Proyecto biblioteca

```
public class App {
    private static final Scanner entrada = new
Scanner(System.in);
    private static final String MENU_PRINCIPAL = """
            MENU PRINCIPAL
    1. Gestionar libros
    2. Gestionar ejemplares
    3. Gestionar usuarios
    4. Gestionar préstamos
    5. Visualizar información
    0. Salir
    """;

    private static final String MENU_LIBROS = """
            MENU LIBROS
    1. Registrar libro
    0. Salir al menu principal
    """;

    private static final String MENU_EJEMPLARES = """
            MENU EJEMPLARES
    1. Registrar ejemplar
    2. Ver stock
    0. Salir al menu principal
    """;

    private static final String MENU_USUARIOS = """
            MENU USUARIOS
    1. Registrar usuario
    0. Salir al menu principal
    """;

    private static final String MENU_PRESTAMOS = """
            MENU PRÉSTAMOS
    1. Registrar préstamo
    2. Registrar devolución
    0. Salir al menu principal
    """;

    public static void main( String[] args ) {
        EntityManagerFactory emf =
Persistence.createEntityManagerFactory("proyc_biblioteca");
        EntityManager em = emf.createEntityManager();
        boolean admin = false;

        System.out.print("Correo:  ");
        String correo = entrada.next();
        System.out.println();

        System.out.print("Contraseña:  ");
        String password = entrada.next();
        System.out.println();

        Query buscarUsuario = em.createQuery("SELECT u FROM
Usuario u WHERE u.email = :email");
        buscarUsuario.setParameter("email", correo);

        try {
            Usuario usuario = (Usuario)
```

```
buscarUsuario.getSingleResult();
            if (usuario.getPassword().equals(password)) {
                if (usuario.getTipo().equals("Administrador"))
{
                    admin = true;
                }
                System.out.println("Bienvenido, " +
usuario.getNombre());
            } else {
                System.out.println("Correo o contraseña
incorrectos.");
                return;
            }
        } catch (Exception e) {
            System.out.println("Correo o contraseña
incorrectos.");
            return;
        }

        int opcion = -1;

        while (opcion != 0) {
            System.out.println(MENU_PRINCIPAL);
            opcion = entrada.nextInt();

            switch (opcion) {
                case 1 -> {
                    while (opcion != 0) {
                        System.out.println(MENU_LIBROS);
                        opcion = entrada.nextInt();

                        switch (opcion) {
                            case 1 -> registrarLibro(em);

                            case 0 ->
System.out.println("Saliendo al menu principal ...");
                        }
                    }

                    opcion = -1;
                }

                case 2 -> {
                    while (opcion != 0) {
                        System.out.println(MENU_EJEMPLARES);
                        opcion = entrada.nextInt();

                        switch (opcion) {
                            case 1 -> registrarEjemplar(em);

                            case 2 -> verStock(em);

                            case 0 ->
System.out.println("Saliendo al menu principal ...");
                        }
                    }

                    opcion = -1;
                }

                case 3 -> {
                    while (opcion != 0) {
                        System.out.println(MENU_USUARIOS);
                        opcion = entrada.nextInt();
```

```
                        switch (opcion) {
                            case 1 -> registrarUsuario(em);

                            case 0 ->
System.out.println("Saliendo al menu principal ...");
                        }
                    }

                    opcion = -1;
                }

                case 4 -> {
                    while (opcion != 0) {
                        System.out.println(MENU_PRESTAMOS);
                        opcion = entrada.nextInt();

                        switch (opcion) {
                            case 1 -> prestarLibro(em);

                            case 2 -> devolverLibro(em);

                            case 0 ->
System.out.println("Saliendo al menu principal ...");
                        }
                    }

                    opcion = -1;
                }

                case 5 -> verInfo(admin, em, correo);

                case 0 -> System.out.println("Saliendo ...");
            }

        }

        em.close();
        emf.close();
    }

    private static void verInfo(boolean admin, EntityManager
em, String correo) {
        List<Prestamo> prestamos;
        Query query;

        if (admin) {
            query = em.createQuery("SELECT prestamo FROM
Prestamo prestamo");

        } else {
            query = em.createQuery(
                    "SELECT prestamo FROM Prestamo prestamo " +
                    "WHERE prestamo.usuario.email = :email"
            );
            query.setParameter("email", correo);
        }

        prestamos = query.getResultList();

        for (Prestamo p : prestamos) {
            System.out.println(
                    p.getUsuario().getNombre() + " | " +
                    p.getEjemplar().getIsbn().getTitulo() + " |
```

```
" +
                    p.getFechaInicio() + " | " +
                    p.getFechaDevolucion()
            );
        }
    }

    private static void devolverLibro(EntityManager em) {
        Usuario usuario = pedirUsuario(em);
        if (usuario == null) return;

        System.out.println("Indica el isbn del libro a
devolver: ");
        String isbn = entrada.next();
        System.out.println();

        Query buscarEjemplar = em.createQuery(
                "SELECT p.ejemplar FROM Prestamo p " +
                "WHERE p.ejemplar.isbn.isbn = :isbn " +
                "AND p.usuario.id = :usuario " +
                "AND p.fechaDevolucion IS NULL"
        );
        buscarEjemplar.setParameter("isbn", isbn);
        buscarEjemplar.setParameter("usuario",
usuario.getId());
        Ejemplar ejemplar;

        try {
            ejemplar = (Ejemplar)
buscarEjemplar.getSingleResult();
        } catch (NoResultException e) {
            System.out.println("El ejemplar no existe o no esta
prestado.");
            return;
        }

        Query buscarPrestamo = em.createQuery(
             "SELECT p FROM Prestamo p " +
                "WHERE p.usuario.id = :id " +
                "AND p.ejemplar.id = :ejemplarId " +
                "AND p.fechaDevolucion IS NULL"
        );
        buscarPrestamo.setParameter("id", usuario.getId());
        buscarPrestamo.setParameter("ejemplarId",
ejemplar.getId());
        Prestamo prestamo;

        try {
            prestamo = (Prestamo)
buscarPrestamo.getSingleResult();
        } catch (NoResultException e) {
            System.out.println("El ejemplar no esta
prestado.");
            return;
        }

        em.getTransaction().begin();
        if (prestamo.getFechaLimite() == null)
            prestamo.setFechaLimite(prestamo.getFechaInicio());

        if
(prestamo.getFechaLimite().isBefore(LocalDate.now())) {
            System.out.println("Este libro es entregado con
retraso, se pondrá una amonestación de 15 días.");
```

```
            usuario.setPenalizacionHasta(LocalDate.now());
        }

        prestamo.setFechaDevolucion(LocalDate.now());
        ejemplar.setEstado("Disponible");
        em.getTransaction().commit();
    }

    private static void prestarLibro(EntityManager em) {
        Usuario usuario = pedirUsuario(em);
        if (usuario == null) return;

        Query prestamosDeUsuario = em.createQuery(
                "SELECT COUNT(prestamo) FROM Prestamo prestamo"
+
                " WHERE prestamo.usuario.dni = :dni " +
                "AND prestamo.fechaDevolucion IS NULL"
        );
        prestamosDeUsuario.setParameter("dni",
usuario.getDni());

        Long prestamos = (Long)
prestamosDeUsuario.getSingleResult();
        if (prestamos >= 3) {
            System.out.println("El usuario ya tiene 3 libros
prestados, no se pueden prestar mas libros.");
            return;
        }

        if (
                usuario.getPenalizacionHasta() != null &&

usuario.getPenalizacionHasta().isAfter(LocalDate.now())
        ) {
            System.out.println("El usuario tiene una
penalización hasta: " + usuario.getPenalizacionHasta());
            return;
        }

        System.out.println("Indica ISBN: ");
        String isbn = entrada.next();
        System.out.println();

        Query ejemplarDisponible = em.createQuery(
                "SELECT e FROM Ejemplar e " +
                "WHERE e.isbn.isbn = :isbn " +
                "AND e.estado = 'Disponible'"
        );
        ejemplarDisponible.setParameter("isbn", isbn);
        Ejemplar ejemplar;

        try {
            ejemplar = (Ejemplar)
ejemplarDisponible.getSingleResult();
        } catch (NoResultException e) {
            System.out.println("El ejemplar no existe o no esta
disponible.");
            return;
        }

        Prestamo p = new Prestamo(usuario, ejemplar,
LocalDate.now());
        em.getTransaction().begin();
        em.persist(p);
```

```
        ejemplar.setEstado("Prestado");
        em.getTransaction().commit();
    }

    private static void registrarUsuario(EntityManager em) {
        System.out.print("Indica el DNI: ");
        String dni = entrada.next();
        System.out.println();

        System.out.print("Indica el nombre: ");
        String nombre = entrada.next();
        System.out.println();

        System.out.print("Indica el email: ");
        String email = entrada.next();
        System.out.println();

        System.out.print("Indica contrasenia: ");
        String contrasenia = entrada.next();
        System.out.println();

        System.out.println("Indica el tipo de usuario: ");
        String tipo = entrada.next();
        System.out.println();

        Usuario usuario = new Usuario(dni, nombre, email,
contrasenia, tipo);

        em.getTransaction().begin();
        em.persist(usuario);
        em.getTransaction().commit();
    }

    private static void verStock(EntityManager em) {
        System.out.print("Indica el isbn del libro a buscar:
");
        String isbn = verificarISBN();
        System.out.println();

        Query contarDisponibles = em.createQuery(
                "SELECT COUNT(ej.estado) FROM Ejemplar ej " +
                "WHERE ej.isbn.isbn = :isbn " +
                "AND ej.estado = 'Disponible'"
        );
        contarDisponibles.setParameter("isbn", isbn);

        Long stock = (Long)
contarDisponibles.getSingleResult();

        System.out.println(stock);
    }

    private static void registrarEjemplar(EntityManager em) {
        System.out.print("Indica el ISBN (sin guiones): ");
        String isbn = verificarISBN();
        System.out.println();

        System.out.print("Indica el estado del ejemplar
(Disponible; Prestado; Dañado): ");
        String estado = entrada.next();

        System.out.println();

        Libro libro = em.find(Libro.class, isbn);
```

```
        if (libro == null) {
            System.out.println("El libro no existe.");
            return;
        }
        Ejemplar ejemplar = new Ejemplar(libro, estado);

        em.getTransaction().begin();
        em.persist(ejemplar);
        em.getTransaction().commit();
    }

    private static void registrarLibro(EntityManager em) {
        System.out.print("Indica el ISBN (sin guiones): ");
        String isbn = verificarISBN();
        System.out.println();

        System.out.print("Indica el título");
        entrada.nextLine();
        String titulo = entrada.nextLine();
        System.out.println();

        System.out.print("Indica el autor");
        String autor = entrada.next();
        System.out.println();

        Libro libro = new Libro(isbn, titulo, autor);

        em.getTransaction().begin();
        em.persist(libro);
        em.getTransaction().commit();
    }

    private static Usuario pedirUsuario(EntityManager em) {
        System.out.print("Indica dni del usuario: ");
        String dni = entrada.next();
        System.out.println();

        Query buscarUsuario = em.createQuery("SELECT usuario
FROM Usuario usuario WHERE usuario.dni = :dni");
        buscarUsuario.setParameter("dni", dni);
        Usuario usuario;

        try {
            usuario = (Usuario)
buscarUsuario.getSingleResult();
        } catch (NoResultException e) {
            System.out.println("El usuario no existe.");
            return null;
        }

        return usuario;
    }

    private static String verificarISBN() {
        String isbn = entrada.next();

        while (isbn.length() != 13 && isbn.length() != 10) {
            System.out.println("Formato de código invalido. ");
            System.out.print("Indica el ISBN (sin guiones): ");
            isbn = entrada.next();
            System.out.println();
        }

        return isbn;
```

```
    }
}
```

Al inicio hay un inicio de sesión para saber si eres un usuario normal o un admin, si las credenciales son incorrectas se sale del programa diciendo que la contraseña o usuario son incorrectas

```
Correo: a

Contraseña: a

Hibernate: select u1_0.id,u1_0.dni,u1_0.email,u1_0.nombre,u1_0.password,u1_0.penalizacionHasta,u1_0.tipo from usuario u1_0 where u1_0.email=?
Correo o contraseña incorrectos.

Process finished with exit code 0
```

Una vez inicies sesión, hay un menú principal en el que te salen las opciones de submenús o acciones a tomar

```
Hibernate: select u1_0.id,u1_0.dni,u1_0.email,u1_0.nombre,u1_0.password,u1_0.penalizacionHasta,u1_0.tipo from usuario u1_0 where u1_0.email=?
Bienvenido, Juan Pérez
        MENU PRINCIPAL
1. Gestionar libros
2. Gestionar ejemplares
3. Gestionar usuarios
4. Gestionar préstamos
5. Visualizar información
0. Salir
```

MENU LIBROS

Solo se puede registrar un libro o salir al menú principal, abriendo distintas posibilidades en el futuro

```
        MENU LIBROS
1. Registrar libro
0. Salir al menu principal
```

MENU EJEMPLARES

Registras ejemplares y ves el stock del libro que indiques mediante ISBN

```
        MENU EJEMPLARES
1. Registrar ejemplar
2. Ver stock
0. Salir al menu principal
```

```
Indica el isbn del libro a buscar: 9781234567890

Hibernate: select count(e1_0.estado) from ejemplar e1_0 where e1_0.isbn=? and e1_0.estado='Disponible'
2
```

MENU USUARIOS

Registras usuarios

```
        MENU USUARIOS
1. Registrar usuario
0. Salir al menu principal
```

MENU PRESTAMOS

Registras prestamos y devoluciones del usuario, indicando su dni y el ISBN del libro a prestar / devolver, dentro de cada opción hace sus revisiones para confirmar que tanto el libro como el usuario existen, que el libro este prestado y que el usuario no pueda pedir préstamos ya que está penalizado o tiene 3 préstamos, y , que si devuelve tarde, se le penalice.

```
        MENU PRÉSTAMOS
 1. Registrar préstamo
 2. Registrar devolución
 0. Salir al menu principal
```

VER INFO

Dependiendo de si eres admin, o no, ves una cosa u otra.

Si eres un usuario normal solo ves tus prestamos

```
3
Hibernate: select p1_0.id,p1_0.ejemplar_id,p1_0.fechaDevolucion,p1_0.fechaInicio,p1_0.fechaLimite,p1_0.usuario_id from prestamo p1_0 join usuario u1_0 on
 u1_0.id=p1_0.usuario_id where u1_0.email=?
Hibernate: select e1_0.id,e1_0.estado,e1_0.isbn from ejemplar e1_0 where e1_0.id=?
Hibernate: select l1_0.isbn,l1_0.autor,l1_0.titulo from libro l1_0 where l1_0.isbn=?
Juan Pérez | El Quijote | 2024-11-01 | null
       MENU PRINCIPAL
```

Si eres admin ves todos los prestamos

```
3
Hibernate: select p1_0.id,p1_0.ejemplar_id,p1_0.fechaDevolucion,p1_0.fechaInicio,p1_0.fechaLimite,p1_0.usuario_id from prestamo p1_0
Hibernate: select u1_0.id,u1_0.dni,u1_0.email,u1_0.nombre,u1_0.password,u1_0.penalizacionHasta,u1_0.tipo from usuario u1_0 where u1_0.id=?
Hibernate: select e1_0.id,e1_0.estado,e1_0.isbn from ejemplar e1_0 where e1_0.id=?
Hibernate: select l1_0.isbn,l1_0.autor,l1_0.titulo from libro l1_0 where l1_0.isbn=?
Juan Pérez | El Quijote | 2024-11-01 | null
Hibernate: select u1_0.id,u1_0.dni,u1_0.email,u1_0.nombre,u1_0.password,u1_0.penalizacionHasta,u1_0.tipo from usuario u1_0 where u1_0.id=?
Hibernate: select e1_0.id,e1_0.estado,e1_0.isbn from ejemplar e1_0 where e1_0.id=?
Hibernate: select l1_0.isbn,l1_0.autor,l1_0.titulo from libro l1_0 where l1_0.isbn=?
Ana García | Cien Años de Soledad | 2024-11-05 | 2024-11-20
       MENU PRINCIPAL
```

ENLACE REPOSITORIO: https://github.com/Samuhdez17/Proyecto-Biblioteca