Trabajos prácticos

Índice

Consideraciones generales

Se entrega un scope con una breve descripción de cada trabajo práctico. Será responsabilidad del alumno el diseño, consideraciones de negocio, diseño de base de datos (de ser necesario).

Tener en cuenta que no debe haber menos de 4 ni más de 6 entidades en total.

Tener en cuenta que en esta materia no se evalúa el diseño de base de datos, ni el manejo de SQL. En la materia sí se evalúa el diseño orientado a objetos. Se pone especial énfasis en conceptos vistos en la materia como Acoplamiento/Cohesión y reutilización de código.

La primera entrega, que cumple la función de segundo parcial, consistirá en un scope acotado, a saber: tomando **una** única entidad de todo el sistema, se deberá realizar un CRUD completo, incluyendo manejo de excepciones y pantallas en Swing.

Cada trabajo práctico deberá ser "defendido" el día de la evaluación final, explicando cuáles fueron las consideraciones de diseño y justificarlas.

Consideraciones particulares de cada trabajo práctico

1. Los trabajos prácticos, como cualquier instancia de evaluación, se aprueban con 4 (cuatro).
2. Funcionalidad básica: sin esta funcionalidad, al menos, el TP no aprueba. Si está y aplica los conceptos vistos en la materia, entonces se lo considerará simplemente "aprobado" (4 - cuatro).
3. Adicionales: para comenzar a hacer del TP un trabajo más completo y comenzar a sumar por encima de lo entregado en el punto 2.
4. *Bonus points*: se valorará (solo si el punto 2 y 3 están completos) el esfuerzo extra por cumplir con esos puntos.

Tema 1: Clon Outlook Express

Se debe desarrollar un cliente de correo, con funcionalidad similar al cásico Outlook express.

Funcionalidad básica:

- Enviar y recibir correos desde/hacia un servidor, mostrar bandeja de entrada, de salida y de enviados. El envío y la recepción NO deben desarrollarse desde cero, sino que se debe usar una librería. La más común es JavaMail (buscarla en el sitio de Java).
- Debe tener administración de contactos contra una base de datos. Los mensajes recibidos en de la bandeja de entrada y los borradores en la salida se pueden guardar en una base de datos o en archivos.
- Interfaz gráfica: botones en la parte superior para enviar y recibir y administrar contactos, lista de bandejas a la izquierda. Lista de correos a la derecha. Panel de lectura en la parte inferior de la pantalla.

Adicionales:

- Tener una pantalla de configuración del servidor (host, puerto, etc.).
- Manejar carpetas de correos, mostrándolos como un árbol en la parte izquierda.
- Tener los menús y los botones con las mismas opciones (Ejemplo: Menú Correo > Enviar y Recibir que haga y reutilice el mismo evento de enviar y recibir).
- Botones con mnemónicos o combinaciones de teclas.
- Agregar iconos a los botones.

***Bonus points*:**

- Autocompletar direcciones ya utilizadas anteriormente.
- Manejar un calendario.
- Manejar perfiles (diferentes usuarios se loguean y se cargan diferentes configuraciones de servidores).
- Barra de estado en la parte más inferior de la pantalla que indique qué se está haciendo (enviando y recibiendo… conectando… etc.).

Tema 2: Mini home banking

Se debe realizar un sistema de home banking de escritorio. Todos los datos (usuarios, cuentas, etc.) serán almacenados en una base de datos.

Funcionalidad básica:

- Administración de usuarios: cada usuario accede solo a sus cuentas (caja de ahorro, cta. cte., caja ahorro en dólares, etc.) e información. Habrá un usuario administrador que crea los usuarios y sus productos (sería el empleado del banco).
- Productos: cada usuario debe poder tener una o más cuentas, cada cuenta tiene una serie de créditos y débitos.
- Transferencias: se deben poder realizar transferencias entre cuentas, realizando los débitos y créditos correspondientes en las cuentas (en la base de datos). Las cuentas de origen se deben seleccionar de una lista. La cuenta destino sebe ser seleccionada por id o por cbu o por alias.
- Tarjetas: cada usuario puede tener 0 o más tarjetas. Cada tarjeta tendrá un disponible y un saldo a pagar.

Adicionales:

- Resumen: se debe poder emitir un resumen de los movimientos.
- Registrar movimientos de tarjeta; para simplificar la operatoria solo el usuario administrador, oficiando de entidad bancaria, podrá generarle débitos en las tarjetas a los usuarios. Se debe poder emitir un resumen de una tarjeta seleccionando un mes del año.

***Bonus points*:**

- Generar intereses de acuerdo con los diferentes tipos de cuenta.
- Generar archivos con los reportes de movimientos a pedido del usuario administrador (serían los que luego se imprimen y se envían por correo).
- Auditoría: registrar movimientos de cada usuario. Emitir reportes de auditoria (mostrar en pantalla).

Tema 3: Clon de postman

Se deberá realizar un cliente de servicios rest, con funcionalidad limitada, similar a la del clásico Postman. Para las requests http NO se debe realizar la funcionalidad desde cero, sino que se debe utilizar una librería. La más común es apache-commons-http.

Funcionalidad básica:

- Poder escribir una url y ejecutar métodos GET, POST, PUT, DELETE contra ella. Los parámetros de la url (querystring) se ingresan manualmente en la url (ej. servicor.com?a=123&b=xyz).
- Caja de texto o lista para adicionar headers de ser necesario. Uno por renglón.
- Caja de texto para escribir el body de la request.
- Caja de texto para visualizar la response.
- Administrar urls y parámetros "Favoritos" en una base de datos, que se deberán precargar al ser seleccionados.

Adicionales:

- Hacer del ingreso de headers una grilla.
- Poder adicionar parámetros de querystring en una tabla de claves-valores dedicada a ello, similar a la de los headers, con una grilla.
- Visualizar la respuesta en diferentes formatos (si es xml, que lo indente, si es json que lo indente, si es binario que muestre la imagen). Se puede mostrar la respuesta cruda en un tab y la respuesta formateada en otro.

Bonus points:

- Poder adicionar parámetros de querystring en una tabla de claves-valores dedicada a ello, logrando que se adosen a la url a medida que se tipean (hint: patrón observer).
- Cuando se completen parámetros de querystring en la tabla se debe actualizar la url. Cuando se toque la url se deben actualizar los valores en las tablas.
- Guardar historial de las urls en las requests hechas recientemente para poder "autocompletar" si se desea repetir alguna.

Tema 4: Cliente FTP

Se debe desarrollar un cliente FTP, con funcionalidad limitada, similar a la del clásico FileZilla. La comunicación con el servidor NO se debe desarrollar de cero, sino que se debe hacer utilizando una librería. La más común es apache-commons-ftp. No es obligatorio implementar el "modo activo" con el modo "pasivo" alcanza.

Funcionalidad básica:

- Campos para introducir parámetros de conexión (host, puerto, user, pass).
- Pantalla dividida en filesystem local y remoto. Listar archivos locales y remotos.
- Botonera en la parte superior para conectar y desconectar.
- Botonera en la parte inferior con las posibles acciones: Subir, bajar, renombrar local, borrar local, borrar remoto.
- Guardar conexiones favoritas en una base de datos para autocompletar url, puerto, usuario y password.

Adicionales:

- Mostrar el sistema de archivos local en forma de árbol.
- Incorporar íconos a la interfaz gráfica.
- Incorporar menues que tengan las mismas funciones que los botones de subir, bajar, etc.

Bonus points:

- Implementar dran'n'drop.
- Adicionar una caja de texto para tirar comandos manualmente en el filesystem local y contra el servidor ftp.

Tema 5: Bug tracker

Se debe realizar un sistema para seguimiento de incidencias, similar al cásico Bugzilla o MantisBT.

Funcionalidad básica:

- Usuarios y perfiles. Se deben poder administrar usuarios, cada uno con diferentes permisos: reportar un issue, cambiarle el estado, indicar el tiempo invertido, cerrarlo.
- Un usuario administrador será el responsable de cargar usuarios y generar proyectos. Los otros usuarios solo pueden trabajar con incidencias.
- Proyectos: se deben manejar N proyectos, cada uno con múltiples incidencias.
- Incidencias: una incidencia debe tener descripción, estimación, tiempo real invertido y estados.

Adicionales:

- Mantener un historial de los estados y los responsables del cambio de cada uno en cada incidencia.
- Reportes sobre cada proyecto: lista de issues en horas estimadas vs. horas invertidas.
- Mostrar proyectos que van atrasados (sus horas invertidas son mayores a las horas estimadas en cada issue) y adelantados. Emitir un reporte que sea una “foto” del proyecto: cuántos issues hay, cuántos en cada estado, cuántas horas invertidas en cada uno. Solo el administrador puede obtener los reportes.

Bonus points:

- Mantener un historial de los estados, de forma cronológica, de cada incidencia y mostrarlos en la pantalla de la incidencia.
- Mantener un historial de los cambios o comentarios, de forma cronológica, de cada incidencia y mostrarlos en la pantalla de la incidencia.

Tema 6: Turnera médica

Funcionalidad básica:

- Administrar médicos. Cada médico es un usuario del sistema, pudiendo consultar los turnos que tiene para una fecha determinada.
- Cada médico cobra cierta cantidad de dinero por su consulta
- Administrar pacientes.
- Administrar turnos, fecha y hora. No se puede tomar un turno con un mismo médico a una misma hora. El médico debe elegirse de una lista, al igual que el paciente.
- Reportes: se debe poder obtener una lista de cuánto ha cobrado un médico, por cuántas consultas (turnos) entre dos fechas.

Adicionales:

- Los pacientes también son usuarios, pudiendo consultar cuándo tienen que asistir a una consulta.
- Administrar lugares de atención (consultorios) con la consecuencia de que un médico debe atender en un cierto lugar entre una y otra fecha, y el turno debe tomarse en un cierto consultorio (es solo una restricción más al turno).
- Reporte adicional: listar los médicos y su recaudación entre dos fechas.

Bonus points:

- Manejar obras sociales. Si un paciente tiene la misma obra social que la que atiende un médico, se le hace un descuento del 50 % en la consulta.
- Mostar un grilla mensual o semanal con turnos (como si se mostrara un calendario).

Tema 7: Administración de proyectos

Se debe realizar un sistema para seguimiento tareas.

Funcionalidad básica:

- Administrar proyectos y empleados asignados a ellos.
- Administrar tareas: cada proyecto tiene tareas. Tiene una cantidad de empleados asignados.
- Cada tarea tiene título descripción estimación y horas reales, entre otros datos.
- Cada tarea está asignada a un empleado. Cada empleado tiene un costo por hora.
- Se deben manejar n proyectos, cada uno con múltiples tareas.

Adicionales:

- Diferentes empleados cobran diferente su hora.
- Manejar un pool de empleados: tener empleados libres y cuando se crea proyecto, hay una pantalla de asignación de empleado<=>proyecto.
- Reportes sobre cada proyecto: costo en horas y costo en dinero de acuerdo con las horas.
- Mantener un historial de los estados de cada tarea y el responsable del cambio mostrándolo al consultar una tarea.

Bonus points:

- Manejar backlog y sprints (las tareas están agrupadas por fecha - el sprint es solo un rango de fechas - las tareas que están en el backlog no tienen sprint, es decir no tienen fecha estimada de inicio ni de fin).
- Mostrar un tablero, estilo kanban, con columnas con cada uno de los estados y en cada columna listar tareas.

Tema 8: Administración de consorcios

Sistema para ayudar a la liquidación de expensas

Funcionalidad básica:

- Una administradora tiene 1 edificio, cada uno con N unidades funcionales, una serie de entradas y una serie de gastos. Se deben administrar las unidades funcionales (cargarlas a un edificio, cambiarle sus datos, como el nombre del ocupante.
- Administrar las entradas y los gastos del edificio: las entradas son los cobros que se le hacen a los copropietarios. Las salidas son gastos (limpieza, encargado, mantenimientos, etc.).
- Además de administrar edificios, unidades funcionales, entradas y salidas, se debe poder hacer la liquidación de expensas de cada edificio: es un listado de cada unidad funcional con lo que debe pagar, seleccionando el mes del año.

Adicionales:

- Soportar diferente cálculo para el cobro de expensas (por ejemplo, a medida que sube el piso, sube el coeficiente de pago - paga más el del piso 20 que el del piso 5, por más que los departamentos tengan igual superficie).
- Sacar un reporte del total de salidas y entradas, para saber la posición consolidada del consorcio entre dos fechas determinadas.

Bonus points:

- Soportar cuenta corriente (guardar las deudas de cada copropietario y hacer la cuenta al momento de sacar el listado de expensas).
- Soportar N consorcios (edificios).

Tema 9: cliente SQL

Se debe desarrollar un cliente gráfico para consultas SQL.

Funcionalidad básica:

- Pantalla dividida: a la derecha mostrar los objetos de la base de datos (tablas, propiedades de cada tabla) en forma de árbol, a la izquierda hay un lugar para escribir texto libre con la consulta y por debajo la lista con los resultados. En la parte superior pueden estar los datos de conexión.
- Botonera para borrar el texto, ejecutar la consulta, conectar, desconectar.
- Mantener una lista de conexiones favoritas (host, puerto, user y password), puede ser en un archivo o en la misma (u otra - h2 basado en file, por ejemplo) base de datos. Al seleccionar una conexión favorita, se autocompletarán los datos de conexión mencionados

Adicionales:

- Doble clic sobre una tabla en la derecha escribe Select * From <tabla_clicekada>.
- Ordenar los resultados clickeando en los headers de la tabla resultado.
- Agregar iconos y hacer de la interfaz algo más agraciado.
- Chequear sintaxis, verificar paréntesis (hint: hay una estructura de datos para hacer esto), palabras claves básicas (mostar una indicación de color en algún lado de la pantalla si el usuario ingreso SELET en vez de SELECT, solo para las palabras claves más básicas, de las otras validaciones se encarga el motor).

Bonus points:

- Soportar múltiples drivers de base de datos, cargar los drivers dinámicamente desde una pantalla de administración de conexiones.
- Paginar los resultados automáticamente si exceden un máximo predeterminado (configurable).

Tema 10: Encuestas

Desarrollar un sistema de administración de encuestas para locales.

Funcionalidad básica:

- Administrar usuarios (administrador: crea encuetas y preguntas, etc., promotor: está encargado de presentar al público las preguntas para que las respondan, no puede hacer más que mostrar una encuesta).
- Poder administrar N encuestas, cada una con M preguntas y posibles respuestas de cada una. Todas las preguntas tienen una serie fija de respuestas (1- muy malo …. 5-muy bueno).
- Pantalla de responder encuesta: debe mostrar todas las preguntas y sus posibles opciones para que un usuario las conteste.
- Registrar las respuestas.
- Se debe mostrar en una pantalla (de un administrador) el resumen de cada encuesta, filtrada por rango de fechas (de cada pregunta de la encuesta, cuántas respuestas hay de 1, cuántas de 2…).

Adicionales:

- Manejar cantidad de opciones variables (de 1 a 5, o de 1 a 3… en cada pregunta, de forma independiente).
- Poder modificar los labels de cada respuesta, de cada pregunta (en vez de ser 1-muy malo, poder poner 1-muy en desacuerdo…).
- Poder manejar junto a las opciones, respuestas “libres” es decir, que la respuesta sea algo que el usuario ingresa como texto.

Bonus points:

- Mostrar un dashboard con los resultados de las respuestas de tipo de opciones, en una pantalla dividida en las últimas 4 encuestas, con los porcentajes de cada respuesta numérica.
- Mostrar gráficos (investigar jfreechart y hacer un gráfico de torta básico).

Tema 11: Sistema de entradas

Se deberá desarrollar un sistema de venta de entradas a espectáculos públicos.

Funcionalidad básica:

- Administrar usuarios: un administrador puede crear espectáculos (o estadios, o predios, etc., con sus datos, capacidad, precio, etc.) y los usuarios vendedores solo pueden registrar la venta.
- Manejar N espectáculos o estadios. Cada estadio tiene diferentes ubicaciones, con diferentes precios cada una.
- Ventas: el vendedor le vende a una persona, una entrada a un espectáculo. El espectáculo se debe poder elegir de una lista. Si el estadio cuenta con más de una ubicación, seleccionar la ubicación de una lista también.
- Mostrar un reporte de cuánto se vendió, por espectáculo, en un rango de fechas,

Adicionales:

- Verificar capacidad, no se pueden vender más entradas que la capacidad.
- Agregar una foto del estadio y una foto de cada ubicación en la pantalla de administración.
- Manejar capacidad por ubicación, verificando capacidad al momento de la venta.

Bonus points:

- Administrar promociones por hora.
- Manejar abonos (si un cliente tiene un abono, sacar una entrada le cuesta cero pesos, o tiene un descuento - prorratear el valor del abono en la venta, para poder emitir el reporte correctamente).

Tema 12: sistema de alumnos

Se debe desarrollar un sistema de administración de cursos y alumnos.

Funcionalidad básica:

- Administrar usuarios: los administradores pueden crear cursos y dar de alta alumnos. Los usuarios profesores pueden ingresar datos sobre el curso o alumno (calificaciones, por ejemplo).
- Cada curso tiene un precio y un cupo.
- Cada alumno debe tener un límite de cursos a los que se puede anotar a la vez. Como cada curso debe aprobarse, se deben manejar calificaciones (solo finales).
- Cada curso debe tener el parámetro de aprobación (nota). Si un alumno se anotó en 3 cursos (suponiendo que 3 es el cupo), debe finalizar y aprobar un curso para poder anotarse en el otro.
- Se debe poder emitir un reporte de los cursos, sus anotados y su recaudación en dinero.
- Mostrar un reporte de la recaudación de los cursos

Adicionales:

- Manejar calificaciones parciales. No se puede incluir una nota final si no se tienen las N parciales aprobados.
- En este caso cada curso tendrá una configuración de cuantas notas parciales son necesarias.
- Manejar promociones por fechas, donde el valor del curso pueda tener un valor reducido entre un rango de fechas.
- Mostrar un reporte de recaudación por curso. Mostrar un reporte de anotados vs aprobados por curso.

Bonus points:

- Manejar abonos (si un alumno tiene un abono, anotarse le cuesta cero pesos - prorratear el valor del abono en la suscripción, para poder emitir el reporte correctamente de recaudación).