dev Senior

Java Senior con IA

La Ruta Profesional Definitiva

dev Senior

El futuro lo creas tú...

Programa Completo

“Java Senior con IA: La Ruta Profesional Definitiva”

Modalidad
Online - En Vivo

Nivel de dificultad
Inicial

Dedicación
Media - Alta

Proyecto Final.

Acerca del Programa.

“Java Senior con IA: La Ruta Profesional Definitiva”

Bienvenidos al programa Java Senior con IA: La Ruta Profesional Definitiva, un programa diseñado para formar desarrolladores profesionales, capaces de enfrentar los desafíos del mundo real con el respaldo de las herramientas más avanzadas de Inteligencia Artificial.
Este programa te llevará desde los fundamentos esenciales de la programación en Java hasta convertirte en un desarrollador Full Stack altamente competente, con dominio de backend usando Spring Boot, bases de datos modernas, desarrollo frontend con React JS, y un enfoque profesional de calidad, pruebas, DevOps y despliegue en la nube.
A lo largo de esta travesía, trabajarás con herramientas de IA como Herramientas IA y otras asistentes inteligentes para acelerar el desarrollo, mejorar la calidad del código y optimizar tu flujo de trabajo, tal como lo exige la industria actual.
En un entorno donde la velocidad, la precisión y la capacidad de construir soluciones reales marcan la diferencia, este Programa te entrega todo lo necesario para destacar como Desarrollador Java Senior del siglo XXI.

Objetivo principal del programa.

El objetivo principal de este programa es capacitar a los estudiantes para convertirse en Desarrolladores Java Senior con dominio de herramientas de Inteligencia Artificial, capaces de diseñar, desarrollar, probar, desplegar y mantener sistemas reales, seguros y escalables, con calidad profesional y listos para producción.

Esto incluye:

dev Senior

1. Dominio del Backend con Spring Boot + IA
Aprenderás a construir y escalar aplicaciones backend empresariales utilizando Spring Boot, aplicando buenas prácticas, arquitectura limpia y control de versiones. Usarás IA para acelerar el desarrollo, detectar errores, y optimizar la lógica de negocio, mientras implementas microservicios seguros con autenticación JWT.

2. Gestión de Bases de Datos Relacionales y NoSQL
Comprenderás y dominarás el diseño y acceso a bases de datos modernas con PostgreSQL y MongoDB. Aprenderás a aplicar patrones de persistencia avanzados con JPA y Spring Data, integrando validaciones, relaciones y consultas optimizadas con asistencia de IA.

3. Frontend Moderno con React + Integración con APIs
Aprenderás a construir interfaces de usuario modernas, dinámicas y responsivas con React, Tailwind CSS y Vite. Implementarás flujos de autenticación, consumo de APIs seguras, validación de formularios y diseño centrado en el usuario, asistido por IA para crear componentes limpios, reutilizables y accesibles.

4. Integración Full Stack, DevOps y Flujo Profesional de Desarrollo
Integrarás tus habilidades de backend y frontend en sistemas reales para universidades, clínicas y empresas. Aplicarás CI/CD con GitHub Actions, contenerización con Docker, despliegue en Railway y Vercel, y control de calidad con pruebas automatizadas. Te prepararás para trabajar como desarrollador senior en entornos reales, con IA como copiloto de productividad y calidad.

Metodología del Programa

Este programa está estructurado en 12 módulos mensuales, cada uno abordando una etapa fundamental del desarrollo profesional en Java, con integración total de herramientas de Inteligencia Artificial , prácticas reales y acompañamiento intensivo.

Estructura semanal por módulo:

- 1 Clase técnica (2 horas)
- 1 Clase Tutoría de refuerzo técnico (2 horas)
- 1 Clase de inglés técnico con docentes nativos (2 horas) (Programa English
- Boost)

Esto equivale:

- 4 clases técnicas,
- 4 Clases tutorías
- 4 Clases de inglés por módulo.

Metodología integral:

Desarrollo Asistido por IA:
Cada clase incorpora el uso de herramientas inteligentes para escribir, revisar y optimizar código en tiempo real, acelerando el aprendizaje y garantizando calidad desde el inicio.

Proyectos Reales por Dominio:
Cada módulo contiene un proyecto aplicado en un entorno profesional real (universidades, hospitales, empresas), que permite al estudiante aplicar conceptos en contextos de la industria.

Clases Tutorías Estratégicas:
Espacios de 2 horas dedicados exclusivamente al acompañamiento personalizado, resolución de dudas, revisión de código y fortalecimiento técnico.

English Boost – Inglés con Propósito Profesional:
Cada semana los estudiantes asisten a una clase de inglés técnico con docentes nativos, enfocada en mejorar la comunicación para entornos tecnológicos, entrevistas y lectura de documentación.

Preparación Profesional:
Desde el inicio se construye un portafolio sólido. Al finalizar, los estudiantes presentan su proyecto final ante docentes e invitados, y reciben entrenamiento en entrevistas técnicas y proyección laboral.

>Beneficios del programa

Al completar este programa, los estudiantes estarán equipados con:

Habilidades Técnicas Senior + IA:
Dominio profundo en Java moderno, Spring Boot, bases de datos SQL y NoSQL, React JS, y herramientas profesionales como Herramientas IA, GitHub Actions, Docker y arquitecturas modernas. Estarán preparados para construir sistemas empresariales, seguros y escalables.

Experiencia Profesional con Proyectos Reales:
Cada módulo integra un proyecto aplicado en dominios reales como empresas, clínicas y universidades. Los estudiantes ganan experiencia completa desde el análisis hasta el despliegue productivo, con CI/CD, pruebas, documentación y portafolio técnico en GitHub.

Preparación para el Mercado Global:
A través del programa English Boost con docentes nativos, simulaciones de entrevistas técnicas, defensa de proyectos y mentoría directa, los estudiantes se preparan para enfrentar oportunidades en el mercado laboral nacional e internacional como Java Developers Senior.

Este programa no solo te prepara para enfrentar los retos técnicos más exigentes del desarrollo de software moderno, sino que también te entrena para integrarte de inmediato en equipos de desarrollo reales, aplicando buenas prácticas, herramientas de productividad basadas en IA, y flujos de trabajo profesionales de principio a fin.

Nos emociona acompañarte en esta ruta hacia tu transformación profesional. Prepárate para convertirte en un desarrollador completo, competitivo y listo para liderar proyectos de alto impacto.

¡Bienvenido a Java Senior con IA: ¡La Ruta Profesional Definitiva!

dev Senior

ESTRUCTURA DEL PROGRAMA POR MÓDULOS, CLASES Y CLASES-TUTORÍAS

MÓDULO 1: MÓDULO 1: Fundamentos Modernos de Programación con Java e Inteligencia Artificial

En este primer módulo, el estudiante se familiariza con los fundamentos esenciales de la programación en Java, utilizando IA como entorno de desarrollo principal y copiloto inteligente. Se enfatiza el dominio de los conceptos básicos y su aplicación práctica en proyectos iniciales, con asistencia de herramientas de IA para optimizar el aprendizaje.

CLASE 1: Configuración Profesional del Entorno y Primer Proyecto.

Objetivos de aprendizaje:

- Instalar y configurar un entorno de desarrollo profesional moderno.
- Comprender el flujo básico de ejecución de un programa Java.
- Utilizar herramientas de IA para acelerar el aprendizaje.

Contenidos

- Instalación de Java JDK y configuración de variables de entorno.
- Instalación de Herramientas IA como editores.
- Configuración de extensiones clave de desarrollo Java.
- Instalación y vinculación de Git y GitHub (SSH y HTTPS).
- Introducción al uso básico de Git: inicializar repositorios, realizar commits, push y pull.
- Primer proyecto: creación del programa HelloWorld.java con explicación línea por línea.
- Uso de IA para sugerencias de código, correcciones y mejoras de estilo.

Ejercicio práctico:

- Crear una carpeta de trabajo estructurada y conectar con GitHub, y subir el primer programa "Hola Mundo"; , documentado correctamente.

CLASE-TUTORÍA 1: Consolidación del Entorno, Git y Uso Efectivo de Herramientas IA

Objetivos de la CLASE-TUTORÍA:

- Asegurar que todos los estudiantes tengan un entorno de desarrollo funcional.
- Enseñar el uso práctico y estratégico de IA como herramienta de aprendizaje.
- Fortalecer la comprensión del flujo de trabajo con Git y GitHub.

Actividades:

- Asistencia personalizada en la resolución de errores comunes de instalación.
- Ejercicios prácticos con Git: clonación de repositorio remoto, ramas, manejo de conflictos.
- Demostración del uso de IA para entender fragmentos de código desconocidos.
- Simulación de un "par de programación con IA": el estudiante escribe y la IA propone.
- Creación del primer repositorio individual con estructura organizada, buenas prácticas de
- nombres y documentación inicial.

CLASE 2: Variables, Tipos de Datos, Constantes y Operadores.

Objetivos de aprendizaje:

- Comprender y utilizar correctamente los tipos de datos primitivos en Java.
- Declarar variables, constantes y realizar conversiones de tipo.
- Utilizar operadores básicos en expresiones simples y compuestas.

Contenidos

- Tipos de datos: int, double, boolean, char, String.
- Diferencia entre tipos primitivos y objetos (envolventes).
- Reglas de conversión de tipos: implícita y explícita (casting).
- Declaración y uso de constantes con final.
- Operadores:
  - **Aritméticos**: +, -, *, /, %
  - **Relacionales**: ==, !=, >, <, >=, <=
  - **Lógicos**: &&, | |, !

Ejercicio práctico:

- Aplicación simple para convertir medidas (metros a centímetros, kg a libras, grados a Fahrenheit) utilizando variables, operadores y entrada/salida por consola.

CLASE-TUTORÍA 2: Refuerzo de Tipos, Expresiones y Validaciones

Objetivos de la CLASE-TUTORÍA:

- Reforzar la comprensión de los tipos de datos y la lógica detrás de las operaciones básicas.
- Identificar errores comunes y aprender a validarlos usando IA y pruebas manuales.

Actividades:

- Taller de creación de expresiones booleanas complejas con evaluación paso a paso.
- Revisión asistida por IA de fragmentos de código mal estructurados.
- Mini proyecto: sistema para calcular el IMC con ingreso validado de peso y altura.
- Discusión de errores de lógica típicos: precedencia de operadores, mal uso de paréntesis,
- errores por conversión de tipos.

CLASE 3: Estructuras de Control de Flujo (Condicionales y Bucles)

Objetivos de aprendizaje:

- Aplicar estructuras condicionales para control de decisiones en programas.
- Utilizar bucles para ejecutar procesos repetitivos de forma controlada.
- Aplicar validaciones y flujos de lógica que reflejan problemas reales.

Contenidos

- Condicionales:
  - if, else if, else, switch-case
- Bucles:
  - while, do-while, for
- Palabras clave break, continue y su uso estratégico.
- Construcción de flujos de lógica como menús interactivos o validaciones repetitivas.
- Comparación entre estructuras de control generadas por el estudiante y sugeridas por IA.

Proyecto en clase:

- Creación de un menú de opciones en consola que permite calcular: área de figuras, edad futura, o verificar si un número es primo.

CLASE-TUTORÍA 3: Resolución de problemas con lógica y estructuras de control

Objetivos de la CLASE-TUTORÍA:

- Aplicar los conocimientos en escenarios prácticos donde la lógica se vuelve esencial.
- Reforzar el pensamiento algorítmico y la depuración con herramientas inteligentes.

Actividades:

- Retos de programación guiados: validación de contraseñas, sistema de intentos, menú de navegación.
- Diagnóstico de errores lógicos con ayuda de IA: por qué un bucle no se detiene, por qué un caso switch no funciona.
- Desarrollo colaborativo de una "calculadora de lógica", que usa condicionales y bucles para resolver problemas simples.
- Feedback de buenas prácticas en la estructura y legibilidad del código.

CLASE 4: Funciones, Modularidad y Proyecto Práctico Integrador

Objetivos de aprendizaje:

- Aplicar el concepto de modularidad para dividir y organizar el código.
- Utilizar funciones con y sin retorno para resolver problemas.
- Consolidar los conocimientos del módulo en un proyecto estructurado.

Contenidos

- Declaración y definición de métodos: void, con retorno (int, double, boolean, etc.)
- Parámetros y argumentos: paso por valor
- Alcance de variables: local vs global
- Uso de IA para extraer funciones repetidas y mejorar la legibilidad
- Diseño modular y trabajo por capas desde la base

Proyecto práctico:

- Aplicación de consola: Sistema de Registro de Estudiantes Funciones:
  - Registrar nuevo estudiante
  - Mostrar todos los registros
  - Buscar estudiante por nombre
  - Calcular promedio general
  - Validación de entradas y salidas con IA

TCLASE-TUTORÍA 4: Desarrollo asistido y retroalimentación personalizada del proyecto

Objetivos de la CLASE-TUTORÍA:

- Aplicar todo lo aprendido en un entorno de desarrollo simulado.
- Analizar, mejorar y defender el proyecto con acompañamiento profesional.

Actividades:

- Revisión en parejas del proyecto integrador con retroalimentación cruzada
- Simulación de presentación de código ante un líder técnico
- Diagnóstico automático del código con IA: uso adecuado de funciones, modularidad, redundancia, claridad
- Refactorización conjunta del proyecto con sugerencias de IA
- Preparación de README y subida profesional a GitHub

MÓDULO 2: Programación Orientada a Objetos

EEste módulo introduce al estudiante en la Programación Orientada a Objetos (POO), uno de los pilares fundamentales del desarrollo en Java. Se aprenderá a modelar clases, utilizar herencia, aplicar interfaces. Se promueve el uso de herramientas IA para análisis y refactorización de código orientado a objetos.

CLASE 1: Clases, Objetos, Constructores y Encapsulamiento

Objetivos de aprendizaje:

- Comprender la estructura y el propósito de clases y objetos.
- Aplicar la encapsulación como base de la programación robusta.
- Construir y utilizar instancias de clases con propiedades y comportamientos definidos.

Contenidos

- ¿Qué es un objeto? ¿Qué es una clase? Analogías del mundo real.
- Atributos y métodos.
- Uso de constructores por defecto y parametrizados.
- Encapsulamiento: visibilidad con private, public, protected.
- Getters y setters con convenciones de JavaBeans.
- Creación de múltiples instancias y manipulación de objetos en memoria.
- Uso de Herramientas IA para generar clases de manera automatizada y detectar errores de encapsulamiento.

Proyecto en clase:

- Crear una clase Libro con atributos como título, autor, ISBN y precio.
- Crear una clase principal que permita crear objetos Libro y mostrar su información de forma segura.

CLASE-TUTORÍA 1: Refuerzo práctico de modelado de clases y encapsulamiento

Objetivos de la CLASE-TUTORÍA:

- Reforzar la capacidad de crear clases con atributos y comportamientos claros.
- Evaluar el uso correcto del encapsulamiento y buenas prácticas.

Actividades:

- Taller de modelado orientado a objetos: crear clases Producto, Estudiante, Programa.
- Revisión de errores comunes: atributos públicos, falta de encapsulación, duplicación de código.
- IA como asistente de validación: ¿se pueden mejorar los getters/setters? ¿se están exponiendo datos innecesarios?
- Mini proyecto: Sistema de gestión de productos con operaciones básicas desde objetos.

CLASE 2: Herencia, Polimorfismo e Interfaces

Objetivos de aprendizaje:

- Aplicar herencia como mecanismo de reutilización de código.
- Comprender el polimorfismo como capacidad de adaptar comportamientos.
- Introducir el uso de interfaces como contrato de implementación.

Contenidos

- Herencia: uso de extends, reutilización y especialización de clases.
- Clases padre e hijas, superclase y subclase.
- Uso de super() para acceder a constructores y métodos superiores.
- Sobrescritura de métodos (@Override) y uso práctico del polimorfismo.
- Interfaces: declaración con interface, implementación con implements.
- Comparación entre clases abstractas e interfaces.
- Uso de Herramientas IA para sugerir métodos a sobrescribir y validar consistencia jerárquica.

Ejercicio Práctico:

- Crear una jerarquía de clases Empleado, Gerente, Desarrollador.
- Crear una interfaz Trabajable con método trabajar(), e implementarla en clases concretas.

CLASE-TUTORÍA 2: Reforzamiento de jerarquías, abstracción y reutilización

Objetivos de la CLASE-TUTORÍA:

- Identificar jerarquías correctas e incorrectas.
- Aplicar polimorfismo en métodos que interactúan con clases abstractas o interfaces.

Actividades:

- Análisis de diagramas de clases reales: qué se hereda, qué se especializa.
- Ejercicio colaborativo: crear un modelo de zoológico con animales genéricos y específicos.
- Uso de IA para sugerencias de diseño orientado a objetos.
- Validación de objetos polimórficos: Empleado e = new Gerente();
- Casos de mal diseño (herencia innecesaria, duplicación de código).

CLASE 3: Modelado Avanzado de Clases en POO

Objetivos de aprendizaje:

- Diseñar soluciones utilizando técnicas avanzadas de modelado de clases.
- Modelar relaciones complejas entre clases, como composición y agregación.
- Implementar un sistema funcional utilizando buenas prácticas de diseño.

Contenidos

- Relaciones Avanzadas entre Clases
  - Composición y agregación en profundidad.
  - Relaciones bidireccionales y restricciones.
- Utilización de clases abstractas y métodos abstractos.
- Comparación entre clases abstractas e interfaces.

Ejercicio práctico:

- Se modelará un sistema para gestionar un restaurante, incorporando relaciones complejas entre clases.

CLASE-TUTORÍA 3: Modelado de Clases en POO

Objetivos de la CLASE-TUTORÍA:

- Reconocer y aplicar conceptos avanzados de relaciones entre clases, como composición, agregación y herencia.
- Proponer mejoras con IA y aplicar refactorizaciones en el código.

Actividades:

- Realizar ejercicios de modelado y conversión de ellos a clases Java y relaciones manual.
- Herramientas IA como mentores automáticos: sugerencias para mejorar estructuras según OOP.
- IA Generación de modelo con el gráfico del modelo de datos.

CLASE 4: Proyecto Integrador Orientado a Objetos con IA

Objetivos de aprendizaje:

- Integrar los conceptos de POO y principios SOLID en un proyecto funcional.
- Aplicar conocimientos adquiridos para diseñar una solución escalable y mantenible.
- Utilizar la IA para mejorar estructura, identificar errores y optimizar el diseño.

Contenidos

- Análisis de requerimientos y diseño de clases
- Definición de interfaces, herencias y comportamientos
- Separación de lógica en capas: main, servicio, modelo
- Validaciones básicas y manejo de errores
- Documentación de clases y métodos

Proyecto Sugerido:

- Sistema de gestión de clientes con roles, historial de acciones y reglas de negocio diferenciadas.

CLASE-TUTORÍA 4: Presentación, revisión estructural y retroalimentación

Objetivos de la CLASE-TUTORÍA:

- Evaluar el diseño y la implementación del proyecto bajo criterios técnicos y de buenas prácticas.
- Aplicar revisión por pares y retroalimentación crítica.
- Mejorar el código con IA como copiloto de calidad.

Actividades:

- Presentación del diseño de clases (diagramas UML opcional)
- Validación de cumplimiento de principios OOP y SOLID
- Feedback técnico de parte del docente y compañeros
- Refactorizaciones finales con apoyo de IA
- Subida del proyecto a GitHub con README profesional y commits bien documentados

MÓDULO 3: Interfaces Gráficas de Usuario Modernas (GUI Web Embebida y Java FullStack)

Este módulo introduce a los estudiantes al diseño y desarrollo de interfaces gráficas de usuario modernas, reemplazando tecnologías obsoletas como Swing o JavaFX por soluciones actuales basadas en web embebida y frameworks Java contemporáneos como Vaadin Flow. Se busca construir interfaces visuales modernas y funcionales, aplicables a sistemas empresariales reales.

CLASE 1: Introducción a Interfaces Gráficas Modernas en Java

Objetivos de aprendizaje:

- Comprender las limitaciones de los enfoques clásicos (Swing, JavaFX).
- Explorar soluciones modernas basadas en el navegador embebido y en frameworks Java UI reactivos.
- Instalar y preparar el entorno para desarrollo de GUI embebidas y Vaadin Flow.

Contenidos

- ¿Por qué abandonar JavaFX y Swing? Comparativa con interfaces web modernas.
- Fundamentos de Web embebida: uso de WebView para cargar HTML desde Java.
- Introducción a Vaadin Flow: Framework Java para construir interfaces modernas sin escribir HTML.
- Configuración inicial de un proyecto Vaadin en Herramientas IA.
- Estructura básica de una aplicación Vaadin con rutas, vistas y componentes.
- Lanzamiento de servidor embebido (Spring Boot + Vaadin) desde Herramientas IA.
- Uso de Herramientas IA para generación automática de vistas e interacción básica con componentes.

Ejercicio práctico:

- Crear una pequeña aplicación con Vaadin para mostrar un mensaje dinámico en pantalla usando un botón y un formulario básico.

CLASE-TUTORÍA 1: Entorno gráfico en Java: consolidación y práctica asistida

Objetivos de la CLASE-TUTORÍA:

- Asegurar que todos los estudiantes comprendan y puedan ejecutar interfaces gráficas en entornos modernos.
- Diagnosticar errores comunes de configuración y estructuras mal implementadas.
- Aprovechar IA para mejorar el diseño visual, la interacción y la limpieza del código.

Actividades:

- Ayuda personalizada en la creación del primer proyecto Vaadin (estructura, dependencias, rutas).
- Exploración guiada de componentes Vaadin más comunes (TextField, Button, Layouts).
- Pruebas de diseño responsivo y detección de errores de estilos o navegación.
- Simulación de mejoras sugeridas por Herramientas IA para simplificar código UI y aumentar claridad visual.
- Conversión de interfaces de consola creadas en módulos anteriores a interfaces gráficas simples.

CLASE 2: Formularios interactivos, navegación y eventos

Objetivos de aprendizaje:

- Crear formularios web con validaciones y envío de datos.
- Controlar flujos de navegación entre vistas dentro de una aplicación gráfica.
- Gestionar eventos y actualizar dinámicamente la UI con Java puro.

Contenidos

- Creación de formularios con TextField, NumberField, ComboBox, Button.
- Validación de entradas y manejo de errores en tiempo real.
- Envío de datos del formulario a servicios backend.
- Navegación entre vistas: definición de rutas, RouterLink, @Route.
- Actualización de componentes gráficos con eventos.
- Uso de Herramientas IA para autocompletado de formularios y sugerencias de validaciones.

Ejercicio práctico:

- Crear una interfaz para registrar estudiantes con validación de campos y confirmación del registro. Incluir navegación hacia una vista de listado.

CLASE-TUTORÍA 2: Validaciones, rutas y comunicación visual fluida

Objetivos de la CLASE-TUTORÍA:

- Afianzar el desarrollo de formularios con buenas prácticas UX.
- Resolver errores comunes al pasar datos entre vistas o al interactuar con múltiples componentes.

Actividades:

- Taller práctico: crear una interfaz para gestionar empleados (registro, búsqueda, eliminación).
- Acompañamiento en la validación de entradas y el diseño centrado en el usuario.
- Revisión cruzada de vistas entre compañeros para evaluar claridad, organización visual y mensajes de error.
- Refactorización con IA: sugerencias para mejorar nombres de campos, simplificar lógica de eventos y estructurar código visual.

CLASE 3: Embebido de Interfaces Web Personalizadas desde Java

Objetivos de aprendizaje:

- Integrar interfaces HTML/CSS externas dentro de una aplicación Java.
- Establecer comunicación entre código Java y JavaScript embebido.
- Aprovechar el poder del desarrollo frontend moderno desde backend Java.

Contenidos

- Uso del componente WebView (o EmbeddedView) para cargar interfaces HTML externas.
- Integración de frameworks visuales: HTML + TailwindCSS + JavaScript en UI embebida.
- Comunicación entre Java y JavaScript: llamadas entre capas.

- Embebido de dashboards, gráficos o formularios creados con herramientas web.
- Ventajas de este enfoque híbrido en soluciones empresariales.
- Uso de IA para ayudar en generación de HTML, estilos y scripts integrados.

Ejercicio práctico:

- Crear un módulo de estadísticas donde se visualicen datos desde Java en un gráfico embebido hecho con HTML + Chart.js.

CLASE-TUTORÍA 3: Integración Web en Java: retos y optimización

Objetivos de la CLASE-TUTORÍA:

- Reforzar la integración entre código Java y vistas web externas.
- Mejorar el diseño visual y la conexión con servicios Java backend.

Actividades:

- Pruebas prácticas de integración HTML embebido en una app Java.
- Corrección de problemas comunes en rutas relativas, carga de scripts y comunicación entre capas.
- Uso de IA para validar estructuras de archivos web embebidos.
- Revisión de seguridad básica al trabajar con recursos externos.

CLASE 4: Proyecto Final de Interfaz Gráfica con Asistencia IA

Objetivos de aprendizaje:

- Aplicar los conocimientos adquiridos en un sistema gráfico completo.
- Demostrar dominio de diseño de interfaces, navegación y manipulación visual de datos.
- Utilizar herramientas IA para perfeccionar diseño, funcionalidad y experiencia del usuario.

Contenidos

- Desarrollo de una aplicación gráfica moderna (tipo sistema de inventario, agenda de contactos, CRM básico).
- Integración de formularios, navegación, visualización de registros y búsqueda dinámica.
- Buenas prácticas UX: consistencia, feedback visual, accesibilidad.
- Documentación del proyecto, modularización y subida a GitHub.

Recomendación IA:

- Usar herramientas IA como auditor visual del proyecto: estructura, nombres, validaciones, redundancia.

CLASE-TUTORÍA 4: Evaluación, retroalimentación y perfeccionamiento del proyecto

Objetivos de la CLASE-TUTORÍA:

- Consolidar el desarrollo de una interfaz gráfica profesional en Java moderno.
- Recibir retroalimentación crítica del docente y del equipo.
- Perfeccionar la experiencia de usuario a nivel técnico y visual.

Actividades:

- Presentación formal del proyecto: flujos de navegación, vista en vivo, explicación del código.
- Revisión por pares: análisis de usabilidad y limpieza visual.
- Simulación de pruebas de usuario: detectar fallos de experiencia e inconsistencias visuales.
- Refactorización final con IA y despliegue completo del proyecto en GitHub.

MÓDULO 4: Colecciones y Streams en Java

En este módulo el estudiante dominará las estructuras de datos dinámicas más utilizadas en el desarrollo profesional: listas, conjuntos y mapas. También aprenderá a procesar datos de forma funcional con la API de Streams, incluyendo paralelización de operaciones y patrones de manipulación de datos masivos. Se incorpora el uso de herramientas IA para sugerencias de estructuras, transformación de código imperativo a funcional, y optimización de pipelines de datos.

CLASE 1: Introducción a Colecciones en Java (List, Set, Map)

Objetivos de aprendizaje:

- Comprender las estructuras de datos más usadas en Java.
- Aplicar las colecciones adecuadas según el problema a resolver.
- Manipular listas, conjuntos y mapas con operaciones básicas.

Contenidos

- Interfaces y clases del framework de colecciones: List, Set, Map.
- Implementaciones: ArrayList, LinkedList, HashSet, TreeSet, HashMap, TreeMap.
- Métodos fundamentales: add, remove, get, contains, clear, isEmpty.
- Comparación entre estructuras: orden, duplicados, acceso por clave.
- Iteración con bucles clásicos y for-each.
- Uso de herramientas IA para generar estructuras de datos a partir de descripciones en lenguaje natural.

Ejercicio práctico:

- Crear un sistema de gestión de inventario con ArrayList y HashMap, con operaciones CRUD básicas en consola.

CLASE-TUTORÍA 1: Estructuras dinámicas en acción: selección y aplicación práctica

Objetivos de la CLASE-TUTORÍA:

- Consolidar el uso adecuado de cada tipo de colección en problemas reales.
- Identificar errores comunes en el uso de listas, sets y mapas.

Actividades:

- Comparar estructuras: ¿Cuándo usar List vs. Set? ¿Cuándo elegir HashMap?
- Taller práctico: creación de una agenda de contactos con HashMap.
- Identificación y solución de errores de lógica con ayuda de IA: valores repetidos, claves inexistentes, referencias nulas.
- Refactorización del ejercicio práctico con sugerencias de estructura de código.

CLASE 2: Streams API - Procesamiento Funcional de Datos

Objetivos de aprendizaje:

- Procesar colecciones de forma eficiente y declarativa.
- Aplicar transformaciones funcionales como map, filter, reduce.
- Comprender cómo los Streams cambian el enfoque imperativo por uno funcional.

Contenidos

- Introducción a Stream<T>: características, operaciones y flujo.
- Operaciones intermedias: map, filter, sorted, distinct, limit, skip.
- Operaciones terminales: forEach, collect, count, findFirst, reduce.
- Comparación entre bucles tradicionales y programación funcional.
- Uso de Herramientas IA para convertir código iterativo a Streams automáticamente.

Ejercicio práctico:

- Crear una aplicación que procese una lista de empleados: filtrar mayores de 30 años, calcular salario promedio, ordenar por apellido y mostrar los 5 primeros.

CLASE-TUTORÍA 2: Optimización de código con Streams y enfoque declarativo

Objetivos de la CLASE-TUTORÍA:

- Aprender a diseñar flujos de datos usando Streams correctamente.
- Resolver problemas complejos con pipelines funcionales simples.

Actividades:

- Transformación de código clásico a Streams paso a paso.
- Laboratorio: simulación de procesamiento de pedidos y agrupación por cliente.
- Uso de IA para:
  - Generar pipelines de Streams a partir de instrucciones escritas.
  - Detectar redundancia o estructuras ineficientes.
- Comparación de performance entre enfoques imperativos y funcionales en escenarios simples.

CLASE 3: Streams Avanzados y Parallel Streams

Objetivos de aprendizaje:

- Implementar pipelines funcionales más complejos.
- Ejecutar Streams de forma paralela para mejorar el rendimiento.
- Aplicar agrupación y estadísticas avanzadas sobre colecciones..

Contenidos

- Agrupaciones y reducciones: Collectors.groupingBy(), partitioningBy(), summarizingInt().
- Transformaciones en cascada con múltiples map y filter.
- Introducción a parallelStream(): ventajas, riesgos, cuándo utilizarlo.
- Evaluación del costo de procesamiento con System.nanoTime() y comparativas.
- Sugerencias de Herramientas IA para paralelizar y simplificar pipelines complejos.

Ejercicio práctico:

- Analizar una base de datos simulada de ventas: total por producto, cliente con mayor facturación, agrupación de productos por categoría.

CLASE-TUTORÍA 3: Procesamiento eficiente de datos masivos con Streams

Objetivos de la CLASE-TUTORÍA:

- Consolidar el uso profesional de Streams para análisis de datos.
- Aplicar paralelismo y agrupación para resolver escenarios empresariales.

Actividades:

- Análisis guiado: flujo de datos en una app que procesa registros bancarios.
- Taller: sistema de reporte mensual con agrupación por fechas y cálculos estadísticos.
- Diagnóstico de cuellos de botella y puntos de mejora sugeridos por IA.
- Refactorización de proyectos previos del estudiante aplicando Streams para mayor claridad y rendimiento.

CLASE 4: Proyecto Integrador – Gestión de Datos en Java

Objetivos de aprendizaje:

- Aplicar colecciones y Streams en un sistema completo y funcional.
- Demostrar dominio en la selección de estructuras, transformación de datos y presentación de resultados.

Contenido

- Análisis de requerimientos: estructura de datos, operaciones necesarias, validaciones.
- Diseño de clases modelo (Producto, Usuario, Venta, etc.)
- Implementación de operaciones CRUD y consultas con Streams.
- Separación por capas (modelo, lógica, vista) y uso de métodos auxiliares reutilizables.
- Asistencia de IA en generación de código base, refactorización y documentación.

Proyecto en clase

- **Sistema de gestión de productos y análisis de ventas con operaciones como:**
  - Agregar producto
  - Calcular total de ingresos
  - Buscar productos por nombre o categoría
  - Reportes resumidos por vendedor o cliente

CLASE-TUTORÍA 4: Revisión integral del proyecto + feedback profesional

Objetivos de la CLASE-TUTORÍA:

- Evaluar el diseño, rendimiento y claridad del sistema construido.
- Realizar revisión técnica completa con foco en estructuras y rendimiento.

Actividades:

- Presentación de proyectos en pequeños equipos: explicación del diseño, desafíos, y decisiones tomadas.
- Validación del correcto uso de Streams y colecciones según el objetivo.
- Refactorización asistida por IA: duplicidad, código innecesario, orden lógico.
- Recomendaciones finales para convertir el proyecto en parte del portafolio profesional.

MÓDULO 5: Spring Boot desde Cero – Arquitectura Profesional para Aplicaciones Backend

En este módulo, el estudiante se introduce en el framework Spring Boot, una de las herramientas más poderosas y utilizadas en la industria para construir aplicaciones backend modernas, escalables y seguras. Se aprende a estructurar un proyecto real con controladores, servicios y repositorios, aplicar buenas prácticas de desarrollo, y utilizar herramientas como Herramientas IA para automatizar tareas comunes y mejorar la calidad del código.

CLASE 1: Introducción a Spring Boot y Estructura de Proyectos

Objetivos de aprendizaje:

- Comprender la arquitectura y objetivos de Spring Boot.
- Crear un proyecto backend moderno con estructura profesional.
- Iniciar la construcción de APIs RESTful.

Contenidos

- ¿Qué es Spring Boot y por qué es tan usado?
  - Comparación con otros frameworks y herramientas legadas.
- Generación de proyecto con Spring Initializr.
- Estructura de carpetas profesional:
  - controller, service, repository, model, config.
- Configuración básica del archivo application.properties.
- Creación de la primera clase @RestController.
- Primer endpoint: GET /saludo que devuelve un mensaje JSON.
- Uso de herramientas IA para generar controladores y documentación automática del código.

Ejercicio práctico:

- Crear una API básica que devuelva información de prueba sobre un recurso Libro.

CLASE-TUTORÍA 1: Entendiendo la estructura Spring y sus componentes

Objetivos de la CLASE-TUTORÍA:

- Reforzar la estructura limpia y modular de un proyecto Spring Boot.
- Resolver errores comunes al configurar un proyecto por primera vez.

Actividades:

- Validación del árbol de archivos y dependencias configuradas.
- Ejercicios guiados: crear rutas con @GetMapping, @PostMapping, @RequestParam, @PathVariable.

- Simulación de errores comunes: endpoints que no responden, rutas incorrectas, 404 Not Found, y su resolución con IA.
- Revisión con IA de nombres de clases, métodos, rutas y sugerencias de documentación.

CLASE 2: Servicios y Lógica de Negocio con Spring

Objetivos de aprendizaje:

- Separar responsabilidades entre controladores y lógica de negocio.
- Implementar servicios limpios y reutilizables.

Contenidos

- Inversión de dependencias con @Service y @Autowired.
- Separación por capas: patrón Service Layer.
- Creación de clases de servicio para manejar operaciones CRUD lógicas.
- Uso de @Component, @Bean, @Configuration según necesidad.
- Refactorización asistida por IA: dividir lógica compleja, crear métodos reutilizables, renombrar correctamente.

Proyecto en clase:

Crear un servicio LibroService con operaciones:

- listar libros,
- buscar por título,
- agregar libro,
- eliminar libro.
- Todo desde una lista en memoria (List<Libro>).

CLASE-TUTORÍA 2: Refuerzo en servicios, modularidad y depuración asistida

Objetivos de la CLASE-TUTORÍA:

- Afianzar la comprensión de la lógica por capas y su implementación real.
- Practicar la depuración de flujo entre controlador --> Servicio - -> Modelo.

Actividades:

- Rastreo del flujo de ejecución con puntos de ruptura (breakpoints) .
- Revisión del uso de @Autowired y principios de bajo acoplamiento.
- Diagnóstico con Herramientas IA: sugerencias para simplificar clases de servicio, detectar código duplicado o innecesario.
- Ejercicios prácticos: crear mini lógica de negocio para usuarios, productos o cursos.

CLASE 3: Principios SOLID y su aplicación práctica en Java

Objetivos de aprendizaje:

- Comprender los 5 principios SOLID como guía para un código mantenible y escalable.
- Aplicar ejemplos concretos en código Java de cada principio.
- Utilizar herramientas como IA para detectar y refactorizar violaciones a SOLID.

Contenidos

- S: Single Responsibility Principle (SRP)
- O: Open/Closed Principle (OCP)
- L: Liskov Substitution Principle (LSP)
- I: Interface Segregation Principle (ISP)
- D: Dependency Inversion Principle (DIP)
- Aplicación práctica con clases de ejemplo y análisis de código real.
- Refactorización con IA: dividir clases, delegar responsabilidades, abstraer dependencias.

Ejercicio práctico:

- Crear una aplicación Spring Boot que aplique al menos 3 principios SOLID en su diseño.
- Uso obligatorio de interfaces y separación de responsabilidades entre capas.

CLASE-TUTORÍA 3: Diagnóstico y refactorización basada en principios SOLID

Objetivos de la CLASE-TUTORÍA:

- Evaluar si los principios SOLID están siendo respetados en los proyectos personales.
- Proponer mejoras con IA y aplicar refactorizaciones en equipo.

Actividades:

- Lectura crítica de proyectos entregados por estudiantes (antes/después).
- Detección de clases con múltiples responsabilidades o fuerte acoplamiento.
- Taller: aplicar el principio de inversión de dependencias con interfaces y constructor injection.
- IA como mentor automático: sugerencias para mejorar estructuras según OOP.

CLASE 4: Proyecto API RESTful Completa con Spring Boot

Objetivos de aprendizaje:

- Integrar controladores, servicios y persistencia en un sistema real completo.
- Documentar y publicar la API con estándares profesionales.

Contenidos:

- Diseño del proyecto: entidad principal (Libro, Estudiante, Producto, etc.).
- Implementación completa de rutas REST (GET, POST, PUT, DELETE).
- Validaciones con javax.validation: @NotNull, @Email, @Size, etc.
- Introducción a Swagger para documentación automática con springdoc-openapi.
- Refactorización final con IA: código limpio, responsabilidades claras, documentación
- completa.

Proyecto sugerido:

- Crear una API RESTful para gestión de Programas con operaciones CRUD, búsqueda por
- nombre, validación de fechas y documentación pública.

CLASE-TUTORÍA 4: Evaluación del proyecto + revisión estructural completa

Objetivos de la CLASE-TUTORÍA:

- Validar la implementación del proyecto bajo criterios de calidad profesional.
- Aplicar revisión técnica completa con feedback personalizado.

Actividades:

- Presentación de proyectos por parte de los estudiantes: explicación de flujo, endpoints,
- lógica de negocio.
- Revisión cruzada entre compañeros: claridad, arquitectura, naming, validaciones.
- Diagnóstico final con herramientas IA: análisis de arquitectura, sugerencias de mejoras,
- advertencias.
- Publicación del proyecto en GitHub con instrucciones claras (README.md profesional).

MÓDULO 6: Bases de Datos Relacionales y NoSQL con Java y Spring Boot

Este módulo tiene como objetivo desarrollar competencias sólidas en el diseño, conexión y manipulación de bases de datos relacionales y NoSQL desde aplicaciones Java. Se trabajará con PostgreSQL y MongoDB, y se aprenderá a realizar operaciones CRUD, modelado de entidades, relaciones y optimización de consultas utilizando Spring Data JPA. Se incorporará el uso de IA para sugerencias de mapeo, creación de consultas y diseño de esquemas eficientes.

CLASE 1: Introducción a PostgreSQL y conexión desde Java

Objetivos de aprendizaje:

- Entender el rol de las bases de datos relacionales en aplicaciones empresariales.
- Modelar estructuras de datos eficientes y escalables.
- Conectar y consultar una base de datos PostgreSQL desde Java.

Contenidos

- Qué es una base de datos relacional? Características de PostgreSQL.
- Instalación, configuración y uso de herramientas como pgAdmin o DBeaver.
- Creación de esquemas, tablas, tipos de datos y restricciones (PRIMARY KEY, FOREIGN KEY, NOT NULL, etc.).
- Conceptos de normalización y modelado lógico de datos.
- Conexión desde Java usando Spring Boot + Spring Data JPA.
- Configuración del archivo application.properties y dependencias necesarias.
- Prueba de conexión con Herramientas IA: sugerencias para verificación automática.

Ejercicio práctico:

- Crear la base de datos devsenior_db, una tabla usuarios, y conectarla desde un proyecto Spring Boot básico.

CLASE-TUTORÍA 1: Modelado, conexión y validación de entidades

Objetivos de la CLASE-TUTORÍA:

- Asegurar el correcto diseño y conexión de las tablas desde el backend.
- Reforzar el uso de modelos de datos relacionales reales.

Actividades:

- Ejercicios de modelado: de requerimientos a esquemas con entidades y relaciones.
- Análisis de errores comunes: claves mal definidas, datos duplicados, tipos incorrectos.
- Revisión de conexión y mapeo con asistencia de IA: sugerencias de corrección y mejoras en las entidades.
- Refuerzo del concepto de relaciones 1 a 1, 1 a N y N a N en casos prácticos.

CLASE 2: Spring Data JPA – Entidades, Repositorios y Relaciones

Objetivos de aprendizaje:

- Mapear objetos Java a tablas de base de datos usando JPA.
- Implementar relaciones entre entidades de forma correcta.
- Realizar operaciones CRUD de manera profesional con Spring Data JPA.

Contenidos

- Uso de anotaciones @Entity, @Id, @GeneratedValue, @Column, @Table.
- Creación de clases Repository y uso de interfaces JpaRepository y CrudRepository.
- Relaciones entre entidades:
  - @OneToOne, @OneToMany, @ManyToOne, @ManyToMany.
  - Atributos de relación: cascade, fetch, mappedBy.
- Validación de relaciones complejas con ayuda de Herramientas IA.

Ejercicio práctico:

- Implementar las entidades Cliente, Pedido, Producto y sus relaciones.
- Crear controladores básicos para realizar operaciones CRUD.

CLASE-TUTORÍA 2: Refuerzo en entidades, relaciones y pruebas de integridad

Objetivos de la CLASE-TUTORÍA:

- Verificar la correcta implementación de relaciones y operaciones básicas en Spring Data.
- Afianzar el concepto de modelo de dominio coherente y escalable.

Actividades:

- Taller de creación y ajuste de relaciones entre entidades usando diagrama de clases.
- Verificación de consistencia entre clases Java y tablas reales con herramientas externas.
- Pruebas cruzadas con IA: detectar relaciones mal planteadas, nombres de métodos inadecuados y clases mal configuradas.
- Validación de datos: campos obligatorios, formatos, referencias.

CLASE 3: Consultas Avanzadas con JPQL, Criteria API y Spring Queries

Objetivos de aprendizaje:

- Crear consultas avanzadas en base a las necesidades del negocio.
- Optimizar consultas y estructuras para eficiencia en tiempo real.

Contenidos

- Introducción a JPQL (Java Persistence Query Language): queries basadas en clases, no en tablas.
- Uso de anotaciones @Query para crear consultas personalizadas.
- Consultas dinámicas con Criteria API.
- Métodos derivados (findByNombre, findByPrecioGreaterThan).
- Paginación y ordenamiento con Pageable, Sort.
- IA como generador automático de métodos de consulta y optimizador de filtros.

Ejercicio práctico:

Crear consultas personalizadas para obtener:

- Pedidos realizados en el último mes.
- Productos más vendidos.
- Clientes con más de X compras.

CLASE-TUTORÍA 3: Diagnóstico, rendimiento y consultas con IA

Objetivos de la CLASE-TUTORÍA:

- Asegurar la comprensión y correcta aplicación de consultas avanzadas.
- Analizar el rendimiento de las consultas y aplicar optimizaciones.

Actividades:

- Simulación de escenarios reales que requieren múltiples filtros y relaciones.
- Taller: consultas con combinaciones complejas usando Criteria API y paginación.
- Análisis de rendimiento con EXPLAIN ANALYZE desde pgAdmin y revisión con IA.
- Refactorización y limpieza de consultas usando recomendaciones automáticas.

CLASE 4: MongoDB – Introducción a NoSQL y conexión con Spring Boot

Objetivos de aprendizaje:

- Comprender los fundamentos de las bases de datos NoSQL.
- Implementar una aplicación con MongoDB y Spring Boot.

Contenido:

- ¿Qué es MongoDB? Estructura flexible, documentos y colecciones.
- Diferencias entre SQL y NoSQL (modelo, consultas, escalabilidad).
- Instalación de MongoDB local o en Atlas (cloud).
- Uso de spring-boot-starter-data-mongodb en un proyecto.
- Anotaciones @Document, @Id, @Field, repositorios MongoRepository.
- Operaciones CRUD básicas y consultas por campos.
- Uso de IA para transformar consultas relacionales a NoSQL.

Proyecto en clase:

- Crear una colección usuarios en MongoDB con documentos dinámicos y conectarla a una app Java.
- Implementar endpoints REST para registrar, consultar y eliminar usuarios.

CLASE-TUTORÍA 4: Práctica NoSQL y comparativa relacional vs. documental

Objetivos de la CLASE-TUTORÍA:

- Consolidar el conocimiento en bases de datos NoSQL.
- Evaluar en qué escenarios usar MongoDB vs. PostgreSQL.

Actividades:

- Taller práctico: construir una app de mensajería usando MongoDB.
- Comparación de rendimiento en lectura/escritura con colecciones grandes.
- Creación de consultas con filtros anidados y expresiones regulares (regex).
- Asistencia de IA para transformar estructuras rígidas en documentos flexibles.
- Discusión grupal: ¿cuándo preferir SQL? ¿cuándo es mejor usar NoSQL?

MÓDULO 7: Seguridad, JWT y APIs Avanzadas en Spring Boot

En este módulo el estudiante aprenderá a asegurar aplicaciones Java con Spring Security, implementar autenticación basada en tokens JWT (JSON Web Token), gestionar errores globalmente, y construir APIs más sólidas, organizadas y preparadas para producción. Todo será reforzado mediante refactorización asistida por IA para mantener código limpio, seguro y mantenible.

CLASE 1: Fundamentos de Seguridad en Aplicaciones Spring Boot

Objetivos de aprendizaje:

- Comprender los fundamentos de la seguridad en aplicaciones web modernas.
- Implementar mecanismos básicos de autenticación y autorización con Spring Security.
- Conocer el ciclo de vida de una solicitud protegida.

Contenidos

- ¿Qué es Spring Security? Componentes básicos y flujo de autenticación.
- Seguridad por defecto en Spring Boot.
- Configuración de seguridad personalizada con SecurityFilterChain.
- Rutas públicas vs. rutas protegidas (antMatchers, authorizeHttpRequests, etc.).
- Inyección de usuarios en memoria con UserDetailsService.
- Uso de Herramientas IA para generación automática de configuración básica de seguridad.

Ejercicio práctico:

- Proteger una API básica (/api/cursos) y permitir solo el acceso autenticado a rutas específicas.

CLASE-TUTORÍA 1: Flujo de autenticación, errores comunes y configuración de rutas seguras

Objetivos de la tutoría:

- Analizar y reforzar el entendimiento del flujo de seguridad.
- Configurar correctamente el acceso seguro a rutas según roles o permisos.

Actividades:

- Taller práctico: diferenciar rutas públicas y privadas, crear login básico en memoria.
- Diagnóstico de errores comunes (403 Forbidden, 401 Unauthorized).
- Refactorización asistida por IA de la clase de configuración de seguridad para hacerla más limpia y escalable.
- Casos de estudio: cómo proteger rutas en una API pública y qué endpoints deben ser expuestos sin seguridad.

CLASE 2: Implementación de Autenticación con JWT (JSON Web Tokens)

Objetivos de aprendizaje:

- Integrar JWT como mecanismo de autenticación sin sesiones.
- Implementar generación, validación y uso de tokens en APIs REST.

Contenidos

- ¿Qué es un JWT? Encabezado, payload y firma.
- Flujo completo de autenticación sin estado:
    - Inicio de sesión → generación del token → uso del token en peticiones → validación automática.
- Implementación de filtros personalizados para validar JWT en cada solicitud.
- Configuración de cabeceras (Authorization: Bearer TOKEN) y extracción del usuario.
- Uso de IA para autocompletar lógica de generación de token, extracción de claims y validación.

Ejercicio práctico:

- Implementar un endpoint /login que devuelva un JWT al autenticarse y proteger el acceso a /api/privado con dicho token.

CLASE-TUTORÍA 2: Refuerzo de JWT, manejo de cabeceras y validación de seguridad

Objetivos de la CLASE-TUTORÍA:

- Profundizar en la lógica de autenticación sin sesión.
- Asegurar la correcta configuración del flujo completo de login y validación de token.

Actividades:

- Simulación completa: login, obtención de token, uso en cliente (Postman, VSCode REST Client).
- Debug del filtro JWT con breakpoints y validación de claims.
- Asistencia de IA para sugerir estructuras reutilizables en autenticación.
- Discusión: ¿cómo almacenar tokens del lado del cliente? Buenas prácticas para proteger APIs públicas.

CLASE 3: Manejo de Excepciones Globales y Validaciones Personalizadas

Objetivos de aprendizaje:

- Implementar un sistema centralizado de manejo de errores.
- Estandarizar las respuestas de error en toda la API.
- Validar entradas de usuario y mostrar mensajes significativos.

Contenidos

- Creación de manejador global con @ControllerAdvice y @ExceptionHandler.
- Formato unificado de respuesta (timestamp, mensaje, estado, ruta).
- Validaciones con @Valid, @NotBlank, @Size, @Email, etc.
- Respuestas automáticas y personalizadas con BindingResult.
- Refactorización de mensajes y validaciones con ayuda de Herramientas IA.

Ejercicio práctico:

- Crear una clase UsuarioDTO con validaciones, controlar excepciones de forma centralizada y devolver errores estructurados al cliente.

CLASE-TUTORÍA 3: Evaluación y mejora del manejo de errores con IA

Objetivos de la CLASE-TUTORÍA:

- Consolidar la práctica de validaciones personalizadas y manejo uniforme de errores.
- Elevar la calidad del código con respuestas claras, seguras y profesionales.

Actividades:

- Revisión de flujos de error: validación, recursos no encontrados, excepciones técnicas.
- Refactorización con IA como manejador global: organización, claridad, patrones de respuesta.
- Taller: implementación de un endpoint /registro con múltiples validaciones, respuesta adecuada y prueba con datos incorrectos.

CLASE 4: Proyecto API Segura – Autenticación, Validación y Control de Acceso

Objetivos de aprendizaje:

- Integrar seguridad, autenticación JWT, validaciones y manejo de errores en un sistema real.
- Construir una API segura, funcional y lista para producción.

Contenidos:

- Diseño del proyecto: API RESTful segura para gestión de usuarios o productos.
- Roles de usuario (ADMIN, USER) y acceso condicional a rutas.
- Inicio de sesión con generación de JWT, protección de rutas por rol.
- Validaciones exhaustivas de entradas y manejo de errores personalizado.
- Documentación básica con Swagger para facilitar pruebas externas.

Proyecto en clase:

- **Sistema de gestión de usuarios con endpoints:**
  - /registro
  - o /login
  - o /usuarios (GET solo para ADMIN)
  - o /perfil (GET solo para el propio usuario autenticado)

CLASE-TUTORÍA 4: Evaluación del sistema seguro y mejora integral con IA

Objetivos de la tutoría:

- Evaluar la seguridad, estructura y consistencia del proyecto final del módulo.
- Realizar pruebas completas de flujo autenticado y protección de rutas.

Actividades:

- Presentación del flujo completo: login → token → consumo de recursos protegidos.
- Validación cruzada: pruebas con compañeros para comprobar errores, protección y mensajes.
- Evaluación con herramientas IA: sugerencias de refactorización en filtros, configuraciones, respuestas.
- Publicación del proyecto seguro en GitHub con documentación clara.

MÓDULO 8: React JS Moderno – Interfaces Dinámicas para Aplicaciones Full Stack

Este módulo introduce al estudiante en el desarrollo de interfaces modernas con React JS, utilizando tecnologías actuales como Vite, Hooks, Tailwind CSS y consumo de APIs con autenticación JWT. Se enseñan patrones profesionales para estructurar componentes, gestionar estado, manejar rutas y construir interfaces atractivas, funcionales y seguras. El backend de práctica será Spring Boot (módulos anteriores) y se incluye refuerzo con herramientas de IA para mejorar el rendimiento y la calidad del código.

CLASE 1: Introducción a React con Vite, Hooks y estructura moderna

Objetivos de aprendizaje:

- Crear interfaces modernas con React de forma eficiente y modular.
- Utilizar Hooks para el manejo de estado y efectos.
- Conocer la estructura ideal de proyectos frontend modernos.

Contenidos

- ¿Por qué usar React y Vite en 2025? Ventajas frente a CRA.
- Instalación de Vite + React desde Herramientas IA.
- Estructura profesional de carpetas: components, pages, services, routes.
- Uso de useState y useEffect para manejar el estado y cargar datos iniciales.
- Integración de Tailwind CSS en el proyecto (opcional: DaisyUI o Material UI).
- Creación de componentes dinámicos y reutilizables.
- Introducción a Fetch API y consumo de datos desde un servidor local.
- Asistencia de IA: generación de componentes reutilizables y estructuras base.

Ejercicio práctico:

- Crear un frontend para listar cursos desde un endpoint local (GET /api/cursos).

CLASE-TUTORÍA 1: Consolidación de estructura, Hooks y manejo de listas

Objetivos de la CLASE-TUTORÍA:

- Validar comprensión de la estructura modular y el uso correcto de Hooks.
- Reforzar la lógica para renderizado dinámico de datos y consumo de APIs.

Actividades:

- Acompañamiento en instalación de Tailwind CSS y creación de diseño responsive.
- Ejercicios guiados para practicar useState, useEffect y renderizado condicional.
- Diagnóstico de errores típicos (dependencias infinitas, undefined, mal uso de keys).
- Uso de IA para sugerencias de estructura de props, refactor de listas y lógica de carga.

CLASE 2: Componentes, Props, Formularios y Manejo de Eventos

Objetivos de aprendizaje:

- Diseñar componentes reutilizables y manejar el flujo de datos entre ellos.
- Crear formularios con validaciones y envíos de datos a una API backend.

Contenidos

- Descomposición de la UI en componentes pequeños y reutilizables.
- Paso de datos con props, eventos personalizados (onClick, onChange).
- Control de formularios: input, select, textarea con estado controlado.
- Validación básica de campos en frontend antes de enviar al servidor.
- Envío de datos al backend con fetch o axios (POST /api/curso).
- Refactor con IA: sugerencias para mejorar estructura, reusabilidad y validación.

Ejercicio práctico

- Formulario para agregar cursos con validación de campos requeridos y envío de datos al backend Spring Boot.

CLASE-TUTORÍA 2: Buenas prácticas de componentes, formularios y UX

Objetivos de la CLASE-TUTORÍA:

- Consolidar el conocimiento de construcción de formularios y su conexión con el backend.
- Reforzar patrones de comunicación entre componentes y uso profesional de props.

Actividades:

- Taller colaborativo: creación de componentes InputGroup, Button, FormCard.
- Uso de mensajes de error visuales (toast, alertas, validaciones inline).
- Diagnóstico con IA: análisis de duplicación de código, errores de lógica y
- sugerencias para UX.
- Implementación de feedback visual y loading states.

CLASE 3: Ruteo, Estado Global y Manejo de Autenticación con JWT

Objetivos de aprendizaje:

- Implementar navegación entre vistas de forma eficiente.
- Manejar estado global básico con Context API y proteger rutas.

Contenidos

- Instalación y configuración de react-router-dom.
- Definición de rutas públicas y privadas (/login, /dashboard, /perfil).
- Uso de useNavigate, useParams, useLocation.
- Manejo de tokens JWT:
  - Guardar y recuperar del localStorage.
  - Añadir token a headers para proteger llamadas al backend.
- Estado global simple con React Context + useReducer.
- IA para refactorización de rutas, autenticación y estructuras de navegación.

Ejercicio práctico:

- **Crear un flujo de autenticación:**
  - Login
  - Guardado del token
  - Acceso a una ruta protegida con información del usuario autenticado

CLASE-TUTORÍA 3: Navegación dinámica y flujo seguro de usuarios autenticados

Objetivos de la tutoría:

- Asegurar la correcta implementación de autenticación, navegación y protección de vistas.
- Reforzar la lógica de estado global y validación del flujo de sesión.

Actividades:

- Revisión del flujo login → token → dashboard.
- Simulación de navegación protegida y redirección automática si el usuario no está autenticado.
- Uso de IA para refactorizar funciones de login, lógica condicional y renderizado por rol.
- Validación cruzada entre estudiantes: acceso correcto a rutas, token válido, expiración.

CLASE 4: Proyecto Frontend Integrador – Panel de Gestión con React

Objetivos de aprendizaje:

- Aplicar todos los conceptos en un proyecto completo y funcional.
- Desarrollar un frontend moderno, seguro y conectado a una API real.

Contenidos:

- Diseño de vistas:
  - Inicio de sesión
  - Dashboard
  - Formulario de registro
  - Listado dinámico
  - Perfil de usuario
- Manejo completo del estado, rutas, autenticación y comunicación con el backend Spring Boot.
- Aplicación de Tailwind para estilo moderno y responsivo.
- Refactor con herramientas IA:
  - Limpieza de código
  - Mejora de estructura
  - Estilo de componentes
  - Eliminación de redundancia

Proyecto sugerido:

- Sistema de gestión académica (estudiantes, cursos, calificaciones) con login, rutas protegidas y CRUD completo desde frontend.

CLASE-TUTORÍA 4: Evaluación visual, funcional y técnica del proyecto React

Objetivos de la CLASE-TUTORÍA:

- Evaluar la calidad visual, técnica y funcional del proyecto.
- Detectar errores comunes, problemas de usabilidad y sugerencias de mejora.

Actividades:

- Presentación de proyectos: estructura, diseño, navegación, flujo de datos.
- Revisión entre compañeros: experiencia de usuario, fluidez, feedback visual.
- Revisión de código asistida por IA: estructura de componentes, uso adecuado de hooks,modularidad.
- Publicación del proyecto en GitHub + Netlify/Vercel con instrucciones para clonar y correr.

MÓDULO 9: Integración Full Stack Moderna – Spring Boot + React con Seguridad JWT

En este módulo, los estudiantes integrarán sus conocimientos de backend y frontend para construir un sistema real completo con comunicación segura. Se utilizarán herramientas modernas (Spring Boot, React, JWT, TailwindCSS, Axios), y se aplicarán buenas prácticas profesionales para asegurar escalabilidad, modularidad y mantenimiento de código. El proyecto real estará basado en un Sistema de Gestión Académica Universitaria.

CLASE 1: Conexión Full Stack – Arquitectura y Flujo de Comunicación

Objetivos de aprendizaje:

- Comprender el flujo completo de datos entre frontend y backend.
- Configurar correctamente CORS y la comunicación HTTP segura.
- Realizar pruebas de consumo de APIs protegidas con JWT.

Contenidos

- Arquitectura cliente-servidor RESTful.
- Configuración de CORS en Spring Boot (WebMvcConfigurer).
- Estructura de carpetas profesional en React y Spring.
- Consumo de endpoints seguros desde React (axios + Authorization: Bearer TOKEN).
- Uso de IA para configurar cabeceras, rutas seguras y diagnóstico de errores HTTP comunes.

Ejercicio práctico:

- Conectar una interfaz React con Spring Boot para consultar la lista de cursos desde el endpoint /api/cursos, protegido por token JWT.

CLASE-TUTORÍA 1: Diagnóstico de Integración y Comunicación Segura

Objetivos de la CLASE-TUTORÍA:

- Validar la conexión entre proyectos.
- Resolver problemas comunes al consumir APIs con token.

Actividades:

- Pruebas con Postman y cliente HTTP.
- Simulación de errores: token expirado, sin token, CORS mal configurado.
- Configuración de axios.interceptors para añadir token automáticamente.
- Refactorización de servicios frontend con ayuda de IA.

CLASE 2: CRUD Real en Aplicación Universitaria – Cursos y Estudiantes

Objetivos de aprendizaje:

- Implementar un CRUD completo con conexión segura en ambos lados.
- Organizar el proyecto con arquitectura real de empresa.

Contenidos

- Endpoints REST en backend: GET, POST, PUT, DELETE.
- Frontend: formularios con validaciones, modales, estados de éxito/error.
- Cabeceras autenticadas y manejo de errores (try/catch, toast).
- Separación de lógica en capas (services, utils, pages, components).
- Uso de IA para sugerencias de organización y reducción de código duplicado.

Ejercicio práctico:

- Implementar un CRUD de asignaturas universitarias, conectado a estudiantes registrados.

CLASE-TUTORÍA 2: Validación de Datos, UX y Flujo Seguro

Objetivos de la CLASE-TUTORÍA:

- Reforzar el flujo de creación, actualización y eliminación de datos.
- Garantizar buena experiencia de usuario y seguridad en la gestión de recursos.

Actividades:

- Pruebas cruzadas con datos incorrectos, duplicados, inválidos.
- Evaluación de feedback visual (toast, loading, confirmaciones).
- Refactorización de componentes y lógica en frontend con ayuda de IA.
- Validación de mensajes de error estructurados provenientes del backend.

CLASE 3: Diseño Visual, Accesibilidad y Modularización de Vistas

Objetivos de aprendizaje:

- Aplicar diseño moderno y responsive con herramientas visuales profesionales.
- Modularizar vistas por roles y funcionalidades reales.

Contenidos

- Uso de Tailwind CSS y Componentes reutilizables (Card, Button, Input, Modal).
- Accesibilidad, diseño centrado en el usuario, contraste, interacciones suaves.
- Organización por roles: vistas para estudiante, docente y administrador.
- IA como diseñador: sugerencias de refactorización visual y estructura limpia.

Proyecto en clase:

- Rediseñar las vistas de gestión de asignaturas y perfil académico del estudiante con estilo limpio, navegación fluida y feedback visual claro.

CLASE-TUTORÍA 3: Revisión de UX, Accesibilidad y Arquitectura Visual

Objetivos de la CLASE-TUTORÍA:

- Asegurar que la aplicación sea clara, accesible y profesional.
- Mejorar estructura de carpetas y componentes visuales.

Actividades:

- Testeo de la app desde distintos roles: ¿qué ve cada usuario?, ¿cómo interactúa?
- Diagnóstico con IA del uso de componentes repetidos, estilos inconsistentes.
- Refactor visual: extractar componentes comunes, limpiar CSS, modularizar rutas.
- Retroalimentación entre equipos: presentación de la app como si fuera entregada a un cliente.

CLASE 4: Proyecto Real Full Stack – Sistema de Gestión Académica

Objetivos de aprendizaje:

- Integrar conocimientos completos en un sistema funcional y seguro.
- Presentar una aplicación profesional con estándares de la industria.

Proyecto en clase:

- Proyecto real: Sistema Universitario de Gestión Académica
- Roles: ADMIN, DOCENTE, ESTUDIANTE.
- Funcionalidades:
  - Autenticación con JWT.
  - Inscripción y asignación de cursos.
  - Gestión de usuarios y materias.
  - Consulta de historial académico.
- Comunicación protegida por token.
- Diseño visual completo y limpio.
- Refactorización con IA: control de calidad, sugerencias de mejora, documentación.

CLASE-TUTORÍA 4: Evaluación Final y Presentación Profesional del Proyecto Real

Objetivos de la CLASE-TUTORÍA:

- Validar el cumplimiento completo del sistema.
- Dejar lista la aplicación para incluir en portafolios y GitHub profesional.

Actividades:

- Presentación por equipos del proyecto real con enfoque técnico y funcional.
- Evaluación por rúbrica profesional: estructura, calidad, navegación, código, seguridad.
- Refuerzo de publicación en GitHub con README.md detallado, estructura clara y despliegue en Railway (backend) + Netlify (frontend).
- Asistencia IA para checklist de código limpio, estructura óptima, roles y tokens funcionales.

Proyecto Final Real (Integrador del Módulo):

- **Sistema de Gestión Académica para Universidades**
  - **Frontend:** React + Tailwind + Axios
  - **Backend:** Spring Boot + PostgreSQL + JWT
  - **Despliegue:** Railway + Netlify
  - **Funcionalidades clave:**
    - Login y Registro por rol
    - Gestión de asignaturas
    - Asignación de estudiantes y docentes
    - CRUD de cursos
    - Historial académico por estudiante

MÓDULO 10: Testing, Calidad y Buenas Prácticas para Aplicaciones Profesionales

Este módulo enseña a validar la calidad funcional, estructural y visual de los sistemas Full Stack construidos con Java + Spring Boot + React, mediante pruebas automatizadas, métricas de cobertura, pruebas end-to-end y principios de buenas prácticas profesionales. Se incorpora el uso de IA para análisis de código, refactorización y detección de errores. El proyecto real será un Sistema de Gestión de Pacientes para una Clínica u Hospital.

CLASE 1: Pruebas Unitarias y Mocks en Java con JUnit + Mockito

Objetivos de aprendizaje:

- Validar componentes del backend de forma aislada.
- Utilizar pruebas automatizadas para asegurar comportamiento esperado de servicios.

Contenidos

- ¿Qué es una prueba unitaria? ¿Por qué se debe automatizar?
- Instalación y configuración de JUnit 5.
- Pruebas básicas de clases de servicio (assertEquals, assertThrows, assertTrue, etc.).
- Introducción a Mockito:
    - @Mock, @InjectMocks, when().thenReturn()
    - Pruebas de servicios con dependencias falsas
- Revisión y generación de pruebas con Herramientas IA: sugerencias de assertions, mocks y refactorización.

Ejercicio práctico:

- Escribir pruebas unitarias para los servicios PacienteService y ConsultaMedicaService de una API hospitalaria.

CLASE-TUTORÍA 1: Refuerzo de pruebas unitarias y principios TDD

Objetivos de la CLASE-TUTORÍA:

- Consolidar las pruebas unitarias como herramienta profesional de prevención de errores.
- Introducir el enfoque Test-Driven Development (TDD).

Actividades:

- Revisión de pruebas unitarias: cobertura, legibilidad, claridad de casos de prueba.
- Refactorización con IA de pruebas largas, repetitivas o mal estructuradas.
- Mini reto TDD: escribir primero las pruebas y luego la clase que las hace pasar (ej:agendamiento de citas).
- Discusión: qué probar, qué no probar, y cuándo hacerlo.

CLASE 2: Pruebas Automatizadas de Frontend con Vitest + React Testing Library

Objetivos de aprendizaje:

- Validar componentes visuales y lógicos del frontend.
- Automatizar el comportamiento de interfaces desde el código.

Contenidos:

- Instalación de Vitest y React Testing Library en un proyecto Vite.
- Pruebas de renderizado, interacción (click, change), visibilidad y estructura del DOM.
- Mocks de funciones, componentes y API.
- Captura de errores de lógica visual (ej: botón que no se desactiva, inputs vacíos).
- Uso de IA para generación de pruebas automáticas y cobertura de eventos.

Proyecto en clase:

- Crear pruebas para el componente FormularioConsulta del sistema hospitalario (validación de campos, respuesta visual, renderizado de mensajes).

CLASE-TUTORÍA 2: Diagnóstico de pruebas de frontend y mejora con IA

Objetivos de la CLASE-TUTORÍA:

- Evaluar calidad, utilidad y legibilidad de las pruebas de frontend.
- Promover el testing visual como parte del desarrollo profesional.

Actividades:

- Simulación de errores visuales en componentes y su captura con pruebas.
- Revisión asistida por Herramientas IA: sugerencias de refactor en casos de prueba y estructura de test suites.
- Integración de screen.debug() y visualización de DOM de prueba.
- Comparación entre pruebas manuales vs pruebas automatizadas.

CLASE 3: Pruebas de Integración y E2E con Cypress + Métricas de Cobertura

Objetivos de aprendizaje:

- Simular flujos reales del usuario en la aplicación completa.
- Medir cobertura del código y evaluar calidad del sistema.

Contenidos:

- Instalación de Cypress para pruebas End-to-End.
- Grabación de pruebas con el Test Runner.
- Simulación de escenarios reales:
  - Login
  - Agendar cita médica
  - Eliminar paciente
  - Navegar entre vistas protegidas
- Integración de métricas de cobertura con JaCoCo (backend) y vitest-coverage (frontend).
- Análisis de resultados con ayuda de Herramientas IA.

Ejercicio práctico:

- Implementar 3 pruebas e2e completas sobre el sistema hospitalario (crear paciente, editar datos, ver historial).

CLASE-TUTORÍA 3: Evaluación de cobertura y automatización completa de flujo real

Objetivos de la CLASE-TUTORÍA:

- Validar que la aplicación reacciona correctamente bajo flujos reales de usuario.
- Diagnosticar código no cubierto por pruebas.

Actividades:

- Medición de cobertura del backend (.exec → report.html de JaCoCo).
- Pruebas automatizadas con Cypress ejecutadas sobre servidor local (localhost:5173, localhost:8080).
- Revisión con IA de código no cubierto y recomendaciones de testing adicionales.
- Discusión: ¿qué es suficiente cobertura?, ¿cuándo automatizar y cuándo no?

CLASE 4: Proyecto Real – Sistema de Gestión Hospitalaria con Control de Calidad

Objetivos de aprendizaje:

- Consolidar todas las técnicas de testing en un sistema completo y real.
- Demostrar dominio de pruebas y control de calidad.
- **Contenidos del Proyecto Real: Sistema de Gestión Hospitalaria con Seguridad y Pruebas Automatizadas**

  Backend (Spring Boot):
  - Control de usuarios (médicos, pacientes, administrativos)
  - Gestión de citas, diagnósticos, historial clínico
  - Autenticación con JWT

  Frontend (React):
  - Login, panel administrativo, agenda de citas
  - Visualización de pacientes, carga de historias médicas

  Pruebas:
  - Unitarias: servicios, repositorios (JUnit + Mockito)
  - Frontend: componentes y formularios (Vitest)
  - E2E: flujos completos (Cypress)

  Cobertura:
  - generación de reportes de calidad de backend y frontend

  Refactorización con IA:
  - Pruebas no cubiertas
  - Componentes innecesarios
  - Código redundante

  Documentación final:
  - README.md con instrucciones para correr pruebas, ejemplos,credenciales de prueba.

CLASE-TUTORÍA 4: Revisión Final de Calidad, Presentación y Auditoría Profesional

Objetivos de la CLASE-TUTORÍA:

- Realizar una evaluación final del sistema desde la perspectiva de calidad.
- Dejar el proyecto listo para producción o portafolio profesional.

Actividades:

- Presentación del sistema con enfoque en calidad: ¿qué se probó?, ¿cómo se automatizó?,
- ¿qué errores se previenen?

- Revisión de reportes de cobertura: ¿qué quedó sin probar?, ¿por qué?
- Revisión con Herramientas IA: última limpieza, análisis de flujos, detección de puntos débiles.
- Preparación para publicación: badges de cobertura, scripts de test, instrucciones y mejoras sugeridas.

MÓDULO 11: DevOps y CI/CD Moderno – Automatización y Despliegue en Entornos Reales

Este módulo enseña a implementar prácticas modernas de DevOps: integración continua, despliegue automático, contenerización, variables de entorno, monitoreo y ejecución en plataformas cloud. Los estudiantes trabajarán en un proyecto real de una empresa de servicios digitales, desplegando su sistema en la nube usando Docker, GitHub Actions, Railway y Vercel, con calidad de código y seguridad.

CLASE 1: Introducción al Pensamiento DevOps y Automatización de Procesos

Objetivos de aprendizaje:

- Comprender la filosofía DevOps y su impacto en el desarrollo moderno.
- Automatizar tareas repetitivas para aumentar la productividad y seguridad.

Contenidos

- ¿Qué es DevOps y por qué es indispensable en la industria?
- Principios: Integración Continua (CI), Despliegue Continuo (CD), entrega continua.
- Herramientas clave: GitHub Actions, Docker, Railway, Vercel.
- Configuración de scripts automatizados para pruebas, compilación y validación.
- Herramientas IA como asistente de automatización: generación de flujos, Dockerfiles, YAMLs.

Ejercicio práctico:

Crear un pipeline básico de CI para un backend Spring Boot:

- Build automático + pruebas + chequeo de código al hacer push a GitHub.

CLASE-TUTORÍA 1: Implementación de CI y verificación de calidad automatizada

Objetivos de la CLASE-TUTORÍA:

- Consolidar el uso de GitHub Actions como herramienta de automatización.
- Validar el correcto funcionamiento del pipeline.

Actividades:

- Análisis de workflows YAML con IA: estructura, triggers, pasos.
- Ejercicio: modificar el pipeline para ejecutar pruebas unitarias en cada commit.
- Integración con SonarCloud o cobertura (JaCoCo) como parte del pipeline.
- Lectura e interpretación de logs de ejecución y fallos automáticos.

CLASE 2: Docker – Contenerización Profesional de Backend y Frontend

Objetivos de aprendizaje:

- Crear imágenes portables y consistentes de backend y frontend.
- Preparar los servicios para producción en cualquier entorno.

Contenidos

- ¿Qué es Docker y por qué es fundamental?
- Instalación y configuración en local.
- Creación de Dockerfile para backend Spring Boot.
- Construcción y ejecución de contenedores: docker build, docker run, docker ps.
- Docker para React: Vite + Nginx + Dockerfile.
- Uso de Docker Compose para ejecutar múltiples servicios.
- Asistencia de IA para generación y optimización de Dockerfiles y docker-compose.yml.

Ejercicio práctico:

- Crear contenedores para backend y frontend y ejecutarlos en red local usando Docker Compose.

CLASE-TUTORÍA 2: Diagnóstico de contenedores, puertos y redes

Objetivos de la CLASE-TUTORÍA:

- Validar la integración y funcionamiento de los servicios dentro de contenedores.
- Diagnosticar problemas comunes y optimizar la configuración.

Actividades:

- Revisión del flujo de puertos expuestos.
- Diagnóstico de errores: redirección de React, CORS entre contenedores.
- Testeo cruzado: colegas consumen tu servicio desde sus máquinas.
- Refactorización IA de Dockerfile para mejorar caché, build time y limpieza.

CLASE 3: Despliegue Profesional en Railway y Vercel + Variables de Entorno

Objetivos de aprendizaje:

- Desplegar aplicaciones reales con separación de entornos y configuración segura.
- Automatizar despliegues a plataformas modernas.

Contenidos

- Creación de cuenta en Railway (backend) y Vercel (frontend).
- Deploy automático con GitHub como origen.
- Configuración de env vars (secretos, URL del backend, claves).
- Conexión entre proyectos ya desplegados (React ↔ Spring).
- Testing en producción con curl, Postman, navegador.
- Asistencia IA para gestión de variables, optimización de CI/CD.

Ejercicio prácico:

- Subir el sistema completo de la empresa de servicios digitales (usuarios, facturación, módulos internos) a producción en Railway y Vercel.

CLASE-TUTORÍA 3: Validación del sistema en producción y refuerzo de automatización

Objetivos de la CLASE-TUTORÍA:

- Evaluar que la app corra correctamente en la nube.
- Asegurar calidad, escalabilidad y comportamiento esperado.

Actividades:

- Testeo real: login, token, navegación, CRUD, errores.
- Revisión de logs en Railway: errores de deploy, puertos, rutas.

- Carga de imágenes, uso de HTTPS, manejo de tokens en headers.
- Revisión con IA: análisis de logs, fallos y mejoras sugeridas.
- Recomendaciones de escalabilidad y monitoreo básico.

CLASE 4: Proyecto Real DevOps – Sistema Empresarial con Despliegue Automatizado

Objetivos de aprendizaje:

- Integrar DevOps en un sistema empresarial completo.
- Implementar CI/CD, contenerización y despliegue real con entornos separados.

Proyecto Real:

- Sistema de Gestión de Proyectos para una Empresa de Servicios Digitales

Módulos:

- Login y autenticación por rol
- Gestión de proyectos y tareas
- Administración de clientes y facturación
- Historial de actividades y notificaciones

CI/CD: GitHub Actions para pruebas + deploy automático
Contenerización: Docker + Compose
Despliegue: Backend en Railway / Frontend en Vercel
Documentación: Instrucciones técnicas, variables, rutas, usuarios de prueba
IA: Revisión final y optimización del pipeline y estructura

CLASE-TUTORÍA 4: Evaluación del Proyecto DevOps + Checklist Profesional de Producción

Objetivos de la CLASE-TUTORÍA:

- Validar que el sistema esté completamente desplegado y automatizado.
- Certificar la preparación para entornos reales de producción.

Actividades:

- Verificación en vivo: abrir el sistema online, navegar, ejecutar flujos.
- Validación técnica: logs, velocidad, comportamiento esperado.
- Revisión cruzada con Herramientas IA: sugerencias finales, mejoras al pipeline, optimización de imagen Docker.
- Publicación del proyecto: portafolio, presentación institucional, defensa técnica del flujo CI/CD.

MÓDULO 12: Proyecto Final Integrador + Preparación Profesional del Desarrollador Senior

Este módulo es la culminación de la ruta formativa. El estudiante desarrollará un sistema empresarial real, aplicando todas las tecnologías y buenas prácticas aprendidas: Java, Spring Boot, JWT, React, Docker, CI/CD, pruebas automatizadas, despliegue, diseño UX/UI y control de calidad. Además, se brindará una preparación intensiva para la vida profesional, incluyendo defensa técnica del proyecto, armado de portafolio, revisión de CV y simulación de entrevistas técnicas.

PROYECTO FINAL REAL

Sistema Integral de Gestión Administrativa para una Red de Clínicas Privadas

- Características del proyecto real:

Desarrollo Full Stack completo:

- **Backend:** Spring Boot + PostgreSQL + JWT
- **Frontend:** React JS + Vite + Tailwind CSS
- I**nfraestructura:** Docker, GitHub Actions, Railway, Vercel
- **Calidad:** JUnit, Cypress, SonarCloud, cobertura + documentación

Acceso por roles:

- Administrador General, Personal Médico, Personal Administrativo

Módulos:

- Gestión de pacientes y usuarios
- Agendamiento de citas
- Facturación electrónica
- Historial clínico digital
- Reportes administrativos y dashboards

Seguridad y privacidad de datos conforme a estándares reales de la industria

CLASE 1: Kickoff del Proyecto Real + Análisis de Requerimientos Profesionales

Objetivos de aprendizaje:

- Asumir el rol de desarrollador en un proyecto real del mundo empresarial.
- Organizar el trabajo en equipo, estructurar un backlog técnico y diseñar la arquitectura.

Contenidos

- Revisión de requerimientos técnicos y funcionales del sistema clínico.
- Elaboración de documentos iniciales:
  - Documento de alcance funcional
  - Documento de arquitectura
  - Diagrama de entidades y relaciones
- Revisión de estándares exigidos: CI/CD, pruebas, documentación técnica, roles, logs, seguridad.
- Asignación de responsabilidades por capas (Frontend, Backend, DevOps, QA).
- Uso de Herramientas IA como arquitecto asistente: sugerencias para organización, refactor de responsabilidades y control de deuda técnica.

CLASE-TUTORÍA 1: Evaluación de arquitectura y retroalimentación profesional del planteamiento

Objetivos de la CLASE-TUTORÍA:

- Validar la propuesta arquitectónica y asegurar una base técnica sólida.
- Prevenir errores comunes de modelado y desacoplamiento.

Actividades:

- Presentación técnica del diseño ante el mentor.
- Análisis de riesgos del proyecto.
- Revisión del modelo ER, lógica de negocio y autenticación esperada.
- IA como validador: detección de posibles problemas de flujo, acoplamiento excesivo o rutas mal diseñadas.

CLASE 2: Desarrollo Guiado del Sistema – Backend, Frontend y Automatización

Objetivos de aprendizaje:

- Construir los módulos funcionales del sistema bajo estándares reales.
- Asegurar seguridad, modularidad, pruebas y automatización desde el desarrollo.

Contenidos

- **Backend:**
  - API REST documentada y autenticada con JWT
  - Servicios, entidades, repositorios, validaciones
  - Seguridad por rol, manejo de errores, logs
- **Frontend:**
  - Vistas protegidas por sesión
  - Formulario de agendamiento de citas
  - Visualización del historial médico
  - Dashboards y reportes en tiempo real
- **Automatización:**
  - CI/CD con GitHub Actions
  - Docker Compose multi-contenedor
  - Variables de entorno por entorno (dev/prod)

Ejercicio clave:

- Sistema funcional completo con flujo agendamiento → confirmación → historial.

CLASE-TUTORÍA 2: Revisión de Código Profesional y Simulación de QA Interno

Objetivos de la CLASE-TUTORÍA:

- Validar calidad del código, uso de patrones, pruebas y seguridad.
- Simular auditoría interna de una empresa antes de salida a producción.

Actividades:

- Validación de criterios: código limpio, buenas prácticas, nomenclatura, organización.
- Revisión cruzada entre equipos.
- Checklist de calidad técnica (mentores + Herramientas IA):
  - ¿Está probado?,
  - ¿Es seguro?,
  - ¿Está desacoplado?,
  - ¿Está documentado?,
  - ¿Es escalable?

CLASE 3: Despliegue Final y Documentación Técnica del Proyecto

Objetivos de aprendizaje:

- Preparar y desplegar el sistema completo con estabilidad y documentación profesional.
- Dejar el sistema funcionando en entornos productivos reales.

Contenidos:

***Despliegue final:**

- Backend → Railway
- Frontend → Vercel
- Base de datos en cloud
- Documentación de variables, endpoints y rutas

***Pruebas End-to-End con Cypress y carga de datos reales**
***Configuración de logs, errores personalizados, backups**
***Documentación final:**

- Instructivo de despliegue
- Diagramas técnicos
- Manual de usuario y administrador
- README.md completo con scripts de pruebas y credenciales de demo

***IA como auditor final del sistema antes de entrega**

CLASE-TUTORÍA 3: Testeo final, verificación funcional y checklist pre-producción

Objetivos de la CLASE-TUTORÍA:

- Validar el correcto funcionamiento completo del sistema.
- Ejecutar checklist de preproducción.

Actividades:

- Pruebas funcionales y técnicas completas (API, UI, CI/CD).
- Validación de seguridad, rendimiento, manejo de errores.
- Revisión final con Herramientas IA: cobertura, duplicaciones, estructura de carpetas, comentarios.
- Preparación para presentación pública / institucional.

CLASE 4: Defensa Técnica del Proyecto + Preparación Profesional para el Empleo

Objetivos de aprendizaje:

- Defender el proyecto como en una entrevista laboral o presentación a cliente.
- Prepararse como desarrollador profesional listo para el mundo laboral.

Contenidos:

Presentación técnica:

- Stack utilizado
- o Arquitectura lógica y física
- o Roles, rutas, flujos y seguridad

Simulación de entrevista técnica:

- Justificación de decisiones tecnológicas
- Explicación de diseño de base de datos
- Ejecución de pruebas en vivo
- Preguntas tipo cliente / líder técnico

Preparación profesional:

- Revisión de CV técnico
- Optimización de perfil de LinkedIn
- Cómo documentar un proyecto en GitHub profesionalmente
- Recursos para aplicar a ofertas laborales internacionales

CLASE-TUTORÍA 4: Retroalimentación final y hoja de ruta postgraduación

Objetivos de la CLASE-TUTORÍA:

- Cerrar el ciclo formativo con retroalimentación individual.
- Proyectar los próximos pasos en la carrera profesional del estudiante.

Actividades:

- Evaluación personalizada del proyecto y portafolio.
- Recomendaciones personalizadas por el mentor.
- Reto profesional: aplicar a 3 ofertas reales con el proyecto como carta de presentación.
- Conclusión emocional y profesional del proceso.

Conclusiones

Java Senior con IA: La Ruta Profesional Definitiva no es solo un programa, es una experiencia de transformación profesional. A lo largo de 12 módulos intensivos, con acompañamiento experto, proyectos reales y herramientas de Inteligencia Artificial, te convertirás en un desarrollador capaz de enfrentar los desafíos técnicos y humanos del mundo real.

Te formarás en backend, frontend, bases de datos, DevOps, testing y arquitectura moderna, pero también desarrollarás visión estratégica, pensamiento crítico y habilidades para integrarte con éxito en equipos de desarrollo del más alto nivel.

Al finalizar esta travesía:

- Habrás construido y desplegado sistemas reales utilizados por empresas, clínicas y universidades.
- Tendrás un portafolio profesional en GitHub, con documentación, calidad y CI/CD implementado.
- Estarás preparado para entrevistas técnicas, retos laborales y trabajo remoto internacional.
- Hablarás el idioma del código y del mercado global, gracias al programa de inglés técnico English Boost con docentes nativos.
- Y lo más importante: tendrás la confianza, el criterio y la preparación para ser un verdadero Java Developer Senior.

Porque el mundo necesita desarrolladores que no solo sepan programar, sino que sepan pensar, construir, liderar y evolucionar con la tecnología.

Tu futuro comienza ahora. La industria te está esperando.

dev Senior

“Tu ruta ya está trazada. El futuro empieza contigo.”
La industria te está esperando.