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

IIC2513 — Tecnologías y Aplicaciones Web

Proyecto: Entrega 2

Actualización: 12 de octubre de 2023

Entrega

- **Fecha y hora**: Viernes 27 de octubre del 2023, a las 22:00
- **Lugar:** Repositorio grupal en la organizacion del curso en GitHub

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 *backend* que da soporte a las funcionalidades de su aplicación en desarrollo. Esto implica modelar e implementar la base de datos, diseñar e implementar la API que permite la interacción con la capa de persistencia, y desarrollar las funcionalidades de la aplicación web utilizando los elementos anteriores. En específico, los componentes a implementar en esta entrega son:

- **Base de datos en PostgreSQL**: se debe diseñar un modelo apropiado para el contexto, además de documentar como levantar la base de datos para probar la aplicación en su README. Para interactuar con la base de datos, se deberá utilizar **Sequelize**. Para desarrollar este modelo, deberán usar el diagrama E/R que realizaron en la Entrega 0, con los cambios apropiados luego de la reunión con su ayudante guía. Este diagrama (actualizado) también tendrá que estar subido en su repositorio.
- **API RESTful** para interactuar con la base de datos: la aplicación debe permitir a los usuarios realizar a lo menos una operación CRUD (Crear, Leer, Actualizar y Eliminar) en los datos. Esto implica crear cuatro *endpoints* que tengan la capacidad de:
  - `POST`: agregar nuevos registros.
  - `GET`: visualizar datos existentes.
  - `PUT/PATCH`: actualizar información en la base de datos.
  - `DELETE`: eliminar registros de manera segura.

  El *frontend* de su aplicación deberá conectarse a este servicio para garantizar la transferencia de datos sin problemas y una experiencia de usuario fluida. Esta API deberá ser implementada en **Koa**. Recuerden que, al conectar estas dos partes, es de suma importancia el manejo de errores para garantizar un uso apropiado de la página web que se está creando.

  Además, deberán incluir en el README las instrucciones para instalar dependencias de la API, indicando comandos a ejecutar y una breve descripción de lo que hacen.

- **Documentación de la API**: cada grupo deberá proporcionar una documentación detallada, indicando (como mínimo) para cada *endpoint*:

  - Método HTTP a utilizar.
  - Ruta del *endpoint*.
  - Argumentos que recibe (y su formato).
  - Lo que retorna este *endpoint* (y su formato).

  Se recomienda el uso de Swagger o Postman para documentar la API e incluir ejemplos en la documentación. En el repositorio del curso podrán encontrar una cápsula con un ejemplo de cómo usar Postman.

- **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.

Entregables

Como equipo, deberán entregar:

- Implementacion del *backend* de la aplicación, incluyendo la base de datos en PostgreSQL y la API en Koa con al menos un CRUD.
- Implementación del *frontend* de la aplicación con por lo menos un endpoint conectado y en uso.
- Documentación para levantar la aplicación y montar la base de datos.
- Documentación de uso de los endpoints implementados.
- Diagrama E/R en formato `.pdf` con las actualizaciones apropiadas realizadas desde la E0.

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 % considerando un total de 20 puntos.

Nota: Los ítems marcados como ESENCIAL son requisitos mínimos indispensables para la entrega. No cumplirlo, reducirá la **nota máxima a 4,0**. Es muy importante recordar que esto significa que ahora el 4,0 equivale al puntaje máximo y la escala se ajusta acorde a esto. Por lo tanto, si originalmente obtuvieron una nota mayor a 4,0, no se convertirá en un 4,0, sino que se calculará proporcionalmente en función de esta nueva escala.

- **Base de datos [1 puntos] (ESENCIAL)**: se espera que se haya implementado una base de datos en PostgreSQL que respalde el funcionamiento de la aplicación.
- **Diagrama E/R [2 puntos]**: se entrega un diagrama en el formato pedido que tenga todos los componentes necesarios para modelar su aplicación. Estos deben haber sido modificados desde la Entrega 0 para concidir con lo hablado con el ayudante guía y con la implementación realizada en Sequelize. Los componentes deben estar relacionados correctamente entre sí y tener cardinalidades adecuadas.
- **Modelo Sequelize [4 puntos]**: se evalúa la calidad y coherencia del modelo de datos implementado. Este debe reflejar adecuadamente la estructura de la base de datos y la lógica de la aplicación como fue descrito en el diagrama E/R. Además, debe integrarse de manera adecuada con la base de datos para garantizar que las operaciones CRUD se puedan realizar de manera eficiente y segura.
- **API RESTful [3 puntos]**: se implementa una API utilizando Koa. Esta es capaz de realizar al menos una operación 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.
- **Conexión con frontend [2 puntos] (ESENCIAL)**: se debe realizar a lo menos una *request* desde el frontend hacia un *endpoint* del backend. La información obtenida de dicha solicitud debe ser utilizada en alguna de las vistas de su aplicación web.
- **Documentación API [2 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 [2 puntos]**: se espera que se entregue documentación para guiar a los usuarios en la instalación y configuración del backend 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.
- **Gitflow [2 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).
- Para esta entrega, tienen permitido el uso de bibliotecas de componentes HTML/CSS/JS como Bootstrap o Zurb Foundation y de preprocesadores de CSS como EJS, SASS/SCSS, Less y Stylus.
- 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.