MINISTERIO DE EDUCACIÓN

DIRECCIÓN DE FORMACIÓN TÉCNICA SUPERIOR
Instituto de Formación Técnica Superior Nº 29

PROGRAMA COMPLETO DE LA CÁTEDRA

DESARROLLO DE SISTEMAS WEB BACK END

CURSO: Segundo
CICLO LECTIVO: 2025 **CUATRIMESTRE:** PRIMERO
CANTIDAD DE HORAS SEMANALES: Nueve horas cátedra
PROFESOR/A: García Ontiveros Emir Eliezer

PRESENTACIÓN Y FUNDAMENTACIÓN DE LA CÁTEDRA

El módulo **Desarrollo de Sistemas Web Back End** tiene como objetivo brindar a los estudiantes una formación sólida en la creación de aplicaciones web, enfocándose en el desarrollo del lado del servidor. Como parte de la carrera de **Técnico Superior en Desarrollo de Software**, este curso profundiza en los principios de la **Programación Orientada a Objetos (POO)** y la **Programación Basada en Eventos**, aplicados a **JavaScript** dentro del entorno **Node.js**.

El desarrollo **backend** es un pilar fundamental en la construcción de aplicaciones web modernas, ya que permite la interacción eficiente con bases de datos, la gestión de usuarios y la implementación de lógica de negocio. Los desarrolladores backend son responsables de garantizar el correcto procesamiento de datos, la seguridad de la información y la optimización del rendimiento de las aplicaciones.

A través de un enfoque práctico basado en **proyectos**, los estudiantes desarrollarán habilidades clave en el diseño y la implementación de soluciones computacionales, utilizando algoritmos y estructuras de datos adecuadas. Además, se hará especial hincapié en buenas prácticas de seguridad, arquitectura de software y manejo eficiente de APIs.

Este curso no solo fortalece el pensamiento lógico y resolutivo de los estudiantes, sino que también los prepara para enfrentar desafíos avanzados en el desarrollo de software, brindándoles herramientas esenciales para su inserción en el mundo profesional.

OBJETIVOS DE LA MATERIA

El módulo se encuentra en el segundo año de la carrera Técnico Superior en Desarrollo de Software. Las materias correlativas incluyen Desarrollo de Aplicaciones para Dispositivos Móviles, Metodología de Pruebas de Sistemas, y Tecnologías de la

Información y de la Comunicación. Se requiere conocimiento previo en gestión de bases de datos y desarrollo orientado a objetos.

Objetivos principales:

- Desarrollar software por encargo y para productos propios.
- Integrar equipos de proyecto para el desarrollo o mantenimiento de software.
- Liderar grupos de trabajo o asumir roles especializados.
- Desempeñarse de manera autónoma en el desarrollo de sistemas de baja complejidad.

OBJETIVOS DE APRENDIZAJE

Los estudiantes aprenderán mediante un enfoque basado en proyectos, motivado e inspirado en realidades concretas. Este enfoque promueve un cambio de paradigma en el aprendizaje, combinando contenidos disciplinares para reflejar situaciones reales y pertinentes.

Cada clase comenzará con una pregunta que generará un desafío, fomentando el debate y el pensamiento crítico. Los estudiantes investigarán y profundizarán en los temas tratados.

La enseñanza se desarrollará de manera abierta, flexible y creativa, utilizando estrategias como exposición dialogada, lectura dirigida, dinámicas grupales, trabajo de investigación e indagación bibliográfica.

El curso culminará en un proyecto final, promoviendo la reflexión y la profundización en el conocimiento más allá del contenido teórico.

PROPÓSITOS DEL DOCENTE

- Reutilizar y profundizar conceptos previos de introducción a la programación.
- Utilizar herramientas para resolver problemas a través del pensamiento abstracto.
- Conocer principios del desarrollo de software y familiarizarse con la terminología específica de la industria.
- Desarrollar habilidades para plantear soluciones racionales y lógicas a problemas de programación.
- Aplicar análisis crítico y procedimientos propios de la programación.
- Reflexionar sobre la aplicación del desarrollo de software en la sociedad y la economía.
- Incentivar la creación de programas por parte de los estudiantes y enseñarles a

representar ideas mediante algoritmos.
- Utilizar listas y arreglos correctamente y comprender el funcionamiento de las computadoras en la ejecución de programas.

EXPECTATIVAS DE LOGRO

Se espera que los estudiantes:

- Construyan una base sólida para continuar desarrollándose de manera autónoma.
- Adquieran herramientas para el aprendizaje de distintos lenguajes de programación.
- Demuestren pensamiento crítico en su trabajo y aprendizaje colaborativo.
- Diseñen algoritmos propios y se comuniquen de manera efectiva mediante comentarios detallados.
- Vinculen conocimientos con el mundo laboral y orientación vocacional.
- Establezcan diferencias entre lenguajes de programación orientados a objetos y diseñen estrategias de resolución de problemas de forma autónoma.

RECURSOS

- **Humanos:** Docentes, estudiantes
- **Espacial:** Meet
- **Materiales didácticos:** Campus, Códigos prearmados, formularios, videos explicativos

---

DETALLE DE LAS CLASES SINCRÓNICAS Y ASINCRÓNICAS

Durante las clases, el contenido se alineará con el avance en el proyecto de los parciales. Se utilizarán formularios para consultas sobre temáticas específicas y foros de discusión para resolver dudas.

RECORRIDOS DETALLADOS DEL CONTENIDO

- M1 - Fundamentos de backend
    - 1. Introducción
    - 2. Tipos de aplicaciones
    - 2.1. Comparación de aplicaciones
    - 3. Desarrollo web front-end vs. desarrollo web back-end
    - 3.1. ¿Qué es Frontend?

- 3.2. ¿Qué es el Backend?
- 3.3. ¿Cuáles son las diferencias entre desarrollo Frontend y Backend?
- 3.4. Desarrolladores backend
- 4. ¿Cómo funciona internet?
- 4.1. Protocolos y funcionamiento
- 4.2. Componentes
- 4.3. Requerimiento
- 4.4. Infraestructura
- 4.5. Contenido
- 5. Navegadores web
- 5.1. Algunos navegadores web
- 6. Direcciones web
- 6.1. Ruta
- 7. El modelo cliente-servidor
- 7.1. Ventajas
- 7.2. Desarrollo de aplicaciones web basadas en servidor
- 8. Protocolo HTTP
- 8.1. Códigos y verbos HTTP
- 8.2. Codificación adicional
- 8.3. Códigos de respuesta
- 8.4. HTTP caching
- 8.5. HTTP cookies
- 8.6. Ejemplo de petición HTTP y su respuesta
- 8.7. Respuesta HTTP
- 9. Tipos de sitios
- 9.1. Sitios dinámicos
- 10. API
- 10.1. API REST
- 10.2. Lenguajes backend para desarrollar una API
- 11. ¿Qué es Node .js?
- 11.1. JavaScript en el lado del servidor
- 11.2. ¿Por qué aprender Node.js?
- 11.3. Instalación de Node.js / Visual Studio Code
- 11.4. NPM
- 11.5. El comando Node: "Hola mundo" con Node

○ M2 - Conociendo el entorno

- 1. Introducción al lenguaje de programación
- 2. Valores, tipos y operadores
- 2.1. Strings
- 2.2. Operadores unarios
- 2.3. Valores booleanos
- 2.4. Operadores lógicos
- 2.5. Conversión de tipo automática
- 2.6. Convertir a valor false
- 2.7. Variables
- 2.8. Vinculación que señala a un valor
- 2.9. Nombres vinculantes
- 2.10. Vinculaciones y alcances

- 2.11. Alcance anidado
- 2.12. Var
- 2.13. Let
- 2.14. Const
- 3. El entorno
- 4. Funciones
- 4.1. Definiendo una función
- 4.2. Notaciones de declaración
- 4.3. Funciones de flecha
- 4.4. La función console.log
- 4.5. Valores de retorno
- 4.6. Recursión
- 5. Flujo de control
- 5.1. Ejecución condicional (if-else)
- 5.2. Ciclos while y do
- 5.3. Ciclo do
- 5.4. Ciclos for
- 5.5. Rompiendo un ciclo
- 5.6. Actualizando vinculaciones de manera sucinta
- 5.7. Switch
- 5.8. Comentarios
- 6. Estructura de datos: objetos y arrays
- 6.1. Arrays
- 6.2. Iterar arrays
- 6.3. Transformando con map
- 6.4. Filtrando arrays
- 6.5. Reducción de arrays
- 6.6. Propiedades
- 6.7. Objetos
- 6.8. Notación de punto
- 6.9. Espacio de nombres secundarios
- 6.10. Notación de corchetes
- 6.11. Establecer miembros de objetos
- 6.12. ¿Qué es "this" (este)?
- 6.13. Operadores con objetos
- 6.14. Prototipos de objetos
- 7. JSON
- 7.1. Estructura de JSON
- 7.2. Arreglos como JSON
- 7.3. Aspectos a tener en cuenta
- 7.4. JSON.stringify y JSON.parse

- M3 - Modelado de estructuras
  - 1. Introducción al paradigma orientado a objetos
  - 2. Javascript orientado a objetos
  - 3. Clases
  - 3.1. Instanciar una clase
  - 3.2. La palabra clave this
  - 3.3. Clases en ficheros externos
  - 4. Propiedades de una clase

- 4.1. Propiedades públicas
- 4.2. Propiedades privadas
- 4.3. Propiedades computadas
- 5. Métodos de clase
- 5.1. Constructor de clase
- 5.2. ¿Qué es un método estático?
- 5.3. Visibilidad de métodos
- 6. Herencia de Clases
- 6.1. Extender una clase
- 6.2. El método super
- 6.3. Acceder a métodos del padre

- M4 - Tareas asíncronas y web
  - 1. Introducción
  - 2. La sincronía en Javascript
  - 3. Lenguaje no bloqueante
  - 3.1. Ejemplos de tareas asincrónicas
  - 3.2. ¿Cómo gestionar la asincronía?
  - 4. Funciones Callbacks
  - 4.1. Otras funciones
  - 4.2. Asincronía con callbacks
  - 4.3. Desventajas de los callbacks
  - 5. Promesas
  - 5.1. Promesas en Javascript
  - 5.2. Consumir una promesa
  - 5.3. Asincronía con promesas
  - 6. Promesas en grupo (promise api)
  - 6.1. API de las promesas
  - 6.2. Promise.all()
  - 6.3. Promise.allSettled()
  - 6.4. Promise.any()
  - 6.5. Promise.race()
  - 6.6. Promise.resolve() y Promise.reject()
  - 7. ASYNC/AWAIT
  - 7.1. La palabra clave async
  - 7.2. La palabra clave await
  - 7.3. Asincronía con async/await
  - 7.4. Top-level await
  - 8. Peticiones HTTP
  - 8.1. Métodos de peticiones AJAX
  - 8.2. XMLHttpRequest (XHR)
  - 8.3. FETCH: Peticiones asincrónicas
  - 8.4. Política CORS
  - 8.5. Cabeceras (Headers)
  - 8.6. Respuesta de la petición HTTP
  - 8.7. Métodos de procesamiento
  - 8.8. Ejemplo utilizando promesas
  - 8.9. Ejemplo utilizando Async/await
  - 9. URL en JAVASCRIPT
  - 9.1. Partes de una URL

- 9.2. Partes opcionales de una URL
- 10. Política CORS
- 10.1. Access-Control-Allow-Origin
- 10.2. CORS en entornos de desarrollo

- M5- Módulos en Node.js
  - 1. Fundamento de Node.JS
  - 2. Javascript del lado del servidor
  - 3. El proceso de ejecución de NodeJS
  - 3.1. Single Thread (único hilo)
  - 3.2. No bloqueante, concurrente y asincrónico
  - 4. Motor V8
  - 5. Loop de eventos
  - 6. Objeto process
  - 7. Módulos
  - 7.1. El módulo HTTP
  - 7.2. El módulo del sistema de archivos (fs)
  - 7.3. Método readFileSync síncrono
  - 7.4. Módulos propios
  - 8. Gestor de paquetes NPM
  - 8.1. El comando npm
  - 8.2. El archivo package.json
  - 8.3. Control de versiones y dependencias de desarrollo
  - 9. Otros gestores de paquetes
  - 10. Modelo de eventos
  - 10.1. APIs Asíncronas
  - 11. Eventos en NodeJS
  - 11.1. Módulo de eventos
  - 11.2. Cómo definir un evento en NodeJS
  - 12. Entornos de ejecución
  - 12.1. Variables de entorno
  - 12.2. NODE_ENV
  - 12.3. Definición de las variables de entorno en un JSON
  - 12.4. Require de las variables de entorno
  - 12.5. Definir un module para la gestión de variables de entorno
  - 12.6. Middlewares
  - 13. Arquitectura MVC
- M6- Marco de desarrollo
  - 1. Fundamentos de un framework
  - 2. Introducción a Express
  - 3. Qué es un framework de desarrollo web
  - 4. Qué es Express
  - 4.1. ¿Cómo funciona Express?
  - 4.2. Instalación y “Hola Mundo” en Express
  - 4.3. Usar Express para crear nuestro primer servidor
  - 4.4. Manejo de Rutas en Express
  - 4.5. Ejemplo de rutas en Express
  - 5. Manejo de Rutas con Routers

- 5.1. Rutas dinámicas
- 5.2. Rutas coincidentes con patrón
- 5.3. Rutas no válidas
- 6. Middlewares en Express
- 6.1. Middlewares personalizados
- 6.2. Middleware de terceros
- 6.3. Middleware sin rutas
- 6.4. Sirviendo archivos estáticos
- 7. Formato de respuestas a solicitudes
- 8. Plantillas
- 8.1. Motor de plantillas propio
- 8.2. Pug
- 9. Gestión de errores
- 10. Estructura de un proyecto con Node y Express
- 10.1. ¿Cómo está organizada la estructura de express?
- 11. MVC con Node y Express
- 11.1. Creación de una aplicación con Express Application Generator
- 11.2. Análisis del código generado: bin/www
- 11.3. Análisis del código generado: app.js
- 12. Estructura de carpetas

○ M7- Peticiones y respuestas
- 1. Fundamento
- 2. Introducción
- 3. Express Requests (El objeto request)
- 4. Express Responses (El objeto response)
- 5. Nodemon
- 6. REST
- 6.1. Obteniendo un solo recurso
- 6.2. Eliminar recursos
- 7. Postman
- 7.1. Probando requests con Postman
- 8. Ejemplo API REST

○ M8- Estructura de documentos
- 1. Fundamento de bases de datos
- 2. Introducción
- 3. MongoDB
- 3.1. Ventajas
- 4. Estructura de gestión de la información
- 4.1. Tipos de datos BSON
- 5. MongoDB Query Language (MQL)
- 5.1. Creación de colecciones
- 5.2. Inserción de documentos
- 5.3. Búsqueda de documentos
- 5.4. Actualización de documentos
- 5.5. Borrado de documentos
- 6. Operadores en MongoDB
- 6.1. Operadores de comparación
- 6.2. Operadores de elementos

- 6.3. Operadores lógicos
- 7. Operaciones de agregación
- 7.1. Tuberías y transformaciones
- 7.2. Filtrado de documentos con criterios
- 7.3. Orden de documentos
- 7.4. Selección de campos en específico
- 7.5. Agrupación de documentos
- 7.6. Métodos de agregación de propósito único
- 8. Relaciones en MongoDB
- 8.1. Operador Lookup
- 8.2. Relación 1:1
- 8.3. Relación 1:N
- 8.4. Relación N:M
- 9. Funcionamiento de MongoDB
- 10. Migración de SQL a NoSql
- 11. Operaciones CRUD
- 11.1. SQL: Create y Alter
- 11.2. NoSQL con MongoDB: CREATE y ALTER
- 11.3. Insert con SQL y NoSQL
- 11.4. Update y Delete con SQL y NoSQL
- 11.5. Select con SQL y NoSQL
- 12. ODM
- 13. Mongoose
- 13.1. Instalación
- 13.2. Esquemas
- 13.3. Construcción de documentos
- 13.4. Guardar documentos

○ M9- Creación de una solución

- 1. Fundamentos de una solución
- 2. Introducción
- 3. Implementación de una API REST completa
- 3.1. Creación del proyecto y estructura
- 3.2. Inicializando el servidor HTTP
- 3.3. Agregar Nodemon
- 3.4. Creando los modelos de nuestra API REST
- 3.5. Explicación
- 4. Implementando los controladores de nuestras rutas o endpoints
- 4.1. Obtener todas las series de tv (READ - GET)
- 4.2. Explicación
- 4.3. Obtener una serie de tv (READ - GET)
- 4.4. Explicación
- 4.5. Crear una serie de tv (CREATE - POST)
- 4.6. Explicación
- 4.7. Actualizar una serie de tv (UPDATE - PUT)
- 4.8. Explicación
- 4.9. Eliminar una serie de tv (DELETE)
- 4.10. Explicación
- 5. Conectar los controladores

- 5.1. Explicación
- 6. Probando nuestra API REST
- 6.1. POST
- 6.2. GET
- 6.3. GET por ID
- 6.4. PUT
- 6.5. Delete
- 6.6. Error 500
- 7. Buscar series
- 8. Middlewares y Hooks en Mongoose

○ M10- Autenticación, cookies y sesiones

- 1. Introducción
- 2. Autenticación y autorización
- 3. Autenticación
- 4. Autorización
- 5. Json Web Token (JWT)
- 5.1. Base 64
- 5.2. Campos
- 5.3. Token JWT en Node
- 5.4. RSA 256
- 5.5. ¿Cómo firmar un token JWT con RSA 256?
- 5.6. Expiración de tokens
- 5.7. Buenas prácticas al utilizar JWT
- 6. Ejemplo de autenticación y autorización con nodejs, express, JWT y mongoDB
- 6.1. Creación del proyecto
- 6.2. Referencias entre colecciones
- 6.3. Esquema de Mongoose para usuarios
- 6.4. Creando usuarios
- 6.5. Usuarios únicos
- 6.6. Creación de una nueva publicación
- 6.7. Combinar usuarios y posts
- 6.8. Autenticación de usuarios con token
- 6.9. Limitación de la creación de nuevos posts a los usuarios registrados
- 6.10. Bearer
- 6.11. Manejo de errores
- 7. Caché, cookies y sesiones
- 7.1. Qué son las cookies y las sesiones
- 8. Passport
- 8.1. Autenticar Aplicaciones Node.js con Passport
- 8.2. Creación del modelo de Mongoose
- 8.3. Configuración
- 8.4. Serializar y deserializar instancias de usuario
- 8.5. Uso de estrategias de Passport
- 8.6. Estrategia de acceso
- 8.7. Estrategia de registro
- 8.8. Creación de rutas
- 8.9. Creación de vistas de Jade

- 8.10. Continuando con vistas de Jade
- 8.11. Implementación de la funcionalidad de cierre de sesión
- 8.12. Proteger las rutas
- 8.13. Probar la aplicación con Passport

- M11- Testing en aplicaciones backend
  - 1. Fundamentos del testing
  - 2. Introducción
  - 3. Principios del testing
  - 3.1. Importancia de las pruebas y la automatización
  - 3.2. Ciclo de vida del software
  - 3.3. Calidad y defectos
  - 4. Test Driven Development
  - 4.1. Ciclo Red, Green, Refactor
  - 4.2. Pasos de TDD
  - 4.3. Ventajas
  - 4.4. Historias de usuario y criterios de aceptación
  - 4.5. Tipos de pruebas de software o tipos de testing
  - 4.6. Herramientas de TDD
  - 4.7. Testing unitario
  - 4.8. Otros tests
  - 5. JEST
  - 5.1. Nuestro primer test
  - 5.2. Matchers
  - 5.3. Configuraciones y desmontaje
  - 5.4. Grouping test
  - 5.5. Pruebas de funcionamiento asíncronas
  - 5.6. Mocking
  - 6. Ejemplo práctico de unit testing con Jest
  - 7. Ejemplo de TDD con Node
  - 7.1. Setup inicial del proyecto
  - 7.2. Cómo hacer la primera prueba con TDD
  - 7.3. Unit Testing aplicando TDD en Node
  - 7.4. Unit tests VS Integration tests
  - 7.5. Integration Testing aplicando TDD en Node
  - 7.6. Estructura de Node JS con Express aplicando MVC
- M12- Websockets
  - 1. Fundamentos
  - 2. Introducción
  - 3. ¿Cómo funciona un web socket?
  - 4. ¿Para qué se utiliza WebSocket?
  - 5. Ventajas
  - 6. Websockets en Node
  - 7. Socket.io
  - 8. Ejemplo con node, express y socket.io
  - 8.1. Iniciando la aplicación
  - 8.2. Recibiendo mensajes
  - 8.3. Enviando mensajes
- M13- Despliegue de una aplicación con Node.js
  - 1. Aplicaciones en Node.JS

- 1.1. Deployment o despliegue de aplicaciones
- 1.2. Stacks Tecnológicos
- 1.3. Componentes de una aplicación
- 1.4. Aplicaciones monolíticas vs microservicios
- 1.5. Servicios de alojamiento de servidores backend
- 2. Despliegue de aplicaciones backend
- 3. DevOps
- 3.1. Tipos de Deploy
- 3.2. 5 buenas prácticas para hacer deploy
- 4. MongoDB Atlas
- 4.1. Desplegando una base de datos NoSql en Mongo Atlas
- 5. Vercel
- 5.1. Despliegue de aplicaciones en Vercel

RÉGIMEN DE CURSADA Y EXÁMENES

La evaluación de la materia es de carácter formativo y procesual. Para regularizar la cursada, los y las estudiantes deberán:

- realizar **2 (dos) entregas obligatorias como parte** de un **proyecto grupal**, establecidas en el calendario de entregas al final de los recorridos **9 y 13**, respectivamente.
- participar de manera optativa en **prácticas formativas no obligatorias**, las cuales forman parte del repaso, pero no se computan en la evaluación. Estas consistirán en una colección de ejercicios teórico-prácticos, que pueden incluir preguntas de opción múltiple y/o código.

Todos los trabajos deberán ser de producción grupal, de hasta 5 estudiantes, y entregados en las fechas indicadas en el cronograma, a través del aula virtual y en el espacio correspondiente.

En caso de que la calificación de las entregas resultara insuficiente, serán devueltos y podrán ser entregados con correcciones durante las semanas de recuperación.

La cursada será aprobada si ambas entregas cuentan con una calificación mínima de **4 (cuatro)**, ya sea en la primera instancia o en la instancia de recuperación.

Condiciones de **aprobación** de la materia

PROMOCIÓN: Se obtiene con una calificación mínima de **7 (siete)** en cada una de las dos entregas del proyecto.

FINAL: Quienes obtengan una calificación menor a **7 (siete)** en alguna de las entregas deberán presentar un anexo al proyecto en el examen final.

La evaluación de los estudiantes incluirá los siguientes criterios:

- **Reflexión Crítica**: Evaluación de la capacidad para reflexionar sobre errores propios y ajenos, así como la habilidad para elaborar argumentaciones racionales.

- **Trabajo en Grupo**: Valoración del trabajo en equipos cooperativos, incluyendo la capacidad para expresar opiniones, ideas y el grado de compromiso y tolerancia en el trabajo grupal.
- **Desempeño en Trabajos Prácticos**: Evaluación del cumplimiento de las consignas establecidas en los trabajos prácticos y actividades de la materia.
- **Discusión Bibliográfica**: Análisis del material bibliográfico trabajado en clase.
- **Informes y Argumentaciones**: Redacción de informes escritos individuales sobre la tarea realizada, incluyendo la defensa de conclusiones y la exposición oral de la información seleccionada.
- **Utilización de IA Consciente:** La tecnología ha avanzado, la cátedra propone la utilización de la misma, siempre y cuando el estudiante pueda a la hora de defender el trabajo, explicarlo con apropiación del contenido.

BIBLIOGRAFÍA OBLIGATORIA

- **Material de Agencia de Aprendizaje a lo largo de la Vida - IFTS 29** https://aulasvirtuales.bue.edu.ar
- **Documentación Oficial de Node.js**: Node.js Documentation - Información actualizada sobre las API y características de Node.js.
- **Documentación Oficial de Express.js**: Express.js Documentation - Guía sobre el uso de Express en el desarrollo web con Node.js.
- **Documentación Oficial de MongoDB**: MongoDB Documentation - Información sobre MongoDB y su uso.
- **Casciaro, M. (2019)**: *Node.js Design Patterns*. Packt Publishing - Explora patrones de diseño en Node.js.
- **Wilson, J. (2021)**: *Node.js 14 The Right Way: Practical, Server-Side JavaScript That Scales*. Pragmatic Bookshelf - Guía práctica sobre el uso efectivo de Node.js.

BIBLIOGRAFÍA COMPLEMENTARÍA

- **Anderson, J. D. (2020)**: *Node.js for JavaScript Developers*. Apress - Ayuda a los desarrolladores de JavaScript a adaptarse a Node.js.
- **Mardan, A. (2020)**: *Pro Express.js: Master Express.js: The Node.js Framework For Your Web Development*. Apress - Aspectos avanzados de Express.js.
- **Giamas, A. (2019)**: *Mastering MongoDB 4.x: A Practical Guide to MongoDB 4.x, Its Data Model, and Its Capabilities*. Packt Publishing - Características avanzadas de MongoDB.
- **Mead, A. (2020)**: *Learning Node.js Development*. Packt Publishing - Introducción completa a Node.js.
- **Hahn, E. (2022)**: *Express in Action*. Manning Publications - Guía detallada sobre el uso de Express.
- **Bradshaw, S., & Chodorow, K. (2023)**: *MongoDB: The Definitive Guide (4ª ed.)*. O'Reilly Media - Referencia exhaustiva sobre MongoDB.