Universidad Tecnológica Nacional
Facultad Regional Buenos Aires

ANEXO I

RESOLUCIÓN Nº...../21

ASIGNATURA:	ARQUITECTURA DE SOFTWARE CONCURRENTE	**CÓDIGO:**	
DEPARTAMENTO:	INGENIERÍA EN SISTEMAS DE INFORMACIÓN	**CLASE:**	Cuatrimestral
ÁREA:	INGENIERÍA EN SISTEMAS DE INFORMACIÓN	**HORAS SEM.:**	6 hs.
BLOQUE:	ELECTIVAS	**HORAS / AÑO:**	Reloj 72hs./ Catedra 96hs

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.

Objetivos:

Distinguir los conceptos de paralelismo, concurrencia, asincronismo, y 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.

Programa analítico:

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
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.

Distribución de carga horaria entre actividades teóricas y prácticas:

Tipo de actividad	Carga horaria total en hs. reloj	Carga horaria total en hs. cátedra
Teórica	49.5	66
Formación Práctica	22.5	30
Formación experimental	0	0
Resolución de problemas	0	0
Proyectos de diseño	0	0
Práctica supervisada	0	0
Total	72	96

Articulación Horizontal y vertical con otras materias

La asignatura de Arquitecturas de Software Concurrente se articula en forma vertical con tres (3) asignaturas que la preceden en el plan de estudios: Ingeniería en Software, Diseño de Sistemas y Sistemas Operativos. Cada estudiante deberá tener cursada y regularizada las dos primeras materias mencionadas, y en el caso de Sistemas Operativos, el estudiante deberá tenerla aprobada. El estudio de los nuevos modelos tradicionales de concurrencia y de los componentes y desarrollo de un proyecto de Software, son conocimientos previos que se requieren para poder abordar e incorporar eficientemente el contenido impartido en Arquitecturas de Software Concurrente.

Además, los conocimientos adquiridos en Arquitecturas de Software Concurrentes serán de gran utilidad para extender los alcances de estas materias, ya que se abordan nuevas técnicas y conocimientos que se suman a los contenidos incorporados en estas materias previas. Adicionalmente, la presente asignatura será de utilidad para afrontar los nuevos desafíos laborales que se presentan diariamente en estos tiempos.

En cuanto a la articulación horizontal, Arquitecturas de Software Concurrentes será de utilidad para poder afrontar el desarrollo del Proyecto Final (en caso de querer soportar alta disponibilidad en el proyecto a desarrollar). Al mismo tiempo, la asignatura brinda conocimientos complementarios que tienen buena sinergia con materias tales como Implementación de Bases de Datos NoSQL, fomentando la interdisciplinariedad.

Cronograma estimado de clases:

Unidad temática	Duración en horas cátedra
1	3
2	3
3	6
4	6
5	12
6	12
7	12
8	12
9	18
10	12

Bibliografía:

BIBLIOGRAFÍA

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

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:

Cursadas: Redes de Información
Administración de Recursos
Simulación
Ingeniería de Software

Aprobadas: Diseño de Sistemas
Sistemas Operativos
Gestión de Datos

PARA RENDIR:

Aprobadas: Redes de Información
Administración de Recursos
Simulación
Ingeniera en Software