# Proyecto 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 comunes y específicos, correspondientes a una de las ideas de proyecto disponibles y que será asignada a su equipo de trabajo.

### Objetivos específicos para toda idea de proyecto

1. Implementar una API RESTful, usando el framework Koa con el template del curso. La especificación de la API estará definida previamente y le será entregada a cada equipo de trabajo
2. Documentar la API RESTful implementada. Se recomienda usar OpenAPI con SwaggerUI
3. Diseñar la interfaz de usuario (frontend en browser) desde cero, utilizando como input la especificación de la API RESTful
4. Implementar una single page application (SPA) en React, que se conecte a la API RESTful descrita anteriormente, consumiendo los endpoints correspondientes
5. Incorporar scripts de pruebas unitarias, de integración y de sistema
6. Implementar medidas de seguridad ante vulnerabilidades web más comunes

## 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. Por esa razón, en este proyecto los estudiantes desarrollarán 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 en el browser utilizando para ello React.

## Ideas de proyecto

Para este proyecto existen 5 ideas disponibles, que fueron seleccionadas a partir de ideas propuestas por ustedes mismos. A cada equipo de trabajo se le asignará una de estas ideas para que la desarrollen durante lo que queda del semestre. Si bien tendrán la posibilidad de comunicar sus preferencias, que por supuesto serán consideradas, las ideas de proyecto serán distribuidas de forma equitativa entre todos los grupos de trabajo de ambas secciones.

Las 5 ideas de proyecto seleccionadas son:

1. El negocio de la esquina
2. ReservOK!
3. FindHomy
4. Bookkers
5. SalApp

El detalle de cada idea de proyecto lo podrán encontrar en el documento independiente enlazado en cada uno de los puntos anteriores.

## Criterios de diseño

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

1. La solución propuesta debe ser adecuada en interés y complejidad para el contexto de una aplicación web (i.e. seguir un diseño de estilo MVC accesible desde un browser)
2. La solución propuesta debe incluir un flujo de navegación que el usuario pueda seguir (o al menos debe poder extraerse a partir de lo propuesto), que incluya diferentes vistas (que se traducirán en rutas)
3. La solución propuesta debe poder mapearse a un conjunto de endpoints de una API REST, parte de los cuales se encuentran detallados en el documento correspondiente a cada idea de proyecto
4. 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)
5. El frontend de la solución propuesta debe poder conectarse de forma simple a la API REST 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)
6. Puede necesitar consumirse 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
7. 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

Este proyecto se compone de **3 entregas**. Si bien todas las entregas serán evaluadas, sólo la primera y la última tendrán una nota asociada. Las entregas estarán separadas una de otra por un intervalo cercano a 2 semanas.

### Fechas de entregas

- **Entrega 2.1:** 14 de junio
- **Entrega 2.2**: 25 de junio (evaluada pero sin nota)
- **Entrega 2.3**: 07 de julio

## 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 y otro para la aplicación frontend. 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 el Proyecto 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 toda consideración, supuesto, decisiones de diseño y otra información relevante que pueda aclarar las razones de su desarrollo. 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 será bajo su propia responsabilidad.

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 corresponda al último commit en Github antes de la

fecha y hora límite.

Como verán más adelante, para la primera entrega sólo trabajarán con la aplicación backend, por lo que sólo deberán dejar su aplicación disponible en Heroku. Para las entregas restantes, la aplicación backend deberá estar disponible en Heroku y la aplicación frontend en Netlify.

# Especificación de cada entrega

## Entrega 2.1

### Objetivo: Implementación API RESTful

**Fecha de entrega:** 14 de junio a las 22:00 horas

En esta entrega los estudiantes deben focalizar sus esfuerzos en construir la aplicación backend, generando **la primera versión funcional de la API RESTful** asociada a su idea de proyecto, **utilizando el template del curso**.

Los requerimientos generales son los siguientes:

1. Implementar el grueso de la API RESTful, que contempla **los endpoints marcados como principales** en el detalle que se encuentra en el documento de la idea de proyecto respectiva. Sólo deben considerar el happy path en esta etapa
2. Se espera que se entregue el diseño, la estructura base y su comportamiento (request payload cuando corresponda, response status codes y payload), entre otros
3. El criterio base es que se pueda consumir sin considerar mayormente temas de alta seguridad (a excepción de inicio de sesión) o manejo de errores o validaciones muy específicas
4. Tests de rutas de los endpoints de la API implementados
5. Documentación de los endpoints de la API implementados
6. Utilización de buenas prácticas de diseño, implementación y codificación

Al menos toda la parte de manejo de usuarios y del recurso principal debe estar implementada (fijarse en endpoints marcados como principales). Pueden quedar algunos endpoints secundarios pendientes para las siguientes entregas.

Cualquier duda general que tengan los estudiantes sobre el trabajo a realizar puede ser consultada en el foro de proyectos del curso. Cualquier duda particular sobre su proyecto, que no pueda servir a otros grupos, pueden hacerla a sus ayudantes. Se hace hincapié en que toda duda la expresan 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 commits en el repositorio** (en caso contrario, existirá penalización que podría llegar a la nota mínima para algún integrante).

# Entrega 2.2 (evaluada pero sin nota)

## Objetivo: Comenzando a implementar una single page application (SPA)

**Fecha de entrega**: 25 de junio a las 22:00 horas

En esta segunda entrega el equipo de trabajo deberá finalizar todos los aspectos que hayan quedado pendientes de su API RESTful y **comenzar el desarrollo de su single page application en React**.

Los requerimientos generales son los siguientes:

1. Terminar de implementar **completamente** la API RESTful, incluyendo todos los endpoints especificados en el documento de la idea de proyecto respectiva
2. Validaciones de servidor asociadas a datos y acceso de usuarios, donde sea pertinente. Esto involucra retornar los errores con mensajes y status codes correspondientes
3. Tests de rutas de los nuevos endpoints de la API implementados
4. Documentación de los nuevos endpoints de la API implementados
5. Comenzar la implementación de la single page application (SPA), al menos a nivel de un esqueleto visual, con la navegación implementada de forma básica. Se les sugiere pre-llenar las vistas con datos estáticos
6. Deben implementar rutas en el frontend (utilizando alguna librería como react-router)
7. Si bien su SPA no tendrá mucha lógica aún, es una buena oportunidad para incorporar aspectos de presentación (CSS), por lo que la recomendación es enfocarse en avanzar lo más posible en este punto.
8. Documento de uso de su aplicación (manual de uso de usuario, puede ser un documento o puede ser online junto con la misma aplicación).
9. Utilización de buenas prácticas de diseño, implementación y codificación.

Para esta entrega, en cuanto al backend los estudiantes deben utilizar el mismo repositorio indicado por su ayudante para la entrega 2.1. **Para el frontend, su ayudante les indicará un repositorio nuevo.**

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, pueden consultarla directamente a su ayudante. Se hace hincapié en que **toda duda la expresan 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 inicial de crear una aplicación web en el cliente. Sin embargo, al igual que para la Entrega 2.1, 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).

## Entrega 2.3

### Objetivo: Finalizar completamente su proyecto: backend, frontend y aspectos de seguridad

**Fecha de entrega**: 07 de julio a las 22:00 horas

En esta última entrega del proyecto los equipos de trabajo deben **finalizar ambas aplicaciones web (backend y frontend)** en cuanto a funcionalidades, e integrarlas entre sí para **obtener una única la plataforma web** de cara al usuario final. Además, se evaluarán aspectos de seguridad web específicos que debe cumplir su plataforma.

Los requerimientos generales son los siguientes:

1. Terminar de implementar completamente la SPA, incluyendo conexión a la API, aspectos visuales y de UX, lo cual implica cuidar la presentación, pulcritud, facilidad de uso, navegación intuitiva, entre otros aspectos.
2. Terminar cualquier detalle de la API RESTful que haya quedado pendiente
3. 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. No olviden usar la guía de sus ayudantes
4. Testing de frontend utilizando las funcionalidades de Jest que permiten realizar testing en React, así como sobre el DOM. Las métricas asociadas a este punto serán entregadas junto con la evaluación de la Entrega 2.2
5. Pruebas de aceptación que deben abarcar las funcionalidades asociadas al caso de uso de un usuario de su aplicación. Deberán **cumplir con el plan de pruebas** entregado por su ayudante (el que abarca qué y cómo se probará su aplicación, para que esta sea aceptada desde el punto de vista de un usuario final)
6. Artefacto final que incluye el documento de diseño, detalle de pruebas de aceptación ejecutadas, manual de uso, producto final entregado, guías de instalación y documentación de su APIs
7. Para esta entrega deben considerar todos los aspectos acordados con su ayudante para el producto final, incluyendo el plan de pruebas acordado

Los requerimientos de seguridad web son los siguientes:

- Mitigación de vulnerabilidades web comunes
    - SQL Injection
    - XSS
    - CSRF
- Seguridad de datos
    - Uso de variables de entorno
    - Hashing de contraseñas (bcrypt)
    - Uso de JWT para autenticación
- Análisis de logs de aplicaciones web utilizando librería REmatch
    - Al cumplir con esto podrán optar a un bonus para la nota del proyecto 2, que varía entre 1 y 3 décimas de acuerdo al grado de completitud logrado

Para esta entrega, los estudiantes deben utilizar los mismos repositorios (frontend y backend) que han utilizado para las entregas anteriores.

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, pueden consultarla directamente a su ayudante.

Finalmente, es recomendable que trabajen en grupo y no en forma individual repartiéndose el trabajo, pues de esta manera aprenderán todo el proceso final de 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. Deben incluir tests de lo siguiente:

- Rutas (status codes, retorno y efectos asociados)
- Modelos (sólo métodos que no sean los típicos finders)
- Middlewares interesantes de su aplicación

- Cualquier función o helper que incluya su aplicación
- Frontend en DOM y React

# 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 | Muy buen trabajo. La entrega excede con creces 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 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 entregas 2.1 y 2.3 (luego de evaluación de pares) serán **promediadas** y eso generará la calificación de este segundo proyecto (P2). Recuerden que este proyecto tiene una ponderación del 60% en la Nota Práctica (NP).

# Consideraciones y restricciones

- La aplicación web de backend debe desarrollarse en Node.js y **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.

- NO está permitido el uso de lenguajes que compilen a JavaScript, como CoffeeScript o TypeScript.
- **Cualquier *package* o librería adicional que quieran usar debe ser aprobado por el equipo docente. Pregunten a través del foro del curso 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.