PROYECTO FINAL
BDD empleada para administrar el negocio de la empresa NetMAX – PARTE 3

1.1. TRANSPARENCIA PARA INSTRUCCIÓN SELECT

El siguiente paso a desarrollar es la implementación de transparencia. Para efectos del proyecto, se realizará únicamente transparencia para las instrucciones `select`, `insert` y `delete`.

1.1.1. Creación de sinónimos.

- Crear un archivo por sitio llamado `s-04-netmax-`**`<pdb>`**`-sinonimos.sql`. Ejemplo: `s-04-netmax-`**`jrc-s1`**`-sinonimos.sql`
- Incluir los sinónimos requeridos por PDB los cuales serán empleados para implementar transparencia de localización.
- El script deberá conectarse a la PDB y crear los sinónimos correspondientes.
- Observar en el siguiente ejemplo la convención a emplear: `<nombre_global>_f<n>` Donde N es el número de fragmento. A partir de este punto se oculta la información de ubicación de los fragmentos. Por ejemplo, se oculta `jrc_s1, jrc_s2`, etc.
- Observar que en también se crean sinónimos para fragmentos locales. Por ejemplo: para el nodo `jrcbd_s1` se crea el sinónimo `usuario_f1` a partir de `usuario_f1_jrc_s1`

Ejemplo:

Sinónimos para `jrcbd_s1`.

```
--@Autor:          Jorge A. Rodríguez C
--@Fecha creación:  dd/mm/yyyy
--@Descripción:     Creacion de sinonimos para jrcbd_s1

--USUARIO
create or replace synonym USUARIO_F1 for USUARIO_F1_JRC_S1;
create or replace synonym USUARIO_F2 for USUARIO_F2_ARC_S1@arcbd_s1.fi.unam;
create or replace synonym USUARIO_F3 for USUARIO_F3_JRC_S2@jrcbd_s2.fi.unam;
create or replace synonym USUARIO_F4 for USUARIO_F4_ARC_S2@arcbd_s2.fi.unam;
create or replace synonym USUARIO_F5 for USUARIO_F5_ARC_S1@arcbd_s1.fi.unam;

--completar
```

1.1.1.1. Validación de sinónimos

- Para verificar que los sinónimos y fragmentos han sido creados correctamente en sus correspondientes sitios, crear un solo script llamado `s-04-netmax-valida-sinonimos.sql` El script mostrará un conteo del número de registros existente principalmente para validar que el sinónimo esté creado correctamente.
- Cabe mencionar que el manejador no verifica si la tabla realmente existe durante la creación del sinónimo, por lo que este script será útil para detectar posibles errores.

Ejemplo:

```
--@Autor:          Jorge A. Rodríguez C
--@Fecha creación:  dd/mm/yyyy
--@Descripción:     Script de validacion de sinonimos

Prompt validando sinonimos para USUARIO
select
 (select count(*) from USUARIO_F1) as usuario_f1,
 (select count(*) from USUARIO_F2) as usuario_f2,
 (select count(*) from USUARIO_F3) as usuario_f3,
 (select count(*) from USUARIO_F4) as usuario_f4,
 (select count(*) from USUARIO_F5) as usuario_f5
from dual;

Prompt validando sinonimos para PLAYLIST

-- completar
```

1.1.1.2. Ejecución de script para sinónimos.

- Crear un solo script llamado `s-04-netmax-main-sinonimos.sql` El script se deberá conectar a cada PDB y ejecutar los scripts creados anteriormente. Notar que el script de validación se ejecuta en cada una de las PDBs para confirmar su correcta creación.

```
--@Autor:          Jorge A. Rodríguez C
--@Fecha creación:
--@Descripción:     Creación de  sinónimos - main
clear screen
whenever sqlerror exit rollback;

prompt ====================================
prompt Creando sinonimos para jrcbd_s1
prompt ====================================
connect netmax_bdd/netmax_bdd@jrcbd_s1
@s-04-netmax-jrc-s1-sinonimos.sql
@s-04-netmax-valida-sinonimos.sql

prompt ====================================
prompt creando sinonimos para jrcbd_s2
prompt ====================================
connect netmax_bdd/netmax_bdd@jrcbd_s2
@s-04-netmax-jrc-s2-sinonimos.sql
@s-04-netmax-valida-sinonimos.sql

prompt ====================================
prompt creando sinonimos para arcbd_s1
prompt ====================================
connect netmax_bdd/netmax_bdd@arcbd_s1
@s-04-netmax-arc-s1-sinonimos.sql
@s-04-netmax-valida-sinonimos.sql

prompt ====================================
prompt creando sinonimos para arcbd_s2
prompt ====================================
connect netmax_bdd/netmax_bdd@arcbd_s2
@s-04-netmax-arc-s2-sinonimos.sql
@s-04-netmax-valida-sinonimos.sql

prompt Listo!
```

1.1.2. Creación de vistas.

1.1.2.1. Vistas para tablas globales sin columnas BLOB/CLOB

- Crear un solo script llamado `s-05-netmax-vistas.sql.` El script contendrá la definición de las vistas que se van a crear en cada sitio.
- ***No incluir*** en este script las vistas que correspondan a tablas con datos BLOB/CLOB. Esto ***sin importar*** el sitio en el que se encuentre. Las vistas para estas tablas se crearán en la siguiente sección.

Ejemplo:

```
--@Autor:          Jorge A. Rodríguez C
--@Fecha creación:
--@Descripción:     Creación de vistas comunes a todos los nodos
--                  Se excluyen las vistas que contienen columnas BLOB

--PLAYLIST
Prompt creando vista PLAYLIST
create or replace view PLAYLIST as
 select playlist_id,calificacion,indice,num_reproducciones,programa_id,
  usuario_id
 from playlist_f1
 union all
 select playlist_id,calificacion,indice,num_reproducciones,programa_id,
  usuario_id
 from playlist_f2
 union all
 select playlist_id,calificacion,indice,num_reproducciones,programa_id,
  usuario_id
  from playlist_f3
 union all
 select playlist_id,calificacion,indice,num_reproducciones,programa_id,
  usuario_id from playlist_f4;

--completar
```

- Observar que se listan los nombres de cada columna. Evitar el uso de "*" ya que el orden en el que se definen los atributos en cada fragmento puede cambiar y generar inconsistencias y/o errores. Hacer uso de `union all`.
- Observar que este código se puede ejecutar en las 4 PDBs ya que el código es exactamente el mismo para todos los sitios.

1.1.2.2. Creación de objetos adicionales para acceso a datos BLOB/CLOB

- En prácticas anteriores se revisaron 2 técnicas para implementar transparencia de distribución para acceder a un dato BLOB/CLOB de forma remota. En este proyecto se empleará únicamente la estrategia 2 en la que se obtiene un dato binario a través de su Id. Para ello, se deberán crear los siguientes objetos adicionales:
  - Tabla temporal para almacenar el dato BLOB/CLOB para transparencia de operaciones `select`. Se empleará la notación **`ts`**`_<nombre_global>_<num_fragmento>`. Por ejemplo, `ts_archivo_programa_1` es una tabla temporal que se empleará para almacenar el dato BLOB que se obtiene del fragmento 1
  - Tabla temporal para almacenar el dato BLOB/CLOB para transparencia de operaciones `insert`. Se empleará la notación **`ti`**`_<nombre_global>_<num_fragmento>`.
  - Funciones encargadas de obtener un dato BLOB/CLOB de un sitio remoto. Se empleará la sintaxis `get_remote_<nombre_columna_blob>_f<numero_fragmento>_by_id`.
    Por ejemplo, la función `get_remote_trailer_f1_by_id` será una función que obtiene el contenido del dato BLOB asociado a la columna `trailer` del fragmento 1 (`documental_f1`).
- Notar que no se requiere crear otros objetos que fueron empleados en prácticas anteriores (objetos `type, table`) ya que se estará empleando la estrategia 1
- Observar que estas tablas temporales y funciones son accedidas por prácticamente todos los nodos. Algunos nodos pudieran no requerirlas ya que el dato BLOB/CLOB se encontrará localmente. Lo ideal sería crear estas funciones únicamente en los sitios donde se requieren, pero para evitar duplicidad de código y aumento de la complejidad de los scripts, se creará uno solo y será ejecutado en todos los sitios.

Para implementar estos objetos, realizar las siguientes acciones:

- Crear un solo script llamado `s-05-netmax-tablas-temporales.sql` . El script deberá contener todas las tablas temporales que requieran manejo de datos CLOB/BLOB tanto para operaciones `insert` como para operaciones `select`. El código es exactamente el mismo, solo varían por el nombre, pero se usarán para propósitos diferentes.

Ejemplo:

```
--@Autor:          Jorge A. Rodríguez C
--@Fecha creación:
--@Descripción:     Definición de tablas temporales para acceso a BLOBs
--

Prompt tablas temporales para transparencia - Select

-- Tablas temporales para DOCUMENTAL
create global temporary table ts_documental_1 (
 programa_id number(10,0) constraint ts_documental_1_pk  primary key,
 trailer blob not null
) on commit preserve rows;

create global temporary table ts_documental_2 (
 programa_id number(10,0) constraint ts_documental_2_pk  primary key,
 trailer blob not null
) on commit preserve rows;

create global temporary table ts_documental_3 (
 programa_id number(10,0) constraint ts_documental_3_pk  primary key,
 trailer blob not null
) on commit preserve rows;




--tablas temporales para ARCHIVO_PROGRAMA
```

```
create global temporary table ts_archivo_programa_1 (
 num_archivo number(5,0),
 programa_id number(10,0),
 archivo blob not null,
 constraint ts_archivo_programa_1_pk primary key(num_archivo,programa_id)
) on commit preserve rows;

create global temporary table ts_archivo_programa_2 (
 num_archivo number(5,0),
 programa_id number(10,0),
 archivo blob not null,
 constraint ts_archivo_programa_2_pk primary key(num_archivo,programa_id)
) on commit preserve rows;


Prompt Prompt tablas temporales para transparencia - Insert

-- Tablas temporales para DOCUMENTAL
create global temporary table ti_documental_1 (
 programa_id number(10,0) constraint ti_documental_1_pk  primary key,
 trailer blob not null
) on commit preserve rows;

--completar
```

- Observar que, para la tabla `DOCUMENTAL`, se crean 6 tablas temporales: Las primeras 3 se crean para implementar transparencia con la instrucción `select` (una por fragmento), y las otras 3, para implementar transparencia con la instrucción `insert`.

- Crear un solo script llamado `s-05-netmax-funciones-blob.sql` El script contendrá la definición de las funciones requeridas para realizar acceso remoto a los datos CLOB/BLOB. Se requiere generar una función que extraiga el dato binario de cada fragmento.

Ejemplo:

```
--@Autor:          Jorge A. Rodríguez C
--@Fecha creación:
--@Descripción:     Definición de funciones para acceso a BLOBs


Prompt funciones  para acceso de blobs - DOCUMENTAL

--Funcion que obtiene BLOB del fragmento 1
create or replace function get_remote_trailer_f1_by_id(v_id in number)
 return blob is
 pragma autonomous_transaction;
 v_temp_trailer blob;
begin
 --asegura que no haya registros
 delete from ts_documental_1;
 --inserta los datos obtenidos del fragmento remoto a la tabla temporal.
 insert into ts_documental_1 select programa_id,trailer
  from documental_f1 where programa_id = v_id;
 --obtiene el registro de la tabla temporal y lo regresa como blob
 select trailer into v_temp_trailer from ts_documental_1 where programa_id = v_id;
 --elimina los registros de la tabla temporal una vez que han sido obtenidos.
 delete from ts_documental_1;
 commit;
  return v_temp_trailer;
 exception
  when others then
   rollback;
   raise;
end;
/
show errors
--completar para los demás fragmentos y tablas.
```

- Observar que la función obtiene el dato binario del fragmento 1, y se emplea la tabla temporal `ts_documental_1` . Esta función será empleada por todos aquellos sitios que requieran acceder de forma remota al fragmento 1
- Se deberán crear otras 2 funciones para los fragmentos 2 y 3. Es decir, las funciones `get_remote_trailer_f2_by_id` y `get_remote_trailer_f3_by_id`
- Aplicar la misma técnica para las demás tablas globales que contentan datos binarios.

1.1.2.3. Creación de vistas con datos CLOB/BLOB

- Crear un script por cada PDB llamado `s-05-netmax-<pdb>-vistas-blob.sql` Cada script contendrá la definición de las vistas que contienen columnas con datos CLOB/BLOB. El script deberá hacer uso de las funciones creadas en la sección anterior únicamente cuando sea necesario hacer un acceso remoto para obtener un dato CLOB/BLOB.

Ejemplo:

```
--@Autor:          Jorge A. Rodríguez C
--@Fecha creación:
--@Descripción:     Definición de vistas con columnas BLOB para jrcbd_s1
--

--DOCUMENTAL
Prompt creando vista DOCUMENTAL
create or replace view DOCUMENTAL as
 select programa_id,tematica,duracion,trailer,pais_id
 from documental_f1
 union all
 select programa_id,tematica,duracion,
  get_remote_trailer_f2_by_id(programa_id),pais_id
 from documental_f2
 union all
 select programa_id,tematica,duracion,
  get_remote_trailer_f3_by_id(programa_id),pais_id
 from documental_f3;


--ARCHIVO_PROGRAMA
Prompt creando vista ARCHIVO_PROGRAMA
create or replace view ARCHIVO_PROGRAMA as
 select num_archivo,programa_id,
  get_remote_archivo_f1_by_id(num_archivo,programa_id) as archivo,
  tamanio from archivo_programa_f1
 union all
 select num_archivo,programa_id,
  get_remote_archivo_f2_by_id(num_archivo,programa_id),
  tamanio from archivo_programa_f2;
```

- Observar que en el caso de la vista `documental`, se requiere un acceso local al fragmento 1 y un acceso remoto al fragmento 2 para obtener el dato binario, por lo tanto se hace uso de la función `get_remote_trailer_f2_by_id`.

1.1.2.4. Ejecución de scripts de creación de vistas

Generar un archivo llamado `s-05-netmax-vistas-main.sql`. El script deberá conectarse a cada PDB y ejecutar los scripts anteriores.

Ejemplo:

```
--@Autor:          Jorge A. Rodríguez C
--@Fecha creación:
--@Descripción:     Creación de  vistas para todos los sitios

clear screen
whenever sqlerror exit rollback;

prompt ========================================
prompt Creando vistas para jrcbd_s1
prompt ========================================

connect netmax_bdd/netmax_bdd@jrcbd_s1
@s-05-netmax-vistas.sql
@s-05-netmax-tablas-temporales.sql
@s-05-netmax-funciones-blob.sql
@s-05-netmax-jrc-s1-vistas-blob.sql

prompt ========================================
prompt Creando vistas para jrcbd_s2
prompt ========================================

prompt Creando vistas para jrcbd_s2
connect netmax_bdd/netmax_bdd@jrcbd_s2
@s-05-netmax-vistas.sql
@s-05-netmax-tablas-temporales.sql
@s-05-netmax-funciones-blob.sql
@s-05-netmax-jrc-s2-vistas-blob.sql
```

```
prompt ========================================
prompt Creando vistas para arcbd_s1
prompt ========================================

prompt Creando vistas para arcbd_s1
connect netmax_bdd/netmax_bdd@arcbd_s1
@s-05-netmax-vistas.sql
@s-05-netmax-tablas-temporales.sql
@s-05-netmax-funciones-blob.sql
@s-05-netmax-arc-s1-vistas-blob.sql


prompt ========================================
prompt Creando vistas para arcbd_s2
prompt ========================================

prompt Creando vistas para arcbd_s2
connect netmax_bdd/netmax_bdd@arcbd_s2
@s-05-netmax-vistas.sql
@s-05-netmax-tablas-temporales.sql
@s-05-netmax-funciones-blob.sql
@s-05-netmax-arc-s2-vistas-blob.sql

prompt Listo!
```

- Observar que se ejecutan los mismos scripts en todos los sitios excepto el último (marcado en negritas). Este último script contiene la definición de las vistas que contienen datos CLOB/BLOB y su definición es diferente en cada caso.

1.2. TRANSPARENCIA PARA INSTRUCCIONES DML.

- El alcance del proyecto se limita a transparencia para instrucciones `insert` y `delete`
- Para implementar transparencia para instrucciones `insert`, se deberán crear `instead of triggers` para cada tabla fragmentada.
- Solo se deberá implementar los eventos de inserción y eliminación. El trigger deberá enviar un error cuando se intente realizar una operación `update`, indicando que estas aún no están implementadas.

1.2.1. Creación de triggers.

- En algunos casos, el código del trigger es distinto para cada sitio, por ejemplo, en fragmentaciones horizontales derivadas, o en tablas con datos CLOB/BLOB. En otros casos el código es el mismo. Con la finalidad de evitar duplicidad de código, realizar lo siguiente:
  - Si el código es diferente para cada sitio, crear un archivo llamado `s-06-netmax-trigger-<pdb>-<tabla>.sql` para cada trigger
  - Si el código es el mismo, crear un archivo llamado `s-06-netmax-trigger-<tabla>.sql`

Ejemplo:

- Código del trigger `s-06-netmax-trigger-usuario.sql`. Por sus características, este se puede ejecutar en todos los sitios.

```
--@Autor:          Jorge A. Rodríguez C
--@Fecha creación:
--@Descripción:     Definición del trigger para usuario,
--                  se puede ejecutar en todos los nodos.

create or replace trigger t_insert_usuario
 instead of insert
 or update
 or delete  on usuario
 declare
begin
case
 when inserting then
  --validacion para fragmento 1
  if :new.tipo_cuenta_id = 1 and :new.vigente = 1 then
   insert into usuario_f1 (usuario_id,email,nombre,ap_paterno,ap_materno,
    fecha_ingreso,vigente,fecha_cuenta_fin,tipo_cuenta_id)
   values (:new.usuario_id,:new.email,:new.nombre,:new.ap_paterno,:new.ap_materno,
    :new.fecha_ingreso,:new.vigente,:new.fecha_cuenta_fin,:new.tipo_cuenta_id);

  --validacion para fragmento 2

  --validacion para  fragmento 3
```

```
--validacion para fragmento 4

--Fragmento 5 vertical, siempre se inserta el num de tarjeta y password

 when updating then
  raise_application_error(-20002, 'Operacion no implementada aun');

 when deleting then
  if :old.tipo_cuenta_id = 1 and :old.vigente = 1 then
   delete from usuario_f1 where usuario_id = :old.usuario_id;
  elsif :old.tipo_cuenta_id = 2 and :old.vigente = 1 then
   delete from usuario_f2 where usuario_id = :old.usuario_id;
  elsif :old.tipo_cuenta_id = 3 and :old.vigente = 1 then
   delete from usuario_f3 where usuario_id = :old.usuario_id;
  elsif :old.tipo_cuenta_id  not in (1,2,3) and :old.vigente = 0 then
   delete from usuario_f4 where usuario_id = :old.usuario_id;
  else
   raise_application_error(-20001,'Valor invalido de tipo_cuenta_id y vigente: '
         || :old.tipo_cuenta_id || ', '||:old.vigente);
  end if;
  --elimina el fragmento vertical
  delete from usuario_f5 where usuario_id = :old.usuario_id;
 end case;
end;
/
show errors;
```

Ejemplo:

Trigger para Documental en uno de los 4 sitios.

```
--@Autor:          Jorge A. Rodríguez C
--@Fecha creación:
--@Descripción:     Definición del trigger para documental
--                  para jrcbd_s1

create or replace trigger t_insert_documental
 instead of insert
 or delete
 or update on documental
declare
v_exists number(1,0);

begin
 case
 when inserting then
  -- insercion local f1
  select count(*) into v_exists
  from programa_f1
  where programa_id = :new.programa_id;
  if v_exists = 1  then
   insert into documental_f1(programa_id,tematica,duracion,trailer,pais_id)
    values(:new.programa_id,:new.tematica,:new.duracion,:new.trailer,:new.pais_id);
   return;
  end if;

  -- insercion remota f2
  select count(*) into v_exists
  from programa_f2
  where programa_id = :new.programa_id;
  if v_exists = 1  then

   --uso de tabla temporal
   delete from ti_documental_2
   where programa_id = :new.programa_id;

   insert into ti_documental_2(programa_id,tematica,duracion,trailer,pais_id)
    values(:new.programa_id,:new.tematica,:new.duracion,:new.trailer,:new.pais_id);

   insert into documental_f2(programa_id,tematica,duracion,trailer,pais_id)
    select programa_id,tematica,duracion,trailer,pais_id
     from ti_documental_2
     where programa_id = :new.programa_id;
   delete from ti_documental_2
   where programa_id = :new.programa_id;
   return;
  end if;
```

```
  -- insercion remota f3
  select count(*) into v_exists
  from programa_f3
  where programa_id = :new.programa_id;
  if v_exists = 1  then

   --uso de tabla temporal
   delete from ti_documental_3
   where programa_id = :new.programa_id;

   insert into ti_documental_3(programa_id,tematica,duracion,trailer,pais_id)
    values(:new.programa_id,:new.tematica,:new.duracion,:new.trailer,:new.pais_id);

   insert into documental_f3(programa_id,tematica,duracion,trailer,pais_id)
    select programa_id,tematica,duracion,trailer,pais_id
     from ti_documental_3
     where programa_id = :new.programa_id;
   delete from ti_documental_3
   where programa_id = :new.programa_id;
   return;
  end if;

  raise_application_error(-20001,'Registro padre no encontrado en nodos. programa_id: '
            || :new.programa_id );

 when updating then
  raise_application_error(-200002,'Operacion no implementada aun');

 when deleting then
  --eliminacion local f1
  delete from documental_f1 where programa_id = :old.programa_id;
  if(sql%rowcount > 0) then
   return;
  end if;

  --eliminacion remota f2
  delete from documental_f2 where programa_id = :old.programa_id;
  if(sql%rowcount > 0) then
   return;
  end if;

  --eliminacion remota f3
  delete from documental_f3 where programa_id = :old.programa_id;
  if(sql%rowcount > 0) then
   return;
  end if;
 end case;
end;
/
show errors
```

- Al terminar de crear los scripts anteriores, invocarlos en uno nuevo llamado `s-06-netmax-main-trigger.sql`. Tener cuidado con invocar cada script con base al sitio donde deben ejecutarse.

Ejemplo:

```
--@Autor:          Jorge A. Rodríguez C
--@Fecha creación:
--@Descripción:     Script principal - creación de triggers
clear screen
whenever sqlerror exit rollback;

prompt ====================================
prompt Creando triggers para jrcbd_s1
prompt ====================================
connect netmax_bdd/netmax_bdd@jrcbd_s1
@s-06-netmax-trigger-usuario.sql
@s-06-netmax-trigger-programa.sql
@s-06-netmax-trigger-jrc-s1-serie.sql
@s-06-netmax-trigger-jrc-s1-pelicula.sql
@s-06-netmax-trigger-jrc-s1-documental.sql
@s-06-netmax-trigger-jrc-s1-archivo-programa.sql
@s-06-netmax-trigger-jrc-s1-playlist.sql

prompt ====================================
prompt Creando triggers para jrcbd_s2
prompt ====================================
connect netmax_bdd/netmax_bdd@jrcbd_s2
@s-06-netmax-trigger-usuario.sql
@s-06-netmax-trigger-programa.sql
```

```
@s-06-netmax-trigger-jrc-s2-serie.sql
@s-06-netmax-trigger-jrc-s2-pelicula.sql
@s-06-netmax-trigger-jrc-s2-documental.sql
@s-06-netmax-trigger-jrc-s2-archivo-programa.sql
@s-06-netmax-trigger-jrc-s2-playlist.sql


prompt ====================================
prompt Creando triggers para arcbd_s1
prompt ====================================
connect netmax_bdd/netmax_bdd@arcbd_s1
@s-06-netmax-trigger-usuario.sql
@s-06-netmax-trigger-programa.sql
@s-06-netmax-trigger-arc-s1-serie.sql
@s-06-netmax-trigger-arc-s1-pelicula.sql
@s-06-netmax-trigger-arc-s1-documental.sql
@s-06-netmax-trigger-arc-s1-archivo-programa.sql
@s-06-netmax-trigger-arc-s1-playlist.sql


prompt ====================================
prompt Creando triggers para arcbd_s2
prompt ====================================
connect netmax_bdd/netmax_bdd@arcbd_s2
@s-06-netmax-trigger-usuario.sql
@s-06-netmax-trigger-programa.sql
@s-06-netmax-trigger-arc-s2-serie.sql
@s-06-netmax-trigger-arc-s2-pelicula.sql
@s-06-netmax-trigger-arc-s2-documental.sql
@s-06-netmax-trigger-arc-s2-archivo-programa.sql
@s-06-netmax-trigger-arc-s2-playlist.sql

prompt Listo!
```

- Observar que solo los triggers para usuario y programa son los únicos cuyo código se puede reutilizar debido a que su fragmentación es horizontal primaria.

Continuar con la última parte del Proyecto.