**🏠 Introducción a React**

**👋 Introducción**

**¡Te damos la bienvenida al curso de React!**En este primer paso, nos sumergiremos en la teoría esencial para la creación de una aplicación de React utilizando Vite.

Durante esta sección, nos concentraremos en desarrollar una aplicación de página única utilizando React, una de las bibliotecas más populares y modernas de JavaScript. También exploraremos el funcionamiento de la navegación en este tipo de aplicaciones, así como la definición de rutas para garantizar una experiencia fluida y bien estructurada. ¡Prepárate para adentrarte en el fascinante mundo de React y las SPAs!

¡Empecemos! 🚀

**📖 Material Descargable**

Aquí encontrarás el material descargable correspondiente que te dará las bases para tus próximos pasos.

👉[**Teoría Introducción a React**](https://drive.google.com/file/d/1Z4BCHAJd79Aum_O9x5PX9zudDNRHhtJE/view?usp=drive_link)

**🧰Instalación de Node.js**

Para comenzar este recorrido, vamos a instalar  node.js

Descargar el instalador ,  puedes ingresar al siguiente enlace **👉**[**Instalador de Node.js**](https://nodejs.org/en)

**👉**Si necesitas ayuda para iniciar la descarga, te dejamos un video para guiarte[Intro React | React | Egg](https://youtu.be/vNgGeevWUC0?feature=shared)

**🧰Herramienta para el siguiente paso**

En tu curso de JavaScript, gestionaste mejoras en tu proyecto “Tienda Virtual”, donde comenzaste a conocer como dar dinamismo a tus HTML.

En este curso, usaremos como base el mismo proyecto, pero te proveemos  el proyecto a utilizar, ya que, respetando lo aprendido en tu curso anterior, tiene mayor cantidad de datos y te asegurarás de tener la estructura y componentes necesarios para comenzar tu práctica con React.   **👉**[**Descargar Proyecto App**](https://drive.google.com/file/d/1g21wJhEYVDUjQUnpnAgJhlemhgF9DGhZ/view?usp=drive_link)

**✅ Checklist**

Asegúrate de:

* Leer el material teórico de Introducción a React
* Tener Visual Studio Code instalado como editor de código recomendado.
* Tener  Node.JS instalado.
* Haber descargado el proyecto provisto.

*💡 Revisar estos puntos es muy importante dado que ayudarán a que todos puedan tener la mejor experiencia posible y logren resolver los desafíos correspondiente*s.

**🧰Material Complementario**

Te proporcionamos acceso a un recurso de aprendizaje opcional que te brindará la oportunidad de profundizar en diversos aspectos:

* [**“La importancia de las convenciones de codificación”**](https://geeks.ms/jorge/2019/03/24/la-importancia-de-las-convenciones-de-codificacion-pascalcase-camelcase-snake_case-y-kebab-case/)

**👣 Creación del Proyecto**

En este paso veremos cómo crear una aplicación de React, valiéndonos de Vite. Aprenderemos a estructurar las carpetas según los tipos de componentes (views & components) y crearemos la vista principal de la aplicación.

**📖 Material Audiovisual**

Si lo deseas, te invitamos a mirar este video que amplía como se estructura un proyecto básico.

**💡*Conocer de manera fluida como es la arquitectura de un proyecto de React con js, y el fin de uso de cada uno de sus archivos, te permitirá trabajar con mayor facilidad con cada elemento.***

<https://www.youtube.com/watch?v=MPdRvXLzBhI>

**✏️ Actividad : Investigar la versión de JavaScript "puro/vanilla"**

Explorar los archivos de la aplicación del proyecto provisto en el paso anterior para analizar las distintas vistas y sus partes, comprender las funcionalidades de cada página y examinar el desarrollo de los estilos aplicados. Este proceso es crucial, dado que la construcción de la Single Page Application (SPA) se basará en el análisis de estos archivos.

**❗*Recuerda que en el paso anterior, te dimos acceso a un link de descarga del proyecto base a utilizar en este curso.***

**✏️ Actividad : Crear una aplicación de React con Vite**

Abrimos una terminal (CMD) en la carpeta donde queremos crear la aplicación. Luego ejecutaremos el siguiente comando:

npm create vite@latest virtual-store -- --template react-ts

La consola nos informa que debemos ingresar a la carpeta de la aplicación ejecutando en consola:

cd virtual-store

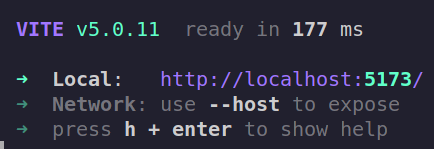
Luego debemos instalar los módulos necesarios para que la app funcione, con el comando:

npm install

Inicializamos la aplicación de React con:

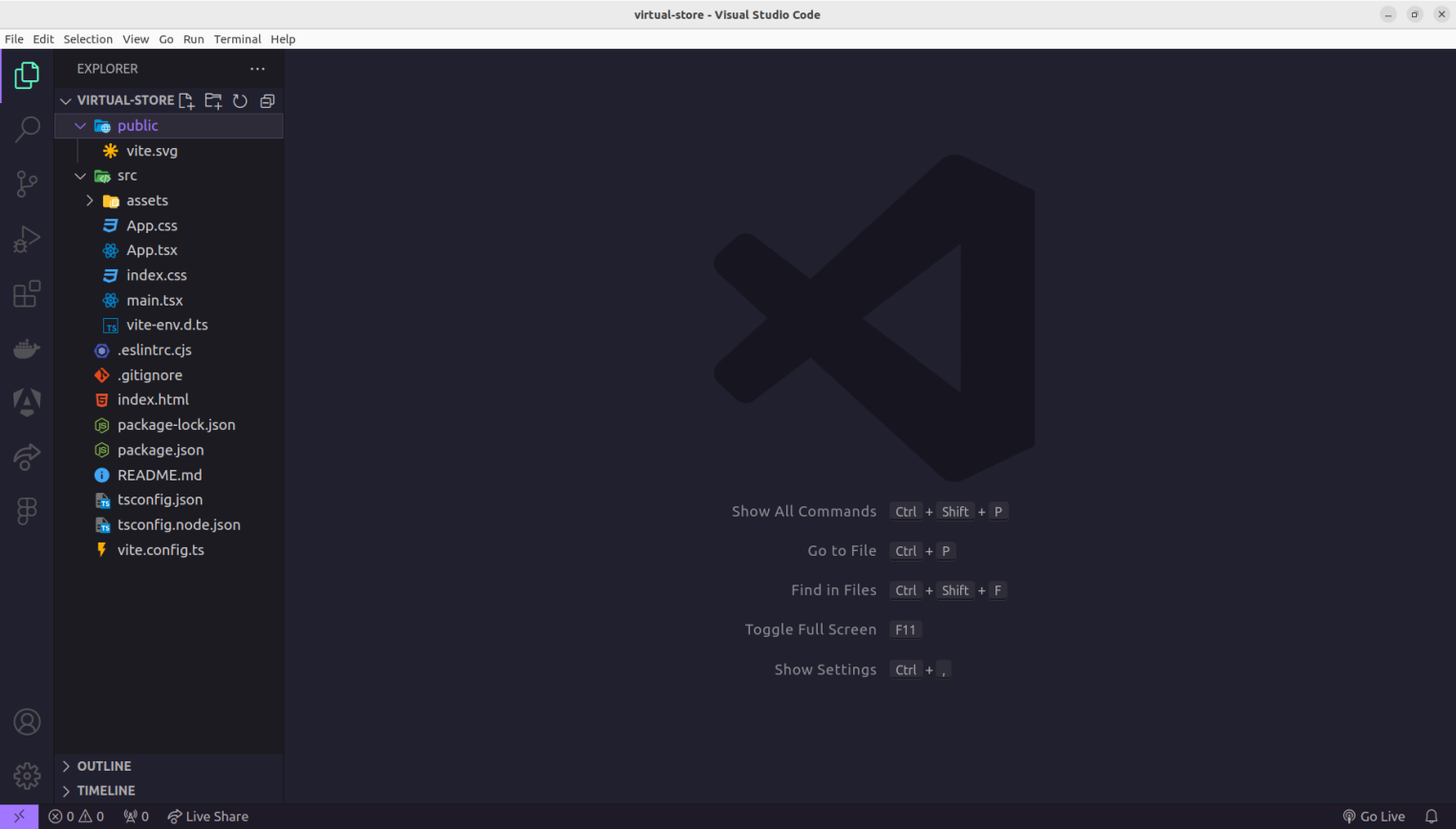
npm run dev

En este punto deberías ver en la consola, algo similar a esto:



Si accedemos al link que nos informa la consola, podemos ver en el navegador la interfaz de React generada por Vite.

Para el siguiente paso necesitaremos que abras VSC o tu editor preferido en la carpeta de la aplicación. Verás las siguientes carpetas y archivos generados por Vite:



**✏️ Actividad: Crear la estructura de carpetas de la aplicación**

Dentro de la carpeta /src, crearemos dos sub-carpetas:

* views: para todos los archivos que renderizan vistas
* components: para todos los archivos que renderizan componentes.

**✏️ Actividad: Limpiar archivos y carpetas**

Al seguir estos pasos, optimizarás y organizarás el código de tu aplicación, eliminando archivos y contenido innecesario para preparar el terreno para el desarrollo de la SPA.

1. **Archivo /tsconfig.json:**
   * Dirígete al apartado de /\* Linting \*/.
   * Agrega la propiedad noImplicitAny con el valor false de la siguiente manera:

"noImplicitAny": false,

1. **Carpeta /public:**
   * Elimina todos los archivos de la carpeta /public.
   * Copia todos los recursos de la carpeta assets del curso de JavaScript 1.
2. **Archivo /src/index.css:**
   * Elimina el contenido actual.
   * Busca el archivo styles.css del curso JavaScript 1.
   * Copia los estilos de body, html, de la Estructura del sitio, de la Hero section y del Footer.
   * Pega estos estilos en /src/index.css. A modo de ejemplo:
3. **Archivo /src/App.css:**
   * Elimina este archivo.
4. **Carpeta /src/assets:**
   * Elimina esta carpeta.
5. **Archivo /src/App.tsx:**
   * Elimina todas las importaciones.
   * Deja el componente "vacío", que solo contenga el return con la etiqueta **fragment** <></>.

**✏️ Actividad: Crear el componente de vista principal**

* En la carpeta /src/views crea el archivo Home.tsx.
* En este archivo, define la función Home que renderizará la vista principal.

/\* /src/views/Home.tsx \*/

function Home() {

  return <> ... </>;

}

export default Home;

* Busca el archivo index.html del curso JavaScript 1.
* Copia todas las etiquetas del contenido del cuerpo <body> de la vista.
* En el archivo Home.tsx, pega el contenido de la vista en el return del componente Home.
* Modifica los cierres de etiquetas correspondientes (como img o input), sustituye los atributos for por htmlFor que correspondan y cambia los atributos class existentes por className.
* Actualiza las rutas de las imágenes para acceder directamente desde la carpeta public. Ejemplo:

<!-- Antes -->

<img src="../assets/tiendamia-logo.svg" width="218" alt="Logo store" />

<!-- Después -->

<img src="/tiendamia-logo.svg" width="218" alt="Logo store" />

* Dentro de la carpeta /src/views, crea el archivo Home.module.css.
* Busca el archivo styles.css del curso JavaScript 1.
* Copia los estilos necesarios para la vista (Bloque de productos y Card de producto).
* Pega los estilos copiados en el archivo Home.module.css.
* En la hoja de estilos, verifica y actualiza las rutas de las imágenes para acceder directamente desde la carpeta public.
* Por último, en el archivo Home.tsx, importa el archivo de estilos y modificar las clases correspondientes.

/\* /src/views/Home.tsx \*/

import styles from "./Home.module.css"

function Home() {

  return <>

    <header className={styles.header}> ... </header>

    <section className={styles.section}> ... </section>

    <main className={styles...}> ... </main>

    <footer className={...}> ... </footer>

  </>;

}

export default Home;

**✏️ Actividad: Mostrar la vista principal**

Para visualizar la vista del componente Home.tsx debemos:

* Importar Home.tsx en App.tsx en las primeras líneas del archivo.
* Retornar el componente de vista que queremos visualizar:

/\* /src/App.tsx \*/

import "./views/Home.tsx"

function App() {

  return  <Home />;

}

export default App;

* Verificar el correcto renderizado de la vista Home en “/”.

# 👣 Componentización de Home

Este paso del curso está dedicado a uno de los conceptos fundamentales en React: **los componentes**.

Los componentes son los bloques de construcción de cualquier aplicación de React. Aprenderás a diseñarlos, reutilizarlos y gestionar su estado para crear interfaces dinámicas y eficientes. Este conocimiento es crucial para avanzar en el desarrollo de aplicaciones más complejas y personalizadas.

## 📖 Material Audiovisual

Si lo deseas, te invitamos a mirar este video que amplía el uso de componentes

**❗*No es necesario que crees el ejemplo mostrado en el video, lo importante es que comprendas de manera clara lo aprendido para que  puedas aplicar lo visto en las actividades posteriores.***

<https://www.youtube.com/watch?v=idflvZZTx30>

### ✏️ Actividad: Crear un componente para la Barra de navegación

Esta actividad consiste en crear un componente NavBar, lo que te permitirá experimentar con la modularización de una vista, dividiéndola en componentes más pequeños y manejables. Esta técnica es crucial para entender la transmisión de datos entre componentes y su visualización.

* **Crear el componente NavBar**:
  + Crear el archivo NavBar.tsx en la carpeta /src/components
  + Definir la estructura básica de cualquier componente.
  + Cortar el elemento <header> del componente Home.
  + Pegar el contenido dentro del retorno return de NavBar.
* **Definir los estilos de NavBar**:
  + Crear el archivo NavBar.module.css en la carpeta /src/components
  + Mover los estilos relevantes desde /index.css a /src/components/NavBar.module.css.
  + Importar los estilos en el componente NavBar.tsx y modificar las clases correspondientes.

/\* /src/components/NavBar.tsx \*/

import styles from "./NavBar.css"

function NavBar() {

  return <header className={styles...}> ... </header>

}

export default NavBar;

* **Implementar el componente NavBar en la vista**:
  + Importar NavBar en Home.
  + Llamar al componente <NavBar /> donde estaba el <header>.

/\* /src/views/Home.tsx \*/

import NavBar from "../components/NavBar";

function Home() {

  return <>

    <NavBar />

    <section> ... </section>

    <main> ... </main>

    <footer> ... </footer>

  </>;

}

export default Home;

* **Verificar el correcto renderizado de la vista.**

### ✏️ Actividad: Reutilizar Componentes

La reutilización es una práctica esencial en React. Crearás el componente NavButton para entender cómo las propiedades props permiten la configuración dinámica de componentes.

1. **Crear el componente NavButton**:
   * Crear el archivo NavButton.tsx en la carpeta /src/components
   * Definir la estructura básica de cualquier componente.
   * Cortar algún <a> de los enlaces de NavBar.
   * Pegar el contenido dentro del retorno return de NavButton.
2. **Definir los estilos de NavButton**:
   * Crear el archivo NavButton.module.css en la carpeta /src/components
   * Mover los estilos relevantes desde /NavBar.css a /src/components/NavButton.module.css.
   * Importar los estilos en el componente NavButton.tsx y modificar las clases correspondientes.

/\* /src/components/NavButton.tsx \*/

import styles from "./NavButton.module.css"

function NavButton() {

  return <a className={...}> ... </a>

}

export default NavButton;

1. **Implementar el componente NavButton en la vista**:
   * Importar NavButton en Home.
   * Llamar al componente <NavButton /> donde estaba el <a>.
   * Analizar y definir las propiedades dinámicas de NavButton, por ejemplo, title (para el texto del enlace) y href (para el link).
   * Configurar los atributos de NavButton con estas propiedades para personalizar cada link. A modo de ejemplo:

<NavButton  title="Ofertas de la semana" href="/ofertas.html" />

<NavButton  title="Cómo comprar" href="/como.html" />

1. **Utilizar las props de NavButton**:
   * Con esta configuración de atributos, el componente NavButton recibe un objeto con estas propiedades, por lo tanto, es necesario desestructurarlo para poder utilizar estos datos interpolando con {}:

/\* /src/components/NavButton.tsx \*/

import styles from "./NavButton.module.css"

function NavButton(props) {

  const { title, href } = props

  return <a className={...} href={href}> {title} </a>

}

export default NavButton;

1. **Verificar el correcto renderizado de la vista.**

### ✏️ Actividad Complementaria

Se sugiere ampliar la componentización de la vista con, al menos:

1. Un componente Hero, para la etiqueta section del componente Home.
2. Un componente Footer, para la etiqueta footer del componente Home.
3. Un componente ColumnFooter, para cada columna del componente Footer.

Evaluar si es necesario pasarles propiedades.