

Postmortem Cambios Maestros

06/05/17

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\_Cambios\_Maestros-060517.docx | 0.1 | 06/05/17 | 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**

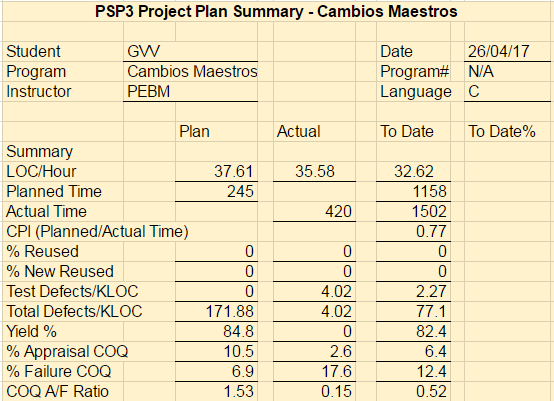
[**IX.- PIP**](#_4ocjjwajumc0) **4**

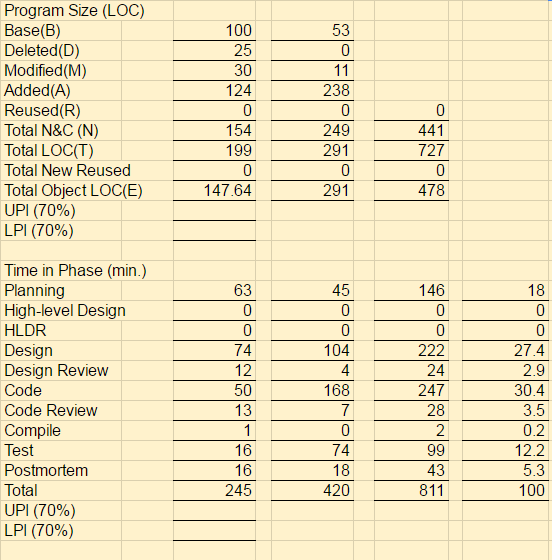
[**X.- Size Estimating Template**](#_w1ych8jaj7as) **5**

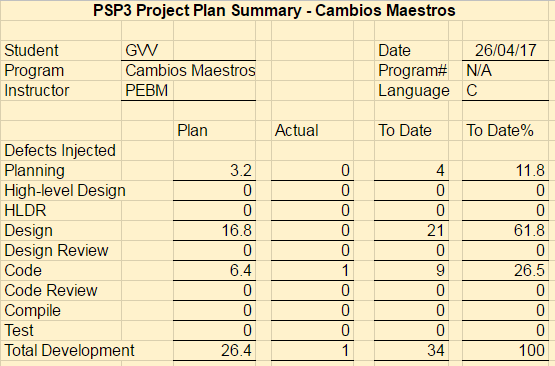
[**XI.- Task and Schedule**](#_195jonwtvlxs) **5**

# I.- Forma PSP Project Plan Summary.

Se inició el proceso el 26/04/17

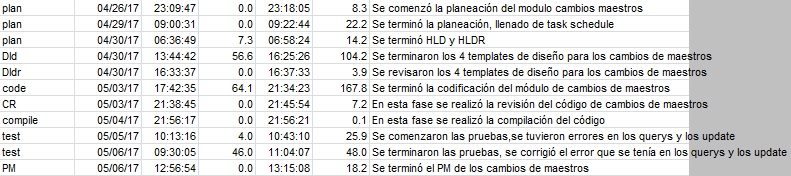






# 

# II.- Time Recording Log.



# III.- Defect Recording Log.



# IV.- Diseño.

**State Specification Template**

|  |  |  |  |
| --- | --- | --- | --- |
| Student | GVV | Date | 30/04/17 |
| Program | Cambios Maestros | Program# |  |
| Instructor | PEBM | Language | C |
| Object |  | Routine |  |

# 

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **State Name** | | | | **Description** | | | |
| Inicio | | | | Inicia el programa. | | | |
| Cambios Maestro | | | | Función para empezar el cambio de Maestros. | | | |
| Insert Maestros | | | | Realiza la inserción de los datos nuevos del maestro y los almacena en la base de datos el alta. | | | |
| Insert Materias-Planes | | | | Realiza la inserción de los datos nuevos de los planes y materias que el maestro puede impartir. | | | |
| Query Maestros | | | | Imprime desde la base de datos el registro (consulta). | | | |
| Salir | | | | Termina el programa. | | | |
|  | | | |  | | | |
| **Function/Parameter** | | | | **Description** | | | |
| printf() | | | | Imprime datos en la pantalla | | | |
| scanf() | | | | Lee datos de la pantalla | | | |
| cambio\_maestro() | | | | Enlace para el alta de maestros | | | |
| insert\_mtr() | | | | Guarda maestro en base de datos | | | |
| insert\_mtr\_plan() | | | | Guarda vinculación maestros- materias | | | |
| Return() | | | | Regresa un valor. | | | |
| Query\_Maestros | | | | Imprime registro desde base de datos | | | |
| cantidad\_materias | | | | Para determinar cuántas materias se elegirán. | | | |
| **States/Next States** | | | | **Transition Condition** | | **Action** | |
| Inicio | | | |  | |  | |
|  | Cambios Maestro | | | Inicia el programa | | cambio\_maestro() | |
| Cambios Maestros | | | |  | |  | |
|  | Insert Maestros | | | ID == id\_maestro | | insert\_mtr() | |
|  | Insert Materias-Planes | | | mientras cantidad\_materias == i con i=0 para i++ && ID == id\_maestro | | insert\_mtr\_plan() | |
|  | Query Maestros | | | Ya se insertó en la base de datos | | Query\_Maestros() | |
|  | Salir | | | Se imprimió el Alta maestros | | Return() | |

# Logic Specification Template

|  |  |  |  |
| --- | --- | --- | --- |
| Student | GVV | Date | 30/04/17 |
| Program | Cambios Maestros | Program # |  |
| Instructor | PEBM | Language | C |
| Object |  | Function |  |

INCLUDES:

#include <stdio.h>

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

#include<string.h>

#include “sqlite3.h”

TYPE DEFINITIONS:

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

|  |  |
| --- | --- |
| Logic reference numbers | Program logic, in pseudocode |
| 1 | char \*sql |
|  | char \*err |
|  | int rc |
|  | const char\* data = "Datos Maestro" |
|  | asigna a sql=”SELECT \* FROM MAESTROS WHERE Id\_Ma =’”\id\_maestro”’);”; |
|  | 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 IMPARTE WHERE Id\_Maestro =’”\id\_maestro”’);”; |
|  | 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:** | insert\_mtr() |
| **Reference:** | Functional template |
|  |  |

|  |  |
| --- | --- |
| Logic reference numbers | Program logic, in pseudocode |
| 1 | char \*sql |
|  | char \*err |
|  | int rc |
|  | asigna a sql=”INSERT INTO MAESTROS (Id\_Ma, Nombre, ApPaterno, ApMaterno, FNacimiento, Direccion, Telefono, Email) “ \ “VALUES (‘“ \id\_materia \ ”’,’” \ nombre \ ”’,’” \ ap\_paterno \ ”’,’” \ ap\_materno \ ”’,’” \ fecha\_nac \ ”’,’” \ direccion \ ”’,’” \ telefono \ ”’,’” \ email \ “‘);”; |
|  | 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:** | insert\_mtr\_plan |
| **Reference:** | Functional template |
|  |  |

|  |  |
| --- | --- |
| Logic reference numbers | Program logic, in pseudocode |
| 1 | char \*sql |
|  | char \*err |
|  | asigna a sql=”INSERT INTO IMPARTE (ID\_Maestro, ID\_Materia, Status) “ \ “VALUES (‘“ \ id\_maestro ”’,’” \ id\_materia ”’,’” \ status “‘);”; |
|  | 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:** | Cambio\_maestro() |
| **Reference:** | Functional template |
|  |  |

|  |  |
| --- | --- |
| Logic reference numbers | Program logic, in pseudocode |
| 1 | char \*sql |
|  | char \*err |
|  | int rc |
|  | leer id\_maestro |
|  | query\_maestros() |
|  | Selecciona que deseas cambiar |
|  | Leer opcion |
|  | Sí opción == 1 entonces :: insert\_mtr() |
|  | Si opción == 2 entonces :: insert\_mtr\_plan() |
|  | Si opción == 3 entonces :: no realizamos nada y sales al menú. |

|  |  |
| --- | --- |
| **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 | Cambio\_maestro(); |
| 3 | System(“pause”); |
| 4 | Return 0; |
| 5 | FIN |

# Functional Specification Template

|  |  |  |  |
| --- | --- | --- | --- |
| Student | GVV | Date | 30/04/17 |
| Program | Cambios Maestros | Progra# |  |
| Instructor | PEBM | Language | C |

|  |  |  |  |
| --- | --- | --- | --- |
| Object/Class Name  main | | Parent Classes  Base | Attributes  Op (1-3) |
|  | Method Declaration | Method External Specification | |
|  | insert\_mtr\_plan | Guarda los datos del plan y de las materias que el maestro puede impartir | |
|  | insert\_mtr | Guarda los datos del maestro en la base de datos. | |
|  | Cambio\_maestro | Función que llevará a cabo el ingreso de nuevos datos para cierto maestro los cuales se insertarán en su registro correspondiente de la base de datos. | |
|  | query\_maestros | Muestra que los valores han sido ingresados correctamente a la base (consulta) | |
|  | Int\_callback () | Esta función se encarga de imprimir los datos encontrados en la base de datos | |
|  | | | |
|  | | | |
|  | | | |
|  | | | |
|  | | | |
|  | | | |
|  | | | |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Class Name** | | void insert\_mtr() | | | | |
| **Parent Class** | | N/A | | | | |
|  | | | | | | |
| **Attributes** | | | | | | |
|  | **Declaration** | | | **Description** | | |
|  | sqlite3 \*db | | | Apuntador a la base de datos. | | |
|  | char \*sql | | | Corresponde a la cadena de consulta que se va a hacer en la base de datos. | | |
|  | char \*Err | | | Corresponde a el error si existe que regresa al momento de hacer la consulta de la base de datos. | | |
|  | int rc | | | Para guardar el resultado de la consulta en la base de datos. | | |
|  | string ap\_materno | | | apellido paterno del maestro | | |
|  | string ap\_paterno | | | apellido materno del maestro | | |
|  | string nombre | | | nombre del maestro | | |
|  | string fecha\_nac | | | fecha de nacimiento del maestro | | |
|  | string direccion | | | dirección del maestro | | |
|  | string telefono | | | teléfono del maestro | | |
|  | string email | | | email del maestro | | |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Items** | | | | | | |
|  | **Declaration** | | | **Description** | | |
|  | Insert\_mtr() | | | Guarda los datos del maestro en la base de datos. | | |
|  | Insert\_mtr\_plan() | | | Guarda los datos del plan y de las materias del maestro | | |
|  | Query\_Maestros() | | | Imprime los datos de un maestro | | |
|  | printf() | | | Para imprimir directamente en la interfaz de usuario. | | |
|  | scanf() | | | Para leer datos de la interfaz de usuario. | | |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Class Name** | | int insert\_mtr\_plan() | | | | |
| **Parent Class** | | N/A | | | | |
|  | | | | | | |
| **Attributes** | | | | | | |
|  | **Declaration** | | | **Description** | | |
|  | sqlite3 \*db | | | Apuntador a la base de datos. | | |
|  | char \*sql | | | Corresponde a la cadena de consulta que se va a hacer en la base de datos. | | |
|  | char \*Err | | | Corresponde a el error si existe que regresa al momento de hacer la consulta de la base de datos. | | |
|  | int rc | | | Para guardar el resultado de la consulta en la base de datos. | | |
|  | string id\_materia | | | Identificador de la materia | | |
|  | string id\_maestro | | | Apellido materno del maestro | | |
|  | int status | | | Para definir si el maestro está impartiendo o no la materia. | | |
|  | | | | | | |
| **Items** | | | | | | |
|  | **Declaration** | | | **Description** | | |
|  | callback() | | | Regresa el mensaje que de error de la base de datos. | | |
|  | sqlite3\_exec() | | | Ejecutar una acción en la base de datos | | |
|  | SQLITE\_OK | | | Parámetro de comparación correcto de instrucción a la base de datos. | | |
|  | printf() | | | Para imprimir directamente en la interfaz de usuario. | | |
|  | sqlite3\_free() | | | Para limpiar el mensaje de error. | | |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Class Name** | | void Query\_Maestros() | | | | |
| **Parent Class** | | N/A | | | | |
|  | | | | | | |
| **Attributes** | | | | | | |
|  | **Declaration** | | | **Description** | | |
|  | sqlite3 \*db | | | Apuntador a la base de datos. | | |
|  | char \*sql | | | Corresponde a la cadena de consulta que se va a hacer en la base de datos. | | |
|  | char \*Err | | | Corresponde a el error si existe que regresa al momento de hacer la consulta de la base de datos. | | |
|  | int rc | | | Para guardar el resultado de la consulta en la base de datos. | | |
|  | string id\_maestro | | | ID único (identificador del maestro) | | |
|  | const char\* data = "Consulta" | | | Para indicar el tipo de ejecución. | | |
|  | | | | | | |
| **Items** | | | | | | |
|  | **Declaration** | | | **Description** | | |
|  | callback() | | | Regresa el mensaje que de error de la base de datos. | | |
|  | sqlite3\_exec() | | | Ejecutar una acción en la base de datos | | |
|  | SQLITE\_OK | | | Parámetro de comparación correcto de instrucción a la base de datos. | | |
|  | printf() | | | Para imprimir directamente en la interfaz de usuario. | | |
|  | sqlite3\_free() | | | Para limpiar el mensaje de error. | | |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Class Name** | | static int callback() | | | | |
| **Parent Class** | | N/A | | | | |
|  | | | | | | |
| **Attributes** | | | | | | |
|  | **Declaration** | | | **Description** | | |
|  | void \*data | | | Función hacia los datos. | | |
|  | int argc | | | Número de datos encontrados en la selección de la base de datos. | | |
|  | char \*\*argv | | | Lista de datos resultantes de la base de datos en la parte de datos | | |
|  | char \*\*azColName | | | Datos del nombre de la columna de los datos de la base de datos. | | |
|  | int i | | | Contador del for | | |
|  | | | | | | |
| **Items** | | | | | | |
|  | **Declaration** | | | **Description** | | |
|  | printf() | | | Imprime en cadena y variables determinadas. | | |
|  | return() | | | Regresa un valor determinado. | | |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Class Name** | | void Cambio\_maestro() | | | | |
| **Parent Class** | | N/A | | | | |
|  | | | | | | |
| **Attributes** | | | | | | |
|  | **Declaration** | | | **Description** | | |
|  | sqlite3 \*db | | | Apuntador a la base de datos. | | |
|  | char \*sql | | | Corresponde a la cadena de consulta que se va a hacer en la base de datos. | | |
|  | char \*Err | | | Corresponde a el error si existe que regresa al momento de hacer la consulta de la base de datos. | | |
|  | int rc | | | Para guardar el resultado de la consulta en la base de datos. | | |
|  | string id\_maestro | | | ID maestro (identificador único) | | |
|  | const char\* data = "Consulta" | | | Para indicar el tipo de ejecución. | | |
|  | | | | | | |
| **Items** | | | | | | |
|  | **Declaration** | | | **Description** | | |
|  | callback() | | | Regresa el mensaje que de error de la base de datos. | | |
|  | sqlite3\_exec() | | | Ejecutar una acción en la base de datos | | |
|  | SQLITE\_OK | | | Parámetro de comparación correcto de instrucción a la base de datos. | | |
|  | insert\_mtr | | | Función que realizará la inserción de los nuevos datos en el registro del maestro dado en la base de datos | | |
|  | insert\_mtr\_plan | | | Función que realizará la inserción de los nuevos datos del plan y de las materias en el registro del maestro dado en la base de datos. | | |
|  | printf() | | | Para imprimir directamente en la interfaz de usuario. | | |
|  | sqlite3\_free() | | | Para limpiar el mensaje de error. | | |

# Operational Scenario Template

|  |  |  |  |
| --- | --- | --- | --- |
| Student | GVV | Date | 30/04/17 |
| Program | Cambios Maestros | Program # |  |
| Instructor | PEBM | Language | C |

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

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Scenario Number: 1 | | | User Objective: Solicitar al usuario el ID del maestro, y mediante ese ID preguntar qué cambios se le desean realizar al maestro (Nombres, Apellido paterno, Apellido materno, Fecha de nacimiento, Dirección, Correo electrónico, Teléfono, Materias que imparte., Materias posibles a impartir.) | |
| 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 | Muestra al usuario un menú con los campos que puede cambiar | | El menú es similar al que se realiza en las altas de maestros |
| Usuario | 4 | El usuario ingresa una opción para realizar el cambio | | Pueden realizarse cambios a los campos indicados en la parte superior (User objective) |
| Sistema | 5 | Se solicita al usuario ingresar los datos nuevos que serán los que reemplazarán a los que se tenían anteriormente. | | Esto es prácticamente un update sobre los datos que se tenían anteriormente para ese usuario en la base de datos. |
| Sistema | 6 | Se mostrará al usuario los datos que ingresó y el deberá confirmar si son correctos o no. | |  |
| Usuario | 7 | El usuario elige la opción 1 que confirma que los datos son correctos y se hacen los cambios que indicó para ese maestro en la base de datos y se procede ir al paso 9. | |  |
| Sistema | 8 | El usuario elige la opción 2 que significa que se volverán a repetir los pasos del 5 al 6. | |  |
| Sistema | 9 | El sistema pregunta al usuario si desea realizar algún otro cambio | |  |
| 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 #: Cambios Maestros

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 \*/

/\*Componente: Cambios Maestros \*/

/\*Fecha: 03/05/17 \*/

/\*Descripción: Codificación que muestra los cambios para los maestros\*/

/\*Versión: 0.1 \*/

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

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

/\* El programa contiene lo siguiente: \*/

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

/\*void query\_maestros(sqlite3 \*db,char\* id\_maestro) \*/

/\*void update\_maestros(sqlite3 \*db,char\* type,char\* value,char\* id\_maestro)\*/

/\*void update\_imparte(sqlite3 \*db,char\* type,char\* value,char\* id\_maestro)\*/

/\*void cambios\_maestros(sqlite3 \*db)\*/

/\*int main() \*/

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

#include <cstdio>

#include <cstdlib>

#include <iostream>

#include "sqlite3.h"

using namespace std;

/\*\*\*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 se encarga de buscar los valores del maestro en la base de datos\*\*\*/

void query\_maestros(sqlite3 \*db,char\* id\_maestro){

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 Maestros where ID\_Ma LIKE @Id\_M;"; /\* Selecciona a los maestros con un id especifico sentencia sql\*/

rc=sqlite3\_prepare\_v2(db,sql,-1,&res,0); /\* Verifica si es válida la instrucción 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 maestro\n"); /\* Se aprueba la sentencia sql\*/

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

sqlite3\_bind\_text(res, idx, id\_maestro,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 instrucción sql, valores del maestro\*/

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

printf("\t\t 2)Apellido paterno: %s\n", sqlite3\_column\_text(res, 2));

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

printf("\t\t 4)Fecha: %s\n", sqlite3\_column\_text(res, 4));

printf("\t\t 5)Direccion: %s\n", sqlite3\_column\_text(res, 5));

printf("\t\t 6)Telefono: %s\n", sqlite3\_column\_text(res, 6));

printf("\t\t 7)Email: %s\n", sqlite3\_column\_text(res, 7));

printf("\t\t 8)RFC: %s\n", sqlite3\_column\_text(res, 0));

}

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

sql="SELECT ID\_Materia,Status FROM Imparte where ID\_Maestro LIKE @Id\_M;";/\* Instrucción sql para la selección de materias en tabla imparte\*/

rc=sqlite3\_prepare\_v2(db,sql,-1,&res,0); /\* Verifica si es correcta la instrucción sql\*/

if(rc!=SQLITE\_OK){

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

}

else{

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

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

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

}

step = sqlite3\_step(res);

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

printf(" %s\t 9) %s\n", sqlite3\_column\_text(res, 0),sqlite3\_column\_text(res, 1));

step=sqlite3\_step(res);

}

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

}

/\*\*\*Esta función se encarga de actualizar al alumno en la base de datos\*\*\*/

void update\_maestros(sqlite3 \*db,int type,char\* value,char\* id\_maestro)

{

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

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

int rc;

sqlite3\_stmt \*res;

switch(type){

case 1:{

sql="UPDATE Maestros SET Nombre = @Value where ID\_Ma LIKE @Id\_M;"; /\* Selecciona a los alumnas con un id especifico sentencia sql\*/

break;

}

case 2:{

sql="UPDATE Maestros SET ApPaterno = @Value where ID\_Ma LIKE @Id\_M;"; /\* Selecciona a los alumnas con un id especifico sentencia sql\*/

break;

}

case 3:{

sql="UPDATE Maestros SET ApMaterno = @Value where ID\_Ma LIKE @Id\_M;"; /\* Selecciona a los alumnas con un id especifico sentencia sql\*/

break;

}

case 4:{

sql="UPDATE Maestros SET FNacimiento = @Value where ID\_Ma LIKE @Id\_M;"; /\* Selecciona a los alumnas con un id especifico sentencia sql\*/

break;

}

case 5:{

sql="UPDATE Maestros SET Direccion = @Value where ID\_Ma LIKE @Id\_M;"; /\* Selecciona a los alumnas con un id especifico sentencia sql\*/

break;

}

case 6:{

sql="UPDATE Maestros SET Telefono = @Value where ID\_Ma LIKE @Id\_M;"; /\* Selecciona a los alumnas con un id especifico sentencia sql\*/

break;

}

case 7:{

sql="UPDATE Maestros SET Email = @Value where ID\_Ma LIKE @Id\_M;"; /\* Selecciona a los alumnas con un id especifico sentencia sql\*/

break;

}

case 8:{

sql="UPDATE Maestros SET RFC = @Value where ID\_Ma LIKE @Id\_M;";

break;

}

}

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

if(rc!=SQLITE\_OK){

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

}

else{

printf("\t\t Actualizando!!!\n"); /\* Se aprueba la sentencia sql\*/

int idx = sqlite3\_bind\_parameter\_index(res, "@Id\_M"); /\* Se busca en qué lugar está el identificador a cambiar id\*/

int idv = sqlite3\_bind\_parameter\_index(res, "@Value"); /\* Se busca en qué lugar está el identificador a cambiar value\*/

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

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

}

int step = sqlite3\_step(res);

if (step == SQLITE\_ROW) { /\* indica el resultado de la instrucción sql, valores del maestro\*/

printf("\t\t Actualizado!!!\n"); /\* Se aprueba la sentencia sql ya echa\*/

}

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

}

/\*\*\*Esta función se encarga de actualizar las materias que imparte el maestro en la base de datos\*\*\*/

void update\_imparte(sqlite3 \*db,int type,char\* value,char\* id\_maestro,char\* id\_materia)

{

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

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

int rc;

sqlite3\_stmt \*res;

switch(type){

case 9:{

sql="UPDATE Imparte SET Status = @Value where ID\_Maestro LIKE @Id\_M and ID\_Materia LIKE @Id\_Mat;"; /\* Selecciona a los alumnas con un id especifico sentencia sql\*/

break;

}

}

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

if(rc!=SQLITE\_OK){

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

}

else{

printf("\t\t Actualizando!!!\n"); /\* Se aprueba la sentencia sql\*/

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

int idt = sqlite3\_bind\_parameter\_index(res, "@Id\_Mat"); /\* Se busca en qué lugar está el identificador a cambiar id\*/

int idv = sqlite3\_bind\_parameter\_index(res, "@Value"); /\* Se busca en qué lugar está el identificador a cambiar value\*/

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

sqlite3\_bind\_text(res, idt, id\_materia,40,SQLITE\_STATIC);

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

}

int step = sqlite3\_step(res);

if (step == SQLITE\_ROW) { /\* indica el resultado de la instrucción sql, valores del maestro\*/

printf("\t\t Actualizado!!!\n"); /\* Se aprueba la sentencia sql ya echa\*/

}

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

}

/\*\*\*Esta función es la función principal del cambio de maestros\*\*\*/

void cambios\_maestros(sqlite3 \*db){

char opcion;

char valor[40];

char id[40];

char id\_mat[40];

char status[40];

do{

system("cls");

printf("\t\to----------------------------------------------------------o\n"); /\* Menú de cambios alumnos\*/

printf("\t\t| ================ Cambios Maestros =================== |\n");

printf("\t\to----------------------------------------------------------o\n");

printf("\t\to----------------------------------------------------------o\n");

printf("\t\t| ¿Que desea hacer? |\n");

printf("\t\t| 1)Buscar Maestro |\n");

printf("\t\t| 2)Regresar Men%c Maestros |\n",163);

printf("\t\t| Ingresa la opci%cn que desea: |\n",162);

printf("\t\to----------------------------------------------------------o\n\t\t");

cin>>opcion;

switch(opcion){

case '1':{

do{

printf("\t\t Ingresa el ID.-"); /\* Busca a un alumno para hacer cambios\*/

scanf("%s",&id);

system("cls");

printf("\t\to----------------------------------------------------------o\n");

query\_maestros(db,id); /\* Imprime datos del alumno\*/

printf("\n");

printf("\t\t| 0)Salir |\n");

printf("\t\t| Ingresa la opci%cn que desea cambiar: |\n",162);

printf("\t\to----------------------------------------------------------o\n\t\t");

cin>>opcion;

switch(opcion){

case '1':{

printf("\t\t Ingresa el nuevo valor.- "); /\* Nuevo valor a actualizar en la base de datos\*/

getchar();

gets(valor);

update\_maestros(db,1,valor,id); /\* Actualiza nombre\*/

system("pause");

system("cls");

break;

}

case '2':{

printf("\t\t Ingresa el nuevo valor.- "); /\* Nuevo valor a actualizar en la base de datos\*/

scanf("%s",valor);

update\_maestros(db,2,valor,id); /\* Actualiza ap paterno\*/

system("pause");

system("cls");

break;

}

case '3':{

printf("\t\t Ingresa el nuevo valor.- "); /\* Nuevo valor a actualizar en la base de datos\*/

scanf("%s",valor);

update\_maestros(db,3,valor,id); /\* Actualiza ap materno\*/

system("pause");

system("cls");

break;

}

case '4':{

printf("\t\t Ingresa el nuevo valor.- "); /\* Nuevo valor a actualizar en la base de datos\*/

scanf("%s",valor);

update\_maestros(db,4,valor,id); /\* Actualiza fecha nacimiento\*/

system("pause");

system("cls");

break;

}

case '5':{

printf("\t\t Ingresa el nuevo valor.- "); /\* Nuevo valor a actualizar en la base de datos\*/

scanf("%s",valor);

update\_maestros(db,5,valor,id); /\* Actualiza dirección\*/

system("pause");

system("cls");

break;

}

case '6':{

printf("\t\t Ingresa el nuevo valor.- "); /\* Nuevo valor a actualizar en la base de datos\*/

scanf("%s",valor);

update\_maestros(db,6,valor,id); /\* Actualiza teléfono\*/

system("pause");

system("cls");

break;

}

case '7':{

printf("\t\t Ingresa el nuevo valor.- "); /\* Nuevo valor a actualizar en la base de datos\*/

scanf("%s",valor);

update\_maestros(db,7,valor,id); /\* Actualiza email\*/

system("pause");

system("cls");

break;

}

case '8':{/\* Actualiza nombre\*/

printf("\t\t Ingresa el nuevo valor.- "); /\* Nuevo valor a actualizar en la base de datos\*/

scanf("%s",valor);

update\_maestros(db,8,valor,id); /\* Actualiza RFC\*/

system("pause");

system("cls");

break;

}

case '9':{

getchar();

printf("\t\t Ingresa el nuevo valor.- "); /\* Nuevo valor a actualizar en la base de datos\*/

gets(valor);

printf("\t\t Ingresa el ID De La Materia.-");

gets(id\_mat);

update\_imparte(db,9,valor,id,id\_mat); /\* Actualiza semestre\*/

system("pause");

system("cls");

break;

}

case '0':{

printf("\t\t Regresando a men%c\n",163);

break;

}

default:{

printf("\t\t Valor incorrecto!!! intenta otro \n");

break;

}

}

}while(opcion!='0');

break;

}

case '2':{

printf("\t\t Regresando a men%c\n",163);

break;

}

default:{

printf("\t\t Valor incorrecto!!! intenta otro \n");

break;

}

}

}while(opcion !='2');

}

int main()

{

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

int res;

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

cambios\_maestros(db); /\*A cambios maestros\*/

return 0;

}

# VII.- Code CheckList.

# Revisión codificación

|  |  |  |  |
| --- | --- | --- | --- |
| Student | GVV | Date | 03/05/17 |
| Program | Cambios Maestros | 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. | X |  |  |  |
| File Open and Close | Verify that all files are  - properly declared  - opened  - closed | X  X  X |  |  |  |

# VIII.- Reporte de resultados.

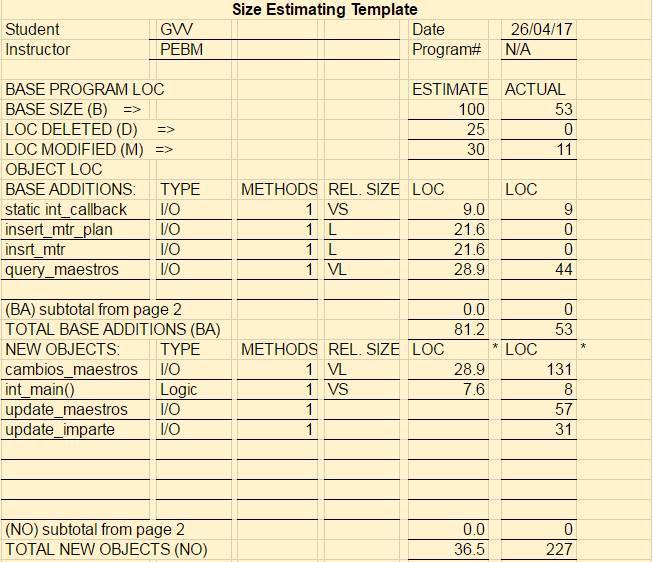
# Template de pruebas

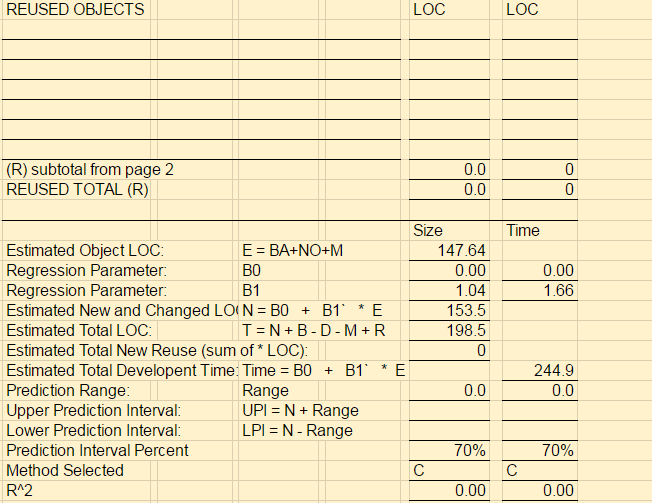
|  |  |
| --- | --- |
| Nombre de la Prueba | Prueba de cambios de maestros |
| Caso de Uso | 3.2 |
| Objetivo de la Prueba | Verificar en DB que la modificación realizada sea correcta. |
| Descripción de la Prueba | Se ingresa a Sistema como usuario Administrador y se realiza por medio de la opción de Cambio Maestro. |
| Test | 1.- Direccionarse a la opción Cambio Maestro.  2.- Ingresar ID maestro.  3.- Seleccionar Campo a modificar.  4.- Ingresar datos nuevos.  5.- Confirmar  6.- Se repite el paso 3 hasta seleccionar la opción salir. |
| Resultados Esperados | Mensaje de Éxito.  Datos Maestro (Se hará consulta a DB)   1. Nombres 2. Apellido paterno 3. Apellido materno 4. Fecha de nacimiento 5. Dirección 6. Correo electrónico 7. Teléfono   Mensaje de Error:  “No se ha modificado el valor ‘Campo a Cambiar’” |
| Capturas De Pantalla |  |
| Estado caso de prueba | Se realizan los cambios correctamente, se probó con el maestro con RFC VIVAG |
| Errores obtenidos | Los querys estaban mal diseñados y no imprimían los valores almacenados en la BD, los update se optimizaron porque no se actualizaban los campos indicados. |
| Responsable de prueba | GVV |
| Comentarios | Sin comentarios |

# IX.- PIP

|  |  |  |
| --- | --- | --- |
| Menú Reportes Planes | | |
| No. PIP | Descripción del Problema | Descripción de la propuesta. |
| 1 | Se llevó demasiado tiempo en codificación | Planear un tiempo mayor en esta tarea |
| 2 | Problemas y detección de errores en la fase de pruebas. | Analizar el código bien con el encargado de desarrollo en la fase de codificación y evitar que se lleve demasiado tiempo en pruebas. |
| Nombre Módulo | | |
| No. PIP | Descripción del Problema | Descripción de la propuesta. |

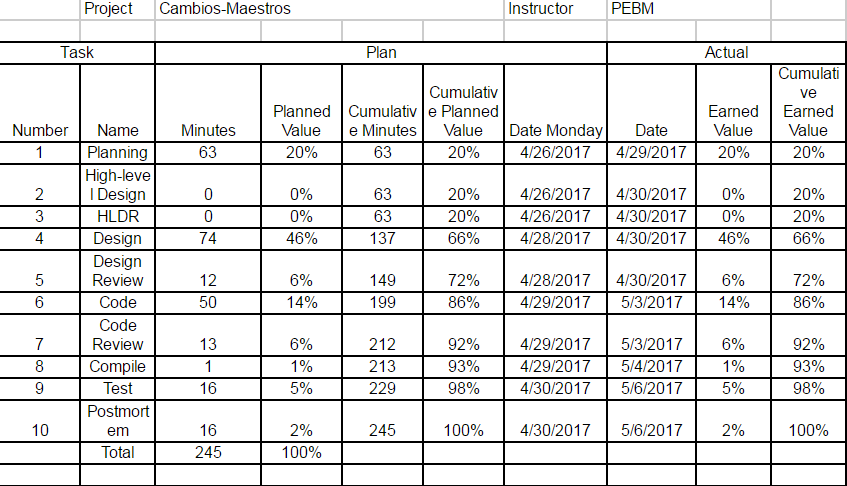
# X.- Size Estimating Template





**XI.- Task and Schedule**

Task



Schedule

