Bibliotecas Reunidas

Sistemas de Gestión Empresarial

Grupo B
Belén Marín Sánchez
Daniel Ibáñez Martínez

Fecha de entrega: 15 diciembre 2023

ÍNDICE

1. Entrevista con el cliente

La empresa se llama “Bibliotecas Reunidas”.
En cuanto al diseño el color de la marca es el violeta. Por lo demás no hay ninguna pauta más.

¿Qué información quiere incluir de cada libro, socio…?

- SOCIO:

  - Dar de alta, dar de baja, modificar.
  - Poder consultar los socios que no han devuelto libros → Lista negra.

  datos de contacto → Dirección, tlf, email…
  cuota de 5€ al mes.
  No hace falta ser mayor de edad pero se incluirá info de los padres.
  Se pueden consultar libros de diferentes bibliotecas.

  Puedo coger un libro de Madrid y devolverlo en Bilbao.
  App tiene registro de todos los libros de los socios que van sacando

  Hay penalizaciones. Fuera fecha. Deteriorado.
  No hay sistema de puntos.

  Cada socio va ligado a un id de biblioteca. Un socio solo puede coger un libro en su biblioteca, sin embargo, puede devolverlo en cualquier otra. Este libro no se sumará a la biblioteca donde ha sido devuelto, sino que se retornará a la biblioteca correspondiente).

- LIBROS:

  Tema, título, ISBN, autor, número de libro (es decir identifica el volúmen, el id, idioma)
  Ubicación de los libros → Pasillo x, columna y, estante k…
  Si está cogido o no.

  No se puede reservar los libros. Cuando coges un libro, de esos 3 por ejemplo que hay, marcas qué socio se lleva.

  Se pueden hacer donaciones de libros.
  Idioma en el que está.
  Biblioteca → 5 pasillos con 10 estantes cada una.

  Cuando se entrega un libro NUEVO a una persona hay una sección de comentarios que está en blanco. Si al recibirlo viene maltratado, se pondrá un comentario sobre su estado. Si además viene con un maltrato muy grande se le pondrá una penalización del doble de la cuota (10€).

La app nos tiene que calcular las fechas tope de devolución de los libros. Tiene que quedar almacenado todo el registro de datos. Hay que incluir:

- Fecha de préstamo.
- Fecha prevista de entrega. (esta se tiene que calcular de manera automática).
- Fecha de entrega.

(Poder consultar los libros que no se han devuelto).

- RECIBOS:

Todos los 1 de mes se genera un registro automáticamente con todos los socios. Será en ese registro donde haremos el check de si han pagado o no.

¿Cuántos tipos de préstamos hay? (tiempo)

- 2 semanas sólo. Puedes entregarlo y volverlo a sacar (renovar el plazo) si no tienes 3 libros.

¿Cuántos libros puede sacar cada socio de una vez? ¿Y en total?

- 3 al mes (mes del día 1 al 30). Pueden ser a la vez. Si lo sacas a la vez lo devuelves a la vez.

¿Orden alfabético o ISBN o por autor? Cómo ordenar los libros.

- ISBN.

¿Qué persona va a usar la aplicación? ¿Se necesita log in?

- La app es para los administrativos de la biblioteca.
- El socio no entra en la app.

¿Qué perfiles tenemos que crear? ¿Tienen diferentes permisos?

- Administrativos, que vienen a ser los empleados.
- Administrador. Puede hacer todo. Es el único que crea los usuarios y los perfiles.

¿Puede haber el mismo libro en cada una de las bibliotecas?

- Asocia el número del libro (no es el ISBN, que identifica el título solo, es el tomo/volumen/ejemplar en sí)
- Hay 3 libros por cada ISBN en cada biblioteca.
- Controlar el ISBN
- Puede haber bibliotecas sin ningún volumen.
- Máximo 3 volúmenes de la misma editorial (ISBN) del mismo libro. Es decir, puede haber 3 Quijotes de la misma editorial, 2 de otra y 1 de otra.

¿Se puede hacer una búsqueda por biblioteca?

- Sí. Pero no realiza envíos de una comunidad a otra.

¿Dispone de equipos para el uso de la aplicación? ¿y de un servidor?

- Ordenadores Windows 10, con licencia y conectadas todas las bibliotecas entre sí.

- Servidores Windows 2016

¿Qué pasa en la app si el socio no devuelve a tiempo el/los libro/s?

- Penalización del doble de la cuota. También habrá lista negra de socios.

¿Qué son los recibos? Es el cobro de la cuota de socio. ¿Cuesta dinero hacerse socio?

- 5€ al mes
- Los recibos se tienen que poder imprimir. El cliente puede solicitar el recibo.
- Pagar metálico o transferencia. (El pago no se hace a través de la app).
- Aparece el listado de socios y si han pagado o no. Tener la posibilidad de marcar con un check si esa persona ha pagado.

¿Qué pasa si vamos a añadir un libro y su autor no está previamente creado?

- No te va a dejar.
- Crear autor primero y luego meter sus libros

¿Tienes que ser mayor de edad para ser socio? Qué pasa si no es necesario.

- No hace falta.

PERFILES:

- Administrador: Tiene acceso a toda las app
- Trabajador: no puede modificar socios y etc…

- 10 bibliotecas en total. 1 por provincia.
- Estaría bien incluir una funcionalidad para incorporar nuevas bibliotecas. (Esto no se pide en la práctica pero si queda tiempo se puede considerar incluirlo).

2. Backlog

Decidimos de manera grupal que el desarrollo de todas las unidades funcionales se completará a través de 4 sprints.

- **Tablero inicial del programa y división del trabajo:**

	SPRINT 1	SPRINT 2	SPRINT 3	SPRINT 4
BELÉN	U.F Base de Datos	Log-in Ventana Menú	U.F. Libros	U.F. Usuarios U.F. Socios
DANIEL	U.F Base de Datos	U.F. Prestamos	U.F. Prestamos	U.F. Recibos y multas

- **Especificación de cada tarea del backlog:**

SEMANA 13 - 19 NOVIEMBRE: **Tarea grupal** -------- SPRINT 1

La unidad funcional de la base de datos es desarrollada por ambos integrantes del grupo de manera conjunta.

- Estructurar la base de datos.
- Creación de tablas. Modelo relacional.
- Realización del diagrama Entidad-Relación.
- Establecer la relación entre las tablas y buscar si las claves se propagan o se crean tablas intermedias.
- Creación de la base de datos y de las tablas mediante Xamp y MySQL.
- Realización de los "inserts" en la base de datos.
- Resolución de problemas con caracteres especiales en la base de datos.

SEMANA 20 - 26 NOVIEMBRE: **Tarea individual** -------- SPRINT 2

--BELÉN–

- Creación de un JFrame "ventana" que va a tener un CardLayout en el cual se irán pintando el resto de paneles.

- Unidad funcional Log-In:
  - Creación del JPanel "Login" y diseñar la ubicación de los elementos.
  - Creación del método que comprueba el usuario introducido.
  - Creación del método que comprueba la contraseña introducida.
  - Si el resultado de ambas comprobaciones es satisfactorio se pasará del JPanel "Login" al de "Menu". Si no es así, se mostrará en la aplicación un mensaje de error advirtiendo de que el usuario o la contraseña no son correctos.
  - Se tendrá en cuenta si el usuario es administrador o un empleado, ya que el empleado no tendrá acceso al apartado "usuarios" en el menú y el administrador sí. Se le pasará por parámetros al JPanel "Menu" un booleano que indicará el tipo de perfil del usuario que está accediendo a la aplicación.
  - Además, al realizar la comprobación en la base de datos se obtendrá el "id_biblioteca" con el que está asociado el usuario. Este id se le pasará a todas las ventanas y a casi todos los métodos de base de datos para que todo lo que se muestre en las ventanas esté relacionado con la biblioteca a la que pertenece el usuario.
  - Comprobación de posibles errores, tales como inserción de datos no válidos o la imposibilidad de establecer conexión con la base de datos.

- Ventana Menú:
  - Creación del JPanel "Menu". Diseño y ubicación de los elementos.

- A través de este JPanel se accederá al resto de paneles (libros, socios, préstamos…).
- Comprobación de que la movilidad entre paneles es correcta.
- Se añadirá un botón para cerrar la sesión que nos mandará al panel de “Login”.
- Si el usuario que entra a la aplicación es un administrador, el botón que nos lleva al panel de usuarios será visible. En caso de ser un empleado normal el que entra, este botón no será visible.

--DANIEL–

- Unidad funcional Préstamos:
  - Creación de un JPanel llamado "Hacer_Prestamo_Ventana". Este panel se encargará de gestionar la realización de préstamos en la biblioteca.
  - Creación del constructor Préstamo.
  - Creación de un JComboBox y un método para Socios cuya información es rellenada con los ‘id_socio’ para poder seleccionar aquél desaseado.
  - Creación de un JComboBox y un método para Libros cuya información es rellenada con los ‘id_libro’ para poder seleccionar aquél desaseado.
  - Creación de un JComboBox y un método para Bibliotecas cuya información es rellenada con los ‘id_biblioteca’ para poder seleccionar aquél desaseado.
  - Creación de un TextField para añadir los comentarios que se van a insertar en la base de datos.
  - Añadir el botón y el método para insertar un nuevo préstamo en la base de datos. Este método recoge las variables que contienen la información de los JComboBox y el JTextField y obtiene la fecha actual con formato “yyyy-MM-dd".
  - Creación del método para calcular, con la fecha actual formateada, la fecha correspondiente a dos semanas después de ésta que será el intervalo máximo que el socio tendrá para devolver el libro sin recibir multa.
  - Creación de los métodos para insertar ambas fechas en sus columnas correspondientes de la base de datos.
  - Añadir el botón “Cancelar" que al activarse regresa al usuario a “Libros_Ventana”.
  - Creación del JPanel “Prestamos_Ventana” el cual contendrá un JTable con las columnas de la tabla prestamos para visualizar todos aquellos que estén insertados en ese momento.

SEMANA 27 NOVIEMBRE - 3 DICIEMBRE: **Tarea individual** -------- SPRINT 3

--BELÉN–

- Unidad funcional de Libros:
  - Creación del JPanel “Libros_Ventana”, diseño y colocación de los elementos.

- Creación de la clase “Libros” que se usará para crear objetos de esta clase y se rellenarán sus atributos a través de la información obtenida de la base de datos.
- Creación del método que nos rellena un ArrayList de objetos de la clase Libros en base a lo que nos devuelve la consulta de la base de datos.
- Creación del método que vuelca los objetos del ArrayList en el JTable.
- Creación de cinco botones diferentes dentro del JPanel:
    1. Volver al menú: botón que nos permitirá regresar al menú de opciones.
    2. Editar libro.
    3. Añadir libro.
    4. Borrar libro.
    5. Realizar un préstamo (*U.F Préstamos*).

- **Editar** → Tras haber seleccionado una fila en el JTable, lo cual se comprobará porque de no haber sido así nos aparecerá un error, se abrirá el JDialog de “Editar_Libro”.
  Encontraremos diferentes JTextfield los cuales serán rellenados automáticamente por el contenido del libro seleccionado en la tabla.
  Una vez se quiera guardar la edición del libro se activará un método que comprobará que los campos no estén vacíos y que cumplan con una longitud predeterminada o un sistema de inserción de fechas correcto.

- **Añadir** → Nos aparecerán distintos campos los cuales deberemos rellenar. Después se comprobará que ninguno esté vacío y que cumplan con los requisitos de formato y longitud. Además, se comprobará que el stock para ese título e ISBN no sea actualmente de tres. En caso de ser así, no se nos permitirá insertar dicho libro porque ya habrá un stock de tres. De no ser así, sí que nos dejará añadirlo.

- **Borrar** → Para realizar esta acción estaremos obligados a seleccionar un registro previamente. Después de haberlo hecho se nos preguntará si estamos seguros de eliminarlo y de ser que sí se llamará al método que realice el borrado.

- **Realizar préstamo** → Pertenece a la unidad funcional de préstamos.

--DANIEL–

- <u>Unidad funcional Préstamos:</u>
    - El sprint anterior se consideró fallido dado que se tuvo que reestructurar por completo la ventana de “Hacer_Prestamo”.
    - En la clase “Libros_Ventana” se crea el JButton de ‘**Realizar Préstamo**’. Éste será activado únicamente cuando se haya seleccionado una fila en el JTable de libros.
    - Este botón Invoca al JDialog“Hacer_Prestamo” que contiene tres JTextfields para recoger la información del nombre del socio, el apellido del socio y el correo del socio que el usuario introduzca.

- Creación de un método para confirmar si los datos introducidos en estos JTextfields existen en la base de datos para un socio concreto y, en caso afirmativo, realizar el insert con toda la información, creando así un nuevo préstamo.
- Creación de la columna de stock en el JTable de "Libros_Ventana" donde aparece el número de tomos/ejemplares (es decir, la cantidad de libros que tienen los mismos datos iguales en la base de datos).
- Creación del método para restar una unidad de stock total del libro cuando se realiza un préstamo del libro seleccionado en la fila correspondiente en la tabla.
- Creación del método para aumentar una unidad al stock total del libro correspondiente en la tabla.
- Creación del método para saber cuántos préstamos existentes hay que contengan la misma información de libro, biblioteca y socio para impedir que este último solicite otro tomo/ejemplar de ese mismo libro si hay ausencia de stock en la biblioteca correspondiente.
- Añadir en el JPanel "Prestamo_Ventana" el JButton '**Registrar Devolución**'. que se activa únicamente si se ha seleccionado una fila en el JTable donde aparecen todos aquellos préstamos vigentes en la base de datos, es decir, que no se han cerrado aún.
- Creación de un método para este botón que calcula y verifica la fecha actual. Realiza una comparación entre ésta y la fecha prevista (es decir, aquella que se corresponde a las dos semanas siguientes a la fecha original del préstamo, la cual ya existe en nuestra base de datos desde el momento que se originó el préstamo).
- Creación de un método para insertar una multa (en la tabla multas) con la información del socio correspondiente al préstamo seleccionado si la fecha de devolución sobrepasa las dos semanas estipuladas.
- Creación del método que elimina la fila del préstamo correspondiente en la base de datos una vez se haya ejecutado la acción del botón de la devolución (tenga o no tenga multa el socio).

SEMANA 4 - 10 DICIEMBRE: **Tarea individual** -------- SPRINT 4

--BELÉN–

- Unidad funcional de Socios:
  - Creación de: JPanel "Socios_Ventana", clase "Socios", JDialog "Insertar_Socio" y JDialog "Editar_Socio".
  - Al igual que en la ventana de libros se mostrará una tabla con todos los socios. Tendremos tres botones (Insertar, borrar, editar). Si seleccionamos un usuario nos permitirá borrar y editar, si no seleccionamos nada nos aparecerá un error. Insertar funciona igual que en la unidad funcional de libros. Rellenaremos los campos y al darle a aceptar se llamará a un método que compruebe que los campos nos estén vacíos y que no sobrepasen la longitud máxima de caracteres.
  - Creamos la clase "Recibos". Cuando insertamos o editamos un socio encontraremos un checkbox llamado "lista negra" Si el checkbox está

seleccionado quiere decir que el socio no ha pagado la mensualidad, y si no lo está significa que sí la ha pagado. Cuando se edita un socio que no había pagado anteriormente para desmarcar la casilla de lista negra, se llama a un método que hará un insert en la tabla préstamos de la base de datos para indicar que ya ha pagado.

- Unidad funcional de Usuarios:
    - Creación de: JPanel "Usuarios_Ventana", clase "Usuarios", JDialog "Insertar_Usuario" y JDialog "Editar_Usuario".
    - Contendrá las mismas funcionalidades que las otras unidades funcionales. Lo único que hay que tener en cuenta es que los inserts y updates tendrás que realizarse sobre dos tablas diferentes: usuarios y otros (la cual contiene las contraseñas de los usuarios).

--DANIEL–

- Unidad funcional de Recibos:
    - Creación de un método para comprobar el número de préstamos vigentes que tiene el socio e impedir que pueda realizar otro préstamo si ya posee tres simultáneos.
- Unidad funcional de Recibos:
    - Creación de JPanel "Recibos_Ventana".
    - Creación de un JTable recibos que visualice todos los socios que haya en la biblioteca correspondiente (la del usuario logeado). Esta tabla, con su método correspondiente, interconecta las columnas de socios, multas y recibos para poder visualizar el nombre, apellido y DNI del socio, si está o no multado, y si ha pagado o no la mensualidad.
    - Modificación de la clase constructor de 'Recibos' que pasa a llamarse "InformacionRecibo". Se añade como atributos las columnas de la tabla anterior, así como varios atributos correspondientes con columnas de la tabla biblioteca (que no se visualiza en la tabla). Esto permite recoger toda la información necesaria de la base de datos para crear correctamente el recibo para el/los socio/s de la biblioteca.
    - Creación de los JButtons:
        - **Imprimir recibo seleccionado:** se activa únicamente si se ha seleccionado una fila en la tabla. Llama a un método pasándole como parámetro un boolean true confirmando que es un usuario específico, dado que este método nos servirá también para el siguiente botón descrito a continuación. El método recibe la información del ArrayList del JTable y selecciona justamente la información de la fila seleccionada: id del recibo; nombre, apellido y DNI del socio; calle, provincia, código postal y teléfono de la biblioteca; y multa obtenida. Este último campo nos ayuda a establecer la cantidad de cuota mensual. Si el campo es false(0) la cuota mensual es de 5€, si es true(1) será de 10€.

Este método por último llama a otro método que crea el fichero "Recibo_Socio.txt" con toda la información de los campos anteriormente mencionados del socio seleccionado.

- **Imprimir todos los recibos:** es exactamente igual que el anterior botón, pero con el boolean a false de 'esSocioEspecifico', por lo cual elimina el filtro de la fila seleccionada y recoge todos los objetos tipo InformacionRecibo del ArrayList, es decir, todos los socios de la biblioteca. Crea un fichero "Recibo_Todos.txt" rellenado con la información de cada campo correspondiente a cada objeto (osea cada socio).

- **Confirmar pago recibo**: se activa únicamente si hay una fila seleccionada en la tabla de recibos de esta ventana. Llama a un método para actualizar la columna de pagado a 1 (true) y así visualizar en la columna de pagado del JTable que "sí" se ha realizado el pago de la cuota mensual por parte de ese usuario correspondiente.

3. Creación de la Base de Datos

- **Tablas - Modelo relacional:**

socios
int autoincremental id_socio
int id_biblioteca
varchar(20) nombre_socio
varchar(20) apellido_socio
varchar(9) dni_socio
varchar(50) direccion_socio
varchar(9) tlf_socio
varchar(50) email_socio
boolean lista_negra
date (AAAA-MM-DD) fechaNac_socio

autores
int autoincremental id_autor
varchar(20) nombre_autor
varchar(20) apellido_autor
varchar(20) nacionalidad_autor

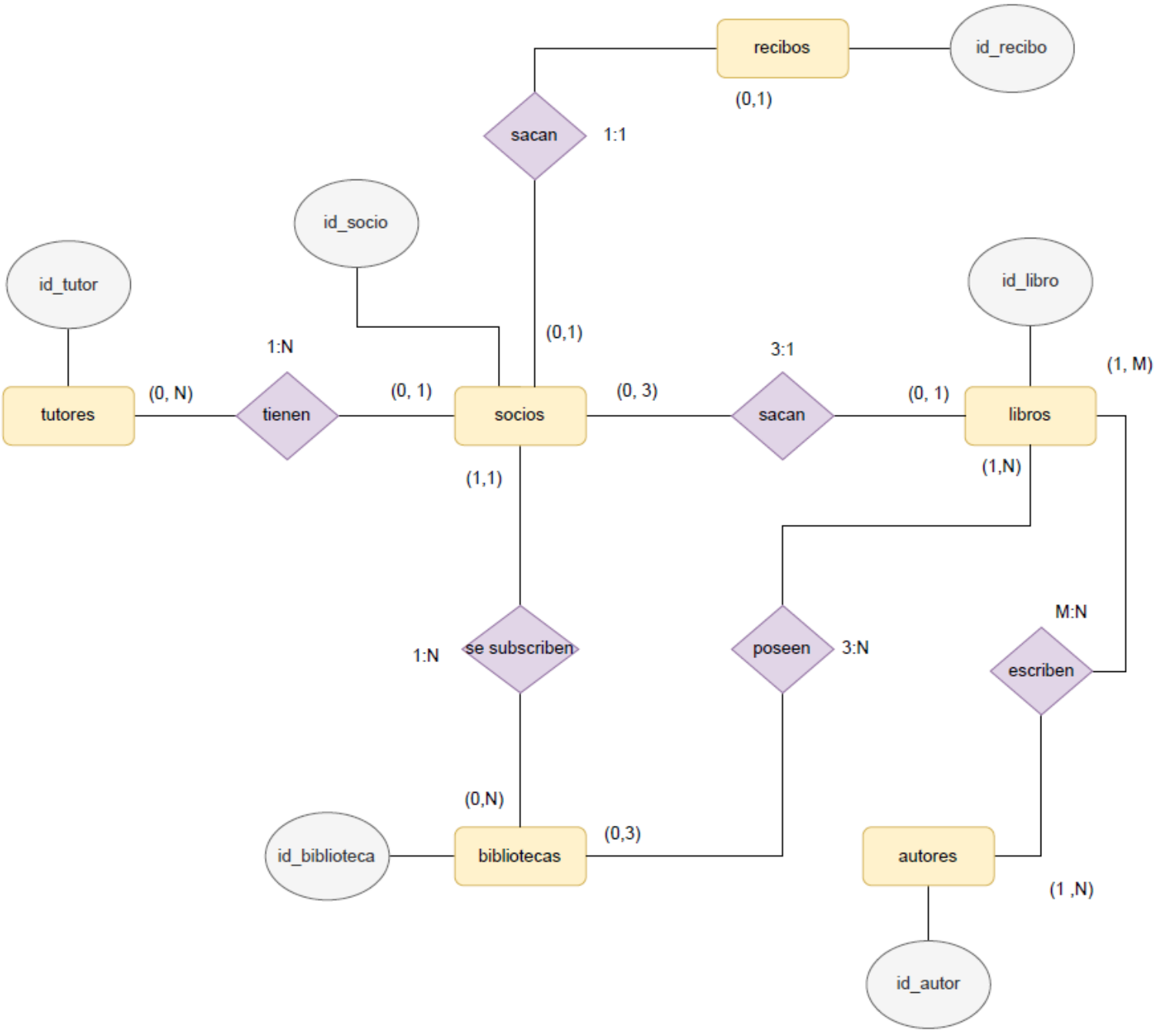
prestamos
int autoincremental id_prestamo
int id_socio
int id_libro
int id_biblioteca
date fecha_prestamo
date fecha_entrega_prevista (automático)
date fecha_entrega

libros
varchar(10) id_libro
int id_biblioteca
varchar(13) isbn_libro
varchar(20) titulo_libro
varchar(100) autores_libro
varchar(20) editorial_libro
varchar (20) género_libro
varchar(20) idioma_libro
varchar(20) edicion_libro
varchar(20) ubicacion_libro
date (AAAA-MM-DD) publicacion_libro
varchar(20) pais_libro
varchar(4) numPaginas_libro

bibliotecas
int autoincremental id_biblioteca
varchar(50) calle_biblioteca
varchar(20) provincia_biblioteca
varchar(9) tlf_biblioteca
varchar(50) email_biblioteca
varchar(5) codigoPostal_biblioteca

usuarios		**recibos**
int autoincremental id_usuario		vint autoincremental id_recibo
int id_biblioteca		int id_socio
varchar(50) email_usuario		int id_biblioteca
		boolean pagado
multas		
int autoincremental id_multa		**otros**
int id_socio		int id_otro (es = a id_usuario)
int id_biblioteca		int id_biblioteca
boolean multa_pagada		varchar(20) material (contraseña)

- **Entidad - Relación:**



4. Estimación de costes y creación de oferta.

- **UF. Base de datos:**

Análisis de los requisitos de la aplicación para llevar a cabo la estructuración de la base de datos. Creación de tablas. Entidad relación. Codificación de la base de datos. Modificaciones en la base de datos → (20 horas x 2 personas) x 35€/h = 1400€

- **UF. Log in:**

Ventana de log-in de codificación sencilla → 5h x 35€/h = 175€

- **Ventana Menú:**

Codificación sencilla → 4h x 35€/h = 140€

- **UF. Socios:**

Ventana consultar, ventana editar, ventana insertar. Botón borrar. Volver al menú. Comprobar campos vacíos y errores de inserción. 3 ventanas x 7 horas aproximadamente cada una = 21h x 35€/h = 735€

- **UF.Libros:**

Ventana consultar, ventana editar, ventana insertar. Botón borrar. Volver al menú. Comprobar campos vacíos y errores de inserción. Que el stock decremente cuando se hace un préstamo o aumente cuando se devuelve. 3 ventanas x 7 horas aproximadamente cada una = 21h x 35€/h = 735€

- **UF. Préstamos:**

Ventana consultar. Ventana realizar préstamo que se abre desde la ventana de libros. Botón cerrar préstamo. 2 ventanas x 15 horas aproximadamente cada una = 30h x 35€/h = 1050€

- **UF. Recibos:**

Ventana consultar. En esta unidad funcional además se tienen en cuenta las multas, es decir, si el socio ha entregado un préstamo fuera de fecha. Imprimir todos los recibos. Imprimir el recibo de un único socio. → 1 ventana x 15h x 35€/h = 525€

- **UF. Usuarios:**

Ventana consultar, ventana editar, ventana insertar. Botón borrar. Volver al menú. Comprobar campos vacíos y errores de inserción. 3 ventanas x +7 horas aproximadamente cada una = 21h x 35€/h = 735€

- Total = 4495€
- Margen de beneficio + colchón por si tardamos más de lo previsto = 50% más del total. → 50% de 4495€ = 2248€

- **Total Final = 6743€**

5. Pasos a seguir para ejecutar la aplicación.

- **Crear la base de datos.**

O bien crearla a través de la consola de Xamp o de PhpMyAdmin. Recomendamos la segunda para que los registros que tengan tildes aparezcan escritos correctamente.

Importar el archivo sql en el gestor de base de datos. Hay que tener en cuenta que antes de importar hay que crear la base de datos:

CREATE DATABASE bibliotecas_reunidas;

- **Ejecutar el programa.**

Se puede abrir el código en Eclipse, NetBeans y ejecutarlo o hacerlo directamente desde el .jar que se encuentra dentro de la carpeta del proyecto.

- **Log-in.**

Existen dos tipos de perfiles en la aplicación:

- Trabajador normal
- Administrador

Si se quiere acceder a todas las funciones (las mismas que tiene el trabajador más la gestión de usuarios) se tendrá que acceder con un perfil de administrador.

Ejemplo:

- emple1B1@gmail.com Emple1 → Perfil de trabajador
- admin1B1@gmail.com Admin1 → Perfil de administrador