



Universidad Tecnológica Nacional
Facultad Regional Buenos Aires

PROGRAMA ANALÍTICO DE ASIGNATURA

DEPARTAMENTO: Ingeniería en Sistemas de Información

CARRERA: Ingeniería en Sistemas de Información

NOMBRE DE LA ACTIVIDAD CURRICULAR: Arquitecturas de Software Concurrente

Año Académico: 2023

Área: Desarrollo de Software

Bloque: Electivas

Nivel: 5º

Tipo: Electiva

Modalidad: Cuatrimestral

Cargas horarias totales:

<i>Horas reloj</i>	<i>Horas cátedra</i>	<i>Horas cátedra semanales</i>
72	96	6

COMPOSICIÓN DEL EQUIPO DE CÁTEDRA:

Profesor Adjunto: Ing. Ernesto Bossi

FUNDAMENTACIÓN

El Ingeniero en Sistemas de la Información debe estar capacitado para afrontar el desarrollo integral de proyectos de alta disponibilidad, especialmente, dentro del ámbito de la construcción de software. La materia permitirá al estudiante poder desempeñarse profesionalmente en dicha rama, desarrollar herramientas para diseñar e implementar arquitecturas de software y elegir tecnologías arquitectónicas que puedan ser escalables.

La presente materia electiva expone conceptos teóricos esenciales para la construcción de software distribuido que se complementan con los modelos tradicionales que el estudiante entra en contacto durante la carrera. A su vez, se presenta siempre actualizada con respecto a las tecnologías de concurrencia moderna disponibles en el mercado.



Universidad Tecnológica Nacional
Facultad Regional Buenos Aires

OBJETIVOS

- Distinguir los conceptos de paralelismo, concurrencia, asincronismo, evento y operación (no) bloqueante para el diseño e implementación de arquitecturas de software.
- Reconocer los modelos de concurrencia no tradicionales, como el paradigma de actores, memoria transaccional, modelo orientado a eventos, *Promises*, *Corrutinas*, *Guilds* y estructuras libres de conflicto como CRDTs.
- Identificar el impacto en la escalabilidad y mantenibilidad de cada uno de los conceptos abordados.
- Desarrollar la capacidad de toma de decisiones de arquitectura relacionadas con las tecnologías y estilos presentados en la materia.

CONTENIDOS ANALÍTICOS

Unidad Temática 1: Contexto.

Estado del arte en la construcción de software. Avances y cambios en la tecnología en la última década. Tendencias en hardware. Desafíos presentes y futuros del desarrollador y el arquitecto de software.

Unidad Temática 2: Nociones fundamentales.

Concurrencia. Paralelismo. Sincronismo y asincronismo. Bloqueante y no bloqueante. Multiprocesamiento. Multiplexación. Distribución. Escalabilidad.

Unidad Temática 3: Modelos tradicionales y modelos nuevos.

Estado compartido. Flujo secuencial imperativo. Estudio comparativo entre los modelos tradicionales basados en locking Fork/Join, frente a los nuevos modelos de concurrencia. Impacto en la arquitectura. Programación reactiva y orientada a Eventos. *Corrutinas*.

Unidad Temática 4: Internals.

Hilos verdes, Fibras, Multiplexación. Tipos de IO. Nociones sobre la implementación de Actores, Fuentes de Eventos y *Promises* utilizando estas primitivas.

Unidad Temática 5: Eventos Explícitos.

Fuentes de eventos. Observers. Reactors. Continuaciones. Continuation Passing Style (CPS). Callback Hell.

Unidad Temática 6: Promises y Streams



Universidad Tecnológica Nacional
Facultad Regional Buenos Aires

Promises/Futures.Streams. Relación con Maybe y List. Motivación. Interfaz primitiva. Relación de la interfaz de una promesa con la interfaz monádica. Diagramas de canicas (Marble Diagrams).

Unidad Temática 7: Memoria Transaccional

Repaso del paradigma: ausencia de efecto, inmutabilidad, transparencia referencial, orden superior, valores vs referencias. STM: noción introductoria. Comparación entre el modelo de lock pesimista vs optimista. Primitivas de STM. Consideraciones sobre su implementación. Comparación con otros mecanismos del lenguaje como par/seq.

Unidad Temática 8: Estructuras libres de conflictos

Repaso de estado compartido y problemas asociados. Introducción de conflictos con estado compartido y problemática en la arquitectura de un sistema. Introducción a estructuras concurrentes libres de posibles conflictos. Resolución de conflictos con CRDTs. Consideraciones sobre su implementación.

Unidad Temática 9: Paradigma de Actores

Conceptos fundamentales: mensajes, actores, referencias, mailbox, ordenamiento de mensajes, procesos, links. Testing. Manejo de errores y comunicación: Señales, errores, trampas, monitores, supervisores, jerarquías de supervisión, estrategias de supervisión. Patrones: cliente-servidor, servidor de eventos, máquina de estados (FSM).

Unidad Temática 10: Introducción a la Distribución y Clustering

Conceptos fundamentales de la distribución y clustering. Introducción a distribución de sistemas de actores. Supervisión distribuida en el modelo de actores con Akka y Elixir. Distribución con Distributed Haskell. Impacto de la distribución en la arquitectura de un sistema. Distribución con malla de servicios con Istio y Kubernetes.

IBLIOGRAFÍA OBLIGATORIA

- Agha, G.(1973). A Model Of Concurrent Computation in Distributed Systems. Ed. MIT Press.
- Butcher, P. (2014). Seven Concurrency Models in Seven Weeks. Ed. O'Reilly.
- Herbert, F. (2013). Learn You Some Erlang For Great Good!. Ed. No Starch Press.
- Lipovaca, M. (2011). Learn You a Haskell for Great Good!. Ed. No Starch Press.
- Odersky, M (2010). Deprecating The Observer Pattern. Ed. EPFL.
- Prokopec, A. (2017). Learning Concurrent Programming in Scala. Ed. Packt Publishing.
- Raynal, M. (2013). Concurrent Programming: Algorithms, Principles, and Foundations. Ed. Springer.



Universidad Tecnológica Nacional
Facultad Regional Buenos Aires

PÁGINAS WEB DE INTERÉS

- Reactive Manifesto: <http://www.reactivemanifesto.org/>
- Documentación de Akka, Sección Definiciones:
<http://doc.akka.io/docs/akka/2.3.5/AkkaScala.pdf>
- Documentación de Node.js: <http://nodejs.org/documentation/>
- Distributed Haskell: <https://haskell-distributed.github.io/>
- STM en Haskell:
<https://hackage.haskell.org/package/stm-2.2.0.1/docs/Control-Concurrent-STM.html>
- Documentación de Elixir: <http://elixir-lang.org/docs.html>
- Documentación de Promises (Scala):
<http://docs.scala-lang.org/sips/completed/futures-promises.html>
- Documentación de Promises (JavaScript): <http://promisesaplus.com/>

CORRELATIVAS

Para cursar y rendir

- Cursadas:
 - Redes de Datos
 - Administración de Sistemas de Información
 - Simulación
 - Ingeniería y Calidad de Software
- Aprobadas:
 - Diseño de Sistemas de Información
 - Sistemas Operativos
 - Bases de Datos