Proyecto Integrador de Programación

Profesor: **Ricardo Pérez**
riperez@utalca.cl

Semestre 2025-II

Descripción general del Proyecto

Instrucciones

Los estudiantes deben conformar equipos de **4 integrantes** para desarrollar una **aplicación completa** que resuelva un problema específico de su elección. El sistema se construirá de forma **progresiva** y debe incluir: **frontend interactivo**, **lógica de negocio en módulos (API interna)**, **persistencia inicial en archivos JSON/CSV**, **migración a SQLite en el segundo tramo**, **exposición REST básica**, **consumo de una API externa**, **autenticación mínima**, **testing esencial** y **documentación profesional**.

Duración: 18 semanas | **Metodología**: ABP con tablero Kanban y sprints de 3 semanas | **Evaluación**: 3 hitos con presentaciones y demos

Competencias a desarrollar

- Diseñar la **arquitectura por capas** (rutas/controladores/servicios/datos).
- Implementar lógica de negocio y **exponerla progresivamente**.
- Manejar **persistencia** primero con archivos y luego con un sistema gestor de bases de datos simple.
- Consumir una **API externa** y manejar errores/red.
- Aplicar **usabilidad** (IHC): wireframes, feedback, estados de error/vacío.
- Colaborar con **Git/GitHub** y **Kanban** (Trello/GitHub Projects).
- Probar y documentar software (**tests**, **README**, manuales).

Requisitos técnicos obligatorios (escalonados y con niveles)

Regla general

Todos los equipos deben alcanzar el **Nivel Estándar** para aprobar sin inconvenientes. El **Nivel Base** garantiza un mínimo funcional acorde al segundo año de la carrera. El **Nivel Avanzado** otorga mérito adicional (bono por criterio).

R1. Arquitectura y stack

Base: Arquitectura por capas (routes → controllers → services → data); Node.js + Express; frontend HTML/CSS/JS.
Estándar: Módulos separados, manejo centralizado de errores/validaciones.
Avanzado: MVC ligero o Server-Side Rendering (SSR); componentes reutilizables y *routing* en FE (enrutamiento en el Front-End).

R2. Persistencia (progresiva)

Base (Hito 1): Lectura/escritura en **JSON/CSV** mediante una capa de datos.
Estándar (Hito 2): Migración a **SQLite** para 1–2 entidades; CRUD real.
Avanzado (Hito 3): 2–3 entidades, relación 1:N o N:M, índices/seeds.

R3. API propia y API externa

Base (Hito 1): "API interna" (servicios) invocada desde controladores sin HTTP.
Estándar (Hito 2): Exponer **REST** con 4–6 endpoints (CRUD + 1 búsqueda/filtrado).
Avanzado (Hito 3): 8–10 endpoints, paginación/ordenamiento y **caché en memoria**.
API externa (mínimo): 1 integración sencilla con manejo de timeout y errores.

R4. Autentificación y seguridad

Base: Registro/login y protección de rutas de escritura (JWT o sesión); variables en `.env`.
Estándar: Validaciones de entrada y **hash** de contraseñas.
Avanzado: Roles simples (2) y prácticas extra (rate limit, CORS afinado, logs).

R5. Testing y calidad

Base: 2 pruebas unitarias (servicios) + 2 pruebas de API (Postman + script).
Estándar: 4–6 pruebas totales; `npm test`; linter básico.
Avanzado: 8+ pruebas; CI (GitHub Actions) ejecutando tests y linter.

R6. Metodología y colaboración

Base: Kanban con *To Do / In Progress / Review / Testing / Done*; PRs con 1 review; roles definidos.
Estándar: 1 retrospectiva por hito; milestones por hito; issues trazables a PRs.
Avanzado: Métricas simples (lead time (Tiempo total que pasa desde que una tarea se crea (se añade al tablero en "Por hacer") hasta que se termina ("Hecho"))/throughput (Número de tareas completadas en un período de tiempo fijo)) reportadas en el Hito 2 (H2) / Hito 3 (H3).

Estructura de los equipos y roles

Rol	Responsabilidades
Líder técnico	Arquitectura, decisiones técnicas, code reviews, integración final
Backend	Servicios, capa de datos, (H2) REST, (H2) SQLite, pruebas de backend
Frontend	Wireframes, UI/UX, estados y validaciones, consumo de servicios/REST
QA & Documentación	Testing, manuales, guías de instalación, tablero Kanban y evidencias

Rotación de roles: cada 6 semanas (o cada 2 sprints).

Metodología de trabajo

Sprints y tablero

- **Sprints de 3 semanas** (6 sprints en total).
- Tablero Kanban mínimo: *To Do / In Progress / Review / Testing / Done*.
- **Retro breve por hito** (1 pág.) y líneas de progreso semanales.
- Protección de `main`, PRs pequeños y al menos **1 review**.

Info obligatoria en cada tarjeta

Título, criterios de aceptación, responsable, estimación (S/M/L/XL), etiquetas (FE/BE/Docs/Test/API), fecha y checklist.

Cronograma detallado de entregables (18 semanas)

Sprint 1: Descubrimiento & diseño (Semanas 1–3)

S1

- Crear repositorio en GitHub con `README.md` inicial (nombre del proyecto y breve descripción).
- Configurar tablero Kanban con columnas: *To Do / In Progress / Review / Testing / Done*.
- Asignar roles y registrar acta de equipo (`docs/acta_equipo.md`) con: nombre del equipo, integrantes, roles, plan de rotación y canal de comunicación oficial.
- Subir primera evidencia: captura y enlace del tablero Kanban y enlace al repositorio.

S2

- Definir requisitos funcionales (RF) y no funcionales (RNF) en `docs/requisitos.md`.
- Crear wireframes (mínimo 3 pantallas clave) y subirlos en formato imagen o PDF a la carpeta `docs/wireframes/`.
- Seleccionar API externa y documentar: nombre, URL, endpoints relevantes y ejemplo de respuesta.

S3

- Definir arquitectura por capas y estructura de carpetas en el repositorio.
- Implementar capa de persistencia inicial (lectura/escritura en archivos JSON/CSV).
- Documentar en `docs/arquitectura.md` el diagrama de capas y explicación breve de cada módulo.

Sprint 2: Prototipo funcional (Semanas 4–6)

S4

- Implementar capa de datos (lectura/escritura JSON/CSV) con datos de prueba reales.
- Asegurar manejo básico de errores (archivo no encontrado, datos inválidos).

S5

- Crear interfaz navegable con HTML/CSS/JS (mínimo: home, listado, formulario).
- Implementar validaciones básicas en frontend (campos obligatorios, formatos).
- Incluir estados visuales para carga, error y datos vacíos.

S6

- Integrar frontend con servicios internos (sin REST todavía).
- Probar flujo completo de inserción y lectura de datos.
- Subir video corto (máx. 2 min) mostrando la navegación y persistencia.

HITO 1 (Semana 7–8)

Hito 1 (30 %)

Presentación: problema, arquitectura por capas, demo con persistencia en archivos, Kanban y organización.
Demo: flujos base, lectura/escritura JSON/CSV, feedback de UI, evidencia de testing mínimo.
Metodología: tablero actualizado, PRs y roles activos.

S8

- Corregir observaciones del hito.
- Planificar conexión a base de datos y API REST.
- Definir script de migración a SQLite.

Sprint 3: Conexión BD & REST (Semanas 9–10)

S9

- Migrar persistencia a SQLite para 1 entidad con CRUD completo.
- Subir script de inicialización en `scripts/init_db.sql`.

S10

- Exponer API REST (4–6 endpoints) con manejo de errores y uso de variables en `.env`.
- Documentar endpoints en `docs/endpoints.md` con ejemplo de petición y respuesta.

Sprint 4: Integración & calidad (Semanas 11–13)

S11

- Integrar API externa en el backend con manejo de timeout y errores.
- Mostrar datos reales en la interfaz (mínimo 1 pantalla con datos de API externa).

S12

- Implementar 4–6 pruebas unitarias/API.
- Configurar y ejecutar linter en el proyecto.
- Actualizar documentación de endpoints.

S13

- **HITO 2 (30 %)**: demo integrada (SQLite + REST + API externa).
- Entregar retro escrita sobre la metodología aplicada hasta ahora.

Sprint 5: Optimización & documentación (Semanas 14–16)

S14

- Implementar búsqueda/filtrado y paginación básica en backend y frontend.
- Mejorar performance en consultas o renderizado.

S15

- Redactar documentación técnica en `docs/instalacion.md`, `docs/usuario.md`, y `docs/endpoints.md`

S16

- Implementar autenticación mínima (registro/login y rutas protegidas).
- Preparar versión candidata (Release Candidate).

Sprint 6: Cierre (Semanas 17–18)

S17

- Cerrar issues y hacer merge final en rama `main`.
- Subir video demo final (5–7 min) mostrando todo el flujo.

S18

- **HITO 3 (40 %)**: presentación final (25 min + 10 min de preguntas).
- Demo end-to-end, defensa individual del aporte y entrega final en GitHub con release etiquetada.

Pauta de evaluación del equipo (ajustada por niveles)

Ponderaciones por Hito

Componente	Peso	Desglose
Hito 1 (Sem. 7–8)	30 %	Presentación (40 %) + Demo (40 %) + Metodología/Kanban (20 %)
Hito 2 (Sem. 13)	30 %	Presentación (40 %) + Demo (40 %) + Metodología/Kanban (20 %)
Hito 3 (Sem. 18)	40 %	Presentación (35 %) + Demo (35 %) + Metodología/Kanban (30 %)

Criterios y niveles (Base / Estándar / Avanzado)

Arquitectura & Código Base: capas claras y manejo básico de errores. Estándar: módulos, validaciones, legibilidad. Avanzado: MVC/MVP ligero, componentes reutilizables.

Persistencia Base: archivos JSON/CSV. Estándar: SQLite para 1–2 entidades (CRUD real). Avanzado: 2–3 entidades, relaciones e índices.

API Propia & Externa Base: servicios internos; integración externa simple. Estándar: REST 4–6 endpoints + manejo de errores. Avanzado: 8–10 endpoints, paginación/ordenamiento y caché.

Autenticación & Seguridad Base: login/registro + rutas protegidas. Estándar: hash de contraseñas, validaciones. Avanzado: roles simples y prácticas extra.

Testing & Calidad Base: 2 unitarias + 2 API. Estándar: 4–6 totales + linter. Avanzado: 8+ y CI.

Metodología & Kanban Base: tablero activo, PRs con review, roles claros. Estándar: retro por hito, milestones, trazabilidad. Avanzado: métricas simples (lead time/throughput) reportadas.

Bono por Avanzado

Cada criterio puede sumar hasta **+10 %** extra del propio criterio (sin exceder 100 % del hito) si el equipo cumple el nivel Avanzado de manera sólida y demostrable.

Metodología de Evaluación Individual

El aporte de cada integrante se medirá combinando métricas objetivas y evidencias cualitativas, registradas durante todo el proyecto. La calificación individual corresponderá al 40 % de la nota final y se calculará así:

1. **Métricas objetivas (50 % de la nota individual):**
   - Actividad en GitHub: commits, pull requests, revisiones de código, issues asignadas y cerradas.
   - Tareas completadas y lead time en el tablero Kanban (Trello o equivalente).
   - Aportes a documentación y bitácoras semanales.
2. **Coevaluación (30 % de la nota individual):** Evaluación entregada por el profesor al final de cada hito, considerando calidad técnica, responsabilidad y colaboración.
3. **Autoevaluación y participación (20 % de la nota individual):** Asistencia a reuniones, cumplimiento de tareas y reflexión semanal sobre avances e inconvenientes encontrados.

La nota final individual se obtendrá mediante:

$$\text{Nota final} = 0{,}6 \times \text{Nota equipo} + 0{,}4 \times \text{Nota individual}$$

Todos los aportes deberán quedar registrados en las herramientas oficiales (GitHub, Trello, actas de equipo) para ser considerados en la evaluación.

Entrega final (detalle del Hito 3)

Evaluación Final (40 % nota final)

Presentación final (25 min + 10 min preguntas):

- Demostración completa end-to-end del sistema.
- Análisis cuantitativo de métricas del proyecto.
- Lecciones aprendidas y decisiones técnicas justificadas.
- Defensa individual del aporte técnico.

Entrega final incluye:

- Código fuente completo en GitHub (tags y releases).
- Documentación técnica (`instalacion.md`, `usuario.md`, `endpoints.md`).
- Sistema ejecutable local; despliegue opcional valorado.
- Suite de testing ejecutándose (comando reproducible).
- Video demo (5–7 min) y materiales de presentación.

Recursos y herramientas recomendadas

Frontend: HTML5, CSS3, JavaScript ES6+, (opcional) React/Vue.
Backend: Node.js + Express; (H2) SQLite con `sqlite3`/`better-sqlite3`.
Testing: Jest/Mocha, Postman/Insomnia; GitHub Actions (opcional).
Gestión: Trello o GitHub Projects, Issues/PRs, Markdown.

Otras consideraciones

- Asistencia a presentaciones: **obligatoria**. Cada integrante debe **defender** su aporte.
- Código en GitHub antes de cada hito;
- Plagio o copia: reprobación inmediata.
- Ausencias justificadas: coordinar con el profesor con suficiente antelación.