Carrera: Ingeniería en Sistemas de Información

Asignatura: Desarrollo de Software

Planificación a partir del Ciclo Lectivo 2025

1. Datos administrativos de la asignatura

Nivel en la carrera	3	Duración	Cuatrimestral
Plan	2023		
Bloque curricular:	Tecnologías Aplicadas		
Carga horaria presencial semanal (hs. cátedra):	8	Carga Horaria total (hs. reloj):	96
Carga horaria no presencial semanal (hs. reloj) (si correspondiese)	0	% horas no presenciales (hs. reloj) (si correspondiese)	0

2. Presentación, Fundamentación

La asignatura fue diseñada para que los estudiantes elaboren un desarrollo web integral contemplando tanto el aprendizaje de los lenguajes de programación involucrados como las herramientas satélites necesarias para su construcción.
Relación de la asignatura con el perfil de egreso: El egresado debe saber y saber hacer. Brinda habilidades técnicas y blandas indispensables en la elaboración de un desarrollo de software. Toma contacto con una solución de software web de manera integral y segura, típica en un contexto de transformación digital.
Relación de la asignatura con los alcances del título: Brinda las herramientas necesarias para construir piezas de software desde el abordaje de distintos componentes, roles y perspectivas que puede abordar un profesional de sistemas en el contexto integral de diseño de un proyecto web. Provee los instrumentos necesarios para capacitar integralmente al estudiante en el saber y saber hacer para la construcción de la tesis contemplando desde los requerimientos hasta la solución completa funcional basada en arquitecturas de referencia.

La materia aporta a las competencias CE1.3, CE2.1, CE4.1 y CE5.1:
Con respecto a CE1.3, la materia brinda soporte para diseñar y desarrollar software considerando arquitecturas de referencia atentos a los cambios tecnológicos.

Con respecto a CE2.1, la asignatura aporta mecanismos de seguridad aplicables según criterios de mejores prácticas y se considera el ciclo de desarrollo seguro al abordar tanto la construcción del front-end como el back-end.
Con respecto a CE4.1, la materia brinda herramientas para asegurar la seguridad y calidad desarrollo del back-end y front-end.
Con respecto a CE5.1, la asignatura aborda múltiples aspectos relativos a la comunicación entre distintas piezas de software.

3. Relación de la asignatura con las competencias de egreso de la carrera

En la tabla siguiente se establece la relación de la asignatura con las competencias de egreso: Específicas, Genéricas Tecnológicas y Genéricas Sociales, Políticas y Actitudinales de la carrera.
Se incluyen las competencias de egreso a las que tributa, aportes reales y significativos de la asignatura, y en qué nivel (no aporta, bajo, medio, alto).

Competencias	Nivel
Competencias genéricas tecnológicas (CG):	
CG.1. Identificación, formulación y resolución de problemas de ingeniería en sistemas de información/informática.	Bajo
CG.2. Concepción, diseño y desarrollo de proyectos de Ingeniería en Sistemas de Información/Informática	Medio
CG.3. Gestión, planificación, ejecución y control de proyectos de Ingeniería en Sistemas de Información/Informática.	No aporta
CG.4. Utilización de técnicas y herramientas de aplicación de Ingeniería en Sistemas de Información/Informática.	Bajo
CG.5. Generación de desarrollos tecnológicos y/o innovaciones tecnológicas.	No aporta
Competencias genéricas sociales, políticas y actitudinales (CG)	
CG.6. Fundamentos para el desempeño en equipos de trabajo.	Bajo
CG.7. Fundamentos para una comunicación efectiva.	Bajo
CG.8. Fundamentos para una actuación profesional ética y responsable.	No aporta
CG.9. Fundamentos para evaluar y actuar en relación con el impacto social de su actividad profesional en el contexto global y local.	No aporta
CG.10. Aprender en forma continua y autónoma.	Medio
CG.11. Fundamentos para el desarrollo de una actitud profesional emprendedora	No aporta
Competencias Específicas de la carrera	

CE1.1. Especificar, proyectar y desarrollar sistemas de información para concebir soluciones tecnológicas que permitan resolver situaciones en las organizaciones mediante el empleo de metodologías de sistemas y tecnologías asociadas a los sistemas de información.	No aporta
CE1.2. Especificar, proyectar y desarrollar sistemas de comunicación de datos, evaluando posibles soluciones tecnológicas disponibles para dar soporte a los sistemas de información en lo referido al procesamiento y comunicación de datos.	No aporta
CE1.3. Especificar, proyectar y desarrollar software para la elaboración de soluciones informáticas con el propósito de resolver problemas estratégicos y operativos, así como de servicios y de negocios, en el marco de una actividad económica que sea social y ambientalmente sustentable.	Medio
CE2.1. Proyectar y dirigir lo referido a seguridad informática para seleccionar y aplicar técnicas, herramientas, métodos y normas, garantizando la seguridad y privacidad de la información procesada y generada por los sistemas de información.	Medio
CE.3.1. Establecer métricas y normas de calidad de software para medir, evaluar, controlar y monitorear el rendimiento, impulsando mejoras de acuerdo a técnicas y normas vigentes definidas por los organismos de estandarización.	No aporta
CE.4.1. Certificar el funcionamiento, condición de uso o estado de sistemas de información, sistemas de comunicación de datos, software, seguridad informática y calidad de software para asegurar la generación de los resultados deseados en función de restricciones de tiempo y recursos establecidos.	Bajo
CE.5.1. Dirigir y controlar la implementación, operación y mantenimiento de sistemas de información, sistemas de comunicación de datos, software, seguridad informática y calidad de software, a los fines de alcanzar los objetivos fijados por la organización.	Bajo
CE.6.1. Asesorar y capacitar a organizaciones, empresas, organismos públicos o privados en la adquisición, instalación y uso, en lo que respecta a sistemas de información, sistemas de comunicación de datos, software, seguridad informática y calidad de software, a los fines de un uso correcto de los sistemas intervinientes.	No aporta
CE.7.1. Realizar pericias, tasaciones y arbitrajes relacionados con su actividad profesional, respetando marcos normativos y jurídicos con el objeto de asesorar a las partes o a los tribunales de Justicia.	No aporta

4. Contenidos Mínimos

Arquitectura de aplicaciones multicapa.

Herramientas de soporte al proceso de desarrollo.

Programación de la interfaz de usuario de una aplicación.

Aplicaciones orientadas a servicios.

Desarrollo Seguro.

Pruebas unitarias.

5. Objetivos establecidos en el DC

- Conocer las arquitecturas, herramientas y patrones para el desarrollo de software.
- Desarrollar interfaces de usuario.
- Crear soluciones de software que den respuestas a necesidades reales.
- Aplicar buenas prácticas y tecnologías en el desarrollo seguro.

6. Resultados de aprendizaje

Los siguientes resultados de aprendizaje se promueven en el desarrollo de la asignatura

Identificador de RA	Redacción
RA1	Construir una solución de software integral compuesta por Backend y Frontend con la finalidad de satisfacer los tipos de requerimientos típicos que tienen en común los portales web, teniendo en cuenta creación, desarrollo y mantenimiento del software involucrado tanto del lado del cliente como del servidor.
RA2	Elaborar una aplicación web utilizando HTML5, CSS 3 y lenguajes de programación modernos con el fin de obtener un producto de software funcional, dinámico y escalable en acuerdo con necesidades específicas de negocio y de usuario final.
RA3	Emplear principios y buenas prácticas de seguridad a lo largo del ciclo de vida del software para garantizar un desarrollo de back-end y front-end seguro según los lineamientos de la industria.
RA4	Aplicar pruebas unitarias para verificar el correcto funcionamiento de componentes y fragmentos de código back-end tomando en cuenta el comportamiento esperado.
RA5	Emplear herramientas de versionado de código con el fin de compartir el código fuente y mantener un registro de los cambios efectuados según las buenas prácticas de la industria.

7. Relación de los RA y las competencias

En la tabla siguiente se indica con X la tributación de cada Resultado de Aprendizaje con las competencias de egreso: específicas, genéricas tecnológicas, sociales, políticas y actitudinales de la carrera.

RA	CE1.1	CE1.2	CE1.3	CE2.1	CE3.1	CE4.1	CE5.1	CE6.1	CE7.1	CG1	CG2	CG3	CG4	CG5	CG6	CG7	CG8	CG9	CG10	CG11
RA1	-	-	X	X	-	X	X	-	-	X	X	-	X	-	-	X	-	-	X	-
RA2	-	-	X	-	-	-	-	-	-	X	X	-	X	-	-	-	-	-	X	-
RA3	-	-	X	X	-	-	X	-	-	X	X	-	X	-	-	-	-	-	X	-
RA4	-	-	X	-	-	X	-	-	-	X	X	-	X	-	-	-	-	-	X	-
RA5	-	-	X	-	-	-	-	-	-	X	X	-	X	-	X	X	-	-	X	-

8. Asignaturas correlativas previas

Para cursar y rendir debe tener cursadas:

- Asignatura/s:

  Paradigmas de Programación

  Análisis de Sistemas de Información

Para cursar y rendir debe tener aprobada:

- Asignatura/s:

  Lógica y Estructuras Discretas

  Algoritmos y Estructuras de Datos

9. Asignaturas correlativas posteriores

Indicar las asignaturas correlativas posteriores:

- Asignatura/s:

  Seguridad en los SI

  Proyecto Final

  Ingeniería y Calidad de Software

10. Programa analítico

Este programa analítico contempla los contenidos mínimos, previstos en el DC vigente, y aquellos que se consideran necesarios para desarrollar los resultados de aprendizaje propuestos.

Unidad Nº: 1
Título: Iniciándose en el desarrollo full stack.
Contenidos

- *Stack de tecnologías*: front-end (App SPA con React) y back-end (App de Microservicios con Node JS y Express JS). Lenguajes Java Script (Vanilla), Pruebas de Java Script (https://jestjs.io/) ECMA Script. Rol del desarrollador full stack. Responsabilidades del front-end y back-end.
- *Arquitectura Cliente Servidor*: Arquitectura de microservicios vs arquitectura monolítica.
- *Versionado de código*: GitLab colaborativo – Configuraciones locales y globales. Comandos básicos. Manejo de ramas. Resolución de conflictos. Buenas prácticas. Uso de IA.

Carga horaria por Unidad: 20 hs.

Unidad Nº: 2
Título: Javascript
Contenidos

- Sintaxis javascript: Variables. Expresiones. Tipos de datos. Operadores. Funciones. Estructuras de control. Arrays. Módulos. Manejo de errores.

- *Objetos javascript*: Trabajo con objetos. Modelo de objetos basado en prototipos. Promesas. Iteradores.
- *Desarrollo seguro y javascript*.

Carga horaria por Unidad: 20 hs

Unidad Nº: 3
Título: Desarrollo web back-end
Contenidos

- *NodeJS*. Paquetes, dependencias, librerías, frameworks. Desarrollo de servidores. Eventos asíncronos.
- *ORM*: Conceptos básicos de estructura de base de datos: Creación de bases de datos, tablas, operaciones para manipular datos. Mapeo de objeto relacional (ORM).
- *Arquitectura*: Principios de arquitectura: Acoplamiento - Cohesión - Modularidad – Unicidad. Identificación de patrones.
- *Rest*: Protocolo HTTP y verbos. Uso de Json. Conceptos REST. Estructura de URI, ubicación de recursos. Express. Servidor Web. Creación de API CRUD. CORS. Swagger – postman – curl.
- *Pruebas unitarias*: Pruebas unitarias. Concepto. Ventajas. Construcción. Uso de Jest.
- *Segurización de APIs*: Alternativas. Uso de Json Web Token. Identificación de patrones.
- *Desarrollo seguro aplicado al back-end*.

Carga horaria por Unidad: 34 hs

Unidad Nº: 4
Título: Desarrollo web front-end
Contenidos

- HTML y CSS: Introducción a front-end (HTML, CSS). Concepto de Servidor estático vs Aplicación Dinámica. Diferencia entre página y aplicación. Servidor de recursos vs servidor de aplicación. SPA (Single Page Application).
- *DOM* (Document Object Model): Vistas responsive, diferentes dispositivos. Bootstrap.
- *React*: Conceptos iniciales de React y estructura de aplicación. Renderizado: Virtual DOM, cómo funciona. Estructura de componentes. Componentes padres, principio de responsabilidad única en componentes hijos, alta cohesión y bajo acoplamiento. Componentes de clase vs funcionales. Propiedades. Uso de funciones callback. Hooks iniciales: useState, useEffect. Ciclo de vida de componentes, jsx. Formularios. Ruteos.
- *Identificación de patrones*.
- *Desarrollo seguro aplicado al front-end*.

Carga horaria por Unidad: 34 hs

Unidad Nº: 5
Título: Integración front-end y back-end.
Contenidos

- *Integración del front-end con el back-end*
- *Log de aplicaciones*.
- *Despliegue*.
- *Desarrollo seguro en la integración front-end y back-end*.

Carga horaria por Unidad: 20 hs

Carga horaria por tipo de formación práctica de toda la asignatura

Tipo de formación práctica	Horas reloj
Formación experimental	18
Análisis y resolución de problemas de ingeniería y estudios de casos	40
Formulación, análisis y desarrollo de proyectos.	11

Bibliografía Obligatoria:

-Design Thinking en Español (31 de octubre 2022) - https://www.designthinking.es/inicio/index.php

-Tecnología para desarrolladores web – JavaScript – Guía de JavaScript (31 de octubre 2022) – https://developer.mozilla.org/es/docs/Web/JavaScript/Guide/Introduction

-Una introducción a javascript (22 de agosto de 2022) – https://es.javascript.info/intro

-Sitio Oficial Nodejs (31 de octubre 2022) – https://nodejs.org/es

-Sitio Oficial de Json (31 de octubre 2022) – https://www.json.org/json-en.html

-Sitio Oficial React (31 de octubre 2022) – https://es.reactjs.org/docs/getting-started.html

-gitbook- Creación de una API (31 de octubre 2022) – https://juanda.gitbooks.io/webapps/content/api/arquitectura-api-rest.html

- Sitio Oficial Git (31 de octubre 2022) – https://git-scm.com/book/es/v2/Inicio---Sobre-el-Control-de-Versiones-Fundamentos-de-Git

Bibliografía optativa y otros materiales a utilizar en la asignatura:

-Justin Albano, Dave Whiteley – Dzone > Refcardz > Node.js (31 de octubre 2022) https://dzone.com/refcardz/nodejs

-Ibm Cloud Ediction – API REST (31 de octubre 2022) – https://www.ibm.com/ar-es/cloud/learn/rest-apis

11. Metodología de enseñanza

La asignatura se apoya en la combinación de varias metodologías de enseñanza:
#Talleres:
-Los temas son abordados llevando a cabo un ejercicio que se inicia y concluye en clase. Se acompañan con material a modo de taller de manera que los alumnos puedan seguirlo a su ritmo. Se acompaña a cada taller con descripción de conceptos fundamentales a fin de profundizar.
-Las clases se realizan en los laboratorios de Informática de la Facultad.
#Aula Virtual:
-En el aula virtual se dispone material de cada clase distribuido en solapas por unidad.
-Se proveen instructivos sobre cómo instalar software y componentes en los ordenadores personales.
-Se disponibilizan cuestionarios múltiple choice que se responden en clase.

-En el GitLab(*) que provee el Laboratorio de Informática se publicarán algunas presentaciones y notas de cátedra.
-Se disponibilizan cuestionarios por tema.
-Se plantean desafíos de programación.
-Se moderan foros para temas específicos.
#Aula Invertida:
-Se promueve la búsqueda de información en sitios oficiales y la formación de criterio para elaborar alternativas de solución ante problemas simples concretos planteados.
-A través de simulacros de parcial que los estudiantes pueden resolver antes de la clase, se resuelve luego el simulacro en clase y se debaten diferentes temas.
#Producción de videos:
-Se plantean situaciones problemáticas y se solicita a los estudiantes grabar un video comentando la solución. Por ejemplo, cómo se generan conflictos en código al interactuar varios desarrolladores y cómo lo han resuelto.
#Pensamiento de Diseño o Design Thinking:
-Se trabaja en grupo con el objetivo de compartir con los pares opiniones, buenas prácticas, experiencias y avanzar en la construcción de un prototipo.
-Se emplea esta metodología para todo el desarrollo de código que se lleva a cabo. A través del uso de las siguientes etapas se busca puede incluso mejorar la propuesta de solución del docente.
Empatía: Empatizar con los usuarios potenciales (Qué, cómo, por qué, para qué y dónde).
Definir: Definir el problema a resolver.
Idear: Etapa de creatividad, innovación y realismo puro y duro.
Prototipar: Construir una maqueta o prototipo lo más cercano a la realidad de la solución deseada.
Probar: Probar si el prototipo satisface los requerimientos.
#Dinámica de clases:
Durante la clase se plantean temas siguiendo la estructura introducción, desarrollo, cierre. Al finalizar, se anticipa el tema de la siguiente y se sugieren videos, lecturas para consultar y de esa manera lograr mayor participación, aprendizaje y profundidad en los temas. Una porción de tiempo de la clase se usa para analizar dudas y elementos relevantes referidos a tareas semanales, ejercicios extras para profundizar o exposición de grupos.
#Aprendizaje cooperativo
-Trabajo en equipo y en la consecución de objetivos comunes, que tan solo se pueden alcanzar si cada miembro del proyecto cumple con su tarea.
Se forman grupos de entre tres y seis miembros, en los que cada persona adquiere un rol determinado. Con el fin de alcanzar el objetivo fijado, deberán trabajar de forma coordinada para resolver los posibles conflictos que puedan surgir entre ellos, dejar a un lado la competitividad y priorizar el bien común.
#Resolución de ejercicios
-Se resuelven ejercicios en forma individual con apoyo del docente y material didáctico con ejercicios y fundamentos relativos al tema.
Se apunta a lograr un perfil híbrido con pensamiento crítico, multiskill que puede desenvolverse tanto en el front-end, como en el back-end de un desarrollo web.
Acorde a la evolución del mercado se revisa el contenido y las implementaciones prácticas vigentes. En consecuencia, se ajusta el contenido por ejemplo, relativo a librerías y desarrollo seguro.
Las tareas semanales integran los conocimientos de las distintas unidades.
*GitLab se basa en varios componentes que forman una solución completa para desarrolladores y la gestión de proyectos. Se pueden crear proyectos o repositorios para alojar código, describir funcionamiento, recomendaciones, guías, colaborar en ellos o identificar problemas. Las funciones nativas de integración continua y entrega continua (Gitlab CI/CD) permiten el desarrollo, las pruebas y el despliegue continuos de una aplicación.

12. Recomendaciones para el estudio

-Analizar los videos sugeridos.

-Leer el material sugerido previo a cada clase.

-Realizar los cuestionarios de autoevaluación o desafíos propuestos semana a semana.

-Llevar a cabo en clases la práctica a modo de taller con la guía del docente.

-Repetir la ejercitación vista en clase a fin de afianzarse.

-Repetir la ejercitación relativa a base de datos.

13. Metodología de evaluación

El modelo de enseñanza basado en competencias implica la aplicación de metodologías e instrumentos de evaluación que permiten conocer, a docentes y estudiantes, el nivel de desarrollo de las competencias que aborda la asignatura.

Al inicio del cursado se hace una evaluación diagnóstica en la primera clase, donde se buscan distinguir los conceptos previos que el grupo posee. Luego se hace un debate y el docente resalta los ítems sobresalientes; normalmente se puede detectar la incidencia de estudiantes recursantes de la asignatura y nociones previas del grupo.
En cuanto a la evaluación sumativa, la cátedra dispone de dos parciales y dos parciales de recuperación junto a un trabajo integrador grupal. No se considera ningún parcial extra o parcial integrador. El parcial y recuperatorio se aprueban al alcanzar los porcentajes indicados en la escala. Tanto los parciales como el recuperatorio contemplan aspectos del saber (medibles a través de un cuestionario) como aspectos del saber hacer (medibles a través de un código particular funcionando de acuerdo a la consigna planteada).
En cuanto a la evaluación formativa se lleva a cabo en los momentos de clases (se lleva un portfolio de participación en clase). También en tiempos extra áulicos, a través de herramientas del aula virtual (cuestionarios semanales, desafíos de programación al finalizar un tema, ambos de carácter individual y una actividad grupal de generación de video para demostrar el uso adecuado de herramientas git y resolución de conflictos).
Utilizamos gitLab como un potente portafolio ya que registra el trabajo individual del estudiante; permite poder acceder a la construcción del aprendizaje a lo largo de toda la cursada. Es una herramienta valiosa porque se puede acceder en todo momento para visualizar la evolución del estudiante.

Criterios:
En general:
-Se miden aspectos como la interpretación de consignas, resolución de problemas usando lógica de programación, implementación de validaciones, manejo de componentes y actualización de datos en una base de datos.
-Puntualidad en evaluaciones y tiempos de entrega.
-Dominio técnico del tema evaluado.
-Capacidad de integrar conceptos y aplicaciones.
-Dominio de terminología técnica.
-Expresión clara y técnicamente correcta.

En particular:
Los trabajos se consideran completos:
--Cuando participan todos los integrantes del grupo (versionado de código).
-Su documentación se encuentra almacenada en el repositorio en GitLab creado a tal efecto.

A continuación, se detallan todos los Resultados de Aprendizajes con sus contenidos a desarrollar para alcanzarlos, la mediación pedagógica, metodologías y estrategias de evaluación, tiempo en horas reloj.

Resultados de Aprendizaje	Contenidos según programa	Mediación Pedagógica	Metodología y Estrategias de Evaluación	Tiempos en hora reloj
RA 1	Stack de tecnologías – Arquitectura cliente servidor – Microservicios – Sintaxis javascript – NodeJS – Rest – Identificación de patrones	**Estrategias** - Lección magistral participativa - Resolución de ejercicios - Taller dirigido - Aprendizaje Cooperativo en Grupos Pequeños - Design Thinking - Observación del portafolio. **Actividades del estudiante** -Usando actividades de las distintas etapas de la metodología Design Thinking (empatía – definir – idear – prototipar - probar) se establece un intercambio de opiniones entre el docente y el estudiante para decidir alternativas de solución. Se usa la metodología Design Thinking para pensar en etapas el diseño de un prototipo mínimo que contemple desarrollo back.end y front-end con participación activa del estudiante. -Partipación del estudiante en los temas relativos a	**Instrumentos** - Evaluación diagnóstica sobre contenidos previos. - Cuestionarios sobre los contenidos trabajados. - Portafolio individual. - Portafolio grupal. - Trabajo integrador grupal. - Documentación del trabajo integrador grupal. - Parcial. - Evaluación por pares (pares individuales o pares grupos). **Criterios** -Participación en clase. -Evidencia de trabajo en grupo a través de las intervenciones en el código y defensa en el coloquio final. -Interpretación de consignas. -Resolución de problemas usando lógica de programación.	Trabajo en clase: 8 Horas de laboratorio: 20 Horas extra áulicas: 2

		introducción en stack de tecnologías, microservicios y rest. -Resolución de ejercicios definidos en la guía de trabajos prácticos. -Presentación oral en grupo comparten los patrones que han identificado en los frameworks usados en el desarrollo full stack, se basan en trabajos exploratorios y se discuten en clase. -A través del portafolio individual comparte aprendizajes alcanzados, conclusiones, evidencia de versionado de código y funcionamiento acorde a los requerimientos solicitados. -Desde cero haciendo uso de talleres dirigidos (por ejemplo, para construir APIs) paso a paso construimos en clase un proyecto full stack de punta a punta. Cada clase iniciamos un tema, lo desarrollamos y finalizamos. El estudiante repite el proceso en clase o bien en otro momento guiado por instructivos puede resolver	-Manejo adecuado de componentes. -Actualización de datos en una base de datos. -Evidencia de funcionamiento según la consigna. -Vocabulario técnico y preciso en la documentación del trabajo grupal integrador.	

		ejercicios de la guía de trabajos prácticos similares.		
RA 2	DOM – Virtual DOM – HTML – CSS - REACT	**Estrategias** - Lección magistral participativa - Resolución de ejercicios - Taller dirigido - Aprendizaje Cooperativo en Grupos Pequeños **Actividades** -A través de clase magistral participativa se presentan componentes tipo, se ejemplifica la construcción de vistas. -Aula invertida para explorar componentes tipo. -Taller dirigido para que el estudiante implemente vistas guiado por el docente. -Resolución de ejercicios sobre vistas. - Actividades de exploración individual guiadas para el uso de herramientas satélites necesarias en la construcción de un proyecto de tipo full stack. Se analiza y debate la aplicación de cada herramienta en proyectos públicos como es el caso de consumo de APIs mediante postman. Luego, se aplican a	**Instrumentos** - Evaluación diagnóstica sobre contenidos previos. - Cuestionarios sobre los contenidos trabajados. - Portafolio individual. - Trabajo integrador grupal. - Parcial. - Evaluación por pares (pares individual o grupal). **Criterios** -Participación en clase. -Identificación de componentes. -Implementación de validaciones. -Manejo adecuado de componentes. -Vocabulario técnico y preciso en la documentación. -Manejo adecuado de postman.	Trabajo en clase: 8 Horas de laboratorio: 16 Horas extra áulicas: 2

		APIs creadas a lo largo del cursado de la materia.		
RA 3	Ciclo de vida de Desarrollo Seguro – Segurización de APIs – Despliegue - Logs	**Estrategias** - Lección magistral participativa - Aprendizaje Cooperativo en Grupos Pequeños - Seminario **Actividades** -El docente expone buenas prácticas de ciclo de vida de desarrollo seguro con participación activa de los estudiantes. Se discute en clase las ventajas y desventajas de su aplicación. Luego se pide a los grupos aplicar las buenas prácticas y compartirlas a través de una exposición donde se debate. -Se solicita a los grupos conformados incorporar prácticas de desarrollo seguro en la construcción del trabajo individual.	**Instrumentos** - Portafolio individual. - Documentación del trabajo. -Presentación oral (todos los grupos preparan, se elige a uno para exponer y el resto debate). **Criterios** - Vocabulario técnico y preciso sobre desarrollo seguro en la documentación del trabajo. - Incorporación de buenas prácticas de desarrollo seguro.	Trabajo en clase: 8 Horas de laboratorio: 12 Horas extra áulicas: 2
RA 4	Pruebas unitarias en back-end – Construcción y ejecución – Buenas prácticas	**Estrategias** - Taller dirigido **Actividades** -Los estudiantes completan el Taller dirigido sobre uso de Jest	**Instrumentos** - Repositorio de trabajo individual. - Documentación del trabajo integrador grupal.	Trabajo en clase: 8 Horas de laboratorio: 6 Horas extra áulicas: 2

		y lo repiten con funcionalidad similar -Los estudiantes agregan al menos una prueba unitaria en el portafolio individual. -Los estudiantes aportan su opinión respecto del uso de pruebas unitarias.	**Criterios** -Participación con pruebas unitarias aplicadas. -Lenguaje técnico apropiado en la documentación.	
RA 5	Versionado de código fuente – Resolución de conflictos.	**Estrategias** - Lección magistral participativa - Aprendizaje Cooperativo en Grupos Pequeños - Producción de videos **Actividades** Mediante un taller dirigido se inicia al uso de versionado de código fuente. Se acompaña el desarrollo de un trabajo grupal integrador donde se comparte código a través del versionado. Los estudiantes agregan y modifican código generando conflictos y acuerdan como resolverlos. Mediante el aprendizaje colaborativo se asignan distintos roles en la actividad.	**Instrumentos** - Portafolio grupal: video producido para demostrar la resolución de conflicto. **Criterios** -Demostrar habilidad den la resolución de conflictos. - Participación de los distintos integrantes del grupo agregando, quitando y modificando código fuente en el repositorio. -Participación en la resolución de conflictos en el versionado de código a través de la producción de videos explicativos creados por los estudiantes integrantes de cada grupo.	Trabajo en clase: 4 Horas de laboratorio: 6 Horas extra áulicas: 2

14. Condiciones de aprobación

Se detallan a continuación las distintas condiciones del estudiante:

Para la regularización:

-Parciales o recuperatorios aprobados (se conserva la mejor nota) con nota igual o superior a 4.

-Realizar 80% de los cuestionarios múltiple choice propuestos en Moodle. Resolver en clase.

-Realizar 80% de las actividades obligatorias propuestas.

-Realizar el 80% de los desafíos propuestos.

-Evidencia de paso a paso frontend y backend en el portafolio individual con todos requerimientos necesarios y nota igual o superior a 4.

-La condición se alcanza al aprobar las instancias de evaluación con nota de 4 (cuatro) o superior, y sin haber cumplido con las condiciones de alcanzar la Aprobación Directa.

-Quien llega a esta condición, deberá rendir el examen final completo, con contenidos del programa vigente.

Para la aprobación de aprobación directa:

-Parciales o recuperatorios aprobados (se conserva la mejor nota), nota mínima 6.

-Evidencia de Paso a paso frontend – backend en portafolio individual aprobado con todos requerimientos necesarios y nota igual o superior a 7.

-Realizar 100% de los cuestionarios múltiple choice propuestos en Moodle con promedio de nota mayor igual a 7. Realizar en clases con opción a recuperatorio en fecha a definir.

-Realizar el 100 % de los desafíos.

-Realizar 100% de actividades obligatorias propuestas.

-Promedio final igual o superior a 7.

-Haber aprobado las evaluaciones planteadas con nota no inferior a 7 (siete), aunque haya accedido a recuperatorio. Esto es válido tanto para trabajos individuales o grupales, como para evaluaciones sumativas e integradoras. Quienes alcancen la Aprobación Directa sólo deben inscribirse al examen final y presentarse personalmente, con su libreta completada, para el registro de su aprobación.

Escala de notas para regularidad / aprobación directa

NOTAS	PORCENTAJE	CALIFICACION
1		No Aprobado
2		No Aprobado
3		No Aprobado
4	55% a 57%	Aprobado
5	58% a 59%	Aprobado
6	60% a 68%	Aprobado
7	69% a 77%	Aprobado
8	78% a 86%	Aprobado
9	87% a 95%	Aprobado
10	96% a 100%	Aprobado

15. Modalidad de examen

Condición Regular

- Examen Individual. Incluye completar el desarrollo en PC de laboratorio de un proyecto/programa que cumpla las consignas de un enunciado particular contemplando funcionalidad backend – frontend.
- Coloquio

Condición Aprobación Directa

- La condición de Aprobación Directa exime al estudiante de realizar el examen final. La calificación se obtiene a partir de un promedio de las notas obtenidas.

Escala de notas para examen final:

NOTA	PORCENTAJE	CALIFICACIÓN
1		Insuficiente
2		Insuficiente
3		Insuficiente
4		Insuficiente
5		Insuficiente
6	60% a 68%	Aprobado
7	69% a 77%	Bueno
8	78% a 86%	Muy Bueno
9	87% a 95%	Distinguido
10	96% a 100%	Sobresaliente

16. Recursos necesarios

Las clases se llevan a cabo en laboratorio, donde se requiere:

IDE Visual Studio Code

ORM sequelize

Nodejs
Expressjs
Jest
React
Postman
Nexus
Curl
GitLab
Git bash
Máquina virtual
Guía de trabajo: Publicada en GitLab
Proyector.