

Postmortem Alta De Planes

04/04/2017

0.1

|  |  |
| --- | --- |
| **Nombre.** | **Matricula** |
| **Jesús Alberto Goiz Barrales.** | **24400085** |
| **Juan Alberto Gutiérrez Canto.** | **24400063** |
| **Brenda Robles Antonio.** | **24400073** |
| **Rene Moratilla Montes.** | **24400075** |
| **Guillermo Vivaldo Vazquez** | **24400093** |

**Control De Versiones**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Nombre Del Archivo | Versión | Fecha | Autor | Comentarios |
| Csoft-PM\_Alta-De-Planes-040417.docx | 0.1 | 04/04/17 | GVV | Creación del Documento |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

# Índice

[**Índice**](#_kd5bfhkalq0n) **2**

[**I.- Forma PSP Project Plan Summary.**](#_jhy44tc5k06j) **3**

[**II.- Time Recording Log.**](#_1fob9te) **3**

[**III.- Defect Recording Log.**](#_3znysh7) **3**

[**IV.- Diseño.**](#_m4hw86ggwsmr) **3**

[**V.- Design CheckList.**](#_92tu2pewgguj) **3**

[**VI.- Programa Fuente.**](#_6bvnqqrwimi9) **4**

[**VII.- Code CheckList.**](#_87ns93g4124c) **4**

[**VIII.- Reporte de resultados.**](#_bo7s7l2hg3ra) **4**

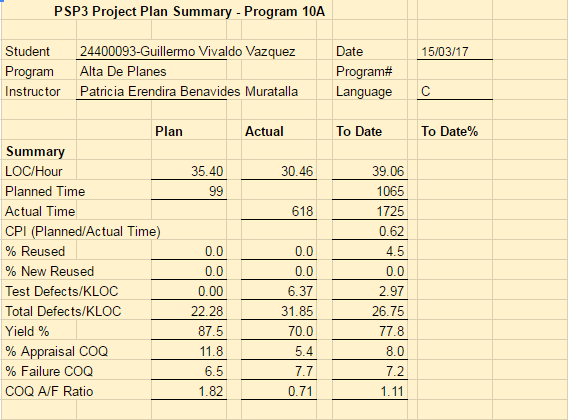
[**IX.- PIP**](#_4ocjjwajumc0) **4**

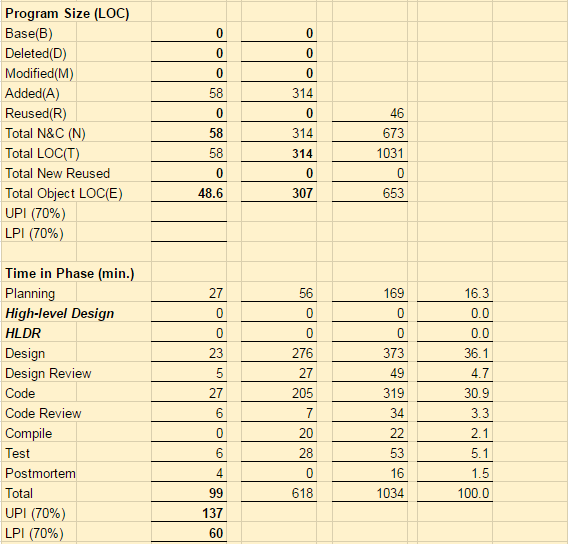
[**X.- Size Estimating Template**](#_w1ych8jaj7as) **5**

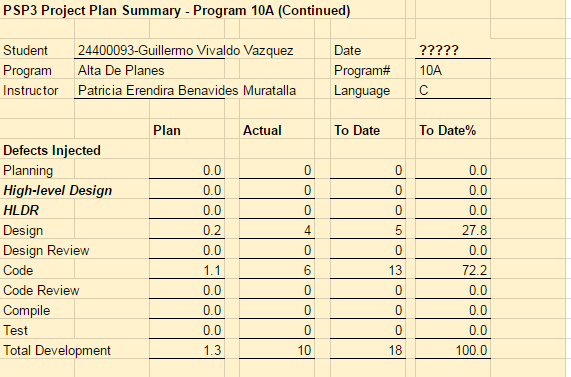
[**XI.- Task and Schedule**](#_195jonwtvlxs) **5**

# I.- Forma PSP Project Plan Summary.

Se inició el proceso el 15/03/2017







# 

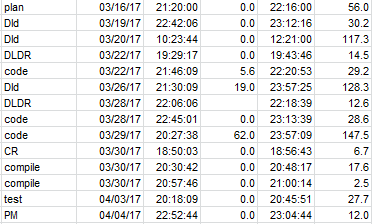
# 

# 

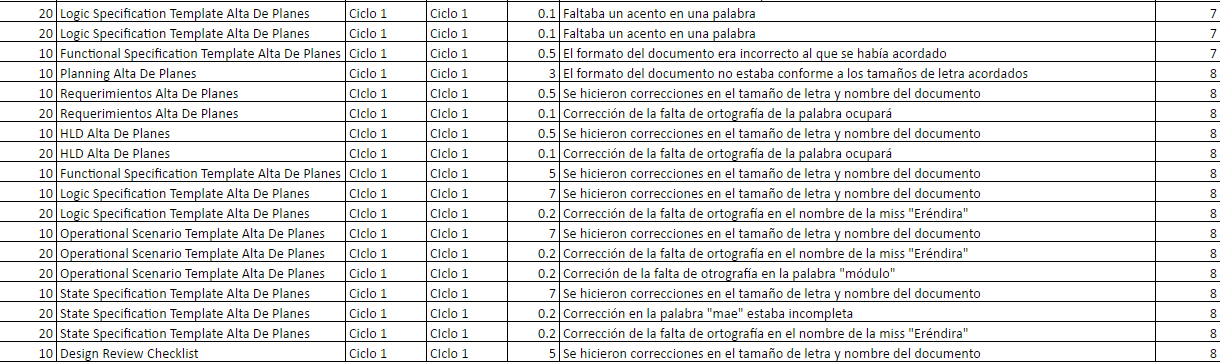
# 

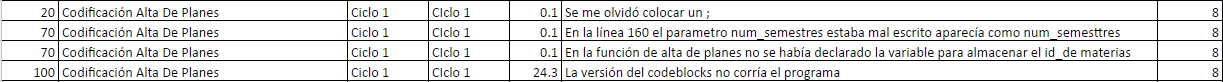
# 

# II.- Time Recording Log.



# III.- Defect Recording Log.





# IV.- Diseño.

**State Specification Template**

|  |  |  |  |
| --- | --- | --- | --- |
| Student | Guillermo Vivaldo Vazquez | Date | 26/03/17 |
| Program | Alta De Planes | Program # |  |
| Instructor | Patricia Eréndira Benavides Muratalla | Language | C |
| Object |  | Routine |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Menú principal | | | |
| **State Name** | | **Description** | |
| Inicio | | Inicia el programa. | |
| Insert\_Materias | | Guarda en la base de datos el alta de materias | |
| Alta\_Planes | | Función para empezar el alta de maestros. | |
| Insert\_Planes | | Guarda en la base de datos el alta de planes. | |
| Num\_semestres | | Generará un ciclo para que a cada semestre se le asignen materias | |
| Insert Materias-Planes | | Guarda en la base de datos la vinculación de planes y materias | |
| Query Planes | | Imprime desde la base de datos el registro. | |
| Query Planes-Materia | | Imprime desde la base de datos el registro | |
| Salir | | Termina el programa. | |
|  | |  | |
| **Function/Parameter** | | **Description** | |
| printf() | | Imprime datos en la pantalla | |
| scanf() | | Lee datos de la pantalla | |
| Insert\_Materias | | Guarda las materia en la Base | |
| Alta\_Planes() | | Enlace para el alta de planes | |
| Insert\_Planes() | | Guarda los planes en base de datos | |
| Insert\_Mtr\_Plan() | | Guarda vinculación materias-plan | |
| Retrurn() | | Regresa un valor. | |
| Query\_Planes | | Imprime registro desde base de datos | |
| actualizar | | Variable para ver si quiere actualizar lista de planes. | |
| cantidad\_materias | | Para determinar cuántas materias se añadirán al plan. | |
| **States/Next States** | | **Transition Condition** | **Action** |
| Inicio | |  |  |
|  | Alta Planes | Inicia el programa | Alta\_planes() |
| Alta Materias | |  |  |
|  | Muestra Materias | actualizar==true | Actualizar\_planes() |
|  | Insert Planes |  | Insert\_Planes() |
|  | Insert Materias-Planes | mientras cantidad\_materias ==i con i=0 para i++ | Insert\_Mtr\_Plan() |
|  | Query Planes | Ya se insertó en la base de datos | Query\_Planes() |
|  | Salir | Se imprimió el Alta de planes | Return() |

# Logic Specification Template

|  |  |  |  |
| --- | --- | --- | --- |
| Student | Guillermo Vivaldo Vazquez | Date | 26/03/17 |
| Program | Alta De Planes | Program # |  |
| Instructor | Patricia Eréndira Benavides Muratalla | Language | C |
| Object |  | Function |  |

INCLUDES:

#include <stdio.h>

#include<stdlib.h>  
#include<cstdlib>

#include<string.h>

#include “sqlite3.h”

TYPE DEFINITIONS:

|  |  |
| --- | --- |
| **Declaration:** | void query\_planes() |
| **Reference:** | Functional template |
|  |  |

|  |  |
| --- | --- |
| Logic reference numbers | Program logic, in pseudocode |
| 1 | char \*sql |
|  | char \*err |
|  | int rc |
|  | const char\* data = "Datos Planes" |
|  | asigna a sql=”SELECT \* FROM PLANES WHERE Id\_Pl =’”\id\_plan”’);”; |
|  | se asigna a rc=sqlite3\_exec(db, sql, callback, (void\*)data, &err); |
|  | si rc != SQLITE\_OK |
|  | printf(“SQL error: %s\n”,err); |
|  | sqlite3\_free(err); |
|  | si no |
|  | printf(“Valores ingresados correctamente\n”); |
|  | data = "Con las siguientes materias"; |
|  | asigna a sql=”SELECT \* FROM POSEE WHERE Id\_Plan =’”\id\_plan”’);”; |
|  | se asigna a rc=sqlite3\_exec(db, sql, callback, (void\*)data, &err); |
|  | si rc != SQLITE\_OK |
|  | printf(“SQL error: %s\n”,err); |
|  | sqlite3\_free(err); |
|  | si no |
|  | printf(“Valores ingresados correctamente\n”); |

TYPE DEFINITIONS:

|  |  |
| --- | --- |
| **Declaration:** | Alta\_planes() |
| **Reference:** | Functional template |
|  |  |

|  |  |
| --- | --- |
| Logic reference numbers | Program logic, in pseudocode |
| 1 | string id\_plan |
|  | string nombre |
|  | string fecha |
|  | Int num\_semestes |
| Int | Int num\_materias, i=0, j=0; |
|  | int status=1 |
|  | mientras status != 0 |
|  | imprime menu para pedir datos al usuario |
|  | lee id\_plan |
|  | lee nombre |
|  | lee fecha |
|  | lee num\_semestres |
|  | se imprimen los datos leídos y se preguntan si son     correctos 0)si 1)no |
|  | leer num\_semestres |
|  | Hacer mientras i<=num\_semestres |
| impr | Imprimir ¿Cuántas materias se le asiganara al semestre i? |
|  | Leer num\_materias |
|  | Hacer mientras j<=num\_materias |
|  | se llama a la función insert\_mtr\_plan(id\_plan,         id\_materia, semestre) |
|  | se llama a la función query\_maestros() y se muestra los datos introducidos. |

|  |  |
| --- | --- |
| **Declaration:** | Alta\_materias |
| **Reference:** | Functional template |
|  |  |

|  |  |
| --- | --- |
| Logic reference numbers | Program logic, in pseudocode |
| 1 | char \*sql |
|  | char \*err |
|  | int rc |
|  | asigna a sql=”INSERT INTO MATERIAS (Id\_Materia, Nombre) “ \ “VALUES (‘“ \id\_materia \ ”’,’” \ nombre \ ”’);”; |
|  | se asigna a rc=sqlite3\_exec(db, sql, callback, 0, &err); |
|  | si rc != SQLITE\_OK |
|  | printf(“SQL error: %s\n”,err); |
|  | sqlite3\_free(err); |
|  | si no |
|  | printf(“Valores ingresados correctamente\n”); |

|  |  |
| --- | --- |
| **Declaration:** | Asignación\_materia() |
| **Reference:** | Functional template |
|  |  |

|  |  |
| --- | --- |
| Logic reference numbers | Program logic, in pseudocode |
| 1 | char \*sql |
|  | char \*err |
|  | asigna a sql=”INSERT INTO POSEE (ID\_Plan, ID\_Materia, Semestre) “ \ “VALUES (‘“ \ id\_plan ”’,’” \ id\_materia ”’,’” \ semestre “‘);”; |
|  | se asigna a rc=sqlite3\_exec(db, sql, callback, 0, &err); |
|  | si rc != SQLITE\_OK |
|  | printf(“SQL error: %s\n”,err); |
|  | sqlite3\_free(err); |
|  | si no |
|  | printf(“Valores ingresados correctamente\n”); |

|  |  |
| --- | --- |
| **Declaration:** | Int\_callback () |
| **Reference:** | Functional template |
|  |  |

|  |  |
| --- | --- |
| Logic reference numbers | Program logic, in pseudocode |
| 1 | int i |
|  | imprime en pantalla con printf( "%s:\n ", (const char\*)data); |
|  | para i=0 mientras i<argc en i++ |
|  | printf("%s = %s\n", azColName[i], argv[i] ? argv[i] : "NULL"); |
|  | printf(”\n”); |
|  | return 0; |

|  |  |
| --- | --- |
| **Declaration:** | Int\_main () |
| **Reference:** | Functional template |
|  |  |
| **Logic reference numbers** | **Program logic, in pseudocode** |
| 1 | INICIO |
| 2 | Alta\_planes(); |
| 3 | System(“pause”); |
| 4 | Return 0; |
| 5 | FIN |

# Functional Specification Template

|  |  |  |  |
| --- | --- | --- | --- |
| Student | Guillermo Vivaldo Vazquez | Date | 20/03/2017 |
| Program | Alta De Planes | Program # |  |
| Instructor | Patricia Eréndira Benavides Muratalla | Language | C |

|  |  |  |  |
| --- | --- | --- | --- |
| Object/Class Name  main | | Parent Classes  Base | Attributes  Op (1-3) |
|  | Method Declaration | Method External Specification | |
|  | Alta\_plan | Función que solicitará al usuario el nombre de un plan, un ID que el usuario deberá asignar, número de semestres y fecha. | |
|  | Alta\_materia | Solicitará la cantidad de semestres para ese plan de estudios. | |
|  | Asignación\_materia | Dependiendo del semestre se le asignará un conjunto de materias | |
|  | Query\_planes | Muestra que los valores han sido ingresados correctamente a la base | |
|  | Int\_callback () | Esta función se encarga de imprimir los datos encontrados en la base de datos | |
|  | | | |
|  | | | |
|  | | | |
|  | | | |
|  | | | |
|  | | | |
|  | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Object/Class Name  Alta\_plan | | Parent Classes  main | Attributes  id(0 - n)  nombre(datos)  semestre(0 – n)  fecha(datos) |
|  | Method Declaration | Method External Specification | |
|  | Asignación\_materia | Dependiendo del semestre se le asignará un conjunto de materias | |
|  | Int\_callback () | De acuerdo a la materia :: mostrará datos de la materia. | |
|  | | | |
|  | | | |
|  | | | |
|  | | | |
|  | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Object/Class Name  Alta\_materia | | Parent Classes  main | Attributes  id(0 - n)  nombre(datos) |
|  | Method Declaration | Method External Specification | |
|  | Int\_callback () | De acuerdo a la materia :: mostrará datos de la materia. | |
|  | | | |
|  | | | |
|  | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Object/Class Name  Asignación\_materia | | Parent Classes  Alta\_plan | Attributes  Materia(datos) |
|  | Method Declaration | Method External Specification | |
|  | Asignación\_materia | De acuerdo al semestre :: se ingresará un conjunto de materias a ese semestre. | |
|  | | | |
|  | | | |
|  | | | |

# Operational Scenario Template

|  |  |  |  |
| --- | --- | --- | --- |
| Student | Guillermo Vivaldo Vazquez | Date | 19/03/17 |
| Program | Módulo Alta De Planes | Program # |  |
| Instructor | Patricia Eréndira Benavides Muratalla | Language | C |

Construct operational scenarios to cover the normal and abnormal program uses, including user errors.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Scenario Number: 1 | | | User Objective: Solicitar al usuario dar de alta la información de un plan solicitando un ID de plan, su nombre, ingresar un semestre y a ese semestre asignarle ciertas materias. | |
| Scenario Objective: Mostrar el funcionamiento general del programa, así como los casos alternativos al elegir una opción incorrecta y para probar el sistema. | | | | |
| Source | Step | Action | | Comments |
| Sistema | 1 | Solicita al usuario ingresar un ID del plan. | |  |
| Usuario | 2 | El usuario ingresa un ID . | | Los ID son únicos y no deben de repetirse. |
| Sistema | 3 | Solicita al usuario ingresar el nombre del plan. | | Se ingresa una cadena de caracteres el cual representará el nombre de ese plan. |
| Usuario | 4 | El usuario ingresa el nombre de un plan de estudios. | | Puede haber una gran cantidad de planes de estudios los cuales son definidos por la universidad. |
| Sistema | 5 | Se solicita al usuario ingresar los semestres del plan. | | Hay planes que tienen más semestres que otros dependiendo a la carga de materias que tiene cada plan. |
| Sistema | 6 | Se mostrará el número de semestre y a cada semestre se le solicitará al usuario asignar un número de materias que vayan de acuerdo al semestre. | |  |
| Usuario | 7 | El usuario agrega las materias correspondientes al semestre solicitado. | |  |
| Sistema | 8 | Se repiten los pasos 6 y 7 hasta que se hayan registrado materias a todos los semestres. | |  |
| Sistema | 9 | El sistema pregunta al usuario si desea agregar otro plan. | |  |
| Usuario | 10 | El usuario elige la opción 1 que corresponde a si y se repiten los pasos del 1 al 9. | |  |
| Usuario | 11 | El usuario elige la opción 2 que corresponde a no y la ejecución del módulo termina. | |  |
|  |  |  | |  |
|  |  |  | |  |
|  |  |  | |  |
|  |  |  | |  |
|  |  |  | |  |
|  |  |  | |  |

# V.- Design CheckList.

# Revisión diseño

PROGRAM NAME AND #: Alta De Planes

1. Template de Especificación Funcional
2. Template de Escenario Operacional
3. Template de Especificación de Estados
4. Templates de Especificación Lógica

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Purpose | To guide you in conducting an effective design review | 1 | 2 | 3 | 4 |
| General | • As you complete each review step, check off that item in the box to the right.  • Complete the checklist for one program unit before you start to review the next unit. | \* | \* | \* | \* |
| Complete | Ensure the requirements, specifications, and high-level design are completely covered by the design:  • All specified outputs are produced.  • All needed inputs are furnished.  • All required includes are stated. | \* | \* | \* | \* |
| State Machine | Verify the state machine design:  • The structure has no hidden traps or loops.  • It is complete; that is, all possible states have been identified.  • It is orthogonal; that is, for every set of conditions there is one and only one possible next state.  • The transitions from each state are complete and orthogonal. That is, from every state, a unique next state is defined for every possible combination of state machine input values. | ***\**** | ***\**** | ***\**** | ***\**** |
| Logic | • Verify that program sequencing is proper:   * Stacks, lists, and so on are in the proper order. * Recursion unwinds properly.   • Verify that all loops are properly initiated, incremented, and terminated. | \* | \* | \* | \* |
| Special Cases | Check all special cases:  • Ensure proper operation with empty, full, minimum, maximum, negative, zero values for all variables.  • Protect against out-of-limits, overflow, underflow conditions.  • Ensure “impossible” conditions are absolutely impossible.  • Handles all incorrect input conditions. | \* | \* | \* | \* |
| Functional Use | • Verify that all functions, procedures, or objects are fully understood and properly used.  • Verify that all externally referenced abstractions are precisely defined. | \* | \* | \* | \* |
| Names | Verify the following.  • All special names and types are clear or specifically defined.  • The scopes of all variables and parameters are self-evident or defined.  • All named objects are used within their declared scopes. | \* | \* | \* | \* |
| Standards | Review the design for conformance to all applicable design standards. | \* | \* | \* | \* |

# VI.- Programa Fuente.

/\*=======================================================================================\*/

/\*Autor: Guillermo Vivaldo Vazquez \*/

/\*Nombre del Programa: Alta De Planes \*/

/\*Fecha: 22/Marzo/2017 \*/

/\*Descripción: Este programa realiza el alta de planes y de materias que tendra cada plan\*/

/\*Version: 1.0 \*/

/\*=======================================================================================\*/

/\*=====================================\*/

/\*Contenido \*/

/\*static int callback() \*/

/\*void query\_planes() \*/

/\*void insert\_mtr\_plan() \*/

/\*void insert\_materias() \*/

/\*void insert\_planes() \*/

/\*static int print\_archivo() \*/

/\*void alta\_materias() \*/

/\*void alta\_planes() \*/

/\*int\_main() \*/

/\*=====================================\*/

/\*====================================================================\*/

/\*Instrucciones De Reuso \*/

/\*No se utilizó reuso \*/

/\*====================================================================\*/

/\*Declaración De Librerias\*/

#include <iostream>

#include <cstdio>

#include <cstdlib>

#include <string>

#include "sqlite3.h"

/\*Interacción base de datos\*/

using namespace std;

FILE \*pfile;

/\*\*\*Esta función se encarga de imprimir los datos encontrados en la base de datos\*\*\*/

static int callback (void \*data,int argc, char \*\*argv,char \*\*az\_col\_Name)

{

int i;

printf( "%s:\n ", (const char\*)data);

for(i=0;i<argc;i++)

{

printf("%s = %s\n", az\_col\_Name[i], argv[i] ? argv[i] : "NULL");

}

printf("\n");

return 0;

}

/\*\*\*\*Esta función busca los valores del plan en la base de datos\*\*\*\*/

void query\_planes(sqlite3 \*db,char\* id\_plan)

{

char \*sql; /\*Para la sentencia sql \*/

char \*err=0; /\*Para el mensaje de error de la sentencia sql \*/

int rc;

sqlite3\_stmt \*res;

sql="SELECT \* FROM PLANES where ID=@Id\_P;"; /\* Selecciona los planes con un id especifico sentencia sql\*/

rc=sqlite3\_prepare\_v2(db,sql,-1,&res,0); /\* Verifica si es valida la instruccion sql que se solicita\*/

if(rc!=SQLITE\_OK)

{

printf(" QUERY-SQL error %s\n",err); /\* Imprime el error de la base de datos\*/

}

else

{

printf(" Valores del plan\n"); /\* Se aprueba la sentencia sql\*/

int idx = sqlite3\_bind\_parameter\_index(res, "@Id\_P"); /\* Se busca en que lugar esta el identificador a cambiar\*/

sqlite3\_bind\_text(res, idx, id\_plan,40,SQLITE\_STATIC); /\* Se cambia el identificador por un valor determinado\*/

}

int step = sqlite3\_step(res);

if (step == SQLITE\_ROW) /\* indica el resultado de la instruccion sql, valores del plan\*/

{

printf(" Nombre:\t%s\n", sqlite3\_column\_text(res, 1));

printf(" Numero De Semestre:\t%d\n", sqlite3\_column\_int(res, 2));

printf(" Fecha:\t%s\n", sqlite3\_column\_text(res, 3));

}

sqlite3\_finalize(res); /\* Finaliza el objeto creado\*/

sql="SELECT ID\_Materia,ID\_Plan,Semestre FROM POSEE where ID\_Plan=@Id\_P;"; /\* Instruccion sql para la seleccion de materias de cada plan en tabla POSEE\*/

rc=sqlite3\_prepare\_v2(db,sql,-1,&res,0);

printf("\n"); /\* Verifica si es correcta la instruccion sql\*/

if(rc!=SQLITE\_OK)

{

printf("SQL error %s\n",err); /\* Imprime el error de la consulta\*/

}

else

{

printf(" Materias del Plan\n"); /\* Cambia los valores para el identificador en especifico\*/

int idx = sqlite3\_bind\_parameter\_index(res, "@Id\_P");

sqlite3\_bind\_text(res, idx, id\_plan,40,SQLITE\_STATIC);

}

step = sqlite3\_step(res);

while(step == SQLITE\_ROW) /\* imprime los valores de la consulta de diferentes registros\*/

{

printf(" %s\t%s\t\%d\n", sqlite3\_column\_text(res, 0),sqlite3\_column\_text(res, 1),sqlite3\_column\_int(res,2));

step=sqlite3\_step(res);

}

sqlite3\_finalize(res); /\* Libera el espacio del objeto\*/

}

/\*\*\*Esta función se encarga de insertar los valores de la materia y el plan en la base de datos\*\*\*/

void insert\_mtr\_plan(sqlite3 \*db,char\* id\_materia, char\* id\_plan, int semestre)

{

char \*sql;

char \*err=0;

int rc;

sqlite3\_stmt \*res;

sql="INSERT INTO POSEE (ID\_Materia, ID\_Plan, Semestre) VALUES (@Id\_M,@Id\_P,@semestre);";/\* Inserta los maestros en la tabla imparte los tres valores de esta tabla\*/

rc=sqlite3\_prepare\_v2(db,sql,-1,&res,0); /\* Prepara la sentencia sql\*/

if(rc!=SQLITE\_OK)

{

printf(" POSEE-SQL error %s\n",err); /\* Imprime el error de la sentencia\*/

}

else{

printf(" Valores ingresados correctamente.\n"); /\* Cambia los identificades por valores en la sentencia\*/

int idm = sqlite3\_bind\_parameter\_index(res, "@Id\_M"); /\* busca en donde esta el identificador\*/

int idt = sqlite3\_bind\_parameter\_index(res, "@Id\_P");

int idsemestre = sqlite3\_bind\_parameter\_index(res, "@semestre");

sqlite3\_bind\_text(res, idm, id\_materia,40,SQLITE\_STATIC); /\* Cambia el identificador por un valor text\*/

sqlite3\_bind\_text(res, idt, id\_plan,40,SQLITE\_STATIC); /\* Cambia el identificador por un valor text\*/

sqlite3\_bind\_int(res, idsemestre, semestre); /\* Cambia el identificador por un valor int\*/

}

int step = sqlite3\_step(res); /\* Realiza la consulta sql\*/

if (step == SQLITE\_ROW)

{

printf(" %s\n", sqlite3\_column\_text(res, 0)); /\* Imprime datos devueltos de la consulta\*/

}

}

/\*\*\*Esta función se encarga de insertar los valores de la materia en la base de datos\*\*\*/

void insert\_materias(sqlite3 \*db,char\* id\_materia, char\* nombre)

{

char \*sql;

char \*err=0;

int rc;

sqlite3\_stmt \*res; /\* Inserta todos los valores de maestro sentncia sql\*/

sql="INSERT INTO MATERIAS (Id\_M, Nombre) VALUES (@Id\_M,@nom);";

rc=sqlite3\_prepare\_v2(db,sql,-1,&res,0); /\* Prepara la sentencia sql\*/

if(rc!=SQLITE\_OK){

printf("SQL error %s\n",err); /\* Imprime el mensaje de error\*/

}

else{

printf(" Valores de materias guardados correctamente.\n");/\* Cambia los identificador por los valores\*/

int idm = sqlite3\_bind\_parameter\_index(res, "@Id\_M"); /\* Busca el lugar del identificador\*/

int nom = sqlite3\_bind\_parameter\_index(res, "@nom");

sqlite3\_bind\_text(res, idm, id\_materia,40,SQLITE\_STATIC); /\* cambia el valor por elidenitficador \*/

sqlite3\_bind\_text(res, nom, nombre,40,SQLITE\_STATIC);

}

int step = sqlite3\_step(res);/\* \*/

if (step == SQLITE\_ROW)

{

printf(" Correcto %s\n", sqlite3\_column\_text(res, 0)); /\* Imprime el resultado de la consulta\*/

}

}

/\*\*\*\*Esta función realiza un insert de los valores de las materias en la base de datos\*\*\*\*/

void insert\_planes(sqlite3 \*db,char\* id\_plan, char\* nombre, int num\_semestres, char\* fecha)

{

char \*sql;

char \*err=0;

int rc;

sqlite3\_stmt \*res;

sql="INSERT INTO PLANES (ID, Nombre, Num\_Semestres, Fecha) VALUES (@Id\_P,@nom,@sem,@dat);";

rc=sqlite3\_prepare\_v2(db,sql,-1,&res,0); /\* Prepara la sentencia sql\*/

if(rc!=SQLITE\_OK)

{

printf("SQL error %s\n",err);

}

else

{

printf("Valores de los planes guardados correctamente.\n");

int idm = sqlite3\_bind\_parameter\_index(res, "@Id\_P"); /\* Busca el lugar del identificador\*/

int nom = sqlite3\_bind\_parameter\_index(res, "@nom");

int semes = sqlite3\_bind\_parameter\_index(res, "@sem");

int fch = sqlite3\_bind\_parameter\_index(res, "@dat");

sqlite3\_bind\_text(res, idm, id\_plan,40,SQLITE\_STATIC); /\* cambia el valor por elidenitficador \*/

sqlite3\_bind\_text(res, nom, nombre,40,SQLITE\_STATIC);

sqlite3\_bind\_int(res, semes, num\_semestres);

sqlite3\_bind\_text(res, fch, fecha,40,SQLITE\_STATIC);

}

int step = sqlite3\_step(res);/\* \*/

if (step == SQLITE\_ROW)

{

printf(" Correcto %s\n", sqlite3\_column\_text(res, 0)); /\* Imprime el resultado de la consulta\*/

}

}

/\*\*\*Esta función se encarga de imprimir los datos encontrados en la base de datos a un archivo de txt\*\*\*/

static int print\_archivo(void \*data,int argc, char \*\*argv,char \*\*az\_col\_name)

{

int i;

for(i=0;i<argc;i++)

{

fprintf(pfile,"%s\t",argv[i]); /\* Imprime en el archivo txt las materias disponibles en la bd\*/

}

fprintf(pfile,"\n");

return 0;

}

/\*\*\*Esta función se encarga de presentar alta de materias\*\*\*/

/\*\*\*Contiene todo el proceso y redirecciona hacia las funciones de inserción y select con la base de datos\*\*\*/

void alta\_materias(sqlite3 \*db)

{

char id\_materia[40];

char nombre[40];

bool correcto=false;

int status=2;

while(status != 1){ /\* Pide datos haste que esten correctos aprobados por el usuario\*/

system("cls");

printf(" °----------------------------------------------------------°\n"); /\*Menu Alta maestros\*/

printf(" | ============== Alta Materias =============== |\n");

printf(" °----------------------------------------------------------°\n");

printf(" °----------------------------------------------------------°\n");

printf(" ! Introduce los siguientes datos: !\n\n");

printf(" ! 1) ID 2) Nombre !\n\n");

printf(" °----------------------------------------------------------°\n");

fflush( stdin ); /\* Limpiar buffer\*/

printf(" 1)"); /\* ID\*/

gets(id\_materia);

printf(" 2)"); /\* Nombre\*/

fflush( stdin );

gets(nombre);

system("cls");

printf(" °----------------------------------------------------------°\n"); /\*Muestra datos\*/

printf(" | ============== Alta Materias =============== |\n");

printf(" °----------------------------------------------------------°\n");

printf(" °----------------------------------------------------------°\n");

printf(" ! ¿Son correctos los datos? 1)si 2)no: !\n\n");

printf(" °----------------------------------------------------------°\n");

printf(" ID:\t%s\n",&id\_materia);

printf("\n Nombre:\t%s\n",&nombre);

printf("\n Opci%cn .-",162);

cin>>status;

}

system("cls");

insert\_materias(db,id\_materia,nombre);/\* Mqndamos a guardar datos a la bd\*/

system("pause");

}

/\*\*\*Esta función se encarga de presentar alta de planes\*\*\*/

/\*\*\*Contiene todo el proceso y redirecciona hacia las funciones de inserción y select con la base de datos\*\*\*/

void alta\_planes(sqlite3 \*db)

{

char id\_plan[40];

char nombre[40];

int num\_semestres=0;

int num\_materias;

char fecha[40];

char id\_materia[40];

int i=0,j=0;

bool correcto=false;

int status=2;

while(status != 1)

{ /\* Pide datos haste que esten correctos aprobados por el usuario\*/

system("cls");

printf(" °----------------------------------------------------------°\n"); /\*Menu Alta maestros\*/

printf(" | ============== Alta Planes =============== |\n");

printf(" °----------------------------------------------------------°\n");

printf(" °----------------------------------------------------------°\n");

printf(" ! Introduce los siguientes datos: !\n\n");

printf(" ! 1) ID 2) Nombre !\n\n");

printf(" ! 3) Numero\_Semestres 4) Fecha DD/MM/YY !\n\n");

printf(" °----------------------------------------------------------°\n");

fflush( stdin ); /\* Limpiar buffer\*/

printf(" 1)"); /\* ID\*/

gets(id\_plan);

printf(" 2)"); /\* Nombre\*/

fflush( stdin );

gets(nombre);

printf(" 3)"); /\* Numero de semetre\*/

fflush( stdin );

scanf("%d",&num\_semestres);

correcto=true;

do{ /\* Validacion fecha\*/

printf(" 4)");

fflush( stdin );

scanf("%s",fecha);

if(fecha[6]>='0' && fecha[6]<='9' && fecha[7]>='0' && fecha[7]<='9'){ /\*año validacion numeros\*/

if(fecha[3]>='0' && fecha[3]<'2' && fecha[4]>='0' && fecha[4]<='9'){ /\*mes validacion numeros\*/

if(fecha[3]=='1'){

if(!(fecha[4]>='0' && fecha[4]<='2')){ /\*mes validacion 12\*/

printf(" mes incorrecto\n");

}

}

if(fecha[0]>='0' && fecha[0]<'4' && fecha[1]>='0' && fecha[1]<='9'){ /\*dia validacion numero\*/

if(fecha[0]=='3'){

if(!(fecha[1]>='0' && fecha[1]<='1')){ /\*dia validacion 31\*/

printf(" dia incorrecto\n"); /\*dia incorrecto\*/

continue;

}

}

correcto=false; /\*validado correctamente\*/

}

else{

printf(" dia incorrecto\n"); /\*dia incorrecto\*/

}

}

else{

printf(" mes incorrecto\n"); /\*mes incorrecto numero \*/

}

}

else{

printf(" anio incorrecto\n"); /\*año incorrecto numero\*/

}

}while(correcto);

system("cls");

printf(" °----------------------------------------------------------°\n"); /\*Muestra datos\*/

printf(" | ============== Alta Planes =============== |\n");

printf(" °----------------------------------------------------------°\n");

printf(" °----------------------------------------------------------°\n");

printf(" ! ¿Son correctos los datos? 1)si 2)no: !\n\n");

printf(" °----------------------------------------------------------°\n");

printf(" ID PLAN:\t%s\n",&id\_plan);

printf("\n Nombre:\t%s\n",&nombre);

printf("\n Numero De Semestres:\t%d\n",num\_semestres);

printf("\n Fecha:\t%s\n",&fecha);

printf("\n Opci%cn .-",162);

cin>>status;

}

system("cls");

insert\_planes(db,id\_plan,nombre,num\_semestres,fecha);/\* Mandamos a guardar datos a la bd\*/

printf(" °----------------------------------------------------------°\n"); /\*Actualizacion lista\*/

printf(" | ============== Alta Planes =============== |\n");

printf(" °----------------------------------------------------------°\n");

printf(" °----------------------------------------------------------°\n");

printf(" ! Para dar de alta las materias del plan !\n\n");

printf(" ! es necesario conocer el ID de la materia. !\n\n");

printf(" °----------------------------------------------------------°\n");

//printf(" Ingrese el numero de semestres del plan.-\n "); /\* Numero de semestres del plan\*/

//scanf("%d",&num\_semestres);

while(i<num\_semestres)

{

printf(" Ingrese el numero de materias del semestre %d.-\n ",i+1); /\* para cada semestre\*/

scanf("%d",&num\_materias);

while(j<num\_materias)

{

printf(" ID materia.-\n "); /\* id de cada materia\*/

scanf("%s",&id\_materia);

insert\_mtr\_plan(db,id\_materia,id\_plan,num\_semestres); /\* guarda las materias del plan en la db\*/

j++;

}

i++;

}

system("cls");

printf(" °----------------------------------------------------------°\n"); /\*Datos ya guardados del plan\*/

printf(" | ============== Alta Planes =============== |\n");

printf(" °----------------------------------------------------------°\n");

printf(" °----------------------------------------------------------°\n");

printf(" ! Los datos del plan son: !\n\n");

printf(" °----------------------------------------------------------°\n");

query\_planes(db,id\_plan); /\* selecciona los datos del plan y los imprime de la db\*/

printf(" Ingresa un 1 para continuar\n");

scanf("%d",&status);

system("cls");

}

int main()

{

sqlite3 \*db; /\* Base de datos en sqlite3\*/

int res;

res=sqlite3\_open("test3.db", &db); /\* Se abre la db\*/

alta\_materias(db); /\*Invoca alta de planes\*/

alta\_planes(db); /\*Invoca alta de planes\*/

return 0;

}

# VII.- Code CheckList.

# Revisión codificación

|  |  |  |  |
| --- | --- | --- | --- |
| Student | Guillermo Vivaldo Vazquez | Date | 30/03/17 |
| Program | Alta De Planes | Program # | N/A |
| Instructor | PEBM | Language | C++ |

|  |  |
| --- | --- |
| Purpose | To guide you in conducting an effective code review |
| General | - Review the entire program for each checklist category; do not attempt to review for more than one category at a time!  - As you complete each review step, check off that item in the box at the right.  - Complete the checklist for one program or program unit before reviewing the next. |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Complete | Verify that the code covers all of the design. | MB | B | R | M |
| Includes | Verify that the includes are complete. | X |  |  |  |
| Initialization | Check variable and parameter initialization.  - at program initiation  - at start of every loop  - at class/function/procedure entry | X  X  X |  |  |  |
| Calls | Check function call formats.  - pointers  - parameters  - use of ‘&’ | X  X  X |  |  |  |
| Names | Check name spelling and use.  - Is it consistent?  - Is it within the declared scope?  - Do all structures and classes use ‘.’ reference? | X  X  X |  |  |  |
| Strings | Check that all strings are  - identified by pointers  - terminated by NULL | X  X |  |  |  |
| Pointers | Check that all  - pointers are initialized NULL  - pointers are deleted only after new  - new pointers are always deleted after use | X  X  X |  |  |  |
| Output Format | Check the output format.  - Line stepping is proper.  - Spacing is proper. | X  X |  |  |  |
| () Pairs | Ensure that () are proper and matched. |  |  |  |  |
| Logic Operators | - Verify the proper use of ==, =, ||, and so on.  - Check every logic function for (). | X  X |  |  |  |
| Line-by-line check | Check every line of code for  - instruction syntax  - proper punctuation |  | X  X |  |  |
| Standards | Ensure that the code conforms to the coding standards. |  |  |  |  |
| File Open and Close | Verify that all files are  - properly declared  - opened  - closed | X  X  X |  |  |  |

# VIII.- Reporte de resultados.

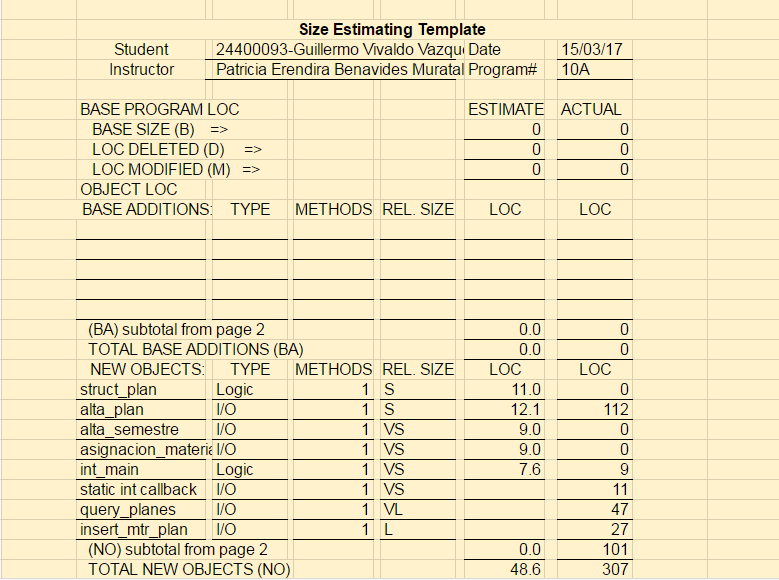
# Template de pruebas

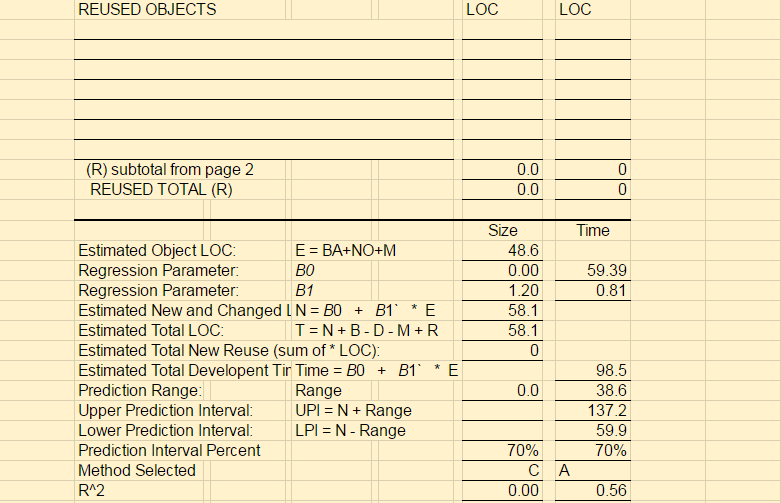
|  |  |
| --- | --- |
| Nombre de la Prueba | Prueba de Alta de Maestros |
| Caso de Uso | 4.0 |
| Objetivo de la Prueba | Verificar que los datos del plan sean correctos en la base de datos. |
| Descripción de la Prueba | Se ingresará al sistema como Administrador y se seleccionará la opción de “Alta Plan” |
| Test | 1.- Direccionarse a la opción de Alta Plan.  2.- Ingresar Datos del Plan.  3.- Se le asigna semestres.  4.- Se le asigna materias existentes o nuevas.  5.- Se confirma alta de plan. |
| Resultados Esperados | Mensaje de Éxito:  Datos Enviados: (Se mostrará los datos almacenados en variables locales)   1. Nombre de plan 2. Semestres 3. Materias   Datos Almacenados: (Se hará consulta a DB)   1. Nombre de plan 2. Semestres 3. Materias   Mensaje de Error:  Datos Enviados: (Se mostrará los datos almacenados en variables locales)   1. Nombre de plan 2. Semestres 3. Materias   “Error de consulta el Plan no se ha dado de alta” (Debido a que no se cargó correctamente el maestro)  En caso de tener un campo vacío:  “Error  ‘Campo Vacío’ Se encuentra vacío Favor de ingresar un valor” |
| Capturas de pantalla |  |
| Estado caso de prueba | Posibles estados:   * Ejecutado * Exitoso\*\*\*\* * Fallido * Frenado * Pendiente de ejecución * En construcción |
| Errores obtenidos | Se corrigió la impresión del formato de salida del número de semestre en el campo del plan y de la consulta de la materia asignada al plan. |
| Responsable de prueba | GVV |
| Comentarios | Sin comentarios |

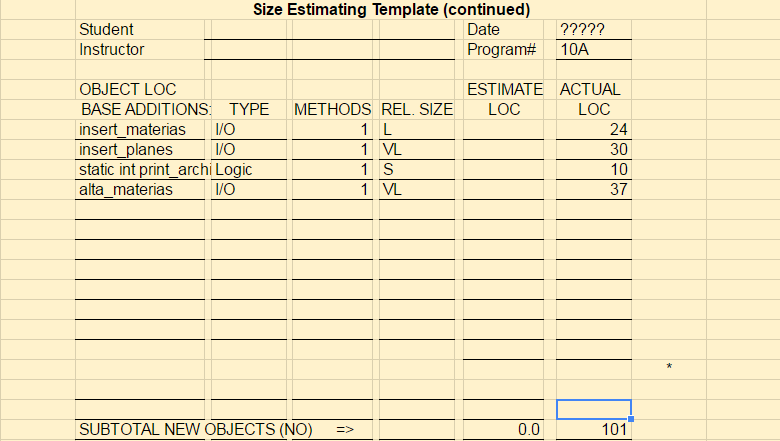
# IX.- PIP

|  |  |  |
| --- | --- | --- |
| Menú Planes | | |
| No. PIP | Descripción del Problema | Descripción de la propuesta. |
| 1 | No se sabía el uso de sqlite3 | Preparar una asesoría antes de empezar el módulo. |
| 2 | Mala planeación del módulo | No subestimar algún módulo porque puede ser más complejo de lo que parece. |
| Nombre Módulo | | |
| No. PIP | Descripción del Problema | Descripción de la propuesta. |

# X.- Size Estimating Template

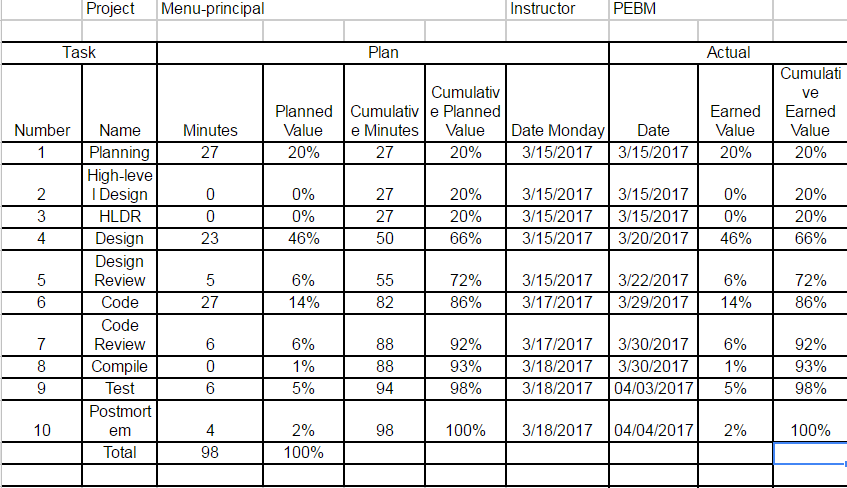






**XI.- Task and Schedule**

Task



Schedule