Planificación anual por trimestre Informática Personal y Profesional

ESPACIO CURRICULAR:	Programación II
CURSO:	5 “E” y “F”
DOCENTE:	Araujo Oscar, Kolb, Mariela Elizabeth

FUNDAMENTACIÓN

En este espacio curricular, los estudiantes podrán escribir algoritmos utilizando estructuras más complejas que permitirán mayor eficiencia en la resolución de sus problemas. Verificar y depurar el producto desarrollado para asegurarse que cumple con las especificaciones recibidas.

Se continúa con lo desarrollado en el espacio curricular del mismo nombre durante el año anterior, Programación I. Se busca la adquisición de más experiencia, en cuanto a técnicas, del lenguaje utilizado en esa oportunidad (PHP).

PROPÓSITOS

Tomar el hábito de la diagramación, pruebas de escritorio y otras técnicas, para luego codificar con seguridad, como así también, optimizar código mediante “buenas prácticas”. Solucionar problemas complejos con la ayuda de grupos y docentes.

OBJETIVOS

- Que los alumnos sean capaces de resolver de manera eficaz distintos problemas de carácter general, con independencia del lenguaje de programación utilizado. Este objetivo requiere que los alumnos adquieran el conocimiento teórico y práctico sobre las técnicas básicas de programación estructurada y orientada a objetos logrando la *obtención de ideas intuitivas y claras de los conceptos y técnicas estudiados* facilitando la aplicación práctica de los algoritmos.

COMPETENCIAS

A) **BÁSICAS**

- RESUELVE problemas complejos mediante algoritmos basados en el paradigma estructurado y orientado a objetos.
- OPTIMIZA el código de programación aplicando las buenas prácticas.

B) **ESPECIFICAS**

- TOMA el hábito de las buenas prácticas para codificar.
- VALORA la importancia de la documentación de su código.
- REALIZA las pruebas necesarias para evitar errores en tiempo de ejecución.
- INTEGRA eficientemente bases de datos en sus soluciones.
- COMPRENDE los conceptos básicos de la programación orientada a objetos (POO).
- APLICA la POO para resolver problemas prácticos.

CONTENIDOS

PRIMER TRIMESTRE	CAPACIDADES	ACTIVIDADES	INDICADORES/ EVIDENCIAS DE DESEMPEÑO
Revisión y repaso de temas de programación estructurada, mediante prácticas usando lenguaje PHP (complementadas con HTML-CSS-JS): • Arreglos unidimensionales (vectores) y bidimensionales (matrices). • Declaración. • Métodos de búsqueda y ordenamiento. • Arreglos de caracteres.	▪ Domina el uso de vectores y matrices (PHP). ▪ Comprende cómo implementar funciones para simplificar problemas complejos. ▪ Comprende cómo acceder archivos (PHP). ▪ Comprende cómo conectar una base de datos según el tipo de conexión.	Mediante caso práctico: ▪ Crear un algoritmo que utilice vectores. Codificarlo en PHP. Hacer pruebas para depurar posibles errores. ▪ Crear un algoritmo que utilice matrices. Codificarlo en PHP. Hacer pruebas para depurar posibles errores. Mediante caso práctico: ▪ Crear algoritmo que implemente funciones. Codificarlo en PHP. Hacer	▪ Resuelve satisfactoriamente ejercicio práctico implementando vectores. ▪ Resuelve satisfactoriamente ejercicio práctico implementando matrices. ▪ Resuelve satisfactoriamente ejercicio práctico implementando funciones. ▪ Resuelve satisfactoriamente ejercicio práctico implementando archivos.

• Funciones: Ventajas de dividir la problemática en problemas menores. • Pasaje de parámetros por valor y por referencia. • Procesamiento de archivos. • Acceso a bases de datos: conexión. Tipos de conexiones.		pruebas para depurar posibles errores. Mediante caso práctico: ▪ Crear algoritmo que implemente el uso de archivos. Codificarlo en PHP. Hacer pruebas para depurar posibles errores. Mediante caso práctico: ▪ Codificar el acceso a una base de datos en PHP. Hacer pruebas para depurar posibles errores.	▪ Resuelve satisfactoriamente ejercicio práctico para conectar bases de datos.
SEGUNDO TRIMESTRE	**CAPACIDADES**	**ACTIVIDADES**	**INDICADORES/ EVIDENCIAS DE DESEMPEÑO**
• Operaciones de manipulación de datos en una base de datos: insertar, modificar, borrar y/o consultar registros. • Acceso a datos de una base de datos mediante permisos. Registrar un nuevo usuario y/o acceder a una cuenta de usuario registrado. • Uso de Sesiones en PHP. **▶ PROYECTO SCYF** Módulo de autenticación y gestión de usuarios del sistema SCYF:	▪ Comprende cómo insertar un nuevo registro, modificar y eliminar un registro existente en una base de datos. ▪ Comprende cómo realizar autenticación de usuarios y permisos, usando base de datos. ▪ Comprende el uso de sesiones en aplicaciones que requieren autenticación de acceso de usuarios. Implementa un sistema de autenticación funcional con roles	Mediante caso práctico: ▪ Mediante el paradigma estructurado, realizar el código para el registro de nuevo usuario y posterior autenticación de acceso a panel de acciones CRUD (crear, leer, actualizar, borrar) sobre tablas. Actividad SCYF — Backend: autenticación y CRUD contable: • Crear tabla 'usuarios' con campos: id, nombre, email, password_hash, rol.	• El sistema de registro e inicio de sesión funciona correctamente con datos de prueba. • Las operaciones CRUD sobre la tabla elegida se ejecutan sin errores y reflejan cambios reales en la BD. • Las sesiones se inician, mantienen y cierran correctamente. • El código está organizado en funciones y comentado. • El login rechaza credenciales incorrectas y acepta las correctas, redirigiendo según el rol del usuario. • El CRUD sobre cuentas contables funciona completo: crear, leer, editar y eliminar sin errores.

• Registro de usuarios con roles: Administrador y Contador • Login con validación contra la BD (contraseña hasheada con password_hash) • Sesiones PHP para mantener el acceso autenticado • Restricción de vistas según rol (Admin ve todo; Contador solo su módulo) • Logout y destrucción de sesión Módulo CRUD sobre datos contables del sistema SCYF: • Alta, baja y modificación de cuentas del Plan de Cuentas • Registro de transacciones (compras, ventas, pagos, cobros) • Validación de la partida doble: debe = haber antes de confirmar el INSERT • Prevención de inyección SQL con prepared statements (PDO) • Paradigma Orientado a Objetos: introducción. • Clases vs. Objetos. Ciclo de vida de los objetos. • Propiedades de un objeto. **▶ PROYECTO SCYF** Primeras clases del sistema SCYF en POO: • Clase Cuenta: propiedades id, codigo, nombre, tipo	(Admin / Contador) aplicado al sistema SCYF. Codifica las operaciones CRUD sobre tablas contables reales, validando la integridad de los datos antes de ejecutar la consulta. Comprende que la validación de entradas (prepared statements) no es opcional: es la primera medida de seguridad del sistema. Relaciona el uso de sesiones con el control de acceso por rol dentro del sistema contable. ▪ Comprende los conceptos de la programación orientada a objetos. ▪ Comprende cómo y cuando crear/definir clases y objetos. ▪ Comprende cómo implementar propiedades de un objeto. Define las primeras clases del sistema SCYF (Cuenta, Usuario) con sus propiedades y constructores. Refactoriza el CRUD procedural de cuentas para utilizar objetos de la clase Cuenta. Comprende que modelar entidades del dominio (contable) como clases mejora la organización y mantenibilidad del código.	• Implementar registro de usuario con password_hash() y validación de email único. • Implementar login: verificar contraseña con password_verify(), iniciar sesión y redirigir según rol. • Construir panel CRUD para gestión del Plan de Cuentas (solo accesible con rol Admin). • Implementar formulario de registro de transacción con validación PHP: si debe ≠ haber, mostrar error y no ejecutar INSERT. • Usar PDO con prepared statements en todas las consultas a la BD. • Entregable: módulo funcional subido al repositorio Git del equipo SCYF. • Vinculación BD2: las tablas y datos usados aquí son los mismos creados con DDL/DML en Base de Datos 2. ▪ Convertir el caso práctico de Rgistro-Autenticación+CRUD, utilizando el paradigma orientado a objetos. Actividad SCYF — Primeras clases del sistema contable: • Crear la clase Cuenta con todas sus propiedades y un constructor que las inicialice. • Crear la clase Usuario con sus propiedades y un método estático login() que encapsule la lógica de autenticación.	• El formulario de transacción rechaza y notifica al usuario cuando debe ≠ haber (partida doble). • Ninguna consulta a la BD usa concatenación directa de variables: todas usan prepared statements. • El código está versionado en Git con commits descriptivos. • Evidencia cruzada con Seguridad Informática: el módulo supera una revisión de las medidas de protección implementadas. Define correctamente una clase con propiedades y un constructor. • Instancia objetos y accede a sus propiedades sin errores. • Explica oralmente la diferencia entre clase y objeto usando el caso práctico como ejemplo. • Las clases Cuenta y Usuario están correctamente definidas, con propiedades y constructores funcionales. • El código refactorizado con POO produce el mismo resultado que el procedural y supera las mismas pruebas. • El alumno justifica por escrito al menos dos ventajas de usar POO en el contexto del sistema SCYF. • Evidencia: diagrama de clase (UML básico) de Cuenta y Usuario entregado junto al código.

(activo/pasivo/patrimonio/ingreso/egreso), saldo • Clase Usuario: propiedades id, nombre, email, passwordHash, rol • Constructor e instanciación de objetos desde datos de la BD • Refactorización del CRUD de cuentas usando la clase Cuenta.		• Refactorizar el formulario de alta de cuenta para instanciar un objeto Cuenta antes de hacer el INSERT. • Comparar el código procedural vs. el orientado a objetos: ¿cuál es más fácil de mantener? • Vinculación BD2: los atributos de Cuenta deben coincidir exactamente con las columnas de la tabla 'cuentas_contables'.	
TERCER TRIMESTRE	**CAPACIDADES**	**ACTIVIDADES**	**INDICADORES/ EVIDENCIAS DE DESEMPEÑO**
• Métodos de acceso. Calificadores de los métodos. Nivel de acceso de métodos y propiedades (public, protected, private). **▶ PROYECTO SCYF** Métodos del sistema SCYF (encapsulamiento de lógica contable): • Clase Cuenta: método getSaldo(), método estaActiva() • Clase Transaccion: propiedades fecha, descripcion, debe, haber, cuenta_id; método validarPartidaDoble() • Clase Usuario: método tienePermiso($accion), método cerrarSesion() • Clase ConexionBD: método estático getConexion() — patrón Singleton • Visibilidad: propiedades private, getters/setters públicos.	▪ Comprende cómo implementar métodos en un objeto. ▪ Comprende el nivel de acceso de métodos y propiedades. Implementa métodos que encapsulan la lógica de negocio contable dentro de las clases del sistema SCYF. Aplica correctamente los calificadores public/private/protected según la responsabilidad de cada método. Comprende que encapsular la validación (validarPartidaDoble) dentro de la clase Transaccion garantiza que ninguna parte del sistema pueda saltarse la regla contable. Implementa el patrón Singleton para la conexión a la BD, evitando conexiones duplicadas.	Mediante trabajo práctico integrador: • Implementar lo abordado sobre el paradigma orientado a objetos. Mediante trabajo práctico integrador: • Implementar herencia y polimorfismo en el proyecto. Actividad SCYF — Métodos y encapsulamiento en el sistema contable: • Agregar método validarPartidaDoble() a la clase Transaccion: retorna true solo si debe == haber. • Agregar método getSaldo() a Cuenta: consulta la BD y retorna SUM(debe)-SUM(haber). • Implementar ConexionBD con patrón Singleton: una sola instancia PDO reutilizable. • Implementar tienePermiso() en Usuario: verifica si el rol del usuario puede ejecutar la acción solicitada.	▪ Desarrolla satisfactoriamente una aplicación para la gestión contable que responda los criterios solicitados por los docentes de las asignaturas implicadas en el trabajo. • El método validarPartidaDoble() rechaza correctamente transacciones donde debe ≠ haber. • getSaldo() retorna el valor correcto verificable contra los datos de BD2. • ConexionBD devuelve siempre la misma instancia PDO (se puede verificar con spl_object_id). • tienePermiso() bloquea el acceso a funciones restringidas para el rol 'Contador'. • Evidencia: prueba de escritorio documentada con al menos 3 casos de prueba por método crítico.

• Conceptos de herencia. Interfaces. Clases abstractas. Polimorfismo. **▶ PROYECTO SCYF** Herencia y polimorfismo aplicados al sistema SCYF: • Clase abstracta DocumentoContable: propiedades fecha, descripcion; método abstracto generarReporte() • Clase LibroDiario extiende DocumentoContable: implementa generarReporte() listando asientos del período • Clase LibroMayor extiende DocumentoContable: implementa generarReporte() agrupando por cuenta • Clase EstadoResultados extiende DocumentoContable: implementa generarReporte() con ingresos - egresos • Interfaz Exportable: método exportarHTML() / exportarPDF() • Polimorfismo: un array de DocumentoContable[] recorre todos los reportes y llama a generarReporte(). • Aplicación de paradigma orientado a objetos en Proyecto Integrador. **▶ PROYECTO SCYF** Integración final del sistema SCYF en Programación II: • Módulo de Ingreso de Datos: formularios HTML/PHP para facturas y	▪ Comprende cómo implementar herencia y polimorfismo. Diseña una jerarquía de clases que modela los documentos contables del sistema SCYF usando herencia y polimorfismo. Implementa una clase abstracta DocumentoContable con métodos concretos y abstractos. Comprende que el polimorfismo permite agregar nuevos tipos de reporte sin modificar el código que los genera. Aplica interfaces para garantizar que todos los documentos puedan exportarse, independientemente de su tipo. Aplica de forma integrada todos los conceptos de POO vistos en el trimestre para resolver un problema complejo. Integra en un sistema funcional todos los módulos del sistema SCYF desarrollados durante el año, aplicando POO, acceso a BD, autenticación, validación y generación de reportes. Produce documentación técnica profesional (diagrama UML + manual de usuario). Trabaja en equipo con control de versiones Git de forma colaborativa y ordenada. Presenta el sistema ante la comunidad escolar con exposición oral fundamentando cada decisión técnica.	• Refactorizar el módulo de registro de transacciones para llamar a $transaccion->validarPartidaDoble() antes del INSERT. • Vinculación Seguridad: tienePermiso() es el control de autorización a nivel de código, complementa las VIEWs de BD2. Actividad SCYF — Jerarquía de reportes contables: • Definir la clase abstracta DocumentoContable con el método abstracto generarReporte(). • Implementar LibroDiario, LibroMayor y EstadoResultados como subclases concretas. • Cada generarReporte() ejecuta la query SQL correspondiente (las mismas construidas en BD2) y devuelve los datos. • Implementar la interfaz Exportable y aplicarla a las tres clases. • Crear un controlador que reciba un array de DocumentoContable, llame a generarReporte() polimórficamente y muestre los reportes en pantalla. • Vinculación BD2: las queries dentro de generarReporte() son exactamente las vistas (VIEW) creadas en Base de Datos 2. Mediante trabajo práctico integrador: • Implementar lo abordado sobre el paradigma orientado a objetos en el proyecto integrador interdisciplinar. Actividad SCYF — Presentación y entrega final del sistema:	• Define una jerarquía de clases con herencia y demuestra el polimorfismo con al menos dos clases hija. • Justifica cuándo es conveniente usar una clase abstracta vs. una interfaz. • Las tres clases de reporte implementan correctamente generarReporte() y producen resultados coherentes con los datos de la BD. • El controlador recorre el array polimórfico y muestra los tres reportes sin código condicional (sin if/switch por tipo). • La interfaz Exportable está implementada en las tres clases y el método exportarHTML() genera una salida visual. • Evidencia cruzada con BD2: los datos mostrados por EstadoResultados en PHP coinciden con el reporte SQL generado en Base de Datos 2. • El alumno expone oralmente la jerarquía de clases y explica por qué cada decisión de diseño (abstracta, interfaz, polimorfismo) fue la adecuada. • El proyecto integrador aplica correctamente: clases, objetos, encapsulamiento, herencia, polimorfismo e interfaces. • El código está documentado, versionado en Git y supera una revisión de pares. • El sistema completo es funcional: un usuario Contador puede iniciar sesión, registrar transacciones y ver los tres reportes. • La partida doble es validada en todo el flujo: formulario PHP (Transaccion->validarPartidaDoble) y BD (COMMIT/ROLLBACK en BD2).

recibos usando clases Transaccion y Cuenta • Módulo de Cálculo: clase AsientoAutomatico que genera la partida doble a partir de los datos del formulario • Módulo de Reportes: controlador polimórfico que genera LibroDiario, LibroMayor y EstadoResultados • Módulo de Seguridad: autenticación por rol + prepared statements + logs de auditoría (clase AuditoriaLog) • Documentación técnica: diagrama de clases UML + manual de usuario • Control de versiones: historial Git con ramas por módulo y commits descriptivos.		• Integrar todos los módulos en un sistema navegable desde un menú principal según el rol del usuario. • El módulo de ingreso genera el asiento contable automático aplicando la clase AsientoAutomatico. • El módulo de reportes muestra LibroDiario, LibroMayor y EstadoResultados en pantalla con opción de exportar a HTML. • La clase AuditoriaLog registra en una tabla BD cada acción del usuario (quién, qué, cuándo). • Entregar: repositorio Git con el sistema completo + diagrama UML de clases + manual de usuario. • Presentación oral ante docentes y pares: demostración en vivo del sistema funcionando con datos reales. • Vinculación interdisciplinar: los reportes del sistema deben coincidir con los generados en BD2 y con los estados financieros de SIC.	• El log de auditoría registra correctamente todas las acciones realizadas durante la demostración. • El diagrama UML refleja exactamente las clases implementadas. • Evidencia interdisciplinar: tabla comparativa entregada por el equipo mostrando que los datos del sistema SCYF coinciden en las tres materias (Programación II, BD2 y SIC). • La exposición oral incluye explicación del diseño de clases, decisiones de seguridad y dificultades resueltas.

PROPUESTA METODOLÓGICA PARA LA ENSEÑANZA

Inicialmente, se hace una revisión y repaso de los temas vistos en Programación I para reforzarlos. Luego, cada unidad temática se aborda presentando al alumno una perspectiva teórica de los conceptos, para luego exponer ejemplos prácticos que permiten desarrollar los aspectos avanzados. A partir del segundo trimestre, las actividades prácticas se enmarcan progresivamente en el Proyecto Interdisciplinario SCYF (Sistema de Gestión Contable y Financiera para PYMES), aplicando la metodología de Aprendizaje Basado en Proyectos (ABP). Esto permite que el alumno transfiera de inmediato los contenidos de PHP y POO a un sistema real en construcción, articulando con las materias de Base de Datos 2, SIC, Seguridad Informática, entre otras. En todo momento se busca la integración tanto de los conocimientos previamente adquiridos como también de los presentados en ese momento.

INSTRUMENTOS DE EVALUACIÓN

La evaluación del alumno es permanente.

Se utilizarán como instrumentos de evaluación:

- T.P.O.s: Realización de Ejercicios Prácticos obligatorios dónde la cantidad de líneas de código para resolver un problema es la unidad de calificación (50 líneas = ☺, >50 líneas = 😐, >100 líneas = ☹). Los entregables del Proyecto SCYF (módulos, clases, sistema integrado) se evalúan como TPO de mayor complejidad.
- Exposición Oral, presentación del sistema SCYF funcionando en vivo, con fundamentación de las decisiones de diseño.
- Evaluación Escrita, ejercicios de programación estructurada y POO.
- Realización de Trabajo Integrador (interdisciplinar) SCYF, sistema de gestión contable para PYMES (en caso de grupos, hasta 5 integrantes) utilizando el lenguaje PHP, donde intervenga como funcionalidades:
  - la gestión contable,
  - acciones CRUD,
  - registro y autenticación de usuarios (manejo de sesiones y roles).

FIRMA DE LOS DOCENTES

PROGRAMA

UNIDAD N°1: Revisión y repaso de temas de programación estructurada, mediante prácticas usando lenguaje PHP (complementadas con HTML-CSS-JS):

- Arreglos unidimensionales (vectores) y bidimensionales (matrices).
- Declaración.
- Métodos de búsqueda y ordenamiento.
- Arreglos de caracteres.
- Funciones: Ventajas de dividir la problemática en problemas menores.
- Pasaje de parámetros por valor y por referencia.
- Procesamiento de archivos.

UNIDAD N°2: Incorporación de Bases de Datos y uso de SQL.

- Acceso a bases de datos: conexión. Tipos de conexiones.
- Operaciones de manipulación de datos en una base de datos (CRUD): insertar, modificar, borrar y/o consultar registros.
- Acceso a datos de una base de datos mediante permisos. Registrar un nuevo usuario y/o acceder a una cuenta de usuario registrado.
- Uso de Sesiones en PHP.

UNIDAD N°3: Paradigma Orientado a Objetos

- Introducción a la POO
- Clases vs. Objetos. Ciclo de vida de los objetos.
- Propiedades de un objeto.
- Métodos de acceso. Calificadores de los métodos. Nivel de acceso de métodos y propiedades.
- Conceptos de herencia. Interfaces. Clases abstractas. Polimorfismo.
- Aplicación de paradigma orientado a objetos en Proyecto Integrador SCYF.

BIBLIOGRAFÍA

- Programación orientada a objetos en PHP, AVA - Formación en Ambientes Virtuales de Aprendizaje - SENA - Servicio Nacional de Aprendizaje.
- Nixon, R. (2015). Learning PHP, MySQL & Javascript, Fourth Edition. Cambridge: O'reilly
- https://www.php.net/manual/es/index.php.