DIGITALHOUSE

Certified Tech Developer

The Ultimate Degree

Especialización en **Back End I**

Especialización en Back End I

Fundamentación

Desde la llegada de arquitecturas ideadas en la nube y frente a la necesidad de contar con aplicaciones más robustas, escalables, con alta disponibilidad manteniendo un bajo desperdicio de procesamiento, nacen las arquitecturas de software basadas en microservicios.

Estos son, entonces, un enfoque arquitectónico y organizativo para el desarrollo de software donde el mismo está compuesto por pequeños servicios independientes que se comunican a través de protocolos bien definidos. Los propietarios de estos servicios son equipos pequeños independientes que promueven este modelo arquitectónico, una solución tecnológica a la propuesta brindada por las metodologías ágiles para el proceso de desarrollo de software.

Es por ello que, frente a los desafíos que presentan las compañías —por las necesidades de implementar nuevos negocios y por consiguiente la necesidad de evolucionar rápidamente el software que soporta los mismos a través de pequeños incrementos—, es imprescindible conocer en detalle los conceptos y componentes de software que rigen este modelo arquitectónico.

Objetivos de aprendizaje

- Adquirir las bases y desarrollar la capacidad de programar aplicaciones de software desde la perspectiva de una arquitectura basada en microservicios.
- Comprender y analizar los diferentes desafíos que enfrentan los actuales equipos de trabajo al momento de desarrollar software en la nube robusto, escalable y con alta disponibilidad.
- Los conceptos se aplicarán en el lenguaje de programación Java, tomando como referencia el framework de microservicios diseñado por Netflix, Spring Cloud, uno de los más utilizados para desarrollos en empresas IT hoy en día.

Metodología de enseñanza-aprendizaje

Desde Digital House, proponemos un modelo educativo que incluye entornos de aprendizaje sincrónicos y asincrónicos con un enfoque que vincula la teoría y la práctica, mediante un aprendizaje activo y colaborativo.

Nuestra propuesta incluye clases en vivo con tu grupo de estudiantes y docentes, a los que podrás sumarte desde donde estés. Además, contamos con un campus virtual a medida, en el cual encontrarás las clases virtuales, con actividades, videos, presentaciones y recursos interactivos, para realizar a tu ritmo antes de cada clase en vivo.

A lo largo de tu experiencia de aprendizaje en Digital House lograrás desarrollar habilidades técnicas y blandas, como ser el trabajo en equipo, la creatividad, la responsabilidad, el compromiso, la comunicación efectiva y la autonomía.

En Digital House utilizamos la metodología de “aula invertida”. ¿Qué quiere decir? Cada semana te vamos a pedir que te prepares para la que sigue, leyendo textos, viendo videos, realizando actividades, entre otros recursos. De esta forma, cuando llegues al encuentro en vivo, estarás en condiciones de abordar el tema y aprovechar esa instancia al máximo.

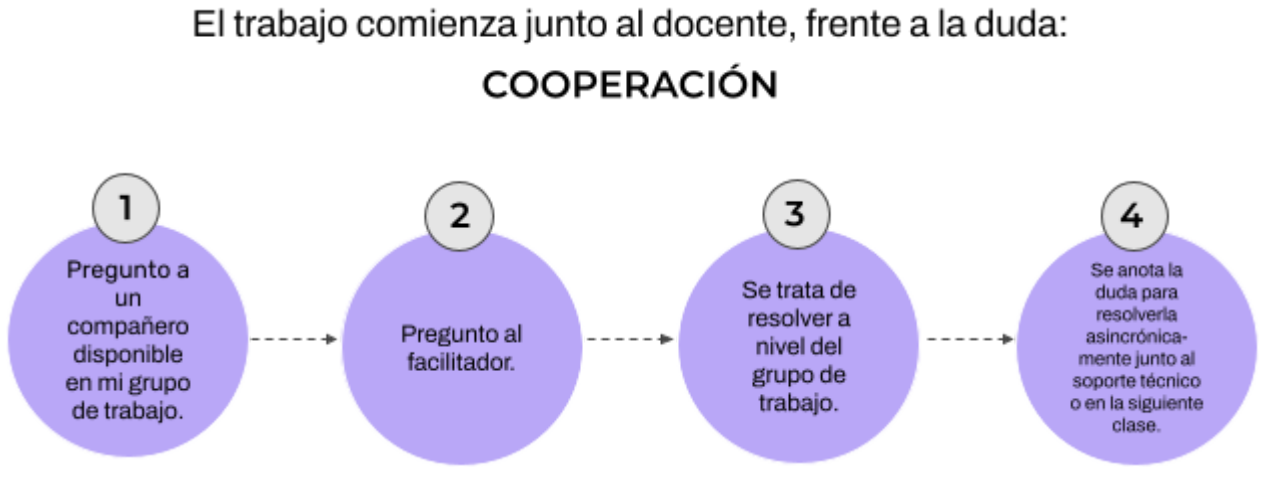
Empleamos actividades y estrategias basadas en los métodos participativos y activos para ponerte en movimiento, ya que uno solo sabe lo que hace por sí mismo. Por ese motivo, organizamos las clases para que trabajes en ellas de verdad y puedas poner en práctica las distintas herramientas, lenguajes y competencias que hacen a la formación de un programador. En otras palabras, concebimos la clase como un espacio de trabajo.

Una de las cuestiones centrales de nuestra metodología de enseñanza es el aprendizaje en la práctica. Por ese motivo, a lo largo de la cursada estarán muy presentes las ejercitaciones, es decir, la práctica de actividades de diversos tipos y niveles de complejidad que te permitirán afianzar el aprendizaje y comprobar que lo hayas asimilado correctamente. De esta forma, se logra la incorporación de los contenidos de una forma más significativa y profunda, la asimilación de los conocimientos se vuelve más eficaz y duradera. Relacionar lo aprendido con la realidad de los desarrolladores web, fomentar la autonomía y el autoconocimiento, mejorar el análisis, la relación y la comprensión de conceptos ayuda a ejercitar múltiples competencias.

El aprendizaje entre pares es uno de los elementos centrales de nuestra metodología, por eso, en cada clase te propondremos que trabajes en mesas de trabajo junto a tus compañeros —a lo largo de la cursada, iremos variando la composición de los grupos para potenciar la cooperación—. Lo que se propone es un cambio de mirada sobre el curso en cuestión, ya no se contempla al estudiante transitando su camino académico de manera individual, sino como parte de un equipo que resulta de la suma de las potencialidades de cada uno. La distribución en grupos de trabajo fomenta la diversidad y el aprovechamiento del potencial de cada integrante para mejorar el rendimiento del equipo.

La explicación recíproca como eje del trabajo cotidiano no solo facilita el aprendizaje entre compañeros, sino que sobre todo potencia la consolidación de conocimientos por parte de quien explica. Se promueve la responsabilidad, la autonomía, la proactividad, todo en el marco de la cooperación. Lo que lleva a resignificar la experiencia de aprendizaje y a que la misma esté vinculada con emociones positivas.

El trabajo cooperativo permite entablar relaciones responsables y duraderas, aumenta la motivación y el compromiso, además de promover un buen desarrollo cognitivo y social. La cooperación surge frente a la duda. Si un estudiante tiene una pregunta, le consulta a algún miembro de su grupo asignado que esté disponible. Si la duda continúa, se convoca al facilitador. En caso de que no lo resuelvan, el facilitador pedirá a todos que se detengan para cooperar como equipo en la resolución del conflicto que ha despertado la pregunta. Así debatirán todos los integrantes de la mesa buscando la solución. Si aun así no pueden resolverlo, anotarán la duda que será abordada asincrónicamente por el soporte técnico o de forma sincrónica en la siguiente clase por parte del profesor.



Todos los días, finalizada la jornada, los estudiantes reconocerán a uno de los integrantes del grupo con quienes compartieron ese día. El criterio para ese reconocimiento es la cooperación.

Cada grupo tendrá un facilitador que será elegido a partir de los reconocimientos y desarrollarán un sistema de rotación donde cualquiera pueda pasar por dicho rol. El facilitador no es una figura estática, sino que cumple un rol dinámico y versátil: se trata de un estudiante que moviliza el alcance de los objetivos comunes del equipo, poniendo en juego la cooperación. Es aquel que comparte con la mesa su potencial en favor del resto del equipo y que, por lo tanto, promueve la cooperación.

Información de la materia

- Modalidad 100% a distancia.
- Cantidad de semanas totales: 9.
- Cantidad de clases virtuales en Playground: 27.
- Cantidad de clases en vivo totales: 27.

Requisitos y correlatividades

Para cursar este primer tramo de la Especialización en Back End es requisito haber aprobado el Proyecto Integrador 1. A su vez, la aprobación de esta materia es requisito para cursar el Proyecto Integrador 2.

Modalidad de trabajo

Nuestra propuesta educativa está diseñada especialmente para la modalidad 100% a distancia, mediante un aprendizaje activo y colaborativo bajo nuestro lema “aprender haciendo”. Es por esto que los entornos de aprendizaje son tanto sincrónicos como asincrónicos, con un enfoque que vincula teoría y práctica, por lo que ambas están presentes en todo momento.

Contamos con un campus virtual propio en el cual vamos a encontrar actividades, videos, presentaciones y recursos interactivos con instancias de trabajo individual y en equipo para profundizar en cada uno de los conceptos.

Además, realizaremos encuentros online y en vivo con el grupo de estudiantes y docentes, a los que podremos sumarnos desde donde estemos a través de una plataforma de videoconferencias con nuestra cámara y micrófono para generar una experiencia cercana.

Metodología de evaluación

La evaluación formativa es un proceso continuo que genera información sobre la formación de nuestros estudiantes y de nosotros como educadores. Esto genera conocimiento de carácter retroalimentador, es decir, tiene una función de conocimiento, ya que nos permite conocer acerca de los procesos de enseñanza y aprendizaje. También tiene una función de mejora continua porque nos permite saber en qué parte del proceso nos encontramos, validar si continuamos por el camino planificado o necesitamos tomar nuevas decisiones para cumplir los objetivos propuestos.

Por último, la evaluación desempeña un papel importante en términos de promover el desarrollo de competencias muy valiosas. Nuestro objetivo es diferenciarnos de la evaluación tradicional, que muchas veces resulta un momento difícil, aburrido y tenso. Para ello, vamos a utilizar la gamificación, la cual es una técnica donde se aplican elementos de juego para que el contenido sea más atractivo, los participantes se sientan motivados e inmersos en el proceso, utilicen los contenidos de aprendizaje como retos que realmente quieren superar y aprendan del error.

A su vez, para registrar dicha formación, se utiliza un conjunto de instrumentos, para los cuales es fundamental utilizar la mayor variedad posible, y técnicas de análisis.

Criterios de aprobación

- Realizar las actividades de Playground (80% de completitud).
- Asistencia a los encuentros sincrónicos (90% de asistencia).
- Obtener un puntaje de 7 o más en la evaluación final.

- Obtener un puntaje de 7 o más en la nota final de la materia.

Contenidos

Módulo 1: Introducción a microservicios

En este módulo abordaremos los conceptos y características que definen a una arquitectura de software basada en microservicios. Su diferencia con las arquitecturas más tradicionales y las ventajas frente a estas en un contexto de alta demanda, escalabilidad y cambios en el software.

Clase 1: ¿Qué es un microservicio?

- Bienvenida de la materia
- ¿Qué es la arquitectura de software?
- ¿Qué es un microservicio?
- Monolito vs. microservicios
- Ventajas de una arquitectura orientada a microservicios

Clase 2: Introducción a microservicios

- Desafíos de los microservicios
- Patrones de diseño
    - Service registry
    - Service discovery
    - Edge server
    - Central configuration
    - Log aggregation
    - Distributed tracing

- Circuit breaker
- Reactive microservices
- Centralized monitoring and alarms

Clase 3: Cierre de la semana

Módulo 2: Framework Spring Cloud

Estudiaremos en detalle los diferentes componentes del framework de Spring Cloud, utilizado por Netflix para la construcción de una arquitectura orientada a servicios, desde sus características, configuraciones, desarrollos e implementación de los mismos.

Clase 4: Eureka server

- Arquitectura Eureka server
- Registro y descubrimiento de microservicios: ¿qué problemas viene a solucionar?
- Configuración de Eureka server
- Spring boot actuator

Clase 5: Configuración en sistemas distribuidos

- Introducción a spring cloud config server
- Configuración de microservicios
- Conexión de spring cloud config server a Git
- Debugging con spring cloud config server

Clase 6: Cierre de la semana

Clase 7: Invocaciones REST declarativas y balanceo de carga

- Introducción a Feign
- Feign REST client para invocación de servicios
- Balanceo de carga con Eureka, Feign y Spring Cloud Loadbalancer

Clase 8: API Gateway

- Configuración API Gateway
- Configuración de Zuul API Gateway
- Discovery Locator
- Ruteo
- Logging Filter

Clase 9: Cierre de la semana

Clase 10: Taller de coding

Clase 11: Práctica preevaluación

Clase 12: Evaluación

Clase 13: API Gateway - Seguridad

- Seguridad con OAuth

Clase 14: Patrón Circuit Breaker

- ¿Qué es la tolerancia a fallas?
- Resilience4j
- Reintentos (Retry & Fallback)

Clase 15: Cierre de la semana

Clase 16: Traceo distribuido

- ¿Qué es traceo distribuido?
- Implementación de Spring Cloud Sleuth
- Zipkin
- Zipkin dashboard

Módulo 3: Mensajería asincrónica

Nos introduciremos en la mensajería asincrónica, muy utilizada en aplicaciones empresariales de alto rendimiento y seguridad. Aprenderemos cómo implementar una cola de mensajes en aplicaciones que necesiten garantizar que no se pierda ninguno de ellos.

Clase 17: Comunicación asincrónica

- Comunicación sincrónica vs. asincrónica
- RabbitMQ
- Spring Cloud Stream
- Implementación de Zipkin con RabbitMQ

Clase 18: Cierre de la semana

Clase 19: Log Aggregation

- ¿Qué es? ¿Qué problema soluciona? ¿Cómo?
- Implementación con ELK

Módulo 4: Implementación sobre Docker

En este último módulo nos dedicaremos a abordar la problemática de la infraestructura más adecuada donde implementar los componentes de Spring Cloud.

Clase 20: Docker y microservicios - Parte I

- Deployando microservicios con Docker
- Administrando los microservicios con docker compose
- Bases de datos con Docker Compose
- Eureka server con Docker Compose

Clase 21: Cierre de la semana

Clase 22: Docker y microservicios - Parte II

- Zipkin y RabbitMQ sobre Docker
- Spring Cloud API Gateway sobre Docker
- Config Server con Docker

Clase 23: Práctica preevaluación

Clase 24: Evaluación

Clase 25: Clase especial I

Clase 26: Clase especial II

Clase 27: Cierre

Material de referencia

- Larsson, M. (2021). *Microservices with Spring Boot and Spring Cloud - Second Edition: Build resilient and scalable microservices using Spring Cloud, Istio, and Kubernetes*. Packt Publishing. (http://library.lol/main/2C1CEAC5CF6E2070C9823C97265A3AFC)
- Mińkowski, P. (2018). *Mastering Spring Cloud*. Packt Publishing. (http://library.lol/main/062FC41CE4504CE94E1DD4CDA1AAA767)
- Sharma, S. (2017). *Mastering microservices with Java 9 : build domain-driven microservice-based applications with Spring, Spring Cloud, and Angular*. Packt Publishing. (http://library.lol/main/E5F71FCCE5BDD3BF72CE62084DADA426)
- Karanam, R. R. (2017). *Mastering Spring 5.0: Master reactive programming, microservices, Cloud Native applications, and more*. Packt Publishing. (http://library.lol/main/C54624B796FBC87FE2F5D8770CF1BE8E)
- Long, J. y Bastani, K. (2017). *Cloud Native Java Designing Resilient Systems with Spring Boot, Spring Cloud, and Cloud Foundry*. O' Reilly. (http://library.lol/main/EEC3E15D8687C0784E938F2B20B5D1BD)