Requisitos No Funcionales:

RNF PLSQL-1: Subdivisión de paquetes.

Se deberán crear los siguientes paquetes:

- BASE: contendrá las funciones/procedimientos básicos relacionados con el tratamiento básico del esquema de LIFEFIT
- ICALC: contendrá las funciones/procedimientos relacionados con el tratamiento de las reglas RRULE relacionadas con la disponibilidad y las citas.

RNF PLSQL-2: Sobre tipos, funciones y procedimientos.

Todas las funciones, procedimientos y tipos visibles deberán cumplir los siguientes requisitos:

- Todo tipo, función o procedimiento deberá estar definido dentro de un paquete.
- Toda función deberá ser una función pura (como se entiende en programación funcional) y no tener ningún efecto colateral. Esto es: todos los argumentos deben ser de entrada, no puede realizar modificaciones sobre la base de datos (insert/update/delete/commit/savepoint/rollback/create/alter/drop/… estarán completamente prohibidos).
- Únicamente los procedimientos están autorizados a tener efectos colaterales, así como múltiples parámetros de salida.
- Si un procedimiento puede ser implementado como una función (con las características antes indicadas) deberá ser implementado como función.

RNF PLSQL-3: Sobre la gestión de transacciones.

Relacionado con la gestión de transacciones se observarán las siguientes consideraciones:

- Las funciones al no poder tener ningún efecto colateral no podrán realizar commit/rollback.
- Si un procedimiento se especifica como “autoconfirmado”, entonces el procedimiento realizará un commit (bien explícito o implícito) al final de su ejecución en caso de no haberse producido ningún error.
- Si no se especifica nada, el comportamiento normal de todo procedimiento que termine sin ningún error será dejar la transacción en curso sin confirmar.
- El comportamiento de todo procedimiento (ya sea autoconfirmado o no) ante un error será deshacer única y exclusivamente las modificaciones realizadas por él mismo. Para ello todo procedimiento establecerá un SAVEPOINT al inicio de su ejecución y, caso de producirse un error, realizará un ROLLBACK TO SAVEPOINT.

Este requisito pretende mantener el principio de responsabilidad, intentando que un procedimiento únicamente pueda cancelar o confirmar aquellas modificaciones realizadas por él mismo directa o indirectamente. La única excepción son los procedimientos confirmados que

podrán confirmar todas las modificaciones previas a su llamada, pero no podrá deshacerlas en caso de error.

RNF PLSQL-4: Excepciones.

Se definirán como mínimo las siguientes excepciones:

- EXCEPCION_CREACION: será lanzada ante un error en cualquier funcionalidad de creación (operación de create en un CRUD).
- EXCEPCION_MODIFICACION: será lanzada ante un error en cualquier funcionalidad de modificación (operación de update en un CRUD).
- EXCEPCION_ELIMINACION: será lanzada ante un error en cualquier funcionalidad de eliminación (operación de delete en un CRUD).
- EXCEPCION_LECTURA: será lanzada ante un error en cualquier funcionalidad de lectura (operación read de un CRUD).

Requisitos Funcionales:

RF PLSQL-5: Creación de clientes.

El procedimiento de creación de clientes seguirá el siguiente esquema (únicamente pueden cambiarse los nombres de columnas si difieren):

```
TYPE TCLIENTE IS RECORD (
        NOMBRE       USUARIO.NOMBRE%TYPE,
        APELLIDOS    USUARIO.APELLIDOS%TYPE,
        TELEFONO     USUARIO.TELEFONO%TYPE,
        DIRECCION    USUARIO.DIRECCION%TYPE,
        CORREOE      USUARIO.CORREOE%TYPE,
        OBJETIVOS    CLIENTE.OBJETIVOS%TYPE,
        DIETA        CLIENTE.DIETA%TYPE,
        PREFERENCIAS CLIENTE.PREFERENCIAS%TYPE,
        CENTRO       CLIENTE.CENTRO%TYPE
        );

PROCEDURE CREA_CLIENTE(
        P_DATOS    IN  TCLIENTE,
        P_USERPASS IN  VARCHAR2,
        P_USUARIO  OUT USUARIO%ROWTYPE,
        P_CLIENTE  OUT CLIENTE%ROWTYPE
    );
```

Este procedimiento será autoconfirmado. Además de insertar el usuario y el cliente correspondientes en sus tablas respectivas, deberá crear igualmente el usuario en la base de datos, asignar permisos correspondientes y crear sinónimos, vistas y demás objetos relacionados con el nuevo usuario. Para cumplir RNF PLSQL-3, este procedimiento deberá hacer uso de SAVEPOINT, ROLLBACK TO SAVEPOINT y procedimientos auxiliares para hacer uso de las directivas PRAGMA AUTONOMOUS_TRANSACTION. En los argumentos de salida deberán estar todos los valores correspondientes a las filas creadas (incluyendo los datos de las columnas que se han creado automáticamente, como el ID del cliente y el nombre del UsarioOracle)

RF PLSQL-6: Creación de entrenadores.

Siguen las mismas consideraciones que la creación de clientes, pero con el siguiente esquema (únicamente pueden cambiarse los nombres de columnas si difieren):

```
TYPE TENTRENADOR IS RECORD (
        NOMBRE          USUARIO.NOMBRE%TYPE,
        APELLIDOS       USUARIO.APELLIDOS%TYPE,
        TELEFONO        USUARIO.TELEFONO%TYPE,
        DIRECCION       USUARIO.DIRECCION%TYPE,
        CORREOE         USUARIO.CORREOE%TYPE,
        DISPONIBILIDAD ENTRENADOR.DISPONIBILIDAD%TYPE,
        CENTRO          ENTRENADOR.CENTRO%TYPE
        );

PROCEDURE CREA_ENTRENADOR(
        P_DATOS      IN  TENTRENADOR,
        P_USERPASS   IN  VARCHAR2,
        P_USUARIO    OUT USUARIO%ROWTYPE,
        P_ENTRENADOR OUT ENTRENADOR%ROWTYPE
    );
```

RF PLSQL-7: Creación de gerentes.

Siguen las mismas consideraciones que la creación de clientes, pero con el siguiente esquema (únicamente pueden cambiarse los nombres de columnas si difieren):

```
TYPE TGERENTE IS RECORD (
        NOMBRE    USUARIO.NOMBRE%TYPE,
        APELLIDOS USUARIO.APELLIDOS%TYPE,
        TELEFONO  USUARIO.TELEFONO%TYPE,
        DIRECCION USUARIO.DIRECCION%TYPE,
        CORREOE   USUARIO.CORREOE%TYPE,
        DESPACHO  GERENTE.DESPACHO%TYPE,
        HORARIO   GERENTE.HORARIO%TYPE,
        CENTRO    GERENTE.CENTRO%TYPE
        );

PROCEDURE CREA_GERENTE(
        P_DATOS    IN  TCLIENTE,
        P_USERPASS IN  VARCHAR2,
        P_USUARIO  OUT USUARIO%ROWTYPE,
        P_GERENTE  OUT GERENTE%ROWTYPE
    );
```

RF PLSQL-8: Eliminación de usuarios.

Tendremos un procedimiento autoconfirmado con el siguiente esquema:

```
PROCEDURE ELIMINA_USER(P_ID USUARIO.ID%TYPE);
```

Este procedimiento eliminará toda la infraestructura creada para la conexión del "UsuarioOracle". Pondrá a null el atributo UsuarioOracle de la fila correspondiente de la tabla USUARIO. Pero dejará intacta la información de dicho usuario en el esquema.

Este procedimiento se utilizará cuando un usuario finalice su relación con LIFEFIT. De esta forma eliminamos toda infraestructura relacionada con su conectividad pero mantenemos su información en la base de datos.

RF PLSQL-9: Eliminación de clientes, gerentes y entrenadores.

Tendremos varios procedimientos (todos autoconfirmados) con el siguiente esquema:

```
PROCEDURE ELIMINA_CLIENTE(P_ID USUARIO.ID%TYPE);
PROCEDURE ELIMINA_GERENTE(P_ID USUARIO.ID%TYPE);
PROCEDURE ELIMINA_ENTRENADOR(P_ID USUARIO.ID%TYPE);
```

Este procedimiento eliminará, además de toda la infraestructura creada para la conexión del "UsuarioOracle", toda la información de la base de datos relacionada con dicho cliente/gerente/usuario. Ello implica la eliminación de cualquier información relativa a eventos de calendarios, citas, entrenamientos, planes y/o sesiones.

Este procedimiento se utilizará para eliminar completamente de nuestra base de datos toda información relacionada con un usuario. Generalmente se usará para eliminar usuarios de prueba.

RF PLSQL-10: Eliminación de centros.

Procedimiento autoconfirmado con el siguiente esquema:

```
PROCEDURE ELIMINA_CENTRO(P_ID CENTRO.ID%TYPE);
```

Este procedimiento eliminará toda la información relacionada con el centro. Si no se mantiene RNF PLSQL-3 es tan simple como eliminar todos sus gerentes/entrenadores/clientes. Pero mantener RNF PLSQL-3 con este procedimiento es especialmente complejo, ya que impide usar las funciones creadas anteriormente de eliminación de gerentes/entrenadores/clientes, ya que si las usamos y falla la destrucción de un usuario X habría que deshacer la destrucción destruidos antes.

Nota: Este procedimiento tiene varios niveles de complejidad, obteniéndose más puntuación según se implemente la versión más compleja:

- Nivel 1: Eliminación sin mantener RNF PLSQL-3.
- Nivel 2: Eliminar información, infraestructura y si falla la eliminación de la infraestructura recomponer la información y dejar la infraestructura no íntegra. Dejando la transacción original intacta.
- Nivel 3: Eliminar información, infraestructura y si falla la eliminación de la infraestructura recomponer la información y la infraestructura. Dejando la transacción original intacta.

RF PLSQL-11: Generación de elementos de calendario.

Tendremos un procedimiento con el siguiente esquema:

```
PROCEDURE CREA_ELEMENTOS(
    P_ID   IN ENTRENADOR.ID%TYPE,
    P_ANNO IN NUMBER,
    P_MES  IN NUMBER
);
```

Este procedimiento procederá a generar los elementos de calendario de la tabla ElementoCalendario para un entrenador para el año/mes especificado.

El campo DISPONIBILIDAD de un entrenador tiene el siguiente formato (además asumiremos que jamás contendrá ningún error de formato, ni caracteres inútiles o separadores no autorizados, como espacios, tabuladores, retornos de carro, etc.):

```
regla|regla|regla|…|regla
```

Donde "regla" tiene el formato de una RRULE específica de iCalc RDF 5545 y en caso de múltiples reglas se separan con el carácter '|'. Exclusivamente tendrá el formato:

```
RRULE:FREQ=WEEKLY;BYDAY=días;BYHOUR=horas;BYMINUTE=minutos;
```

Horas y minutos se corresponderá con uno o varios números separados por coma indicando.

Días contendrá un día o una lista de días separada por coma. Cada día se identifica siguiendo con las dos primeras letras del nombre del día en inglés.

Así un ejemplo de campo disponibilidad válido sería (separamos cada regla en líneas diferentes por razones de legibilidad).

```
RRULE:FREQ=WEEKLY;BYDAY=MO,WE;BYHOUR=9,10,11,12,13;BYMINUTE=0,15,30,45|
RRULE:FREQ=WEEKLY;BYDAY=TU,TH;BYHOUR=17,18,19,20;BYMINUTE=0,20,40|
RRULE:FREQ=WEEKLY;BYDAY=FR;BYHOUR=18,19,20,21;BYMINUTE=0,30|
RRULE:FREQ=WEEKLY;BYDAY=SA;BYHOUR=9,10;BYMINUTE=0|
RRULE:FREQ=WEEKLY;BYDAY=SA;BYHOUR=11;BYMINUTE=35
```

Respectivamente cada RRULE de esta disponibilidad genera elementos:

- Primera regla: lunes y miércoles a las 9:00, 9:30, 9:45, 10:00… 13:30, 13:45.
- Segunda regla: martes y jueves a las 17:00, 17:20, 17:40, 18:00… 20:30, 20:45.
- Tercera regla: viernes a las 18:00, 18:30, 19:00… 21:00, 21:30.
- Cuarta regla: sábado a las 9:00 y a las 10:00.
- Quinta regla: sábado a las 11:35.

Si al intentar crear un elemento de calendario se produce un error debido a que ya existe dicho elemento de calendario deberemos ignorar dicho error y proseguir generando los elementos. De esta forma si ejecutamos dos veces consecutivas la misma llamada, la primera generaría los elementos de calendario que faltasen y la segunda no generaría nada (ya que todos los elementos existirían). En ambos casos no se elevaría ningún tipo de error.

Cualquier otro motivo de error al generar un elemento de calendario elevará la correspondiente EXCEPCION_CREACION. Este procedimiento debería seguir todos los demás requisitos, RNF PLSQL-3 incluido.

RF PLSQL-12: Job creación de elementos de calendario.

Crear un job de Oracle que cree los eventos de calendario de todos los entrenadores para el mes siguiente el día 20 de cada mes a las 01:00 de la madrugada.