Proyecto Parte 2: Aplicación web moderna

Objetivos

Objetivo general

Construir una **aplicación web moderna** que incluya dos componentes: una **API RESTful** como backend, y un **single page application (SPA)** como frontend. Ambos componentes deben estar integrados de tal forma de incluir las funcionalidades y objetivos generales y específicos, correspondientes a la idea de proyecto asignada a su equipo de trabajo.

Objetivos específicos para toda idea de proyecto

1. Implementar una API RESTful, usando el framework Koa junto con el template del curso. Cada equipo de trabajo deberá diseñar la especificación de su API de acuerdo a la idea de proyecto asignada
2. Documentar la API RESTful implementada. Se recomienda usar OpenAPI con SwaggerUI o Postman
3. Diseñar la interfaz de usuario (frontend en browser) **desde cero**, utilizando como base para su diseño el realizado en la Parte 1 del proyecto junto a la especificación de la API RESTful de los puntos precedentes (1 y 2)
4. Implementar una single page application (SPA) en React, que se conecte a la API RESTful descrita anteriormente, consumiendo los endpoints correspondientes
5. Incorporar aspectos de calidad en flujo de desarrollo: testing (Jest), análisis estático de código y *code review*
6. Implementar medidas de seguridad de datos y ante vulnerabilidades en la Web

Contexto

Este proyecto busca acercarlos al estado del arte del desarrollo web, donde podemos entender la Web como un conjunto de servicios que se exponen y se consumen entre sí. Hoy en día la separación entre tecnologías frontend y backend es clara, al punto de poder desarrollarse como mundos independientes (aunque comunicados) entre sí, lo que se conoce como una aplicación Web moderna.

En este proyecto los estudiantes continuarán con la idea de proyecto asignada al inicio del semestre, extendiendo su implementación con el desarrollo de dos aplicaciones: una del lado del servidor que consistirá en una API RESTful, y otra del lado del cliente que consistirá en una single page application para browsers utilizando React.

Criterios de diseño

Los equipos de trabajo deben proponer y generar una solución de acuerdo a la idea de proyecto que les fue asignada. Esta solución **debe** cumplir con los criterios descritos a continuación:

1. La solución propuesta debe poder mapearse a un conjunto de endpoints de una API RESTful, el cual deberá ser diseñado e implementado por el equipo de trabajo, tomando en cuenta las consideraciones vistas en cátedras y ayudantías
2. La solución propuesta debe poder implementarse mediante un frontend moderno en React, que incluya algunos aspectos interesantes de interacción y experiencia de usuario (diseño reactivo, popups, menús laterales, etc)
3. El frontend de la solución propuesta debe poder conectarse de forma simple a la API RESTful asociada, lo que en general involucra un mapping 1:1 entre una vista frontend y un API endpoint (por ejemplo, una lista de usuarios en general significa consumir un endpoint que entrega esa lista)
4. El frontend de la solución puede necesitar consumir más de un endpoint para una determinada vista en el frontend, siempre que sea algo que apoye el correcto funcionamiento y visualización de la información
5. El desarrollo de la solución debe incluir buenas prácticas de diseño, testing, seguridad, control de código (flujo de desarrollo en Github), estilo de código y documentación

Entregas

Esta segunda y última parte del proyecto se compone de **2 entregas**. Las entregas estarán separadas una de otra por un intervalo cercano a 3 semanas.

Fechas de entregas

- **Entrega 3:** 11 de noviembre
- **Entrega 4**: 06 de diciembre

Forma de entrega

Dado que para este proyecto los grupos de trabajo desarrollarán dos aplicaciones web (una frontend y otra backend), el código será entregado a través de **dos repositorios Git alojados en Github** y asociados a su grupo: uno para la aplicación backend (el mismo utilizando en la Parte 1) y otro para la aplicación frontend (un nuevo repositorio). El equipo docente les proveerá a los estudiantes repositorios gratuitos y privados para cada equipo.

Los ayudantes revisarán el branch main únicamente, en ambos repositorios. Deben usar otras ramas durante el desarrollo y realizar merge a main sólo de lo que están seguros de querer entregar. Al igual que para la Parte 1, tendrán que seguir un flujo definido para hacer merge a main, mediante el uso de Pull Requests.

Los estudiantes pueden incluir un archivo README en el repositorio del proyecto que corresponda, con cualquier consideración, supuesto, decisiones de diseño u otra

información relevante que pueda ser importante para el ayudante que los evaluará. Este README puede ser actualizado con cada entrega.

Para alojar sus aplicaciones en la Web, las plataformas soportadas y recomendadas dependen del tipo de aplicación:

- Heroku para backend
- Netlify para frontend

Si los estudiantes desean utilizar otro proveedor pueden hacerlo, pero no tendrá soporte por parte del equipo docente y, por lo tanto, su uso estará bajo la responsabilidad del equipo de trabajo.

El resultado final esperado es que realicen un deploy real de su aplicación en modo producción y que esta quede disponible para cualquier usuario de la Web, es decir, que sea pública. **No cumplir con este requisito es equivalente a no haber entregado su trabajo, lo cual se traducirá en un 1.0 como calificación**.

El avance logrado en cada entrega se evaluará en base a **lo subido al branch main del repositorio Github correspondiente en la fecha y hora límite de la entrega**, y a **lo mostrado en Heroku/Netlify** en el momento de la reunión con el ayudante.

Se verificará que el último commit en Heroku/Netlify (o la plataforma que hayan seleccionado) corresponda al último commit en Github antes de la fecha y hora límite.

Como verán más adelante, para la Entrega 3 sólo trabajarán con la aplicación backend, por lo que sólo deberán dejar su aplicación disponible en Heroku. Para la Entrega 4, el foco será la aplicación frontend (React), que deberán dejar disponible en Netlify, y mantener la aplicación backend disponible en Heroku.

Entrega 3

Objetivo: Implementación API RESTful

Fecha de entrega: 11 de noviembre a las 22:00 horas

En esta entrega los estudiantes deben generar **una API RESTful** asociada a su idea de proyecto, **extendiendo la aplicación backend** que ya hace uso del template del curso.

Los requerimientos generales son los siguientes:

1. Implementar una API RESTful que incluya **todos los endpoints que permitan acceder a la funcionalidad de la aplicación** de la idea de proyecto asignada. Como punto de partida, es recomendable diseñar la estructura base de los endpoints y su comportamiento (request payload cuando corresponda, response status codes y payload, entre otros), considerando lo visto en cátedras, ayudantías y cápsulas
2. Validaciones de servidor asociadas a datos y acceso de usuarios (manejo de permisos), donde sea pertinente. Esto involucra retornar los errores con mensajes y status codes adecuados
3. Tests de rutas de los endpoints de la API RESTful (responses con snapshots)
4. Documentación online de los endpoints de la API RESTful
5. Utilización de buenas prácticas de diseño, implementación y codificación:
    a. Reutilizar lógica a través de middlewares
    b. Usar Sequelize API para consultas a la BD por sobre SQL (a menos que no se pueda o SQL sea mucho más eficiente)
    c. Validaciones bien definidas en los modelos
    d. Construir las rutas o URL de links usando ctx.router.url o similar (en lugar de hardcoded)
    e. Uso de modularización y evitar escribir funciones gigantes en las rutas de koa-router ("fat models / skinny controllers")
    f. Interfaz uniforme de endpoints, según arquitectura REST
    g. Respetar separación de responsabilidades según MVC
    h. Guía de estilos JavaScript de Airbnb (por medio de linter)

Para esta entrega los estudiantes deben utilizar el mismo repositorio asignado para la Parte 1 del proyecto, dado que extenderán lo ya implementado hasta el momento.

Cualquier duda general que tengan los estudiantes sobre el trabajo a realizar puede ser consultada en el foro del curso. Cualquier duda particular sobre su proyecto, que no pueda servir a otros grupos, pueden consultarla directamente a sus ayudantes. Se hace hincapié en que **toda duda la expresen temprana y oportunamente**, pues es muy difícil tratar de manera adecuada issues o preguntas con plazos muy cercanos a la fecha de entrega.

Finalmente, es recomendable que trabajen en grupo y no en forma individual repartiéndose el trabajo, pues de esta manera aprenderán parte importante de cómo construir una API RESTful. Sin embargo, se tomará en consideración que **todos los integrantes del equipo de trabajo tengan una participación similar en base a commits en el repositorio** (en caso contrario, existirá penalización que podría llegar a la nota mínima para algún integrante).

Entrega 4

Objetivo: Implementación single page application (SPA) y lanzamiento aplicación

Fecha de entrega: 06 de diciembre a las 22:00 horas

En esta última entrega del proyecto los equipos de trabajo deben **desarrollar una single page application en React**, e integrarla con su API RESTful para **obtener una única plataforma web** de cara al usuario final. Esto significa que deben **finalizar ambas aplicaciones web (backend y frontend)** en cuanto a funcionalidades. Además, se evaluarán aspectos de seguridad web específicos que debe cumplir su plataforma.

Los requerimientos generales son los siguientes:

1. Implementar una single page application (SPA), incluyendo conexión a la API, aspectos visuales y de UX. Esto implica cuidar la presentación, pulcritud, facilidad de uso, navegación intuitiva, entre otros aspectos.
2. La SPA debe incluir la utilización de rutas en el cliente (utilizando alguna librería como react-router) y validación de datos en el cliente
3. Terminar cualquier detalle de la API RESTful que haya quedado pendiente.
4. Cualquier funcionalidad extra que los estudiantes decidan implementar siempre que no altere la funcionalidad requerida para la idea de proyecto asignada, y mantenga una coherencia con esta
5. Considerar todos los aspectos acordados con su ayudante para el producto final
6. Testing de frontend utilizando las funcionalidades de Jest que permiten realizar testing en React. Las métricas asociadas a este punto serán entregadas junto con la evaluación de la Entrega 3
7. Documentación final que incluya el modelo de datos, links a producción del producto final entregado, guías de instalación y documentación de su API
8. Utilización de buenas prácticas de diseño, implementación y codificación
    a. Composición de componentes utilizando sub-componentes que sigan el principio de responsabilidad única
    b. Creación de componentes y hooks que puedan ser reutilizados para evitar código duplicado
    c. Uso de PropTypes en componentes para minimizar errores de tipos de datos
    d. No usar Inline CSS
    e. Uso correcto de etiquetas HTML (mejor si incluye HTML semántico). No utilizar HTML para estilo ni posicionamiento (por ejemplo, tablas para layout)

Los requerimientos de seguridad web son los siguientes:

- Mitigación de vulnerabilidades web comunes
    - SQL Injection (en casos en que se permita input de usuario)
- Seguridad de datos
    - Uso de variables de entorno
    - Hashing de contraseñas (bcrypt)
    - Uso de JWT para autenticación

Para esta entrega, en cuanto al backend los estudiantes deben utilizar el mismo repositorio de las entregas anteriores. **Para el frontend, su ayudante les indicará un repositorio nuevo.**

Deben considerar, además, que **esta entrega equivale al lanzamiento de su plataforma web**, por lo que deben asegurarse de verificar detalles, como por ejemplo, incorporar datos razonables y realistas.

Cualquier duda general que tengan los estudiantes sobre el trabajo a realizar puede ser consultada en el foro del curso. Cualquier duda particular sobre su proyecto, que no pueda servir a otros grupos, pueden consultarla directamente a sus ayudantes. Se hace hincapié en que **toda duda la expresen temprana y oportunamente**, pues es muy difícil tratar de manera adecuada issues o preguntas con plazos muy cercanos a la fecha de entrega.

Finalmente, es recomendable que trabajen en grupo y no en forma individual repartiéndose el trabajo, pues de esta manera aprenderán todo el proceso de crear una aplicación web en el cliente, y cómo lanzar una plataforma web a producción. Sin embargo, al igual que para las entregas anteriores, se tomará en consideración que **todos los integrantes del equipo de trabajo tengan commits en el repositorio** (en caso contrario, existirá penalización que podría llegar a la nota mínima para algún integrante).

Requisitos de calidad de software

Para todas las entregas los grupos de trabajo deberán cumplir con los requisitos de calidad de software indicados a continuación.

Metodología de *Code Reviews* utilizando *Pull Requests*

Los repositorios de los estudiantes estarán configurados de tal forma que sólo puedan hacer merge a main a través de la interfaz de *Pull Requests* (no será posible realizar merge directos por consola). Además se requerirá la revisión de código y aprobación de al menos un estudiante del grupo para hacer merge a main.

Análisis estático de código a través de ESLint

Los repositorios de los estudiantes estarán configurados para ejecutar ESLint cada vez que se haga un commit asociado a un Pull Request. Si el *check* no es exitoso, cualquier Pull Request que incluya ese commit no podrá ser *merged* en main.

Testing utilizando Jest

Los repositorios de los estudiantes estarán configurados para ejecutar Jest cada vez que se haga un commit asociado a un Pull Request. Si el *check* no es exitoso, cualquier Pull Request que incluya ese commit no podrá ser *merged* en main. Los tests que deben incluir se encuentran en el detalle de cada entrega.

Evaluación

Tal como está indicado en el programa del curso, cada entrega de proyecto será evaluada con una nota de 1.0 a 7.0 siguiendo una escala discreta que se detalla a continuación:

Calificación	Nota asociada	Observaciones
A	7.0	Trabajo sobresaliente. La entrega excede con creces lo solicitado.
A-	6.0	Muy buen trabajo. La entrega cumple con más de lo solicitado.
B	5.5	Buen trabajo. La entrega cumple con lo mínimo solicitado.
C	4.0	Regular. La entrega cumple parcialmente con lo mínimo solicitado.
D	2.5	Deficiente. La entrega está muy por debajo del mínimo solicitado.

En el caso de que un grupo no entregue nada, o esté fuera de plazo (habiendo utilizado la totalidad de sus cupones de atraso), la nota será de un 1.0.

Además, posterior a la calificación de cada entrega grupal, los estudiantes integrantes del grupo deberán calificarse entre sí en una **evaluación de pares**. Pueden revisar los detalles de cómo se llevará a cabo esta evaluación en el programa del curso.

Para evaluar cada entrega, todos los miembros del equipo tendrán una **reunión (virtual o presencial) de corrección** con el ayudante que se les haya asignado. La no asistencia (salvo justificación de peso) de alguno de los integrantes implicará un **descuento de 0.2** para ese estudiante **en el ponderador por evaluación de pares** en esa entrega (no 2 décimas).

Finalmente, las notas de las entregas parciales del proyecto (luego de evaluación de pares y considerando las partes 1 y 2) serán promediadas y eso generará la calificación final del trabajo práctico, denominada **Nota Práctica (NP)**.

Consideraciones y restricciones

- La aplicación web de backend debe ser una extensión de **la misma utilizada en la Parte 1 del proyecto**, es decir, la que fue implementada en **koa utilizando el template del curso**.
- La aplicación web de frontend debe desarrollarse con **React utilizando create-react-app**.
- Podrán usar SASS/SCSS, less, CSS in JS u otros preprocesadores o herramientas de CSS.

- Se permitirá el uso de librerías y/o frameworks CSS, siempre que sólo incorporen CSS y no tengan una contraparte en JavaScript. De todas formas, si trabajaron el estilo de su aplicación para la Parte 1 y llegaron al resultado esperado, no debieran tener que realizar mayores cambios para esta parte del proyecto.
- NO está permitido el uso de librerías de componentes UI para React como Material UI, React Bootstrap, Ant Design, entre otros.
- NO está permitido el uso de lenguajes que compilen a JavaScript, como TypeScript o CoffeeScript.
- **Cualquier *package* o librería adicional que quieran usar debe ser aprobado por el equipo docente. Pregunten a través del foro del proyecto si tienen intención de utilizar alguna**.
- Deberán usar Jest para los *tests* de las aplicaciones
- Es recomendable escribir código en inglés. Sacarán el máximo provecho de las convenciones que puedan haber y el código quedará en armonía con el lenguaje de programación que usen (que también estará en inglés).
- La interfaz de la aplicación web puede estar en inglés o en español.