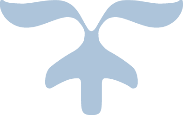


informe avance fase 2

grade

Versión 1.3





# Abstract

### Español

El proyecto GRADE (Generación y Registro Automatizado de Evaluaciones) desarrolla un sistema integral para digitalizar y centralizar el ciclo completo de evaluaciones académicas: creación de preguntas, generación de pruebas, calificación automática y registro de resultados. Durante la segunda fase se consolidaron los módulos de banco de preguntas y gestión de evaluaciones, alcanzando cerca del 70 % de avance del MVP. El sistema combina una arquitectura web + móvil basada en Next.js, Node.js y PostgreSQL, diseñada para garantizar escalabilidad, seguridad y trazabilidad de datos. La evaluación económica demuestra la viabilidad del proyecto con un horizonte de seis años, un VAN positivo ($71,9 millones CLP), TIR de 51,8 % y recuperación de la inversión en tres años. GRADE representa una solución tecnológica innovadora para optimizar los procesos de evaluación en entornos educativos.

### Inglés

The GRADE (Project for the Automated Generation and Recording of Evaluations) develops an integrated platform that digitalizes the entire academic assessment cycle—question bank management, exam generation, automatic grading, and result recording. By Phase 2, the project achieved approximately 70 % of its MVP completion, delivering fully functional modules for question bank and evaluation management. The system adopts a web and mobile architecture based on Next.js, Node.js, and PostgreSQL, focused on scalability, security, and data traceability. The economic evaluation confirms feasibility over a six-year horizon with a positive NPV ($71.9 million CLP), an IRR of 51.8 %, and investment recovery within three years. GRADE emerges as an innovative technological solution to enhance academic assessment processes in educational institutions.

# 

# Tabla de Contenidos

[**Abstract 2**](#_heading=h.6em23p8hzk76)

[Español 2](#_heading=h.d7mut79pikiz)

[Inglés 2](#_heading=h.gcxtg8cebp5x)

[**Tabla de Contenidos 3**](#_heading=h.4gcrgdekod3k)

[**1. Control de Cambios 4**](#_heading=h.8p8l1z263wn1)

[**2. Avances 5**](#_heading=h.apkgqxcup61)

[**3. Planificación inicial 6**](#_heading=h.nwuhnedqy0b3)

[**4. Recopilación y análisis 7**](#_heading=h.861ket5clpbw)

[**5. Definición de alcance 8**](#_heading=h.jskn02qrbihc)

[**6. Diseño 10**](#_heading=h.dbarplkrng09)

[**7. Arquitectura 10**](#_heading=h.t7ob41ai3m1s)

[**8. Base de datos 14**](#_heading=h.fgee3idok9d7)

[**9. Entidades y colecciones principales 15**](#_heading=h.9bnnerjwqb1z)

[**10. Relaciones 19**](#_heading=h.i86tc083cekz)

[**11. Características clave del modelo 22**](#_heading=h.dlc4nt8myw6a)

[**12. Casos de uso 25**](#_heading=h.1u8fgou462m5)

[**13. Avance etapa 28**](#_heading=h.dnzdpjoqdpr)

[**14. Desarrollo 31**](#_heading=h.z7if2kdwn4b1)

[**15. Evaluación Económica 34**](#_heading=h.nib3d45fh0ah)

[Inversión inicial 34](#_heading=h.dt40yofzh2ji)

[Flujo de caja proyectado 35](#_heading=h.m68rjynh2xeb)

[Indicadores Financieros 35](#_heading=h.wrt0ftgjmeb)

[Análisis de sensibilidad 35](#_heading=h.tj4egvyqfdrc)

[Conclusión económica 36](#_heading=h.4vooxytolu2o)

[**16. Estado general de avance 36**](#_heading=h.fa90r3h4p1i9)

[**17. Próximos pasos 38**](#_heading=h.enkp7o4w48fk)

[**18. Conclusiones 41**](#_heading=h.xwwsxl69jrae)

[Conclusion — María Fernanda Garay 41](#_heading=h.1ie15mdgc3hd)

[Conclusion — Maximiliano Toledo 41](#_heading=h.8cjo6h22kvy)

[Conclusion — Rodrigo Ulloa 42](#_heading=h.7ufza4wqy1rq)

[**19. Reflexión 43**](#_heading=h.p4jlhkp4yh25)

[**20. Anexos 44**](#_heading=h.tq75q2dfqza9)

# 1. Control de Cambios

* **Stack frontend:** Se incorporó Next.js al frontend React para renderizado del lado del servidor, optimizando desempeño y SEO del aplicativo web.
* **Tecnología móvil:** Se decidió cambiar la tecnología de la aplicación móvil de React Native a **Ionic + Angular**, buscando aprovechar la base Angular del ecosistema y capacidades híbridas; esta decisión difiere del plan inicial que consideraba React Native.
* **Alcance de usuarios:** Se confirmó que el sistema estará dirigido exclusivamente a **Docentes y Coordinadores**, excluyendo la interacción directa de estudiantes u otros roles en esta fase. Esto implica que no se desarrollarán interfaces para estudiantes ni apoderados en el MVP.
* **Contratos de API:** La definición formal de las interfaces **API** (especificaciones OpenAPI/Swagger) se pospuso para la **Fase 3** (fase final del proyecto). Originalmente se contemplaba tempranamente, pero se reprogramó su elaboración para el cierre del desarrollo, alineado con la integración final del sistema.
* **Pruebas y QA:** El plan de pruebas inicial se ha enfocado únicamente en el módulo de **Banco de Preguntas**, diseñando casos de prueba para este componente. La cobertura de pruebas de integración y usuario para otros módulos se difirió para etapas posteriores.
* **Módulo OMR retrasado:** El sub-sistema de **Ingesta Móvil (OMR)** presenta un desfase en su desarrollo: hasta ahora solo se ha completado el diseño del modelo de datos correspondiente, sin avance significativo en la implementación de lógica de backend ni interfaz móvil. Este cambio en la secuencia obligará a reforzar su desarrollo en la fase actual.

# 2. Avances

A la fecha, el proyecto **GRADE** alcanza aproximadamente un **70%** de avance general. Se han completado los módulos base y funcionalidades núcleo del sistema según lo planificado. En particular, ya están implementadas las capacidades fundamentales de gestión académica y banco de ítems. Por ejemplo, se concluyó la funcionalidad de **gestión de evaluaciones y banco centralizado de preguntas**, sentando las bases para armar evaluaciones. Asimismo, se desarrolló por completo el flujo de **creación y publicación de evaluaciones**: los docentes pueden seleccionar preguntas del banco, asignar puntajes y generar pruebas en formato PDF con código QR único. Esta segunda fase entregó un sistema ya capaz de emitir evaluaciones listas para aplicar.

En cuanto al **frontend web**, se tiene un **avance del 80%**, habiendo implementado la mayor parte de las vistas y componentes en React + Next.js. La interfaz de usuario cubre prácticamente todos los casos de uso principales para docentes y coordinadores, incluyendo pantallas para la gestión de preguntas, creación de pruebas y visualización de resultados. En el **backend**, se encuentran en definición los servicios y endpoints principales (autenticación, manejo de usuarios, registro de preguntas, creación de evaluaciones, etc.), respaldados por la base de datos unificada. Este núcleo backend soporta ya el flujo completo desde la elaboración de una prueba hasta su publicación, incluyendo el almacenamiento de las estructuras de evaluaciones y respuestas.

Por otro lado, el **módulo de Ingesta Móvil (OMR)** – encargado de la carga y procesamiento automático de respuestas escaneadas – está en una etapa temprana. Si bien se definió completamente su **modelo de datos** y estructura de almacenamiento, aún **no** se han implementado las funcionalidades de backend ni la aplicación móvil asociada para este módulo. En concreto, GRADE ya cuenta con el diseño de tablas y relaciones necesarias para capturar resultados vía OCR, priorizando trazabilidad de extremo a extremo e idempotencia en el procesamiento[[4]](https://github.com/Wanku-CL/wiki-wanku/blob/89467939d09200f1c22aba5a466290c8d955c175/products/grade/06-data-model/mobile-ingest/mer.md#L5-L9), pero el desarrollo de la lógica de reconocimiento y la interfaz móvil de captura se realizarán en la siguiente fase. De momento, la ingesta de respuestas solo está soportada en forma básica (por ejemplo, carga de archivos CSV), quedando pendiente la integración del flujo de captura fotográfica y escaneo desde dispositivos móviles.

# 3. Planificación inicial

El proyecto se estructuró desde un inicio con una **planificación por fases** bien definida, abarcando aproximadamente 9 a 10 semanas de trabajo intensivo[]](https://github.com/Wanku-CL/wiki-wanku/blob/89467939d09200f1c22aba5a466290c8d955c175/products/grade/01-roadmap/ROADMAP.md#L10-L17). Cada fase (F1–F5) fue delineada con objetivos concretos y entregables específicos, siguiendo un esquema incremental:

* **Fase 1 – Fundamentos:** Implementación de las bases del sistema: gestión de **cursos, alumnos, banco de preguntas y roles de usuario**. El objetivo fue dejar la plataforma lista para comenzar a armar evaluaciones.
* **Fase 2 – Evaluaciones:** Desarrollo de la funcionalidad de **creación, programación y aplicación de evaluaciones**, incluyendo la generación de pruebas en PDF con código QR para identificación. Esta fase apuntó a tener evaluaciones completas listas para ser aplicadas.
* **Fase 3 – Ingesta:** En esta etapa se planificó incorporar la **ingesta de respuestas** por múltiples canales: carga de archivos CSV, **aplicación móvil de escaneo (OCR)** y manejo de errores en el procesamiento. Es la fase que integra la captura automatizada de respuestas al flujo.
* **Fase 4 – Calificación:** Desarrollo del módulo de **calificación automática**, que toma las respuestas ingresadas y calcula las notas según criterios establecidos, generando resultados de forma automática.
* **Fase 5 – Resultados y Auditoría:** Implementación de las funcionalidades de **publicación de resultados, reportes finales y auditoría** del proceso (trazabilidad de acciones, registro inmutable de calificaciones). Esta es la fase final para cerrar el ciclo completo con controles y validaciones finales.

Adicional a las fases, la planificación inicial contempló **hitos clave** con fechas objetivas: inicio del proyecto, finalización de cada fase en las fechas indicadas, y cierre con el MVP completo. Se incluyó un período de pruebas de aceptación al final (UAT) previo al lanzamiento. La estrategia supuso trabajo **paralelo en dos frentes**: el equipo web/backend y el equipo móvil/OMR avanzando simultáneamente en sus respectivos componentes.

En la planificación original también se previó la preparación temprana de ciertos artefactos técnicos. Por ejemplo, durante la etapa de diseño (previa al desarrollo) se buscaba elaborar un **documento de especificación de APIs** del sistema (contratos de servicio) para guiar la integración futura. Sin embargo, como se mencionó, la formalización completa de estos contratos de API se reprogramó para el final del desarrollo (fase 3) debido a la necesidad de priorizar funcionalidades internas en las primeras fases.

# 4. Recopilación y análisis

En las etapas iniciales del proyecto se llevó a cabo un proceso exhaustivo de **descubrimiento, recopilación de requisitos y análisis** del contexto. Este trabajo incluyó reuniones con stakeholders (docentes, coordinadores y expertos pedagógicos), revisión de procesos actuales y definición de la **visión** y los objetivos estratégicos de GRADE. Fruto de este análisis temprano, se identificaron las **necesidades funcionales clave** que el sistema debe satisfacer y las expectativas operativas a largo plazo. Por ejemplo, se estableció que GRADE “debe ser el único punto para crear, publicar, aplicar y registrar evaluaciones” centralizando integralmente el ciclo de pruebas, evitando herramientas aisladas. Asimismo, se determinó la importancia de contar con un **banco de preguntas versionado y trazable** que permita reutilizar ítems con metadatos pedagógicos, manteniendo control de versiones en todo momento. Estas necesidades (N1, N2, etc.) fueron mapeadas directamente a objetivos estratégicos para asegurar alineamiento con la propuesta de valor.

Del lado de los requisitos no funcionales y expectativas, el análisis resaltó principios de **escalabilidad, seguridad y usabilidad** que debían guiar la arquitectura. Se documentaron expectativas operativas como garantizar **rendimiento bajo picos de carga**, cumplimiento de estándares de **accesibilidad** (WCAG 2.1 AA) e integridad de datos académicos. También se identificaron riesgos potenciales (ej. baja adopción por parte de docentes, precisión del OCR por debajo de lo esperado) junto con estrategias de **mitigación proactiva** para cada uno.

Durante esta fase de recopilación se definió claramente el **público objetivo** y los actores del sistema. Se concluyó que los usuarios directos del MVP serían únicamente docentes (usuarios finales que generan y califican evaluaciones) y coordinadores académicos, excluyendo a estudiantes y otros roles de la interacción con la plataforma en esta etapa. Esta decisión de alcance respondió al análisis de necesidades: se priorizó resolver el problema de la gestión de evaluaciones para quienes las crean y administran, asumiendo que los estudiantes seguirían utilizando medios tradicionales (o plataformas integradas externamente) para rendir las pruebas.

# 5. Definición de alcance

La **definición de alcance** del proyecto GRADE quedó establecida en términos de las funcionalidades incluidas en el MVP y aquellas excluidas (fuera de alcance). Para el MVP se consideraron las siguientes capacidades funcionales principales:

* Un **banco centralizado de preguntas** autocorregibles, con posibilidad de crear, editar y versionar ítems de tipo selección múltiple o verdadero/falso, incluyendo metadatos como unidad, dificultad y resultado de aprendizaje.
* **Generación de evaluaciones** (pruebas) a partir de esas preguntas, permitiendo al usuario armar exámenes seleccionando ítems del banco, definir puntajes o ponderaciones, y exportar la prueba a formato **PDF** (con código QR para su identificación) o a un formato digital seguro.
* **Calificación automática** de evaluaciones: capacidades de ingesta de respuestas (vía carga de archivo CSV, formulario web, o escaneo OCR de las hojas de respuesta) y cálculo de la nota o puntaje total según criterios predefinidos, registrando todas las respuestas de forma estructurada.
* **Reportes básicos de resultados**, que permitan al docente visualizar estadísticas elementales de la evaluación (por ejemplo, promedio del curso, distribución de notas, preguntas con mayor índice de error, etc.) una vez calificada la prueba.
* **Gestión de usuarios y cuentas** con distintos roles (Docente, Coordinador, Administrador) incluyendo registro/login y control de acceso a las funciones según el rol.
* **Seguridad de datos** y auditoría: manejo seguro de la información (cifrado en tránsito y en reposo), bitácora de acciones críticas (ej. publicación de evaluaciones, modificaciones) y aplicación de políticas de roles de manera que solo usuarios autorizados accedan a cada sección.

Asimismo, en la definición de alcance se documentó explícitamente qué elementos **no** formarían parte del MVP (**fuera de alcance**). Entre ellos se incluyó la ausencia de interfaces o funcionalidades destinadas a **estudiantes** o apoderados (es decir, los estudiantes no interactúan directamente con GRADE en esta etapa, sino a través del docente o de otras plataformas integradas) . También quedaron fuera de alcance inicial las **integraciones con sistemas externos** complejos (por ejemplo, integración con sistemas de información estudiantil – SIS, o Single Sign-On institucional), las funcionalidades de **analítica avanzada** (dashboards complejos, generación automática de ítems con IA, etc.) y cualquier mecanismo de pagos en línea o suscripciones. Estos aspectos se pospusieron para versiones futuras una vez validado el MVP, de modo de mantener acotado el alcance a un conjunto manejable de características fundamentales.

# 6. Diseño

Durante la etapa de **diseño** se elaboró la especificación funcional detallada del sistema y la experiencia de usuario pretendida. Se produjeron artefactos de diseño que incluyen: casos de uso, diagramas de secuencia, diagramas de procesos,, todo con el fin de asegurar que la solución cumpla con los requisitos recopilados de manera intuitiva y eficiente.

En términos de **diseño funcional**, se definieron **escenarios de usuario** y **casos de uso** describiendo paso a paso cómo un docente o coordinador interactuaría con la plataforma para lograr sus objetivos (por ejemplo, crear una pregunta, generar una evaluación, calificar respuestas). Estos casos de uso guiaron la definición de la interfaz y de las operaciones de backend necesarias. Cada caso de uso se relacionó además con las necesidades identificadas en el análisis para mantener la trazabilidad (matriz de trazabilidad de requisitos a casos de uso).

En cuanto al **diseño técnico**, antes de iniciar la codificación se produjo un bosquejo de la arquitectura (ver sección 7) y del **modelo de datos** (ver sección 8 y 9) asegurando que ambos soportaran los casos de uso definidos. Se prepararon diagramas de componentes y algunos diagramas de secuencia para visualizar la interacción entre frontend, backend y servicios externos (como el servicio OCR) en casos clave como la calificación automática. También se decidió en esta etapa la distribución de responsabilidades entre microservicios: se aisló el componente de OCR/ingesta móvil como servicio independiente para mantener desacoplado el procesamiento pesado de imágenes del resto del sistema.

# 7. Arquitectura

La **arquitectura** de GRADE se diseñó siguiendo principios de modularidad, escalabilidad y seguridad, acorde a las necesidades identificadas. En un alto nivel, el sistema adopta una arquitectura de varios niveles (multitier) dividida en: **frontends de cliente** (aplicación web y aplicación móvil), un **backend** central de servicios API, y una **capa de datos** consistente de bases de datos y almacenamiento.

* **Arquitectura lógica:** El backend se estructura en una serie de **módulos o microservicios** que corresponden a los principales subsistemas: por un lado el módulo core de **gestión académica** (que incluye usuarios, cursos, banco de preguntas, evaluaciones, resultados), y por otro el módulo especializado de **ingesta y procesamiento OCR** para calificar automáticamente. Estos componentes se comunican a través de interfaces bien definidas (principalmente APIs REST internas). Esta separación garantiza escalabilidad y aislación: por ejemplo, el servicio de OCR puede escalarse independientemente en caso de alta demanda durante períodos de exámenes.
* **Frontend Web:** Implementado con React (biblioteca JavaScript), apoyado por Next.js para el renderizado del lado del servidor, lo que mejora los tiempos de carga y la optimización SEO. El frontend web constituye la aplicación utilizada por docentes y coordinadores desde navegadores de escritorio o móviles. Emplea React-Bootstrap para la construcción de una interfaz coherente, responsiva y estéticamente uniforme, basada en los componentes nativos de Bootstrap integrados con React. La aplicación web consume las APIs del backend para todas las operaciones —consulta del banco de preguntas, creación de evaluaciones, envío de respuestas, entre otras—, funcionando esencialmente como un cliente ligero que comunica y presenta los servicios del sistema.
* **Aplicación Móvil:** Concebida para dispositivos iOS/Android usando el framework híbrido **Ionic + Angular** (tras la revisión tecnológica mencionada). Esta app móvil se centra en la función de **captura de respuestas** mediante la cámara (fotos o escaneos de hojas de respuesta con código QR). Operará tanto **offline** (capturando y almacenando temporalmente imágenes si no hay conectividad) como online (sincronizando con el backend cuando esté disponible). Dado que Ionic permite desplegar la misma base de código en múltiples plataformas, se facilita cubrir la mayoría de dispositivos de docentes sin desarrollos nativos separados. La app se comunica con el backend mediante API REST para enviar los lotes de páginas escaneadas y recibir el procesamiento.
* **Backend y API:** El núcleo backend está desarrollado sobre **Node.js** con el framework **Express**, siguiendo una arquitectura RESTful de servicios web. Expone un conjunto de **APIs JSON** seguras para las distintas funcionalidades (ej.: endpoints para crear/editar preguntas, generar evaluaciones, cargar respuestas, obtener reportes). Por motivos de seguridad y control, estas APIs requieren **autenticación JWT** para los clientes internos (frontend web y móvil). Además, se ha previsto un mecanismo de autenticación OAuth2 en caso de integraciones con aplicaciones de terceros a futuro – es decir, el sistema podría emitir tokens con scopes limitados para plataformas educacionales externas que quieran integrarse (por ejemplo, un LMS que consuma los resultados).
* **Gestión de datos:** Como capa de datos principal se utiliza una base de datos **relacional PostgreSQL 14+** que almacena la información transaccional del sistema. El modelo de datos está normalizado (3FN) cubriendo las entidades descritas en la sección 9.
* **Infraestructura y despliegue:** La infraestructura de GRADE está diseñada bajo un enfoque ligero, modular y de bajo costo, aprovechando servicios cloud serverless y contenedores escalables que permiten asegurar disponibilidad y rendimiento sin requerir infraestructura compleja.

El sistema se apoya en tres pilares tecnológicos principales:

Base de datos: se utiliza Neon, una solución PostgreSQL serverless que ofrece autoescalado dinámico de CPU y almacenamiento, con un plan gratuito en etapa piloto y costos progresivos según uso. Esta tecnología permite disponer de un entorno persistente, confiable y económico para las operaciones académicas de GRADE, con recuperación automática y soporte para múltiples conexiones concurrentes.

Frontend y backend: ambos se despliegan mediante Cloudflare Workers, una plataforma de cómputo en el borde (edge computing) que garantiza baja latencia y tiempos de respuesta rápidos desde cualquier ubicación. Esta arquitectura permite que las funciones de API y los recursos de interfaz estén distribuidos globalmente, mejorando la experiencia de docentes y estudiantes. Además, el plan estándar de Cloudflare incluye monitoreo, protección DDoS y certificados HTTPS automáticos end-to-end, eliminando la necesidad de servidores dedicados tradicionales.

Contenedores y servicios auxiliares: el microservicio de OCR/OMR, encargado del reconocimiento óptico de marcas, se despliega mediante Fly.io, utilizando contenedores ligeros basados en Docker. Esto posibilita el escalamiento horizontal de instancias de cómputo de acuerdo con la carga de procesamiento (por ejemplo, durante semanas de evaluación intensiva). El modelo de pago por uso de Fly.io permite mantener una infraestructura operativa 24/7 a un costo mensual estimado de 15 USD, con capacidad para ejecutar varias instancias simultáneamente.

En términos de seguridad y calidad, GRADE sigue los lineamientos de la OWASP Foundation, implementando validaciones de entrada, manejo seguro de sesiones con JWT y comunicación cifrada con TLS/HTTPS. La infraestructura considera además el uso de herramientas de monitoreo y registro para trazabilidad de eventos, lo que permitirá auditar el comportamiento del sistema y detectar incidentes de manera temprana.

Finalmente, el proyecto contempla la adopción de prácticas de Integración y Despliegue Continuo (CI/CD) mediante GitHub Actions, que automatiza la construcción, pruebas y publicación de los distintos componentes. Gracias a esta configuración, el despliegue de GRADE combina eficiencia operativa, seguridad y sostenibilidad económica, pudiendo funcionar en entornos piloto con un costo aproximado de 20–25 USD mensuales, y escalar progresivamente conforme aumente el uso

# 8. Base de datos

El sistema GRADE emplea como base de datos principal un **sistema relacional PostgreSQL** (versión 14 o superior), seleccionado por su robustez, soporte de transacciones y capacidades avanzadas adecuadas para garantizar integridad referencial en datos académicos. Toda la información crítica (usuarios, preguntas, evaluaciones, respuestas, etc.) se almacena en esquemas relacionales normalizados en esta base de datos. Se definió un **modelo entidad-relación unificado** que abarca los tres módulos principales (Banco de Preguntas, Gestión de Evaluaciones e Ingesta Móvil), de modo que las relaciones entre entidades queden explícitas y controladas por llaves foráneas.

La organización de la base de datos se realizó dividiendo lógicamente las tablas por **contexto de negocio**, aunque compartan la misma BD física. Por ejemplo, las tablas del banco de preguntas (preguntas, opciones, temas, etc.) se agrupan bajo el módulo "QB", las de evaluaciones y cursos bajo "GE", y las de ingesta OCR bajo "IM"[[27]](https://github.com/Wanku-CL/wiki-wanku/blob/89467939d09200f1c22aba5a466290c8d955c175/products/grade/06-data-model/data-dictionary.md#L14-L21). No obstante, existen vínculos inter-módulo (por ejemplo, una evaluación está ligada tanto a preguntas del banco como a resultados de ingesta) que también se plasman en el modelo.

Para cada tabla se definieron cuidadosamente tipos de datos apropiados, restricciones de **unicidad** y **checks** para asegurar la calidad de los datos. Por convención, cada tabla cuenta con una clave primaria numérica (identificador serial) y llaves foráneas hacia tablas relacionadas, siguiendo nomenclatura consistente (sufijos \_fk para foreign keys). Se aprovechó la capacidad de PostgreSQL para imponer **restricciones a nivel de base** en apoyo a las reglas de negocio: por ejemplo, unicidad compuesta para evitar inscripciones duplicadas de un mismo estudiante en un curso, o checks en campos enumerados (estados válidos de una evaluación, tipo de pregunta, etc.) para evitar valores fuera de catálogo. Esto agrega una capa de seguridad de datos adicional a las validaciones de aplicación.

La base de datos también incorpora campos de **auditoría temporal y lógica** en varias tablas clave: campos como created\_at, updated\_at registran fechas de creación y modificación; flags lógicos como active indican vigencia de ciertos registros para habilitar un esquema de **borrado suave (soft delete)** en lugar de eliminación física de datos sensibles. Por ejemplo, en la tabla de preguntas existe un campo booleano active para marcar si una pregunta está activa o fue desactivada (posiblemente reemplazada por una versión nueva). Este enfoque permite conservar histórico y trazabilidad de cambios, algo fundamental en entornos académicos donde no se desea perder el rastro de evaluaciones o preguntas aunque se dejen de usar.

Dado el requerimiento de procesamiento de imágenes para la calificación automática, se consideró también el almacenamiento de ciertos datos de gran volumen o no estructurados fuera de las tablas principales. Para ello, se prevé el uso de un repositorio de blobs (por ejemplo **AWS S3**) para guardar las **imágenes escaneadas** de las hojas de respuesta, almacenando en la base de datos solo referencias (URLs o URIs) a dichos objetos. Esto evita sobrecargar la base de datos SQL con datos binarios pesados. Los resultados del procesamiento OCR (detecciones de marcas, etc.) sí se almacenan en tablas especializadas dentro del esquema IM, dado que son datos estructurados necesarios para auditoría y recalculabilidad.

Por último, se ha diseñado una política de **backup y recuperación** acorde con la criticidad de los datos: respaldos periódicos de la base PostgreSQL, potencialmente utilizando herramientas nativas (pg\_dump/Point-In-Time Recovery) y replicación en caliente a un servidor de respaldo. Esto garantiza la continuidad del servicio ante cualquier falla y cumple con lineamientos de continuidad operacional definidos para el proyecto.

# 9. Entidades y colecciones principales

A continuación, se enumeran las **entidades (tablas)** más importantes del modelo de datos de GRADE, organizadas por módulo principal del sistema:

* **Banco de Preguntas (QB):** Contiene las entidades relacionadas a la definición y clasificación de preguntas. Incluye:
* Subject (Asignatura): representa una materia o ramo; agrupa unidades temáticas.
* Unit (Unidad): subdivisión de una asignatura; agrupa tópicos.
* Topic (Tópico): tema específico dentro de una unidad; se asocia a las preguntas.
* Question (Pregunta): ítem de evaluación autocorregible, con atributos como enunciado, tipo (VF, selección múltiple), dificultad y resultado de aprendizaje asociado. Soporta **versionamiento**, de modo que una pregunta puede tener múltiples versiones a lo largo del tiempo (campo de versión y referencia a pregunta original).
* QuestionOption (Alternativa de pregunta): cada posible respuesta/opción de una pregunta (ej. alternativas A, B, C, D para una de selección múltiple), indicando cuál es la correcta. Se vincula a una pregunta y almacena texto de la opción, orden, marca de correcta, puntaje parcial, etc.
* Entidades auxiliares como QuestionType (catálogo de tipos de pregunta, ej. true/false, múltiple única, múltiple múltiple), Difficulty (catálogo de niveles de dificultad, e.g. fácil, media, difícil) y Outcome (resultado de aprendizaje medible, para trazar cada pregunta a objetivos pedagógicos).
* **Gestión de Evaluaciones (GE):** Abarca las entidades para estructurar evaluaciones (pruebas) y registrar su aplicación a estudiantes:
* Course (Curso/Sección): representa un curso académico o sección, con un nombre, código y un docente responsable asignado[[34]](https://github.com/Wanku-CL/wiki-wanku/blob/89467939d09200f1c22aba5a466290c8d955c175/products/grade/06-data-model/data-dictionary.md#L132-L140).
* Student (Estudiante): catálogo de estudiantes (nombre, identificador institucional, etc.) necesario para asociar resultados. Aunque los estudiantes no interactúan con el sistema, se registran para ligar sus resultados.
* CourseStudent (Matrícula): tabla intermedia que relaciona estudiantes con cursos (inscripciones), indicando qué estudiantes pertenecen a cada curso.
* Evaluation (Evaluación): corresponde a una prueba o examen creado en el sistema. Contiene título, fecha programada, duración, escala de notas definida (ej. 1-7), estado (borrador, publicada, aplicada, calificada, etc.), referencia al curso al que pertenece y al docente/coordinador que la creó.
* EvaluationQuestion (Pregunta en evaluación): representa cada ítem/pregunta incluida en una evaluación particular. Es una **instancia/snapshot** de una Question del banco en el contexto de la evaluación, con su puntaje asignado y posición en la prueba. Esta entidad permite congelar la versión de la pregunta y sus opciones en el momento de generar la prueba, garantizando que cambios futuros en el banco no afecten evaluaciones ya publicadas.
* EvaluationOption (Opción en evaluación): almacena las alternativas de respuesta asociadas a cada EvaluationQuestion. Es el snapshot de QuestionOption para esa evaluación, copiando el texto de la opción tal como estaba al momento de la prueba. Se utiliza para comparar con las respuestas del estudiante posteriormente.
* StudentEvaluation (Rendición de Estudiante): registra la instancia de un estudiante presentándose a una evaluación. En otras palabras, indica que un determinado estudiante (de los inscritos en el curso) tiene una entrega/participación en la evaluación correspondiente. Contiene datos como fecha de rendición (si se rindió en distinto horario), y se relaciona con múltiples respuestas.
* StudentAnswer (Respuesta de Estudiante): detalle de cada respuesta que un estudiante entregó para una pregunta dada de la evaluación. Se vincula tanto a la StudentEvaluation (quién responde) como a la EvaluationQuestion (qué pregunta respondió). Para preguntas de selección, puede estar ligada además a una o varias StudentAnswerOption.
* StudentAnswerOption (Opción seleccionada): en caso de preguntas de selección múltiple, detalla cuál(es) alternativa(s) escogió el estudiante. Se relaciona con la EvaluationOption correspondiente (la alternativa de esa pregunta en la evaluación) para indicar, por ejemplo, que el estudiante marcó la opción B.
* **Ingesta Móvil / Calificación (IM):** Incluye entidades para el proceso de captura y reconocimiento automático de respuestas mediante la app móvil y servicios OCR:
* IngestDevice (Dispositivo de Ingesta): registra los dispositivos móviles autorizados (p. ej., el teléfono de un docente) para realizar escaneos de evaluaciones. Almacena info del dispositivo, su registro por un usuario y estado (activo/inactivo).
* IngestBatch (Lote de Ingesta): representa una **sesión de captura** de respuestas, típicamente asociada a una evaluación y curso específicos. Un lote agrupa un conjunto de páginas escaneadas en un periodo de tiempo determinado (por ejemplo, todas las copias de una prueba escaneadas al finalizar la clase). Registra qué evaluación y curso corresponden, qué dispositivo realizó la captura, horarios de inicio/cierre y estado (abierto, procesando, cerrado).
* ScannedPage (Página Escaneada): cada imagen o foto de una hoja de respuestas capturada se almacena como un registro en esta tabla[[31]](https://github.com/Wanku-CL/wiki-wanku/blob/89467939d09200f1c22aba5a466290c8d955c175/products/grade/06-data-model/mobile-ingest/mer.md#L67-L75). Incluye metadatos como referencia al lote y dispositivo, la ruta/URI del archivo de imagen (almacenado externamente), un hash SHA-256 de la imagen para evitar duplicados (idempotencia), marca temporal de captura, dimensiones (dpi, tamaño) y estado de procesamiento (en cola, decodificada, error).
* PageQR (QR decodificado): resultado de leer el código QR en una página escaneada. Del QR se obtiene la identificación de la evaluación correspondiente (y potencialmente el identificador del estudiante si el código lo incluye). Esta tabla vincula la página con la evaluación (vía un código único) y guarda el contenido decodificado del QR.
* PageDetection (Detección de Página): almacena información del proceso de detección general en la página, como marcas de alineación encontradas, calibración de la imagen, y métricas de calidad. Indica si la página pudo procesarse correctamente (ej. nivel de desenfoque, skew, etc.).
* BubbleDetection (Detección de Burbuja): representa cada marca detectada en la hoja (cada burbuja o casilla que el estudiante pudo rellenar). Por cada pregunta en la hoja habrá una o varias burbujas (según tipo de pregunta) y esta entidad guarda la posición de la burbuja, nivel de rellenado o confianza del OCR, etc.
* RecognitionMapping (Mapeo de Reconocimiento): es una entidad clave que **mapea** cada burbuja detectada (BubbleDetection) con la opción de la evaluación correspondiente (EvaluationOption). En esencia, conecta lo capturado en la imagen con la pregunta y opción reales que representa en el sistema. Esto permite luego registrar automáticamente la respuesta del estudiante: por ejemplo, mapear "burbuja en fila 10 columna B" a "Pregunta 10, opción B" de la evaluación X.
* IngestResult (Resultado de Ingesta): una vez procesadas todas las páginas de un lote, esta entidad consolida los resultados por estudiante y evaluación. Se asocia tanto al estudiante (vía StudentEvaluation) como a la evaluación, indicando cuántas respuestas fueron reconocidas correctamente, cuántas omitidas, etc., y sirviendo de puente para generar las notas finales.
* ProcessingJob y ProcessingLog (Tarea de Proceso y Log): tablas para seguimiento técnico del pipeline de OCR. Cada lote procesado genera uno o varios jobs de procesamiento asíncrono registrados en ProcessingJob (con timestamps de inicio/fin, estado, etc.), y cualquier evento o error durante la ejecución se anota en ProcessingLog. Esto facilita la **auditoría técnica** y debugging en caso de problemas, complementando la auditoría funcional.

Estas son las principales entidades que componen el modelo de datos de GRADE. En total, abarcan desde la definición estática de contenidos (preguntas, cursos) hasta el registro dinámico de transacciones (resultados, capturas). Las relaciones entre ellas se describen en la siguiente sección.

# 10. Relaciones

El modelo entidad-relación de GRADE establece **vínculos lógicos** entre las entidades descritas, reflejando las reglas de negocio del dominio de evaluaciones. A continuación se destacan algunas de las relaciones más importantes:

* En el **Banco de Preguntas**, las entidades de clasificación forman una jerarquía: cada Subject (asignatura) puede tener múltiples Unit (unidades) asociadas, y cada unidad a su vez múltiples Topic (tópicos) relacionados. Las Question (preguntas) pertenecen a un tópico específico (mediante topic\_fk), heredando así indirectamente la unidad y asignatura, lo que permite estructurar el banco por materia y tema. Además, cada pregunta está ligada a su autor mediante user\_fk apuntando a la tabla de usuarios (docente o administrador que creó la pregunta). Otras relaciones en este módulo incluyen: preguntas a sus opciones (Question tiene muchas QuestionOption asociadas), preguntas a nivel de dificultad (difficulty\_fk) y a resultado de aprendizaje (outcome\_fk) cuando corresponda. Estas relaciones aseguran, por ejemplo, que no existan opciones huérfanas sin pregunta padre, o que una pregunta siempre tenga un tópico válido definido.
* En el módulo de **Evaluaciones**, un Course (curso) puede tener varios estudiantes inscritos (relación uno a muchos a través de CourseStudent hacia Student) y puede tener también varias evaluaciones programadas. Cada Evaluation pertenece a un curso determinado (course\_fk) y es creada por un usuario docente/coordinador (user\_fk). Existe una relación de composición entre Course y Evaluation: un curso *incluye* muchas evaluaciones a lo largo del periodo. A su vez, cada evaluación *contiene* múltiples preguntas específicas (EvaluationQuestion): por cada pregunta del banco seleccionada, se crea un registro en EvaluationQuestion vinculado a la evaluación y apuntando a la pregunta original del banco (relación de muchos a uno hacia Question). Las EvaluationOption están relacionadas tanto a su EvaluationQuestion (cada pregunta en la prueba tiene sus opciones congeladas) como a la QuestionOption original del banco de preguntas de la cual derivan. Esto permite trazar cada alternativa de la evaluación de regreso a su fuente en el banco.
* En cuanto a la **participación de estudiantes**: un estudiante puede estar inscrito en múltiples cursos y cada curso tiene varios estudiantes (Student a CourseStudent a Course). Para cada evaluación publicada, los estudiantes inscritos en el curso correspondiente tienen la posibilidad de rendirla, lo que se refleja en la relación de Evaluation con StudentEvaluation: una evaluación *admite* muchos registros de estudiante-evaluación (uno por alumno que rinde). Cada StudentEvaluation enlaza a un estudiante y a una evaluación particular, y a su vez *agrupa* muchas respuestas (StudentAnswer) que ese estudiante dio en dicha prueba. Cada respuesta de estudiante está asociada a la pregunta específica de la evaluación que responde (vía evaluation\_question\_fk en StudentAnswer), estableciendo así qué ítem fue contestado y permitiendo comparar con la respuesta correcta. Para respuestas de selección, StudentAnswer se relaciona con las opciones elegidas a través de StudentAnswerOption, que vincula con la opción de la evaluación seleccionada por el estudiante. En conjunto, estas relaciones garantizan que para cada estudiante podamos reconstruir exactamente qué respondió en cada pregunta y comparar contra la clave (opción correcta) almacenada en la EvaluationOption correspondiente.
* En el módulo de **Ingesta Móvil (OMR)**, las relaciones manejan la agrupación y mapeo de las capturas de imágenes con las entidades anteriores de evaluaciones y respuestas. Un IngestBatch (lote) está asociado a una única evaluación (campo evaluation\_fk) y a un curso (para validar que el lote corresponde a la evaluación del curso correcto). A su vez un lote agrupa muchas ScannedPage (páginas escaneadas) – relación uno a muchos, donde cada página tiene referencia al lote al que pertenece. Cada página escaneada es capturada por un dispositivo (device\_fk hacia IngestDevice) indicando qué dispositivo realizó la carga. Una vez que la imagen se procesa, se generan detecciones: una ScannedPage puede tener uno o más PageQR decodificados (en la práctica debería ser uno principal por página) y múltiples detecciones (PageDetection y BubbleDetection) asociadas. Cada BubbleDetection (posible marca) pertenece a una detección de página y se mapea mediante RecognitionMapping a una opción de evaluación específica – es decir, se establece la relación entre una marca en la imagen y la respuesta que representa (ejemplo: *burbuja detectada X -> opción B de la pregunta 5 en la Evaluación Y*). Varios mapeos pueden provenir de la misma página (cada burbuja relevante encontrada produce un mapping). Finalmente, todas las páginas de un lote, al completarse el procesamiento, producen resultados consolidados: un IngestResult por evaluación (y potencialmente por estudiante) que se asocia tanto al lote/páginas como a la Evaluation original. La evaluación mantiene así una relación con los lotes de ingesta (evaluation tiene muchos ingest\_batch asociados como intentos de carga de respuestas) y con los resultados procesados (evaluation tiene muchos ingest\_result). Esto cierra el ciclo vinculando directamente lo capturado por la app móvil con los objetos centrales de evaluación y respuestas en el sistema.

# 11. Características clave del modelo

El modelo de datos de GRADE presenta varias **características destacadas** que lo hacen especialmente apto para el dominio de gestión de evaluaciones automatizadas:

* **Normalización con modularidad:** El esquema fue diseñado en **3ª Forma Normal (3FN)** para eliminar redundancias y asegurar consistencia. Cada concepto fundamental (pregunta, curso, evaluación, respuesta, etc.) está representado por su propia entidad, lo que evita anomalías de actualización. A la vez, se mantuvo una separación modular de subdominios (banco de preguntas, evaluaciones, ingesta) dentro de un mismo esquema unificado, lo cual previene duplicación de datos entre módulos. Por ejemplo, la entidad Question es única y compartida, de modo que tanto el banco como las evaluaciones referencian la misma fuente de verdad para el contenido de la pregunta, mientras que la entidad EvaluationQuestion actúa como snapshot para aislamiento sin duplicar la lógica de preguntas en sí.
* **Mecanismo de *snapshot* y versionamiento:** Una de las capacidades más importantes del modelo es cómo maneja los cambios en el banco de preguntas a lo largo del tiempo y su efecto en evaluaciones ya creadas. Se implementó un **sistema de snapshot**: al generar una evaluación, cada pregunta incluida crea un registro EvaluationQuestion y sus correspondientes EvaluationOption que capturan el estado exacto de esa pregunta (enunciado, alternativas correctas) en el momento de la prueba. Esto significa que si posteriormente la pregunta es editada o versionada en el banco (por ejemplo, para corregir una opción o reformular texto), las evaluaciones pasadas no se ven afectadas – siempre apuntarán a la versión congelada que tuvieron al publicarse. Además, las preguntas en el banco cuentan con un campo de **versión** incremental y una auto-referencia a la pregunta original raíz en caso de versiones sucesivas. De esta manera, se mantiene un historial de versiones por pregunta, permitiendo saber si, por ejemplo, una pregunta fue actualizada después de cierto examen. Este enfoque de versionamiento explícito y snapshot garantiza **trazabilidad** y coherencia histórica en un contexto donde la validez de las evaluaciones no debe alterarse retroactivamente.
* **Integridad referencial y reglas de negocio en la BD:** El modelo aprovecha ampliamente las capacidades del SGBD para mantener integridad. Se definieron **llaves foráneas** para todas las relaciones, con políticas de borrado en cascada o restricción según corresponda (p.ej., al eliminar una pregunta del banco – operación restringida solo a admin – se elimina en cascada sus opciones, pero no se elimina ninguna pregunta ya usada en evaluaciones publicadas debido a la separación por snapshot). También se impusieron **restricciones de unicidad y checks** para encapsular reglas de negocio: por ejemplo, una misma pregunta no puede estar repetida dos veces en una evaluación (UNIQUE(evaluation\_fk, question\_fk) en EvaluationQuestion garantiza eso); un estudiante no puede inscribirse dos veces en el mismo curso (UNIQUE(course\_fk, student\_fk) en CourseStudent); una alternativa de pregunta no puede tener texto duplicado dentro de la misma pregunta (UNIQUE(question\_fk, text) en QuestionOption); el estado de una evaluación solo puede ser uno de los predefinidos válidos (check en Evaluation.state) etc. Estas reglas, implementadas a nivel de base de datos, refuerzan la calidad de datos y evitan situaciones inválidas antes incluso de llegar a la lógica de la aplicación.
* **Auditoría y trazabilidad de cambios:** El modelo incorpora un esquema estándar de **auditoría "soft"** para las entidades de negocio principales. Campos de timestamp (created\_at, updated\_at) permiten rastrear cuándo se crearon o modificaron registros, y campos booleanos como active señalan eliminaciones lógicas en lugar de físicas. Además, en el módulo de ingesta OCR se sigue un patrón de **auditoría técnica detallada**: las tablas de procesamiento (ScannedPage, ProcessingJob, ProcessingLog, etc.) almacenan múltiples marcas de tiempo (captured\_at, decoded\_at, started\_at, finished\_at) para registrar cada paso del pipeline. Esto proporciona un rastro completo de cada hoja procesada, lo cual es crítico para depurar errores de reconocimiento o atender reclamos (por ejemplo, poder mostrar la imagen escaneada original y cómo se interpretó cada marca). Cabe señalar que muchas de estas tablas de log en ingesta son consideradas "inmutables" una vez creadas (no se actualizan ni borran, solo crecen), privilegiando así la *integridad de auditoría* por encima del almacenamiento – un trade-off aceptado para garantizar que siempre haya evidencia consultable de lo ocurrido en cada calificación.
* **Idempotencia en procesamiento OCR:** Dado que la ingesta de respuestas puede involucrar procesos repetibles (reintentos de escaneo, etc.), el modelo incluye mecanismos para asegurar **idempotencia**. Concretamente, se usa el hash SHA-256 de cada imagen (image\_sha256 en ScannedPage) como identificador único de página capturada, marcándolo como clave única. Esto impide almacenar dos veces la misma hoja (misma imagen) por error, evitando duplicaciones en la calificación. Si un docente intenta escanear dos veces la misma página, el sistema reconocerá el hash duplicado y puede omitir o marcar el segundo ingreso. Este detalle de diseño previene distorsiones en los resultados (p. ej., que una misma respuesta se contabilice doble) y facilita que el proceso sea confiable incluso si se realizan reintentos manuales o reenvíos desde la app móvil.
* **Desempeño y escalabilidad considerados:** Aunque el modelo es principalmente normalizado, se tomaron decisiones orientadas a **rendimiento** en escala. Por ejemplo, ciertas consultas frecuentes (como obtener todas las preguntas de una evaluación con sus opciones y respuestas de un alumno) están optimizadas gracias al modelo de snapshot, que permite un join directo entre EvaluationQuestion, EvaluationOption y StudentAnswer sin tener que buscar en las tablas globales de preguntas salvo para trazabilidad. Asimismo, se han planificado **índices** en campos clave para acelerar búsquedas (índices por evaluation\_id en muchas tablas, por student\_id, por topic\_id en preguntas para filtrado por tema, etc.). Incluso se consideró, de ser necesario, la creación de **vistas materializadas** o tablas resumen para reportes agregados (por ej., desempeño promedio por dificultad) en caso de cargas muy altas de datos, aunque esto se evaluará con el uso. La escalabilidad horizontal está soportada al poder particionar por contexto (por ejemplo, si un componente como la ingesta produce muchísimos registros, podría aislarse en una base o schema separado y escalable independientemente).

# 12. Casos de uso

A partir de los requisitos y actores identificados, se definió un conjunto de **casos de uso principales** que ilustran cómo interactuarán Docentes y Coordinadores con el sistema GRADE. A continuación se describen los más relevantes:

* **Gestión del Banco de Preguntas:** Un *Docente* (o Administrador del sistema) crea y administra preguntas en el banco centralizado. En este caso de uso, el usuario ingresa al módulo de Banco de Preguntas, añade una nueva pregunta proporcionando el enunciado, tipo de pregunta (p.ej. selección múltiple), las alternativas con su marcación de respuesta correcta, y metadatos como dificultad, unidad temática, etc. El sistema valida la información (por ejemplo, que haya al menos una opción marcada como correcta si corresponde) y guarda la pregunta en el banco con estado activa. El docente puede posteriormente editar la pregunta para actualizarla (lo cual genera una nueva versión interna) o desactivarla si ya no se debe usar. Este caso de uso asegura que el banco de ítems esté siempre disponible y actualizado para elaborar evaluaciones.
* **Diseño de evaluación estandarizada (Coordinador):** Un *Coordinador académico* utiliza GRADE para diseñar y publicar una evaluación común para varias secciones o cursos, garantizando la coherencia curricular. En este caso de uso, el coordinador crea una nueva **evaluación** en el sistema, definiendo sus parámetros generales (título, fecha, duración, criterios de calificación). Luego selecciona preguntas del banco de preguntas institucional (posiblemente filtrando por asignatura, unidad, tópico, etc.) y las agrega a la evaluación, ajustando puntajes si es necesario. El coordinador puede emplear funcionalidades de **plantillas**, por ejemplo clonando una evaluación previa como base, o compartiendo un modelo de prueba con otros docentes. Una vez satisfecho con la composición, el coordinador **publica** la evaluación, lo que notifica a los docentes involucrados y bloquea la edición further. Este caso de uso permite mantener consistencia: todos los docentes de un mismo ramo pueden aplicar exámenes alineados en contenido y dificultad porque provienen de un diseño centralizado del coordinador.
* **Generación y publicación de una evaluación (Docente):** Un *Docente* prepara una prueba para sus estudiantes a través del sistema. Este caso de uso inicia cuando el docente ingresa al módulo de **Evaluaciones** y elige crear una nueva evaluación para su curso. El docente selecciona el curso/sección, ingresa detalles como el título del examen y fecha programada, luego procede a armar la prueba seleccionando preguntas desde el banco personal o institucional. Puede buscar preguntas por tema o dificultad, visualizar sus detalles y añadirlas. Para cada pregunta incluida, asigna un puntaje o peso específico. Tras compilar todas las preguntas deseadas, el docente genera un **PDF de la evaluación** con un clic: el sistema compone automáticamente el documento con el listado de preguntas, formateado con numeración, espacio para respuestas y un código QR único (que codifica la identificación de la evaluación y eventualmente la del estudiante si se generan copias nominadas). El docente descarga este PDF para imprimirlo o distribuirlo digitalmente. Alternativamente, el sistema podría permitir **publicar en formato digital seguro** para rendición en línea (si estuviera contemplado). En cualquiera de los casos, la evaluación queda lista y oficialmente **publicada** en el sistema (cambia a estado "Publicada"), quedando habilitada para recibir respuestas.
* **Aplicación y calificación automática de la evaluación:** Este caso de uso cubre el desarrollo posterior a la aplicación de la prueba en el aula. Tras realizar el examen, el *Docente* recopila las hojas de respuesta de los estudiantes (si fue en papel). Con la app móvil de GRADE, el docente inicia un **lote de ingesta** para esa evaluación: abre la app, selecciona la evaluación correspondiente y utiliza la cámara para escanear cada hoja con respuestas. La app captura cada página, decodifica el QR para saber de qué estudiante (o al menos de qué evaluación) se trata, y envía las imágenes al servidor. El backend procesa las imágenes: por cada página genera detecciones de marcas y determina las opciones seleccionadas por el estudiante, comparándolas con la clave correcta almacenada (evaluation\_options) para calcular puntajes por pregunta. Una vez procesado todo el lote, el sistema calcula la **nota final** de cada estudiante según el criterio definido (por ejemplo, convierte el porcentaje de aciertos a una nota en escala 1-7). El docente puede alternativamente cargar un archivo CSV con resultados (si la prueba se tomó en Google Forms u otra plataforma) o incluso ingresar manualmente algunas respuestas vía interfaz web, y el sistema igualmente ejecutará la lógica de calificación automática. El resultado de este caso de uso es que las evaluaciones pasan al estado "Calificada" y se registran los resultados individuales de cada estudiante sin intervención manual del docente en la corrección. Esto ahorra tiempo significativo al docente y reduce errores de corrección.
* **Revisión de resultados y generación de reportes:** Tras la calificación, el *Docente* accede a los **reportes de desempeño** provistos por GRADE. En este caso de uso, el docente entra al módulo de Resultados, selecciona una evaluación específica y visualiza un resumen: promedio del curso, distribución de notas (p. ej. cuántos estudiantes obtuvieron sobre cierto puntaje), además de indicadores como la **dificultad** promedio de las preguntas (según porcentaje de aciertos) o identificación de preguntas que resultaron más difíciles (mayor tasa de error). También puede detallar por estudiante, viendo la nota individual de cada uno y, si fuese necesario, las respuestas marcadas por cada estudiante en cada pregunta (por ejemplo, para revisar con el alumno sus errores). GRADE almacena también el **historial de evaluaciones** del curso, permitiendo al docente comparar el rendimiento de sucesivas pruebas. Un *Coordinador* podría acceder a reportes agregados a nivel de curso o cohorte (sin ver datos individuales sensibles) para evaluar la coherencia y dificultad de las evaluaciones aplicadas en distintas secciones. Este caso de uso ofrece valor analítico, ayudando a reflexionar sobre la enseñanza: el docente identifica qué contenidos no fueron bien comprendidos (preguntas con bajo porcentaje de acierto) y puede retroalimentar esas áreas en clases posteriores. Todos estos reportes se pueden exportar o imprimir directamente desde la plataforma para archivado o compartir con dirección académica.

Los casos de uso anteriores cubren el flujo completo para los usuarios clave del sistema. En esencia, GRADE permite a los **Coordinadores** estandarizar y supervisar las evaluaciones, y a los **Docentes** ejecutar todo el ciclo de evaluar – desde la preparación hasta la corrección y análisis – de forma eficiente y soportada por la tecnología. Cada interacción del usuario con el sistema corresponde a uno de estos casos de uso, los cuales fueron elaborados con detalle en la fase de diseño para garantizar que cubrimos todas las situaciones relevantes (incluyendo manejo de errores, excepciones, etc., en cada caso).

Vale notar que GRADE también considera integraciones futuras con otros sistemas (como LMS institucionales) a través de APIs, lo cual introduce potenciales casos de uso para *Aplicaciones Externas*. Por ejemplo, una plataforma externa podría invocar la API de GRADE para crear una evaluación automáticamente desde su interfaz, o para obtener los resultados de una prueba y mostrarlos en su propio dashboard[[68]](https://github.com/Wanku-CL/wiki-wanku/blob/89467939d09200f1c22aba5a466290c8d955c175/products/grade/02-discovery/actors.md#L2-L5). Si bien estos casos de uso de integración no se implementan en el MVP, la arquitectura los ha contemplado para una fase posterior.

# 13. Avance etapa

El proyecto GRADE se encuentra actualmente transitando la tercera etapa de desarrollo, habiendo completado satisfactoriamente las dos primeras según la planificación. A continuación se detalla el **estado de avance por fase** (etapa):

* **Fase 1 – Fundamentos:** *Completada.* En esta etapa inicial (finalizada el 06 de octubre) se implementaron los cimientos del sistema. Esto incluyó la configuración de la base de datos, la creación de las entidades fundamentales (usuarios, cursos, estudiantes, preguntas, etc.) y las operaciones CRUD básicas asociadas. Al cierre de Fase 1, el sistema ya permitía gestionar el banco de preguntas (crear/editar ítems) y contar con un catálogo de cursos y usuarios con roles. Esta fase sentó la plataforma mínima sobre la cual construir las funcionalidades de evaluaciones.
* **Fase 2 – Evaluaciones:** *Completada.* Culminada hacia mediados de octubre (18 de octubre), en esta fase se desarrollaron todas las funcionalidades relacionadas con la creación y publicación de evaluaciones. El logro principal fue habilitar que un usuario pudiera armar una prueba seleccionando preguntas, asignando puntajes y generando automáticamente un PDF con formato de examen y código QR. También se implementó el manejo de estados de la evaluación (borrador, publicada) y las restricciones correspondientes (por ejemplo, una vez publicada ya no se puede editar). Al terminar Fase 2, GRADE era capaz de generar exámenes listos para aplicar, cumpliendo con el objetivo esperado (“pruebas listas para aplicar”[[69]](https://github.com/Wanku-CL/wiki-wanku/blob/89467939d09200f1c22aba5a466290c8d955c175/products/grade/01-roadmap/ROADMAP.md#L13-L16)). Se realizaron pruebas internas con evaluaciones de ejemplo para verificar la integridad del proceso.
* **Fase 3 – Ingesta:** *En curso.* Esta fase, correspondiente a la ingesta multicanal de respuestas, está en desarrollo durante la segunda quincena de octubre. Incluye varias sub-tareas: la carga de respuestas vía archivos CSV (cuya implementación está casi terminada), el desarrollo de la **aplicación móvil de captura** y la integración del servicio **OCR** para reconocimiento de marcas, además del manejo de errores (ej. páginas ilegibles, respuestas no detectadas). A la fecha, en el backend se ha avanzado con la estructura de datos y endpoints necesarios para recibir resultados (CSV u OCR) y con la lógica de cálculo de puntajes a partir de respuestas ingresadas. Sin embargo, el componente de **app móvil** y la incorporación efectiva del OCR están retrasados respecto al plan: la app Ionic/Angular se encuentra en etapa de prototipo y el pipeline de procesamiento de imágenes aún no se integra. Como contingencia temporal, se ha priorizado asegurar que la carga de respuestas vía CSV/Web funcione correctamente para disponer de algún mecanismo de calificación automática mientras se termina la app. La Fase 3 se considera completada cuando el sistema permita cargar lotes de respuestas escaneadas desde el móvil y procesarlas automáticamente – hito que se espera lograr a finales de octubre, recuperando el atraso en el desarrollo móvil.
* **Fase 4 – Calificación Automática:** *Pendiente (no iniciada).* Programada para inicios de noviembre, esta fase abordará el refinamiento final del motor de calificación. Si bien el cálculo básico de puntajes se implementa en Fase 3, en Fase 4 se afinarán detalles como la conversión de puntaje bruto a nota en escala 1-7 u otras escalas configurables, la aplicación de criterios de aprobación (umbrales) y el registro definitivo de las notas en la base de datos. También se incluirá aquí la generación de estadísticas básicas post-calificación de manera automática. Dado que parte de esta lógica ya está esbozada, se espera que Fase 4 sea relativamente breve enfocándose en consolidar y probar el proceso de calificación end-to-end.
* **Fase 5 – Resultados & Auditoría:** *Pendiente (no iniciada).* Prevista para la segunda semana de noviembre, en esta fase final se desarrollarán las funcionalidades de cierre: visualización de resultados, exportación de reportes, y los componentes de auditoría y seguridad faltantes. Se implementarán las pantallas de reportes para docentes (y coordinadores, en su caso), mostrando los indicadores de desempeño de cada evaluación. Asimismo, se activarán los logs de auditoría de acciones de usuario y se prepararán mecanismos de exportación de datos (por ejemplo, para descargar todas las notas en CSV, o generar un informe PDF de resultados por curso). Al culminar Fase 5, el sistema estará funcionalmente completo para su MVP y listo para pruebas de aceptación finales.

Cada fase ha concluido (o concluirá) con una ronda de **pruebas UAT** (User Acceptance Testing) internas, según lo planificado, con involucramiento de usuarios simulados (docentes) para verificar que las características implementadas cumplen los requisitos. Hasta ahora, las fases 1 y 2 superaron sus respectivas pruebas UAT sin hallazgos críticos, validando la usabilidad de la gestión de preguntas y generación de exámenes. La fase 3 en curso tendrá pruebas parciales (por ejemplo, verificando la carga CSV y cálculos de nota) y una vez completa, se realizará una UAT integral cubriendo la ingesta móvil.

En términos de **cronograma**, las Fases 1 y 2 se completaron dentro de los plazos estimados. La Fase 3 lleva un ligero retraso en el aspecto móvil, pero el equipo está aplicando esfuerzos adicionales para finalizarla sin que esto aplace significativamente el hito final. Se monitorea el cronograma cuidadosamente para iniciar Fase 4 a tiempo. La meta continúa siendo entregar el MVP totalmente operativo al **23 de noviembre de 2025**, tal como se marcó en el roadmap[[71]](https://github.com/Wanku-CL/wiki-wanku/blob/89467939d09200f1c22aba5a466290c8d955c175/products/grade/01-roadmap/ROADMAP.md#L64-L68), con todas las fases concluidas y aprobadas en pruebas.

# 14. Desarrollo

En esta sección se describe el estado actual del **desarrollo** del software, incluyendo prácticas de ingeniería, herramientas utilizadas y aspectos de calidad:

* **Metodología de desarrollo:** El equipo ha seguido una aproximación **tradicional cascada incremental**, desarrollando por iteraciones alineadas con las fases del roadmap. Se realizan reuniones de seguimiento frecuentes para evaluar progreso y obstáculos. Al final de cada fase se efectúa una retrospectiva y una planificación detallada de la siguiente. Esta forma de trabajo ha permitido adaptarse a cambios menores y repriorizar tareas (por ejemplo, ajustar el enfoque a CSV ingest cuando la app móvil se atrasó) sin desvíos mayores. El proyecto es administrado usando herramientas como **Github Projects** para gestionar historias de usuario, tareas y documentación respectivamente.
* **Gestión de código fuente:** Todo el código se encuentra versionado en repositorios Git (plataforma GitHub) organizados por componente. Se estableció una estructura de ramas con main/develop y ramas feature para desarrollo paralelo, siguiendo convención *GitFlow*. Se definió una **guía de estilo de código** para uniformar la sintaxis (nombres de variables en inglés, formato de comits, etc.) y se realizan *code reviews* para cada pull request, principalmente por el Tech Lead, asegurando calidad y compartiendo conocimiento del código entre el equipo.
* **Integración continua (CI) y despliegue:** Se configuró un pipeline de **Integración Continua** mediante **GitHub Actions**. Cada vez que se hace push a la rama develop o se abre un pull request, el pipeline ejecuta automáticamente los siguientes pasos: instalación de dependencias, compilación (en caso del frontend), y ejecución de la suite de **pruebas automatizadas**. Esto garantiza que nuevas incorporaciones no rompan funcionalidades existentes. Solo si el pipeline pasa verde se procede a mergear cambios a main. Adicionalmente, se prepararon entornos de prueba (staging) donde se despliega el código integrado para pruebas manuales y demostraciones. Aunque el despliegue continuo (CD) a producción no se ha configurado todavía (dado que el MVP aún no se lanza públicamente), se planifica utilizar también GitHub Actions para automatizar el despliegue a un entorno productivo en la nube cuando llegue el momento, siguiendo la estrategia definida (contenedores Docker en infraestructura AWS).
* **Estado de implementación por componente:** El **frontend web** escrito en React/Next.js está casi completo en funcionalidad: todas las vistas principales (login, banco de preguntas, editor de evaluación, visor de resultados) están implementadas y estilizadas según el diseño. Resta pulir detalles de usabilidad menores y validar en múltiples navegadores y tamaños de pantalla (responsividad). El **backend (API)** tiene implementados la mayoría de la logica core: registro/login de usuarios con JWT, CRUD de preguntas, listado de banco, creación/publicación de evaluaciones, carga de resultados (actualmente soportando CSV, y preparado para integrar OCR), cálculo de notas, y obtención de reportes básicos, sin tener aun la exposición de los endpoints. La **aplicación móvil** se encuentra en prototipo inicial: se ha creado el proyecto base en Ionic/Angular y desarrollado la pantalla de login y la de captura de QR, pero falta integrar la funcionalidad de toma de fotografía y envío al backend. El microservicio de **OCR/OMR** está en desarrollo lógico: se han probado llamadas a la API para reconocer texto y patrones de burbujas, con buenos resultados (≥95% de precisión en condiciones controladas), pero falta encajar esto en la arquitectura del sistema (conectar con IngestBatch y ScannedPage en la base de datos de producción). Se espera consolidar esta integración durante la presente fase 3.
* **Pruebas y control de calidad:** Desde el inicio se estableció un **plan de pruebas** abarcando varios niveles. Actualmente se han implementado pruebas **unitarias** para los componentes críticos del backend (por ejemplo, validación de lógica de cálculo de notas, funciones utilitarias de versión de preguntas, etc.) utilizando **Jest** como framework de testing en Node.js. Asimismo, se desarrollaron pruebas unitarias en el frontend con Jest/React Testing Library para algunos componentes de interfaz (como el formulario de agregar pregunta, asegurando que las validaciones funcionan). Para el **módulo de Banco de Preguntas**, en particular, se preparó un completo conjunto de **pruebas de integración**: casos que crean una pregunta, luego la incluyen en una evaluación, luego simulan la carga de una respuesta de estudiante, verificando que en cada paso las entidades se crean y relacionan correctamente en la base de datos. Estas pruebas garantizan que el flujo básico de crear pregunta -> generar prueba -> calificar respuesta funcione de punta a punta. También se han utilizado pruebas manuales exploratorias en la interfaz gráfica para asegurar la usabilidad.

Al concluir cada fase, se llevan a cabo **pruebas de aceptación (UAT)** con miembros del equipo actuando como usuarios finales, siguiendo guiones de prueba basados en los casos de uso. Por ejemplo, después de Fase 2 se simuló que un docente realice todo el proceso de crear y publicar una evaluación, detectar posibles confusiones en la UI o errores no atrapados automáticamente. Hasta ahora las UAT han sido exitosas con ajustes menores derivados (textos de ayuda más claros, mensajes de error más descriptivos, etc.). Se tiene previsto realizar una **UAT global** al finalizar la fase 5, donde se ejecutará el flujo completo integrando todos los módulos en un ambiente de staging lo más cercano a producción, durante al menos 3 días, con la participación de usuarios finales piloto.

Adicionalmente, en la fase final se aplicarán pruebas **no funcionales**: pruebas de carga y estrés (usando herramientas como **JMeter**) para asegurar que el sistema soporta un volumen esperado de concurrentes (por ejemplo, varios docentes calificando al mismo tiempo en semanas de exámenes), y una revisión de **seguridad** (escaneo con **OWASP ZAP**) para identificar vulnerabilidades antes del despliegue[]](https://github.com/Wanku-CL/wiki-wanku/blob/89467939d09200f1c22aba5a466290c8d955c175/products/grade/08-development-and-deployment/README.md#L16-L24). Estas actividades de QA están calendarizadas en la última semana previo al lanzamiento.

En general, el desarrollo avanza con buenas prácticas de ingeniería de software, combinando **automatización de pruebas**, control de versiones, integración continua y revisiones de código para asegurar calidad. El equipo se ha mantenido colaborativo y flexible ante cambios, lo que ha permitido sobrellevar contratiempos (como el cambio de tecnología móvil) con impacto acotado. Se prioriza cumplir con la funcionalidad comprometida sin sacrificar la estabilidad del sistema, apoyándose en la suite de pruebas para validar cada incorporación. Con un 70% del desarrollo completo, el enfoque sigue puesto en culminar las características pendientes manteniendo este estándar de calidad hasta la entrega final.

# 15. Evaluación Económica

El análisis económico-financiero del proyecto GRADE (Generación y Registro Automatizado de Evaluaciones) se realizó considerando un horizonte temporal de seis años y una tasa de descuento del 10,28 %, correspondiente al costo promedio ponderado de capital (WACC).

La proyección del flujo de caja incorpora los ingresos esperados por la implementación y operación del sistema, junto con los costos operativos, la depreciación de activos y la carga impositiva del 25 %.

#### Inversión inicial

La inversión total requerida asciende a $ 24.566.682 CLP, monto destinado principalmente al desarrollo del software, infraestructura tecnológica y recursos humanos especializados.

Este desembolso se registra en el año 0 como flujo negativo dentro del modelo financiero.

#### Flujo de caja proyectado

El flujo de caja presenta un comportamiento creciente durante el periodo evaluado.

Los flujos netos positivos comienzan a partir del segundo año, reflejando un rápido punto de equilibrio operativo.

Al término del sexto año, el flujo acumulado alcanza los **$286.456.643 CLP**, lo que evidencia la generación sostenida de valor económico por parte del proyecto.

#### Indicadores Financieros

| **Indicador** | **Resultado** | **Interpretación** |
| --- | --- | --- |
| **VAN (Valor Actual Neto)** | $ 71.882.960 | El VAN positivo confirma que el proyecto crea valor económico sobre la inversión inicial, demostrando su viabilidad. |
| **TIR (Tasa Interna de Retorno)** | 51,8 % | Supera ampliamente la tasa de descuento (10,28 %), indicando una alta rentabilidad anual esperada. |
| **Payback (Periodo de Recuperación)** | 3 años | La inversión se recupera completamente al inicio del cuarto año de operación. |

#### Análisis de sensibilidad

Se evaluó la rentabilidad ante una variación de la tasa de descuento.  
 Incluso con un aumento hasta el **15 %**, el **VAN** permanece positivo (≈ $ 60 millones), lo que demuestra **baja sensibilidad al riesgo financiero** y una alta estabilidad del modelo de ingresos.

#### Conclusión económica

El proyecto **GRADE** es **financieramente viable y rentable**, con una recuperación temprana de la inversión, un VAN elevado y una TIR considerablemente superior al costo de capital.  
 El comportamiento del flujo de caja confirma que la inversión inicial genera beneficios sostenidos y escalables, respaldando la **sostenibilidad económica del proyecto** en el mediano plazo.

# 16. Estado general de avance

En términos globales, el proyecto GRADE presenta un **estado de avance del ~70%**, lo que indica que más de dos tercios del trabajo planificado para el MVP ya han sido completados. Las funcionalidades esenciales – desde la creación de preguntas hasta la generación de evaluaciones y registro de resultados – se encuentran implementadas y operativas en el sistema. Los **resultados hasta ahora son positivos**: las fases críticas iniciales se cumplieron en tiempo y forma, y el sistema, en su porción ya construida, ha demostrado ser consistente con los objetivos planteados (por ejemplo, ya es posible centralizar el ciclo de evaluaciones en una sola plataforma, uno de los objetivos estratégicos primarios).

El proyecto se mantiene **dentro del cronograma** establecido en la planificación inicial, con la salvedad de un leve retraso en la parte móvil/OMR mencionado anteriormente. Este retraso, no obstante, está siendo gestionado y no compromete la **fecha de entrega final** del MVP, pues se están reasignando recursos y priorizando tareas críticas para recuperar el ritmo. El índice de progreso (70%) corresponde aproximadamente a haber finalizado 2 de las 3 grandes etapas de desarrollo funcional (fundamentos + evaluaciones completas), quedando la última etapa (ingesta/calificación) en proceso de culminación.

En cuanto a **alcance**, no ha habido incrementos fuera de lo previsto; por el contrario, se han respetado los límites definidos. Las decisiones de cambio introducidas (ver Control de Cambios) han sido principalmente reorganizaciones internas o elecciones tecnológicas que no alteran el alcance funcional comprometido, sino que buscan mejor implementabilidad. Por ejemplo, el cambio a Ionic para la app móvil no agrega funcionalidades nuevas, solo modifica la vía para entregarlas, y la postergación de la definición de APIs externas simplemente difiere una tarea sin eliminarla del alcance final.

Desde la perspectiva de **recursos y esfuerzo**, el equipo ha operado de acuerdo a lo planificado (5 desarrolladores equivalentes a tiempo completo). No se han reportado bajas ni incorporaciones de miembros, y la carga de trabajo se ha mantenido balanceada. Sí hubo necesidad de horas extra puntuales en la fase 3 para abordar el rezago del módulo OCR, pero se considera dentro de lo manejable. El riesgo de no completar a tiempo la funcionalidad de ingesta móvil está identificado pero bajo control, con mitigaciones en curso (posible reducción del alcance de la app para MVP, concentrándose solo en captura y enviando imágenes, dejando funcionalidades avanzadas de offline o de verificación para después).

En cuanto a **calidad**, los indicadores son alentadores: las pruebas realizadas muestran un sistema estable en las áreas ya desarrolladas; la retroalimentación interna de usuarios de prueba (docentes del equipo) ha sido favorable en términos de usabilidad; y no se han encontrado defectos críticos en las funcionalidades implementadas (los bugs detectados han sido de severidad baja o trivial, corregidos rápidamente). Esto sugiere que, aunque resta trabajo por hacer, el producto construido hasta ahora cumple con los criterios de aceptación definidos.

En síntesis, el proyecto se encuentra **bien encaminado**, con la mayor parte del trabajo duro ya realizado y una visibilidad clara de las tareas pendientes. El **nivel de avance del 70%** refleja que estamos en la recta final hacia el MVP. La planificación y controles han funcionado para mantener el proyecto orientado, y si bien existen desafíos por resolver en la etapa final, el equipo está confiado en lograr los objetivos dentro del plazo. De mantenerse el ritmo y enfocando esfuerzos en los puntos críticos restantes, GRADE alcanzará su entrega prevista en condiciones óptimas, proporcionándole a Wanku una solución innovadora y funcional para la gestión de evaluaciones automatizadas en el tiempo estipulado[[71]](https://github.com/Wanku-CL/wiki-wanku/blob/89467939d09200f1c22aba5a466290c8d955c175/products/grade/01-roadmap/ROADMAP.md#L64-L68).

# 17. Próximos pasos

Para completar el último tramo del proyecto y asegurar un cierre exitoso, se delinean a continuación los **próximos pasos y actividades** clave que el equipo realizará en las siguientes semanas:

* **Finalizar desarrollo de la aplicación móvil (Ionic/Angular):** Completar las pantallas y flujos restantes de la app de ingesta móvil. Es prioridad implementar la funcionalidad de captura de imágenes con cámara y su envío en tiempo real al backend. Se simplificará inicialmente la UI para enfocarse en la captura básica y se agregará gradualmente las mejoras (como indicación visual de calidad de escaneo, re-captura si es borrosa, etc.). Una vez lista la funcionalidad principal, realizar pruebas en dispositivos Android/iOS representativos para depurar problemas de compatibilidad.
* **Integración completa del módulo OCR/OMR:** Ensamblar el pipeline de procesamiento de imágenes en el entorno de producción. Esto implica conectar la recepción de imágenes en el backend con el servicio OCR (Azure Cognitive Services u otro) y luego traducir los resultados de reconocimiento a registros RecognitionMapping y StudentAnswer en la base de datos. Se pondrá especial atención en calibrar el reconocimiento (p. ej., ajustar parámetros de umbral de detección de marcas para minimizar falsos negativos/positivos). También se implementarán manejos de errores robustos: si una página no puede decodificarse bien, registrar el caso y quizás marcarla para revisión manual. La prioridad es lograr que al escanear una hoja típica, el sistema reconozca automáticamente >95% de las respuestas y las asocie correctamente. Este paso es crítico para cumplir la promesa de calificación automática.
* **Culminar motor de calificación y conversión de notas:** Con los datos de respuestas ya ingresados vía OCR o CSV, finalizar la lógica que calcula la nota final. Esto incluye aplicar la **escala de notas configurada** (por ejemplo, si la evaluación está en escala 1-7 con cierta curva, transformar el puntaje obtenido al valor en esa escala) y respetar los **umbrales de aprobación** definidos (posibilidad de que cada curso tenga un porcentaje mínimo distinto para la nota mínima aprobatoria). Se probarán varios escenarios (100% de aciertos, 0%, casos intermedios, con distintos umbrales) para validar la correcta conversión. Adicionalmente, incorporar la consideración de **ponderaciones** si alguna sección de la prueba tiene peso distinto.
* **Implementar módulos de resultados y reportes:** Desarrollar la interfaz de **visualización de resultados** para docentes (y coordinadores). Esto abarcará presentar en la web los datos ya calculados: lista de estudiantes con sus notas, indicadores agregados de la evaluación, y gráficos básicos (por ejemplo, gráfico de barras del histograma de notas, o un listado de preguntas resaltando el % de estudiantes que acertaron cada una). Paralelamente, implementar la funcionalidad de **exportar resultados** – al menos en CSV o PDF – para que el docente pueda descargar un acta de calificaciones o un informe resumido. Para coordinadores, habilitar vista de estadísticas consolidadas entre cursos si es aplicable (p.ej., comparativa entre secciones del mismo ramo). Estos reportes serán validados con usuarios para asegurarse de su utilidad y claridad.
* **Pruebas finales (QA) y iteración de ajustes:** Una vez integradas todas las funcionalidades, el equipo llevará a cabo un ciclo completo de **pruebas finales**. Esto incluye pruebas funcionales exhaustivas atravesando todos los casos de uso de principio a fin (incluyendo los módulos recién incorporados de móvil y resultados), pruebas de usabilidad con usuarios de prueba adicionales, y **pruebas de carga** simulando múltiples docentes y evaluaciones concurrentes. Se medirá el rendimiento del sistema bajo estrés (latencia en procesamiento OCR con varios lotes simultáneos, tiempo de generación de reportes con muchos estudiantes, etc.) y se optimizará si algún cuello de botella es identificado. También se conducirá una **auditoría de seguridad** final: revisión manual de configuración de permisos, pruebas de penetración básicas, verificación de que los datos sensibles (ej. resultados de estudiantes) no sean accesibles sin autenticación adecuada, etc. Cualquier defecto crítico encontrado en esta etapa se corregirá de inmediato, y se harán ajustes menores para mejorar la experiencia (por ejemplo, si en las pruebas de usuario se detecta confusión en algún mensaje o flujo, se refinará). Este pulido final garantizará que el sistema llegue a la entrega en un estado sólido y confiable.
* **Preparación para despliegue y entrega:** Paralelamente a las pruebas finales, se preparará el entorno productivo para el **despliegue**. Se configurarán los servidores, base de datos en producción, variables de entorno (claves API, etc.) y se realizarán pruebas de **deploy en staging** para simular la instalación en limpio del sistema. Se elaborará documentación complementaria: una **Guía de Usuario** breve para los docentes/coordinadores que usarán el sistema en piloto, explicando cómo realizar las tareas principales; y una **Documentación Técnica** resumida para el equipo de TI de Wanku sobre cómo está construido e instalado el sistema, para facilitar mantenimiento. Finalmente, se coordinará con los patrocinadores y stakeholders una **demostración final** del MVP mostrando todas las funcionalidades operativas antes del lanzamiento oficial.

Con estas acciones, el proyecto encarará su fase de cierre de manera controlada. Los próximos pasos se centran en **integrar y validar** los componentes restantes, poniendo énfasis en calidad y cumplimiento de los requisitos no funcionales (rendimiento, seguridad) en la recta final. Si se siguen rigurosamente, permitirán que GRADE alcance su hito de MVP entregado en tiempo, cumpliendo con los objetivos de facilitar a docentes y coordinadores una herramienta avanzada para la gestión integral de evaluaciones automatizadas. El equipo mantiene el foco y el compromiso de ejecutar estos pasos finales según lo planificado, confiando en que el resultado será un sistema listo para generar impacto positivo en el proceso evaluativo de la institución.

# 18. Conclusiones

#### Conclusion — María Fernanda Garay

Participating in the development of Wiki Wanku (GRADE) has been a deeply formative experience. This project has allowed me to apply the technical knowledge acquired throughout my studies while also strengthening essential soft skills such as organization, communication, and teamwork.  
  
 Looking back, I realize how much I have grown — not only academically, but also personally. I have learned to stay calm under pressure, to manage my time more effectively, and to prioritize what truly matters over what feels urgent. The process has not been free of challenges: balancing studies with personal responsibilities has been demanding, but it has also taught me the value of discipline and resilience.  
  
 Our team has shown commitment, creativity, and adaptability. We have built a solid foundation with a functional, modular, and scalable system that can continue to evolve in the future. I truly believe that GRADE has real potential to become a valuable tool for teachers, improving how they manage their time and provide feedback to their students.  
  
 I feel proud of what we have accomplished so far. Although there are still adjustments to make, I am convinced that every challenge we have overcome brings us closer to a more professional and meaningful result. This project not only reflects what we are capable of creating as a team, but also how much we can grow when we work with passion, consistency, and purpose.

#### Conclusion — Maximiliano Toledo

This document summarizes the progress achieved in the development of the **GRADE (Automated Generation and Recording of Evaluations)** project, ensuring continuous alignment with the objectives defined in the initial phases and consolidating a clear vision of the final product. Throughout this stage, the team successfully fulfilled most of the goals established during Phase 1, strengthening both the technical and organizational components of the project.

During this phase, the overall system architecture was consolidated, integrating the question bank, evaluation management, and data model, which together form the core of the MVP. Likewise, the project schedule and the financial flow analysis allowed for a more precise projection of time, costs, and resources to ensure the continuity of development.

The **economic and financial evaluation** demonstrated the project’s viability, showing a positive NPV and an IRR well above the discount rate, confirming the system’s sustainability in the medium term. Furthermore, the creation of the data model and technical documentation helped standardize processes and guarantee data integrity across all modules.

This process has provided a comprehensive learning experience, combining project management, software engineering, and collaborative teamwork. Coordinated task distribution, effective communication, and the use of agile methodologies enabled the team to address challenges with a practical and results-oriented approach. Ultimately, this phase reaffirms the team’s commitment to quality, innovation, and the development of a functional product that contributes meaningful value to the educational field.

#### Conclusion — Rodrigo Ulloa

Phase 2 of GRADE confirmed that our architecture (web + mobile with Next.js, Node.js, and PostgreSQL) robustly supports the project’s critical objectives. Reaching ~70% of the MVP—with the question bank and evaluation management operational, including PDF exam generation with QR codes and result recording—validates that our technical framing and data design were sound choices. The economic evaluation further supports feasibility (positive NPV, IRR above the cost of capital, and a payback of roughly three years), which—together with implemented traceability and security—strengthens the case for continued investment.

The main challenge was the delay in the Mobile Ingestion (OMR) module. We addressed this risk with concrete actions: prioritizing the web–backend core, providing a CSV-ingestion contingency, and laying out a technical plan to integrate the recognition pipeline and the Ionic/Angular app in the next stage. This experience reinforced three lessons: (1) the importance of freezing a clear MVP and defending it against scope creep; (2) the value of incremental integration and early testing to reduce technical uncertainty; and (3) the need to maintain traceability and versioning at the data level to guarantee correctness and auditability.

Looking ahead to the final phase, my focus will be to close the **capture → recognition → grading** chain, strengthen QA with load and security testing, and enable reports that deliver immediate value to instructors and coordinators. I commit to leading the OMR integration and maintaining the quality standard we have reached. With the team aligned, a clear roadmap, and a product already useful to users, I am confident GRADE will hit its MVP milestone on time and in strong shape—ready to make a real impact on academic assessment management.

# 19. Reflexión

Throughout the development of GRADE, the team strengthened its capacity to integrate engineering principles with educational innovation. Designing a modular architecture and a traceable data model required applying project-management discipline and collaborative coordination between technical and pedagogical perspectives. The experience highlighted the importance of clear communication, version control, and incremental planning to manage complex deliverables. By iterating through each phase, the team learned to balance scope, time, and quality while maintaining focus on user value. Beyond a software product, GRADE became a learning process in leadership, teamwork, and systemic thinking—skills that extend far beyond this project and into future professional practice.

# 20. Anexos

1. Ref. Documentación Matriz FODA.docx
2. Ref. Documentación SRS\_Proyecto\_GRADE.docx
3. Ref. Documentación Modelo\_de\_Datos\_Proyecto\_GRADE.docx
4. Ref Diccionario de datos.docx
5. Ref FLUJO DE CAJA RENTABILIDAD .xlsx