**PROCESO DE GESTIÓN DE FORMACIÓN PROFESIONAL INTEGRAL**

**FORMATO GUÍA DE APRENDIZAJE**

**1. IDENTIFICACIÓN DE LA GUIA DE APRENDIZAJE**

* **Denominación del Programa de Formación:** Tecnólogo en Análisis y Desarrollo de Software.
* **Código del Programa de Formación:** 228118.
* **Nombre del Proyecto Formativo:** Construcción de software integrador de tecnologías orientadas a servicios.
* **Fase del Proyecto:** Construcción.
* **Actividad de Proyecto Formativo:** Desarrollar la estructura de datos y la interfaz de usuario del sistema de información.
* **Competencia:**
  + 220501096 - desarrollar la solución de software de acuerdo con el diseño y metodologías de desarrollo.
* **Resultados de Aprendizaje:**
  + 220501096-03-Crear componentes front-end del software de acuerdo con el diseño.
* **Duración de la Guía de Aprendizaje (horas):** 18 Horas.

**2. PRESENTACIÓN**

Estimado aprendiz, el SENA le extiende una cordial bienvenida a la guía de aprendizaje que corresponde al resultado de aprendizaje técnico: **Crear componentes front-end del software de acuerdo con el diseño**. Este resultado se enfoca en el desarrollo de habilidades fundamentales para la construcción de interfaces gráficas interactivas, utilizando herramientas modernas del ecosistema de JavaScript, especialmente el framework **React**. A lo largo de esta guía, usted abordará temas esenciales como la **gestión de dependencias con npm**, el uso de funcionalidades avanzadas de **JavaScript moderno** (tales como destructuring, arrow functions, y promesas con async/await), y la comprensión de los principios básicos de **React**, incluyendo la creación de componentes, el manejo del estado con hooks, y la organización del entorno de desarrollo.

Igualmente, se incluirán elementos claves del diseño visual, como los fundamentos de **CSS**, técnicas de **diseño responsivo con Flexbox y Grid**, y estrategias para la **estilización de componentes**. También se trabajará en la conexión de las interfaces con datos externos, mediante el consumo de **APIs**, lo cual constituye un paso esencial en la construcción de aplicaciones web completas. El desarrollo de las actividades propuestas en esta guía contará con el acompañamiento permanente del instructor, quien le brindará orientación conceptual, técnica y metodológica, necesaria para la apropiación de los saberes y el cumplimiento de los propósitos de aprendizaje. Este proceso estará apoyado en la práctica constante, el análisis de casos, y la resolución de problemas reales del entorno digital. Por consiguiente, se presentan a continuación cada una de las acciones de aprendizaje que le permitirán fortalecer sus competencias en el desarrollo front-end, con base en estándares actuales de la industria y prácticas de codificación modernas.

**3. FORMULACIÓN DE LAS ACTIVIDADES DE APRENDIZAJE**

* **Descripción de la(s) Actividad(es)**

**3.1 Actividades de reflexión inicial:**

**GA1-220501096-03-AA1-EV01-Estándares y características de interfaces gráficas**

**Descripción de la actividad**:

El aprendiz reflexionará sobre sus experiencias previas al interactuar con aplicaciones o sitios web, identificando elementos de la interfaz gráfica que le hayan facilitado o dificultado la navegación. A partir de esta reflexión, analizará la importancia de los estándares de diseño en la creación de interfaces gráficas. Finalmente, documentará sus conclusiones en un informe breve.

**Instrucciones para el aprendiz:**

* + - 1. **Revisa:** los video sugeridos por el instructor en el material de apoyo.
      2. **Reflexión**: Haz memoria sobre las aplicaciones o sitios web que has usado recientemente. Selecciona dos que te hayan proporcionado una buena experiencia de usuario y dos que te hayan generado dificultades. Reflexiona sobre los aspectos visuales y de navegación de cada una.
      3. **Investigación**: Investiga los estándares de diseño de interfaces gráficas, como Material Design y Human Interface Guidelines. Identifica qué principios o estándares pudieron haber sido aplicados o ignorados en los ejemplos que seleccionaste.
      4. **Informe**: Redacta un informe de una página donde:

Describas los elementos de la interfaz gráfica que facilitaron o dificultaron tu experiencia en las aplicaciones seleccionadas.

Relaciona esos elementos con los estándares de diseño investigados.

Concluye con tu opinión sobre la importancia de adoptar estándares en el diseño de interfaces gráficas.

5. Crea un video 2 min que sustente los elementos identificado en su informe este video debe ser publicado en youtube y compartir el link dentro del informe.

**Ambiente requerido:**

* + - * + Acceso a una computadora o dispositivo móvil con conexión a internet.
        + Espacio de trabajo tranquilo para la reflexión y el análisis.

**Estrategias o técnicas didácticas activas:**

**Materiales de formación**

**Material de apoyo:**

* + - * + **Interfaces gráficas de usuario y experiencia de usuario – Introducción:**

<https://www.youtube.com/watch?v=DDj97-xzxx8>

* + - * + **Construcción de prototipos:** <https://www.youtube.com/watch?v=rVxFBN0q0tk>
        + **Diseño de interfaces Web:** <https://www.youtube.com/watch?v=1fT5SGzQDAM>

**Duración de la actividad:** 3 horas.

**3.2 Actividades de contextualización e identificación de conocimientos necesarios para el aprendizaje:**

**GA1-220501096-03-AA1-EV02-Gestion de dependencias de código según lenguaje de programación**

**seleccionado(Javascript).**

**Descripción de la actividad:**

El aprendiz de forma individual el aprendiz, realizara la definicion de los siguientes términos sobre

los procesos de gestion de dependencias en Javascript según los siguientes interrogantes:

* ¿Que es un gestor dependencias de codigo?
* ¿Qué es npm?
* ¿Para qué se utiliza principalmente npm?
* ¿Qué es el versionado semántico?
* Como está especificado el Versionado Semántico.
* Qué son las dependencias locales
* Qué son las dependencias de desarrollo
* Qué son las dependencias globales
* ¿Que es el archivo package.json y que apartados tiene y cual es su utilidad?
* ¿Qué es el archivo package-lock.json y que utilidad tiene?
* que es la carpeta node\_modules en un proyecto de npm
* Realiza una Cheat sheet de los siguientes procesos (ver ejemplo:https://pin.it/2TauqeogV):
  + como inicializar un proyecto de npm.
  + Cómo instalar dependencias locales
  + Cómo instalar dependencias de desarrollo
  + Cómo instalar dependencias globales
  + Cómo visualizar los dependencias instaladas
  + Cómo instalar una versión específica de un paquete
  + Cómo crear un comando en tu proyecto(scripts en el package.json )
  + Cómo actualizar dependencias
  + Cómo eliminar paquetes
  + Cómo actualizar **node\_modules**

Ambiente requerido:

Estrategias o técnicas didácticas activas:

Materiales de formación

Material de apoyo:

Evidencias de aprendizaje:

* ***Productos para entregar:***Documento en el procesador de texto(word) usando las normas Apa 7th edición, donde organice los puntos del proceso de conceptualización de la actividad. el documento debe tener como mínimo una portada, introducción, Tabla de contenidos, Desarrollo de los puntos de conceptualización, selección de herramientas de versionamiento a utilizar, conclusiones y referencias Bibliograficas.
* ***Productos para entregar:*****Cheat sheet**, documento visual que describa los procesos de gestion de pendencias.

Instrumentos de evaluación:

Duración de la actividad: 6 horas.

**3.3 Actividades de apropiación:**

**GA1-220501096-03-AA1-EV03 Manejo Práctico del gestor de dependencias NPM.**

**Descripción de la actividad:** Los aprendices de manera individual y según las guias explicativas suministradas por el instructor manejaran npm para la gestión de proyectos en Node.js y javascript.

Para realizar esta actividad se debe seguir las siguientes orientaciones:

* + - * Revisar el documento:
        + **Instalacion de NodeJs:** <https://educated-show-144.notion.site/Guia-de-instalaci-n-de-NodeJs-1f74671e02a180cd9284fd1143d69a3e?pvs=4>
        + **Gestión de dependencias con npm:** <https://educated-show-144.notion.site/Manejo-de-dependencias-en-npm-1f74671e02a180059863c5eb9af8229b?pvs=4>
        + **Taller practico de npm:** Replicar los pasos definidos en la guía Manejo de dependencias en npm.

* + - * Generar un informe tipo Bitácora que indique paso a paso los procesos de gestión con npm según las indicaciones del documento **Taller practico de npm**.

**Evidencias de aprendizaje:**

* + - * Bitácora de desarrollo del taller práctico, con portada, introducción, documentación de las 5 secciones indiciada en el taller práctico y conclusiones.

**Ambiente requerido:** Equipos de cómputo, conexión a Internet **Estrategias o técnicas didácticas activas:** Informe escrito, Laboratorio.

**Material de apoyo:**

* + - * **Gestión de dependencias con npm:** <https://educated-show-144.notion.site/Manejo-de-dependencias-en-npm-1f74671e02a180059863c5eb9af8229b?pvs=4>

**Duración de la actividad:** 3 horas.

**GA1-220501096-03-AA1-EV04-Tu primer laboratorio con React y Vite – Instalación, estructura y ejecución**

**Descripción de la actividad:**  
En esta actividad, el aprendiz dará sus primeros pasos en el desarrollo de aplicaciones front-end utilizando el framework **React**, apoyado en **Vite** como herramienta de construcción (build tool) moderna y eficiente. El objetivo es que el aprendiz configure correctamente su entorno de trabajo, instale las dependencias necesarias, explore la estructura del proyecto generado por Vite y ejecute una aplicación React básica desde cero.

**Ambiente requerido:**

* Computador con sistema operativo actualizado
* Navegador moderno (Chrome, Firefox, Edge)
* Instalación de Node.js (versión recomendada LTS)
* Visual Studio Code con extensiones de React y ESLint

**Estrategias o técnicas didácticas activas:**

* **Aprendizaje basado en exploración:** el aprendiz configura y lanza por sí mismo su entorno local.
* **Clase demostrativa + práctica individual:** el instructor realiza el procedimiento paso a paso y luego el aprendiz lo replica.

**Materiales de formación:**

* Computador con Node.js y Visual Studio Code instalados
* Acceso al laboratorio de referencia:
  + [Tu primer laboratorio de React + Vite](https://educated-show-144.notion.site/Tu-primer-laboratorio-de-React-Vite-1f74671e02a180d69676c1327e8ead6b?pvs=4)

**Material de apoyo Git y GitHub:**

**Entrega esperada:**

* Generar un informe tipo Bitácora que indique paso a paso del desarrollo del laboratorio.

**Duración de la actividad:** 3 horas.

**3.4 Actividades de Transferencia el Conocimiento:**

**GA1-220501096-03-AA1-EV05-Construcción de un CV modular en React – Componentes, anidación y control de versiones**

**Descripción de la actividad:**

En esta actividad, el aprendiz implementará una **hoja de vida (CV)** completamente modularizada en **React**, utilizando el enfoque de **componentes funcionales anidados**. Cada sección de la hoja de vida será representada por un componente independiente y organizado dentro de una estructura jerárquica controlada desde el componente principal App.jsx.

El objetivo de esta práctica es que el aprendiz aplique de forma práctica los conceptos de **creación y anidación de componentes**, así como la correcta importación/exportación entre ellos. A su vez, se refuerzan competencias en control de versiones, ya que **cada componente creado deberá registrarse mediante un commit individual y descriptivo** en un repositorio público en GitHub.

**Ambiente requerido:**

* Computador con Visual Studio Code
* Node.js (versión LTS recomendada)
* Proyecto React creado con Vite
* Git instalado y cuenta activa en GitHub
* Navegador web moderno

**Estrategias o técnicas didácticas activas:**

* **Aprendizaje basado en proyectos reales:** desarrollo de una hoja de vida personal utilizando React.
* **División del problema en subcomponentes:** organización del código en componentes reutilizables y anidados.
* **Práctica guiada con control de versiones:** registro incremental del avance con commits organizados.

**Materiales de formación:**

* Computador con entorno React + Vite configurado
* Acceso al recurso:
  + [Creación y anidación de componentes en React](https://www.notion.so/Creaci-n-y-anidaci-n-de-componentes-en-React-1f74671e02a1809c82f2cb03aaff5501?pvs=21)

**Material de apoyo Git y GitHub:**

* [Lab 2: Crear un proyecto local](https://www.notion.so/Lab-2-Crear-un-proyecto-b21776ec662d4cbc8f398e9d352a4c80?pvs=21)
* [Lab 11: Crear repositorio remoto en GitHub](https://www.notion.so/Lab-11-Crear-repositorio-remoto-29363009374342cfb7e58044fd5d1660?pvs=21)
* [Lab 12: Sincronizar repositorio local con remoto](https://www.notion.so/Lab-12-Sincronizar-repositorio-local-con-remoto-ba2de6e096d143c5bf3dfe59e40c8d13?pvs=21)

**Estructura sugerida del proyecto:**

como se indica en el recurso [Creación y anidación de componentes en React.](https://www.notion.so/Creaci-n-y-anidaci-n-de-componentes-en-React-1f74671e02a1809c82f2cb03aaff5501?pvs=21)

**Contenido que debe incluir cada componente:**

1. **CabeceraCV.jsx**
   * Nombre completo
   * Título profesional o rol (ej. "Desarrollador Front-End Jr.")
   * Datos de contacto (teléfono, correo, ciudad)
   * Commit: feat: componente CabeceraCV con datos personales actualizados
2. **Perfil.jsx**
   * Descripción personal redactada por el aprendiz (basado en lo aprendido en clases)
   * Habilidades técnicas o blandas destacadas
   * Commit: feat: componente Perfil con descripción personalizada
3. **Experiencia.jsx**
   * Al menos 4 experiencias laborales o proyectos (pueden ser reales o inventados)
   * Deben contener: cargo, empresa, año, breve descripción de funciones
   * Commit: feat: componente Experiencia con 4 cargos definidos
4. **Educacion.jsx**
   * Listado de cursos, Técnicos, Tecnólogo, cursos complementarios, talleres o seminarios realizados (reales o de interés).
   * Institución, nombre del curso y año
   * Commit: feat: componente Educacion con formación complementaria

**Entrega esperada:**

* Repositorio público en GitHub con nombre: cv-react-nombre-aprendiz
* Archivo README.md que incluya:
  + Nombre del proyecto
  + Captura de pantalla del resultado en navegador
  + Instrucciones para ejecutar el proyecto
* Todos los commits reflejando claramente la creación y avance de cada componente

**Duración de la actividad:** 6 horas.

### ****GA1-220501096-03-AA1-EV06-Construcción Componentes dinámicos en React – Renderizado condicional y renderizado de listas****

**Descripción de la actividad:**

En esta actividad, el aprendiz diseñará una interfaz dinámica utilizando **renderizado condicional** y **renderizado de listas en React**, permitiendo que los componentes respondan al contenido de datos y a condiciones lógicas del programa. La actividad parte de un ejercicio práctico donde el aprendiz debe construir una sección de su CV que muestre un listado de **tecnologías dominadas**, representadas visualmente con etiquetas que cambian de color dependiendo de la tecnología (por ejemplo: JavaScript, React, Node.js, etc.).

El objetivo es reforzar el uso de expresiones condicionales como if, operadores ternarios y la función map() para iterar y renderizar múltiples elementos desde un arreglo. Cada paso debe registrarse mediante **commits individuales y descriptivos**, fortaleciendo el control de versiones y la trazabilidad del código.

**Ambiente requerido:**

* Computador con Visual Studio Code
* Node.js (LTS recomendado)
* Proyecto React creado con Vite
* Git y cuenta activa en GitHub
* Navegador web actualizado

**Estrategias o técnicas didácticas activas:**

* **Aprendizaje basado en práctica guiada:** desarrollo de una lista visual con estilos condicionales.
* **Exploración + desarrollo incremental:** el aprendiz implementa lógica condicional por secciones.
* **Uso activo de Git y GitHub:** cada sección funcional debe estar respaldada por un commit organizado.

**Materiales de formación:**

* Computador con entorno React + Vite configurado
* Acceso al recurso:
  + [Renderizado Condicional y Renderizado de Listas – Tutorial de React](https://www.notion.so/Tutorial-de-React-Renderizado-Condicional-y-Renderizado-de-Listas-2014671e02a180568814fee591a6744d?pvs=21)

Estructura sugerida del proyecto: la presentada en el Taller.

**Lista de pasos esperados como commits:**

1. **Experiencia.jsx**
   * Crea un array de objetos con al menos 10 entre experiencias laborales y proyectos (pueden ser reales o inventados)
   * Deben contener: cargo, empresa, año, breve descripción de funciones
   * Commit: feat: componente Experiencia dinamico con 10 cargos definidos
2. **Educacion.jsx**
   * Crea un array de Objetos con el listado de cursos, Técnicos, Tecnólogo, cursos complementarios, talleres o seminarios realizados (reales o de interés).
   * Institución, nombre del curso y año
   * Commit: feat: componente Educacion dinamico con formación complementaria
3. **StackTecnologias.jsx**
   * Replica el Ejemplo práctico: **CV dinámico con Stack de Tecnologías** [Tutorial de React - **Renderizado Condicional y Renderizado de Listas**](https://www.notion.so/Tutorial-de-React-Renderizado-Condicional-y-Renderizado-de-Listas-2014671e02a180568814fee591a6744d?pvs=21)
   * Commit: feat: componente StackTecnolgias dinamico y con renderizado condicicional
4. docs: actualización del README con instrucciones y captura del resultado

**README.md requerido con la entrega:**

* Nombre del proyecto
* Breve descripción del ejercicio
* Instrucciones para ejecutar el proyecto (npm install / npm run dev)
* Captura de pantalla del resultado final en navegador

**Entrega esperada:**

* Repositorio en GitHub con nombre: cv-react-nombre-aprendiz
* Con los **commits nuevos organizados** siguiendo los pasos indicados
* Proyecto ejecutable desde navegador y visualmente organizado

**Duración de la actividad:** 5 horas.

**GA1-220501096-03-AA1-EV07 – Reutilización de componentes en React – Props y desestructuración con una hoja de vida dinámica**

**Descripción de la actividad:**

En esta actividad, el aprendiz reforzará el uso de **componentes reutilizables en React** aplicando la técnica de **props** (propiedades) y **desestructuración** para construir una hoja de vida (CV) modular, dinámica y organizada. A partir de una estructura básica de CV ya construida en actividades anteriores, el aprendiz deberá ampliar el proyecto para que los datos como nombre, cargo, tecnologías, experiencia, educación, etc., **se pasen como props a los componentes**.

El objetivo es desarrollar la capacidad de crear componentes generales que pueden reutilizarse con diferentes valores, favoreciendo la escalabilidad, mantenibilidad y separación de responsabilidades en el desarrollo con React.

**Ambiente requerido:**

* Proyecto React previamente creado (con Vite recomendado)
* Visual Studio Code con extensiones de React
* Node.js (versión LTS)
* Git instalado y cuenta activa en GitHub

**Estrategias didácticas activas:**

* **Aprendizaje basado en proyectos reales:** expansión modular de un CV digital.
* **Desarrollo guiado + personalización individual:** aplicación de conceptos con contenido propio.
* **Documentación incremental:** uso de Git con commits por componente y props añadidas.

**Materiales de formación:**

* [Componentes Reutilizables con Props en React](https://www.notion.so/Componentes-Reutilizables-con-Props-en-React-2084671e02a18035a211c9aab9585dcc?pvs=21)
* [Ejercicio práctico: Ampliando tu CV con Props y Desestructuración](https://www.notion.so/Ejercicio-Pr-ctico-Ampliando-tu-CV-Din-mico-con-Props-y-Desestructuraci-n-2084671e02a1804fa68bc739c3fca38b?pvs=21)

**Requisitos del proyecto:**

1. Utilizar el proyecto de CV desarrollado anteriormente como base.
2. Crear al menos **4 componentes reutilizables** que reciban props:
   * CabeceraCV.jsx → Recibe: nombre, cargo, ciudad, contacto
   * Perfil.jsx → Recibe: resumen profesional
   * Experiencia.jsx → Recibe: una lista de experiencias (cargo, empresa, periodo)
   * Educacion.jsx → Recibe: una lista de estudios y talleres
3. Aplicar **desestructuración de props** en cada componente.
4. Los datos deben estar definidos como constantes en App.jsx o en un archivo de datos aparte (data.js).
5. Visualizar correctamente cada componente dentro de App.jsx.

**Lista de pasos esperados como commits:**

1. refactor: extracción de datos personales en App.jsx
2. feat: componente CabeceraCV ahora recibe props
3. feat: desestructuración de props en CabeceraCV
4. feat: componente Perfil dinámico con props
5. feat: Experiencia mapeada desde arreglo en App.jsx
6. feat: componente Educación con props y desestructuración
7. docs: actualización del README con explicación del uso de props

**Entrega esperada:**

* Repositorio público en GitHub con nombre: cv-react-nombre-aprendiz (continuacion del proyecto anterior)
* Uso correcto de props en al menos 4 componentes
* Desestructuración aplicada dentro de cada componente
* Código ordenado y ejecutable desde navegador
* Mínimo 7 commits organizados siguiendo los pasos anteriores

**Duración la actividad: 5 horas**

**GA1-220501096-03-AA1-EV08 – Eventos y estado local en React – CV dinámico con interactividad**

**Descripción de la actividad:**

En esta actividad, el aprendiz integrará **eventos** y **estado local (useState)** en React dentro del proyecto de hoja de vida (CV) previamente construido. A través de dos nuevas funcionalidades clave, la aplicación será capaz de **responder a la interacción del usuario** y **actualizar dinámicamente su contenido**, mejorando la experiencia e interactividad de la interfaz.

El objetivo es que el estudiante practique la manipulación del estado y el control de eventos para construir componentes verdaderamente reactivos, entendiendo el flujo de datos entre el componente padre (App.jsx) y los componentes hijos, mediante el uso de **props y funciones callback**.

**Objetivos específicos:**

1. Aplicar useState para almacenar y actualizar valores de forma reactiva.
2. Responder a eventos del usuario usando funciones manejadoras (onClick, onChange, onSubmit).
3. Utilizar props para pasar funciones y valores entre componentes padre e hijo.
4. Crear componentes reutilizables con lógica propia de interacción.

**Requisitos del proyecto:**

1. **Reutilizar tu proyecto de CV dinámico**, manteniendo la estructura en mi-app-react con componentes como CabeceraCV, Perfil, Educacion, Experiencia, StackTecnologias, Proyectos, Habilidades.
2. **Agregar dos nuevos componentes interactivos:**
   * ToggleHabilidades.jsx

→ Muestra u oculta el componente Habilidades con un botón.

→ Usa useState y renderizado condicional.

* + FormularioTecnologia.jsx

→ Permite agregar una tecnología al stack desde un formulario controlado.

→ Usa useState para inputs y eventos onChange, onSubmit.

1. **Modificar App.jsx para manejar:**
   * El estado de la lista de tecnologías (useState).
   * La función agregarTecnologia() como prop que se pasa a FormularioTecnologia.
   * La lógica de visibilidad para Habilidades.
2. **Actualizar el archivo cvData.js para separar los datos iniciales del estado dinámico.**

**Lista de pasos esperados como commits:**

1. chore: organización inicial del proyecto para integración de eventos
2. feat: creación de componente ToggleHabilidades con renderizado condicional
3. feat: implementación de useState para mostrar/ocultar habilidades
4. feat: creación de componente FormularioTecnologia con inputs controlados
5. feat: función agregarTecnologia en App.jsx y paso como prop
6. feat: renderizado dinámico de tecnologías desde estado
7. docs: actualización del README con descripción de eventos y estados

**Autoevaluación sugerida:**

1. ¿Qué ventaja ofrece mantener el estado en App.jsx al manejar datos globales del CV?
2. ¿Cómo evitaste que el formulario recargara la página?
3. ¿Qué función cumple setTecnologias([...prev, nueva]) frente a push()?
4. ¿Cómo podrías adaptar FormularioTecnologia para otros datos como idiomas o proyectos?
5. ¿Cómo se vería afectado tu código si manejaras el estado directamente en el componente hijo?

**Ambiente requerido:**

* Proyecto React creado con Vite
* Visual Studio Code con extensiones de React
* Node.js (versión LTS recomendada)
* Git y cuenta activa en GitHub

**Estrategias didácticas activas:**

* **Aprendizaje por integración:** ampliar un proyecto existente con nuevas funcionalidades.
* **Desarrollo guiado + codificación independiente:** cada aprendiz adapta la lógica a su propio CV.
* **Uso progresivo de Git:** commits por cada componente y función añadida.

**Materiales de formación:**

* [Manejo de Events en React](https://www.notion.so/Manejo-de-Events-en-React-20d4671e02a18054a775e7df20f376f3?pvs=21)
* [Manejo del Estado Local en React](https://www.notion.so/Manejo-del-Estado-Local-en-React-20d4671e02a180258843ef6dd56385e8?pvs=21)
* [Ejercicio práctico integrador – CV dinámico interactivo](https://www.notion.so/Ejercicio-Pr-ctico-Integrador-CV-Din-mico-Interactivo-con-Eventos-y-Estado-Local-en-React-20d4671e02a180e18b9aed91f7ecbee3?pvs=21)

**Entrega esperada:**

* Repositorio en GitHub con nombre: cv-react-nombre-aprendiz (continuacion del proyecto anterior)
* Inclusión de los nuevos componentes funcionales
* Uso correcto de useState, onClick, onChange, onSubmit
* Datos definidos en cvData.js, pero gestionados desde App.jsx
* Al menos **7 commits organizados** que evidencien el progreso
* Archivo README.md que incluya:
  + Breve descripción de las funcionalidades añadidas
  + Capturas de pantalla del formulario y habilidades funcionando
  + Instrucciones para ejecutar el proyecto (npm install + npm run dev)

**Duración de la actividad: 5 horas** distribuidas en:

* 1 hora de revisión teórica
* 3 horas de codificación
* 1 hora de pruebas, ajustes y documentación

**GA1-220501096-03-AA1-EV09 – Implementación de interfaz gráfica – Maquetación web con HTML y CSS según prototipo FitTracker**

**Descripción de la actividad:**

En esta actividad, el aprendiz aplicará los principios fundamentales de **maquetación web** utilizando **HTML5 y CSS3**, tomando como referencia un **UI Design profesional** y un **prototipo visual en Figma** del proyecto **FitTracker – Workout Management**. El propósito es traducir el diseño visual en una estructura semántica de código que sea visualmente fiel al prototipo, accesible, ordenada y responsive.

Durante el desarrollo, se espera que el aprendiz implemente buenas prácticas de estructuración del HTML, uso adecuado de etiquetas semánticas, organización con Flexbox/Grid, y separación entre estructura y estilos.

### ****Ambiente requerido:****

* Editor de código (Visual Studio Code recomendado)
* Navegador web actualizado (Chrome, Firefox, etc.)
* Conexión a internet para acceder al prototipo en Figma
* Repositorio Git (opcional para entrega final)

### ****Estrategias didácticas activas:****

* **Aprendizaje por imitación profesional:** recrear un diseño real desde Figma.
* **Enfoque práctico:** aplicación directa de lo aprendido en maquetación.
* **Evaluación visual:** comparación directa entre diseño esperado y resultado codificado.

### ****Materiales de formación y referencia:****

* [Maquetación web con HTML y CSS (guía técnica)](https://www.notion.so/Maquetacion-web-con-Html-y-Css-20d4671e02a180a49f34d2fd3065bde7?pvs=21)
* [UI Design en Figma: FitTracker – Workout Management](https://www.figma.com/design/Ehg1jgSlbz0PAwdD7IMQD5/FitTracker---Workout-Management?node-id=1-719&t=ScfCidOA0XmocoVu-1)

### ****Instrucciones para el desarrollo:****

1. Analiza el prototipo compartido en Figma y navega por sus vistas para comprender el flujo de la aplicación.
2. Realiza la **maquetación completa** de las pantallas utilizando solo HTML y CSS (sin JavaScript).
3. Sigue las indicaciones de estilo, disposición y jerarquía visual del diseño UI.
4. Prioriza el uso de etiquetas semánticas (<header>, <section>, <article>, <nav>, <footer>) y estructura limpia.
5. Utiliza Flexbox o CSS Grid para distribuir los elementos según el diseño.
6. Guarda tu proyecto en una carpeta organizada y comprueba su visualización en el navegador.
7. Finalizada la actividad cada aprendiz debe sustentar la maquetación desarrollada.

### ****Entrega esperada:****

* Carpeta del proyecto fittracker-ui/ con la siguiente estructura

└── 📁fittracker-ui

└── 📁assets

└── 📁fonts

└── 📁icons

└── 📁images

└── 📁videos

└── 📁styles

└── 📁templates <- otras paginas

├── index.html <- paginal princiapal

* El diseño debe ser visualmente coherente con el prototipo
* Se valorará estructura clara, uso correcto de etiquetas y estilos bien organizados
* Repositorio en GitHub con nombre: fittracker-ui

**Duración de la actividad: 12 horas**

**GA1-220501096-03-AA1-EV10 – Navegación, formularios y consumo de datos en React – Manejo de Rutas y APIs REST**

**Descripción de la actividad:**

En esta actividad, el aprendiz ampliará su dominio del desarrollo frontend utilizando React, implementando **rutas dinámicas**, **formularios controlados** y el **consumo de datos desde una API REST**. A partir del contenido proporcionado en la guía técnica, se construirá una pequeña aplicación que permita navegar entre diferentes vistas, realizar peticiones HTTP y manipular datos en tiempo real.

El propósito es que el aprendiz logre integrar múltiples habilidades esenciales en una sola aplicación: navegación con react-router-dom, captura de datos con formularios controlados, y manejo de datos externos con fetch o axios.

**Ambiente requerido:**

* Proyecto React creado con Vite
* Visual Studio Code con extensiones de React
* Node.js (versión LTS)
* Navegador web
* Cuenta de GitHub activa

**Estrategias didácticas activas:**

* **Aprendizaje basado en integración funcional:** unión de rutas, formularios y API en un solo flujo de trabajo.
* **Codificación guiada e individual:** el aprendiz sigue una estructura, pero con su propio contenido.
* **Versionado progresivo:** cada funcionalidad agregada se documenta mediante commits descriptivos.

**Materiales de formación:**

* [Manejo de Rutas, Formularios y APIs REST en React](https://educated-show-144.notion.site/Manejo-de-Rutas-Formularios-y-APIs-REST-en-React-2244671e02a1802d9dd7d09601505425)

Material de apoyo:

* [Aprendiendo a usar el Hook useEffect en React – Guía para Principiantes](https://educated-show-144.notion.site/Aprendiendo-a-usar-el-Hook-useEffect-en-React-Gu-a-para-Principiantes-2164671e02a180ecb6cbcbbe0972d46b)
* [Manejo de Rutas en React con React Router v6](https://educated-show-144.notion.site/Manejo-de-Rutas-en-React-con-React-Router-v6-2164671e02a180a09d51ee43bcba6ed3)

**Requisitos del proyecto:**

**Se debe implementar lo indicado en la Material que corresponde a**

* Navegación entre páginas
* Crear nuevos todos
* Listar todos existentes
* Marcar como completado/pendiente
* Eliminar todos
* Validación básica de formularios
* Estados de carga
* Manejo básico de errores

Reto

* Editar todos - por implementar
* Filtros - por implementar

**Lista de pasos esperados como commits:**

1. feat: creación de estructura inicial con rutas /, /todos y /registro
2. feat: componente Todos.jsx con fetch inicial usando useEffect
3. feat: implementación de listado de todos desde API
4. feat: formulario para crear nuevos todos con validación básica
5. feat: función para marcar todos como completado o pendiente
6. feat: función para eliminar todos del listado
7. feat: estados de carga y mensajes de espera con useState
8. feat: manejo básico de errores en peticiones con try/catch
9. refactor: separación de lógica en componentes y limpieza
10. docs: actualización del README con funcionalidades implementadas

**Reto (por implementar en siguientes iteraciones):**

1. chore: placeholder y lógica inicial para editar todos (sin funcionalidad completa)
2. chore: placeholder para filtros de tareas (sin lógica implementada)

**Entrega esperada:**

* Repositorio en GitHub con nombre: todo-app
* Proyecto funcional accesible por navegador (vía npm run dev)
* Código modular y bien estructurado (componentes por carpeta)
* Rutas navegables sin errores
* Datos obtenidos correctamente desde la API
* Formulario que muestra los datos capturados por consola
* Mínimo 12 commits claros, progresivos y bien redactados
* **Video mostrando el proyecto ejecutado en el que se muestre su cara explicando su implementacion. Link de youtube agreado al readme del proyecto.**

**Duración de la actividad: 8 horas**

* 1 hora: lectura del material y preparación del entorno
* 6 horas: implementación de rutas, API y formulario
* 1 hora: pruebas, revisión del código y entrega en GitHub

**4. PLANTEAMIENTO DE EVIDENCIAS DE APRENDIZAJE PARA LA EVALUACIÓN EN EL PROCESO FORMATIVO.**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Fase del proyecto formativo** | **Actividad del proyecto formativo** | **Actividad de Aprendizaje** | **Evidencias de Aprendizaje** | **Criterios de Evaluación** | **Técnicas e Instrumentos de Evaluación** |
| Construcción | Desarrollar la estructura de datos y la interfaz de usuario del sistema de información. | GA1-220501096-03-AA1-EV01-Estándares y características de interfaces gráficas | GA1-220501096-03-AA1-EV01-Informe | Crea interfaces de usuario  aplicando buenas prácgcas  de usabilidad y  accesibilidad.  Genera plangllas y esglos de  acuerdo con el diseño  establecido. | Estudio de Caso – Cuestionario |
| GA1-220501096-03-AA1-EV02-Gestion de dependencias de código según lenguaje de programación seleccionado(Javascript). | GA1-220501096-03-AA1-EV02- Documento en el procesador de texto(word) | Observación – Lista de Chequeo |
| GA1-220501096-03-AA1-EV02-Cheat sheet | Observación – Lista de Chequeo |
| GA1-220501096-03-AA1-EV03 Manejo Práctico del gestor de dependencias NPM. | GA1-220501096-03-AA1-EV03 informe tipo Bitácora |  | Observación – Lista de Chequeo |
| GA1-220501096-03-AA1-EV04-Tu primer laboratorio con React y Vite – Instalación, estructura y ejecución | GA1-220501096-03-AA1-EV04- informe tipo Bitácora | Observación – Lista de Chequeo |
| GA1-220501096-03-AA1-EV05-Construcción de un CV modular en React – Componentes, anidación y control de versiones | Repositorio de Github cv-react-nombre-aprendiz con el código de la actividad. |  | Observación – Lista de Chequeo |
| GA1-220501096-03-AA1-EV06-Construcción Componentes dinámicos en React – Renderizado condicional y renderizado de listas | Repositorio de Github cv-react-nombre-aprendiz con el código de la actividad. |  | Observación – Lista de Chequeo |
| **GA1-220501096-03-AA1-EV07 –** Reutilización de componentes en React – Props y desestructuración con una hoja de vida dinámica | Repositorio de Github cv-react-nombre-aprendiz con el código de la actividad. |  | Observación – Lista de Chequeo |
| GA1-220501096-03-AA1-EV08 – Eventos y estado local en React – CV dinámico con interactividad | Repositorio de Github cv-react-nombre-aprendiz con el código de la actividad. |  | Observación – Lista de Chequeo |
|  |  | GA1-220501096-03-AA1-EV09 – Implementación de interfaz gráfica – Maquetación web con HTML y CSS según prototipo FitTracker | Repositorio en GitHub con nombre: fittracker-ui |  | Observación – Lista de Chequeo |
|  |  | **GA1-220501096-03-AA1-EV10 – Navegación, formularios y consumo de datos en React – Manejo de Rutas y APIs REST** | Repositorio en GitHub con nombre: todo-app |  | Observación – Lista de Chequeo |

**5. GLOSARIO DE TÉRMINOS**

* + **npm (Node Package Manager):** Un gestor de paquetes para JavaScript, utilizado para instalar, compartir y administrar bibliotecas y dependencias en proyectos de JavaScript.
  + **package.json:** Un archivo en formato JSON presente en la raíz de un proyecto Node.js. Define las propiedades del proyecto, incluyendo metadatos, dependencias, scripts, y más.
  + **Node.js:** Un entorno de ejecución para JavaScript en el servidor, basado en el motor V8 de Chrome.
  + **Dependencia:** Un paquete o módulo que un proyecto necesita para funcionar correctamente. Las dependencias se instalan y gestionan a través de npm.
  + **devDependencies:** Dependencias necesarias solo durante el desarrollo del proyecto, como herramientas de compilación o pruebas.
  + **Global Installation:** La instalación de un paquete de npm de manera global en el sistema, lo que permite su uso en cualquier proyecto en la máquina.
  + **Local Installation:** La instalación de un paquete de npm específicamente en un proyecto. El paquete se encuentra en la carpeta node\_modules del proyecto.
  + **node\_modules:** Una carpeta en un proyecto Node.js donde se almacenan las dependencias locales instaladas a través de npm.
  + **Semantic Versioning (SemVer):** Un sistema de versionamiento para paquetes, que indica las versiones en formato mayor.menor.patch, gestionando compatibilidad y nuevas características.
  + **npm CLI (Command Line Interface):** La interfaz de línea de comandos de npm, utilizada para ejecutar comandos de npm, como instalar o actualizar paquetes.
  + **npm Registry:** Un repositorio público de paquetes de npm, donde los desarrolladores pueden publicar y compartir paquetes de JavaScript.
  + **npx:** Una herramienta de npm que permite ejecutar paquetes npm sin instalarlos globalmente.
  + **Scripts:** Comandos personalizados definidos en el archivo package.json que se pueden ejecutar a través de npm para automatizar ciertas tareas.
  + **npm audit:** Una funcionalidad de npm que analiza las dependencias del proyecto en busca de vulnerabilidades de seguridad conocidas.
  + **package-lock.json:** Un archivo generado automáticamente por npm que registra la versión exacta de cada paquete instalado, asegurando la consistencia entre instalaciones.

1. **REFERENTES BILBIOGRÁFICOS**

Node.js Foundation. (2023). Node.js User Guide. Node.js Foundation. https://nodejs.org/en/docs/

NPM, Inc. (2023). NPM Documentation. https://docs.npmjs.com/

Richards, J. (2022). Mastering Node.js: Building Scalable and Performant Applications. O'Reilly Media. Thompson, H., & Patel, R. (2021). JavaScript and Node.js: Full Stack Development and Applications. Cambridge University Press.

Green, L. (2020). Effective JavaScript Package Management with npm. Addison-Wesley.

Patel, S. (2023). "Understanding Semantic Versioning for JavaScript Developers", Journal of Web Development, 45(2), 134-145.

Jenkins, M., & O'Connor, T. (2021). Security in Node.js and npm: Managing Dependencies and Deployments. Apress.

Warner, A. (2019). "The Evolution of npm and Node.js: A Look at Modern Web Development", Journal of Software Engineering, 12(3), 201-215.

**7. CONTROL DEL DOCUMENTO**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Nombre** | **Cargo** | **Dependencia** | **Fecha** |
| **Autor (es)** | Carlos Enrique Navia Torres | Instructor | CTMA | 28 de abril de 2025 |

**8. CONTROL DE CAMBIOS** (diligenciar únicamente si realiza ajustes a la guía)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Nombre** | **Cargo** | **Dependencia** | **Fecha** | **Razón del Cambio** |
| **Autor (es)** |  |  |  |  |  |