PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE
ESCUELA DE INGENIERÍA
DEPARTAMENTO DE CIENCIA DE LA COMPUTACIÓN

IIC2513 — Tecnologías y Aplicaciones Web

Proyecto: Entrega 2

Actualización: 3 de junio de 2025

Entrega

- **Fecha y hora**: Viernes 13 de junio del 2025, a las 22:00
- **Lugar**: Repositorio grupal en la organización del curso en GitHub, rama `main`.

Objetivos

- **Construir** aplicaciones usando las tecnologías y herramientas disponibles.
- **Integrar** técnicas de desarrollo de software para construir aplicaciones web de alta calidad.

Descripción

Para esta entrega, deberán trabajar en la definición e implementación del *back-end* que soporta las funcionalidades clave de su aplicación en desarrollo. Este proceso incluye modelar e implementar la base de datos, diseñar e implementar la API para facilitar la interacción con la capa de persistencia y desarrollar las funcionalidades esenciales de la aplicación web utilizando estos componentes. Los componentes específicos que deben implementar en esta entrega incluyen:

- **Base de datos en PostgreSQL**: Deben diseñar un modelo de base de datos que sea adecuado para el contexto del proyecto. Es necesario que documenten detalladamente en el archivo README cómo configurar y levantar la base de datos para facilitar las pruebas de la aplicación. Para la interacción con la base de datos, pueden utilizar pg, knex o un ORM como Sequelize. El desarrollo de este modelo debe basarse en el diagrama E/R elaborado durante la Entrega 0, incorporando las modificaciones necesarias tras la reunión con el ayudante guía. El diagrama E/R deberá estar actualizado en este repositorio.
- **Manejo de sesión**: se debe implementar un sistema que permita a los usuarios autenticarse de manera segura en la aplicación web. Esto incluye la implementación de un proceso de inicio de sesión que valide las credenciales del usuario, genere y administre tokens de sesión y permita el cierre de sesión de manera eficiente. Para ello, deberán hacer uso de Jason Web Tokens, que pueden manejar utilizando la librería de Koa jsonwebtoken.

  Además, se debe asegurar que las rutas y *endpoints* relevantes estén protegidos de manera adecuada para garantizar la seguridad de los datos y la privacidad de los usuarios. El sistema de manejo de sesión debe ser robusto y confiable, utilizando la librería bcrypt para hashear las contraseñas de los usuarios. Específicamente, es necesario que se utilice salt al hashear las contraseñas para proporcionar una capa adicional de seguridad y proteger contra ataques de diccionario y búsqueda de tablas de hash precalculadas.
- **API RESTful para interactuar con la base de datos**: Su aplicación debe contar al menos dos operaciones CRUD completas para dos entidades distintas. Esto requiere la implementación de cuatro *endpoints* en la API, cada uno destinado a una función específica:

- `POST`
- `GET`
- `PUT/PATCH`
- `DELETE`

El *front-end* de su aplicación deberá integrarse con esta API para asegurar una transferencia de datos sin problemas y mejorar la experiencia de usuario. La API debe ser implementada utilizando Koa o express.js. Es crucial que, al integrar el *front-end* con el *back-end*, presten especial atención al manejo de errores para garantizar una funcionalidad robusta y segura de la página web.

Además, el archivo README debe incluir instrucciones detalladas para instalar las dependencias de la API, con comandos específicos y una explicación breve de su propósito.

- **Documentación de la API**: Es necesario que cada grupo proporcione una documentación completa y detallada para cada *endpoint* de la API. La documentación debe incluir, como mínimo:
  - Método HTTP
  - Ruta del *endpoint*
  - Parámetros que recibe
  - Respuestas del *endpoint*, junto con su formato

  Para la elaboración de esta documentación, se recomienda utilizar herramientas como Swagger o Postman, las cuales facilitan la creación y gestión de documentación interactiva y proporcionan la capacidad de incluir ejemplos prácticos.

- **Uso de Linter**: se espera que utilicen una herramienta de *linting* como ESLint para mantener un código limpio y coherente en todo el proyecto, mejorando así la legibilidad y la calidad del código.

Además de lo descrito anteriormente, cada grupo deberá seguir con el uso apropiado de *GitHub*, incluyendo ramas, PRs y *commits* convencionales.

Bonus

Además de los requisitos mínimos mencionados anteriormente, se les otorgará la oportunidad de mejorar aún más su trabajo. Para incentivar las prácticas de desarrollo de alta calidad, se otorgará 5 décimas de bono a aquellos estudiantes que opten por implementar pruebas unitarias en su aplicación utilizando la librería Jest. Las pruebas unitarias son una herramienta esencial para garantizar la estabilidad y el funcionamiento correcto de su aplicación, y demuestran un compromiso con la excelencia en el desarrollo.

Para conseguir esta bonificación adicional, las pruebas que realicen deberán cubrir un 70 % del código que han realizado. En caso de que no logren alcanzar esta meta, se dará puntaje parcial por llegar a, por lo menos, 50 %.

Entregables

Como equipo, deberán entregar los siguientes componentes de su proyecto, asegurándose de que cada elemento cumpla con los estándares especificados y sea completamente funcional:

- **Implementación del *back-end*** de la aplicación, que debe incluir la base de datos en PostgreSQL y la API en Koa. La API deberá implementar al menos dos operaciones CRUD completas.
- **Implementación del *front-end*** de la aplicación, que debe conectar y utilizar al menos un CRUD de la API.
- **Documentación técnica** que detalle los pasos necesarios para configurar y levantar la aplicación, así como para montar la base de datos.
- **Documentación de uso** de los *endpoints* implementados, proporcionando ejemplos claros de solicitudes y respuestas.
- **Diagrama E/R actualizado** en formato `.pdf`, que refleje los cambios realizados desde la Entrega 0. Este documento debe estar disponible en la carpeta de *assets*.
- **Despliegue en línea**: un requisito **esencial** es que la aplicación debe estar desplegada en línea, accesible a través de un enlace específico y disponible para cualquier usuario; no se aceptarán entregas de proyectos que solo funcionen en un entorno local.

Rúbrica

A continuación, se describe el criterio de cada uno de los ítems de la rúbrica con la que se evaluará esta entrega. La nota se calcula al 50 % del puntaje total.

Nota: Los ítems marcados como **ESENCIAL** son requisitos mínimos indispensables para la entrega. El incumplimiento de estos requisitos limitará la **nota máxima posible a 4,0**. Es decir que, bajo esta condición, un 4,0 se convierte en el puntaje máximo posible y la escala de calificación se ajustará en base a esto. Por ende, si la nota obtenida inicialmente supera el 4,0, esta se ajustará proporcionalmente de acuerdo con la nueva escala máxima establecida.

- **Corrección de comentarios del ayudante [1 punto] [ESENCIAL]**: Es obligatorio que el equipo haya considerado y corregido los comentarios realizados por el ayudante en la entrega anterior. Se espera que las observaciones hayan sido tomada en cuenta y que los aspectos indicados hayan sido corregidos o mejorados.
- **Base de datos [1 punto] [ESENCIAL]**: se espera que se haya implementado una base de datos en PostgreSQL que respalde el funcionamiento de la aplicación.
- ***Deploy* [1 punto] [ESENCIAL]**: La aplicación debe estar desplegada en línea y accesible para cualquier usuario a través de un enlace específico. **No se aceptarán revisiones de proyectos que solo funcionen localmente.**
- **Diagrama E/R [2 puntos] [ESENCIAL]**: Se entrega un diagrama en el formato pedido que tenga todos los componentes necesarios para modelar su aplicación. Estos componentes deben haber sido modificados desde la Entrega 0 para coincidir con las directrices discutidas con el ayudante guía y con la implementación realizada en *Sequelize*. Además, se debe cumplir con una complejidad mínima, la cual debe ser acordada con el ayudante. Los componentes deben estar relacionados correctamente entre sí y tener cardinalidades adecuadas.

- **Conexión con *front-end* [2 puntos] [ESENCIAL]**: Es requisito realizar al menos un CRUD completo mediante *requests* desde el *front-end* hacia los *endpoint* del *back-end*. Las operaciones CRUD pueden incluir diferentes modelos; por ejemplo, realizar un `GET` para obtener datos de usuarios y un `POST` para enviar datos de productos. La información recibida o enviada mediante estas solicitudes debe ser efectivamente utilizada en al menos una de las vistas de la aplicación web.

- **Manejo de sesión en *back-end* [6 puntos] [ESENCIAL]**: se espera que se implemente una forma de manejar la sesión de usuarios en el lado del servidor. Esto incluye la validación de credenciales, el uso y almacenamiento de JWT y la encriptación de contraseñas. Es de suma relevancia que tengan rutas protegidas para que la funcionalidad del manejo de sesión pueda ser evidenciada.

- **Modelo Sequelize [4 puntos]**: La evaluación se centrará en la calidad y coherencia del modelo de datos implementado con Sequelize. Este modelo debe reflejar correctamente la estructura y la lógica de la aplicación, tal como se especificó en el diagrama E/R. Además, es necesario que el modelo se integre de manera efectiva con la base de datos, asegurando que las operaciones CRUD (Crear, Leer, Actualizar, Eliminar) se ejecuten de manera eficiente y segura.

- **API RESTful [5 puntos]**: se implementa una API utilizando Koa. Esta es capaz de realizar al menos dos operaciones CRUD en los datos. Esto implica la creación de cuatro tipos de endpoints (`POST`, `GET`, `PUT/PATCH` y `DELETE`) que permitan a los usuarios realizar estas operaciones de manera efectiva.

- **Documentación API [4 puntos]**: se debe proporcionar una documentación exhaustiva y de calidad para la API de la aplicación. Esta debe incluir información detallada sobre cada uno de los *endpoints* disponibles en la API, lo que incluye el método HTTP utilizado, la ruta del endpoint, los argumentos que se deben proporcionar (junto con su formato), y lo que se puede esperar como respuesta del endpoint (junto con su formato). Se recomienda el uso de herramientas como *Swagger* o *Postman* y la inclusión de ejemplos para ayudar a los ayudantes a comprender cómo interactuar con la API de manera efectiva.

- **Documentación instalación BDD y API [3 puntos]**: se espera que se entregue documentación para guiar a los usuarios en la instalación y configuración del *back-end* de la aplicación. Para la base de datos, se espera que la documentación incluya información sobre cómo configurar y levantar la base de datos, así como los comandos específicos que deben ejecutarse. En el caso de la API, la documentación debe indicar cómo instalar las dependencias necesarias para ejecutarla, proporcionando comandos específicos para ello. Además de como configurar las variables de entorno para poder acceder correctamente a la base de datos.

- **Gitflow [3 puntos]**: se espera que cada grupo siga buenas prácticas de Git y GitHub en su desarrollo, utilizando *Pull Requests* para la revisión de código y la integración de cambios; aplicando *Conventional Commits* para mensajes descriptivos y significativos; y organizando su trabajo en ramas de manera lógica y coherente.

- **Linter [2 puntos]**: se evalúa si se utiliza una herramienta de linting, como *ESLint*, para mantener un código limpio y legible.

Notas

- Se recomienda que cada equipo se ponga en contacto con su ayudante en caso de que requiera aclarar lo que se espera del proyecto en general y de la entrega en particular, además de aclarar detalles de la entrega que no estén claros.

- Posterior a la entrega, cada equipo deberá coordinar una reunión con su ayudante para recibir *feedback* de la entrega. Además, en esa misma instancia, se le pedirá a cada integrante que hable de detalles funcionales y técnicos de lo que entregaron. Si bien puede pasar que se dividan ciertas responsabilidades, se espera que manejen de manera general lo implementado. Es responsabilidad del grupo ponerse en contacto con

el ayudante y gestionar la reunión. El no reunirse con su ayudante antes de que inicie la siguiente entrega, significara un descuento del 50 % sobre la nota obtenida por él o los individuos.

- Durante el proyecto completo, cada grupo puede utilizar 3 cupones de atraso. En caso de no usarse en esta entrega, pueden ser utilizados durante el resto del proyecto. Los días de atraso incluyen fines de semana (utilizar un cupón de atraso un viernes, mueve la entrega al sábado, no al siguiente día hábil).
- En caso de que lo prefieran, se dará la opción de trabajar con Express, en vez de Koa. Sin embargo, es de suma importancia recordar que, si se elige usar este *framework*, entonces no podrán pedirle ayuda técnica a su ayudante guía.