunta de alternativa y (b) pregunta de desarrollo. Fuente: elaboración propia.

Las respuestas no se almacenan individualmente en la base de datos cada vez que el estudiante avanza. Se mantienen temporalmente en una entrada de sesión equivalente a:

```
request.session["answers_{sesion_id}"]
```

La sesión web se respalda mediante una cookie firmada. Cuando el estudiante finaliza la actividad, el Back-End recupera el conjunto de respuestas, ejecuta `compute_score()`, actualiza el desempeño, calcula los Glims y elimina la información temporal.

Esta estrategia evita que una actividad incompleta se consolide automáticamente como resultado definitivo. También concentra la actualización del desempeño en el momento de cierre. Sin embargo, presenta dos limitaciones.

La primera es que una interrupción antes de finalizar puede provocar la pérdida de las respuestas temporales. La segunda es que el tamaño disponible en una cookie resulta adecuado únicamente para actividades acotadas, no para respuestas extensas o grandes volúmenes de información.

Además, la operación `/finish/` utiliza el método GET a pesar de producir modificaciones persistentes. Aunque el flujo funciona, desde una perspectiva de diseño HTTP sería preferible utilizar POST para una operación que actualiza el estado del sistema. Esta condición se reconoce como una mejora futura del contrato de la API.

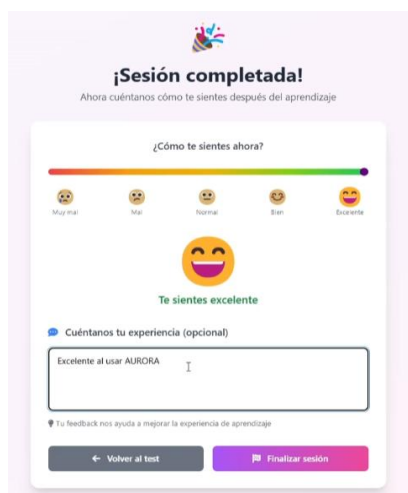


Figura 3.5. Registro emocional posterior al cierre de una sesión de aprendizaje. Fuente: elaboración propia.

El postcheck completa el ciclo al registrar cómo se siente el estudiante después de la actividad. La información previa y posterior se conserva como contexto para la reflexión y el seguimiento docente. Estos registros no se utilizan para emitir diagnósticos psicológicos.

3.9 Algoritmo de cálculo de puntaje

El cálculo se realiza mediante la función `compute_score()`. Debido a que Aurora incorpora diferentes tipos de preguntas, el procedimiento aplica una regla específica según el formato evaluado.

Sea p_i el puntaje máximo asignado a la pregunta i , r_i la respuesta proporcionada por el estudiante y S_i el puntaje obtenido. El resultado total se determina mediante:

$$S_{\text{total}} = \sum_{i=1}^n S_i$$

3.9.1 Preguntas de alternativa

En una pregunta de alternativa, el valor seleccionado se compara directamente con la respuesta correcta:

$$S_i = \begin{cases} p_i, & \text{si } r_i = c_i \\ 0, & \text{si } r_i \neq c_i \end{cases}$$

donde c_i representa la alternativa correcta.

Esta regla entrega un resultado determinista y resulta adecuada para preguntas que poseen una única opción válida.

3.9.2 Preguntas de desarrollo

En las preguntas de desarrollo, la versión implementada verifica únicamente que exista una respuesta no vacía:

$$S_i = \begin{cases} p_i, & \text{si } r_i \neq \emptyset \\ 0, & \text{si no se proporcionó respuesta} \end{cases}$$

Esta regla debe interpretarse como un criterio de participación o completitud. El sistema no comprueba automáticamente la calidad, exactitud ni pertinencia conceptual del texto ingresado.

La decisión evita utilizar un mecanismo semántico no validado para asignar calificaciones. Sin embargo, limita la capacidad de Aurora para medir conocimientos mediante respuestas abiertas. Una evolución futura podría incorporar revisión docente, rúbricas o un componente de asistencia semántica previamente validado.

3.9.3 Preguntas de clasificación y minijuegos

Las actividades de clasificación otorgan puntaje proporcional según el número de elementos ubicados correctamente.

Si a_i corresponde a los aciertos posicionales y t_i representa el total de elementos, entonces:

$$S_i = p_i \left(\frac{a_i}{t_i} \right)$$

La regla permite reconocer respuestas parcialmente correctas. Por ejemplo, si una actividad asigna cuatro elementos y el estudiante clasifica correctamente tres, obtiene tres cuartas partes del puntaje máximo.

El procedimiento debe comprobar que t_i sea mayor que cero para evitar una división inválida. Asimismo, el número de aciertos no puede superar el total de elementos disponibles.

3.10 Desbloqueo progresivo de unidades

El avance del estudiante se controla mediante la función `is_unit_unlocked()`. La primera unidad permanece disponible de manera predeterminada, mientras que el acceso a las siguientes depende de la finalización de todas las sesiones de la unidad inmediatamente anterior.

Una sesión se considera completa cuando existen tres evidencias:

- un RegistroEmocional de tipo pre;
- un registro de Desempeno;
- un RegistroEmocional de tipo post.

Formalmente, para una sesión s :

$$C(s) = Pre(s) \wedge Desempeno(s) \wedge Post(s)$$

Para una unidad $u > 1$, la disponibilidad se expresa como:

$$Disponible(u) = \bigwedge_{s \in Sesiones(u-1)} C(s)$$

Esto significa que la unidad posterior se desbloquea únicamente cuando todas las sesiones de la unidad previa cumplen la condición completa.

La regla evita que el avance dependa solo de responder preguntas. El estudiante debe registrar su estado inicial, desarrollar la actividad, obtener un resultado y cerrar el proceso mediante el postcheck.

Esta secuencia operacionaliza dentro del sistema un ciclo compatible con las fases de anticipación, ejecución y autorreflexión presentadas por (Zimmerman, 2000). No demuestra por sí misma una mejora en autorregulación, pero asegura que el recorrido funcional de Aurora incluya esas tres instancias.

3.11 Gamificación y economía de Glims

Los Glims constituyen la moneda virtual utilizada en Aurora. El Back-End administra su asignación y uso, puesto que una recompensa controlada únicamente por la interfaz podría alterarse, duplicarse o aplicarse de manera inconsistente.

La cantidad de Glims agregada al finalizar el test principal depende de la diferencia entre el nuevo resultado y el mejor resultado anterior:

$$G_{\text{agregados}} = \max(0, S_{\text{nuevo}} - S_{\text{mejor anterior}})$$

Esta regla se encuentra implementada en el flujo principal de evaluación. Los minijuegos utilizan procedimientos independientes y no aplican de manera uniforme la comparación con el mejor resultado anterior, lo que constituye una limitación de la versión desarrollada.

Este mecanismo actúa como protección contra el *farming*, entendido como la repetición mecánica de una actividad con el único propósito de acumular moneda. La recompensa se asocia con la mejora del desempeño y no con la cantidad de intentos.

La actualización se realiza junto con el registro del desempeño. La lógica comprueba el mejor puntaje previo, calcula la diferencia positiva y agrega los Glims correspondientes.

La compra de elementos presenta comportamientos diferentes según el canal utilizado. Tanto la implementación web como la móvil validan la existencia del elemento y el saldo disponible del estudiante. Sin embargo, el endpoint móvil incorpora un bloqueo de fila mediante SELECT ... FOR UPDATE y comprueba si el elemento ya forma parte del inventario. Cuando esto ocurre, la operación responde sin efectuar un nuevo cobro, proporcionando idempotencia frente a solicitudes repetidas.

En la plataforma web, una nueva compra del mismo elemento incrementa la cantidad almacenada en el inventario y no utiliza el mismo mecanismo de bloqueo. En consecuencia, la protección frente a solicitudes duplicadas y el control transaccional

reforzado no se aplican de manera uniforme en ambos canales. Esta diferencia constituye una limitación de consistencia que debe corregirse en una evolución posterior.

3.12 Integración del Back-End con el módulo de inteligencia artificial

Aurora incorpora funcionalidades de inteligencia artificial destinadas al acompañamiento conversacional, la generación de retroalimentación emocional y la elaboración de resúmenes para docentes. Estas capacidades fueron desarrolladas dentro de otro subproyecto del equipo y, por lo tanto, su implementación interna no forma parte del aporte realizado en la presente memoria.

Desde la perspectiva del Back-End, la responsabilidad consiste en facilitar la comunicación entre las interfaces de Aurora, la información persistida y el componente encargado de generar las respuestas. Para ello, el servidor recibe los datos enviados por los clientes, recupera el contexto requerido desde la base de datos y coordina la entrega o almacenamiento de los resultados producidos por el módulo de inteligencia artificial.

Esta separación permite mantener delimitadas las responsabilidades del proyecto. Mientras el módulo especializado concentra la generación y tratamiento de las respuestas, el Back-End administra los datos, las rutas de comunicación y la persistencia necesaria para integrar la funcionalidad con el resto de la plataforma.

Entre las interacciones soportadas se encuentran el acceso al historial conversacional, la asociación de retroalimentaciones con registros emocionales y la recuperación de antecedentes utilizados por el dashboard docente. En estos procesos, el aporte del Back-End se relaciona con la disponibilidad y consistencia de la información, y no con la lógica interna utilizada para generar el contenido.

Las respuestas producidas por el módulo de inteligencia artificial se utilizan como apoyo educativo. No constituyen diagnósticos psicológicos ni reemplazan la interpretación pedagógica del docente.

Tabla 3.3. Usos de GPT-4o-mini en el Back-End de Aurora.

Funcionalidad	Entrada Principal	Salida	Límite configurado	Persistencia
Chatbot Aurora	Mensaje del estudiante e historial	Respuesta conversacional	300 tokens	RegistroIA
Análisis emocional individual	Escala emocional y comentario	Retroalimentación contextual	200 tokens	RegistroEmocional.Resumen
Resumen para docentes	Historial emocional	Síntesis para el dashboard	Sin máximo explícito	No persistido

	del estudiante			
--	-------------------	--	--	--

Fuente: elaboración propia.

3.12.1 Integración con el chatbot

El Back-End proporciona las rutas y mecanismos de persistencia necesarios para relacionar las interacciones del chatbot con los usuarios de Aurora. Los mensajes y respuestas pueden conservarse en RegistroIA, permitiendo recuperar el historial asociado al estudiante.

La generación de la respuesta corresponde al módulo de inteligencia artificial desarrollado por otro integrante del proyecto. Por consiguiente, el alcance de este subproyecto se concentra en el intercambio de datos y en su persistencia dentro de la plataforma.

3.12.2 Integración con el registro emocional

Durante el postcheck, el Back-End almacena el estado emocional y el comentario ingresado por el estudiante. Esta información puede ser utilizada por el módulo de inteligencia artificial para producir una retroalimentación contextual.

La responsabilidad del Back-End consiste en entregar los datos necesarios y asociar el resultado con el registro correspondiente. El análisis generado no se considera una evaluación clínica ni un diagnóstico sobre el estudiante.

3.12.3 Integración con el dashboard docente

El dashboard docente utiliza información histórica almacenada por el Back-End para presentar antecedentes de un estudiante seleccionado. Cuando se solicita una síntesis mediante inteligencia artificial, el servidor recupera los registros necesarios y coordina la comunicación con el módulo correspondiente.

La interpretación de los antecedentes continúa siendo responsabilidad del docente. La inteligencia artificial actúa únicamente como herramienta de apoyo para organizar o resumir la información disponible.

3.13 Gestión de archivos y seguridad de rutas

La plataforma incorpora carga de materiales mediante formularios procesados con python-multipart. El Back-End recibe los archivos, construye su ubicación de almacenamiento y registra los antecedentes correspondientes en la base de datos.

Para disminuir las colisiones entre archivos, cada operación utiliza una carpeta identificada mediante un UUID. Este identificador actúa como nombre de lote y evita depender únicamente del nombre original proporcionado por el usuario.

La operación involucra dos recursos distintos: el sistema de archivos y la base de datos. Debido a que ambos no participan en una única transacción, se implementa una estrategia compensatoria.

Si el archivo se guarda correctamente, pero posteriormente falla la operación sobre la base de datos, el servidor elimina la carpeta creada mediante `shutil.rmtree`. Esta limpieza reduce la acumulación de recursos huérfanos.

Para recuperar materiales se utiliza una función equivalente a `safe_join_uploads()`. Su propósito es construir rutas dentro del directorio autorizado y prevenir intentos de *path traversal*.

Sin esta comprobación, una entrada que incluyera secuencias como `../` podría intentar acceder a ubicaciones externas al directorio de cargas. La validación de la ruta permite rechazar o normalizar estas referencias antes de acceder al sistema de archivos.

La implementación actual depende del almacenamiento local del servidor. Esta alternativa resulta suficiente para el entorno académico, pero limita un eventual despliegue distribuido. Si la aplicación se ejecutara en múltiples instancias, sería necesario centralizar los archivos en un repositorio compartido o servicio de almacenamiento de objetos.

3.14 Gestión de sesiones de base de datos y consistencia transaccional

La dependencia `get_db()` administra el ciclo de vida de las sesiones SQLAlchemy. Cada operación que necesita acceder a MySQL recibe una sesión creada específicamente para la solicitud.

La implementación utiliza `yield` para entregar el recurso y un bloque `finally` para cerrarlo. De esta manera, la conexión se libera aun cuando se produzca una excepción durante el procesamiento.

El motor se configura mediante `pool_pre_ping=True`. Antes de reutilizar una conexión almacenada en el pool, SQLAlchemy comprueba que continúe disponible. Esta medida disminuye errores provocados por conexiones que fueron cerradas durante un periodo de inactividad.

En las operaciones transaccionales se distinguen `flush()` y `commit()`. `flush()` envía las modificaciones pendientes a la base de datos sin finalizar la transacción. Esto permite obtener identificadores generados o evaluar restricciones antes de confirmar todos los cambios.

`commit()`, en cambio, confirma definitivamente la transacción. Si una operación falla antes de este punto, las modificaciones pueden revertirse mediante un `rollback`.

La función encargada de actualizar el desempeño y los puntos utiliza esta secuencia para mantener coherencia entre ambas operaciones. Primero se inserta o actualiza el resultado, posteriormente se sincronizan los cambios con `flush()`, se obtiene el bloqueo necesario para la actualización de puntos y finalmente se ejecuta el `commit()`.

En determinados flujos se utiliza `SELECT ... FOR UPDATE` para bloquear temporalmente la fila del estudiante mientras se actualizan sus puntos o se procesa una compra móvil. Este mecanismo se encuentra presente en las funciones de actualización de resultados y en algunos endpoints móviles, pero no se aplica de manera uniforme en todos los flujos web.

En los componentes donde se utiliza, el bloqueo reduce el riesgo de que dos transacciones lean el mismo saldo y ejecuten modificaciones incompatibles. La ausencia de este mecanismo en otros procedimientos, como la compra web y la finalización web del test, representa una oportunidad de fortalecimiento transaccional.

Estas decisiones muestran que la consistencia no depende solamente del orden de las instrucciones en Python. También requiere administrar explícitamente el comportamiento transaccional de MySQL y SQLAlchemy.

3.15 Documentación automática mediante Swagger y OpenAPI

FastAPI genera automáticamente una especificación OpenAPI a partir de las rutas, parámetros y modelos declarados. Esta especificación se presenta mediante una interfaz Swagger que permite explorar los endpoints y ejecutar solicitudes controladas.

En Aurora, esta documentación facilitó la coordinación entre el Back-End y los clientes que consumen sus servicios. Los desarrolladores de la aplicación móvil pudieron consultar los parámetros requeridos, las estructuras JSON esperadas y los códigos de respuesta definidos.

La documentación automática reduce la dependencia de descripciones informales que pueden quedar desactualizadas. Cuando una ruta o esquema cambia, la especificación refleja la declaración disponible en el código.

No obstante, Swagger no reemplaza una estrategia de pruebas. La ejecución individual de un endpoint permite comprobar su contrato básico, pero no garantiza que un flujo completo mantenga correctamente el estado, las transacciones o las relaciones entre operaciones.

En un despliegue productivo también sería necesario evaluar si la documentación debe mantenerse públicamente accesible o restringirse, especialmente cuando contenga información sobre rutas internas.

3.16 Mantenibilidad, capacidad de evolución y limitaciones actuales

La arquitectura monolítica modular resulta adecuada para el alcance académico del proyecto. Mantener una única unidad de ejecución simplifica el despliegue y evita introducir mecanismos distribuidos que no son necesarios para implementar los flujos actuales.

La división en routers, modelos y funciones auxiliares permite modificar determinadas áreas sin intervenir directamente en toda la aplicación. Por ejemplo, la lógica de cálculo

puede evolucionar dentro del módulo de evaluación, mientras que las rutas conservan el mismo contrato general.

Sin embargo, la mantenibilidad todavía puede fortalecerse. Parte de los routers conserva coordinaciones y consultas específicas que podrían trasladarse a servicios más explícitos. Asimismo, los minijuegos permanecen parcialmente desacoplados del ORM.

Entre las principales limitaciones técnicas reconocidas se encuentran:

- comparación directa de contraseñas sin hashing robusto;
- ausencia de JWT para la aplicación móvil;
- consultas SQL directas para MiniJuego y MiniJuegoItem;
- coexistencia de patrones correspondientes a Pydantic v1 y v2;
- RUT de profesor codificado directamente en una parte del dashboard emocional;
- configuración CORS abierta durante el desarrollo;
- almacenamiento de archivos en el sistema local;
- almacenamiento temporal de respuestas en la cookie de sesión;
- utilización de GET para una operación de finalización que modifica datos;
- clave de firma de la sesión definida directamente en el código fuente;
- ausencia de pruebas automatizadas en el repositorio;
- aplicación no uniforme de bloqueos de fila entre los flujos web y móviles;
- ausencia de la regla anti-farming en determinados minijuegos;
- falta de validación del tipo y tamaño de los archivos cargados.

Estas condiciones no invalidan los flujos demostrados, pero delimitan el nivel de preparación de la versión actual. La aplicación constituye un prototipo funcional integrado, no una plataforma lista para operar sin modificaciones en un ambiente productivo con información real de establecimientos.

Tampoco se realizaron pruebas de carga que permitan afirmar una capacidad determinada de usuarios concurrentes. Por ello, la escalabilidad debe entenderse como una posibilidad de evolución favorecida por la organización modular y no como una característica cuantitativamente validada.

En una etapa futura, algunos componentes podrían separarse si el crecimiento lo justificara. La integración con inteligencia artificial podría ejecutarse mediante procesos diferidos; la gestión de archivos podría trasladarse a almacenamiento externo; y la

autenticación podría concentrarse en un módulo con políticas de seguridad más completas.

Esta evolución no requiere adoptar microservicios de manera inmediata. La prioridad consiste en fortalecer los límites internos, corregir la deuda técnica y validar el comportamiento del sistema bajo condiciones de uso definidas.

3.17 Cierre del capítulo

En este capítulo se presentó la solución implementada para el subproyecto Back-End de Aurora. La arquitectura adoptada permite atender desde una misma aplicación FastAPI los endpoints REST, las solicitudes de la aplicación móvil y las vistas web renderizadas mediante Jinja2.

La organización mediante routers, funciones auxiliares y modelos ORM proporciona una separación funcional entre recepción de solicitudes, reglas de negocio y persistencia. Sobre esta base se implementaron los procesos de autenticación web, verificación móvil, registro emocional, evaluación, cálculo de puntaje, desbloqueo de unidades, asignación de Glms e integración con el módulo de inteligencia artificial desarrollado dentro del proyecto grupal.

El flujo de autorregulación vincula el precheck, el desarrollo de la actividad y el postcheck mediante operaciones coordinadas por el servidor. Asimismo, la economía de Glms incorpora reglas orientadas a recompensar la mejora y mecanismos transaccionales para disminuir inconsistencias.

La implementación también presenta limitaciones relevantes. La ausencia de hashing y tokens, la existencia de consultas SQL directas, el uso de almacenamiento local y determinadas decisiones de contrato HTTP deben fortalecerse antes de considerar un despliegue productivo.

El reconocimiento de estas limitaciones permite delimitar el alcance alcanzado sin disminuir el aporte del subproyecto. Aurora dispone de una infraestructura Back-End capaz de sostener sus principales flujos funcionales, mantener relaciones persistentes y articular los componentes desarrollados por el equipo.

Una vez descrita la solución, corresponde comprobar que sus componentes se comporten de acuerdo con los resultados esperados. Por esta razón, el capítulo siguiente aborda la validación funcional y técnica del Back-End, junto con las instancias exploratorias en las que Aurora fue presentada ante estudiantes, docentes y otros participantes.

CAPÍTULO 4: VALIDACIÓN Y RESULTADOS DEL BACK-END DE AURORA

Una vez presentada la arquitectura, los componentes y las principales decisiones de implementación del Back-End de Aurora, corresponde comprobar que la solución desarrollada responde de manera coherente a los requerimientos funcionales definidos para el proyecto. En este capítulo se presentan la estrategia de validación aplicada y los resultados obtenidos mediante la ejecución de pruebas representativas sobre los principales procesos gestionados por el servidor.

La validación se concentra en el aporte específico del subproyecto Back-End. Por esta razón, las comprobaciones consideran aspectos como la autenticación y el control de acceso, la persistencia de información, la gestión de materiales, el registro emocional, la ejecución de evaluaciones, el cálculo de puntajes, la actualización del desempeño, la asignación de Glims y el desbloqueo progresivo de unidades. También se examina la integración técnica con el servicio de OpenAI y el comportamiento del sistema ante solicitudes inválidas o servicios externos no disponibles.

Junto con estas comprobaciones funcionales y técnicas, se incorporan dos instancias de carácter cualitativo: una experiencia exploratoria realizada con estudiantes en un establecimiento educacional y la presentación del proyecto en una feria tecnológica. Estas actividades permiten describir la recepción inicial de Aurora y observar su funcionamiento como sistema integrado. Sin embargo, debido a que no se utilizaron instrumentos estandarizados ni se efectuaron mediciones longitudinales, sus resultados no se interpretan como evidencia concluyente de una mejora en la autorregulación o en el rendimiento académico.

En consecuencia, el capítulo distingue entre la validación objetiva de las funciones implementadas en el Back-End y la apreciación exploratoria obtenida en contextos de demostración. Esta separación permite presentar los resultados con un alcance preciso, evitando atribuir al sistema efectos que no fueron medidos durante el desarrollo.

4.1 Estrategia de validación

La estrategia de validación tuvo como objetivo comprobar que los componentes desarrollados para el Back-End de Aurora ejecutaran correctamente las operaciones necesarias para sostener el flujo funcional de la plataforma. Para ello, se seleccionaron casos representativos de los procesos principales, priorizando aquellos en los que el servidor aplica reglas de negocio, persiste información o coordina la comunicación entre las interfaces cliente y la base de datos.

Las pruebas funcionales y técnicas se ejecutaron en un entorno local controlado. La aplicación desarrollada con FastAPI fue ejecutada mediante el servidor ASGI Uvicorn, utilizando MySQL como sistema de gestión de base de datos. La interacción con las funcionalidades se realizó principalmente desde la plataforma web de Aurora, mientras

que Swagger UI, generada a partir de la especificación OpenAPI, fue utilizada para comprobar determinados endpoints. La información persistida se verificó mediante consultas de control ejecutadas directamente sobre MySQL.

Cada caso de prueba fue identificado mediante un código interno, desde CP-00 hasta CP-20. Estos códigos se conservaron para mantener la trazabilidad entre la funcionalidad evaluada, el resultado obtenido y las capturas recopiladas. No obstante, los resultados se presentan en este capítulo agrupados temáticamente, en lugar de exponer una sucesión completa de casos, con el propósito de construir una explicación continua y evitar una sobrecarga de evidencias visuales. La matriz consolidada se presenta en la Tabla A.1 del Anexo A; las capturas complementarias, en las Figuras B.1 a B.13 del Anexo B; y el inventario completo, junto con la ubicación de las evidencias, en la Tabla C.1 del Anexo C.

El procedimiento aplicado dependió de la naturaleza de cada funcionalidad. En las operaciones que modificaban información se utilizó, cuando resultaba pertinente, una consulta previa para establecer el estado inicial de los datos. Posteriormente se ejecutó la acción correspondiente desde la plataforma y se realizó una segunda consulta para comprobar el estado resultante. Este método se empleó, por ejemplo, en la validación del registro emocional, la actualización del desempeño y la asignación de Glims.

En las funciones asociadas con autenticación, control de acceso, navegación y gestión de materiales, se verificó la respuesta observable desde la interfaz y la continuidad del flujo según el rol del usuario. Para las solicitudes inválidas se examinó el mensaje generado por el Back-End, mientras que, en el caso de la integración con OpenAI, se consideró tanto la solicitud realizada como el manejo de la indisponibilidad del servicio externo y la persistencia del resultado en el historial correspondiente.

Las modificaciones observadas en la base de datos fueron producidas mediante las operaciones normales de Aurora. No se alteraron manualmente los registros para simular resultados y las consultas directas sobre MySQL se utilizaron principalmente como mecanismos de lectura y control. De este modo, las evidencias representan el comportamiento real del flujo implementado.

Para clasificar los resultados se establecieron tres estados. Un caso se consideró aprobado cuando las evidencias permitieron comprobar completamente su criterio de aceptación. La categoría aprobado con alcance limitado se utilizó cuando se verificó una conducta específica, pero la evidencia disponible no permitía extender la conclusión a otras propiedades técnicas. Finalmente, se asignó el estado parcial cuando solamente se confirmó una parte del flujo, como la carga de una vista o la presentación de un mensaje, sin disponer de evidencia suficiente para acreditar todas las reglas de negocio o la persistencia asociada.

Las instancias consideradas en el proceso se resumen en la Tabla 4.1.

Tabla 4.1. Resumen de las instancias de validación aplicadas al proyecto Aurora.

Instancia	Propósito	Procedimiento	Tipo de Evidencia	Alcance
Validación funcional y técnica del Back-End	Comprobar las funciones, reglas de negocio y operaciones de persistencia implementadas	Ejecución de casos representativos desde la plataforma web, Swagger y MySQL.	Capturas de interfaz, respuestas del servidor y consultas de control en la base de datos	Validación principal del subproyecto Back-End.
Validación exploratoria en contexto escolar	Observar la recepción inicial y la comprensión de la propuesta por parte de los estudiantes.	Presentación y utilización de Aurora con aproximadamente 40 estudiantes acompañada de retroalimentación espontánea.	Observaciones y comentarios cualitativos.	Evidencia exploratoria, sin medición experimental ni longitudinal.
Presentación en feria tecnológica	Observar el funcionamiento integrado de Aurora y recopilar apreciaciones de los asistentes.	Demostración del sistema ante estudiantes, docentes y otros participantes.	Retroalimentación y observaciones obtenidas durante la presentación.	Validación demostrativa y cualitativa.

Fuente: elaboración propia.

La combinación de estas instancias permite analizar el proyecto desde perspectivas complementarias. Las pruebas funcionales y técnicas proporcionan la evidencia principal para evaluar el Back-End, mientras que las actividades realizadas con usuarios aportan antecedentes sobre la aceptación y comprensión inicial de Aurora. En las secciones siguientes se presentan separadamente sus resultados, comenzando por el antecedente diagnóstico que dio origen a la propuesta.

4.2 Antecedente diagnóstico

Antes del desarrollo de Aurora se realizó un levantamiento diagnóstico orientado a conocer las necesidades y dificultades presentes en el contexto educativo abordado por el proyecto. Esta instancia, presentada en el Capítulo 1, permitió recopilar antecedentes relacionados con los procesos de aprendizaje, la organización académica y la dimensión emocional de los estudiantes.

Los resultados obtenidos contribuyeron a fundamentar la necesidad de una solución tecnológica que integrara el acceso a contenidos educativos con mecanismos de seguimiento del progreso y registro emocional. A partir de estos antecedentes se definieron las principales orientaciones funcionales de Aurora, entre ellas la organización del contenido por asignaturas, unidades y sesiones; la incorporación de evaluaciones; el seguimiento del desempeño; y el registro del estado emocional antes y después de las actividades académicas.

La encuesta diagnóstica debe entenderse como un antecedente utilizado para orientar el diseño del sistema y no como una validación de la solución implementada. Debido a que fue aplicada antes de disponer de una versión funcional de Aurora, sus resultados no permiten determinar efectos atribuibles al uso de la plataforma ni demostrar mejoras en la autorregulación o el rendimiento académico.

Por esta razón, el análisis detallado de sus resultados se mantiene en el capítulo donde se fundamenta el problema. En la presente sección se incorpora únicamente para establecer la relación entre las necesidades inicialmente identificadas y las funciones que posteriormente fueron sometidas a validación. La experiencia exploratoria desarrollada con estudiantes una vez disponible la plataforma se presenta separadamente en la sección siguiente.

4.3 Validación exploratoria en contexto escolar

Una vez disponible una versión funcional de Aurora, se realizó una actividad exploratoria en un establecimiento educacional con la participación de aproximadamente cuarenta estudiantes pertenecientes al público objetivo de la plataforma. Esta instancia tuvo como propósito observar la recepción inicial de la propuesta, la comprensión general de su funcionamiento y la interacción de los estudiantes con el flujo principal del sistema en un contexto cercano a aquel para el cual fue diseñado.

Durante la actividad, los estudiantes interactuaron con la plataforma y recorrieron las principales etapas de la experiencia educativa, entre ellas el registro del estado emocional, el desarrollo de las actividades académicas y el cierre de la sesión. Esto permitió observar el funcionamiento integrado de los componentes de Aurora y la manera en que los participantes comprendían la secuencia presentada por el sistema.

La recopilación de información se realizó mediante observación directa y comentarios espontáneos de los estudiantes. No se aplicó un cuestionario, una rúbrica de usabilidad ni otro instrumento estandarizado. Tampoco se efectuaron mediciones anteriores y

posteriores que permitieran comparar cambios en el desempeño, la motivación o la autorregulación de los participantes.

A partir de las observaciones realizadas, se identificó una recepción inicialmente positiva hacia la propuesta de Aurora. Los estudiantes pudieron reconocer el flujo general de interacción y manifestaron comentarios favorables respecto de la experiencia de uso. Estos antecedentes sugieren que la organización de la plataforma resultó comprensible durante la actividad y que su propuesta generó interés entre los participantes.

Los principales aspectos observados se sintetizan en la Tabla 4.2.

Tabla 4.2. Síntesis de la validación exploratoria realizada en contexto escolar.

Aspecto observado	Resultado de la actividad	Alcance de la observación
Interacción con la plataforma	Los estudiantes recorrieron el flujo que integraba el registro emocional, las actividades académicas y el cierre de la sesión.	Permitió observar la utilización general de Aurora en un contexto escolar.
Comprensión del flujo	Los participantes lograron reconocer la secuencia general presentada por la plataforma.	Corresponde a una observación cualitativa y no a una medición formal de usabilidad.
Recepción inicial	Se obtuvieron comentarios espontáneos favorables y se observó interés por la propuesta.	Sugiere una aceptación inicial, pero no representa un indicador cuantitativo de satisfacción.
Funcionamiento integrado	La actividad permitió utilizar Aurora como una solución compuesta por diferentes módulos y servicios.	Complementa las pruebas técnicas, pero no sustituye la validación específica de las funciones del Back-End.

Fuente: elaboración propia.

Debido al carácter exploratorio de la actividad, estos resultados no permiten concluir que el uso de Aurora produzca una mejora comprobada en la autorregulación escolar o en el rendimiento académico. Para establecer ese tipo de efecto habría sido necesario aplicar instrumentos validados, definir indicadores cuantificables, contar con mediciones comparativas y realizar un seguimiento durante un periodo más prolongado.

Asimismo, esta experiencia evaluó la plataforma como un sistema integrado y no exclusivamente el Back-End desarrollado en este subproyecto. Su aporte al proceso de validación consiste en demostrar que la solución pudo ser utilizada por estudiantes del público objetivo y en aportar una primera aproximación cualitativa sobre su comprensión y recepción. Las reglas de negocio, las operaciones de persistencia y los procesos

específicos gestionados por el Back-End se comprueban posteriormente mediante los casos de prueba funcionales y técnicos.

Por resguardo de la identidad de los participantes, no se incorporan fotografías en las que puedan reconocerse los rostros de los estudiantes. La descripción de la actividad se limita, por tanto, a información agregada y a observaciones que no permiten identificar individualmente a sus participantes.

4.4 Presentación en feria tecnológica

Aurora también fue presentado en una feria tecnológica ante estudiantes, docentes y asistentes vinculados con los ámbitos educativo y tecnológico. Esta instancia tuvo como propósito exponer la propuesta, observar el funcionamiento conjunto de sus componentes y recopilar apreciaciones externas sobre la solución desarrollada.

Durante la presentación se mostró la integración de las actividades educativas, el registro emocional, la gamificación y el apoyo proporcionado mediante inteligencia artificial. A diferencia de las pruebas funcionales ejecutadas de manera controlada, esta actividad permitió presentar Aurora como un sistema completo y observar la continuidad de su flujo en un contexto de demostración.

Desde la perspectiva del subproyecto Back-End, la presentación permitió comprobar de manera práctica que la infraestructura implementada podía sostener la interacción entre los clientes de usuario, las operaciones de persistencia, el flujo de evaluación, los registros emocionales y los servicios asociados con inteligencia artificial dentro de un mismo entorno operativo. De esta forma, el servidor proporcionó el soporte necesario para articular los diferentes componentes desarrollados por el equipo.

La recepción obtenida durante la feria fue favorable. Los asistentes valoraron positivamente la integración de las dimensiones académica y emocional, así como la incorporación de elementos de gamificación dentro de una plataforma educativa. Estas apreciaciones aportaron una referencia inicial sobre la comprensión y pertinencia de la propuesta frente a personas externas al equipo de desarrollo.

Los principales aspectos observados durante esta actividad se resumen en la Tabla 4.3.

Tabla 4.3. Síntesis de la presentación de Aurora en la feria tecnológica.

Aspecto observado	Resultado	Alcance
Presentación de la propuesta	Aurora fue expuesto ante estudiantes, docentes y otros asistentes.	Permitió comunicar el propósito y las funciones generales de la plataforma.
Funcionamiento integrado	Se presentó la interacción entre las actividades académicas, el registro emocional, la gamificación y los servicios de inteligencia artificial.	Permitió observar el sistema como una solución integrada.
Soporte del Back-End	El servidor sostuvo la comunicación con los clientes, la persistencia de datos y los procesos centrales de la plataforma.	Proporcionó una comprobación demostrativa de la integración técnica.
Recepción de los asistentes	Se recibieron apreciaciones favorables sobre la propuesta y la integración de sus componentes.	Corresponde a retroalimentación cualitativa, no a una medición formal de aceptación o usabilidad.

Fuente: elaboración propia.

Esta actividad no contó con un cuestionario, una rúbrica ni otro instrumento estructurado para registrar y analizar la opinión de los asistentes. Tampoco se efectuaron mediciones cuantitativas de usabilidad, rendimiento o impacto educativo. Por consiguiente, la retroalimentación obtenida debe interpretarse como evidencia exploratoria de aceptación y comprensión inicial, y no como una validación experimental de los efectos de Aurora.

Asimismo, debido a que la feria presentó el trabajo conjunto del equipo, sus resultados no pueden atribuirse exclusivamente al Back-End. Su importancia para este subproyecto radica en haber permitido observar la capacidad de la infraestructura desarrollada para sostener una demostración funcional integrada frente a usuarios externos.

Las experiencias descritas en esta sección y en la validación realizada en el establecimiento educacional complementan la evaluación del proyecto desde una perspectiva cualitativa. Sin embargo, la comprobación específica de las funciones, reglas de negocio y operaciones de persistencia del Back-End requiere evidencias controladas. Estas se presentan en la sección siguiente mediante los casos de validación funcional ejecutados sobre Aurora.

4.5 Validación funcional del Back-End

La validación funcional se orientó a comprobar que el Back-End ejecutara correctamente los procesos necesarios para sostener la interacción de los usuarios con Aurora. Para ello se evaluaron la autenticación y el control de acceso, la navegación por la estructura académica y la gestión de materiales didácticos.

Los casos fueron ejecutados desde la plataforma web, utilizando usuarios asociados con los roles Estudiante y Profesor. Las evidencias obtenidas permitieron observar las respuestas de la aplicación y la continuidad de cada flujo. Cuando la funcionalidad implicaba persistencia, la comprobación se complementó mediante consultas directas sobre MySQL.

En las subsecciones siguientes se presentan los resultados agrupados según la función evaluada. Las capturas principales se incorporan en el cuerpo del capítulo, mientras que las evidencias complementarias de navegación se presentan en las Figuras B.3 a B.8 del Anexo B.

4.5.1 Autenticación y control de acceso

La autenticación web constituye el punto de entrada a las funciones de Aurora. Su propósito es validar las credenciales ingresadas y establecer una sesión asociada con el usuario autenticado. Posteriormente, el Back-End consulta la relación registrada en PersonaRol para determinar el rol correspondiente y dirigir al usuario hacia las funciones habilitadas para su perfil.

Para evaluar este comportamiento se utilizaron los casos CP-01 y CP-08. El primero comprobó el rechazo de credenciales inválidas y el acceso de un usuario con rol Estudiante, mientras que el segundo verificó el acceso de un usuario con rol Profesor.

En la primera comprobación se ingresaron credenciales que no correspondían con un usuario válido. Como se observa en la Figura 4.1, la aplicación mantuvo al usuario en la pantalla de inicio de sesión y presentó el mensaje “RUT o contraseña inválidos”. Por consiguiente, no se permitió continuar hacia las áreas internas de la plataforma.

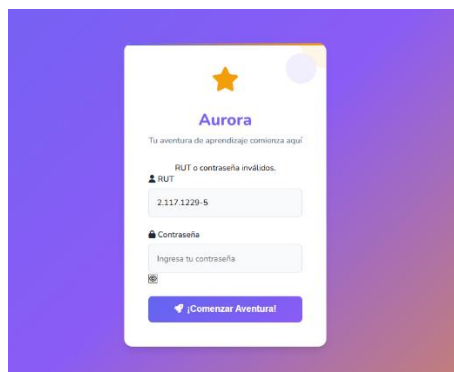


Figura 4.1 Rechazo de credenciales inválidas durante el inicio de sesión en Aurora.

Fuente: elaboración propia.

El resultado anterior permite comprobar el comportamiento esperado frente a un intento de autenticación incorrecto. La evidencia demuestra el rechazo funcional de las credenciales utilizadas, aunque no corresponde a una auditoría exhaustiva de seguridad ni permite evaluar por sí sola la resistencia del sistema ante otros tipos de ataques.

Posteriormente se realizó el inicio de sesión con las credenciales de un usuario asociado con el rol Estudiante. En este caso, Aurora permitió el acceso y redirigió al usuario hacia su panel principal, como muestra la Figura 4.2 La vista resultante presenta las funciones destinadas al estudiante, entre ellas el módulo de estudio, el acceso a los desafíos, la tienda, el inventario y el chatbot.

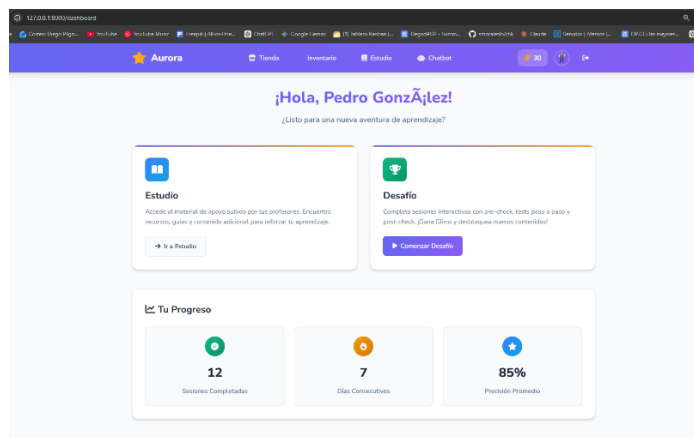


Figura 4.2 Inicio de sesión válido y redirección al panel correspondiente al rol Estudiante.

Fuente: elaboración propia.

La redirección obtenida confirma que el Back-End reconoció la relación del usuario con el rol Estudiante y permitió continuar hacia el flujo correspondiente. Esta comprobación resulta relevante porque las funciones posteriores de registro emocional, evaluación, desempeño y asignación de Glims dependen de que el usuario haya sido autenticado e identificado correctamente.

El acceso del Profesor se evaluó de manera independiente mediante el caso CP-08. Después de ingresar credenciales válidas, el sistema dirigió al usuario hacia el menú destinado a este rol. Como se observa en la Figura 4.3, la interfaz presenta funciones diferentes de las disponibles para el estudiante, específicamente la revisión del análisis emocional y la carga de materiales didácticos.

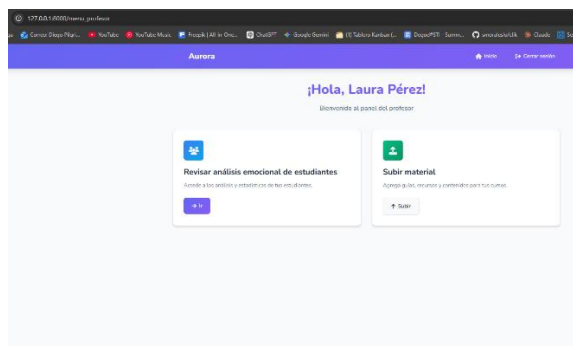


Figura 4.3 Inicio de sesión válido y acceso al menú correspondiente al rol Profesor.

Fuente: elaboración propia.

La diferencia entre las vistas obtenidas permite comprobar que Aurora no aplica una redirección única para todos los usuarios, sino que determina el destino de acuerdo con el rol asociado. De esta manera, el estudiante accede a las funciones vinculadas con su proceso de aprendizaje, mientras que el profesor dispone de las herramientas relacionadas con el seguimiento y la gestión de recursos educativos.

La Tabla 4.4 resume los resultados de los casos utilizados en esta subsección.

Tabla 4.4. Resultados de la validación de autenticación y control de acceso.

Caso	Comprobación	Resultado obtenido	Estado
CP-01	Rechazo de credenciales inválidas y acceso del Estudiante.	Las credenciales incorrectas fueron rechazadas y el usuario válido fue dirigido al panel del Estudiante.	Aprobado
CP-08	Acceso del Profesor.	El usuario válido fue dirigido al menú correspondiente al rol Profesor.	Aprobado

Fuente: elaboración propia.

En conjunto, los resultados permiten aprobar los casos CP-01 y CP-08. Se comprobó el rechazo de un intento de acceso inválido, la creación del flujo autenticado para usuarios válidos y la redirección diferenciada de acuerdo con los roles Estudiante y Profesor. Estas pruebas validan el comportamiento funcional observable del mecanismo de acceso; las limitaciones generales de seguridad de la implementación se analizarán posteriormente en la sección correspondiente.

4.5.2 Navegación académica

Una vez autenticado, el estudiante puede acceder a los contenidos y actividades organizados por Aurora. Para comprobar la continuidad de esta navegación se ejecutaron los casos CP-02 a CP-07, los cuales abarcan el módulo de materiales de estudio, la selección de asignaturas, la visualización de unidades y sesiones, el registro emocional previo y el acceso a la evaluación interactiva.

El caso CP-02 permitió comprobar la carga del módulo de materiales didácticos. Como se observa en la Figura 4.4, la vista presentó el recurso disponible junto con las opciones para abrirlo o descargarlo. Este resultado confirma que el usuario autenticado pudo acceder a la sección y consultar la información del material registrada por la plataforma.

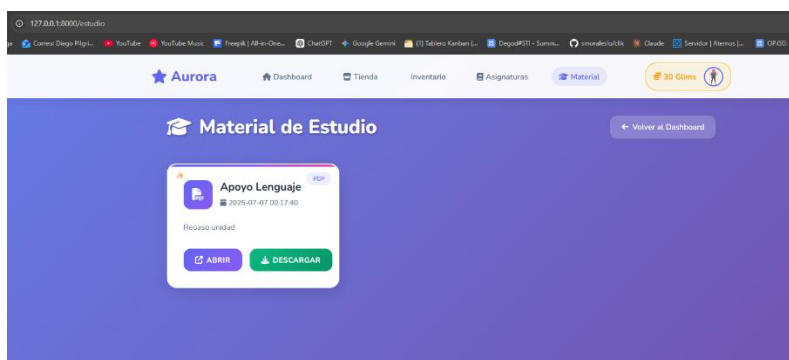


Figura 4.4. Visualización de los materiales didácticos disponibles para el estudiante.

Fuente: elaboración propia.

La Figura 4.4 también representa el estado del módulo antes de cargar el nuevo material utilizado posteriormente en el caso CP-09. Por esta razón, la misma evidencia servirá como referencia para comparar el estado anterior y posterior del proceso de publicación, sin necesidad de repetir la captura en la subsección siguiente.

La navegación hacia las actividades académicas se inició mediante la selección de una asignatura. En el caso CP-03, Aurora presentó las alternativas disponibles, entre ellas Matemáticas, Ciencias, Lenguaje e Historia. Al seleccionar una asignatura, el sistema permitió avanzar hacia la vista de unidades correspondiente.

El caso CP-04 permitió observar la aplicación inicial del esquema de progresión. En la asignatura seleccionada, la Unidad 1 se encontraba disponible, mientras que la Unidad 2 aparecía bloqueada. Esta evidencia muestra el estado inicial de la interfaz, pero la comprobación completa de la regla de desbloqueo requiere contrastarla con los registros de finalización y con el estado posterior de la unidad. Dicho comportamiento se analizará en la sección 4.6.5 mediante el caso CP-17.

Posteriormente, el caso CP-05 comprobó que la unidad disponible conducía hacia la vista de sesiones. La plataforma presentó la Sesión 1, su estado pendiente, el objetivo de

aprendizaje, la bitácora asociada y el control para iniciar la actividad. De este modo, se verificó la continuidad observable entre la selección de la asignatura, la unidad y la sesión correspondiente.

Al iniciar la sesión, Aurora dirigió al estudiante hacia el formulario emocional previo, evaluado inicialmente mediante el caso CP-06. La vista presentó cinco alternativas para representar el estado emocional del estudiante antes de comenzar la actividad. La aparición del formulario demuestra la continuidad de la navegación; su persistencia en MySQL se valida separadamente en la sección 4.6.1 mediante el caso CP-10.

Después de completar esta etapa, el estudiante pudo acceder a la evaluación interactiva. En el caso CP-07 se observó la presentación del contenido de apoyo, la pregunta correspondiente, sus alternativas y el control para continuar. Esta evidencia permite comprobar que el flujo alcanzó correctamente la evaluación. La finalización del test, el cálculo del resultado y la persistencia de sus efectos se examinan posteriormente mediante los casos CP-11 a CP-14.

Las capturas correspondientes a estas etapas intermedias se presentan como evidencias complementarias en las Figuras B.3 a B.8 del Anexo B. Esta decisión evita incorporar una imagen por cada pantalla y permite concentrar el cuerpo del capítulo en las reglas de negocio y operaciones de persistencia más relevantes para el Back-End.

La Tabla 4.5 resume los resultados obtenidos durante la validación de la navegación académica.

Tabla 4.5. Resultados de la validación de la navegación académica del estudiante.

Caso	Función evaluada	Resultado obtenido	Estado
CP-02	Acceso al módulo de estudio	La vista presentó el material disponible y las opciones para abrirlo y descargarlo.	Aprobado para interfaz y consulta de materiales
CP-03	Selección de asignaturas	Se mostraron las asignaturas disponibles y sus controles de acceso	Parcial: interfaz verificada
CP-04	Visualización de unidades	Se observó la Unidad 1 disponible y la Unidad 2 bloqueada en el estado inicial.	Aprobado junto con CP-17
CP-05	Visualización de sesiones	La plataforma presentó la sesión correspondiente y	Parcial: interfaz verificada

		permitió iniciar su flujo.	
CP-06	Acceso al Pre-check	El formulario emocional previo se presentó antes de la evaluación.	Aprobado junto con CP-10
CP-07	Acceso a la evaluación	Se cargaron el contenido, la pregunta, las alternativas y el control para continuar.	Aprobado para la continuidad del flujo junto con CP-11

Fuente: elaboración propia.

En conjunto, los casos permitieron recorrer de manera continua la estructura académica de Aurora, desde el acceso a los contenidos hasta el inicio de la evaluación. Las capturas verifican principalmente la disponibilidad de las vistas y la continuidad observable del recorrido. Por esta razón, las conclusiones sobre persistencia, cálculo de puntajes y desbloqueo progresivo se sustentan en los casos específicos presentados en las secciones posteriores.

4.5.3 Gestión de materiales didácticos

La gestión de materiales didácticos permite que un usuario con rol Profesor incorpore recursos educativos y que estos queden posteriormente disponibles para los estudiantes. Desde la perspectiva del Back-End, esta función requiere procesar la solicitud de carga, registrar la información asociada con el recurso y permitir su recuperación desde el módulo de estudio.

Este proceso se evaluó mediante el caso CP-09. Para establecer el estado inicial se utilizó la evidencia presentada anteriormente en la Figura 4.4. En dicha vista, el módulo de estudio contenía el recurso denominado “Apoyo Lenguaje”, mientras que el material “Cuento Pedro y el Lobo” todavía no se encontraba disponible. Esta condición previa permitió comparar el contenido existente antes y después de ejecutar la carga.

A continuación, se accedió a la función correspondiente desde el perfil Profesor y se cargó el material “Cuento Pedro y el Lobo”. Una vez procesada la solicitud, Aurora presentó un mensaje confirmando que el recurso había sido cargado correctamente, como se observa en la Figura 4.5.

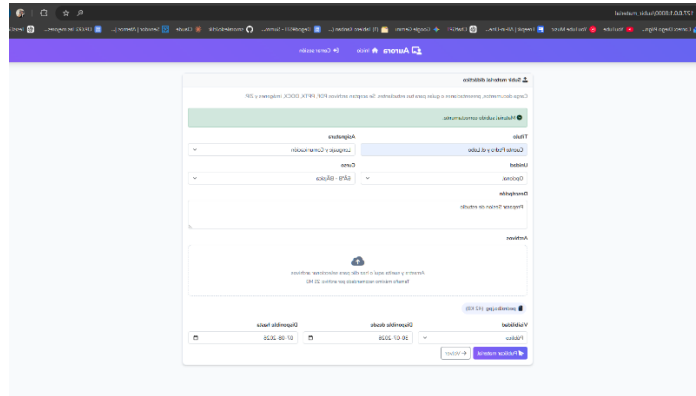


Figura 4.5. Confirmación de la carga del material “Cuento Pedro y el Lobo” desde el perfil Profesor.

Fuente: elaboración propia.

La confirmación presentada en la Figura 4.5 demuestra que el Back-End recibió y procesó la solicitud sin generar un error observable. Sin embargo, este mensaje por sí solo no resulta suficiente para comprobar que el material quedó disponible para sus destinatarios. Por esta razón, la validación se complementó accediendo posteriormente al módulo de estudio desde el perfil Estudiante.

Como se muestra en la Figura 4.6, el material “Cuento Pedro y el Lobo” apareció en la sección de estudio junto con las opciones para abrirlo y descargarlo. Este resultado representa el estado posterior del proceso y permite comprobar que el recurso cargado desde el perfil Profesor pudo ser consultado desde el perfil Estudiante.



Figura 4.6. Disponibilidad del material “Cuento Pedro y el Lobo” en el módulo de estudio del Estudiante.

Fuente: elaboración propia.

La comparación entre las Figuras 4.4 y 4.6 permite distinguir claramente la modificación producida por la operación. Antes de ejecutar el caso, el recurso evaluado no aparecía en el módulo de estudio; después de la carga, el mismo material quedó visible y asociado con las opciones de acceso correspondientes. De esta manera, la validación no se limitó al

mensaje de confirmación recibido por el Profesor, sino que incluyó la comprobación del resultado desde la perspectiva del Estudiante.

La Tabla 4.6 resume las etapas y los resultados del caso evaluado.

Tabla 4.6. Resultado de la validación de la gestión de materiales didácticos.

Etapas	Comprobación realizada	Resultado obtenido
Estado anterior	Revisión del módulo de estudio antes de ejecutar la carga.	El módulo mostraba “Apoyo Lenguaje” y todavía no contenía “Cuento Pedro y el Lobo”.
Carga del material	Incorporación del recurso desde el perfil Profesor.	Aurora confirmó que el material fue cargado correctamente.
Estado Posterior	Acceso al módulo de estudio desde el perfil Estudiante.	El nuevo material quedó disponible con opciones para abrirlo y descargarlo.
Resultado de CP-09	Validación del flujo completo de publicación.	Aprobado.

Fuente: elaboración propia.

Los resultados permiten aprobar el caso CP-09, debido a que se comprobó la continuidad del proceso desde la carga realizada por el Profesor hasta la disponibilidad posterior del recurso para el Estudiante. La evidencia valida el comportamiento funcional de publicación y consulta de materiales. No obstante, no corresponde a una evaluación de propiedades adicionales, como el análisis de seguridad del archivo cargado, la capacidad máxima de almacenamiento o el comportamiento ante múltiples cargas simultáneas.

Con esta comprobación finaliza la validación funcional relacionada con el acceso, la navegación académica y la gestión de materiales. La sección siguiente se concentra en el flujo principal de autorregulación de Aurora, incluyendo el registro emocional, la evaluación, la actualización del desempeño, la asignación de Glims y el desbloqueo progresivo de unidades.

4.6 Validación del flujo de autorregulación y reglas de negocio

El flujo de autorregulación constituye uno de los procesos centrales de Aurora, debido a que integra el registro emocional del estudiante con el desarrollo de las actividades académicas, la evaluación de sus respuestas y la actualización de su progreso. Desde la perspectiva del Back-End, este recorrido requiere coordinar diferentes operaciones y conservar la consistencia de la información generada durante cada etapa.

La validación de este flujo se efectuó mediante casos de prueba que permitieron comprobar el registro emocional previo, la finalización de la evaluación, el cálculo del puntaje, la actualización del desempeño, la asignación de Glims, la repetición de

actividades y el desbloqueo progresivo de unidades. Posteriormente, también se verificó el registro emocional realizado al finalizar la sesión.

A diferencia de las comprobaciones de navegación presentadas anteriormente, los casos de esta sección no se limitaron a observar la carga de las vistas. Las acciones ejecutadas desde la plataforma se complementaron con consultas directas sobre MySQL, con el propósito de comprobar que los resultados fueran efectivamente persistidos por el Back-End.

4.6.1 Registro emocional previo

Antes de comenzar una evaluación, Aurora solicita al estudiante seleccionar una alternativa que represente su estado emocional. Esta etapa, denominada pre-check, permite asociar la percepción emocional informada por el usuario con la actividad académica que está por iniciar.

La aparición del formulario había sido comprobada previamente mediante el caso CP-06. Sin embargo, dicha evidencia solamente demostraba que la interfaz se encontraba disponible. Para verificar que la selección fuera procesada y almacenada por el Back-End se ejecutó el caso CP-10, utilizando una consulta anterior y otra posterior sobre la tabla `estudiante_has_emocion`.

Antes de efectuar el registro se realizó una consulta de control para establecer el estado inicial de los datos. Esta comprobación permitió identificar los registros existentes antes de que el estudiante enviara una nueva selección emocional. La captura correspondiente al estado inicial se presenta como evidencia complementaria en la Figura B.9 del Anexo B.

Posteriormente, el estudiante seleccionó un estado emocional desde el formulario presentado por Aurora y confirmó su elección, como se observa en la Figura 4.7.

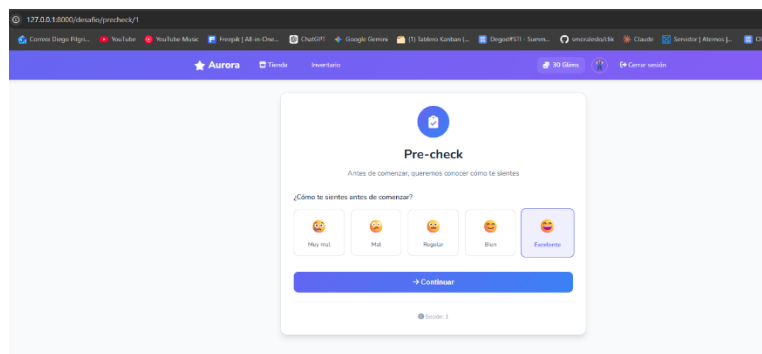


Figura 4.7. Selección y envío del estado emocional previo desde la plataforma Aurora.

Fuente: elaboración propia.

La interacción mostrada en la Figura 4.7 comprueba que el usuario pudo completar la etapa de pre-check y continuar con el flujo. No obstante, para verificar que la acción no se limitara a un cambio de pantalla, se ejecutó una segunda consulta sobre la tabla utilizada para almacenar la relación entre el estudiante y la emoción seleccionada.

Como se presenta en la Figura 4.8, la consulta posterior permitió identificar el nuevo registro generado después del envío del formulario. La comparación con el estado inicial confirma que la operación realizada desde la plataforma produjo una modificación persistente en MySQL.

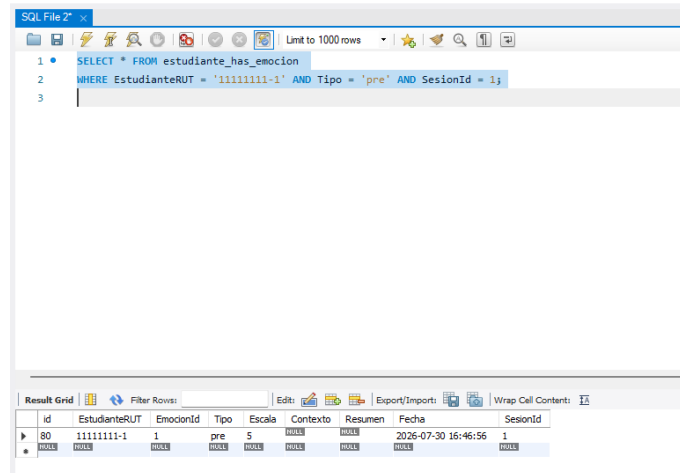


Figura 4.8. Verificación en MySQL de la persistencia del registro emocional previo.

Fuente: elaboración propia.

La modificación fue generada mediante el flujo normal de Aurora, sin insertar o editar manualmente el registro desde MySQL. La base de datos se utilizó únicamente como mecanismo de consulta y comprobación. Por consiguiente, la evidencia permite relacionar directamente la acción ejecutada por el estudiante con el resultado almacenado por el Back-End.

La Tabla 4.7 resume las etapas consideradas en esta validación.

Tabla 4.7. Resultado de la validación del registro emocional previo.

Etapas	Comprobación realizada	Resultado obtenido
Estado anterior	Consulta de control sobre estudiante_has_emocion antes de enviar el pre-check.	Se estableció el estado inicial de los registros utilizados para la comparación.
Acción del estudiante	Selección y envío de un estado emocional desde la plataforma.	Aurora procesó la selección y permitió continuar con el flujo académico.
Estado posterior	Nueva consulta sobre estudiante_has_emocion.	Se identificó el registro persistido después del envío.
Resultado de CP-10	Validación del registro emocional previo.	Aprobado.

Fuente: elaboración propia.

Los resultados permiten aprobar el caso CP-10. Se comprobó que el estudiante pudo enviar su estado emocional y que el Back-End persistió la selección correspondiente en MySQL. Esta validación acredita el registro técnico de la respuesta informada por el usuario, pero no implica una interpretación psicológica ni permite establecer conclusiones sobre su condición emocional.

Una vez almacenado el pre-check, Aurora habilita la continuidad de la sesión académica. La siguiente subsección examina la finalización de la evaluación y el cálculo del puntaje obtenido por el estudiante.

4.6.2 Finalización de la evaluación y cálculo del puntaje

Después de completar el pre-check, el estudiante desarrolla la evaluación asociada con la sesión académica. Durante este proceso, Aurora presenta las preguntas de manera secuencial y conserva temporalmente las respuestas en la sesión HTTP correspondiente al usuario. Al alcanzar el término de la evaluación, el Back-End recupera estas respuestas, calcula el puntaje obtenido y dirige al estudiante hacia la etapa de post-check.

Este comportamiento se evaluó mediante los casos CP-11 y CP-12. El primero permitió comprobar la continuidad y finalización del test, mientras que el segundo se concentró en el cálculo del puntaje de acuerdo con las respuestas proporcionadas.

El cálculo es realizado mediante la función `compute_score()`, que examina cada pregunta según su tipo y suma los puntajes parciales obtenidos. Las reglas aplicadas por el Back-End son las siguientes:

- En las preguntas de alternativa, se compara la respuesta seleccionada con la alternativa correcta. La coincidencia otorga el puntaje asignado a la pregunta; una respuesta incorrecta obtiene cero puntos.
- En las preguntas de desarrollo, se comprueba la existencia de una respuesta no vacía. Este tipo de pregunta no realiza una evaluación semántica de su contenido.
- En las preguntas de clasificación, el puntaje se determina proporcionalmente según la cantidad de elementos ubicados en el orden correcto.

Por consiguiente, el resultado total corresponde a la suma de los puntajes parciales calculados para las preguntas que forman parte del test:

Puntaje total = suma de los puntajes parciales obtenidos.

En el escenario ejecutado, correspondiente al test con identificador 1, el Back-End calculó un resultado de 25 puntos. Una vez finalizado el procesamiento, Aurora presentó la vista de post-check e informó al estudiante el puntaje alcanzado, como se observa en la Figura 4.9.

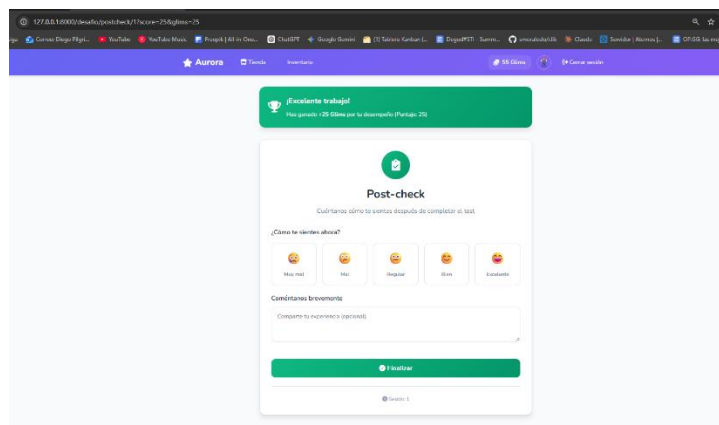


Figura 4.9. Finalización de la evaluación y presentación del puntaje calculado por el Back-End.

Fuente: elaboración propia.

La aparición de esta vista permite comprobar que el flujo no se interrumpió durante la transición entre las preguntas, el procesamiento de las respuestas y la finalización de la evaluación. Asimismo, el valor mostrado confirma que el resultado calculado por el servidor fue incorporado correctamente en la respuesta presentada al estudiante.

La Figura 4.9 también informa la asignación de 25 Glims. Sin embargo, este valor no se analiza todavía como evidencia de persistencia, debido a que la interfaz solamente demuestra el resultado comunicado al usuario. La actualización efectiva del desempeño y del saldo del estudiante se comprueba en la subsección siguiente mediante las consultas realizadas sobre MySQL.

La consulta previa asociada con el caso CP-11 estableció que el estudiante poseía inicialmente 30 Glims y que todavía no existía una calificación registrada para el test evaluado. Las consultas previas se incorporan en las Figuras B.10 y B.11 del Anexo B, donde funciona como estado de control para la comparación posterior. De esta manera, el cuerpo del capítulo conserva la evidencia principal de la finalización, mientras que la comprobación de las modificaciones persistentes se presenta junto con la regla de negocio correspondiente.

La Tabla 4.8 resume los resultados obtenidos en esta etapa.

Tabla 4.8. Resultados de la validación de la finalización y el cálculo del puntaje.

Caso	Función evaluada	Resultado obtenido	Estado
CP-11	Continuidad y finalización de la evaluación	El estudiante completó el recorrido del test y fue dirigido al post-check, donde se presentó un resultado de 25 puntos.	Aprobado

CP-12	Cálculo del puntaje	El Back-End procesó las respuestas del test con identificador 1 y obtuvo un puntaje total de 25, aplicando las reglas definidas según el tipo de pregunta.	Aprobado para el escenario ejecutado
-------	---------------------	--	--------------------------------------

Fuente: elaboración propia.

Los resultados permiten aprobar los casos CP-11 y CP-12. Se comprobó la continuidad del recorrido hasta la finalización de la evaluación y la generación del puntaje utilizado por las etapas posteriores del sistema. La prueba acredita el comportamiento del escenario ejecutado, pero no constituye una evaluación exhaustiva de todas las combinaciones posibles de preguntas y respuestas.

Una vez calculado el resultado, el Back-End debe registrar el desempeño obtenido y determinar la recompensa correspondiente. La siguiente subsección examina estas operaciones mediante la comparación entre el estado inicial y los valores persistidos en MySQL.

4.6.3 Actualización del desempeño y asignación de Glims

Una vez calculado el puntaje de la evaluación, el Back-End debe registrar el resultado académico del estudiante y determinar la cantidad de Glims que corresponde asignar. Ambas operaciones forman parte del mismo flujo de negocio, debido a que la recompensa depende de la comparación entre el puntaje obtenido y el mejor resultado almacenado previamente.

Este comportamiento se evaluó mediante los casos CP-13 y CP-14. El primero verificó la persistencia del desempeño, mientras que el segundo comprobó la actualización del saldo de Glims. Ambos casos fueron ejecutados como parte del flujo completo iniciado en CP-11.

Antes de finalizar la evaluación se realizaron dos consultas de control. La primera confirmó que todavía no existía un desempeño asociado con el test evaluado, mientras que la segunda permitió establecer que el estudiante poseía un saldo inicial de 30 Glims. Estas consultas se presentan en las Figuras B.10 y B.11 del Anexo B como evidencia del estado anterior.

Después de finalizar la evaluación, se consultó nuevamente la tabla desempeño. Como se observa en la Figura 4.10, MySQL contenía un registro con NotaTest = 25, correspondiente al puntaje presentado previamente por Aurora.

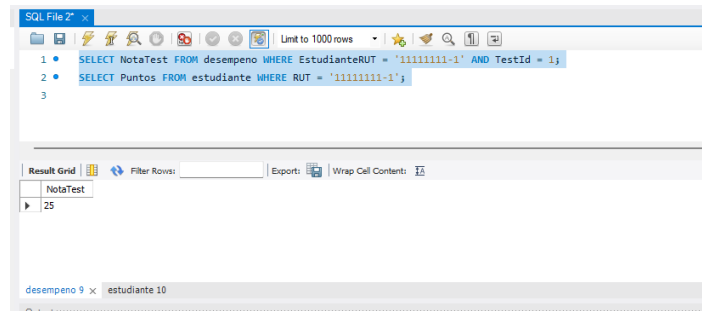


Figura 4.10. Persistencia del desempeño obtenido después de finalizar la evaluación.

Fuente: elaboración propia.

La coincidencia entre el valor mostrado por la plataforma y el almacenado en MySQL permite comprobar que el resultado calculado por el Back-End no se mantuvo únicamente en la sesión HTTP ni en la interfaz. El puntaje fue persistido como parte del desempeño académico del estudiante, por lo que puede ser utilizado posteriormente para consultar su progreso y aplicar otras reglas de negocio.

La asignación de Glims se determina mediante una regla incremental. El Back-End compara el resultado recién obtenido con el mejor puntaje previo y solamente recompensa la diferencia positiva:

$$\Delta \text{Glims} = \max(0, \text{puntaje nuevo} - \text{mejor puntaje previo})$$

En el escenario ejecutado no existía un desempeño anterior para el test, por lo que el resultado de 25 puntos generó una recompensa de 25 Glims. Al sumar esta cantidad al saldo inicial de 30, el estudiante alcanzó un total de 55 Glims:

$$\text{Saldo posterior} = 30 + 25 = 55 \text{ Glims}$$

La consulta posterior sobre el registro del estudiante confirmó esta actualización, como se presenta en la Figura 4.11.

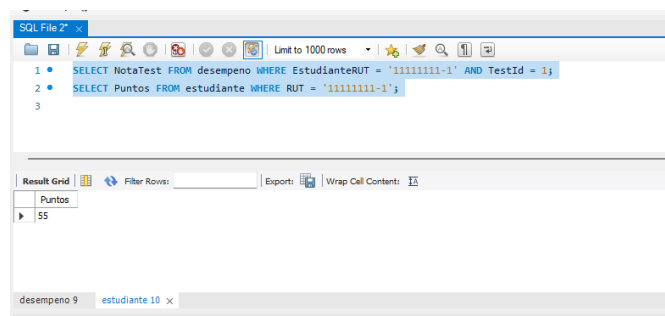


Figura 4.11. Actualización del saldo de Glims después de finalizar la evaluación.

Fuente: elaboración propia.

La Figura 4.11 muestra que el valor almacenado en el campo Puntos, utilizado por Aurora para representar el saldo de Glims, aumentó desde 30 hasta 55. Este resultado coincide con la recompensa de 25 Glims informada al estudiante en la Figura 4.9.

La comprobación conjunta de las Figuras 4.9, 4.10 y 4.11 permite relacionar las diferentes etapas del procesamiento. Primero, Aurora comunicó un puntaje de 25 y una recompensa de 25 Glims; posteriormente, MySQL registró el desempeño obtenido y reflejó el incremento correspondiente en el saldo del estudiante.

La Tabla 4.9 resume los resultados de los casos evaluados.

Tabla 4.9. Resultados de la actualización del desempeño y la asignación de Glims.

Caso	Función evaluada	Resultado obtenido	Estado
CP-13	Persistencia del desempeño	MySQL registró NotaTest = 25 para la evaluación ejecutada.	Aprobado
CP-14	Asignación de Glims por mejora	La interfaz informó una recompensa de 25 Glims y el saldo almacenado aumentó desde 30 hasta 55.	Aprobado

Fuente: elaboración propia.

Los resultados permiten aprobar los casos CP-13 y CP-14. Se comprobó que el Back-End persistió el resultado de la evaluación y actualizó el saldo de acuerdo con la mejora obtenida. La utilización de la diferencia positiva evita que la recompensa dependa únicamente de completar repetidamente una actividad y establece como condición la superación del mejor desempeño anterior.

Sin embargo, este primer escenario solamente demuestra la asignación correspondiente a una evaluación sin puntaje previo. Para comprobar que la regla evita recompensas duplicadas, la siguiente subsección analiza la repetición de la actividad sin superar la mejor marca registrada.

4.6.4 Prevención de recompensas duplicadas

La asignación incremental de Glims busca recompensar la mejora del estudiante y evitar la acumulación de moneda virtual mediante la repetición de una actividad que ya fue completada con el mismo o un mejor resultado. Para comprobar este comportamiento se ejecutó el caso CP-15, repitiendo la evaluación utilizada en las validaciones anteriores.

Antes de esta repetición, el estudiante poseía un mejor puntaje histórico de 25 puntos y un saldo de 55 Glims. Estos valores habían sido comprobados previamente mediante las consultas presentadas en las Figuras 4.10 y 4.11. En la nueva ejecución, el estudiante volvió a obtener 25 puntos, por lo que el resultado no superó la marca almacenada.

De acuerdo con la regla implementada, la recompensa corresponde únicamente a la diferencia positiva entre el nuevo puntaje y el mejor resultado anterior:

$$\Delta\text{Glims} = \max(0, \text{puntaje nuevo} - \text{mejor puntaje previo})$$

Para el escenario evaluado, el cálculo fue el siguiente:

$$\Delta\text{Glims} = \max(0, 25 - 25) = 0$$

Por consiguiente, la repetición no debía incrementar el saldo. Como se observa en la Figura 4.12, Aurora informó un puntaje de 25 y una asignación de 0 Glims. La misma vista mostró que el estudiante continuaba disponiendo de un saldo total de 55 Glims.

Figura 4.12. Resultado de la repetición de una evaluación sin superar el mejor puntaje registrado.

Fuente: elaboración propia.

La respuesta presentada por la plataforma es coherente con la regla de negocio, debido a que diferencia el resultado académico obtenido de la recompensa asignada. Aunque el estudiante completó nuevamente la actividad y alcanzó 25 puntos, el Back-End no generó una recompensa adicional porque ese puntaje ya correspondía a su mejor marca.

Para comprobar que el valor mostrado no se limitara a la interfaz, se consultó posteriormente el saldo almacenado en MySQL. Como se presenta en la Figura 4.13, el campo Puntos permaneció con un valor de 55 después de repetir la evaluación.

```

1 • SELECT NotaTest FROM desempeno WHERE EstudianteRUT = '11111111-1' AND TestId = 1;
2 • SELECT Puntos FROM estudiante WHERE RUT = '11111111-1';
3

```

Puntos
55

Figura 4.13. Verificación en MySQL de la conservación del saldo de Glims después de repetir una evaluación sin mejorar el puntaje histórico.

Fuente: elaboración propia.

La comparación con el estado anterior confirma que no se produjo un nuevo incremento. El estudiante poseía 55 Glims antes de la repetición y conservó el mismo saldo después de obtener nuevamente 25 puntos. En consecuencia, la finalización repetida de la actividad no generó una duplicación de la recompensa previamente otorgada.

La Tabla 4.10 resume las comprobaciones efectuadas mediante el caso CP-15.

Tabla 4.10. Resultado de la validación de la prevención de recompensas duplicadas.

Etapas	Comprobación realizada	Resultado obtenido
Estado anterior	Revisión del mejor puntaje y del saldo registrados después de la primera ejecución.	El estudiante poseía un mejor puntaje de 25 y un saldo de 55 Glims.
Repetición de la evaluación	Nueva finalización del test sin superar el resultado histórico.	Aurora presentó nuevamente 25 puntos e informó una recompensa de 0 Glims.
Estado posterior	Consulta del saldo almacenado en MySQL.	El saldo permaneció en 55 Glims.
Resultado de CP-15	Validación de la prevención de Glims duplicados.	Aprobado.

Fuente: elaboración propia.

Los resultados permiten aprobar el caso CP-15 para la prevención de recompensas duplicadas. Se comprobó que la repetición de una evaluación con un puntaje igual a la mejor marca anterior no incrementó el saldo del estudiante. Esta conducta respalda el propósito de los Glims como incentivo asociado con la mejora y no solamente con la repetición de actividades.

El alcance de la evidencia se limita a la conservación del saldo en el escenario ejecutado. La consulta presentada permite confirmar directamente que los Glims no aumentaron, pero no demuestra por sí sola el comportamiento frente a solicitudes concurrentes ni permite afirmar que el registro de desempeño permaneciera inmutable. Estas propiedades requerirían pruebas específicas adicionales.

Después de validar la evaluación y sus recompensas, corresponde comprobar la regla que controla el avance dentro de la estructura académica. La subsección siguiente examina el desbloqueo progresivo de unidades cuando se cumplen las etapas requeridas de la sesión anterior.

4.6.5 Desbloqueo progresivo de unidades

El avance académico en Aurora se encuentra controlado mediante una regla de desbloqueo progresivo. Su propósito es impedir que el estudiante acceda directamente a unidades posteriores sin haber completado previamente las actividades requeridas. Desde la perspectiva del Back-End, esta comprobación relaciona el estado de las sesiones con los registros emocionales y el desempeño académico almacenados.

La primera unidad de cada asignatura se encuentra disponible inicialmente. Para habilitar una unidad posterior, el Back-End verifica que todas las sesiones pertenecientes a la unidad anterior cuenten con los siguientes registros:

- Pre-check emocional completado.
- Desempeño correspondiente a la evaluación.
- Post-check emocional completado.

Este comportamiento se evaluó mediante el caso CP-17. El estado inicial había sido observado anteriormente en CP-04, cuando la Unidad 1 se encontraba disponible y la Unidad 2 aparecía bloqueada. Posteriormente, el estudiante completó las etapas requeridas para la sesión de la unidad anterior.

Para comprobar las condiciones previas se consultaron los registros almacenados en MySQL. La evidencia CAP4_CP17_01_db.png permitió verificar la existencia del pre-check, el desempeño y el post-check requeridos para considerar completa la sesión evaluada. Esta captura se incorpora en la Figura B.12 del Anexo B, debido a que funciona como respaldo técnico de la condición de desbloqueo.

Después de verificar estos registros, se accedió nuevamente a la vista de unidades desde la plataforma. Como se observa en la Figura 4.14, la Unidad 2 apareció habilitada y permitió continuar con el avance académico.

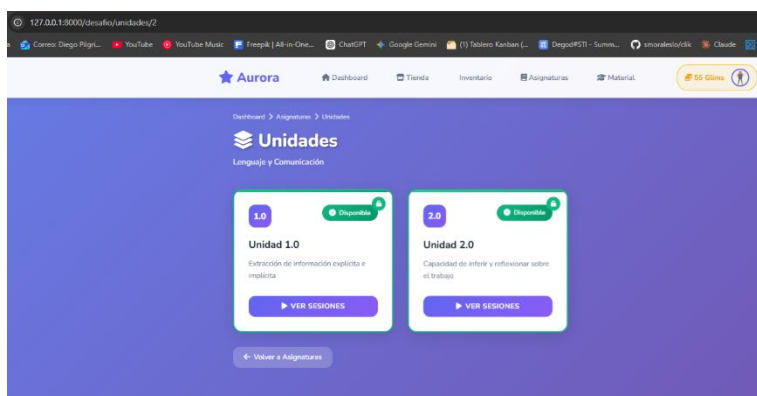


Figura 4.14. Habilidad de la Unidad 2 después de completar las condiciones requeridas en la unidad anterior.

Fuente: elaboración propia.

La comparación con el estado inicial observado en CP-04 permite identificar el cambio producido durante el flujo. Antes de completar las condiciones, la Unidad 2 se mostraba bloqueada; después de registrarse las etapas requeridas, la misma unidad quedó disponible. De esta manera, la habilitación no dependió únicamente de haber ingresado a la Unidad 1, sino del cumplimiento de los registros utilizados por el Back-End para determinar la finalización de sus actividades.

La comprobación conjunta de la consulta en MySQL y la Figura 4.14 permite relacionar la información persistida con el resultado presentado por la plataforma. Los registros almacenados acreditaron el cumplimiento de las condiciones previas, mientras que la vista de unidades reflejó la aplicación de la regla de desbloqueo.

La Tabla 4.11 resume las etapas evaluadas mediante CP-17.

Tabla 4.11. Resultado de la validación del desbloqueo progresivo de unidades.

Etapas	Comprobación realizada	Resultado obtenido
Estado inicial	Revisión de las unidades antes de completar las actividades requeridas.	La Unidad 1 se encontraba disponible y la Unidad 2 aparecía bloqueada.
Condiciones previas	Consulta de los registros asociados con la sesión de la unidad anterior.	Se comprobaron los registros de pre-check, desempeño y post-check requeridos.
Estado posterior	Acceso nuevamente a la vista de unidades.	La Unidad 2 apareció habilitada en la plataforma.
Resultado de CP-17	Validación de la regla de desbloqueo progresivo.	Aprobado.

Fuente: elaboración propia.

Los resultados permiten aprobar el caso CP-17. Se comprobó que el Back-End evaluó las condiciones asociadas con la finalización de la unidad anterior y habilitó la unidad siguiente una vez satisfechos sus requisitos. Esta regla contribuye a mantener un orden progresivo en el recorrido académico y vincula el avance del estudiante con la finalización efectiva de las actividades.

El alcance de la prueba se limita a la transición observada desde la Unidad 1 hacia la Unidad 2. Por consiguiente, la evidencia no permite afirmar que se hayan evaluado todas las combinaciones posibles ni una cadena de desbloqueo compuesta por múltiples unidades. Tampoco corresponde a una prueba de concurrencia frente a varias solicitudes simultáneas.

Entre las condiciones utilizadas para habilitar la siguiente unidad se encuentra la realización del registro emocional posterior. La subsección siguiente examina específicamente la persistencia de este post-check y su incorporación al cierre de la sesión académica.

4.6.6 Registro emocional posterior

Al finalizar la evaluación, Aurora solicita al estudiante informar nuevamente su estado emocional mediante un post-check. Esta etapa completa el flujo iniciado con el registro emocional previo y permite almacenar la percepción comunicada por el estudiante después de desarrollar la actividad académica.

La persistencia de este registro se evaluó mediante el caso CP-16. El formulario utilizado durante la ejecución puede observarse previamente en la Figura 4.12, donde el estudiante seleccionó la alternativa “Excelente” y dispuso del control para finalizar la sesión. Aunque dicha figura fue presentada anteriormente para analizar la prevención de recompensas duplicadas, la misma pantalla representa el punto de entrada del registro emocional posterior.

Después de confirmar la selección, el Back-End procesó la solicitud y almacenó la información asociada con el estudiante y la sesión académica. Para comprobar esta operación se ejecutó una consulta sobre la tabla `estudiante_has_emocion`, filtrando los registros correspondientes a la sesión evaluada y al tipo emocional posterior.

Como se observa en la Figura 4.15, MySQL presentó un registro con Tipo = post, Escala = 5, Contexto = Excelente y SesionId = 1. Estos valores coinciden con la alternativa seleccionada desde la plataforma.

```

1 SELECT * FROM estudiante_has_emocion
2 WHERE EstudianteRUT = '11111111-1' AND Tipo = 'post' AND SesionId = 1;
3

```

id	EstudianteRUT	EmocionId	Tipo	Escala	Contexto	Resumen	Fecha	SesionId
82	11111111-1	2	post	5	Excelente		2026-07-30 17:32:08	1

Figura 4.15. Verificación en MySQL de la persistencia del registro emocional posterior a la evaluación.

Fuente: elaboración propia.

La presencia del valor post permite distinguir este registro del pre-check almacenado al comenzar la actividad. A su vez, el identificador de sesión permite asociar la emoción posterior con la evaluación desarrollada por el estudiante. De este modo, el Back-End conserva ambos momentos del flujo emocional como registros diferenciados dentro de la misma estructura de datos.

La comprobación no se limitó a la aparición del formulario. La consulta presentada en la Figura 4.15 confirma que la selección realizada desde Aurora fue transformada en un registro persistente y vinculada con la sesión correspondiente. Este post-check también

forma parte de las condiciones consideradas por la regla de desbloqueo progresivo examinada en la subsección anterior.

La Tabla 4.12 resume las comprobaciones efectuadas mediante el caso CP-16.

Tabla 4.12. Resultado de la validación del registro emocional posterior.

Etapas	Comprobación realizada	Resultado obtenido
Acción del estudiante	Selección de la alternativa “Excelente” en el formulario de post-check	La plataforma permitió completar el registro emocional posterior y finalizar la sesión.
Persistencia	Consulta de estudiante_has_emocion filtrada por la sesión y el tipo de registro.	MySQL presentó un registro con Tipo = post, Escala = 5, Contexto = Excelente y SesionId = 1.
Diferenciación del registro	Revisión del campo utilizado para identificar el momento del chequeo emocional.	El valor post permitió distinguirlo del registro previo almacenado al comenzar la actividad.
Resultado de CP-16	Validación de la persistencia del post-check.	Aprobado.

Fuente: elaboración propia.

Los resultados permiten aprobar el caso CP-16. Se comprobó que el estudiante pudo completar el post-check y que el Back-End persistió la información emocional posterior, diferenciándola del pre-check y vinculándola con la sesión académica correspondiente.

Al igual que en la validación del registro previo, esta evidencia acredita únicamente el almacenamiento técnico de la alternativa informada por el usuario. Los datos obtenidos no constituyen una evaluación psicológica ni permiten establecer conclusiones clínicas sobre el estado emocional del estudiante.

Con esta comprobación se completa la validación del flujo principal de autorregulación de Aurora. El recorrido evaluado abarcó el pre-check, la evaluación, el cálculo del puntaje, la actualización del desempeño, la asignación de Glims, la prevención de recompensas duplicadas, el post-check y el desbloqueo progresivo de unidades. La sección siguiente examina la integración con el servicio externo de inteligencia artificial y el manejo controlado de respuestas no satisfactorias.

4.7 Validación de integraciones externas y manejo de errores

Además de las reglas asociadas con el flujo académico, el Back-End de Aurora debe interactuar con servicios externos y responder de manera controlada cuando una operación no puede completarse satisfactoriamente. Estas situaciones resultan relevantes

porque una dependencia no disponible, una ruta inválida o la solicitud de una entidad inexistente no deberían provocar una interrupción general del servidor.

La validación de esta sección considera la interacción con el servicio externo de inteligencia artificial, el control aplicado sobre las rutas utilizadas para descargar materiales y el tratamiento de solicitudes asociadas con sesiones inexistentes. Para estas comprobaciones se utilizaron los casos CP-18, CP-19 y CP-20.

El alcance de las pruebas se concentra en el comportamiento observable del Back-End frente a los escenarios ejecutados. Por consiguiente, los resultados no representan una evaluación exhaustiva de seguridad, tolerancia a fallos o disponibilidad, sino una comprobación funcional del manejo implementado para cada situación.

4.7.1 Manejo de indisponibilidad del servicio de inteligencia artificial

Además de las reglas asociadas con el flujo académico, el Back-End de Aurora debe interactuar con servicios externos y responder de manera controlada cuando una operación no puede completarse satisfactoriamente. Estas situaciones resultan relevantes porque una dependencia no disponible, una ruta inválida o la solicitud de una entidad inexistente no deberían provocar una interrupción general del servidor.

Aurora incorpora un chatbot que permite al estudiante enviar consultas y recibir respuestas generadas mediante un servicio externo de inteligencia artificial. Desde la perspectiva del Back-End, este proceso requiere recibir el mensaje del usuario, solicitar una respuesta al proveedor externo y conservar el historial de la interacción en la tabla registro_ia.

Este comportamiento se examinó mediante el caso CP-18. Antes de iniciar la interacción se ejecutó una consulta de control sobre registro_ia, con el propósito de establecer el estado anterior de los datos. La captura correspondiente al estado anterior se presenta en la Figura B.13 del Anexo B como evidencia complementaria.

Durante la ejecución del caso, el servicio externo no pudo generar una respuesta satisfactoria debido a la indisponibilidad de la clave utilizada para acceder a la API. Como se observa en la Figura 4.16, la interfaz presentó el mensaje controlado “Error en la respuesta”.

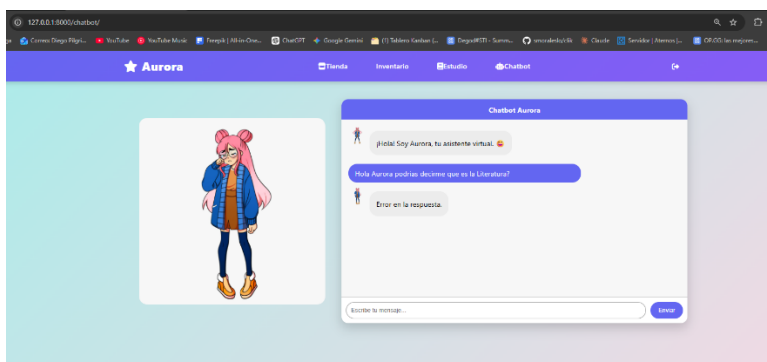
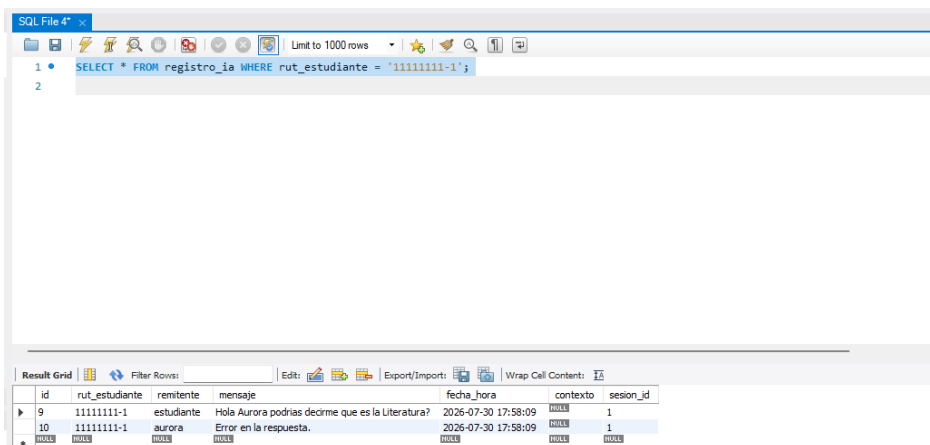


Figura 4.16. Mensaje controlado presentado por Aurora ante la indisponibilidad del servicio externo de inteligencia artificial.

Fuente: elaboración propia.

La Figura 4.16 no demuestra una respuesta exitosa generada por el modelo de inteligencia artificial. Sin embargo, permite comprobar que Aurora recibió la consulta y presentó un mensaje identificable ante el fallo de la dependencia externa, en lugar de dejar al usuario sin información sobre el resultado de la solicitud.

Después de la interacción se consultó nuevamente la tabla registro_ia. Como se presenta en la Figura 4.17, el Back-End almacenó tanto el mensaje enviado por el estudiante como el mensaje de error registrado por Aurora.



The screenshot shows a SQL client window titled 'SQL File 4*'. The query editor contains the following SQL statement:

```
SELECT * FROM registro_ia WHERE rut_estudiante = '11111111-1';
```

The results are displayed in a 'Result Grid' table with the following data:

id	rut_estudiante	remite	mensaje	fecha_hora	contexto	sesion_id
9	11111111-1	estudiante	Hola Aurora podrias decirme que es la Literatura?	2026-07-30 17:58:09	1	1
10	11111111-1	aurora	Error en la respuesta.	2026-07-30 17:58:09	1	1

Figura 4.17. Persistencia de la interacción y del mensaje de error del chatbot en la tabla registro_ia.

Fuente: elaboración propia.

La consulta permite comprobar que el fallo del servicio externo no impidió conservar el evento ocurrido durante la interacción. El mensaje del estudiante y el resultado informado por Aurora quedaron almacenados como registros diferenciados, manteniendo el historial correspondiente al escenario ejecutado.

Esta persistencia resulta relevante porque permite conservar trazabilidad sobre las solicitudes procesadas por el chatbot, incluso cuando el proveedor externo no entrega una respuesta satisfactoria. No obstante, la evidencia disponible no permite validar la calidad de las respuestas generadas, la exactitud de su contenido ni el comportamiento del modelo ante diferentes tipos de consulta.

La Tabla 4.13 resume las comprobaciones efectuadas mediante CP-18.

Tabla 4.13. Resultado de la validación del chatbot ante la indisponibilidad del servicio externo.

Etapa	Comprobación realizada	Resultado obtenido
Estado anterior	Consulta de control sobre la tabla registro_ia antes de utilizar el chatbot.	Se estableció el estado inicial utilizado para la comparación.

Interacción	Envío de un mensaje desde la interfaz del estudiante.	Aurora recibió la consulta y presentó el mensaje “Error en la respuesta”.
Persistencia	Nueva consulta sobre registro_ia después de la interacción.	Se almacenaron el mensaje del estudiante y el mensaje de error informado por Aurora.
Resultado de CP-18	Validación del manejo de indisponibilidad externa y conservación del evento.	Aprobado con alcance limitado.

Fuente: elaboración propia.

El caso CP-18 se considera aprobado con alcance limitado. La prueba comprobó el manejo controlado de la indisponibilidad externa y la persistencia de la interacción, pero no validó una respuesta exitosa generada por OpenAI debido a la condición presentada por la clave de acceso.

Esta limitación debe diferenciarse de un fallo general del Back-End. La funcionalidad continuó recibiendo la consulta y almacenando el resultado ocurrido, pero quedó pendiente la comprobación de una interacción exitosa con el proveedor externo mediante una credencial operativa.

La siguiente subsección examina una operación interna del Back-End: la validación de las rutas utilizadas para solicitar la descarga de materiales didácticos.

4.7.2 Validación de rutas de descarga

Los materiales didácticos cargados por el Profesor pueden ser consultados posteriormente desde el módulo de estudio del Estudiante. Para permitir su descarga, el Back-End debe resolver la ubicación del archivo solicitado sin admitir rutas que conduzcan fuera del directorio destinado al almacenamiento de los recursos.

Aurora almacena estos archivos dentro del directorio uploads/. Antes de entregar un recurso, el Back-End valida la ruta solicitada mediante una función auxiliar que comprueba que el archivo resultante permanezca dentro de este directorio. Esta medida busca impedir que una solicitud construida de manera incorrecta acceda a ubicaciones ajenas al almacenamiento autorizado.

El funcionamiento de esta validación se examinó mediante el caso CP-19. Para ello, se efectuó una solicitud de descarga utilizando deliberadamente una ruta inválida que intentaba referenciar una ubicación fuera del directorio permitido.

Como se observa en la Figura 4.18, el Back-End rechazó la solicitud y devolvió una respuesta JSON con el mensaje {"detail":"Ruta inválida"}.

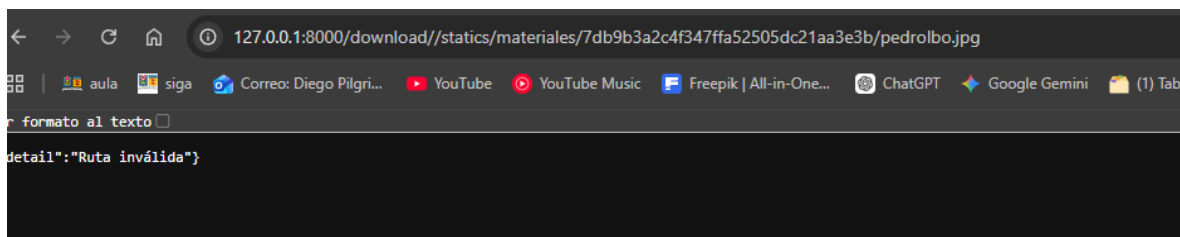


Figura 4.18. Respuesta controlada del Back-End ante una ruta de descarga inválida.

Fuente: elaboración propia.

El resultado presentado demuestra que la ruta proporcionada no fue procesada como una descarga válida. En lugar de entregar el archivo solicitado o exponer su contenido, Aurora interrumpió la operación y comunicó de manera explícita que la ruta no cumplía las condiciones establecidas.

Esta comprobación permite acreditar el funcionamiento del control para el escenario ejecutado. Sin embargo, la captura fue obtenida desde el navegador y solamente muestra el contenido de la respuesta JSON. Debido a que no presenta las herramientas de desarrollo ni los registros del servidor, no permite verificar directamente el código de estado HTTP asociado con la respuesta. Por esta razón, el caso se considera parcialmente validado y no se afirma que la evidencia visual demuestre específicamente una respuesta HTTP 400.

La Tabla 4.14 resume las comprobaciones realizadas.

Tabla 4.14. Resultado de la validación de rutas de descarga.

Etapas	Comprobación realizada	Resultado obtenido
Solicitud inválida	Envío de una ruta que no correspondía con una ubicación permitida dentro del directorio de archivos.	El Back-End rechazó la solicitud de descarga.
Respuesta del servidor	Revisión del contenido presentado en el navegador.	Aurora devolvió el mensaje JSON {"detail":"Ruta inválida"}.
Código HTTP	Revisión de la información visible en la evidencia.	La captura no permite identificar directamente el código de estado devuelto.
Resultado de CP-19	Validación del rechazo controlado de la ruta inválida.	Parcial: respuesta controlada verificada.

Fuente: elaboración propia.

El caso CP-19 permite comprobar que el Back-End dispone de un control para rechazar la ruta inválida utilizada durante la prueba. La solicitud no produjo la descarga del recurso indicado y fue respondida mediante un mensaje JSON identificable.

El alcance de esta evidencia se limita al escenario ejecutado. Por consiguiente, no constituye una auditoría exhaustiva de seguridad ni permite afirmar que se hayan evaluado todas las variantes posibles de manipulación de rutas. Una evaluación más amplia requeriría probar diferentes codificaciones, separadores, rutas relativas y combinaciones diseñadas específicamente para intentar evadir la validación.

Además de controlar las rutas utilizadas para acceder a los archivos, el Back-End debe responder adecuadamente cuando una solicitud hace referencia a información que no existe. La siguiente subsección examina el tratamiento aplicado al intentar acceder a una sesión académica inexistente.

4.7.3 Manejo de solicitudes a entidades inexistentes

El Back-End de Aurora debe responder de manera controlada cuando una solicitud hace referencia a una entidad que no se encuentra registrada. Esta situación puede producirse, por ejemplo, cuando se utiliza un identificador inexistente al intentar acceder a una sesión académica.

Este comportamiento se evaluó mediante el caso CP-20. Para ejecutar la prueba, se realizó una solicitud utilizando el identificador de sesión 99999, el cual no correspondía con una sesión almacenada en la base de datos.

Como se observa en la Figura 4.19, el Back-End rechazó la solicitud y devolvió una respuesta JSON con el mensaje `{"detail":"Sesión no encontrada"}`.

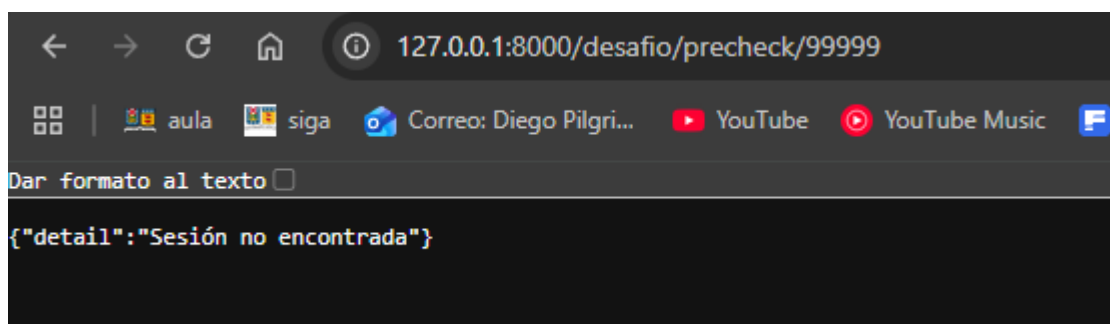


Figura 4.19. Respuesta controlada del Back-End ante la solicitud de una sesión académica inexistente.

Fuente: elaboración propia.

El resultado permite comprobar que Aurora identificó la inexistencia del recurso solicitado y comunicó la situación mediante un mensaje específico. En lugar de continuar el procesamiento con información inválida o presentar una respuesta sin explicación, el Back-End interrumpió la operación de manera controlada.

La captura demuestra el contenido de la respuesta presentada en el navegador. No obstante, debido a que la evidencia no incluye las herramientas de desarrollo ni los registros del servidor, el código de estado HTTP no puede observarse directamente. Por esta razón, la comprobación se concentra en el reconocimiento de la sesión inexistente y en la respuesta JSON generada.

La Tabla 4.15 resume las comprobaciones efectuadas mediante el caso CP-20.

Tabla 4.15. Resultado de la validación de solicitudes asociadas con entidades inexistentes.

Etapas	Comprobación realizada	Resultado obtenido
Solicitud	Acceso a una sesión académica utilizando el identificador inexistente 99999.	El Back-End determinó que la sesión solicitada no se encontraba registrada.
Respuesta controlada	Revisión del contenido presentado por el servidor.	Aurora devolvió el mensaje JSON {"detail": "Sesión no encontrada"}.
Código HTTP	Revisión de la información visible en la captura.	La evidencia no permite observar directamente el código de estado devuelto.
Resultado de CP-20	Validación del manejo de una entidad inexistente.	Aprobado para el comportamiento observable.

Fuente: elaboración propia.

Los resultados permiten aprobar el caso CP-20 para el escenario ejecutado. Se comprobó que el Back-End reconoció la inexistencia de la sesión solicitada y devolvió un mensaje controlado, evitando continuar el flujo con una entidad inválida.

El alcance de esta prueba se limita a la solicitud de una sesión académica inexistente. Por consiguiente, la evidencia no permite generalizar el mismo comportamiento a todas las entidades administradas por Aurora ni demuestra directamente el código HTTP asociado con la respuesta. La validación exhaustiva requeriría repetir el procedimiento sobre otros recursos y registrar tanto el contenido como el estado de cada respuesta.

Con esta comprobación finaliza la validación de integraciones externas y manejo de errores. Los casos ejecutados permitieron examinar la indisponibilidad del servicio de inteligencia artificial, el rechazo de una ruta de descarga inválida y el tratamiento de una sesión inexistente. En conjunto, estas pruebas demuestran que el Back-End dispone de respuestas controladas para los escenarios evaluados, aunque su alcance no equivale a una auditoría integral de seguridad o tolerancia a fallos.

CAPÍTULO 5: CONCLUSIONES Y RECOMENDACIONES

El presente capítulo expone las conclusiones obtenidas a partir del diseño, implementación y validación del Back-End de Aurora. Para ello, se relacionan los resultados presentados en el capítulo anterior con el objetivo general y los objetivos específicos definidos para el subproyecto, identificando el grado en que fueron alcanzados.

Posteriormente, se analizan las principales contribuciones técnicas y funcionales del trabajo realizado, junto con su aporte dentro de la plataforma educativa. También se presentan los alcances, limitaciones y aprendizajes identificados durante el desarrollo, diferenciando las funcionalidades efectivamente implementadas de aquellas propiedades que no fueron evaluadas de manera exhaustiva.

Finalmente, se proponen recomendaciones y líneas de trabajo futuro orientadas a fortalecer la seguridad, mantenibilidad, calidad, escalabilidad y validación del Back-End. Estas propuestas consideran que Aurora corresponde a un proyecto grupal y que el alcance de esta memoria se limita específicamente a la infraestructura Back-End que permite articular la plataforma web, la aplicación móvil, la base de datos y los servicios externos utilizados por el sistema.

5.1 Discusión de resultados y cumplimiento de objetivos

La validación desarrollada en el Capítulo 4 permitió examinar los principales flujos funcionales soportados por el Back-End de Aurora. Los casos ejecutados abarcaron la autenticación y redirección según el rol, la navegación por la estructura académica, la gestión de materiales, los registros emocionales, el desarrollo de evaluaciones, el cálculo y persistencia de puntajes, la asignación de Glims, el desbloqueo progresivo de unidades y el manejo de situaciones no satisfactorias.

Los resultados no poseen el mismo nivel de comprobación en todos los casos. La mayoría de las funcionalidades principales fueron verificadas mediante acciones realizadas desde la plataforma y consultas posteriores en MySQL. No obstante, el cálculo detallado del puntaje, la integración exitosa con el servicio de inteligencia artificial y los códigos HTTP de determinadas respuestas poseen limitaciones específicas en sus evidencias. Por esta razón, el análisis de cumplimiento considera tanto los resultados satisfactorios como el alcance real de las pruebas efectuadas.

El objetivo general del subproyecto fue el siguiente:

“Diseñar e implementar la infraestructura Back-End de Aurora, procurando su mantenibilidad, consistencia y capacidad de evolución, con el propósito de soportar los principales flujos funcionales de la plataforma”.

Los resultados obtenidos permiten concluir que este objetivo fue alcanzado de manera satisfactoria para el alcance funcional del subproyecto. Se implementó un servidor basado en FastAPI que centraliza el procesamiento de solicitudes, la aplicación de reglas de negocio, la comunicación con MySQL y la integración de los diferentes clientes de Aurora.

La solución permite atender endpoints REST, rutas especializadas para la aplicación móvil y vistas web renderizadas mediante Jinja2. Aunque estas modalidades son atendidas desde un único proceso de aplicación, la organización interna separa rutas, modelos, esquemas y funciones auxiliares según sus responsabilidades. Esta estructura corresponde a una arquitectura monolítica modular y no a un conjunto de microservicios independientes.

La consistencia funcional se comprobó principalmente mediante la relación entre las acciones ejecutadas en la plataforma y los valores persistidos en MySQL. El pre-check emocional, el desempeño académico, el post-check y el saldo de Glims fueron verificados mediante consultas anteriores y posteriores. De esta manera, se comprobó que la información procesada por el Back-End no permaneció únicamente en la interfaz o en la sesión HTTP del estudiante.

Asimismo, la capacidad de evolución se encuentra respaldada por la separación de responsabilidades, la utilización de routers organizados por dominio, el empleo de SQLAlchemy como capa de acceso a datos y la centralización de determinadas reglas de negocio en funciones auxiliares. Sin embargo, esta capacidad corresponde a una propiedad respaldada por el diseño y la organización del código. No fue medida mediante indicadores cuantitativos de mantenibilidad ni mediante la incorporación experimental de nuevos módulos.

Cumplimiento del primer objetivo específico

El primer objetivo específico consistió en:

“Definir una arquitectura Back-End modular, organizada mediante componentes con responsabilidades separadas, favoreciendo su mantenibilidad y evolución”.

Este objetivo se considera cumplido. La solución fue organizada mediante módulos diferenciados para las rutas web, los endpoints de la aplicación móvil, las operaciones generales de la API, los modelos de persistencia, los esquemas de validación y las funciones auxiliares de negocio.

La separación permite que los flujos de autenticación, evaluación, emociones, materiales, gamificación y progresión académica sean gestionados desde componentes identificables. A su vez, la inyección de sesiones de base de datos mediante dependencias facilita la apertura y cierre controlado de las conexiones utilizadas durante cada solicitud.

Los casos de prueba no midieron directamente la mantenibilidad de esta arquitectura. No obstante, la ejecución de funcionalidades correspondientes a estudiantes y profesores demuestra que los módulos implementados pueden coexistir dentro del mismo servidor y compartir la capa de persistencia sin impedir la continuidad de los flujos principales.

Por consiguiente, el cumplimiento de este objetivo se sustenta principalmente en la estructura implementada y en su funcionamiento integrado. Una evaluación más exhaustiva requeriría aplicar métricas de acoplamiento, cohesión, complejidad o mantenibilidad sobre el código fuente.

Cumplimiento del segundo objetivo específico

El segundo objetivo específico estableció:

“Diseñar y estructurar la lógica de negocio del sistema, gestionando correctamente el acceso e integración de sus componentes”.

Este objetivo se considera cumplido. El Back-End incorpora reglas de negocio que no se limitan al almacenamiento de información, sino que controlan el desarrollo del recorrido académico.

Los casos CP-01 y CP-08 comprobaron el acceso y redireccionamiento de los roles Estudiante y Profesor. Los casos CP-10 y CP-16 verificaron la persistencia diferenciada de los registros emocionales previo y posterior a una evaluación. Los casos CP-11, CP-12 y CP-13 examinaron la continuidad del test, la generación del resultado y la persistencia del desempeño.

La gamificación fue evaluada mediante los casos CP-14 y CP-15. En el primer escenario, un puntaje de 25 produjo una recompensa de 25 Glims, aumentando el saldo desde 30 hasta 55. En la repetición posterior, el estudiante obtuvo nuevamente 25 puntos, por lo que el Back-End aplicó la regla incremental y asignó 0 Glims. El saldo permaneció en 55, evitando que la repetición de una actividad sin mejora generara recompensas duplicadas.

El caso CP-17 demostró la aplicación de la regla de desbloqueo progresivo. La Unidad 2 se encontraba inicialmente bloqueada y fue habilitada después de comprobarse la existencia del pre-check, el desempeño y el post-check requeridos para la sesión anterior.

Estos resultados permiten concluir que la lógica de negocio implementada relaciona autenticación, evaluación, emociones, persistencia, gamificación y progresión académica. No obstante, las pruebas se concentraron en escenarios representativos y no abarcaron todas las combinaciones posibles de respuestas, puntajes, sesiones y unidades.

Cumplimiento del tercer objetivo específico

El tercer objetivo específico fue:

“Implementar modelos y relaciones en una base de datos relacional, adaptados a la estructura del dominio educativo de Aurora”.

Este objetivo se considera cumplido. La solución utiliza MySQL como sistema de gestión de base de datos y SQLAlchemy como herramienta de mapeo entre las entidades del dominio y las operaciones realizadas por el Back-End.

El modelo implementado permite representar personas, roles, establecimientos, asignaturas, unidades, sesiones, evaluaciones, preguntas, emociones, desempeños, materiales, registros de inteligencia artificial y componentes de gamificación. Las relaciones permiten vincular las acciones realizadas con el estudiante, el colegio y la sesión académica correspondiente.

La persistencia fue comprobada directamente en distintos casos. CP-09 demostró la carga de un material desde el módulo del Profesor y su disponibilidad posterior en el módulo de estudio del Estudiante. CP-10 confirmó el registro del pre-check en `estudiante_has_emocion`. CP-13 comprobó la creación de un desempeño con `NotaTest = 25`, mientras que CP-14 verificó el aumento del saldo almacenado desde 30 hasta 55 Glims. CP-16 confirmó la existencia de un registro posterior con `Tipo = post`.

El caso CP-18 también demostró que la interacción con el chatbot y el mensaje de error producido ante la indisponibilidad del proveedor externo fueron conservados en `registro_ia`. Aunque este caso no validó una respuesta satisfactoria generada por el modelo de inteligencia artificial, sí permitió comprobar la persistencia del evento ocurrido.

La utilización de una base de datos relacional permitió mantener vínculos explícitos entre las entidades del dominio. Sin embargo, las pruebas no constituyen una evaluación exhaustiva de integridad referencial ni examinan el comportamiento del sistema frente a un volumen elevado de registros o transacciones concurrentes.

Cumplimiento del cuarto objetivo específico

El cuarto objetivo específico consistió en:

“Desarrollar las rutas internas del sistema, cubriendo los flujos esenciales como login, sesiones, test y seguimiento emocional”.

Este objetivo se considera cumplido. Los casos CP-01 a CP-09 permitieron comprobar la autenticación, redirección por rol, navegación del estudiante, visualización de asignaturas, unidades, sesiones y evaluaciones, además de la carga de materiales desde el perfil del Profesor.

Los casos CP-10 a CP-17 evaluaron las operaciones principales del recorrido académico. El flujo validado comenzó con el pre-check emocional, continuó con el desarrollo de la evaluación y el cálculo del resultado, persistió el desempeño, actualizó los Glims, solicitó el post-check y habilitó la unidad siguiente cuando se cumplieron las condiciones establecidas.

Los casos CP-19 y CP-20 demostraron que determinadas solicitudes inválidas son tratadas mediante respuestas controladas. El primero rechazó una ruta de descarga no permitida mediante el mensaje `{"detail":"Ruta inválida"}`, mientras que el segundo reconoció una sesión inexistente y devolvió `{"detail":"Sesión no encontrada"}`.

En ambos casos, las capturas disponibles muestran el contenido de la respuesta, pero no permiten observar directamente el código de estado HTTP. Por ello, la validación acredita el comportamiento observable y el contenido entregado, pero no demuestra visualmente el código HTTP asociado.

Cumplimiento del quinto objetivo específico

El quinto objetivo específico estableció:

“Establecer lineamientos de documentación, mantenimiento y control de versiones, facilitando la evolución futura del sistema”.

Este objetivo se considera cumplido para el nivel alcanzado por el subproyecto. FastAPI genera una especificación OpenAPI a partir de las rutas y esquemas declarados, proporcionando una base para consultar y probar los endpoints disponibles. La memoria también documenta la arquitectura, el modelo de persistencia, las principales reglas de negocio, las decisiones técnicas y la estrategia de validación aplicada.

El uso de control de versiones permitió registrar los cambios realizados durante el desarrollo y mantener una estructura compartida para el trabajo del equipo. Además, la organización modular del código y la separación de configuraciones contribuyen a facilitar futuras modificaciones.

No obstante, estos mecanismos representan un nivel inicial de documentación y mantenimiento. La evolución hacia un entorno productivo requiere complementar lo realizado con una estrategia formal de migraciones, automatización de pruebas, integración continua, documentación de despliegue, registro de cambios y procedimientos operacionales.

La Tabla 5.1 resume el cumplimiento de los objetivos definidos para el subproyecto.

Tabla 5.1. Cumplimiento de los objetivos del subproyecto Back-End de Aurora.

Objetivo	Evidencia principal	Evaluación
Objetivo general	Implementación del servidor FastAPI, integración con MySQL y funcionamiento de los principales flujos evaluados mediante CP-01 a CP-20.	Cumplido para el alcance funcional validado.
Objetivo específico 1	Organización modular mediante routers, modelos, esquemas, dependencias y funciones auxiliares con responsabilidades diferenciadas.	Cumplido a nivel de diseño e implementación.
Objetivo específico 2	Autenticación por rol, evaluaciones, registros emocionales, desempeño, Glms y desbloqueo progresivo.	Cumplido en los escenarios ejecutados.

Objetivo específico 3	Persistencia relacional de materiales, emociones, desempeño, saldo e interacciones del chatbot.	Cumplido en las operaciones verificadas.
Objetivo específico 4	Funcionamiento de rutas de login, navegación, sesiones, test, emociones, materiales y manejo de solicitudes inválidas.	Cumplido, con limitaciones específicas en algunas evidencias.
Objetivo específico 5	Documentación de la arquitectura y API, organización del código y utilización de control de versiones.	Cumplido en un nivel inicial, susceptible de fortalecimiento.

Fuente: elaboración propia.

En términos generales, la validación permite concluir que el Back-End proporciona la infraestructura necesaria para soportar los principales flujos funcionales de Aurora. Sin embargo, este cumplimiento no debe interpretarse como una certificación de preparación para producción. Las pruebas acreditan el comportamiento del prototipo en el entorno y los escenarios evaluados, pero no comprueban propiedades como alta disponibilidad, escalabilidad bajo carga, seguridad integral o efectividad pedagógica.

5.2 Contribuciones e impacto del trabajo realizado

La principal contribución del trabajo corresponde al diseño e implementación de una infraestructura Back-End capaz de articular los componentes que forman parte de Aurora. El servidor concentra la lógica necesaria para recibir solicitudes desde los distintos clientes, procesar las reglas del dominio educativo, consultar o modificar la información persistida y devolver las respuestas requeridas por cada flujo.

Desde una perspectiva técnica, la utilización de FastAPI permitió construir una solución que combina endpoints JSON, rutas destinadas a la aplicación móvil y vistas web renderizadas mediante Jinja2. Esta arquitectura híbrida permitió responder a las necesidades del proyecto sin desplegar servidores independientes para cada cliente.

La organización modular constituye otra contribución relevante. Aunque Aurora se ejecuta como una aplicación monolítica, sus responsabilidades se encuentran distribuidas entre routers, modelos, esquemas, funciones auxiliares y servicios. Esta separación facilita localizar las operaciones asociadas con cada dominio y disminuye la necesidad de concentrar toda la lógica en un único componente.

En relación con la persistencia, el trabajo permitió representar el dominio educativo mediante un modelo relacional. La información correspondiente a estudiantes, profesores, establecimientos, asignaturas, sesiones, evaluaciones, emociones y desempeños puede mantenerse vinculada y ser reutilizada por diferentes funcionalidades.

Una contribución particularmente significativa se encuentra en la lógica de evaluación y progresión. El Back-End administra el recorrido formado por el pre-check, el test, el cálculo del puntaje, el desempeño y el post-check. La finalización de estas etapas se utiliza posteriormente para determinar si el estudiante puede acceder a nuevas unidades.

La economía de Glims también representa un aporte funcional. La recompensa se calcula a partir de la diferencia positiva entre el nuevo puntaje y la mejor marca anterior:

$$\Delta\text{Glims} = \max(0, \text{puntaje nuevo} - \text{mejor puntaje previo})$$

Esta regla permite orientar la recompensa hacia la mejora del desempeño. En el escenario validado, el primer resultado de 25 puntos generó 25 Glims, mientras que la repetición con el mismo resultado no produjo una nueva recompensa. De esta manera, el Back-End evita que la moneda virtual pueda acumularse únicamente mediante la repetición mecánica de una actividad.

El seguimiento emocional constituye otra contribución del subproyecto. La persistencia diferenciada de los registros pre y post permite conservar la percepción comunicada por el estudiante antes y después de una sesión académica. Estos datos pueden ser utilizados por las funcionalidades de seguimiento de Aurora, sin que su almacenamiento implique una interpretación clínica o psicológica automática.

La integración del chatbot demuestra la posibilidad de conectar el Back-End con un servicio externo de inteligencia artificial y mantener el historial de las interacciones. La prueba realizada no permitió validar una respuesta exitosa del proveedor, pero sí comprobó que el sistema conservó el mensaje del estudiante y el error controlado generado ante la indisponibilidad de la dependencia.

Desde una perspectiva metodológica, la combinación de evidencias obtenidas desde la interfaz y consultas realizadas en MySQL constituye un aporte para la validación del trabajo. Esta estrategia permitió diferenciar entre un resultado solamente visible y una modificación efectivamente persistida.

En cuanto a su impacto, el Back-End entrega la base técnica necesaria para que Aurora pueda desarrollar una experiencia educativa orientada a la autorregulación. Sin embargo, el trabajo realizado no evaluó directamente cambios en el aprendizaje, motivación o autorregulación de estudiantes reales. Por consiguiente, el impacto educativo debe entenderse como una capacidad potencial habilitada por la infraestructura y no como un efecto pedagógico demostrado.

También debe considerarse que Aurora corresponde a un proyecto grupal. El aporte desarrollado y analizado en esta memoria se concentra exclusivamente en el Back-End. La aplicación móvil y los demás componentes construidos por integrantes del equipo se relacionan con este trabajo como clientes o consumidores de sus servicios, pero no forman parte de la autoría técnica evaluada en el presente subproyecto.

5.3 Alcances, limitaciones y aprendizajes

El alcance del trabajo comprendió el diseño, implementación y validación funcional del Back-End de Aurora. Se incluyeron la organización de la arquitectura, el acceso a MySQL, la definición de modelos y relaciones, la autenticación web, las rutas para los distintos clientes, la gestión de evaluaciones, la persistencia emocional, el desempeño académico, la gamificación, la administración de materiales y la integración con servicios externos.

La validación permitió comprobar un recorrido funcional representativo para los roles Estudiante y Profesor. No obstante, los resultados deben interpretarse considerando las siguientes limitaciones.

En primer lugar, las pruebas fueron ejecutadas en un entorno controlado y principalmente mediante procedimientos manuales. Los casos permitieron verificar los escenarios definidos, pero no constituyen una cobertura exhaustiva del código ni de todas las combinaciones posibles de datos. Tampoco se ejecutaron pruebas automatizadas de regresión, integración continua o cobertura estructural.

En segundo lugar, no se realizaron pruebas de carga, estrés, rendimiento o disponibilidad. Por este motivo, no se puede determinar cuántos usuarios simultáneos podría atender el servidor ni cómo se comportaría MySQL frente a un volumen elevado de solicitudes. La arquitectura permite una evolución posterior, pero la escalabilidad productiva no fue demostrada experimentalmente.

Una tercera limitación corresponde a la seguridad de la autenticación. La plataforma web utiliza sesiones almacenadas en cookies firmadas, lo que permite detectar alteraciones en su contenido. Sin embargo, las contraseñas son comparadas con valores almacenados sin hashing y el acceso móvil no utiliza un mecanismo basado en tokens como JWT. Por consiguiente, la autenticación implementada resulta funcional para el prototipo, pero requiere fortalecimiento antes de utilizarse con información real.

También se identificó que la configuración CORS permite orígenes abiertos. Esta decisión facilita la integración durante el desarrollo, pero no corresponde a una configuración recomendable para producción. En un despliegue real deberían definirse explícitamente los dominios autorizados.

En relación con la evaluación, el caso CP-12 comprobó el resultado final de 25 puntos, pero las capturas no desglosan visualmente la contribución de cada tipo de pregunta. Además, las respuestas de desarrollo son evaluadas únicamente según la presencia de contenido y no mediante un análisis semántico. Por lo tanto, no corresponde afirmar que el sistema determina la calidad académica de una respuesta abierta.

La integración con inteligencia artificial presenta otra limitación. El caso CP-18 fue ejecutado cuando la credencial del servicio externo no se encontraba operativa. La prueba permitió observar un mensaje controlado y la persistencia de la interacción, pero no validó una respuesta satisfactoria generada por el modelo. Tampoco se evaluaron la calidad, pertinencia, seguridad o exactitud pedagógica de las respuestas.

Los casos CP-19 y CP-20 comprobaron mensajes JSON controlados para una ruta inválida y una sesión inexistente. Sin embargo, las evidencias visuales no muestran directamente los códigos HTTP. En consecuencia, no deben utilizarse para afirmar que se comprobaron visualmente estados específicos como HTTP 400 o HTTP 404.

Desde la perspectiva de mantenibilidad, se identificaron elementos que pueden mejorarse. Parte de la lógica asociada con minijuegos utiliza consultas SQL directas en lugar de modelos ORM. También existe una combinación de convenciones pertenecientes a diferentes versiones de Pydantic y determinadas configuraciones se encuentran incorporadas directamente en algunos componentes. Estos aspectos no impidieron las validaciones realizadas, pero aumentan el costo de futuras modificaciones.

El almacenamiento de materiales se realiza en el sistema de archivos del servidor. Esta estrategia es suficiente para el entorno local, pero puede dificultar la escalabilidad horizontal, el respaldo y la disponibilidad de archivos en múltiples instancias.

Otra limitación importante es la ausencia de una validación con usuarios finales. No se efectuó un estudio con estudiantes de sexto básico, profesores o establecimientos educacionales. En consecuencia, esta memoria no permite concluir que Aurora mejore efectivamente la autorregulación, el aprendizaje, la motivación o el bienestar emocional.

A pesar de estas limitaciones, el desarrollo permitió obtener aprendizajes relevantes. El primero corresponde a la importancia de mantener la documentación alineada con la implementación real. Durante la revisión fue necesario diferenciar la arquitectura efectivamente construida de mecanismos que no se encontraban presentes, evitando atribuir al sistema tecnologías como JWT, cifrado de contraseñas o microservicios.

Un segundo aprendizaje se relaciona con la validación de la persistencia. Una confirmación mostrada en la interfaz no demuestra por sí sola que los datos hayan sido almacenados. La comparación entre estados anteriores y posteriores en MySQL permitió construir evidencia más sólida sobre el funcionamiento del Back-End.

También se comprobó la importancia de centralizar las reglas de negocio. Operaciones como el cálculo de Glims, la creación del desempeño y el desbloqueo de unidades deben ser resueltas por el servidor, evitando que su aplicación dependa exclusivamente de la interfaz utilizada.

Otro aprendizaje corresponde al tratamiento de dependencias externas. Los servicios de inteligencia artificial pueden encontrarse temporalmente no disponibles o rechazar una solicitud. El Back-End debe manejar estas situaciones mediante tiempos de espera, respuestas controladas y mecanismos de registro que permitan conservar la trazabilidad.

Finalmente, el trabajo demostró que una arquitectura monolítica modular puede resultar adecuada para un prototipo educativo de este alcance. Esta alternativa reduce la complejidad de despliegue y, al mismo tiempo, permite mantener responsabilidades separadas. La evolución hacia arquitecturas distribuidas debería responder a necesidades comprobadas y no solamente a una expectativa teórica de escalabilidad.

5.4 Recomendaciones y trabajo futuro

Las recomendaciones y líneas de trabajo futuro se orientan a evolucionar el Back-End desde un prototipo funcional hacia una solución con mayores niveles de seguridad, calidad, mantenibilidad y preparación operativa.

La primera prioridad debe ser fortalecer la autenticación. Las contraseñas deberían almacenarse utilizando un algoritmo de hashing adecuado, como Argon2id, acompañado de políticas para la creación y recuperación segura de credenciales. El acceso móvil debería incorporar autenticación basada en tokens con expiración, renovación y verificación de roles.

Las cookies utilizadas por la plataforma web deberían configurarse con atributos como HttpOnly, Secure y SameSite, de acuerdo con las características del entorno de despliegue. También deberían definirse explícitamente los dominios permitidos por CORS, reemplazando la configuración abierta utilizada durante el desarrollo.

La segunda recomendación consiste en implementar una estrategia automatizada de pruebas. Esta debería incluir pruebas unitarias para las reglas de negocio, pruebas de integración para las operaciones con MySQL y pruebas de extremo a extremo para los principales recorridos del estudiante y del profesor.

Entre los escenarios prioritarios se encuentran las diferentes modalidades de cálculo de puntaje, la repetición de evaluaciones, la asignación incremental de Glims, el desbloqueo de múltiples unidades, las compras concurrentes, el rechazo de rutas de archivo y el tratamiento de entidades inexistentes. Las pruebas deberían registrar tanto el contenido como el código HTTP de cada respuesta.

También se recomienda integrar estas comprobaciones en un proceso de integración continua. De esta forma, cada cambio podría ser validado automáticamente antes de incorporarse a la versión principal del proyecto.

En el ámbito de la persistencia, resulta conveniente incorporar una herramienta de migraciones como Alembic. Esto permitiría registrar de manera ordenada las modificaciones realizadas sobre el esquema, reproducirlas en diferentes entornos y disminuir el riesgo de divergencias entre las bases de datos.

También debería completarse la representación ORM de las entidades que actualmente utilizan consultas SQL directas, particularmente aquellas asociadas con minijuegos. La estandarización sobre una única versión de Pydantic y la eliminación de valores incorporados directamente en componentes específicos contribuirían a disminuir la deuda técnica.

La integración con inteligencia artificial debería validarse nuevamente utilizando una credencial operativa y contemplando tanto respuestas satisfactorias como fallos del proveedor. El Back-End debería definir tiempos máximos de espera, reintentos limitados, control de excepciones y respuestas alternativas cuando el servicio no se encuentre disponible.

Además, se recomienda revisar la información enviada al proveedor externo, limitar la inclusión de datos personales y establecer mecanismos de supervisión sobre las respuestas generadas. Antes de utilizar esta funcionalidad con estudiantes, debería evaluarse su pertinencia pedagógica junto con profesionales del área educativa. La inteligencia artificial no debería presentarse como un mecanismo de diagnóstico emocional ni reemplazar la evaluación realizada por especialistas.

Los registros emocionales requieren medidas específicas de privacidad. Una evolución futura debería definir quién puede consultar estos datos, durante cuánto tiempo se conservan y con qué propósito son utilizados. También sería conveniente incorporar registros de auditoría y mecanismos de anonimización o seudonimización para análisis agregados.

Respecto de los materiales didácticos, el almacenamiento local puede reemplazarse por un servicio de almacenamiento de objetos. Esta evolución debería mantener la validación de rutas e incorporar restricciones de tamaño, tipo de contenido, extensión y nombre. Para archivos privados podrían utilizarse enlaces temporales y controles de autorización antes de cada descarga.

También se recomienda ejecutar pruebas de rendimiento. Estas pruebas deberían medir tiempos de respuesta, consumo de recursos, comportamiento de la conexión con MySQL y capacidad frente a solicitudes concurrentes. Los resultados permitirían decidir si se requiere optimizar consultas, incorporar índices, aplicar paginación, utilizar caché o desplegar múltiples instancias del servidor.

La Tabla 5.2 presenta una priorización de las principales mejoras propuestas.

Tabla 5.2. Priorización de recomendaciones para la evolución del Back-End.

Área	Acción recomendada	Prioridad	Resultado esperado
Autenticación	Incorporar hashing de contraseñas y autenticación segura para el cliente móvil.	Alta	Reducir el riesgo de exposición y uso indebido de credenciales
Configuración	Restringir CORS, proteger cookies y administrar secretos mediante variables de entorno.	Alta	Fortalecer la seguridad del despliegue.
Pruebas	Implementar pruebas unitarias, de integración y de extremo a extremo.	Alta	Detectar regresiones y aumentar la confiabilidad de los cambios.

Persistencia	Incorporar migraciones y completar la representación ORM.	Alta	Mantener el esquema de datos de manera reproducible.
Privacidad	Definir acceso, retención y auditoría de los registros emocionales.	Alta	Proteger información sensible de los estudiantes.
Inteligencia artificial	Validar respuestas exitosas, errores, privacidad y pertinencia pedagógica.	Media-alta	Utilizar la dependencia externa de manera controlada.
Archivos	Migrar los materiales a almacenamiento administrado y reforzar sus validaciones.	Media	Mejorar disponibilidad, respaldo y escalabilidad.
Rendimiento	Ejecutar pruebas de carga y optimizar consultas e índices.	Media	Determinar la capacidad real del sistema.
Documentación	Ampliar las guías de instalación, despliegue, migración y operación.	Media	Facilitar la incorporación de nuevos desarrolladores.
Validación educativa	Realizar un estudio piloto con estudiantes y profesores.	Media-alta	Evaluar usabilidad, aceptación y aporte pedagógico real.

Fuente: elaboración propia.

Finalmente, se recomienda desarrollar una validación educativa con usuarios finales. Un estudio piloto permitiría observar la facilidad de uso, comprensión de los flujos, percepción de los registros emocionales y efecto de la gamificación. Esta evaluación debería contar con consentimiento informado, resguardo de los datos y criterios definidos junto con especialistas del área educativa.

En conclusión, el trabajo desarrollado proporciona una base funcional para sostener los principales flujos de Aurora. El Back-End implementado permite integrar los clientes del sistema con MySQL, administrar reglas académicas, conservar información emocional,

procesar evaluaciones, controlar recompensas y responder de manera identificable frente a determinados errores.

Los resultados demuestran el cumplimiento de los objetivos establecidos para el subproyecto dentro del alcance evaluado. Al mismo tiempo, las limitaciones identificadas muestran que todavía existen tareas necesarias antes de considerar la solución preparada para un entorno productivo o afirmar un impacto educativo comprobado.

Por consiguiente, el Back-End de Aurora debe entenderse como una infraestructura funcional y evolutiva sobre la cual pueden incorporarse mejoras de seguridad, automatización, escalabilidad, privacidad y validación pedagógica. Su principal aporte consiste en transformar los requerimientos educativos y funcionales del proyecto en reglas, datos y servicios capaces de operar de manera integrada, manteniendo una base técnica para la continuidad futura de la plataforma.

REFERENCIAS BIBLIOGRÁFICAS

Deterding, S., Dixon, D., Khaled, R., & Nacke, L. (2011). From game design elements to gamefulness: Defining “gamification”. En *Proceedings of the 15th International Academic MindTrek Conference: Envisioning Future Media Environments* (pp. 9–15). Association for Computing Machinery. <https://doi.org/10.1145/2181037.2181040>

Fielding, R. T. (2000). *Architectural styles and the design of network-based software architectures* [Tesis doctoral, University of California, Irvine]. <https://ics.uci.edu/~fielding/pubs/dissertation/top.htm>

OpenAPI Initiative. (2024). *OpenAPI specification* (versión 3.1.1). <https://spec.openapis.org/oas/v3.1.1.html>

OWASP Foundation. (s. f.). *Password storage cheat sheet*. OWASP Cheat Sheet Series. Recuperado el 31 de julio de 2026, de https://cheatsheetseries.owasp.org/cheatsheets/Password_Storage_Cheat_Sheet.html

Pintrich, P. R. (2000). The role of goal orientation in self-regulated learning. En M. Boekaerts, P. R. Pintrich, & M. Zeidner (Eds.), *Handbook of self-regulation* (pp. 451–502). Academic Press. <https://www.sciencedirect.com/science/chapter/edited-volume/abs/pii/B9780121098902500433?via%3Dihub>

Ryan, R. M., & Deci, E. L. (2000). Self-determination theory and the facilitation of intrinsic motivation, social development, and well-being. *American Psychologist*, 55(1), 68–78. <https://psycnet.apa.org/doiLanding?doi=10.1037%2F0003-066X.55.1.68>

Schwaber, K., & Sutherland, J. (2020). *The Scrum guide: The definitive guide to Scrum: The rules of the game*. Scrum Guides. <https://scrumguides.org/docs/scrumguide/v2020/2020-Scrum-Guide-US.pdf>

Schunk, D. H., & Zimmerman, B. J. (Eds.). (2012). *Motivation and self-regulated learning: Theory, research, and applications*. Routledge. <https://www.taylorfrancis.com/books/edit/10.4324/9780203831076/motivation-self-regulated-learning-dale-schunk-barry-zimmerman>

Zimmerman, B. J. (2000). Attaining self-regulation: A social cognitive perspective. En M. Boekaerts, P. R. Pintrich, & M. Zeidner (Eds.), *Handbook of self-regulation* (pp. 13–39). Academic Press. <https://doi.org/10.1016/B978-012109890-2/50031-7>

Zimmerman, B. J. (2002). Becoming a self-regulated learner: An overview. *Theory Into Practice*, 41(2), 64–70. https://doi.org/10.1207/s15430421tip4102_2

Ministerio de Educación de Chile. (2012). *Bases Curriculares Educación Básica*. [Currículum Nacional](#).

Ministerio de Educación de Chile. (2018). *Decreto N.º 67: Normas mínimas nacionales sobre evaluación, calificación y promoción*. [Biblioteca del Congreso Nacional](#).

UNICEF. (2011). *Estado Mundial de la Infancia 2011: La adolescencia, una época de oportunidades*. <https://data.unicef.org/resources/the-state-of-the-worlds-children-2011-adolescents-an-age-of-opportunity/>

Agencia de Calidad de la Educación. (2024). Aplicación Simce 2024 a sexto básico. <https://www.agenciaeducacion.cl/con-un-9885-de-cobertura-a-nivel-nacional-finalizo-la-aplicacion-del-simce-2024-a-los-sextos-basicos-del-pais/>

[Khan Academy](#), [Smartick](#), [Colegium](#), [Napsis](#) y [Lirmi](#): sitios oficiales utilizados para validar el análisis de competencia.

Anexo A. MATRIZ CONSOLIDADA DE CASOS DE PRUEBA

El presente anexo reúne los casos ejecutados durante la validación funcional del Back-End de Aurora. La matriz sintetiza las precondiciones, el procedimiento aplicado, los resultados esperados y obtenidos, las evidencias asociadas y el estado final de cada comprobación.

Las pruebas fueron desarrolladas en un entorno controlado y se concentraron en los principales procesos gestionados por el servidor. Por consiguiente, los resultados acreditan el comportamiento observado en los escenarios ejecutados, pero no constituyen una cobertura exhaustiva del sistema ni una certificación de preparación para producción.

Tabla A.1. Matriz consolidada de casos de prueba del Back-End de Aurora.

Código	Funcionalidad	Precondición	Procedimiento resumido	Resultado esperado	Resultado obtenido	Evidencia	Estado
CP-00	Comunicación FastAPI–MySQL	Servidor FastAPI operativo y base de datos cargada.	Consultar un endpoint geográfico y comparar la respuesta con los registros de MySQL.	El endpoint debe responder y devolver los mismos datos almacenados.	Se obtuvo una respuesta satisfactoria y los registros coincidieron con MySQL.	CAP4_CP00_01, CAP4_CP00_02	Aprobado
CP-01	Autenticación del Estudiante	Usuario de prueba registrado con Rol Estudiante.	Intentar ingresar con credenciales inválidas y posteriormente con credenciales válidas.	Rechazar el acceso inválido y redirigir al Estudiante autenticado.	Las credenciales incorrectas fueron rechazadas y el usuario válido fue dirigido a su panel.	CAP4_CP01_01, CAP4_CP01_02	Aprobado
CP-02	Módulo de estudio	Estudiante autenticado y materiales registrados.	Acceder al módulo de estudio.	Mostrar los materiales disponibles y sus controles.	La vista presentó materiales con opciones para abrirlos y descargarlos.	CAP4_CP02_01	Aprobado para interfaz y consulta de materiales
CP-03	Navegación por asignaturas	Estudiante autenticado con asignaturas asociadas.	Acceder al recorrido académico y revisar las asignaturas disponibles.	Mostrar las asignaturas correspondientes al estudiante.	La interfaz cargó las opciones disponibles.	CAP4_CP03_01	Parcial: interfaz verificada
CP-04	Visualización y bloqueo de unidades	Asignatura con unidades y prerrequis	Consultar las unidades antes y después de	Mostrar las unidades y mantener bloqueadas	Se observó el bloqueo inicial y posteriormente el	CAP4_CP04_01, CAP4_CP17_02	Aprobado junto con CP-17

Código	Funcionalidad	Precondición	Procedimiento resumido	Resultado esperado	Resultado obtenido	Evidencia	Estado
		datos configurados.	completar las condiciones requeridas.	aquellas cuyos requisitos no se hayan cumplido.	desbloqueo de la Unidad 2 en CP-17.		
CP-05	Visualización de sesiones	Unidad académica seleccionada.	Acceder a la vista de sesiones de la unidad.	Mostrar las sesiones correspondientes y sus controles.	La interfaz cargó las sesiones disponibles.	CAP4_CP05_01	Parcial: interfaz verificada
CP-06	Formulario de Pre-check	Estudiante dentro de una sesión académica.	Iniciar la sesión y abrir el formulario emocional previo.	Presentar el formulario de Pre-check antes de la evaluación.	El formulario se mostró correctamente; su persistencia fue comprobada en CP-10.	CAP4_CP06_01, evidencias de CP-10	Aprobado junto con CP-10
CP-07	Evaluación interactiva	Pre-check completado y test disponible.	Ingresar a la evaluación, visualizar la primera pregunta y continuar el recorrido.	Mostrar las preguntas y permitir avanzar en el test.	La evaluación se presentó y el flujo pudo completarse posteriormente en CP-11.	CAP4_CP07_01, evidencias de CP-11	Aprobado para continuidad del flujo
CP-08	Autenticación del Profesor	Usuario registrado con Rol Profesor.	Iniciar sesión con las credenciales del Profesor.	Redirigir al usuario hacia el menú correspondiente a su rol.	El inicio de sesión condujo al menú del Profesor.	CAP4_CP08_01	Aprobado
CP-09	Carga y publicación de material	Profesor autenticado y material didáctico disponible para cargar.	Revisar el estado inicial, cargar el material y comprobar su disponibilidad desde el perfil del Estudiante.	Confirmar la carga y mostrar posteriormente el recurso en el módulo de estudio.	El sistema informó la carga correcta y el material apareció después en /estudio.	CAP4_CP02_01, CAP4_CP09_01, CAP4_CP09_02	Aprobado
CP-10	Persistencia del Pre-check	Estudiante dentro de una sesión y sin el nuevo registro	Consultar el estado inicial, enviar el formulario y consultar	Insertar el registro emocional previo después del envío.	El registro fue almacenado en estudiante_has_emocion.	CAP4_CP10_01, CAP4_CP10_02, CAP4_CP10_03	Aprobado

Código	Funcionalidad	Precondición	Procedimiento resumido	Resultado esperado	Resultado obtenido	Evidencia	Estado
		emocional.	nuevamente MySQL.				
CP-11	Flujo completo de evaluación	Pre-check completado, saldo inicial de 30 Glims y ausencia de desempeño o previo.	Finalizar la evaluación y comparar los estados anteriores y posteriores.	Mostrar el resultado y persistir el desempeño y la recompensa.	Aurora mostró 25 puntos y 25 Glims; MySQL registró NotaTest = 25 y el saldo aumentó de 30 a 55.	Cinco capturas de CP-11	Aprobado
CP-12	Cálculo del puntaje	Respuestas del test registradas durante CP-11.	Procesar la evaluación y observar el puntaje final.	Calcular el resultado según las reglas asociadas con cada tipo de pregunta.	Se obtuvo un resultado final de 25 puntos, pero las capturas no desglosan la contribución de cada pregunta.	CAP4_CP11_02, CAP4_CP11_03	Parcial: resultado final verificado
CP-13	Persistencia del desempeño	Ausencia de desempeño o previo para el test ejecutado.	Comparar la tabla desempeño antes y después de finalizar la evaluación.	Crear o actualizar el registro correspondiente.	MySQL presentó un registro con NotaTest = 25.	CAP4_CP11_01, CAP4_CP11_03	Aprobado
CP-14	Asignación de Glims por mejora	Saldo inicial de 30 Glims y ausencia de una mejor marca anterior.	Finalizar la evaluación y comparar el saldo anterior y posterior.	Incrementar el saldo según la mejora obtenida.	La interfaz informó 25 Glims y el saldo aumentó de 30 a 55.	CAP4_CP11_01_db antes2, CAP4_CP11_02, CAP4_CP11_03_db después2	Aprobado
CP-15	Prevención de recompensas duplicadas	Mejor puntaje previo de 25 puntos y saldo de 55 Glims.	Repetir la evaluación sin superar el resultado anterior.	No conceder Glims adicionales.	La interfaz informó 0 Glims y el saldo permaneció en 55.	CAP4_CP15_01, CAP4_CP15_02	Aprobado para prevención de Glims duplicados
CP-16	Persistencia del Post-check	Evaluación terminada y formulario posterior	Seleccionar "Excelente", finalizar la sesión y	Almacenar el registro emocional posterior.	MySQL presentó Tipo = post, Escala = 5, Contexto = Excelente y SesiónId = 1.	CAP4_CP16_03	Aprobado

Código	Funcionalidad	Precondición	Procedimiento resumido	Resultado esperado	Resultado obtenido	Evidencia	Estado
		disponible.	consultar MySQL.				
CP-17	Desbloqueo progresivo	Unidad 2 inicialmente bloqueada y requisitos de la Unidad 1 completados.	Comprobar pre-check, desempeño y post-check; acceder nuevamente a la vista de unidades.	Habilitar la unidad siguiente al cumplirse las condiciones previas.	Los registros requeridos fueron comprobados y la Unidad 2 apareció habilitada.	CAP4_CP17_01, CAP4_CP17_02	Aprobado
CP-18	Manejo de indisponibilidad del chatbot	Chatbot accesible, consulta previa de registro_ia y credencial externa no operativa.	Enviar un mensaje, observar la respuesta y consultar posteriormente la tabla.	Presentar un error controlado y conservar la interacción.	La interfaz mostró "Error en la respuesta" y registro_ia almacenó el mensaje y el error.	CAP4_CP18_01, CAP4_CP18_02, CAP4_CP18_03	Aprobado con alcance limitado
CP-19	Validación de rutas de descarga	Endpoint de descarga disponible.	Solicitar deliberadamente una ruta fuera de la ubicación permitida.	Rechazar la ruta inválida mediante una respuesta controlada.	El navegador mostró {"detail": "Ruta inválida"}; el código HTTP no quedó visible.	CAP4_CP19_01	Parcial: respuesta controlada verificada
CP-20	Manejo de entidades inexistentes	No existe una sesión con identificador 99999.	Solicitar la sesión utilizando dicho identificador.	Informar que la sesión no existe y evitar continuar el procesamiento.	El navegador mostró {"detail": "Sesión no encontrada"}; el código HTTP no quedó visible.	CAP4_CP20_01	Aprobado para el comportamiento observable

Fuente: elaboración propia.

Anexo B. EVIDENCIAS COMPLEMENTARIAS DE VALIDACIÓN

B.1 Objetivo y alcance

El presente anexo reúne las evidencias complementarias obtenidas durante la validación funcional del Back-End de Aurora. Estas capturas respaldan la preparación del entorno, la continuidad de la navegación académica y los estados iniciales utilizados para comparar las modificaciones efectuadas durante los casos de prueba.

Las imágenes complementan las figuras principales incorporadas en el Capítulo 4. Su propósito es conservar la trazabilidad de las comprobaciones sin sobrecargar el cuerpo de la memoria con una captura por cada pantalla recorrida.

Las evidencias de interfaz acreditan únicamente el comportamiento observable durante los escenarios ejecutados. Por su parte, las consultas realizadas sobre MySQL muestran el estado de los datos en el momento de cada comprobación. En consecuencia, este material no representa una cobertura exhaustiva del sistema ni una certificación para su utilización en producción.

B.2 Verificación de la comunicación entre FastAPI y MySQL

Antes de iniciar los recorridos funcionales se comprobó que el servidor FastAPI pudiera responder solicitudes y recuperar información almacenada en MySQL. Para ello se consultó un endpoint geográfico y se comparó su respuesta con los registros presentes en la base de datos.

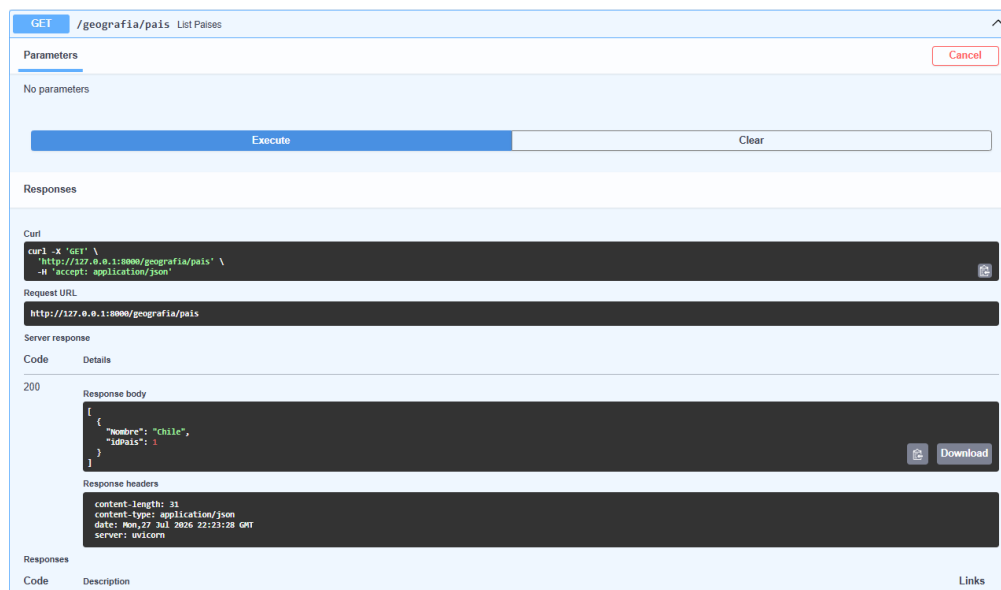


Figura B.1. Respuesta satisfactoria del endpoint geográfico mediante FastAPI.

Fuente: elaboración propia.

La evidencia muestra que el endpoint respondió y devolvió información geográfica en formato JSON.

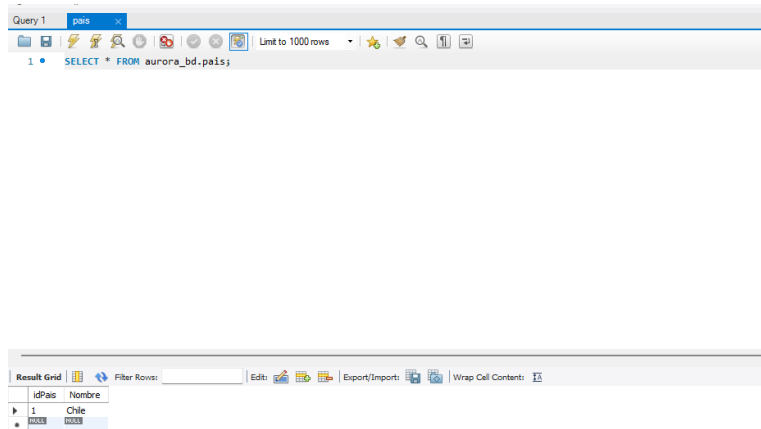


Figura B.2. Verificación de los datos geográficos consultados directamente en MySQL.

Fuente: elaboración propia.

La consulta permitió comprobar que los registros observados en MySQL coincidían con la información devuelta por el Back-End. Estas dos evidencias respaldan el caso CP-00.

B.3 Evidencias de navegación académica

Las siguientes capturas documentan las etapas intermedias recorridas por el estudiante desde el módulo de estudio hasta el inicio de la evaluación. Estas imágenes respaldan los casos CP-02 a CP-07.

Su alcance se limita a demostrar que las vistas correspondientes fueron presentadas y que el usuario pudo continuar la navegación. La persistencia y las reglas de negocio asociadas se comprobaron posteriormente mediante los casos CP-09, CP-10, CP-11 y CP-17.

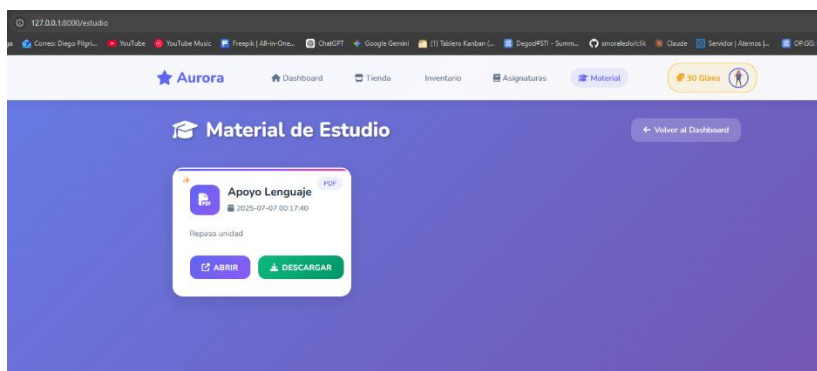


Figura B.3. Estado del módulo de estudio antes de cargar el nuevo material didáctico.

Fuente: elaboración propia.

Esta evidencia respalda CP-02 y también funciona como estado anterior para la comparación realizada en CP-09.

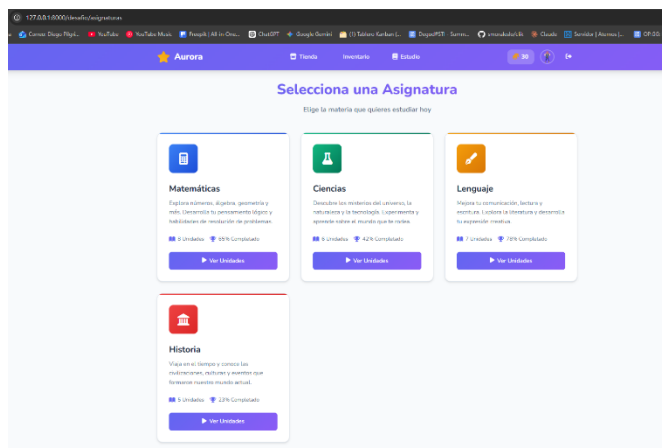


Figura B.4. Visualización de las asignaturas disponibles para el estudiante autenticado.

Fuente: elaboración propia.

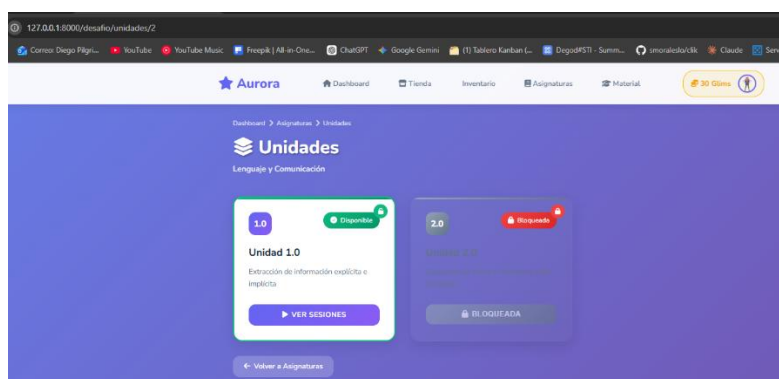


Figura B.5. Visualización de las unidades académicas y del estado de bloqueo previo al cumplimiento de los requisitos.

Fuente: elaboración propia.

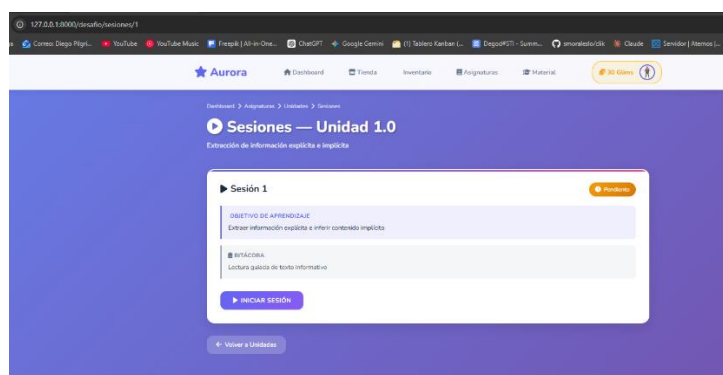


Figura B.6. Visualización de las sesiones disponibles dentro de la unidad académica seleccionada.

Fuente: elaboración propia.

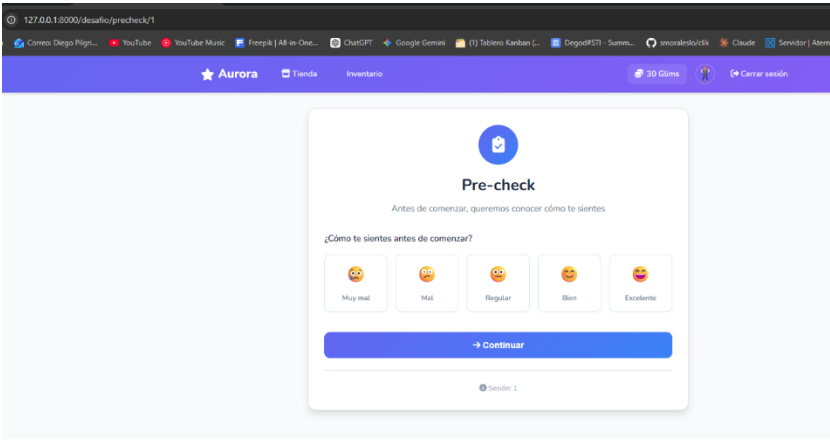


Figura B.7. Presentación del formulario emocional previo al comienzo de la evaluación.

Fuente: elaboración propia.

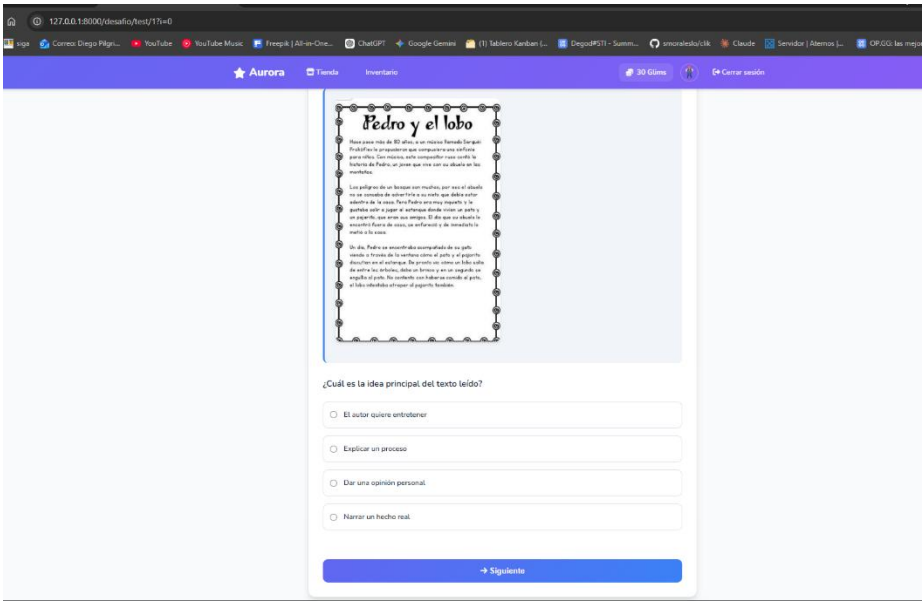


Figura B.8. Inicio de la evaluación interactiva y presentación de una pregunta al estudiante.

Fuente: elaboración propia.

B.4 Estado anterior al registro del Pre-check

Antes de enviar la selección emocional del estudiante se consultó la tabla estudiante_has_emocion. Esta operación estableció el estado inicial utilizado posteriormente para verificar la incorporación del nuevo registro.

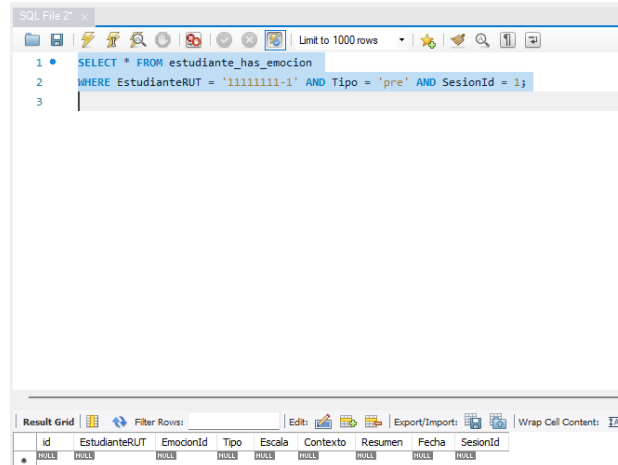


Figura B.9. Consulta de control en la base de datos antes del registro del chequeo emocional previo.

Fuente: elaboración propia.

Esta evidencia corresponde al estado anterior de CP-10. La acción realizada por el estudiante y el registro persistido después del envío permanecen documentados en las Figuras 4.7 y 4.8 del Capítulo 4. La memoria indica expresamente que esta consulta previa debe mantenerse como evidencia complementaria.

B.5 Estado anterior a la finalización de la evaluación

Para validar el flujo de evaluación se establecieron dos condiciones iniciales: la inexistencia de un desempeño previo para el test ejecutado y un saldo inicial de 30 Glims. Estas consultas proporcionaron la referencia necesaria para comparar los resultados almacenados después de finalizar la evaluación.

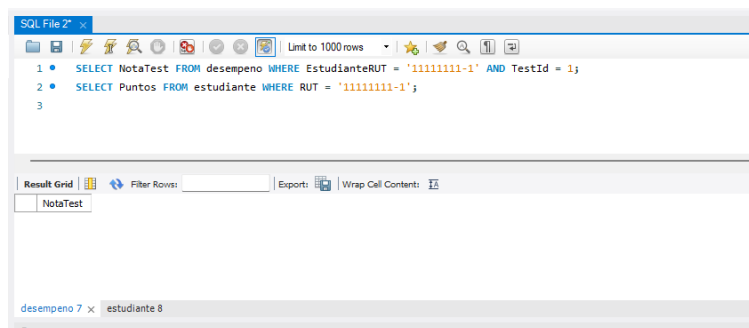


Figura B.10. Consulta previa de la tabla desempeno antes de finalizar la evaluación.

Fuente: elaboración propia.

La consulta permitió comprobar que todavía no existía una calificación registrada para el test evaluado.

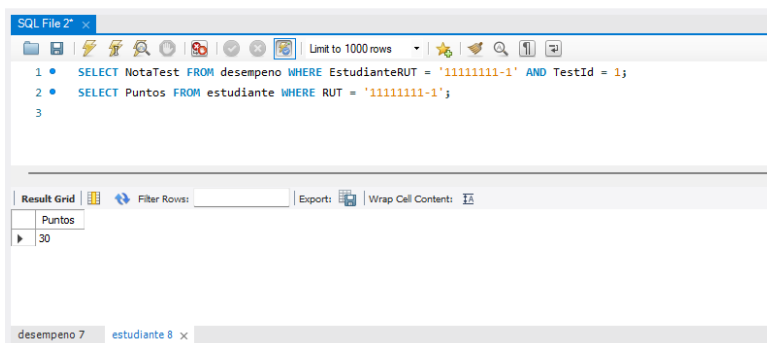


Figura B.11. Consulta del saldo de Glims anterior a la finalización de la evaluación.

Fuente: elaboración propia.

La evidencia muestra que el estudiante poseía inicialmente 30 Glims. Después de la evaluación, Aurora presentó un puntaje de 25 y una recompensa de 25 Glims, mientras que MySQL mostró un desempeño con NotaTest = 25 y un saldo final de 55 Glims.

Las Figuras B.10 y B.11 constituyen los estados anteriores utilizados en CP-11, CP-13 y CP-14. El Capítulo 4 señala que ambas consultas se conservan en los anexos para respaldar la comparación.

B.6 Condiciones del desbloqueo progresivo

El caso CP-17 requirió comprobar que la sesión académica anterior contara con el Pre-check, el desempeño y el Post-check necesarios para considerarla completada.

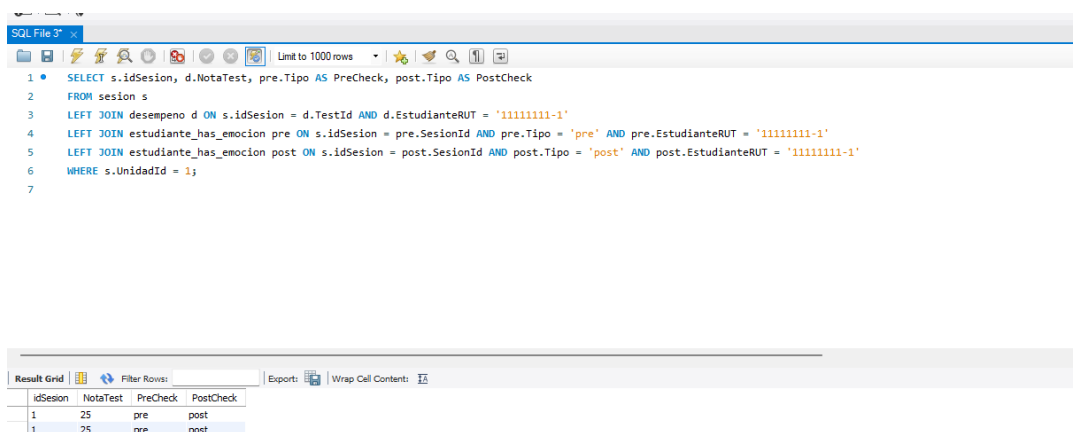


Figura B.12. Verificación en MySQL de las condiciones requeridas para el desbloqueo progresivo.

Fuente: elaboración propia.

La evidencia respalda técnicamente la existencia de las tres condiciones consideradas por la regla de desbloqueo. El resultado observable de esta operación se presenta en la Figura 4.14 del Capítulo 4, donde la Unidad 2 aparece habilitada.

B.7 Estado anterior a la interacción con el chatbot

Antes de enviar el mensaje utilizado en CP-18, se ejecutó una consulta sobre registro_ia para establecer el estado inicial de los datos.

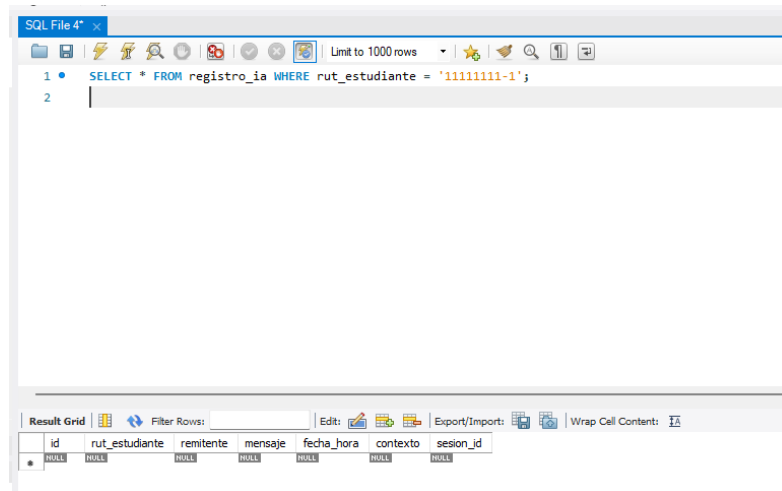


Figura B.13. Estado de la tabla registro_ia antes de la interacción con el chatbot de Aurora.

Fuente: elaboración propia.

Esta consulta funciona como referencia para la evidencia posterior presentada en el Capítulo 4. Debido a la indisponibilidad de la credencial utilizada para acceder a la API externa, el caso no validó una respuesta satisfactoria generada por OpenAI. Las evidencias disponibles demuestran únicamente la presentación de un error controlado y la persistencia de la interacción junto con el mensaje correspondiente.

Anexo C. INVENTARIO COMPLETO DE EVIDENCIAS

C.1 Objetivo y alcance

El presente anexo consolida la identificación y ubicación de las evidencias utilizadas durante la validación funcional del Back-End de Aurora. El inventario relaciona cada captura con el caso de prueba correspondiente, el resultado que respalda y su ubicación definitiva en el Capítulo 4 o en el Anexo B.

Una misma evidencia puede respaldar más de un caso cuando las comprobaciones forman parte de un único flujo funcional. Esto ocurre principalmente en los casos CP-11 a CP-14, relacionados con la finalización de la evaluación, el cálculo del puntaje, la persistencia del desempeño y la asignación de Glims.

En las evidencias de navegación correspondientes a CP-03 a CP-07 se conserva el identificador documental utilizado en la matriz de validación, debido a que sus nombres extendidos no fueron normalizados. Esta condición no afecta su correspondencia con las Figuras B.4 a B.8.

C.2 Inventario consolidado

Tabla C.1. Inventario completo de evidencias de la validación funcional del Back-End de Aurora.

N.º	Caso asociado	Archivo normalizado o identificador	Evidencia que respalda	Ubicación en la memoria
1	CP-00	CAP4_CP00_01_respuesta_endpoint_geografia.png	Respuesta satisfactoria del endpoint geográfico mediante FastAPI.	Figura B.1
2	CP-00	CAP4_CP00_02_verificacion_datos_mysql.png	Coincidencia entre los datos devueltos por el Back-End y los registros de MySQL.	Figura B.2
3	CP-01	CAP4_CP01_01_credenciales_invalidas.png	Rechazo de credenciales inválidas durante el inicio de sesión.	Figura 4.1
4	CP-01	CAP4_CP01_02_redireccion_rol.png	Inicio de sesión válido y redirección al panel del Estudiante.	Figura 4.2
5	CP-08	CAP4_CP08_01_menuprofesor.png	Inicio de sesión válido y acceso al menú del Profesor.	Figura 4.3

N. º	Caso asociado	Archivo normalizado o identificador	Evidencia que respalda	Ubicación en la memoria
6	CP-02 y CP-09	CAP4_CP02_01_modulo_estudio.png	Estado del módulo de estudio antes de cargar el nuevo material.	Figuras 4.4 y B.3
7	CP-03	CAP4_CP03_01	Visualización de las asignaturas disponibles para el Estudiante.	Figura B.4
8	CP-04	CAP4_CP04_01	Estado inicial de las unidades, con la Unidad 2 bloqueada.	Figura B.5
9	CP-05	CAP4_CP05_01	Visualización de las sesiones de la unidad seleccionada.	Figura B.6
10	CP-06	CAP4_CP06_01	Presentación del formulario emocional previo.	Figura B.7
11	CP-07	CAP4_CP07_01	Inicio de la evaluación interactiva.	Figura B.8
12	CP-09	CAP4_CP09_01_material_subido.png	Confirmación de la carga del material desde el perfil Profesor.	Figura 4.5
13	CP-09	CAP4_CP09_02_material_visible_estudiante.png	Disponibilidad posterior del material en el módulo del Estudiante.	Figura 4.6
14	CP-10	CAP4_CP10_01_db_antes.png	Estado de MySQL anterior al registro del Pre-check.	Figura B.9
15	CP-10	CAP4_CP10_02_submit_web.png	Selección y envío del estado emocional previo.	Figura 4.7
16	CP-10	CAP4_CP10_03_db_despues.png	Persistencia del Pre-check en estudiante_has_emocion.	Figura 4.8

N. º	Caso asociado	Archivo normalizado o identificador	Evidencia que respalda	Ubicación en la memoria
17	CP-11 y CP-13	CAP4_CP11_01_db_antes.png	Ausencia de un desempeño anterior para el test evaluado.	Figura B.10
18	CP-11 y CP-14	CAP4_CP11_01_db_antes2.png	Saldo inicial de 30 Glims.	Figura B.11
19	CP-11, CP-12 y CP-14	CAP4_CP11_02_submit_web.png	Resultado de 25 puntos y recompensa de 25 Glims mostrados por Aurora.	Figura 4.9
20	CP-11, CP-12 y CP-13	CAP4_CP11_03_db_despues.png	Persistencia de NotaTest = 25 en MySQL.	Figura 4.10
21	CP-11 y CP-14	CAP4_CP11_03_db_despues2.png	Incremento del saldo de Glims desde 30 hasta 55.	Figura 4.11
22	CP-15 y CP-16	CAP4_CP15_01_submit_web.png	Resultado de 25 puntos, recompensa de 0 Glims y formulario de Post-check.	Figura 4.12
23	CP-15	CAP4_CP15_02_db_despues.png	Conservación del saldo de 55 Glims después de repetir la evaluación.	Figura 4.13
24	CP-17	CAP4_CP17_01_db.png	Existencia del Pre-check, desempeño y Post-check requeridos para el desbloqueo.	Figura B.12
25	CP-17	CAP4_CP17_02_unidades_web.png	Habilitación de la Unidad 2 después de cumplir los requisitos.	Figura 4.14
26	CP-16	CAP4_CP16_03_db_despues3.png	Persistencia del Post-check con Tipo = post, Escala = 5 y contexto "Excelente".	Figura 4.15
27	CP-18	CAP4_CP18_01_db_antes.png	Estado de registro_ia anterior a la interacción con el chatbot.	Figura B.13

N. º	Caso asociado	Archivo normalizado o identificador	Evidencia que respalda	Ubicación en la memoria
28	CP-18	CAP4_CP18_02_web.png	Mensaje controlado ante la indisponibilidad del servicio externo.	Figura 4.16
29	CP-18	CAP4_CP18_03_db_despues.png	Persistencia del mensaje del estudiante y del error en registro_ia.	Figura 4.17
30	CP-19	CAP4_CP19_01_web.png	Rechazo de una ruta de descarga inválida mediante una respuesta JSON.	Figura 4.18
31	CP-20	CAP4_CP20_01_web.png	Respuesta controlada ante la solicitud de una sesión inexistente.	Figura 4.19

Fuente: elaboración propia.

C.3 Consideraciones de trazabilidad

Los casos CP-12, CP-13 y CP-14 no generan capturas independientes, debido a que fueron comprobados dentro del flujo ejecutado en CP-11. Sus resultados se respaldan mediante las evidencias anteriores y posteriores asociadas con el puntaje, el desempeño y el saldo de Glims.

Asimismo, CP-21 no se incorpora como caso de prueba ni requiere una evidencia independiente, debido a que corresponde a la conclusión general de la estrategia de validación.