UTN.BA
DEPARTAMENTO
INGENIERIA EN SISTEMAS DE INFORMACION

ASIGNATURA:	IMPLEMETACION DE ARQUITECTURAS DE SOFTWARE CONCURRENTE
DEPARTAMENTO:	ING. EN SIST. DE INFORMACION
AREA:	ELECTIVA
BLOQUE	COMPLEMENTARIAS

MODALIDAD:	Cuatrimestral
HORAS SEM.:	6 horas
HORAS/AÑO:	96 horas
HORAS RELOJ	72
NIVEL:	5º
AÑO DE DICTADO:	2018

Objetivos

- Comprender problemáticas de los modelos tradicionales secuenciales, imperativos, bloqueantes, y aún de las técnicas imperativas para manejo de concurrencia, tales como semáforos y fork/join
- Entender los conceptos de paralelismo, concurrencia, asincronismo, y evento y operación (no)bloqueante
- Comprender patrones de diseño provenientes del paradigma funcional como functores, functores aplicativos y mónadas
- Aplicar técnicas funcionales a nivel arquitectónico
- Entender y aplicar las ventajas y desventajas de CPS (continuationpassingstyle)
- Utilizar *Promises*y *Streams*en el manejo de concurrencia, tanto utilizando sus interfaces primitivas como sus interfaces monádicas, para la comunicación entre los componentes de una arquitectura software
- Construir arquitecturas bajo el paradigma de actores
- Entender el concepto de tolerancia a fallos, y pueda construir arquitecturas con esto en mente
- Comprender el impacto en la escalabilidad y mantenibilidad de los conceptos dados
- Conocer las técnicas a bajo nivel que posibilitan la implementación de los conceptos anteriores, tales como procesos, hilos, green-threads, event-loops.
- Conocer y emplear distintas tecnologías modernas que implementan los conceptos anteriores
- Adquirir la capacidad de tomar decisiones de arquitectura relacionadas con las tecnologías y estilos presentados en la materia

Contenidos Mínimos (Programa Sintético)

- Nociones y cualidades fundamentales para la construcción de arquitecturas de software modernas como concurrencia, paralelismo, sincronización, distribución, escalabilidad y tolerancia a fallos
- Patrones de diseño funcionales
- Estrategias y tecnologías modernas para estructurar la comunicación entre los componentes de una arquitectura de software
- Nociones sobre los *internals* de las técnicas vistas

Contenidos Analíticos

UNIDAD 1: Contexto
Estado del arte en la construcción de software. Avance y cambios en la tecnología en la última década. Tendencias en hardware. Desafíos presentes y futuros del desarrollador y el arquitecto de software

UNIDAD 2: Nociones fundamentales
Concurrencia. Paralelismo. Sincronismo y asincronismo. Bloqueante y no bloqueante. Multiprocesamiento. Multiplexación. Distribución. Escalabilidad

UNIDAD 3: Modelos tradicionales y modelos nuevos.
Estado compartido. Flujo secuencial imperativo. Primitivas tradicionales basadas en locking. Fork/Join. Impacto en la arquitectura. Programación reactiva. Eventos. Estilos de programación reactiva. Programación reactiva funcional.

UNIDAD 4: Estructuras de control y de datos del paradigma funcional
Repaso del paradigma: ausencia de efecto, inmutablidad, transparencia referencial, orden superior, valores vs referencias. Typeclasses. Polimorfismo. Monoides. Estructuras de datos/de control List, Maybe, IO. List vs Stream.

UNIDAD 5: Patrones del paradigma funcional
Functor, functor aplicativo, mónada, mónada con falla. Semántica, operaciones fundamentales, operaciones derivadas. Aplicación de los patrones. Syntaxsugar para mónadas. Relación entre estos y otros patrones. Implementación de nuevas estructuras de control.

UNIDAD 6: Eventos explícitos.
Fuentes de eventos. Observers.Reactors.Continuaciones.Continuation Passing Style (CPS). Callbackhell.

UNIDAD 7: Promises y Streams
Promieses/Futures.Streams.Relación con Maybe y List. Motivación. Interfaz primitiva. Interfaz monádica. Diagramas de canicas (MarbleDiagrams).

UNIDAD 8: Memoria Transaccional
STM: noción introductoria. Primitivas de STM. Consideraciones sobre su implementación.

UNIDAD 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). Introducción a distribución de sistemas de actores: distribución, teorema CAP, clústers.

Unidad 10: *Internals*
Hilos de sistema operativo e hilos verdes, Fibras, Procesos (SO), Multiplexación, Poll, Select. Tipos de IO. Nociones sobre la implementación de Actores, Fuentes de Eventos y Promises utilizando estas primitivas.

Bibliografía Obligatoria

- Seven Concurrency Models in Seven Weeks, Paul Butcher, 2014
- Concurrent Programming For Scalable Web Architectures, Capítulo 5, Benjamin Erb, 2012

Bibliografía Complementaria

- Deprecating The Observer Pattern, Ingo Mair, TiarRompf, Martin Odersky, 2010
- A Model Of Concurrent Computation in Distributed Systems, Gul Agha, 1973
- Typeclassopedia: http://www.haskell.org/haskellwiki/
- Learn You Some Erlang For Great Good, Fred Hebert, 2013.
- 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/
- Documentación de Clojure STM: http://nodejs.org/documentation/
- 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:

- Administración de Recursos
- Redes de Información
- Simulación
- Ingeniería de Software

Aprobadas:

- Diseño de Sistemas
- Sistemas Operativos
- Gestión de Datos

Para rendir:

Aprobadas:

- Administración de Recursos
- Redes de Información
- Simulación
- Ingeniería de Software