# Sprint 1

Bienvenidos, ha llegado el momento de arrancar con **¡el proyecto integrador!**

## > Requisitos

1. **Grupos definidos:** Para poder iniciar el trabajo integrador, deberá estar conformado el grupo de trabajo. El mismo deberá ser como máximo de tres integrantes.
2. **Cuenta de GitHub:** Todas las entregas se harán, sin excepción, en repositorios públicos de GitHub. Todo el grupo deberá contribuir, sin excepción, al mismo repositorio.

## > Objetivo

Nuestro cliente **DH Venture Capitals** ha puesto en sus manos el desarrollo de un **e-commerce** para poner a prueba el desempeño de nuestro equipo al trabajar con **Node.js y React**.

El cliente solicita que sean ustedes quienes elijan la temática del sitio
y que presenten un boceto gráfico de las principales secciones a implementar, antes de comenzar con el proceso de desarrollo.

## > Metodología

Cómo quizás ya lo imaginan, nuestro cliente pide que, a la hora de desarrollar, se implemente un sistema de trabajo basado en las **metodologías ágiles**.

El desarrollo se dividirá en iteraciones, o hablando en el lenguaje ágil: **sprints**. Este sprint será el primero de ocho que abarcarán la totalidad de las temáticas del curso.


**Su rol será el de desarrollar el sitio asegurándose de que:**

- Sea estéticamente agradable (UI - User Interface).
- Sea fácil de usar (UX - User Experience).
- Sea funcional y cumpla con los requisitos del cliente.
- Las entregas se cumplan en tiempo y forma.

**Nuestro rol como product owners será el de mediar entre ustedes y el cliente asegurándonos que:**

- Los requerimientos sean claros.
- Tengan los conocimientos adecuados en cada etapa.
- Se destraben las situaciones que detengan el desarrollo.
- El código producido sea mantenible y comprensible.

## > Consignas

Durante este sprint, trabajarán en definir la temática del sitio y, luego, producirán un boceto gráfico que les guiará durante todas las etapas del desarrollo.

No se dejen engañar por la simpleza de las consignas, **la etapa de planificación e investigación es de las más importantes de cualquier proyecto**. De hecho, la gran mayoría de los proyectos que fallan, lo hacen por no haber hecho bien esta tarea.

### 1. Crear el repositorio del proyecto y agregar colaboradores

El nombre del repositorio será **grupo_#_nombre** donde # será su número de grupo y **nombre** será el nombre del proyecto.

Deberán agregar como colaboradores al resto de los integrantes del grupo.

**Entregable:** URL del repositorio.

---

### 2. Definir la temática del Market Place

¿Qué productos o servicios brindará nuestro sitio? ¿Quién será nuestra audiencia objetivo? ¿Cómo ajustaremos nuestra oferta a ese público?

**Entregable:** Crear archivo README.md en el repositiorio con:

- Una breve descripción de la oferta de productos y/o servicios ofrecidos por su sitio. También agregar una breve descripción del público al que apunta el sitio.
- Una breve descripción de los integrantes del equipo.

---

### 3. Buscar inspiración en referentes del mercado

Pablo Picasso dijo alguna vez: “Los grandes artistas copian, los genios roban”. Sea cual sea la idea que tengan para su sitio, es muy posible que haya en Internet muy buenos referentes de donde inspirarse y tomar ideas.

Deberán realizar una búsqueda de sitios similares, relacionados o que sean de interés:

- Por los productos o servicios que ofrecen.
- Por los clientes a los que apuntan.
- Por la estética que presentan.
- Por las funcionalidades implementadas.

**Entregable:** En el README.md agregar un listado de al menos 5 sitios de referencia indicando brevemente por qué fueron elegidos.

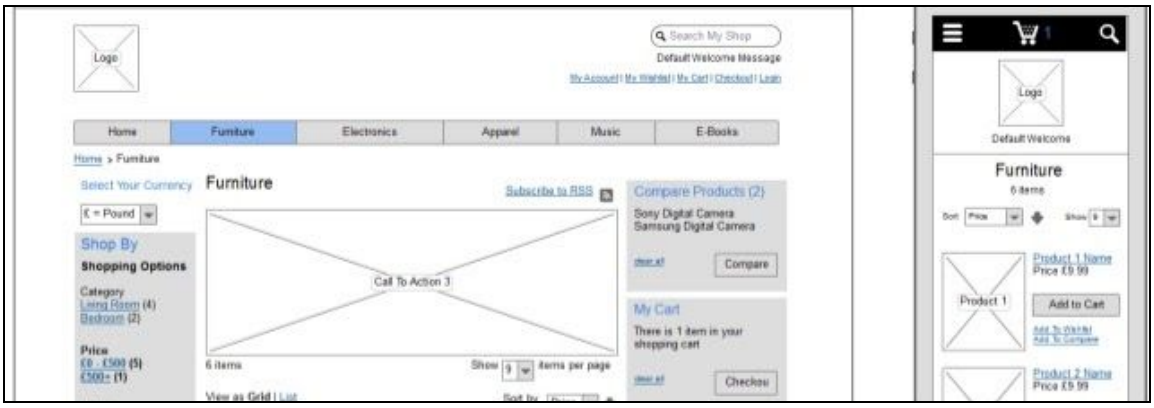
---

## 4. Crear un wireframe y un boceto del sitio

Es necesario que todo el grupo se junte para esta actividad.

Seguramente se haya hablado un poco de los wireframes durante las clases, pero en caso de que deseen profundizar sobre este tema, les recomendamos el siguiente artículo.

El wireframe es una representación sencilla de la estructura general y los componentes de nuestro sitio.




Pueden trabajar sus wireframes en lápiz y papel o bien utilizar una herramienta digital. Cualquier aplicación de diseño gráfico o de interfaces como Illustrator, XD, Sketch u online como Marvel, Figma, Wireframe.cc o Diagrams.net.

**Entregable:** Wireframe digital o analógico de las siguientes secciones del sitio:

- Home
- Detalle de producto
- Carrito de compras
- Formulario de registro
- Formulario de login

**Tip:** les recomendamos crear la carpeta **wireframes** para incluir este contenido.

**Entregable (Opcional):** Boceto o diseño del sitio incluyendo.

- Logo
- Colores
- Tipografías

**Tip:** les recomendamos crear la carpeta **design** para incluir este contenido.

## > Resumen de entregables

★ URL del repositorio con todos los colaboradores agregados.
★ Archivo README.md con:
  ○ Temática del sitio y público objetivo.
  ○ Listado de al menos 5 referentes.
★ Wireframe de las siguientes páginas:
  ○ Home
  ○ Detalle de producto
  ○ Carrito de compras
  ○ Formulario de registro
  ○ Formulario de login
★ Opcional: Boceto o diseño gráfico del sitio (logo, colores, tipografías, etc).

## > Cierre

Ya lo dijimos antes, pero vale la pena repetirlo: si quieren que su proyecto tenga más posibilidades de tener éxito, pónganle esmero a la etapa de planificación.

Tener un documento que explique el objetivo y el contexto del sitio así como tener un listado de referentes provee un marco de referencia para resolver dudas.

Tener un wireframe y un boceto del sitio permite que los integrantes del grupo trabajen por separado y que luego, al unir las piezas, todas coincidan. Por otro lado, es un documento de consulta a la hora de resolver dudas sobre cómo debe quedar tal o cual sección.

Por último, y no menos importante, sepan que las cosas pueden romperse, pueden no salir bien o tan bien como esperaban, y eso es totalmente normal. Lo importante es que aprendan a trabajar en conjunto para que el resultado sea cada vez mejor..

# Sprint 2

Bienvenidos a esta **segunda iteración** del **Trabajo Integrador**.

En la etapa anterior se encargaron de planificar: definiendo la temática del sitio, armando los bocetos y creando el diseño que servirá de guía para las posteriores etapas.

Ahora, es momento de arrancar a codear usando HTML y CSS. Con estas dos herramientas van a poder darle “vida” a los bocetos que prepararon.

## > Requisitos

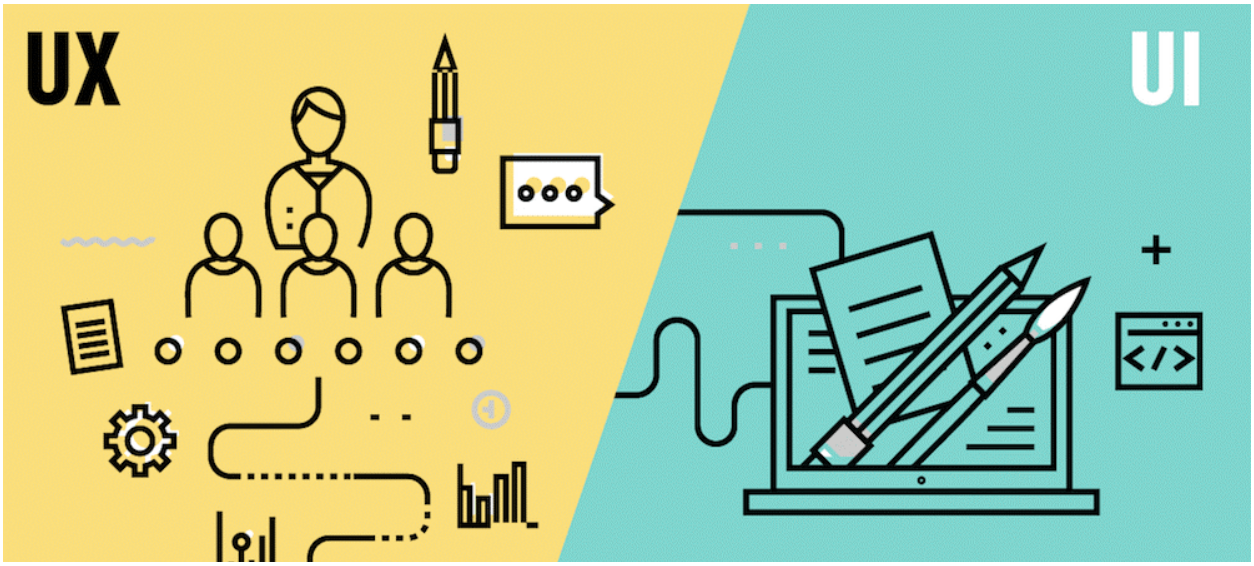
1. **Repositorio creado:** Indispensable para trabajar en equipo y realizar la entrega de cada sprint. Recuerden que todos los integrantes deben contribuir con código.

2. **Sitio definido y bocetos terminados:** La temática del sitio y los bocetos deberán estar definidos. Sin ellos, lo más probable es que tengan problemas para avanzar con rapidez y para trabajar por separado de manera eficiente. Los bocetos también sirven de guía a la hora de resolver dudas o conflictos.

## > Objetivo

Durante este iteración su foco deberá ser construir un sitio que:

- Sea estéticamente agradable (UI - User Interface).
- Sea fácil de usar (UX - User Experience).



# > Metodología

Vamos a seguir practicando las **metodologías ágiles**.

## La retrospectiva

A esta altura ya tenemos un sprint terminado. Eso quiere decir que tienen evidencia de cómo funcionó el grupo durante estas semanas y es un muy buen momento para hacer un análisis.

La **retrospectiva** se centra en **mejorar como equipo**. Los invitamos a que implementen la dinámica de la **estrella de mar** que busca resaltar aquello que hay que:

1. Comenzar a hacer.
2. Hacer más.
3. Continuar haciendo.
4. Hacer menos.
5. Dejar de hacer.

Pueden leer más sobre esta ceremonia aquí.

## El tablero de trabajo

Es una herramienta indispensable para realizar el seguimiento del proyecto. Permite:

- Organizar y dar seguimiento a todas las tareas.
- Saber quién está haciendo cada tarea en cada momento.
- Saber qué tareas están pendientes para comenzarlas.
- Saber qué criterios debe cumplir una tarea para considerarse lista.
- Saber si el sprint viene a buen ritmo o si está atrasado.

# > Consignas

**¡Importante!** Durante este sprint **estaremos trabajando** solamente la estructura y estética del sitio, lo que comúnmente se conoce como **la maqueta**.

No será necesario implementar ninguna funcionalidad más que los enlaces entre las diferentes páginas. Para consignas como la validación de formularios deberán mostrar cómo se verán los errores, aún si el formulario no esté funcional.

En todas las consignas que corresponda, se les presentarán las historias de usuario (*user stories*). Será su responsabilidad determinar las tareas que se desprenden de ellas.

## 1. Realizar una breve retrospectiva

Su equipo es ágil y uno de los secretos de la agilidad es el de fallar rápido para ser exitosos más pronto. Piensen qué hicieron bien en el sprint anterior, qué hicieron mal, qué deberían empezar a hacer, qué deberían dejar de hacer. Les recomendamos la dinámica de estrella de mar que mencionamos anteriormente.

**Entregable:** Crear un archivo **retro.md** en el repositorio y anotar las principales conclusiones de la retro del primer sprint.

## 2. Crear un tablero de trabajo

Indispensable para trabajar en equipo. Les dejamos dos recomendaciones, aunque son libres de utilizar cualquier herramienta o método que deseen:

- Trello: una herramienta muy flexible que permite dar seguimiento a todo tipo de proyectos o procesos. Les dejamos un tablero de muestra:

  https://trello.com/b/mDz4bKmd/proyecto-integrador

- GitHub: otra manera de organizar el trabajo a través de tableros. Les dejamos el artículo de GitHub que explica su funcionamiento:

  https://help.github.com/es/github/managing-your-work-on-github/about-project-boards

**Entregable:** Link al documento o plataforma que utilicen para organizar el trabajo. Agregarlo en el archivo README.md.

## 3. Crear la estructura de archivos utilizando Node.js+Express

Misma estructura que vimos en la clase de Express.

**Entregable:** Implementación de estructura MVC (Modelo-Vista-Controlador) para un proyecto de Node.js.

## 4. Página: inicio

Home del sitio con la oferta de productos y/o servicios.

Historias de usuario:

- ★ ***Como*** *cliente* ***quiero*** *ver un resumen de la oferta de productos* ***para*** *poder decidir rápidamente si quedarme o buscar el producto en otro sitio.*
- ★ ***Como*** *cliente* ***quiero*** *que el sitio funcione bien en mi teléfono celular* ***para*** *poder acceder en cualquier momento.*

**Entregable:** Archivo index.html junto con todos los recursos necesarios, estilos de CSS e imágenes.

## 5. Página: detalle de producto

Página a la que accede el cliente al hacer click en un producto.

Historias de usuario

- ★ ***Como*** *cliente* ***quiero*** *poder ver toda la información relevante del producto* ***para*** *decidir si voy a comprarlo.*
- ★ ***Como*** *cliente* ***quiero*** *poder agregar fácilmente el producto a mi carrito* ***para*** *poder comprarlo.*

**Entregable:** Archivo productDetail.html (o similar) junto con todos los recursos necesarios, estilos de CSS e imágenes.

## 6. Página: carrito de compras

Página donde el cliente puede visualizar su compra actual y proceder al pago.

Historias de usuario

- ★ ***Como*** *cliente* ***quiero*** *poder acceder al detalle de mi carrito de compras en cualquier momento* ***para*** *finalizar la compra.*
- ★ ***Como*** *cliente* ***quiero*** *poder modificar productos del carrito* ***para*** *poder eliminarlos de la compra o modificar la cantidad.*

**Entregable:** Archivo productCart.html (o similar) junto con todos los recursos necesarios, estilos de CSS e imágenes.

## 7. Página: registro

Página donde el cliente puede registrarse como usuario del sitio.

Historias de usuario

- ★ ***Como*** *cliente* ***quiero*** *poder registrarme* ***para*** *estar en su base de datos y acceder a mi perfil e historial de compras de una manera más sencilla y, además, facilitar el proceso de futuras compras.*

**Entregable:** Archivo register.html (o similar) junto con todos los recursos necesarios, estilos de CSS e imágenes.

### 8. Página: login

Página donde el cliente puede ingresar con el usuario que creó previamente.

Historias de usuario

- ★ ***Como*** *cliente* ***quiero*** *poder loguearme* ***para*** *acceder a mi historial de compras.*
- ★ ***Como*** *cliente* ***quiero*** *tener la opción de que el sitio me recuerde* ***para*** *no escribir mi usuario y contraseña cada vez.*
- ★ ***Como*** *cliente* ***quiero*** *poder recuperar la contraseña* ***para*** *poder ingresar si me la olvido.*

**Entregable:** Archivo login.html (o similar) junto con todos los recursos necesarios, estilos de CSS e imágenes.

## > Resumen de entregables

- ★ Archivo **retro.md** con el resultado de la retrospectiva.
- ★ Enlace al tablero de trabajo en el archivo **README.md**.
- ★ Aplicación Node.js+Express con:
  - Home (index.html)
  - Detalle del producto (productDetail.html)
  - Carrito de compras (productCart.html)
  - Formulario de registro (register.html)
  - Formulario de login (login.html)

## > Cierre

El desarrollo web moderno pone en el centro al usuario. Esto quiere decir que vamos a **pensar primero en la necesidad o el problema** que tiene nuestro usuario, **antes de programar la solución**.

En el caso del cliente, la necesidad podría ser la de comprar un artículo en el sitio y, en el caso del administrador, podría ser la de gestionar los productos y las ventas. Queda entonces de su lado la responsabilidad de asegurarse que los usuarios tengan lo necesario para poder cumplir sus tareas con facilidad.

**Las metodologías ágiles** son iterativas y toman como fuente de mejora las iteraciones pasadas. No se preocupen si las cosas no salen como esperaban en un comienzo. Tomen lo bueno, lo malo y úsenlo **para ir mejorando como equipo en cada entrega**.

**Cuidadito con este sprint**. Si bien las consignas son simples, van a estar escribiendo código en forma colaborativa por primera vez. Lleva tiempo organizarse y posiblemente haya conflictos en el camino. **Les recomendamos comenzar lo antes posible** y recuerden arrancar con Git desde el principio para agarrar velocidad con la práctica.

¡Les toca brillar a ustedes! ¡Buena suerte! 🤘😎☝✨

# Programación Web Full Stack

**Trabajo integrador - S3 - M03C11**

# Sprint 3

## > Introducción

¡Llegamos a la **tercera iteración** del **Trabajo Integrador**!

En las etapas anteriores se encargaron de pensar, planificar y maquetar las principales páginas de su sitio. Todo lo que hicieron hasta ahora es completamente estático y sin dudas hay mucho del código que se repite innecesariamente.

Es hora de agregar algo de dinamismo a su web y empezar a reutilizar todos aquellos componentes que se comparten, como: el header, el footer, la navegación y los productos.

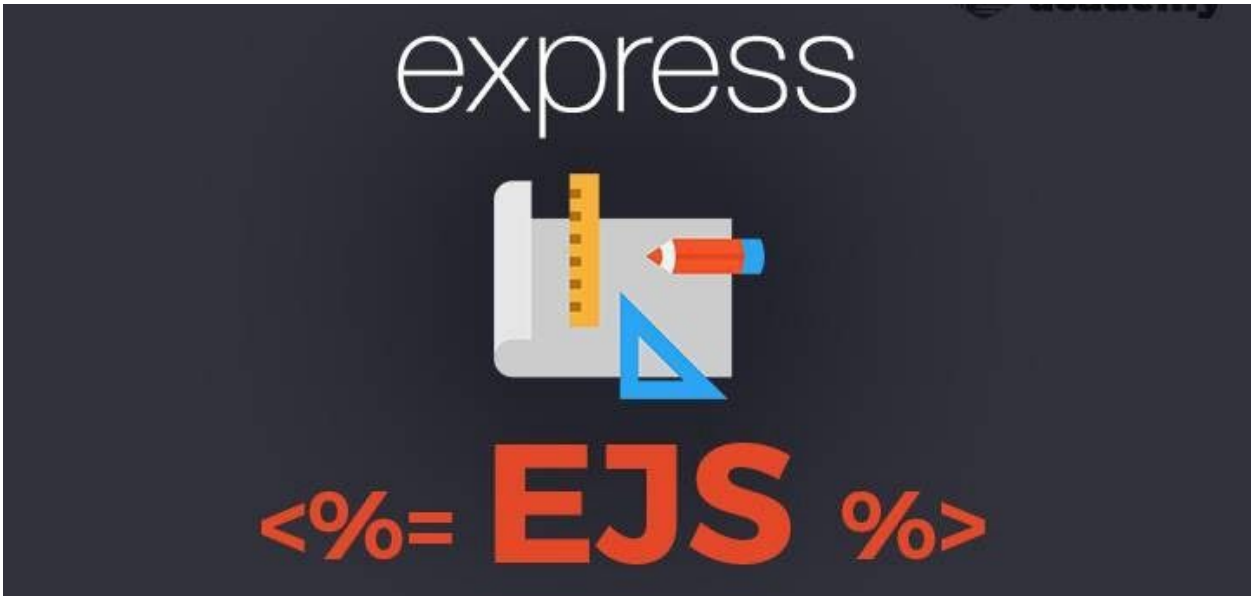
## > Requisitos

1. **Sitio maquetado:** necesitan un sitio completo con su HTML y CSS para poder empezar a separar los componentes y aplicar el motor de templates.

## > Objetivo

Durante esta iteración su foco será el de modificar el sitio para que:

- Reutilice los componentes compartidos: header, menu, footer, etc.
- Muestre contenido dinámicamente a través de un motor de templates (Express+EJS).



# > Metodología

Como ya se habrán dado cuenta, las **metodologías ágiles** son iterativas. Les toca otra vuelta de retro y planificación. 📝🤓👌✨

## La retrospectiva

Ya con dos sprints terminados deberían empezar a marcarse tendencias. Se conocen un poco más entre ustedes, pudieron ver en cuáles aspectos el trabajo fluye y en cuáles no. También debería ser más evidente quién es más hábil para el código, quién para lo estético y quién para lo organizativo.

Usen toda esta experiencia y recuerden que la **retrospectiva** se centra en **mejorar como equipo**, su objetivo es mejorar la dinámica del grupo, no están evaluando el sitio o el producto en sí.

Implementen nuevamente la dinámica de la **estrella de mar**, resaltando aquello que hay que:

1. Comenzar a hacer
2. Hacer más
3. Continuar haciendo
4. Hacer menos
5. Dejar de hacer

Pueden leer más sobre esta ceremonia aquí.

## El tablero de trabajo

Llegó la hora de la **planificación** y les toca **reiniciar el tablero** para acomodar el nuevo sprint. Este proceso es muy importante si quieren tener más chances de llegar bien al final de este tercer tirón.

Si quedan tareas pendientes del anterior sprint, será su responsabilidad priorizarlas y agregarlas a este sprint.

Recuerden que durante la planificación es importante:

- Debatir cada tarea en conjunto para asegurarse de que no haya dudas sobre su alcance (hasta dónde van a hacer) y sobre cómo van a resolverla.
- Estimar la dificultad de la tarea y si esta requiere de que alguna otra tarea esté terminada antes de poder iniciarla (para determinar el orden).
- Asignar tentativamente a los responsables de cada una de ellas.

## (Opcional) La reunión daily o weekly

Recuerden que nada está escrito en piedra y que ser ágil se trata de que puedan ser flexibles para llegar a destino de la mejor manera. Si algo ocurre en el camino que altere los planes, lo mejor es saberlo cuanto antes y decidir un nuevo camino de acción.

La **daily standup** es una reunión que en los equipos de **Scrum** se realiza todos los días, donde cada integrante habla como máximo tres minutos de tres temas puntuales:

- Qué hizo ayer.
- Si se encontró con algún impedimento.
- Qué va a hacer hoy.

El formato está pensado para ser rápido y liviano, solo se transmite la información más importante de las tareas y los impedimentos.

De esta manera, todo el equipo está al tanto de lo que está haciendo cada uno y en el caso de que haya algún impedimento pueden aportar a la solución.

**Importante:** no es necesario que esto lo hagan todos los días, al menos una vez por semana sería ideal.

# > Consignas

### Planificación y trabajo en equipo

1. **Realizar un breve retrospectiva**

   Nuevamente piensen qué hicieron bien en el sprint anterior, qué hicieron mal, qué deberían empezar a hacer, qué deberían dejar de hacer, sigan esta dinámica.

   **Entregable:** actualizar el archivo retro.md con las principales conclusiones de la retro del segundo sprint.

2. **Actualizar el tablero de trabajo**

   Tiempo de planificación: discutan las tareas que se desprenden de este documento, determinen en qué orden deberían realizarlas y asignen integrantes a cada tarea (recuerden que pueden cambiar la organización si es necesario).

   **Entregable:** link al documento o plataforma que utilicen para organizar el trabajo.

3. **Implementar daily/weekly standups (Opcional)**

   Cada equipo habla como máximo tres minutos de tres temas puntuales:

   - Qué hizo ayer.
   - Si se encontró con algún impedimento.
   - Qué va a hacer hoy.

   **Entregable:** Archivo daily.md o weekly.md con un resumen de las tareas completadas, los impedimentos encontrados y las soluciones propuestas indicando los integrantes.

## Template engines

### 4. Implementar el motor de templates

Implementar el módulo EJS y renombrar todas las vistas actuales para que utilicen la extensión **.ejs**.

Modificar los métodos de los controladores para que utilicen el método **render()**.

**Entregable:** sitio actualizado con todas las vistas y rutas implementando EJS.

### 5. Separar las vistas en carpetas (Opcional)

Si tenemos en cuenta que nuestro sitio va a crecer y que muy pronto tendremos un montón de páginas, nos conviene mantener el orden desde el principio.

Crear, dentro de la carpeta **views**, la carpeta **products** y la carpeta **users**. Dentro de products pondremos todas las vistas de productos que tengamos (por ejemplo: listado, detalle, creación, edición, etc.). Dentro de users pondremos todas las vistas de usuarios que tengamos (por ejemplo: registro, login, perfil, etc.).

- Usuarios: src/views/users/
- Productos: src/views/products/

**Entregable:** estructura actualizada de directorios y archivos de las vistas.

### 6. Separar los componentes repetidos en archivos parciales

Crear una carpeta llamada **partials** dentro de la carpeta de **views**, separar las áreas comunes del sitio. Como mínimo nos gustaría ver:

- Head (incluyendo todo el elemento **<head></head>**) → head.ejs
- Header (incluyendo barras de navegación) → header.ejs
- Footer (incluyendo todo el elemento **<footer></footer>**) → footer.ejs
- (Opcional) Otras secciones de su sitio que se repitan.

Pueden separar otros componentes de la misma manera si lo creen útil. Por ejemplo: los productos dentro de un listado.

Recuerden implementar los archivos parciales en todas las páginas que correspondan.

**Entregable:** carpeta **partials** dentro de **views** con todos los archivos parciales.

**Entregable:** sitio actualizado con la implementación de los **partials**.

## 7. Página: creación y edición de productos

Formulario al que accede el usuario administrador para cargar nuevos productos y editar los existentes.

Un buen punto de partida para los campos de estos formularios puede ser el siguiente:

- Nombre del producto (*name*)
- Descripción (*description*)
- Imagen (*image*)
- Categoría (*category*)
- Colores (o cualquier otro campo similar como: tamaños, talles, etc)
- Precio (*price*)

Historias de usuario

★ ***Como** administrador **quiero** poder crear nuevos productos **para** agregarlos a los listados del sitio.*

★ ***Como** administrador **quiero** poder modificar los productos existentes **para** corregir información o actualizar precios.*

**Entregable:** páginas de creación y edición de productos en formato **.ejs** junto con todos los recursos necesarios, estilos de CSS e imágenes.

## > Resumen de entregables

- ★ Archivo **retro.md** con el resultado de la retrospectiva.
- ★ Archivo **daily.md** con sus opiniones sobre las daylies/weeklies. (Opcional)
- ★ Tablero de trabajo actualizado.
- ★ Aplicación Node.js+Express+EJS con:
  - Archivos parciales (head, header, footer, etc.)
  - Home
  - Listado de productos
  - Detalle del producto
  - Carrito de productos
  - Formulario de registro y login
  - Formulario de carga y edición de productos

## > Cierre

Una de las maravillas de la programación es que nos permite automatizar los procesos que se repiten. 🤖 Ya vimos que JavaScript (así como la mayoría de los lenguajes de programación) nos permite escribir funciones para agrupar código que necesitemos reutilizar en el futuro.

Herramientas como EJS nos ayudan a hacer lo mismo con nuestras páginas, agrupando partes que se repiten en muchos lugares y generando componentes que podemos reutilizar en todos los lugares donde sean necesarios.

Otra de las ventajas de **EJS**, o de cualquier otro motor de plantillas, es que, si en algún momento tenemos que actualizar uno de esos componentes, basta con hacerlo en un solo lugar en vez de tener que ir a revisar cada página. 🧙‍♂️✨

¡Así que buena suerte con ese rompecabezas de componentes! 🧩🧩🧩

# Programación Web Full Stack

**Trabajo integrador - S4 - M03C11**

# Sprint 4

## > Introducción

¡Llegamos a la **cuarta iteración** del **Trabajo Integrador**!

Si completaron los tres sprints anteriores (los completaron, ¿verdad 😅?), deberían tener resuelta la parte visual de su sitio. Ya lo podemos navegar, pero todavía no lo podemos usar...

¡Muy bien! Nos toca ahora lograr que el sitio comience a cobrar vida haciendo que todos esos formularios y acciones trabajen con productos y usuarios reales almacenados en formato JSON.

## > Requisitos

1. **Productos y usuarios definidos:** Indispensable para crear la primera versión de nuestra fuente de datos de productos y usuarios. Deberán tener una buena idea de los campos que necesitan guardar para cada ítem y para cada usuario.

## > Objetivo

Durante esta iteración su foco será el de modificar el sitio para que muestre productos dinámicamente a través de una fuente de datos (JSON).

# > Metodología

¡Vamos con otra vuelta de retro y planificación! 📝🤓👌✨

No se salteen este paso, es más importante de lo que piensan. 😊☝️

## La retrospectiva

Para esta altura ya deberían ser expertos en encontrar puntos de mejora.

Implementen nuevamente la dinámica de la **estrella de mar**, resaltando aquello que hay que:

1. Comenzar a hacer
2. Hacer más
3. Continuar haciendo
4. Hacer menos
5. Dejar de hacer

Pueden leer más sobre esta ceremonia aquí.

## El tablero de trabajo

Otra vez toca **reiniciar el tablero** para acomodar este sprint.

Las tareas pendientes del anterior sprint deben priorizarlas y agregarlas a este sprint.

Recuerden que durante la planificación es importante:

- Debatir cada tarea en conjunto para asegurarse de que no haya dudas sobre su alcance (hasta dónde van a hacer) y sobre cómo van a resolverla.
- Estimar la dificultad de la tarea y si esta requiere de que alguna otra tarea esté terminada antes de poder iniciarla (para determinar el orden).
- Asignar tentativamente los responsables de cada una de ellas.

## (Opcional) La reunión daily o weekly

La **daily standup** es una reunión, que en los equipos de **Scrum** se realiza todos los días, donde cada integrante habla como máximo 3 minutos de 3 temas puntuales:

- Qué hizo ayer.
- Si se encontró con algún impedimento.
- Qué va a hacer hoy.

El formato está pensado para ser rápido y liviano, solo queremos la información más importante de las tareas y los impedimentos.

**Importante:** no es necesario que esto lo hagan todos los días, al menos una vez por semana sería ideal.

# > Consignas

## Planificación y trabajo en equipo

### 1. Realizar un breve retrospectiva

Nuevamente piensen qué hicieron bien el sprint anterior, qué hicieron mal, qué deberían empezar a hacer, qué deberían dejar de hacer, sigan esta dinámica.

**Entregable:** actualizar el archivo retro.md con las principales conclusiones de la retro del tercer sprint.

### 2. Actualizar el tablero de trabajo

Discutan las tareas que se desprenden de este documento, determinen en qué orden deberán realizarlas, asignen integrantes a cada tarea.

**Entregable:** link al documento o plataforma que utilicen para organizar el trabajo.

### 3. Implementar daily/weekly standups (Opcional)

Cada equipo habla como máximo 3 minutos de 3 temas puntuales:

- Qué hizo ayer.
- Si se encontró con algún impedimento.
- Qué va a hacer hoy.

**Entregable:** archivo daily.md o weekly.md con un resumen de las tareas completadas, los impedimentos encontrados y las soluciones propuestas indicando los integrantes.

## JSON y métodos de HTTP

### 1. Definir los campos necesarios para los productos y generar archivo JSON

Como paso previo a tener una base de datos relacional, vamos a estar trabajando con archivos JSON. Lo que necesitan hacer es decidir los campos que crean necesarios para sus productos. Una buena base para empezar sería:

- Identificador (ya hablaremos más sobre este campo): id
- Nombre del producto: name
- Descripción: description
- Imagen: image
- Categoría: category
- Colores (o cualquier otro campo similar como: talle): colors
- Precio: price

Una vez definidos los campos de sus productos, pueden utilizar la siguiente herramienta para generar el archivo JSON con datos: https://mockaroo.com/.

Aquí tienen un ejemplo ya generado que pueden tomar de referencia: https://mockaroo.com/9511ef20.

**Entregable:** carpeta **data** con archivo **products.json** con los datos de productos generados.

### 2. Definir los campos necesarios para los usuarios y generar archivo JSON

Similar al punto anterior, pero esta vez deberán generar datos de usuarios. Una buena base para empezar sería:

- Identificador: id
- Nombre: firstName
- Apellido: lastName
- Email: email
- Contraseña: password
- Categoría: category
- Imagen: image

Aquí tienen un ejemplo ya generado que pueden tomar de referencia: https://mockaroo.com/0b5f3440

**Entregable:** carpeta **data** con archivo **users.json** con los datos de usuarios generados.

### 3. CRUD de productos

CRUD es el acrónimo (en inglés) de "Crear, Leer, Actualizar y Borrar", es decir, poder crear, leer, actualizar y borrar un producto en particular.

Su tarea será la de implementar todos los métodos necesarios para poder trabajar con la fuente de datos JSON de productos que crearon en los puntos anteriores.

Recuerden que para cumplir ese objetivo necesitarán de siete rutas:

1. **/products** (GET)
   Listado de productos
2. **/products/create** (GET)
   Formulario de creación de productos
3. **/products/:id** (GET)
   Detalle de un producto particular
4. **/products** (POST)
   Acción de creación (a donde se envía el formulario)
5. **/products/:id/edit** (GET)
   Formulario de edición de productos
6. **/products/:id** (PUT)
   Acción de edición (a donde se envía el formulario):
7. **/products/:id** (DELETE)
   Acción de borrado

**Entregable:** sección funcional con listado, detalle, alta, modificación y baja de productos.

## > Resumen de entregables

- ★ Archivo **retro.md** con el resultado de la retrospectiva.
- ★ Archivo **daily.md** con sus opiniones sobre las daylies/weeklies. (Opcional)
- ★ Tablero de trabajo actualizado.
- ★ Archivos **products.json** y **users.json** con datos de productos y usuarios.
- ★ Administración completa de productos con:
  - Listado
  - Detalle
  - Creación
  - Edición
  - Eliminación

## > Cierre

De la misma manera que usamos wireframes y bocetos para definir el sitio, corregirlo y determinar si es viable, vamos a estar haciendo lo mismo con los datos y la funcionalidad.

Herramientas como Mokaroo nos permiten darle vida a nuestro sitio y probar rápidamente cómo funciona cuando está poblado de productos y de usuarios (en nuestra jerga se denomina Mockear nuestros datos). En la mayoría de los casos esta práctica nos permitirá detectar errores y puntos de mejora que jamás hubiéramos podido ver en un diseño o una maqueta estática.

Es lo que siempre estamos buscando cuando aplicamos agilidad: encontrar los fallos y puntos de mejora lo antes posible en el proceso de desarrollo. No es casualidad, a medida que avanzamos con el armado de un sitio o un producto, se torna más costoso hacer cambios, correcciones y ajustes. Pero no solo lo hacemos para ahorrar tiempo o costos, es un momento mágico cuando empezamos a ver que nuestro sitio cobra vida 🎩✨🐇, y ahora les toca a ustedes. 😎👍

# Programación Web Full Stack

**Trabajo integrador - S5 - M03C14**

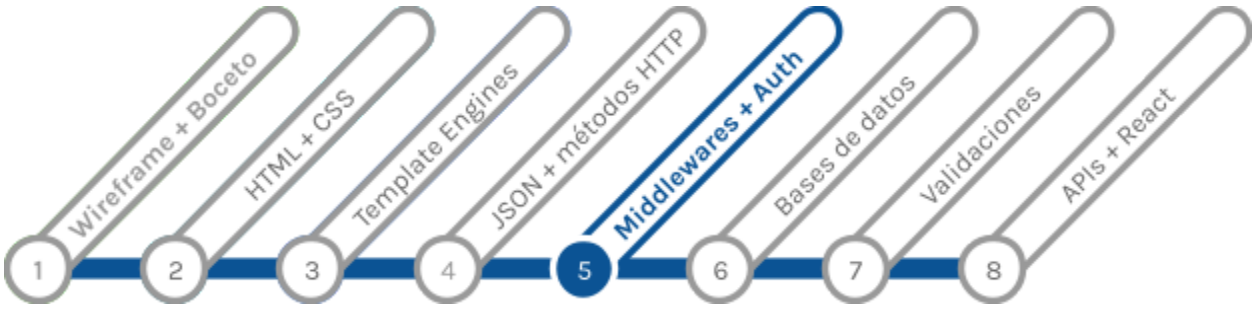
# Sprint 5

## > Introducción

¡Llegamos a la **quinta iteración** del **Trabajo Integrador**!

A esta altura deberían tener un **sitio dinámico utilizando EJS** como motor de plantillas, un **CRUD de productos** funcionando perfectamente con una fuente de datos **basada en JSON**. Es un buen momento para mirar atrás ⌛ y darse cuenta de cuánto avanzaron en tan poco tiempo 🤘😎🤘.

Nos queda entonces enfocarnos en los usuarios. En esta etapa vamos a estar generando el registro, el login, el perfil y, además, vamos a trabajar en las rutas a las cuales podrán acceder nuestros huéspedes (los visitantes que no hicieron login) y otras para los usuarios con login.



## > Requisitos

1. **Fuente de datos de usuarios:** indispensable para darle vida a los formularios de registro, login y a la página de perfil. Deberán tener una buena idea de los campos que necesitan guardar para cada usuario. **Recuerden que en el sprint anterior tienen una sugerencia de los campos mínimos que deberían tener**.

## > Objetivo

Durante esta iteración su foco será el de modificar el sitio para que:

- Permita el flujo de registro, login y logout de usuarios.
- Permita recordar al usuario para que pueda ingresar sin volverse a loguear.
- Tenga rutas accesibles solo por huéspedes (visitantes sin login).
- Tenga rutas accesible solo por usuarios (que hicieron login).

# > Metodología

¡Vamos con otra vuelta de retro y planificación! 📝🤓👌✨

No se salteen este paso, es más importante de lo que piensan. 😊☝️

## La retrospectiva

Para esta altura ya deberían ser expertos en encontrar puntos de mejora.

Implementen nuevamente la dinámica de la **estrella de mar**, resaltando aquello que hay que:

1. Comenzar a hacer.
2. Hacer más.
3. Continuar haciendo.
4. Hacer menos.
5. Dejar de hacer.

Pueden leer más sobre esta ceremonia aquí.

## El tablero de trabajo

Otra vez toca **reiniciar el tablero** para acomodar este sprint.

Las tareas pendientes del anterior sprint deben ser priorizadas y agregadas a este sprint.

Recuerden que durante la planificación es importante:

- Debatir cada tarea en conjunto para asegurarse de que no haya dudas sobre su alcance (hasta dónde van a hacer) y sobre cómo van a resolverla.
- Estimar la dificultad de la tarea y si esta requiere de que alguna otra tarea esté terminada antes de poder iniciarla (para determinar el orden).
- Asignar tentativamente los responsables de cada una de ellas.

### (Opcional) La reunión daily o weekly

La **daily standup** es una reunión, que en los equipos de **Scrum** se realiza todos los días, donde cada integrante habla como máximo 3 minutos de 3 temas puntuales:

- Qué hizo ayer.
- Si se encontró con algún impedimento.
- Qué va a hacer hoy.

El formato está pensado para ser rápido y liviano, solo queremos la información más importante de las tareas y los impedimentos.

**Importante:** no es necesario que esto lo hagan todos los días, al menos una vez por semana sería ideal.

## > Consignas

### Planificación y trabajo en equipo

1. **Realizar un breve retrospectiva**

   Nuevamente piensen qué hicieron bien en el sprint anterior, qué hicieron mal, qué deberían empezar a hacer, qué deberían dejar de hacer, sigan esta dinámica.

   **Entregable:** actualizar el archivo retro.md con las principales conclusiones de la retro del cuarto sprint.

---

2. **Actualizar el tablero de trabajo**

   Discutan las tareas que se desprenden de este documento, determinen en qué orden deberán ser realizadas, asignen integrantes a cada tarea.

   **Entregable:** link al documento o plataforma que utilicen para organizar el trabajo.

### 3. (Opcional) Implementar daily / weekly standups

Cada equipo habla como máximo 3 minutos de 3 temas puntuales:

- Qué hizo ayer.
- Si se encontró con algún impedimento.
- Qué va a hacer hoy.

**Entregable:** archivo daily.md con sus principales impresiones (positivas, neutras o negativas) sobre la utilidad de esta ceremonia.

---

## Usuarios y middlewares

### 4. Implementar la entidad de usuarios

Tomando como ejemplo lo que hicieron para productos, replicar la estructura de archivos y directorios que necesitarán para implementar las funcionalidades.

Les sugerimos esta estructura, aunque si trabajaron diferente, vale igual. 😉

- Rutas: src/routes/users.js
- Controlador: src/controllers/usersController.js
- Vistas: src/views/users/
- Directorio para imágenes: public/images/users/
- Colección: src/data/users.json

**Entregable:** estructura de archivos y directorios de usuarios.

## 5. Implementar el registro de usuarios

Tomando como referencia el formulario de creación de productos, implementar el formulario de registro de usuarios.

- Deberá incluir los campos mínimos mencionados en el sprint anterior.
- Deberá permitir la subida de una imagen de perfil (con Multer).
- Deberá encriptar la contraseña ingresada por el usuario (con bcrypt.js).
- Deberá guardar los datos enviados en el archivo JSON de usuarios.

**Entregable:** formulario funcional de creación de usuarios.

---

## 6. Implementar el login de usuarios

Ahora, es momento de poner en práctica middlewares, sesiones y cookies. Deberán implementar un formulario de login que:

- Incluya los campos de email y password.
- Verifique la información enviada por el usuario y según el caso:
    - Redireccione a la home o a la página de perfil en caso de éxito y muestre los datos del usuario en algún lugar del sitio, como el header.
    - Redireccione nuevamente al login en caso de error.

**Entregable:** formulario funcional de login.

---

## 7. (Opcional) Implementar la función de recordar al usuario

Deberán agregarle al formulario de login la posibilidad de que se recuerde al usuario (checkbox). En caso de que el usuario decida ser recordado:

- Utilizar cookies para guardar esa información en el navegador.
- Implementar un middleware de aplicación que busque la cookie y loguee al usuario en caso de que exista y sus datos sean correctos.

### 8. Implementar rutas de huéspedes y de usuarios

Ahora que tienen un login funcionando, su próximo desafío será el de separar las rutas que se pueden acceder en cualquier momento, de las que se puede acceder solo si uno no está logueado y, por último, de las que requieren estar logueado.

En cada caso deberán implementar el comportamiento que corresponda:

- Rutas accesibles por cualquiera → sin cambios
- Rutas accesibles solo **sin** login → redirigen al perfil
- Rutas accesibles solo **con** login → redireccionan al login

## > Resumen de entregables

- Archivo **retro.md** con el resultado de la retrospectiva.
- (Opcional) Archivo **daily.md** con sus opiniones sobre las dailies/weeklies.
- Tablero de trabajo actualizado.
- Formulario de registro con:
  - Los campos mínimos mencionados en el sprint anterior.
  - Subida de una imagen de perfil.
  - Guardado en JSON con encriptación de contraseña.
- Formulario de login con:
  - Campos de email y password.
  - (Opcional) Función de recordar al usuario.
- Rutas de huéspedes y usuarios:
  - Las de huéspedes deberán redireccionar al perfil si el usuario está logueado.
  - Las de usuarios deberán redireccionar al login si el usuario no está logueado.

## > Cierre

Ya habíamos dicho que en programación siempre buscamos no repetirnos, así que cuando una tarea la estamos haciendo muchas veces en diferentes partes de nuestro código, la abstraemos. Es decir, que aislamos ese comportamiento en un bloque de código reutilizable.

Los middlewares nos permiten justamente eso. Un middleware de aplicación se ejecutará en cada pedido del cliente, mientras que un middleware de ruta se ejecutará cada vez que se pida esa ruta particular.

Con estas dos herramientas se pueden hacer cosas muy poderosas. De hecho, cada uno de los paquetes que han instalado (Express, EJS, method-override, Multer, etc.) están construidos como middlewares. Ahora ya pueden construir los suyos. 😎✌✨

Por otro lado, las sesiones y las cookies son parte del funcionamiento normal de la mayoría de los sitios que visitan a diario. Es otra parte del desarrollo web que van a dominar si siguen poniéndole horas de práctica a todo esto.

El mundo digital que los rodea está escrito utilizando este tipo de tecnologías. A medida que avancen en su aprendizaje les va a ser más fácil entender cómo funciona todo este ecosistema de webs y aplicaciones.

# Programación Web Full Stack

**Trabajo integrador - S6 - M07C17**

# Sprint 6

## > Introducción

¿Qué? ¡Qué de a poco estamos llegando al final! ✋😳✋ ¡Es hora de la **sexta iteración** del **Trabajo Integrador**!

Nos toca ahora dejar atrás el viejo y querido JSON para pasar a algo más profesional que escale mejor cuando nuestro sitio salga a producción. 🚀 Ya casi estamos para salir, ya casi. 😎👌

En este sprint estaremos trabajando con SQL por un lado y con Sequelize por el otro. **¡Vamos!**

## > Requisitos

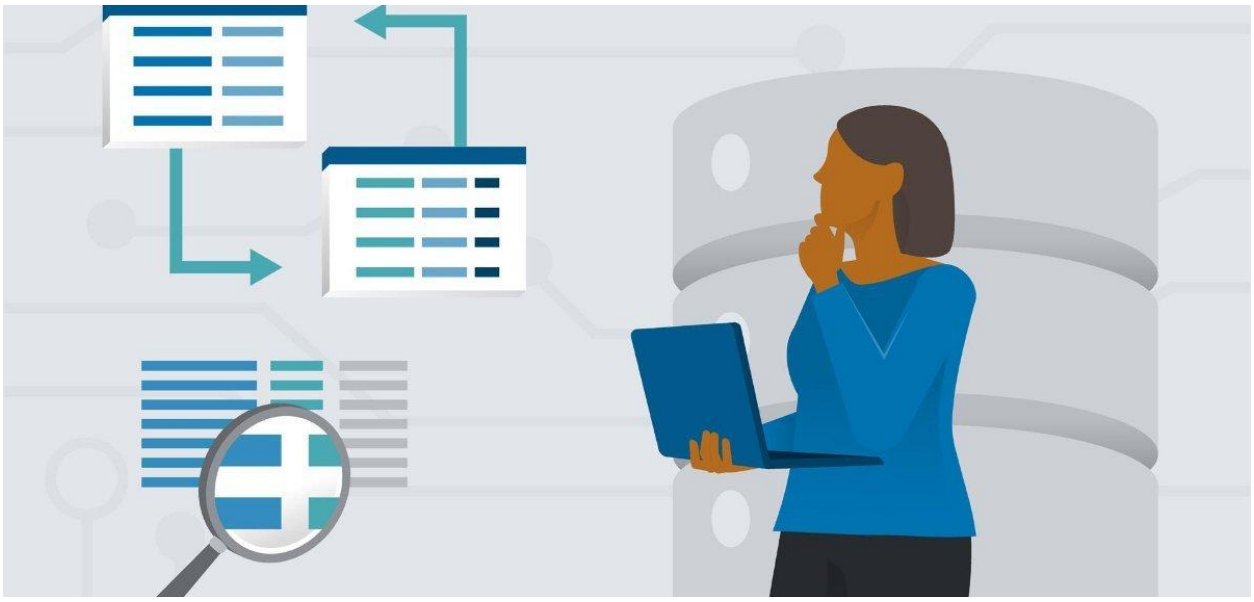
1. **Fuente de datos de usuarios y productos:** los archivos JSON serán su fuente de inspiración. Recuerden que en el sprint 3 tienen una sugerencia de los campos mínimos.
2. **CRUD de productos y usuarios:** que hoy funcionan para JSON y sobre los cuales implementarán la magia de Sequelize. ✨🧙✨

## > Objetivo

Durante esta iteración su foco será el de crear e implementar la base de datos de su sitio.

En la **primera parte**, van a estar pensando en la **estructura** que será necesaria para que la base de datos cumpla con los requisitos del negocio: tablas, campos, tipos de datos y relaciones.

La **segunda parte** la van a pasar implementando la base de datos que crearon en la primera parte, utilizando el módulo de **Sequelize**.



# > Metodología

¡Vamos con la sexta vuelta de retro y planificación! 📝🤓👌✨

Como siempre, les recomendamos que no se salteen este paso, es muy importante. 😊☝️

## La retrospectiva

A esta altura es posible que el equipo haya alcanzado una buena velocidad de trabajo. Si es así, pueden enfocarse en mantenerla. De lo contrario, deberán enfocarse en los puntos que se puedan mejorar.

Implementen nuevamente la dinámica de la **estrella de mar**, resaltando aquello que hay que:

1. Comenzar a hacer.
2. Hacer más.
3. Continuar haciendo.
4. Hacer menos.
5. Dejar de hacer.

Pueden leer más sobre esta ceremonia aquí.

## El tablero de trabajo

Momento de **reiniciar el tablero** para acomodar este sprint de bases de datos. Como siempre, lo que haya quedado del anterior sprint se suma al actual.

Recuerden que durante la planificación es importante:

- Debatir cada tarea en conjunto para asegurarse de que no haya dudas sobre su alcance (hasta dónde van a hacer) y sobre cómo van a resolverla.
- Estimar la dificultad de la tarea y si esta requiere de que alguna otra tarea esté terminada antes de poder iniciarla (para determinar el orden).
- Asignar tentativamente los responsables de cada una de ellas.

### (Opcional) La reunión daily o weekly

La **daily standup** es una reunión, que en los equipos de **Scrum** se realiza todos los días, donde cada integrante habla como máximo 3 minutos de 3 temas puntuales:

- Qué hizo ayer.
- Si se encontró con algún impedimento.
- Qué va a hacer hoy.

El formato está pensado para ser rápido y liviano, solo queremos la información más importante de las tareas y los impedimentos.

**Importante:** no es necesario que esto lo hagan todos los días, al menos una vez por semana sería ideal.

## > Consignas

### Planificación y trabajo en equipo

1. **Realizar un breve retrospectiva**

   Piensen qué hicieron bien en el sprint anterior, qué hicieron mal, qué deberían empezar a hacer, qué deberían dejar de hacer, sigan esta dinámica.

   **Entregable:** actualizar el archivo retro.md con las principales conclusiones de la retro del quinto sprint.

---

2. **Actualizar el tablero de trabajo**

   Discutan las tareas que se desprenden de este documento, determinen en qué orden deberán realizarlas, asignen integrantes a cada tarea.

   **Entregable:** link al documento o plataforma que utilicen para organizar el trabajo.

### 3. (Opcional) Implementar daily / weekly standups

Cada equipo habla como máximo 3 minutos de 3 temas puntuales:

- Qué hizo ayer.
- Si se encontró con algún impedimento.
- Qué va a hacer hoy.

**Entregable:** archivo daily.md con sus principales impresiones (positivas, neutras o negativas) sobre la utilidad de esta ceremonia.

---

## Bases de datos y Sequelize

### 4. Diagrama de base de datos

Toda buena base de datos empieza en la mesa de dibujo. Tendrán que armar el Diagrama de Entidades y Relaciones (DER). Piensen en un buen diseño, armen un diagrama legible, con relaciones correctas y las claves foráneas para representarlas.

Recuerden que luego deberán implementar Sequelize y que, por lo general, los ORMs como este trabajan mejor con los nombres de tablas en inglés.

Les proponemos la siguiente estructura, aunque la pueden ajustar a la necesidad de su proyecto. 😉

- Usuarios (recuerden ver los campos sugeridos en el sprint 3).
- Productos (recuerden ver los campos sugeridos en el sprint 3).
- Tablas secundarias (según lo requiera su proyecto).
    - Para productos: categorías, marcas, colores, talles, etc.
    - (Opcional) Para usuarios: categorías.
- (Opcional) Carrito de compras.
    - Con detalle de quién hizo la compra, cantidad de ítems y precio total.
- (Opcional) Productos de cada carrito de compras.

Les sugerimos utilizar draw.io ya que es fácil de usar y soporta diagramas DER.

**Entregable:** diagrama de entidad-relación de su base de datos en formato PDF.

## 5. Script de estructura

Tomando como referencia el diagrama del punto anterior, tienen que escribir las sentencias de SQL que crearán las tablas y sus relaciones.

- Deberá incluir la creación de la base de datos (create database...).
- Deberá incluir la creación de todas las tablas del sitio (create table...).
- Deberá incluir los tipos de datos de los campos y sus restricciones (primary keys, (not) null, unique, default, etc).
- Deberá incluir las relaciones entre las diferentes tablas (foreign keys).

**Entregable:** archivo **structure.sql** que permita crear la base de datos completa.

---

## 6. (Opcional) Script de datos

Ya tenemos la estructura, ahora faltan los datos. El script de datos permite que cualquier desarrollador (o docente 😉) descargue el proyecto, ejecute el script y ya pueda ver el sitio funcionado sin más pasos.

El script debería:

- Poblar la tabla de usuarios.
- Poblar la tabla de productos.
- Poblar las tablas secundarias (categorías, marcas, colores, talles, etc).
- (Opcional) Poblar la tabla de carrito de compras.
- (Opcional) Poblar la tabla de productos de carritos de compras.

Una vez definidos los campos de sus tablas, nuevamente pueden utilizar Mockaroo, pero esta vez para generar el archivo SQL con datos. 😉☝

**Entregable:** archivo con extensión **data.sql** que permita poblar la base con datos.

## 7. Creación de carpeta Sequelize y archivos de modelos

Mediante la herramienta **sequelize-cli** deberán crear la carpeta que contenga los archivos de configuración de **Sequelize**. Una vez configurado Sequelize, seguirá crear los archivos de modelos para explicarle cómo es la estructura de la base de datos.

La carpeta **database** deberá incluir:

- Los archivos de configuración para que Sequelize se conecte a la base de datos.
- Los archivos de modelos para representar las tablas de:
    - Usuarios.
    - Productos.
    - Tablas secundarias (categorías, marcas, colores, talles, etc).
    - (Opcional) Carrito de compras.
    - (Opcional) Productos de cada carrito de compras.
- Los modelos deben incluir todas las relaciones existentes en la base de datos.

**Entregable:** carpeta database que incluya los archivos de configuración y archivos de modelos junto con sus relaciones.

---

## 8. ¡CRUD!

Ya es hora de tener un CRUD como la gente. Qué bueno que tenemos a Sequelize de nuestro lado. Les pedimos que en su sitio se pueda:

- Para productos:
    - Crear
    - Editar
    - Eliminar
    - Listar
    - Ver el detalle
    - Buscar

- Para usuarios:
    - Crear
    - Editar
    - Ver el detalle
- (Opcional) CRUDs de tablas secundarias.
- (Opcional) Agregar paginado a los listados y buscadores.

**Entregable:** rutas, controladores y vistas necesarias para que suceda lo detallado previamente utilizando Sequelize para trabajar con la base de datos.

## > Resumen de entregables

- Archivo **retro.md** con el resultado de la retrospectiva.
- (Opcional) Archivo **daily.md** con sus opiniones sobre las dailys/weeklys.
- Tablero de trabajo actualizado.
- Diagrama de base de datos.
- Script de creación de estructura de base de datos con:
    - Creación de la base de datos y de todas sus tablas.
    - Tipos de datos de los campos y sus restricciones.
    - Relaciones entre las diferentes tablas.
- (Opcional) Script de datos de base de datos para:
    - Tabla de usuarios.
    - Tabla de productos.
    - Tablas secundarias (categorías, marcas, colores, talles, etc).
    - (Opcional) Tabla de carrito de compras y productos de carritos de compras.
- Creación de carpeta Sequelize con:
    - Archivos de configuración.
    - Modelos con sus relaciones.
- CRUD
    - De productos.
    - De usuarios.
    - (Opcional) De tablas secundarias.

## > Cierre

¡Wow! Las bases de datos son una herramienta que permite que el almacenamiento de los datos crezca de forma escalable y veloz. Puede ser un trabajo arduo (sobretodo si ya teníamos cosas hechas en JSON… 😅), pero el resultado es increíble. No solo es increíble porque nuestra aplicación está lista para crecer radicalmente, sino porque **¡completamos un back-end completo!**

Pensar, diseñar, implementar una base de datos no es sencillo, pero un buen diseño inicial va a asegurar nuestro éxito en el tiempo. Además, implementarlo en Sequelize nos permite correrno un poco de SQL y aprovechar todas las prestaciones de un buen ORM.

Si llegamos hasta acá, felicitaciones, somos cracks. 😎✌✨

## Programación Web Full Stack

**Trabajo integrador - S7 - M08C20**

# Trabajo Integrador - Sprint 7

## > Introducción

Que levante la mano 🙋‍♀️🙋‍♀️ quien nunca haya cargado un formulario de mala gana 📋🖊️😡 y con datos que ni siquiera tenían sentido. Tal vez para acceder al increíble servicio que ofrecía ese sitio que nos contaron, o por participar de esa promoción increíble que solo nos pedía llenar un formulario de 352 campos. ¿Todas y todos culpables? Ya me parecía… 😆

Si el formulario se llena con datos inválidos no hay problema, el problema es si llegan a la base de datos, ahí es donde se desata el caos y se rompe todo. **¡Para eso están las validaciones!**

En este sprint vamos a estar validando tanto desde el front-end como desde el back-end.

## > Requisitos

1. **Base de datos bien configurada:** la protección final y más importante es la de la base de datos. Si todo falla, la base de datos evitará que se carguen datos que puedan dañarla.

## > Objetivo

Vamos a estar validando los formularios del sitio trabajando de atrás para adelante.

En la **primera parte** vamos a estar validando los datos desde el **back-end**. Esta es una protección muy importante ya que como sabemos el usuario no puede acceder al mismo.

En la **segunda parte** vamos a estar validando en el **front-end**. Con esto evitaremos estar enviando información inválida al servidor y, además, mejoraremos la experiencia del visitante.

**Importante** tener en cuenta que la validación del front-end puede ser deshabilitada desde el navegador. Por eso es indispensable que también validemos en el back-end.

# > Metodologías ágiles

¡Vamos con la séptima vuelta de retro y planificación! 📝🤓👌✨

Como siempre les recomendamos que no se salteen este paso, es muy importante. 😊☝️

## La retrospectiva

A esta altura es posible que el equipo haya alcanzado una buena velocidad de trabajo. Si es así, pueden enfocarse en mantenerla. De lo contrario, deberán enfocarse en los puntos que se puedan mejorar.

Implementen nuevamente la dinámica de la **estrella de mar**, resaltando aquello que hay que:

1. Comenzar a hacer.
2. Hacer más.
3. Continuar haciendo.
4. Hacer menos.
5. Dejar de hacer.

Pueden leer más sobre esta ceremonia aquí.

## El tablero de trabajo

Momento de **reiniciar el tablero** para acomodar este sprint de bases de datos. Como siempre, lo que haya quedado del anterior sprint se suma al actual.

Recuerden que durante la planificación es importante:

- Debatir cada tarea en conjunto para asegurarse de que no haya dudas sobre su alcance (hasta dónde van a hacer) y sobre cómo van a resolverla.
- Estimar la dificultad de la tarea y si ésta requiere de que alguna otra tarea esté terminada antes de poder iniciarla (para determinar el orden).
- Asignar tentativamente los responsables de cada una de ellas.

### (Opcional) La reunión daily o weekly

La **daily standup** es una reunión, que en los equipos de **Scrum** se realiza todos los días, donde cada integrante habla como máximo 3 minutos de 3 temas puntuales:

- Qué hizo ayer.
- Si se encontró con algún impedimento.
- Qué va a hacer hoy.

El formato está pensado para ser rápido y liviano, solo queremos la información más importante de las tareas y los impedimentos.

**Importante:** no es necesario que esto lo hagan todos los días, al menos una vez por semana sería ideal.

## > Consignas

### Planificación y trabajo en equipo

1. **Realizar un breve retrospectiva**

   Piensen qué hicieron bien en el sprint anterior, qué hicieron mal, qué deberían empezar a hacer, qué deberían dejar de hacer, sigan esta dinámica.

   **Entregable:** Actualizar el archivo retro.md con las principales conclusiones de la retro del sexto sprint.

---

2. **Actualizar el tablero de trabajo**

   Discutan las tareas que se desprenden de este documento, determinen en qué orden deberán realizarlas, asignen integrantes a cada tarea.

**Entregable:** link al documento o plataforma que utilicen para organizar el trabajo.

(Opcional) Implementar daily / weekly standups

Cada equipo habla como máximo 3 minutos de 3 temas puntuales:

- Qué hizo ayer.
- Si se encontró con algún impedimento.
- Qué va a hacer hoy.

**Entregable:** archivo daily.md con sus principales impresiones (positivas, neutras o negativas) sobre la utilidad de esta ceremonia.

---

## Validaciones de back-end y front-end

### 3. Validaciones del back-end

Si queremos hacer que nuestro sitio sea una fortaleza a prueba de corrupción de datos, necesitamos validar todos aquellos puntos donde nos llegue información desde el lado del usuario.

Como ya se habrán imaginado, estamos hablando de los formularios. 📝🤓👌 Los siguientes puntos son una guía, su sitio puede requerir otras validaciones así que ajústense según lo necesiten.

Vamos con esas consignas:

- **Registro de usuarios**
  - Nombre y apellido
    - Obligatorio.
    - Deberá tener al menos 2 caracteres.
  - Email
    - Obligatorio.
    - Deberá ser un formato de e-mail válido.
    - No puede repetirse con los e-mails ya registrados.
  - Contraseña
    - Obligatoria.
    - Deberá tener al menos 8 caracteres.
    - (Opcional) → Deberá tener letras mayúsculas, minúsculas, un número y un carácter especial.
  - Imagen
    - Deberá ser un archivo válido (JPG, JPEG, PNG, GIF).
- **Login de usuarios** (este ya lo deberían tener de sprints anteriores 😊☝)
  - Email
    - Obligatorio.
    - Deberá ser válido.
    - Deberá existir en base.
  - Contraseña
    - Obligatoria.
    - Deberá coincidir con la existente en base.

- **Creación y modificación de productos**
    - Nombre
        - Obligatorio.
        - Deberá tener al menos 5 caracteres.
    - Descripción
        - Deberá tener al menos 20 caracteres.
    - Imagen
        - Deberá ser un archivo válido (JPG, JPEG, PNG, GIF).
    - (Opcional) Tablas secundarias
        - Verificar que los valores existan en base. Es decir, que los valores de talles, colores, etc. que lleguen sean válidos en la base.
- **(Opcional) Resto de los formularios del sitio**

**Entregable:** implementación de Express Validator en las rutas que reciban formularios.

## 4. Validaciones del front-end

Ya lo dijimos muchas veces y vale repetir: si bien no podemos confiar en las validaciones del front-end porque nuestro visitante puede deshabilitarlas, en el 99% de los casos funcionarán como esperamos.

Nuestro objetivo acá es doble:

1. Por un lado, vamos a validar los datos antes de enviarlos para evitarle procesos innecesarios al servidor.
2. Por el otro, vamos a darle feedback inmediato y de calidad a nuestro visitante. Esto se traduce en una mucho mejor experiencia (UX) dentro de nuestro sitio.

Vamos con esas consignas:

- **Registro de usuarios**
  - Nombre y apellido
    - Obligatorio.
    - Deberá tener al menos 2 caracteres.
  - Email
    - Obligatorio.
    - Deberá ser válido.
    - (Opcional) → No puede repetirse con los e-mails ya registrados.
  - Contraseña
    - Obligatoria.
    - Deberá tener al menos 8 caracteres.
    - (Opcional) → Deberá tener letras mayúsculas, minúsculas, un número y un carácter especial.
  - Imagen
    - Deberá ser un archivo válido (JPG, JPEG, PNG, GIF).
- **Login de usuarios** (este ya lo deberían tener de sprints anteriores 😊☝)
  - Email
    - Obligatorio.
    - Deberá ser válido.
    - (Opcional) → Debe existir en la base.
  - Contraseña
    - Obligatoria.
- **Creación y modificación de productos**
  - Nombre
    - Obligatorio.
    - Deberá tener al menos 5 caracteres.
  - Descripción
    - Deberá tener al menos 20 caracteres.
  - Imagen
    - Deberá ser un archivo válido (JPG, JPEG, PNG, GIF).
- **(Opcional) Resto de los formularios del sitio**

**Entregable:** implementación de ***validator.js*** (documentación/descarga) o validaciones "***custom***" en los formularios mencionados.

## > Resumen de entregables

- Archivo **retro.md** con el resultado de la retrospectiva.
- (Opcional) Archivo **daily.md** con sus opiniones sobre las dailys/weeklys
- Tablero de trabajo actualizado.
- Validaciones del back-end con **Express Validator**:
    - Registro de usuarios.
    - Login de usuarios.
    - Creación y modificación de productos.
    - (Opcional) Resto de los formularios del sitio.
- Validaciones del front-end con **JavaScript**:
    - Registro de usuarios.
    - Login de usuarios.
    - Creación y modificación de productos.
    - (Opcional) Resto de los formularios del sitio.

## > Cierre

Las validaciones son vitales si queremos que nuestro sitio funcione bien a lo largo del tiempo. Ya sea por error, o por mala intención, siempre llegarán datos que son inválidos o que de alguna manera son incorrectos. 👾 👾 👾 Si nos descuidamos y dejamos que pasen, va a ser muy difícil que podamos corregir el problema sin terminar con un gran dolor de cabeza. 🔥 🚒 🔥

Poquito a poquito y sin darse cuenta están terminando su sitio…

- 🔎 Investigación → Lista ✅
- 🎨 Diseño → Listo ✅
- 🌐 HTML + CSS → Listo ✅
- ⚙️ Vistas dinámicas → Listas ✅
- 🗂️ Base de datos → Lista ✅
- 👥 Usuarios con su login y registro → Listo ✅
- 📦 Productos con su CRUD completo → Listo ✅
- 🛡️ Validaciones → En camino ⏳

¡Uf! ¡Cuánto camino recorrido!

Mirando atrás hay que reconocer sin duda que no ha sido un camino fácil 🚪, piensen en todos los conceptos que hemos visto en tan solo algunos meses. Seguro a esta altura tengan la cabeza dando vueltas 🤯 y sueñen con objetos literales en lugar de ovejitas. 🐑🐑🤖🤖

Como dice el dicho: “Nada que valga la pena es fácil de conseguir”, así que felicitaciones por llegar hasta aquí. **¡Vamos por este penúltimo sprint!**

# Programación Web Full Stack

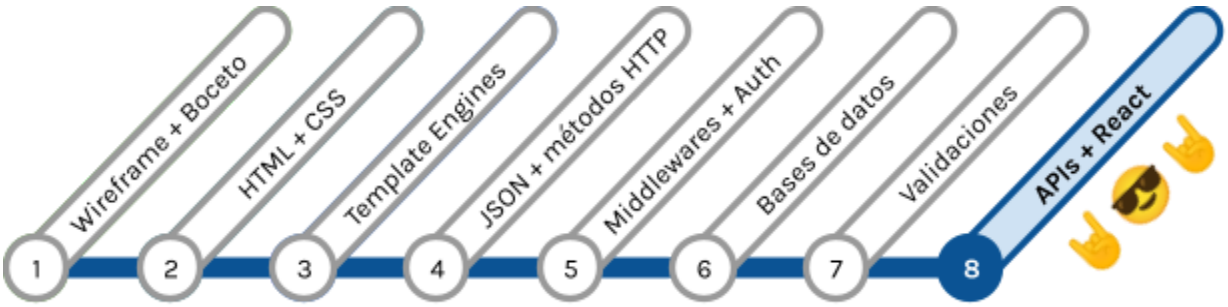
**Trabajo integrador - S8 - M10C24**

# Trabajo Integrador - Sprint 8

## > Introducción

Parece mentira que hayamos llegado al final. 🤯 Hace algunos meses nomás empezamos con la consola a descubrir de qué iba esto de Node.js. 💻🔍 Después vimos HTML, CSS y los motores de plantillas con EJS. 📋➡️📄📄📄 También aprendimos cómo podíamos hacer que un usuario se loguee 👥🔑 y permanezca logueado con Middlewares, Sessions y Cookies. 🍪🤤 Luego nos tocó validar tanto en el back-end como en el front-end. 🔒👮 A lo último vimos que podemos enviar y recibir datos de manera eficiente a través de las APIs 🔵↔️🔵 y que React nos permite trabajar el front-end de una manera modularizada y súper eficiente. 🧙✨

Con estos últimos dos conceptos vamos a trabajar este sprint final. **¡A darle átomos!** 🦸⚛️🦸



## > Requisitos

1. **Base de datos de usuarios y productos:** en este sprint vamos a trabajar con los datos que ya existen en nuestro sitio, por eso necesitamos la base completa y funcionando.

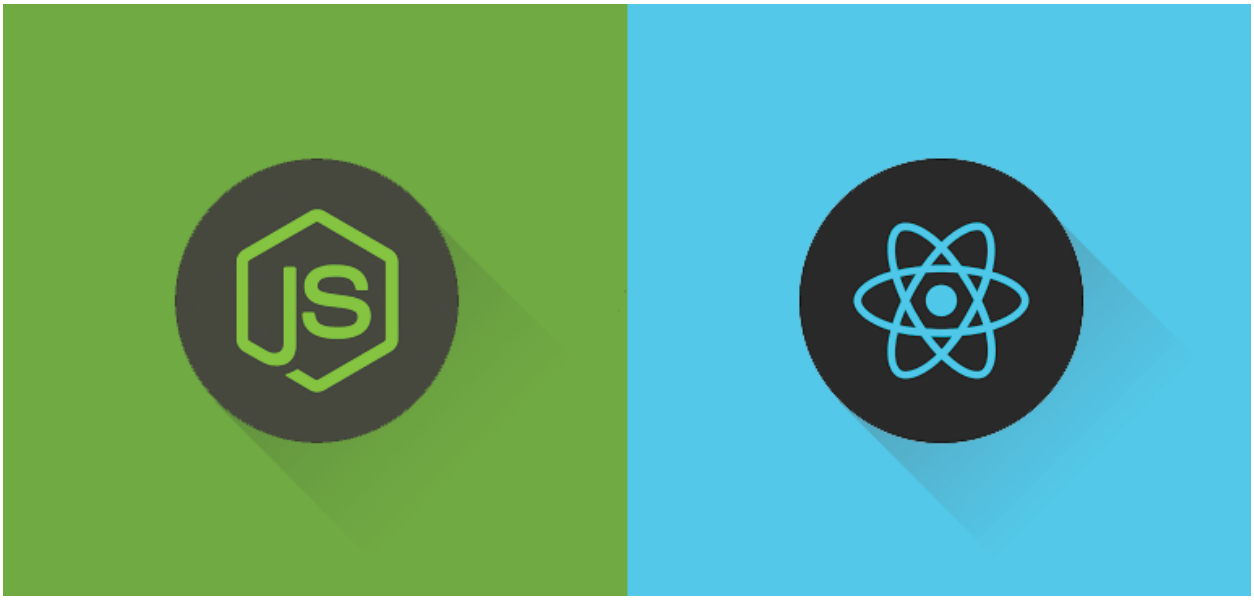
## > Objetivo

Al igual que el sprint anterior, este se divide en dos partes:

En la **primera parte** van a estar armando una **API de usuarios y de productos** que exponga los datos más importantes de su aplicación.

En la **segunda parte** van a estar armando un **dashboard hecho en React** que consuma los datos de la API y muestre de manera resumida las principales métricas de su negocio.

Un dashboard nos permite ver a simple vista si todo está funcionando bien. Pueden pensarlo como el tablero de un auto, donde toda la información que necesitamos está a simple vista.



# > Metodologías ágiles

¡Vamos con la última vuelta de retro y planificación! 📝🤓👌✨

Como siempre les recomendamos que no se salteen este paso, es muy importante. 😊☝️

## La retrospectiva ¡con yapa!

Implementen nuevamente la dinámica de la **estrella de mar**, resaltando aquello que hay que:

1. Comenzar a hacer.
2. Hacer más.
3. Continuar haciendo.
4. Hacer menos.
5. Dejar de hacer.

Ya estamos casi al final del viaje y, cuando terminen este sprint, les sugerimos que prueben la retrospectiva de **línea de tiempo**. A resumidas cuentas, esta consiste en dibujar una línea de tiempo que incluya todos los sprints y en ella ubicar post-its con cosas positivas (arriba de la línea) y negativas (debajo de la línea) que hayan ocurrido en cada etapa. Al terminar tendrán un resumen gráfico de todo el proceso.

## El tablero de trabajo

Momento de **reiniciar el tablero** para acomodar este sprint de bases de datos. Como siempre, lo que haya quedado del anterior sprint se suma al actual.

Recuerden que durante la planificación es importante:

- Debatir cada tarea en conjunto para asegurarse de que no haya dudas sobre su alcance (hasta dónde van a hacer) y sobre cómo van a resolverla.
- Estimar la dificultad de la tarea y si esta requiere de que alguna otra tarea esté terminada antes de poder iniciarla (para determinar el orden).
- Asignar tentativamente los responsables de cada una de ellas.

### (Opcional) La reunión daily o weekly

La **daily standup** es una reunión, que en los equipos de **Scrum** se realiza todos los días, donde cada integrante habla como máximo 3 minutos de 3 temas puntuales

- Qué hizo ayer.
- Si se encontró con algún impedimento.
- Qué va a hacer hoy.

El formato está pensado para ser rápido y liviano, solo queremos la información más importante de las tareas y los impedimentos.

**Importante:** no es necesario que esto lo hagan todos los días, al menos una vez por semana sería ideal.

## > Consignas

### Planificación y trabajo en equipo

#### 1. Realizar un breve retrospectiva

Piensen qué hicieron bien el sprint anterior, qué hicieron mal, qué deberían empezar a hacer, qué deberían dejar de hacer, sigan esta dinámica.

**Entregable:** Actualizar el archivo retro.md con las principales conclusiones de la retro del séptimo sprint.

#### 2. Actualizar el tablero de trabajo

Discutan las tareas que se desprenden de este documento, determinen en qué orden deberán realizarlas, asignen integrantes a cada tarea.

**Entregable:** link al documento o plataforma que utilicen para organizar el trabajo.

### 3. (Opcional) Implementar daily/weekly standups

Cada equipo habla como máximo 3 minutos de 3 temas puntuales:

- Qué hizo ayer.
- Si se encontró con algún impedimento.
- Qué va a hacer hoy.

**Entregable:** archivo daily.md con sus principales impresiones (positivas, neutras o negativas) sobre la utilidad de esta ceremonia.

## APIs y dashboard

### 4. API de usuarios

Nuestra API de usuarios va a proveernos de dos endpoints muy importantes. El primero nos entregará la lista completa de usuarios y el segundo nos permitirá consultar los detalles de un usuario en particular.

Vamos con esas consignas:

- **api/users/**
  - Deberá devolver un objeto literal con la siguiente estructura:
    - count → cantidad total de usuarios en la base.
    - users → array con la colección de usuarios, cada uno con:
      - id
      - name
      - email
      - detail → URL para obtener el detalle.
- **api/users/:id**
  - Deberá devolver un objeto literal con la siguiente estructura:
    - Una propiedad por cada campo en base.
    - Una URL para la imagen de perfil (para mostrar la imagen).
    - Sin información sensible (ej: password y categoría).

**Entregable:** URL funcionales devolviendo datos de usuarios en formato JSON.

## 5. API de productos

Nuestra API de productos será muy similar. Sus dos endpoints entregarán la lista completa de productos y el detalle de un producto en particular.

Vamos con esas consignas:

- **api/products/**
  - Deberá devolver un objeto literal con la siguiente estructura:
    - count → cantidad total de productos en la base.
    - countByCategory → objeto literal con una propiedad por categoría con el total de productos.
    - products → array con la colección de productos, cada uno con:
      - id
      - name
      - description
      - un array con principal relación de uno a muchos (ej: categories).
      - detail → URL para obtener el detalle.
- **api/products/:id**
  - Deberá devolver un objeto literal con la siguiente estructura:
    - una propiedad por cada campo en base.
    - un array por cada relación de uno a muchos (categories, colors, sizes, etc).
    - Una URL para la imagen del producto (para mostrar la imagen).

**Entregable:** URL funcionales devolviendo datos de productos en formato JSON.

## 6. (Opcional) Paginado

Agregar a los endpoints de listado, la posibilidad de paginar los resultados.

- **api/users/**
- **api/products/**
    - 10 resultados por página (recuerden limit y offset 😉☝).
    - next → URL a la próxima página (si corresponde).
    - previous → URL a la página previa (si corresponde).

Pueden tomar de referencia la API de Star Wars:

➔ https://swapi.co/
➔ https://swapi.co/api/people/?page=3

## 7. Dashboard en React

Ya tenemos nuestra fuente de datos y ahora solo queda consumirlas para darle vida a nuestro dashboard.

Para este punto les recomendamos que partan de los archivos que les compartimos durante las ejercitaciones presenciales de React.

El dashboard deberá contener al menos:

- 3 a 6 paneles simples con los siguientes totales:
    - Total de productos
    - Total de usuarios
    - Total de categorías
- Panel de detalle de último producto o usuario creado.
- Panel de categorías con el total de productos de cada una.
- Panel con el listado de productos.

(Opcional) Funcionalidades adicionales:

- Total de productos vendidos / total de ventas.
- Últimos 5 productos vendidos / los 5 más vendidos.
- Vista de creación de productos como Administrador.
- Vista de edición de productos como Administrador.
- Opción de eliminar productos como Administrador.

## > Resumen de entregables

- ★ Archivo **retro.md** con el resultado de la retrospectiva.
- ★ (Opcional) Archivo **daily.md** con sus opiniones sobre las dailies/weeklies.
- ★ Tablero de trabajo actualizado.
- ★ Endpoints de **usuarios**:
  - Listado de usuarios.
    - (Opcional) Paginado.
  - Detalle de usuario.
- ★ Endpoints de **productos**:
  - Listado de productos.
    - (Opcional) Paginado.
  - Detalle de producto.
- ★ Dashboard del sitio hecho en React:
  - 3 a 6 paneles simples con totales.
  - Panel de detalle de último producto o usuario creado.
  - Panel de categorías con el total de productos de cada una.
  - Panel con el listado de productos.
  - (Opcional) Funcionalidades adicionales.

## > Cierre

Decíamos en el anterior sprint que “nada que valga la pena es fácil de conseguir” y, si llegaron hasta aquí, no queda más que aplaudirlos. 👋😊 Pueden considerarse con orgullo Fullstack Developers. 💻🥇🎉

La programación se parece mucho a la carpintería y eso es porque si bien es posible entender cómo se arma un mueble en la teoría 🪑🧠, no es hasta que lo armamos que realmente empezamos a internalizar cada pequeño detalle que supone construir ese objeto. 🪑🤯

No solo eso, así como la carpintería, la programación es un oficio que requiere mucha práctica para llegar a ser realmente hábil. En nuestra humilde opinión, hay pocas cosas más satisfactorias que encontrar un oficio al que dedicarse con empeño. 👩‍💻💖👩‍💻

Esperamos que le hayan agarrado el gusto al mundo de la programación y que este sea el comienzo de una gran aventura. 🏞️🏕️🚶🚶