**BOOTCAMP INTENSIVO DE REACT.JS**

**De Principiante a Profesional**

**Basado en el Roadmap de** [**https://roadmap.sh/react**](https://roadmap.sh/react)  
**Academia:** AcademyCoder.com  
**Curso Personalizado para:** Javier y Naomy  
**Duración:** 12 Semanas  
**Horario:** Lunes a Viernes, 8:00 AM - 10:00 AM (2 horas diarias)  
**Fecha de Inicio:** 11 de agosto de 2025  
**Fecha de Finalización:** 31 de octubre de 2025  
**Fecha de Elaboración:** 12 de agosto de 2025

**Índice**

**1. Introducción al Curso**

**2. Materiales y Recursos Necesarios**

**3. Temario con Explicación de Cada Tema**

**3.1 Módulo 1: Fundamentos de React**

**3.2 Módulo 2: Conceptos Intermedios**

**3.3 Módulo 3: Gestión de Estado y Estilos**

**3.4 Módulo 4: Conexiones con APIs y Carrito de Compras**

**3.5 Módulo 5: Pruebas, Performance y Avanzado**

**3.6 Módulo 6: Frameworks y Mejores Prácticas**

**3.7 Módulo 7: Proyectos y Carrera**

**4. Cronograma de Estudio**

**4.1 Semana 1**

**4.2 Semana 2**

**4.3 Semana 3**

**4.4 Semana 4**

**4.5 Semana 5**

**4.6 Semana 6**

**4.7 Semana 7**

**4.8 Semana 8**

**4.9 Semana 9**

**4.10 Semana 10**

**4.11 Semana 11**

**4.12 Semana 12**

**5. Proyecto Final de Portafolio Profesional**

**6. Incorporación a Upwork y Plataformas Freelance**

**7. Anexos**

**7.1 Glosario de Términos**

**7.2 Recursos Adicionales**

**7.3 Certificado de Finalización (Plantilla)**

**Introducción al Curso**

¡Bienvenidos, Javier y Naomy, al **Bootcamp Intensivo de React.js** ofrecido por **AcademyCoder.com**! Este programa ha sido diseñado específicamente para ustedes, con un enfoque colaborativo que combina sesiones de **pair programming**, discusiones grupales diarias y proyectos conjuntos para maximizar el aprendizaje mutuo.

**React.js** es una biblioteca de JavaScript de código abierto, mantenida por Meta, que permite construir interfaces de usuario (UI) interactivas, escalables y eficientes, especialmente para aplicaciones de una sola página (SPA). Este plan de estudios ampliado se basa en el roadmap de <https://roadmap.sh/react>, pero ha sido enriquecido con temas esenciales como **conexiones a APIs**, **gestión de un carrito de compras con Redux**, y otras habilidades avanzadas para garantizar una preparación completa para el mercado profesional. El curso cubre desde los fundamentos hasta técnicas avanzadas, asegurando que no quede ningún tema relevante de React fuera del programa.

Este bootcamp intensivo consta de sesiones diarias de **2 horas** (8:00 AM - 10:00 AM, lunes a viernes), combinando teoría, práctica, ejercicios, desafíos de codificación y revisiones de código. Cada semana culmina con un **proyecto práctico los viernes**, diseñado para consolidar los conocimientos adquiridos. Al finalizar, construirán un **portafolio profesional** que integre todos los proyectos, y recibirán orientación detallada para incorporarse a plataformas freelance como **Upwork** y **Freelancer.com**, facilitando su entrada al mercado laboral como desarrolladores React.

**Objetivos Generales del Curso**

* Dominar los fundamentos, conceptos intermedios y avanzados de React.js para desarrollar aplicaciones web modernas y robustas.
* Construir un portafolio profesional que demuestre habilidades prácticas a través de proyectos reales y complejos.
* Desarrollar habilidades de colaboración, resolución de problemas, codificación limpia y mejores prácticas profesionales.
* Prepararse para el mercado laboral freelance y corporativo, con énfasis en plataformas como Upwork, creación de perfiles atractivos, redacción de propuestas y búsqueda de clientes.

**Beneficios Específicos para Javier y Naomy**

* **Aprendizaje personalizado**: Sesiones adaptadas a su ritmo, con feedback individual y en pareja.
* **Enfoque práctico**: Más del 60% del tiempo dedicado a codificación en vivo y pair programming.
* **Soporte continuo**: Acceso a mentores de AcademyCoder.com vía email o Slack durante todo el curso.
* **Certificado**: Al finalizar, recibirán un certificado digital personalizado de AcademyCoder.com, válido para LinkedIn y portafolios.

**Requisitos Previos**

Se asume conocimiento básico de **HTML**, **CSS** y **JavaScript** (variables, funciones, arrays, objetos, promesas y async/await). La Semana 1 incluye una revisión rápida de estos temas para alinear sus conocimientos.

**Metodología de Enseñanza**

* **Estructura diaria**: 45-60 minutos de teoría/explicación, seguidos de 60-75 minutos de práctica (ejercicios, desafíos de codificación, pair programming).
* **Evaluación**: Proyectos semanales, quizzes rápidos, revisiones de código en pareja y feedback continuo.
* **Tareas adicionales**: 1-2 horas diarias fuera de clase para revisar documentación, practicar y resolver desafíos.
* **Herramientas de colaboración**:
  + **GitHub**: Repositorios compartidos para proyectos y revisiones.
  + **Zoom/Google Meet**: Sesiones en vivo.
  + **Slack**: Canal dedicado para dudas y soporte.

¡Estamos entusiasmados de guiarlos en este viaje hacia convertirse en desarrolladores React profesionales, listos para destacar en el mercado laboral!

**Materiales y Recursos Necesarios**

Para garantizar un aprendizaje fluido, preparen los siguientes materiales antes del inicio del curso. AcademyCoder.com proporcionará guías detalladas de instalación vía email.

**Hardware Requerido**

* Computadora (laptop o desktop) con procesador Intel Core i5 (o equivalente), mínimo 8GB de RAM, 256GB de almacenamiento SSD.
* Conexión a internet estable (mínimo 10 Mbps) para sesiones en vivo, descargas y APIs.
* Micrófono y cámara web para pair programming y discusiones.

**Software Esencial**

* **Editor de Código**: Visual Studio Code (<https://code.visualstudio.com/>). Extensiones recomendadas:
  + ESLint (linter para JavaScript/React).
  + Prettier (formateador de código).
  + React Developer Tools (debugging).
  + Live Share (colaboración en tiempo real).
* **Entorno de Desarrollo**:
  + Node.js (versión LTS, <https://nodejs.org/>).
  + npm (incluido con Node.js) o Yarn para gestionar paquetes.
* **Navegador**: Google Chrome con React Developer Tools (Chrome Web Store).
* **Control de Versiones**: Git (<https://git-scm.com/>) y cuenta en GitHub (<https://github.com/>).
* **Opcional**: Docker para entornos consistentes (<https://www.docker.com/>).

**Recursos Educativos Gratuitos**

* **Documentación Principal**:
  + React Official Docs (<https://react.dev/>).
  + Redux Official Docs (<https://redux.js.org/>).
  + React Router (<https://reactrouter.com/>).
  + Next.js Docs (<https://nextjs.org/>).
* **Roadmap Base**: <https://roadmap.sh/react> (mapa interactivo para seguir progreso).
* **Plataformas en Línea**:
  + freeCodeCamp: Sección de Front-End Libraries (React/Redux).
  + Codecademy: Curso gratuito "Learn React".
  + YouTube:
    - Traversy Media ("React Crash Course").
    - Net Ninja ("React Tutorial").
    - freeCodeCamp ("React + Redux Tutorial").
* **Libros y Guías**:
  + "The Road to React" por Robin Wieruch (PDF gratuito: <https://www.roadtoreact.com/>).
  + "React Hooks in Action" por John Larsen (extractos gratuitos).
  + "Learning React" por Alex Banks y Eve Porcello (O’Reilly, fragmentos en línea).
* **Herramientas Prácticas**:
  + **Create React App**: Bootstrapping de proyectos (npm create-react-app).
  + **Vite**: Alternativa moderna para proyectos rápidos (<https://vitejs.dev/>).
  + **CodeSandbox/StackBlitz**: Prototipado sin instalación local.
  + **APIs Gratuitas**:
    - JSONPlaceholder (<https://jsonplaceholder.typicode.com/>) para datos mock.
    - Unsplash API (<https://unsplash.com/developers>) para imágenes.
    - PokeAPI (<https://pokeapi.co/>) para proyectos lúdicos.
    - Fake Store API (<https://fakestoreapi.com/>) para simulaciones de e-commerce.

**Recursos Adicionales para Javier y Naomy**

* Carpeta compartida en Google Drive con plantillas de código, ejercicios resueltos, grabaciones de sesiones.
* Comunidad: Subreddits (r/reactjs, r/learnreactjs), Stack Overflow, Discord oficial de React.
* Mentorías: Acceso a soporte técnico en [support@academycoder.com](mailto:support@academycoder.com).

**Temario con Explicación de Cada Tema**

El temario está estructurado en **7 módulos** progresivos, ampliando el roadmap de roadmap.sh/react para incluir **conexiones a APIs**, **gestión de carritos de compras con Redux**, **TypeScript**, **Next.js**, y otras habilidades avanzadas demandadas en el mercado. Cada tema incluye:

* **Explicación Detallada**: Qué es, por qué es importante, ejemplos prácticos.
* **Objetivos de Aprendizaje**: Habilidades a dominar.
* **Tiempo Estimado**: Integrado en el cronograma.
* **Recursos Específicos**: Enlaces para profundizar.

**Módulo 1: Fundamentos de React (Semanas 1-2)**

Establece las bases de React, asumiendo conocimientos básicos de JavaScript.

**Introducción a React y Setup**

* **Explicación**: React es una biblioteca para construir interfaces declarativas usando componentes reutilizables y un Virtual DOM para actualizaciones eficientes. Setup incluye instalación de Node.js, npm, y creación de proyectos con Vite o Create React App.
* **Objetivos**: Configurar un entorno local, crear y ejecutar una app básica, entender el ecosistema React.
* **Importancia**: Base para todo desarrollo React; evita errores comunes en configuración.
* **Recursos**: <https://react.dev/learn/installation>, <https://vitejs.dev/guide/>.

**JSX y Componentes**

* **Explicación**: JSX permite escribir HTML-like en JavaScript, compilado a funciones React.createElement. Componentes funcionales son el estándar moderno. Ejemplo: <Greeting name="Javier" />.
* **Objetivos**: Crear, anidar y reutilizar componentes funcionales.
* **Importancia**: Modularidad y reusabilidad, fundamentales para apps escalables.
* **Recursos**: <https://react.dev/learn/writing-markup-with-jsx>.

**Props y State**

* **Explicación**: Props pasan datos de padres a hijos (inmutables). State (useState) maneja datos internos mutables. Ejemplo: Contador con incremento/decremento.
* **Objetivos**: Crear componentes dinámicos con props y estado.
* **Importancia**: Habilita interactividad y flujo de datos unidireccional.
* **Recursos**: <https://react.dev/learn/passing-props-to-a-component>, <https://react.dev/learn/state-a-components-memory>.

**Manejo de Eventos y Formularios**

* **Explicación**: Eventos como onClick, onChange se manejan con funciones JavaScript. Formularios controlados vinculan inputs a state. Ejemplo: Formulario de login con validación.
* **Objetivos**: Manejar inputs de usuario, validar formularios, gestionar submits.
* **Importancia**: Esencial para aplicaciones interactivas (logins, búsquedas).
* **Recursos**: <https://react.dev/learn/responding-to-events>.

**Listas, Keys y Renderizado Condicional**

* **Explicación**: Mapear arrays a elementos JSX con key para optimizar el DOM. Renderizado condicional con operadores ternarios o &&. Ejemplo: Lista de tareas con eliminación.
* **Objetivos**: Renderizar listas dinámicas, evitar bugs de re-render.
* **Importancia**: Mejora performance en listas grandes, evita errores comunes.
* **Recursos**: <https://react.dev/learn/rendering-lists>.

**Módulo 2: Conceptos Intermedios (Semanas 3-4)**

Avanza a lógica más compleja, introduciendo hooks y enrutamiento.

**Ciclo de Vida de Componentes**

* **Explicación**: Métodos como componentDidMount en clases; en hooks, useEffect para efectos secundarios (fetches, timers). Cleanup evita memory leaks.
* **Objetivos**: Gestionar efectos secundarios como API calls o suscripciones.
* **Importancia**: Controla el comportamiento temporal de componentes.
* **Recursos**: <https://react.dev/learn/lifecycle-of-reactive-effects>.

**Hooks Básicos (useState, useEffect)**

* **Explicación**: useState para estado local; useEffect para efectos secundarios (fetches, DOM updates). Dependencias controlan ejecución.
* **Objetivos**: Construir apps funcionales sin clases.
* **Importancia**: Estándar moderno para código limpio y legible.
* **Recursos**: <https://react.dev/reference/react/useState>, <https://react.dev/reference/react/useEffect>.

**Hooks Avanzados (useContext, useReducer, Custom Hooks)**

* **Explicación**: useContext evita props drilling para estado global; useReducer para lógica compleja (similar a Redux); custom hooks encapsulan lógica reutilizable.
* **Objetivos**: Crear apps con estado compartido y hooks personalizados.
* **Importancia**: Mejora escalabilidad y mantenibilidad.
* **Recursos**: <https://react.dev/reference/react/useContext>, <https://react.dev/reference/react/useReducer>.

**Enrutamiento con React Router**

* **Explicación**: React Router permite navegación en SPAs con <Route>, useParams, useNavigate. Soporta rutas dinámicas y protegidas.
* **Objetivos**: Crear aplicaciones multi-vista con navegación fluida.
* **Importancia**: Esencial para dashboards, e-commerce, blogs.
* **Recursos**: <https://reactrouter.com/en/main>.

**Módulo 3: Gestión de Estado y Estilos (Semanas 5-6)**

Enfocado en estado global, Redux y estilización profesional.

**Gestión de Estado con Redux**

* **Explicación**: Redux usa un store central, actions y reducers para gestionar estado global. react-redux conecta con React; redux-thunk para operaciones asíncronas.
* **Objetivos**: Manejar estado en aplicaciones complejas.
* **Importancia**: Estándar en la industria para apps grandes.
* **Recursos**: <https://redux.js.org/>, <https://react-redux.js.org/>.

**Estilos en React**

* **Explicación**: CSS Modules para estilos locales; styled-components para CSS-in-JS dinámico. Tailwind CSS como alternativa moderna.
* **Objetivos**: Crear interfaces responsivas y temáticas.
* **Importancia**: Evita conflictos de estilos, mejora mantenibilidad.
* **Recursos**: <https://styled-components.com/>, <https://tailwindcss.com/>.

**Módulo 4: Conexiones con APIs y Carrito de Compras (Semanas 7-8)**

Añade integración con APIs y un proyecto completo de e-commerce con Redux.

**Conexiones con APIs**

* **Explicación**: Uso de fetch o axios para consumir APIs REST (GET, POST, PUT, DELETE). Manejo de estados de carga, errores y datos con useEffect y useState. Ejemplo: Consumo de Fake Store API.
* **Objetivos**: Integrar datos externos en apps React, manejar respuestas asíncronas.
* **Importancia**: Clave para aplicaciones reales que dependen de datos dinámicos.
* **Recursos**: <https://axios-http.com/docs/intro>, <https://fakestoreapi.com/>.

**Carrito de Compras con Redux**

* **Explicación**: Implementar un carrito de compras persistente usando Redux para almacenar productos, cantidades y totales. Persistencia con localStorage. Ejemplo: Agregar/quitar productos, calcular total.
* **Objetivos**: Crear un sistema de e-commerce completo con Redux.
* **Importancia**: Proyecto práctico demandado en el mercado (e-commerce).
* **Recursos**: <https://redux.js.org/recipes/using-localstorage>.

**Autenticación y Autorización**

* **Explicación**: Implementar login/logout con JSON Web Tokens (JWT) y APIs. Proteger rutas con guards. Ejemplo: Login con API mock.
* **Objetivos**: Gestionar sesiones de usuario de forma segura.
* **Importancia**: Fundamental para aplicaciones con usuarios autenticados.
* **Recursos**: <https://jwt.io/introduction>.

**Módulo 5: Pruebas, Performance y Avanzado (Semanas 9-10)**

Técnicas profesionales para código robusto y optimizado.

**Pruebas**

* **Explicación**: Jest para assertions; React Testing Library para simular interacciones. Tests unitarios, de integración y snapshots.
* **Objetivos**: Escribir tests fiables para componentes.
* **Importancia**: Garantiza calidad y facilita refactors.
* **Recursos**: <https://testing-library.com/docs/react-testing-library/intro/>.

**Optimización de Performance**

* **Explicación**: React.memo, useMemo, useCallback para evitar re-renders. Lazy loading con Suspense y React.lazy.
* **Objetivos**: Crear apps rápidas para datasets grandes.
* **Importancia**: Mejora UX y reduce carga en producción.
* **Recursos**: <https://react.dev/reference/react/memo>.

**Conceptos Avanzados**

* **Explicación**: Higher-Order Components (HOC) para lógica compartida; Render Props para patrones flexibles; Error Boundaries para manejo de errores; Refs para acceso al DOM.
* **Objetivos**: Resolver casos complejos en apps enterprise.
* **Importancia**: Herramientas para proyectos de gran escala.
* **Recursos**: <https://react.dev/learn/manipulating-the-dom-with-refs>.

**TypeScript en React**

* **Explicación**: Tipado estático para props, state y hooks con interfaces/generics. Integración con Create React App o Vite.
* **Objetivos**: Escribir código tipado y seguro.
* **Importancia**: Reduce errores; usado en 70% de empleos React.
* **Recursos**: <https://www.typescriptlang.org/docs/handbook/react.html>.

**Módulo 6: Frameworks y Mejores Prácticas (Semana 11)**

Explora frameworks y estándares profesionales.

**Server-Side Rendering con Next.js**

* **Explicación**: Next.js permite SSR y SSG para mejorar SEO y performance. Uso de getStaticProps, getServerSideProps para data fetching.
* **Objetivos**: Crear apps optimizadas para SEO y velocidad.
* **Importancia**: Popular en e-commerce y sitios corporativos.
* **Recursos**: <https://nextjs.org/docs>.

**Mejores Prácticas**

* **Explicación**: Código limpio (DRY, KISS); accesibilidad con ARIA; internacionalización con react-i18next; seguridad (evitar XSS, sanitización).
* **Objetivos**: Producir código production-ready.
* **Importancia**: Cumple estándares éticos y profesionales.
* **Recursos**: <https://react.dev/learn/keeping-components-pure>, <https://react-i18next.com/>.

**Módulo 7: Proyectos y Carrera (Semana 12)**

Integración de conocimientos y preparación laboral.

**Proyectos y Portafolio**

* **Explicación**: Construir un sitio portafolio con Next.js, integrando proyectos semanales. Deploy en Vercel.
* **Objetivos**: Crear un showcase profesional.
* **Importancia**: Evidencia tangible para empleadores.
* **Recursos**: <https://vercel.com/guides/deploying-react-with-vercel>.

**Búsqueda de Empleo**

* **Explicación**: Estrategias para freelance; optimización de perfiles en Upwork; redacción de propuestas efectivas.
* **Objetivos**: Conseguir los primeros trabajos como React developer.
* **Importancia**: Transición al mercado laboral.
* **Recursos**: <https://www.upwork.com/resources/how-to-create-freelancer-profile>.

**Cronograma de Estudio**

**Estructura diaria**: 8:00-8:45 AM Teoría/Explicación, 8:45-10:00 AM Práctica/Pair Programming (con 5 min de break).  
**Tareas**: 1-2 horas diarias extra para práctica.  
**Proyectos viernes**: Desarrollo y revisión en pareja.

**Semana 1: Fundamentos Básicos**

*(No modificada, ya impartida el 11 de agosto de 2025)*

* **Lunes**: Intro a React y Setup (Instalación Vite/Create React App, app "Hello World"). *Práctica*: Configurar entorno local.
* **Martes**: JSX y Componentes Funcionales (Sintaxis JSX, componentes simples). *Práctica*: Componente <Greeting> personalizado.
* **Miércoles**: Componentes de Clase y Props (Comparación clases vs. funcionales). *Práctica*: Pasar props en familia de componentes.
* **Jueves**: State y Manejo de Eventos (useState, onClick). *Práctica*: Botón que actualiza state.
* **Viernes**: **Proyecto 1** - App de Todo List simple (Agregar/eliminar tareas, revisión en pareja).

**Semana 2: Interactividad Básica**

* **Lunes**: Formularios y Renderizado Condicional (Inputs controlados, validación). *Práctica*: Formulario de login con validación.
* **Martes**: Listas y Keys (Mapear arrays, uso de keys). *Práctica*: Lista dinámica de usuarios.
* **Miércoles**: Composición vs. Herencia (Patrones de diseño). *Práctica*: Componentes compuestos.
* **Jueves**: Lifting State Up (Compartir estado entre componentes). *Práctica*: App con estado compartido.
* **Viernes**: **Proyecto 2** - Formulario de Registro con validación (Email, contraseña, errores condicionales).

**Semana 3: Ciclo de Vida y Hooks**

* **Lunes**: Ciclo de Vida de Componentes (Métodos de clase, useEffect). *Práctica*: Fetch en mount.
* **Martes**: useState en Profundidad (Casos prácticos). *Práctica*: Contador avanzado con múltiples estados.
* **Miércoles**: useEffect para Efectos Secundarios (Dependencias, cleanup). *Práctica*: Timer con cleanup.
* **Jueves**: Custom Hooks (Crear y reutilizar lógica). *Práctica*: Hook para fetch data.
* **Viernes**: **Proyecto 3** - Contador con timers y efectos (Reloj en vivo con useEffect).

**Semana 4: Hooks Avanzados y Context**

* **Lunes**: useReducer (Alternativa a useState para lógica compleja). *Práctica*: State machine simple.
* **Martes**: useContext y Context API (Estado global). *Práctica*: Theme switcher (claro/oscuro).
* **Miércoles**: Combinar Hooks (useEffect + useReducer). *Práctica*: App con múltiples hooks.
* **Jueves**: Debugging Hooks (Errores comunes, herramientas). *Práctica*: Resolver bugs en app.
* **Viernes**: **Proyecto 4** - App de Autenticación con Context (Login/logout global).

**Semana 5: Enrutamiento**

* **Lunes**: Instalación y Setup de React Router (Rutas básicas). *Práctica*: Menú de navegación simple.
* **Martes**: Rutas Dinámicas y useParams (Parámetros URL). *Práctica*: Perfil de usuario dinámico.
* **Miércoles**: Rutas Anidadas y useNavigate (Navegación programática). *Práctica*: Dashboard con subrutas.
* **Jueves**: Protected Routes (Guardias de autenticación). *Práctica*: Ruta privada con login.
* **Viernes**: **Proyecto 5** - Blog Multi-Página con React Router (Posts, detalles, navegación).

**Semana 6: Gestión de Estado con Redux**

* **Lunes**: Intro a Redux (Store, actions, reducers). *Práctica*: Setup básico de Redux.
* **Martes**: Reducers y Dispatch (Lógica de estado). *Práctica*: Contador con Redux.
* **Miércoles**: Conectar Redux a React (useSelector, useDispatch). *Práctica*: Integración en app existente.
* **Jueves**: Middleware con Redux Thunk (Operaciones asíncronas). *Práctica*: API call con thunk.
* **Viernes**: **Proyecto 6** - App de E-commerce con Carrito (Agregar/quitar items, persistencia).

**Semana 7: Conexiones con APIs**

* **Lunes**: Intro a APIs REST (fetch vs. axios). *Práctica*: Consumir JSONPlaceholder (GET).
* **Martes**: Manejo de Estados en API Calls (Loading, error, data). *Práctica*: Lista de posts con estados.
* **Miércoles**: Operaciones CRUD (POST, PUT, DELETE). *Práctica*: Crear/editar posts.
* **Jueves**: Custom Hook para API Calls (Reutilización). *Práctica*: Hook para Fake Store API.
* **Viernes**: **Proyecto 7** - App de Productos con API (Listado, detalles, CRUD).

**Semana 8: Carrito de Compras con Redux**

* **Lunes**: Diseño de Store para Carrito (Estructura de estado). *Práctica*: Setup de Redux para carrito.
* **Martes**: Acciones y Reducers para Carrito (Agregar, quitar, actualizar). *Práctica*: Carrito funcional.
* **Miércoles**: Persistencia con localStorage (Mantener carrito). *Práctica*: Guardar/restaurar carrito.
* **Jueves**: Cálculos Dinámicos (Total, impuestos). *Práctica*: Resumen de compra.
* **Viernes**: **Proyecto 8** - Tienda E-commerce Completa (Carrito, catálogo, checkout).

**Semana 9: Pruebas y Performance**

* **Lunes**: Intro a Jest y React Testing Library (Setup, tests unitarios). *Práctica*: Test de componente simple.
* **Martes**: Testing de Interacciones (Simular clicks, inputs). *Práctica*: Test de formulario.
* **Miércoles**: React.memo, useMemo, useCallback (Memoization). *Práctica*: Lista optimizada.
* **Jueves**: Lazy Loading con Suspense y React.lazy. *Práctica*: Componentes con carga diferida.
* **Viernes**: **Proyecto 9** - Dashboard Optimizado con Tests (Tabla de datos, coverage).

**Semana 10: Conceptos Avanzados y TypeScript**

* **Lunes**: Refs y Forward Refs (Acceso al DOM). *Práctica*: Input con foco automático.
* **Martes**: Error Boundaries y Portals (Errores, modals). *Práctica*: Modal con portal.
* **Miércoles**: Intro a TypeScript (Setup, tipos básicos). *Práctica*: Componente tipado.
* **Jueves**: TypeScript con Hooks y Redux (Interfaces, generics). *Práctica*: App tipada con Redux.
* **Viernes**: **Proyecto 10** - Refactorizar Todo List con TypeScript (Tipado completo).

**Semana 11: SSR y Mejores Prácticas**

* **Lunes**: Intro a Next.js (Setup, páginas). *Práctica*: Página estática con Next.js.
* **Martes**: Data Fetching en Next.js (getStaticProps, getServerSideProps). *Práctica*: Blog SSR.
* **Miércoles**: Mejores Prácticas (Accesibilidad con ARIA, i18n). *Práctica*: App accesible.
* **Jueves**: Seguridad en React (Evitar XSS, sanitización). *Práctica*: Formulario seguro.
* **Viernes**: **Proyecto 11** - Blog SSR con Next.js (Posts dinámicos, SEO).

**Semana 12: Proyectos Finales y Carrera**

* **Lunes**: Construir Portafolio con Next.js (Estructura, bios). *Práctica*: Página de inicio.
* **Martes**: Integrar Proyectos Semanales (Cards, demos). *Práctica*: Galería de proyectos.
* **Miércoles**: Desplegar Portafolio (Vercel). *Práctica*: Deploy live.
* **Jueves**: Perfiles en Upwork/Freelancer.com (Optimización, propuestas). *Práctica*: Crear perfil.
* **Viernes**: **Proyecto Final** - Portafolio Profesional (Presentación, feedback).

**Proyecto Final de Portafolio Profesional**

El proyecto final es un **portafolio web profesional** construido con **React** y **Next.js**, integrando todos los proyectos semanales y demostrando dominio completo de las tecnologías aprendidas.

**Descripción Detallada**

* **Estructura**:
  + **Página de Inicio**: Bios de Javier y Naomy (fotos, resúmenes profesionales).
  + **Galería de Proyectos**: Cards con descripciones, enlaces a demos y GitHub.
  + **Sección de Habilidades**: Lista de tecnologías (React, Redux, TypeScript, etc.).
  + **Formulario de Contacto**: Integrado con EmailJS o Formspree.
* **Integración de Proyectos**: Cada proyecto semanal como demo embebido (iframe) o enlace.
* **Tecnologías**: React, Next.js, Styled-Components/Tailwind, Redux para estado, TypeScript, Vercel para deploy.
* **Objetivos**: Demostrar habilidades técnicas, diseño responsivo, accesibilidad y optimización.
* **Evaluación**: Revisión por mentores; debe ser responsive, accesible (WCAG 2.1), optimizado (Lighthouse score >90).
* **Beneficios**: Sirve como CV digital para LinkedIn, Upwork y entrevistas.

**Tiempo**: Semana 12, con soporte para debugging y revisiones.

**Incorporación a Upwork y Plataformas Freelance**

Guía práctica para iniciar una carrera como desarrolladores React freelance.

**Upwork**

* **Crear Perfil**: Foto profesional, título "React.js Developer | Front-End Specialist". Incluir enlace al portafolio, skills (React, Redux, TypeScript, Next.js). Completar tests gratuitos (JavaScript, React).
* **Búsqueda de Jobs**: Filtrar por "React", "Front-End". Enviar propuestas personalizadas (ej. "Puedo optimizar su app React con Redux en 2 semanas"). Precio inicial sugerido: $20-30/hora.
* **Consejos**: Completar gigs pequeños ($50-100) para reseñas. Usar templates de AcademyCoder.com.

**Freelancer.com**

* **Estrategia**: Perfil similar a Upwork. Participar en contests React (ej. "Build React Dashboard"). Ofertar en proyectos específicos.
* **Otras Plataformas**:
  + Fiverr: Gigs fijos (ej. "Desarrollo componente React por $50").
  + LinkedIn: Networking con recruiters, compartir portafolio.

**Estrategias Generales**

* Actualizar CV con certificado de AcademyCoder.com.
* Unirse a comunidades: r/reactjs, grupos de Facebook, Discord de React.
* **Metas Iniciales**: 1-2 gigs/mes para construir experiencia real.
* Soporte post-curso: Revisión de perfiles y propuestas por AcademyCoder.com.

**Anexos**

**Glosario de Términos**

* **Componente**: Bloque reutilizable de UI en React.
* **Hook**: Función para usar estado/efectos en componentes funcionales.
* **Redux**: Librería para gestión de estado global.
* **SSR**: Server-Side Rendering para SEO y performance.
* **JSX**: Sintaxis HTML-like para escribir UI en JavaScript.
* **Virtual DOM**: Representación en memoria del DOM para actualizaciones eficientes.
* **useEffect**: Hook para efectos secundarios (fetches, timers).
* **useContext**: Hook para estado global sin props drilling.
* **React Router**: Biblioteca para navegación en SPAs.
* **TypeScript**: Superset de JavaScript con tipado estático.
* **Next.js**: Framework React para SSR/SSG.
* **Styled-Components**: Librería para CSS-in-JS.
* **Jest**: Framework de pruebas para JavaScript.
* **React Testing Library**: Herramienta para tests de componentes React.
* **Memoization**: Técnica para evitar re-renders innecesarios.
* **HOC**: Higher-Order Component, patrón para lógica compartida.
* **Render Props**: Patrón para compartir lógica vía props.
* **Error Boundary**: Componente que captura errores en el árbol.
* **Ref**: Mecanismo para acceder al DOM o valores de componentes.
* **API REST**: Interfaz para consumir datos via HTTP.
* **JWT**: JSON Web Token para autenticación.
* **localStorage**: API del navegador para persistencia de datos.

**Recursos Adicionales**

* **Podcasts**: Syntax.fm, React Podcast.
* **Blogs**: Overreacted (Dan Abramov), Kent C. Dodds.
* **Challenges**: LeetCode (JavaScript), HackerRank (React), Codewars.
* **Comunidades**: r/reactjs, Stack Overflow, Discord de React.
* **Cursos Avanzados**: Udemy ("React - The Complete Guide" por Maximilian Schwarzmüller).

**Certificado de Finalización (Plantilla)**

**Certificado de Finalización**  
**Academia**: AcademyCoder.com  
**Curso**: Bootcamp Intensivo de React.js  
**Estudiante**: [Nombre de Javier o Naomy]  
**Duración**: 12 Semanas (11 de agosto - 31 de octubre de 2025)  
**Habilidades Adquiridas**: React, Redux, TypeScript, Next.js, APIs, Testing, SSR, Mejores Prácticas.  
**Proyecto Final**: Portafolio Profesional con [descripción breve].  
**Fecha de Emisión**: 31 de octubre de 2025  
**Firma**: [Director de AcademyCoder.com]