

Postmortem Alta Maestros

30/03/2017

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\_Maestros-300317.docx | 0.1 | 30/03/20117 | JAGC | 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**

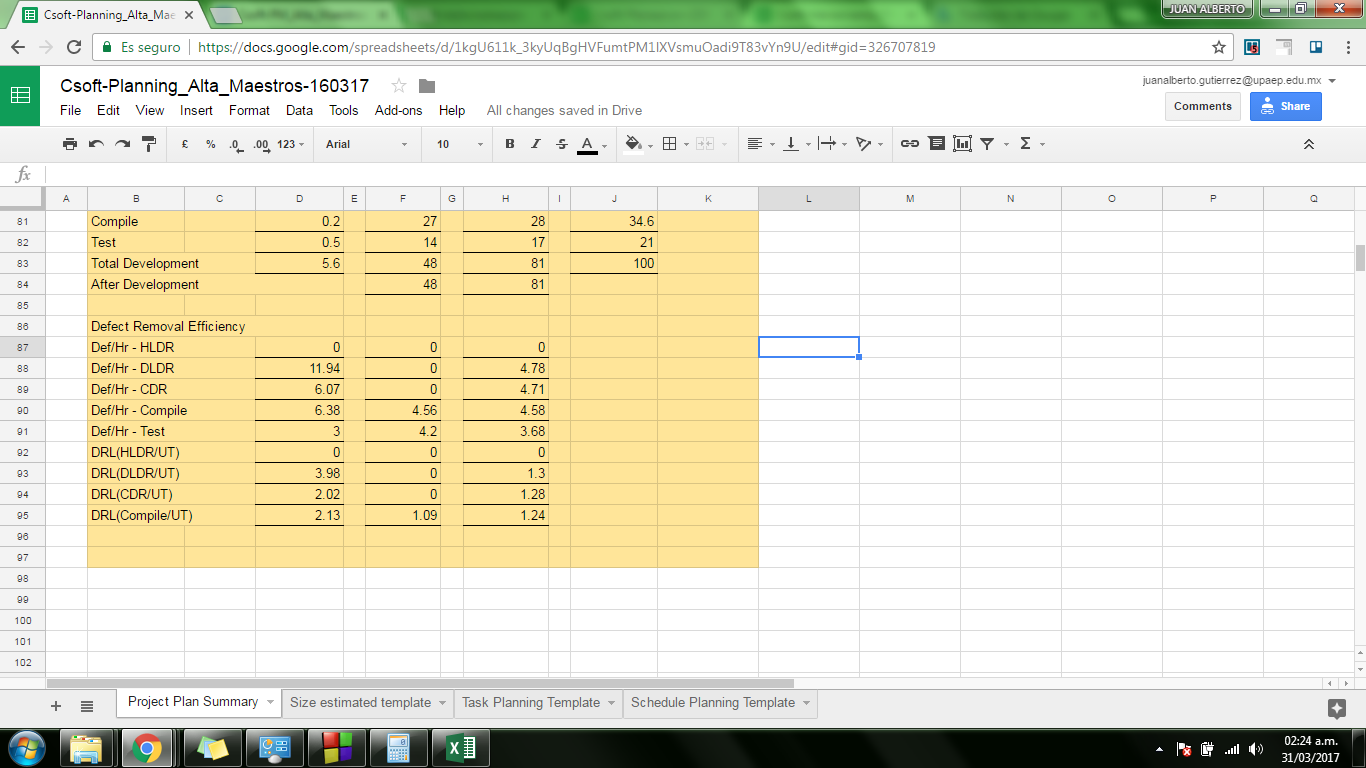
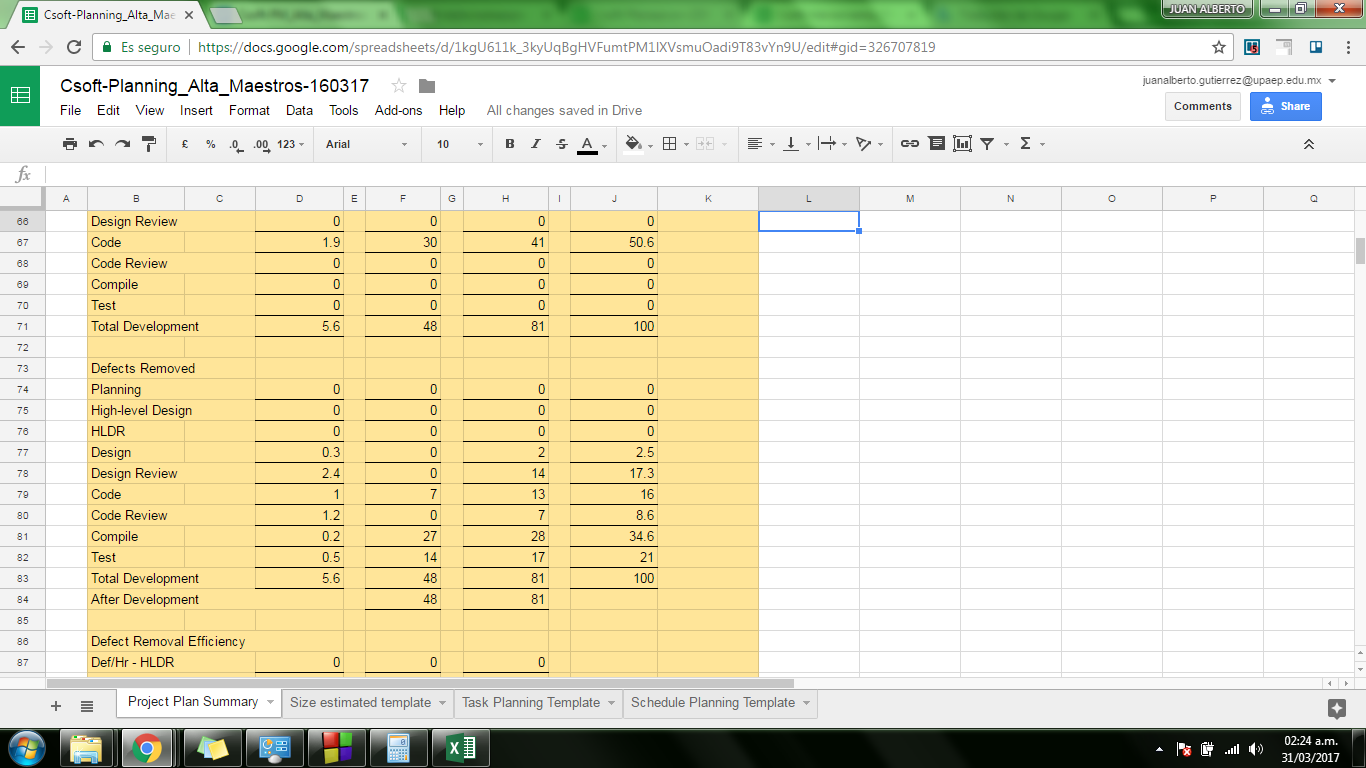
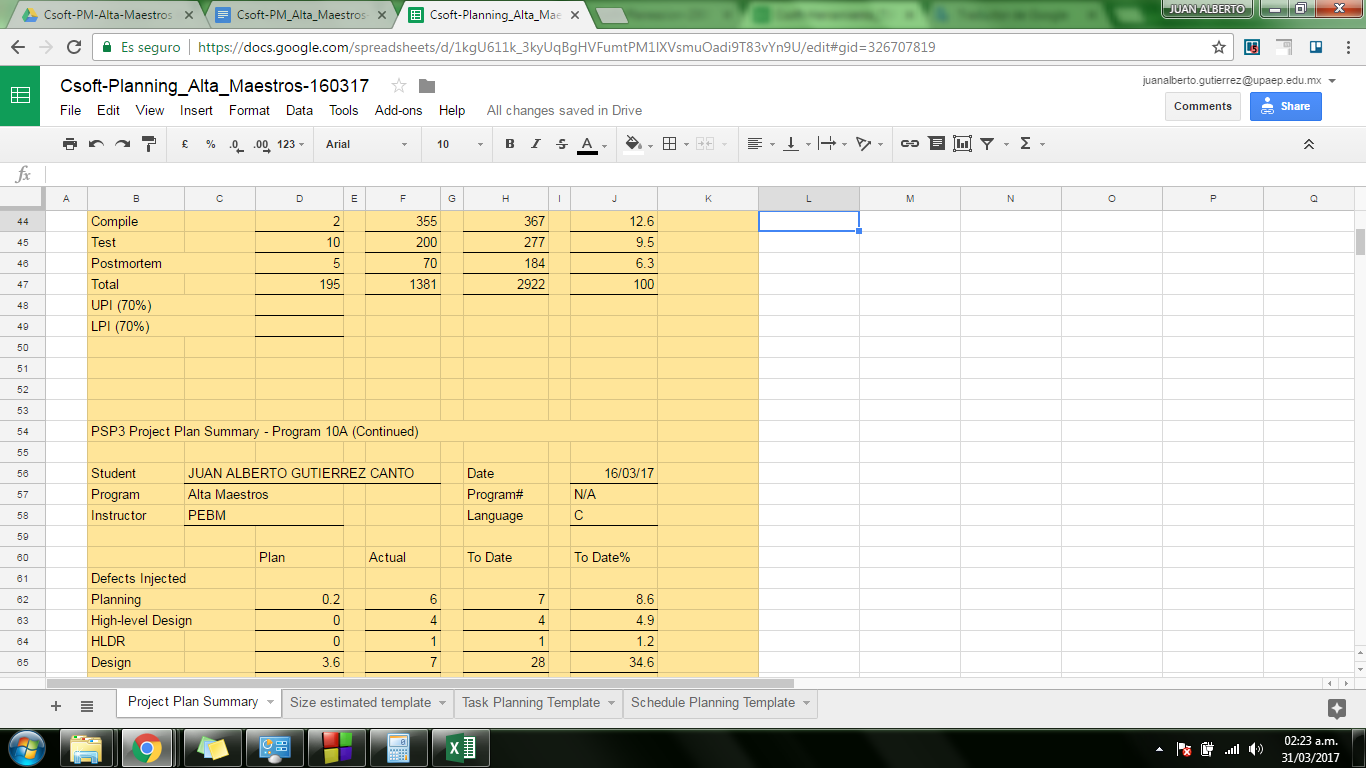
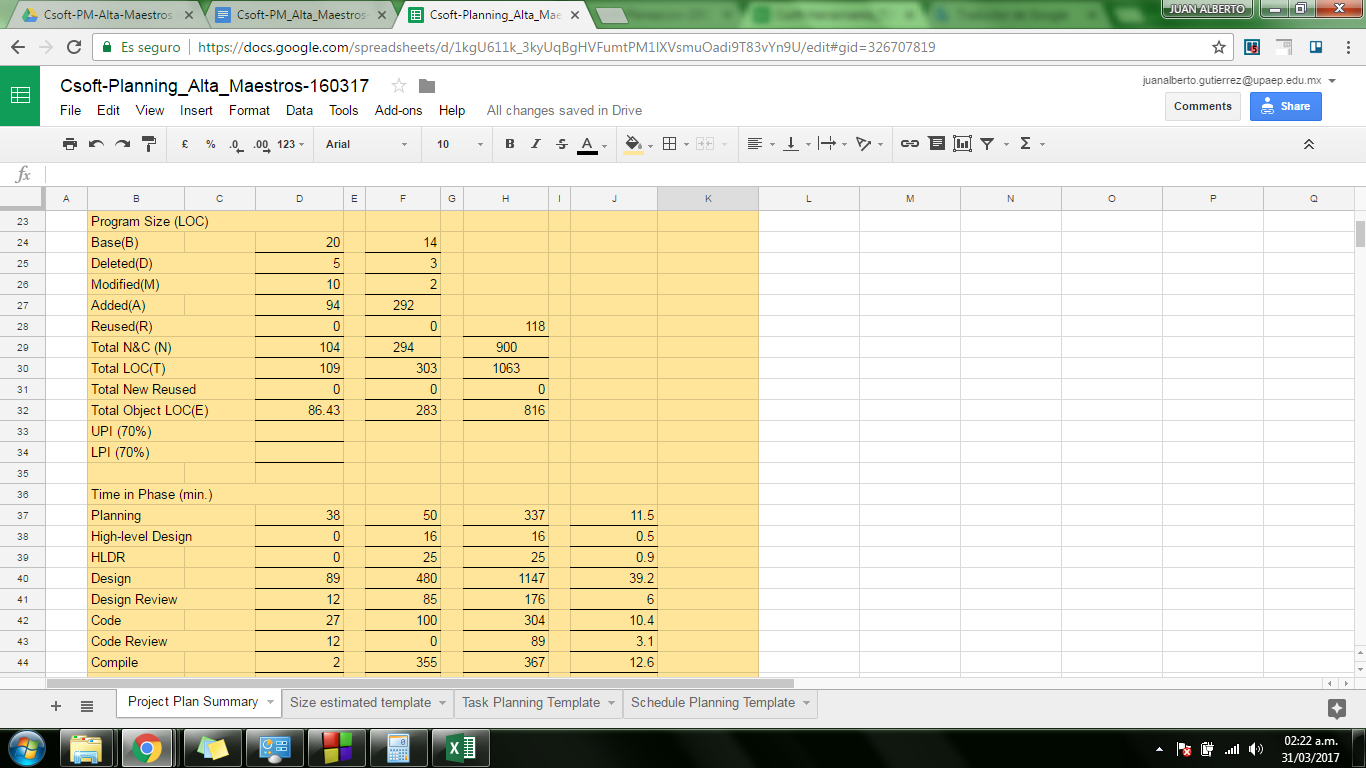
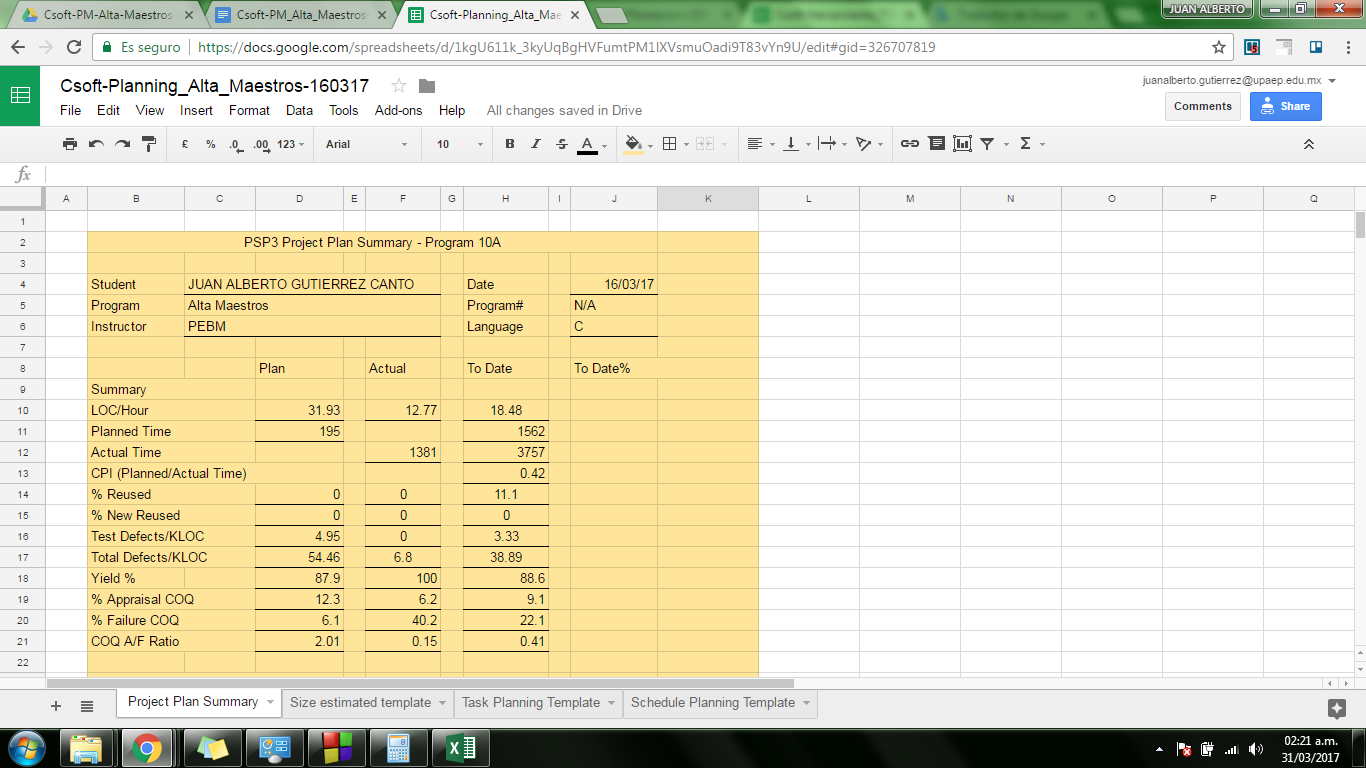
[**XI.- Task and Schedule**](#_195jonwtvlxs) **5**

# 

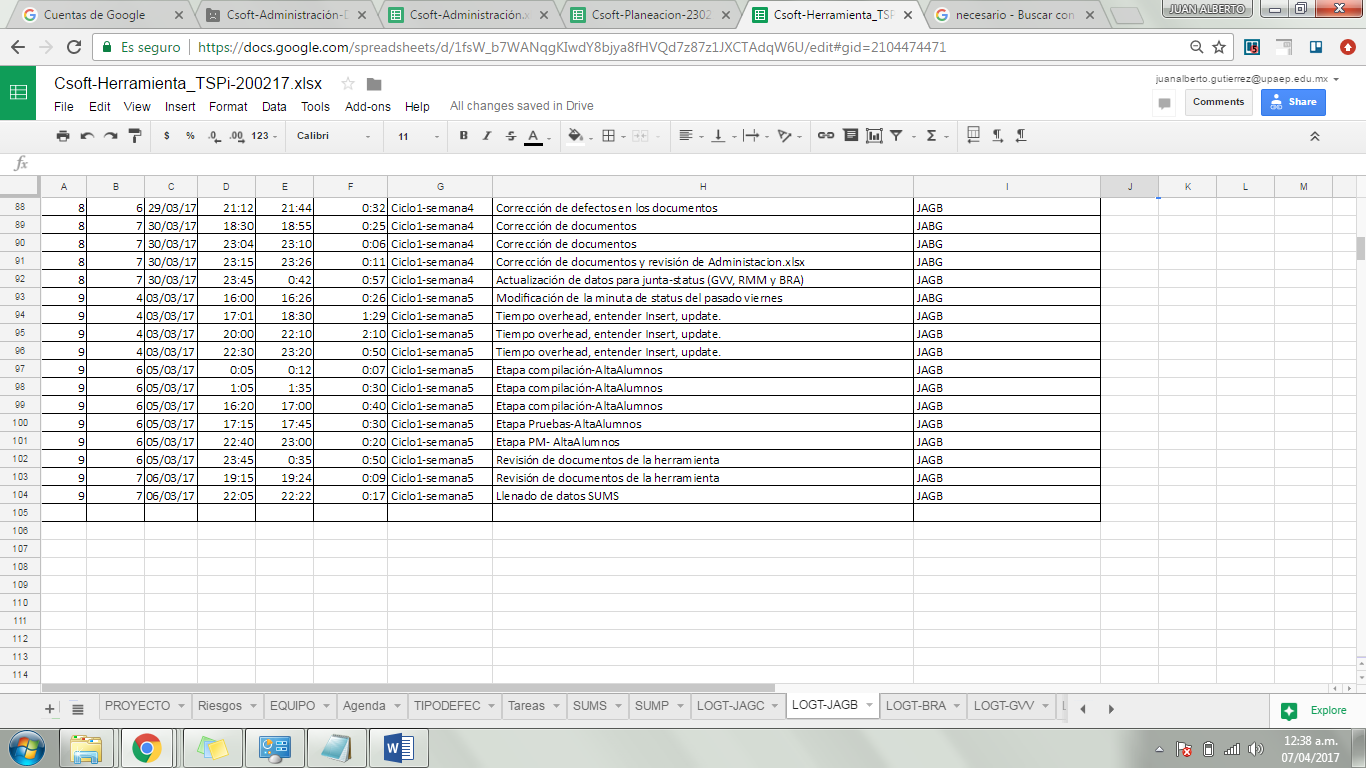
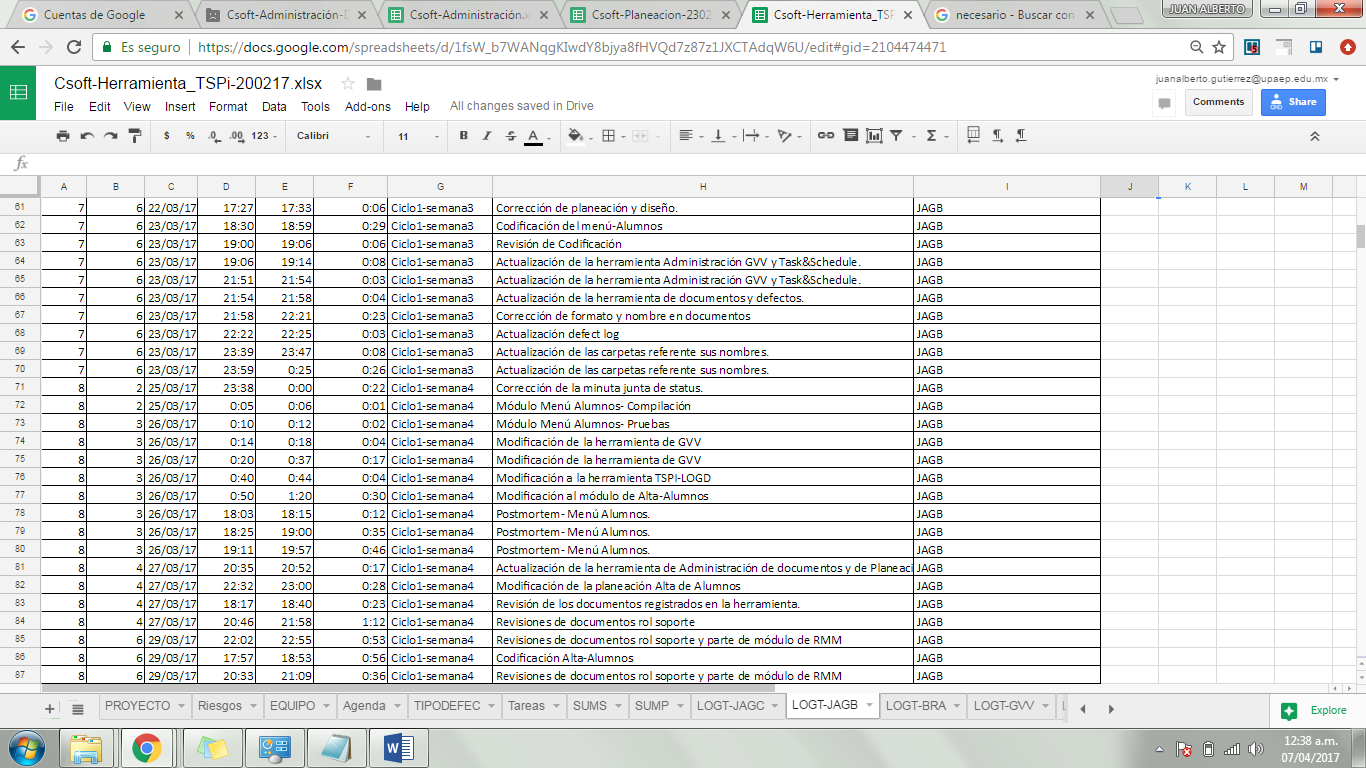
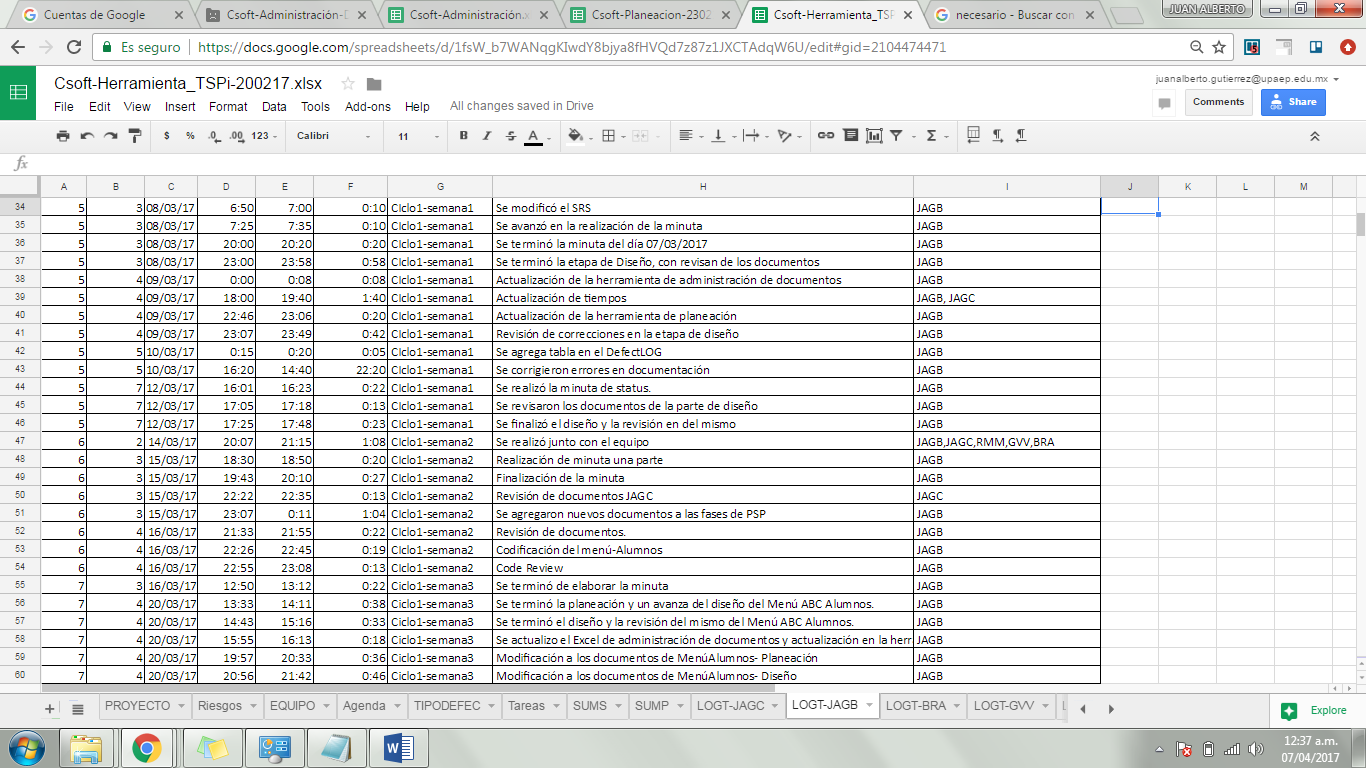
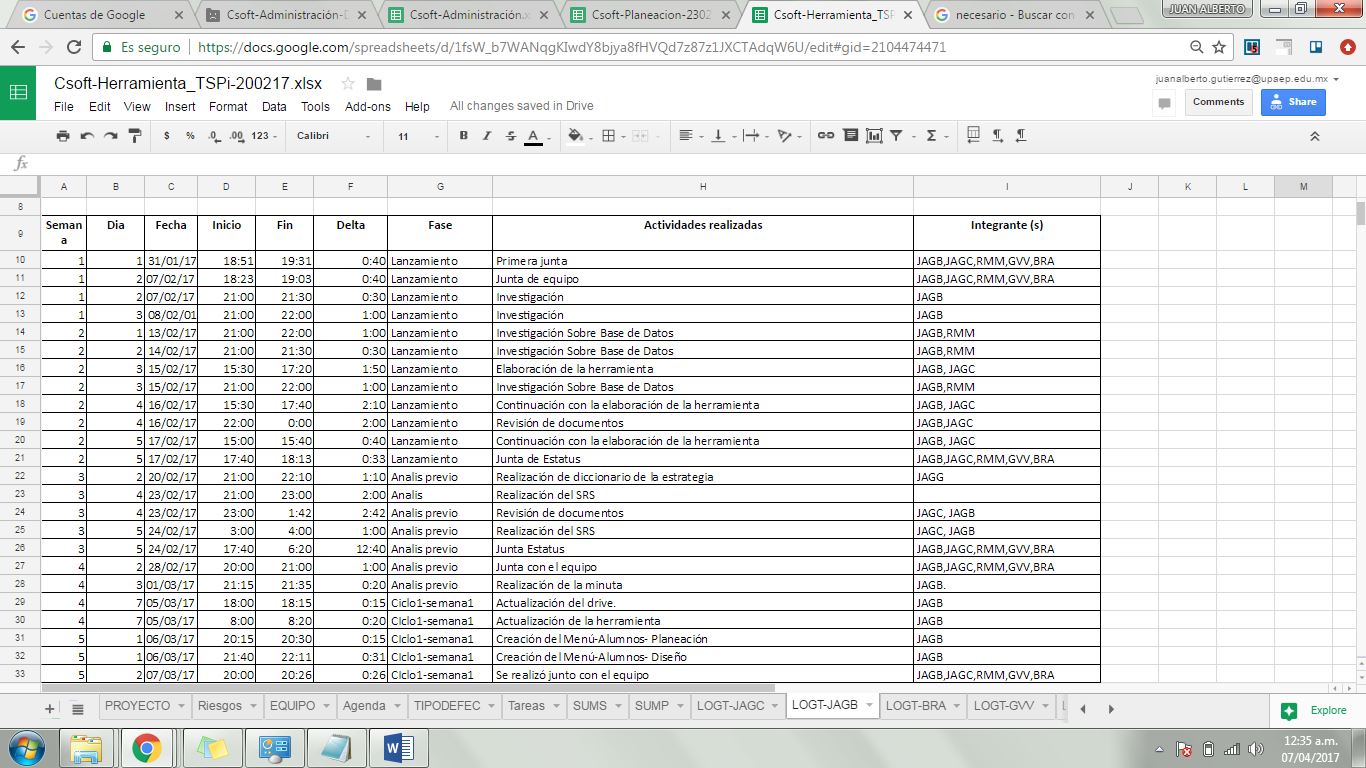
# 

# I.- Forma PSP Project Plan Summary.

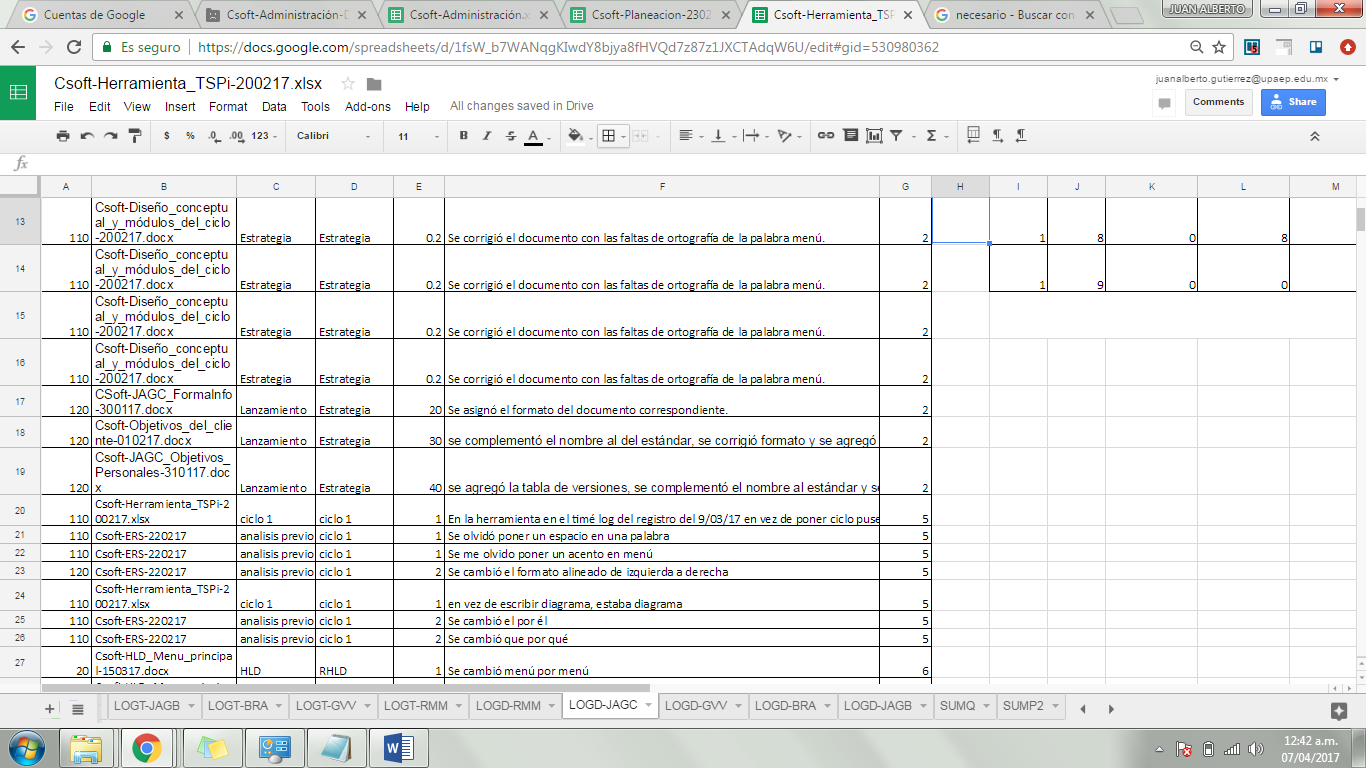
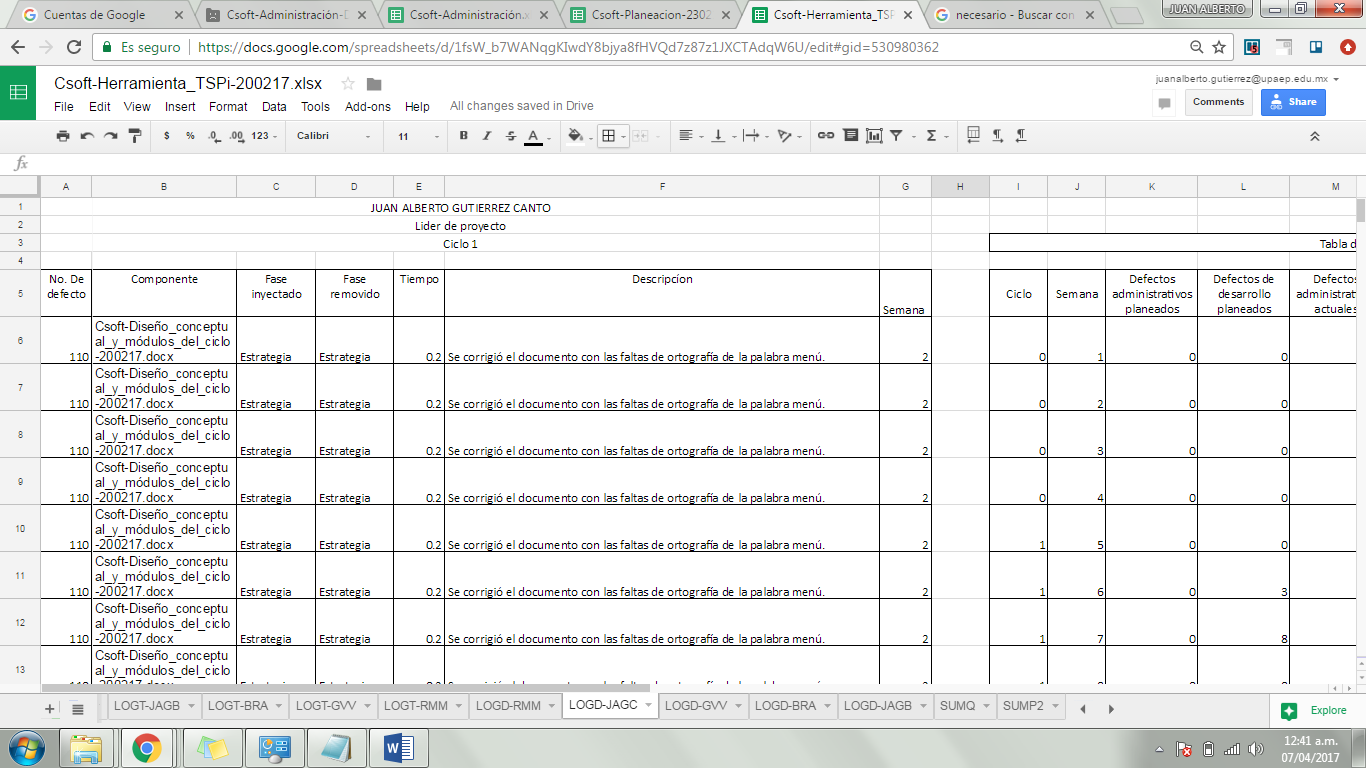
Se inició el proceso el 16/03/2017

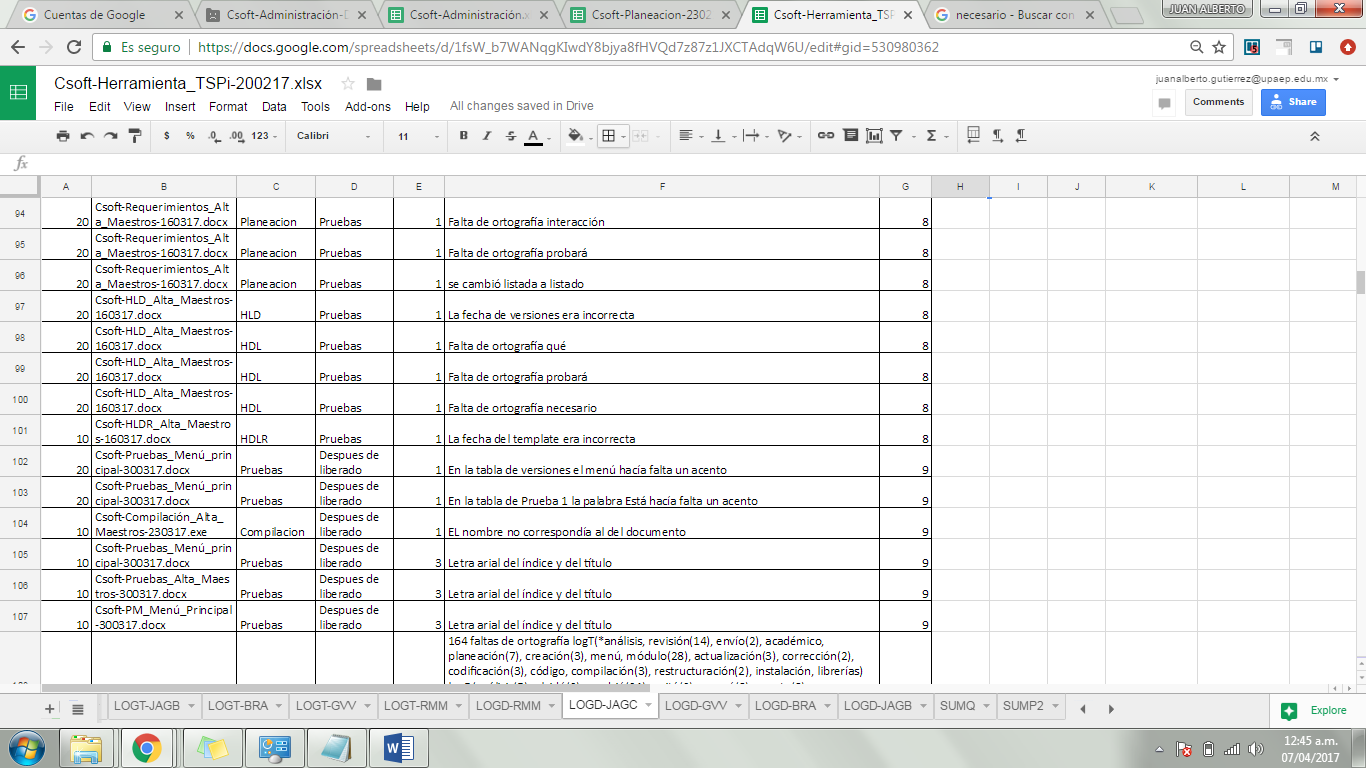
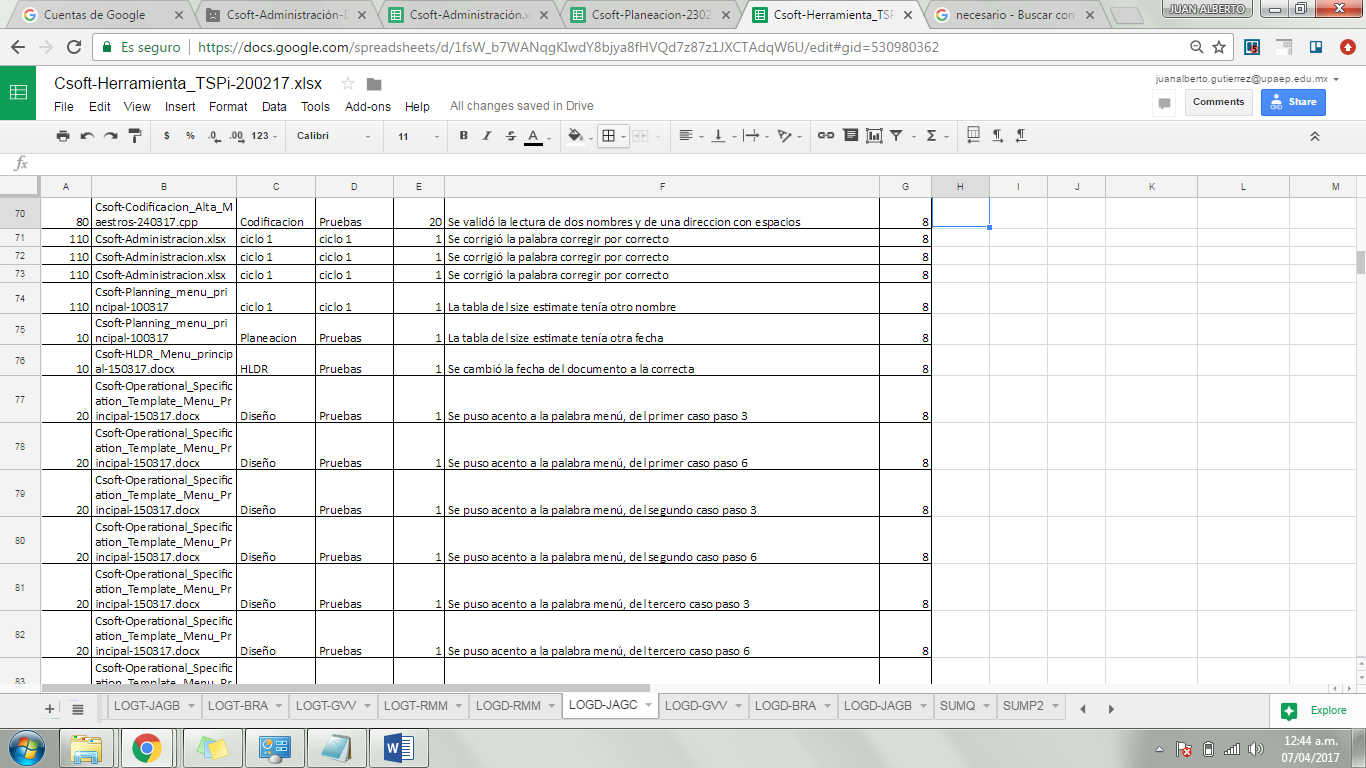
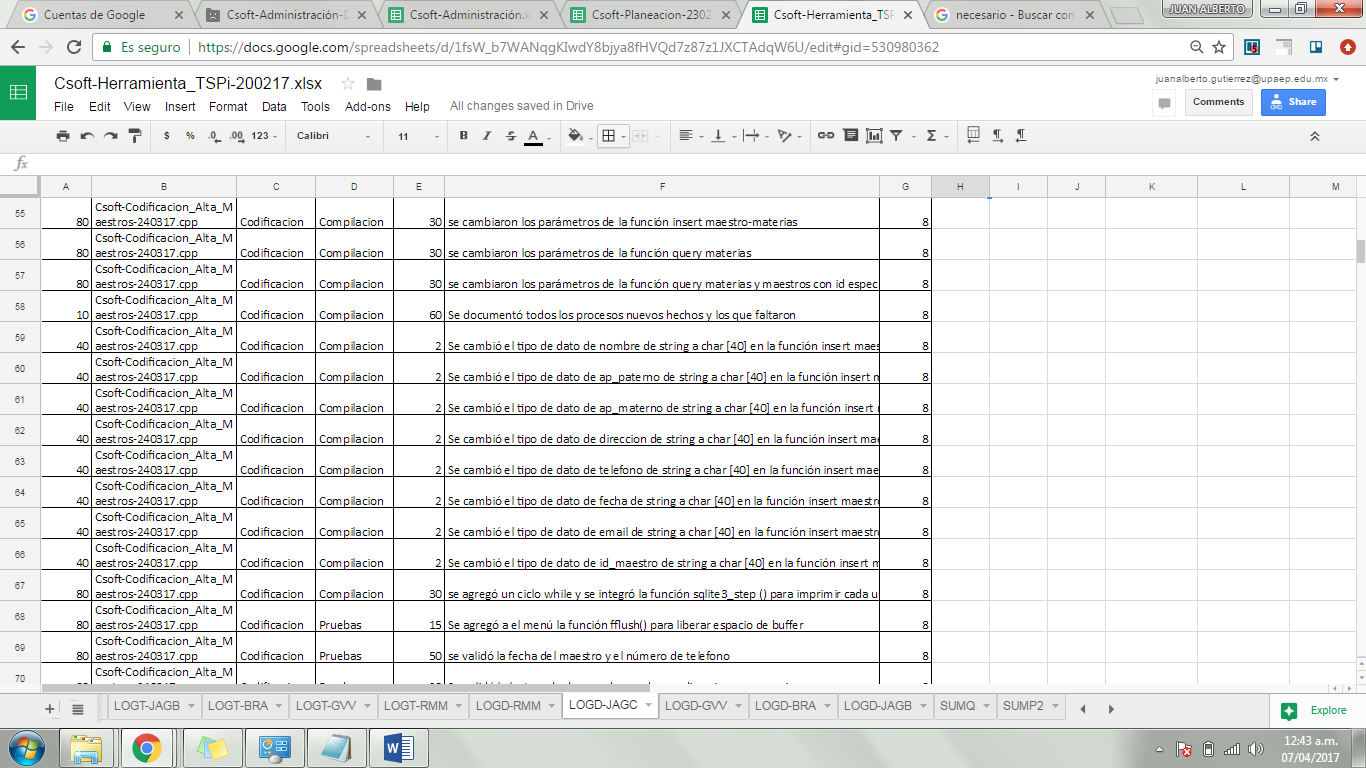
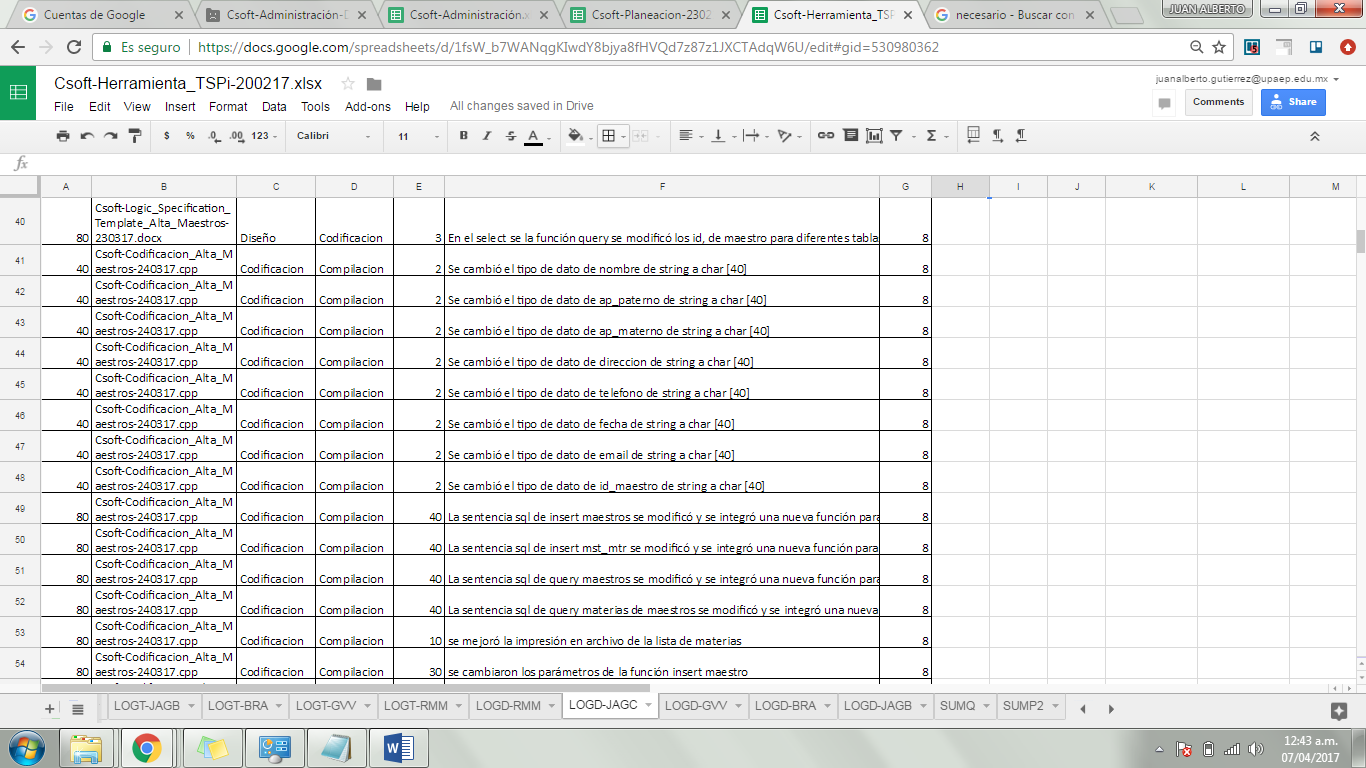
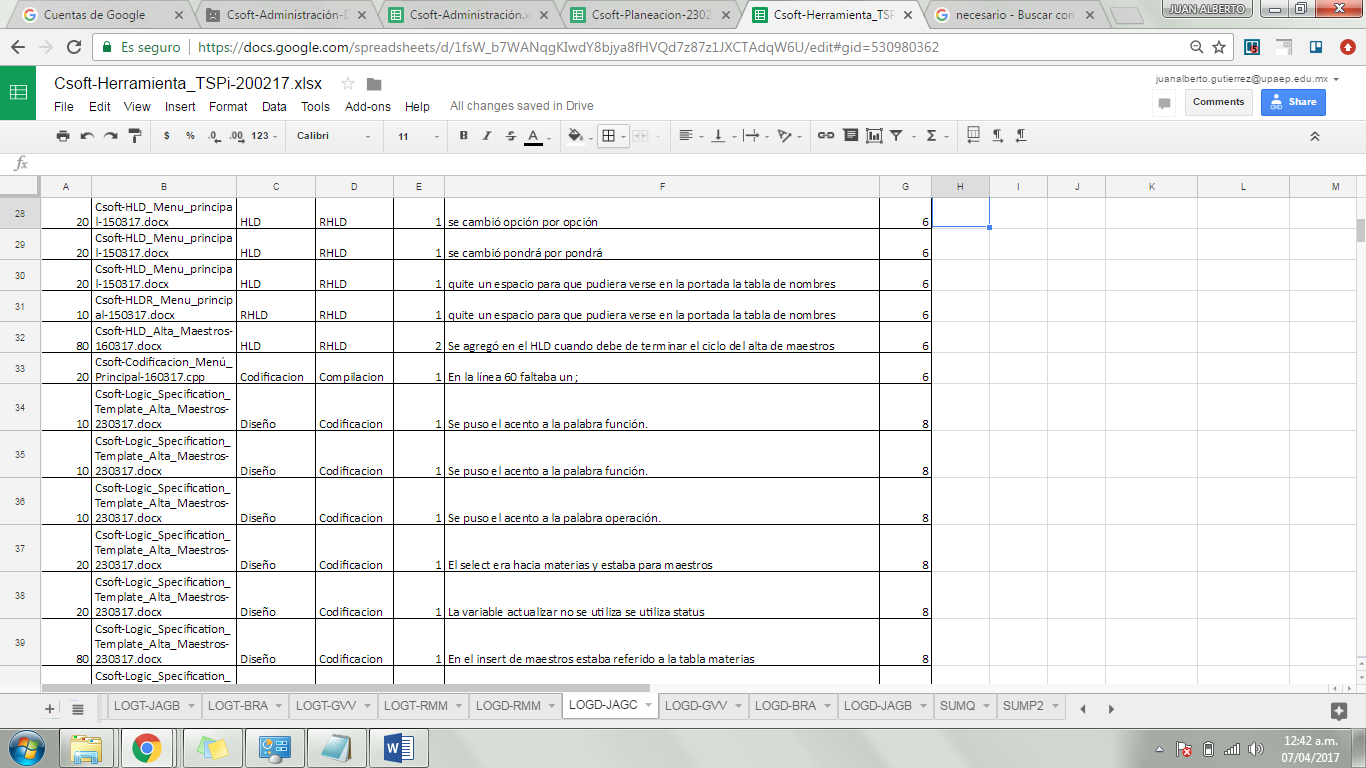


# II.- Time Recording Log.



# III.- Defect Recording Log.





# IV.- Diseño.

# State Specification Template

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Student | | Juan Alberto Gutierrez Canto | | | Date | | 21/03/2017 |
| Program | | Alta Maestros | | | Program # | | N/A |
| Instructor | | PEBM | | | Language | | C |
|  | | |  | | | | |
| Menú principal | | | | | | | |
| **State Name** | | | | **Description** | | | |
| Inicio | | | | Inicia el programa. | | | |
| Alta Maestros | | | | Función para empezar el alta de maestros. | | | |
| Muestras Materias | | | | Muestra las materias que están en la base de datos. | | | |
| Insert Maestros | | | | Guarda en la base de datos el alta. | | | |
| Insert Materias-Maestros | | | | Guarda en la base de datos la vinculación de maestros y materias | | | |
| Query Maestros | | | | 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 | | | |
| Alta\_Maestros() | | | | Enlace para el alta de maestros | | | |
| Actualizar\_Materias() | | | | Muestra materias en base de datos | | | |
| Insert\_Maestro() | | | | Guarda maestro en base de datos | | | |
| Insert\_Mtr\_Mst() | | | | Guarda vinculación maestros- materias | | | |
| Retrurn() | | | | Regresa un valor. | | | |
| Query\_Maestros | | | | Imprime registro desde base de datos | | | |
| actilizar | | | | variable para ver si quiere actualizar lista de materias disponibles. | | | |
| cantidad\_materias | | | | Para determinar cuántas materias se elegirán. | | | |
| **States/Next States** | | | | **Transition Condition** | | **Action** | |
| Inicio | | | |  | |  | |
|  | Alta Maestros | | | inicia el programa | | Alta\_Maestros() | |
| Alta Maestros | | | |  | |  | |
|  | Muestra Materias | | | actualizar==true | | Actualizar\_Materias() | |
|  | Insert Maestros | | | cantidad\_materias == i | | Insert\_Maestros() | |
|  | Insert Materias-Maestros | | | mientras cantidad\_materias ==i con i=0 para i++ | | Insert\_Mtr\_Mst() | |
|  | Query Maestros | | | Ya se insertó en la base de datos | | Query\_Maestros() | |
|  | Salir | | | Se imprimió el Alta maestros | | Return() | |

# Logic Specification Template

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Student | Juan Alberto Gutierrez Canto | | Date | 23/03/17 |
| Program | Alta Maestros | | Program # | N/A |
| Instructor | PEBM | | Language | C |
| Object | N/A | | Function | alta\_maestros() |
| **INCLUDES:**  **TYPE DEFINITIONS:**  **Declaration:** | | #include <cstdio>  #include <cstdlib>  #include <iostream>  #include <string>  #include “sqlite3.h”  Using namespace std; | | |
|  | | sqlite3 \*db | | |

|  |  |
| --- | --- |
| **Reference:** |  |

|  |  |
| --- | --- |
| Logic reference numbers | Program logic, in pseudocode |
| 1 | string ap\_materno |
|  | string ap\_paterno |
|  | string nombre |
|  | string fecha\_nac |
|  | string direccion |
|  | string telefono |
|  | string email |
|  | int cant\_materias |
|  | int cant\_mat\_dar |
|  | string materia |
|  | string id\_materia |
|  | int status=1 |
|  | mientras status != 0 |
|  | imprime menu para pedir datos al usuario |
|  | lee nombre |
|  | lee apellido paterno |
|  | lee apellido materno |
|  | lee fecha nacimiento |
|  | lee direccion |
|  | lee telefono |
|  | lee email |
|  | se imprimen los datos leídos y se preguntan si son   correctos 0)si 1)no |
|  | leer status |
|  | se llama a la función insert\_maestros(nombre,ap\_paterno,ap\_materno,fecha\_nac,direccion, telefono, email) |
|  | escribe que la lista de materias con sus id se encuentra en la carpeta, y se pregunta si quiere actualizarla |
|  | leer status |
|  | si desea actualizar se va a la función actualizar\_materias() |
|  | se pide la cantidad de materias que va a tener el maestro |
|  | leer cant\_materias |
|  | mientras cant\_materias != 0 |
|  | pide el id de la materia |
|  | leer id\_materia |
|  | pide si la impartira o no |
|  | leer status |
|  | se llama a la función insert\_mtr\_mst(id\_maestro,   id\_materia, status) |
|  | se llama a la función query\_maestros() y se muestra los datos introducidos. |

# 

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Object | N/A | | Function | actualizar\_materias() |
| **INCLUDES:**  **TYPE DEFINITIONS:**  **Declaration:** | | #include <cstdio>  #include “sqlite3.h”  Using namespace std; | | |
|  | | sqlite3 \*dp | | |

|  |  |
| --- | --- |
| **Reference:** |  |

|  |  |
| --- | --- |
| Logic reference numbers | Program logic, in pseudocode |
| 1 | char \*sql |
|  | char \*err=0 |
|  | int rc |
|  | const char\* data = "Materias actualizadas\n"; |
|  | se asigna a pfile=fopen(“materias\_disponibles.txt”,”w”) para abrir el archivo |
|  | se asigna la operación a sql=”SELECT \* from MATERIAS”); |
|  | se asigna a rc=sqlite3\_exec(db,sql,print\_archivo,(void\*)data, &err); |
|  | si rc!= SQLITE\_OK entonces |
|  | se imprime “SQL error %s”,err |
|  | sqlite3\_free(err) |
|  | si no |
|  | se imprime “Nueva lista disponible” |
|  | se cierra el documento con fclose(pfile); |

# 

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Object | N/A | | Function | static int print\_archivo() |
| **INCLUDES:**  **TYPE DEFINITIONS:**  **Declaration:** | | #include <cstdio>  #include “sqlite3.h”  Using namespace std; | | |
|  | | void \*data | | |
|  | | int argc | | |
|  | | char \*\*argv | | |
|  | | char \*\*az\_col\_name | | |

|  |  |
| --- | --- |
| **Reference:** |  |

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

# 

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Object | N/A | | Function | void insert\_maestros() |
| **INCLUDES:**  **TYPE DEFINITIONS:**  **Declaration:** | | #include <cstdio>  #include “sqlite3.h”  Using namespace std; | | |
|  | | sqlite3 \*db | | |
|  | | string id\_maestro | | |
|  | | string nombre | | |
|  | | string ap\_paterno | | |
|  | | string ap\_materno | | |
|  | | string fecha\_nac | | |
|  | | string direccion | | |
|  | | string telefono | | |
|  | | string email | | |

|  |  |
| --- | --- |
| **Reference:** |  |

|  |  |
| --- | --- |
| 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”); |

# 

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Object | N/A | | Function | void insert\_mtr\_mst() |
| **INCLUDES:**  **TYPE DEFINITIONS:**  **Declaration:** | | #include <cstdio>  #include “sqlite3.h”  Using namespace std; | | |
|  | | sqlite3 \*db | | |
|  | | string id\_materia | | |
|  | | string id\_maestro | | |
|  | | int status | | |

|  |  |
| --- | --- |
| **Reference:** |  |

|  |  |
| --- | --- |
| 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”); |

# 

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Object | N/A | | Function | void query\_maestros() |
| **INCLUDES:**  **TYPE DEFINITIONS:**  **Declaration:** | | #include <cstdio>  #include “sqlite3.h”  Using namespace std; | | |
|  | | sqlite3 \*db | | |
|  | | string id\_maestro | | |

|  |  |
| --- | --- |
| **Reference:** |  |

|  |  |
| --- | --- |
| 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”); |

# 

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Object | N/A | | Function | static int callback() |
| **INCLUDES:**  **TYPE DEFINITIONS:**  **Declaration:** | | #include <cstdio>  #include “sqlite3.h”  Using namespace std; | | |
|  | | void \*data | | |
|  | | int argc | | |
|  | | char \*\*argv | | |
|  | | char \*\*az\_col\_name | | |

|  |  |
| --- | --- |
| **Reference:** |  |

|  |  |
| --- | --- |
| 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; |

# Functional Specification Template

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Student** | | | Juan Alberto Gutierrez Canto | | **Date** | 21/03/17 |
| **Program** | | | Alta Maestros | | **Program #** | N/A |
| **Instructor** | | | PEBM | | **Language** | C |
|  | | | | | | |
| **Class Name** | | void alta\_maestros() | | | | |
| **Parent Class** | | N/A | | | | |
|  | | | | | | |
| **Attributes** | | | | | | |
|  | **Declaration** | | | **Description** | | |
|  | int actualizar | | | Corresponde a la opción que el usuario eligira para actualizar la lista de materias o no. | | |
|  | 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 | | | direccion del maestro | | |
|  | string telefono | | | teléfono del maestro | | |
|  | string email | | | email del maestro | | |
|  | int cant\_materias | | | cantidad de materias que puede dar | | |
|  | int cant\_mat\_dar | | | cantidad de materias que va a dar | | |
|  | string materia | | | Nombre de la materia | | |
|  | string id\_materia | | | Identificador de la materia | | |
|  | int status | | | para definir si imparte al momento la materia o no | | |
|  | | | | | | |
| **Items** | | | | | | |
|  | **Declaration** | | | **Description** | | |
|  | Actualizar\_Materias() | | | Actualiza la lista de materias disponibles para seleccionar. | | |
|  | Insert\_Maestros() | | | Guarda los datos del maestro en la base de datos. | | |
|  | Insert\_Mtr\_Mst() | | | Guarda los datos de la materia con maestros en datos. | | |
|  | Query\_Maestros() | | | Imprime los datos de un maestros | | |
|  | printf() | | | Para imprimir directamente en la interfaz de usuario. | | |
|  | scanf() | | | Para leer datos de la interfaz de usuario. | | |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | | | | | | |
| **Class Name** | | void Actualizar\_Materias() | | | | |
| **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. | | |
|  | const char\* data = "Consulta"; | | | descripción del proceso. | | |
|  | | | | | | |
| **Items** | | | | | | |
|  | **Declaration** | | | **Description** | | |
|  | fopen() | | | Sirve para abrir el archivo de materias. | | |
|  | 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. | | |
|  | print\_archivo() | | | Para imprimir directamente a el archivo txt. | | |
|  | fclose() | | | Para cerrar un archivo. | | |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Class Name** | | static int print\_archivo() | | | | |
| **Parent Class** | | N/A | | | | |
|  | | | | | | |
| **Attributes** | | | | | | |
|  | **Declaration** | | | **Description** | | |
|  | void \*data | | | Función hacia los datos. | | |
|  | int argc | | | Numero 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** | | |
|  | fprintf() | | | Imprime en un determinado documento lo que le mandes en cadena y variables determinadas. | | |
|  | return() | | | Regresa un valor determinado. | | |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Class Name** | | void Insert\_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 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 | | | direccion del maestro | | |
|  | string telefono | | | teléfono del maestro | | |
|  | string email | | | email del maestro | | |
|  | | | | | | |
| **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** | | int Insert\_Mtr\_Mst() | | | | |
| **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 | | | apellido materno del maestro | | |
|  | const char\* data = "Consulta" | | | Para indicar el tipo de ejecucion. | | |
|  | | | | | | |
| **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 | | | Numero 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. | | |

# Operational Specification Template

|  |  |  |  |
| --- | --- | --- | --- |
| **Student** | Juan Alberto Gutierrez Canto | **Date** | 12/03/17 |
| **Program** | Alta Maestros | **Program #** | N/A |
| **Instructor** | PEBM | **Language** | C |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Scenario Number** | **1** | **User Objective** | Alta Maestros actualizando lista | |
| **Scenario Objective** | | Pasos para hacer el alta de maestros, actualizando archivos de materias. | | |
| **Source** | **Step** | **Action** | | **Comments** |
| Usuario | 1 | Abril el programa. | |  |
| Programa | 2 | Pedir datos Personales | | Datos personales son:  Nombres  Apellido paterno  Apellido materno  Fecha de nacimiento  Dirección  Correo electrónico  Teléfono |
| Usuario | 3 | Ingresara los datos pedidos | | Cada dato con un enter |
| Programa | 4 | Pregunta si muestra las materias que existen en la base de datos o ya las tiene actualizadas | | Si quiere ver la lista actualizada o ya esta actualizada |
| Usuario | 5 | ingresa que si quiere actualizar lista de materias disponibles | | Para imprimir la lista de materias. |
| Programa | 6 | Imprime las materias actualizada | | Imprime en un archivo |
| Programa | 7 | Pide ingresar las materias que puede dar el maestro | | En pantalla |
| Usuario | 8 | Ingresa los id de las materias que puede dar el maestro | | Cada una con un enter |
| Programa | 9 | Pregunta si esa materia la va a impartir en ese momento | | Pregunta al usuario |
| Usuario | 10 | Responde si se está dando el maestro la materia | | enter después de cada una |
| Programa | 11 | Muestra los datos del alta | | En pantalla los imprime todos |
| Programa | 12 | Guarda los datos en la base de datos | | Insertar datos |
| Programa | 13 | Imprime datos desde base de datos | | Hace un query para ver el registro completo. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Scenario Number** | **1** | **User Objective** | Alta Maestros sin actualizar lista | |
| **Scenario Objective** | | Pasos para hacer el alta de maestros, actualizando archivos de materias. | | |
| **Source** | **Step** | **Action** | | **Comments** |
| Usuario | 1 | Abril el programa. | |  |
| Programa | 2 | Pedir datos Personales | | Datos personales son:  Nombres  Apellido paterno  Apellido materno  Fecha de nacimiento  Dirección  Correo electrónico  Teléfono |
| Usuario | 3 | Ingresara los datos pedidos | | Cada dato con un enter |
| Programa | 4 | Pregunta si muestra las materias que existen en la base de datos o ya las tiene actualizadas | | Si quiere ver la lista actualizada o ya esta actualizada |
| Usuario | 5 | ingresa que no quiere actualizar lista de materias disponibles | | Para imprimir la lista de materias. |
| Programa | 6 | Pide ingresar las materias que puede dar el maestro | | En pantalla |
| Usuario | 7 | Ingresa los id de las materias que puede dar el maestro | | Cada una con un enter |
| Programa | 8 | Pregunta si esa materia la va a impartir en ese momento | | Pregunta al usuario |
| Usuario | 9 | Responde si se está dando el maestro la materia | | enter después de cada una |
| Programa | 10 | Muestra los datos del alta | | En pantalla los imprime todos |
| Programa | 11 | Guarda los datos en la base de datos | | Insertar datos |
| Programa | 12 | Imprime datos desde base de datos | | Hace un query para ver el registro completo. |

# V.- Design CheckList.

# Revisión diseño

**PSP2 Design Review Checklist**

|  |  |  |  |
| --- | --- | --- | --- |
| Student | Juan Alberto Gutierrez Canto | Date | 23/07/2016 |
| Program | Alta Maestros | Program # | N/A |
| Instructor | PEBM | Language | C |

|  |  |
| --- | --- |
| Purpose | To guide you in conducting an effective design 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 design covers all of the applicable requirements.  - All specified outputs are produced.  - All needed inputs are furnished.  - All required includes are stated. | MB  X  X | B      X | R | M |
| External Limits | Where the design assumes or relies upon external limits, determine if behavior is correct at nominal values, at limits, and beyond limits. |  |  |  |  |
| 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.  - Examine each conditional statement and verify all cases. | X        X | X |  |  |
| Internal Limits | Where the design assumes or relies upon internal limits, determine if behavior is correct at nominal values, at limits, and beyond limits. |  |  |  |  |
| Special Cases | - Check all special cases.  - Ensure proper operation with empty, full, minimum, maximum, negative, and ero values for all variables.  - Protect against out-of-limits, overflow, and underflow conditions.  - Ensure “impossible” conditions are absolutely impossible.  - Handle all possible incorrect or error conditions. | X  X    X  X  X |  |  |  |
| Functional Use | - Verify that all functions, procedures, or methods are fully understood and properly used.  - Verify that all externally referenced abstractions are precisely defined. | X    X |  |  |  |
| System Considerations | - Verify that the program does not cause system limits to be exceeded.  - Verify that all security-sensitive data are from trusted sources.  - Verify that all safety conditions conform to the safety specifications. | X    X    X |  |  |  |
| Names | Verify that  - all special names are clear, defined, and authenticated  - the scopes of all variables and parameters are self-evident or defined  - all named items are used within their declared scopes |  | X  X  X |  |  |
| Standards | Ensure that the design conforms to all applicable design standards. | X |  |  |  |

# VI.- Programa Fuente.

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

/\*Autor: Juan Alberto Gutierrez Canto \*/

/\*Componente: Alta Maestros \*/

/\*Fecha: 23/03/2017 \*/

/\*Descripción: Codificación que muestra el alta para maestros\*/

/\*Versión: 1.0 \*/

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

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

/\* 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 insert\_mtr\_mst(sqlite3 \*db,int id\_materia, char\* id\_maestro, int status) \*/

/\* void insert\_maestros(sqlite3 \*db,char\* id\_maestro, char\* nombre, char\* ap\_paterno, char\* ap\_materno, char\* fecha\_nac, char\* direccion, char\* telefono, char\* email) \*/

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

/\* void actualizar\_materias(sqlite3 \*db) \*/

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

/\* int main() \*/

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

#include <iostream> /\*Entrada y salida de datos\*/

#include <cstdio> /\*Entrada y salida de datos, file, fflush\*/

#include <cstdlib>

#include "sqlite3.h" /\*Interaccion base de datos\*/

using namespace std;

FILE \*pfile; /\*Archivo para imprimir las materias en la base de datos\*/

/\*\*\*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); /\*Imprime el tipo de consulta que está en data\*/

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

printf("%s = %s\n", az\_col\_name[i], argv[i] ? argv[i] : "NULL"); /\*Imprime cada valor de la consulta de acuerdo a su columna y valor\*/

}

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\_M=@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(" Nombre:\t%s %s %s\n", sqlite3\_column\_text(res, 1),sqlite3\_column\_text(res, 2),sqlite3\_column\_text(res, 3));

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

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

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

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

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

}

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

sql="SELECT ID\_Materia,Status FROM IMPARTE where ID\_Maestro=@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%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 insertar los valores del maestro y materias en la base de datos\*\*\*/

void insert\_mtr\_mst(sqlite3 \*db,int id\_materia, char\* id\_maestro, int status){

char \*sql;

char \*err=0;

int rc;

sqlite3\_stmt \*res;

sql="INSERT INTO IMPARTE (ID\_Maestro, ID\_Materia, Status) VALUES (@Id\_M,@Id\_T,@status);";/\* Inserta los 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(" IMPARTE-SQL error %s\n",err); /\* Imprime el error de la sentencia\*/

}

else{

printf(" Valores ingresados correctamente.\n"); /\* Cambia los identificadas 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\_T");

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

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

sqlite3\_bind\_int(res, idt, id\_materia); /\* Cambia el identificador por un valor int\*/

sqlite3\_bind\_text(res, idstatus,(status==1?"IMPARTE":"NO IMPARTE"),40,SQLITE\_STATIC);/\* cambia el identificador a imparte si status es igual a 1 si no a no imparte\*/

}

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 del maestro en la base de datos\*\*\*/

void insert\_maestros(sqlite3 \*db,char\* id\_maestro, char\* nombre, char\* ap\_paterno, char\* ap\_materno, char\* fecha\_nac, char\* direccion, char\* telefono, char\* email){

char \*sql;

char \*err=0;

int rc;

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

sql="INSERT INTO MAESTROS (Id\_M, Nombre, ApPaterno, ApMaterno, FNacimiento, Direccion, Telefono, Email) VALUES (@Id\_M,@nom,@ap\_p,@ap\_m,@fec,@dir,@tel,@ema);";

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 maestro guardados correctamente.\n");/\* Cambia los identificadores 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");

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

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

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

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

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

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

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

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

sqlite3\_bind\_text(res, ap\_p, ap\_paterno,40,SQLITE\_STATIC);

sqlite3\_bind\_text(res, ap\_m, ap\_materno,40,SQLITE\_STATIC);

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

sqlite3\_bind\_text(res, dir, direccion,40,SQLITE\_STATIC);

sqlite3\_bind\_text(res, tel, telefono,40,SQLITE\_STATIC);

sqlite3\_bind\_text(res, ema, email,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 actualizar el documento Materias\_Disponibles.txt que se encuentra en la carpeta del proyecto\*\*\*/

void actualizar\_materias(sqlite3 \*db){

char \*sql;

char \*err=0;

int rc;

const char\*data;

pfile=fopen("Materias\_Disponibles.txt","w"); /\* abrimos el archivo en modo edición\*/

sql="SELECT \* from materias;"; /\* Instrucción sql para seleccionar todas las materias\*/

fprintf(pfile,"----- MATERIAS DISPONIBLES PARA MAESTROS -----\nID\tNOMBRE\n");/\* Título en el documento txt\*/

rc=sqlite3\_exec(db,sql,print\_archivo,(void\*)data,&err); /\* realiza la consulta sql\*/

if(rc!=SQLITE\_OK){

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

}

else{

printf(" Nueva lista disponible\n");

}

fclose(pfile);

}

/\*\*\*Esta función se encarga de presentar alta de maestros,

contiene todo el proceso y re direcciona hacia las funciones de inserción y select con la base de datos\*\*\*/

void alta\_maestros(sqlite3 \*db){

char nombre[40];

char ap\_paterno[40];

char ap\_materno[40];

char fecha\_nac[40];

char direccion[40];

char telefono[40];

char email[40];

char id\_maestro[40];

int cant\_materias;

int cant\_mat\_dar;

int id\_materia;

int status=2;

bool correcto=false;

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

system("cls");

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

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

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

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

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

printf(" ! 1) Nombre 2) Apellido Paterno !\n\n");

printf(" ! 3) Apellido Materno 4) Fecha Nacimiento DD/MM/YY !\n\n");

printf(" ! 5) Direccion 6) Telefono !\n\n");

printf(" ! 7) Email 8)RFC !\n\n");

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

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

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

scanf("%[^\n]", nombre);

printf(" 2)"); /\* apellido paterno\*/

fflush( stdin );

scanf("%s",ap\_paterno);

printf(" 3)"); /\* apellido materno\*/

fflush( stdin );

scanf("%s",ap\_materno);

correcto=true;

do{ /\* Validación fecha\*/

printf(" 4)");

fflush( stdin );

scanf("%s",fecha\_nac);

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

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

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

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

printf(" mes incorrecto\n");

}

}

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

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

if(!(fecha\_nac[1]>='0' && fecha\_nac[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 número\*/

}

}while(correcto);

printf(" 5)"); /\* direccion\*/

fflush( stdin );

scanf("%[^\n]",direccion);

printf(" 6)"); /\* telefono\*/

fflush( stdin );

scanf("%[0-9]",telefono); /\* solamente acepta los primeros números\*/

correcto=true;

do{ /\* validación del correo\*/

printf(" 7)");

fflush( stdin );

scanf("%s",email);

for(int i=0;i<40;i++){

if(email[i]== '@'){ /\* si tiene @ es correcto\*/

correcto=false;

}

}

if(correcto){

printf(" email incorrecto\n"); /\* no tiene @ es incorrecto\*/

}

}while(correcto);

printf(" 8)"); /\* RFC\*/

fflush( stdin );

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

system("cls");

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

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

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

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

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

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

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

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

printf("\n Direccion:\t%s\n",&direccion);

printf("\n Telefono:\t%s\n",&telefono);

printf("\n Email:\t\t%s\n",&email);

printf("\n RFC:\t\t%s\n",&id\_maestro);

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

cin>>status;

}

system("cls");

insert\_maestros(db,id\_maestro,nombre,ap\_paterno,ap\_materno,fecha\_nac,direccion,telefono,email);/\* MAndamos a guardar datos a la bd\*/

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

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

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

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

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

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

printf(" ! La lista est%c en la carpeta del programa. !\n\n",160);

printf(" ! ¿Desea actualizar la lista? 1)si 2)no: !\n\n");

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

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

cin>>status;

if(status == 1){

actualizar\_materias(db); /\* Se actualiza el txt con las materias de la db\*/

}

printf(" Ingrese el número de materias del maestro.-\n "); /\* Numero de materias del maestro\*/

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

while(cant\_materias--){ /\* para cada materia\*/

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

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

printf(" ¿Está impartiendo la materia? 1)si 2)no\n "); /\* definición si imparte la materia o no\*/

cin>>status;

insert\_mtr\_mst(db,id\_materia,id\_maestro,status); /\* guarda las materias del maestro en la db\*/

}

system("cls");

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

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

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

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

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

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

query\_maestros(db,id\_maestro); /\* selecciona los datos del maestro 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("test.db", &db); /\* Se abre la db\*/

alta\_maestros(db); /\*A alta de maestros\*/

return 0;

}

# VII.- Code CheckList.

# Revisión codificación

**Code Review Checklist**

|  |  |  |  |
| --- | --- | --- | --- |
| Student | Juan Alberto Gutierrez Canto | Date | 24/03/17 |
| Program | Alta 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. |  |  |  |  |
| File Open and Close | Verify that all files are  - properly declared  - opened  - closed | X  X  X |  |  |  |

# VIII.- Reporte de resultados.

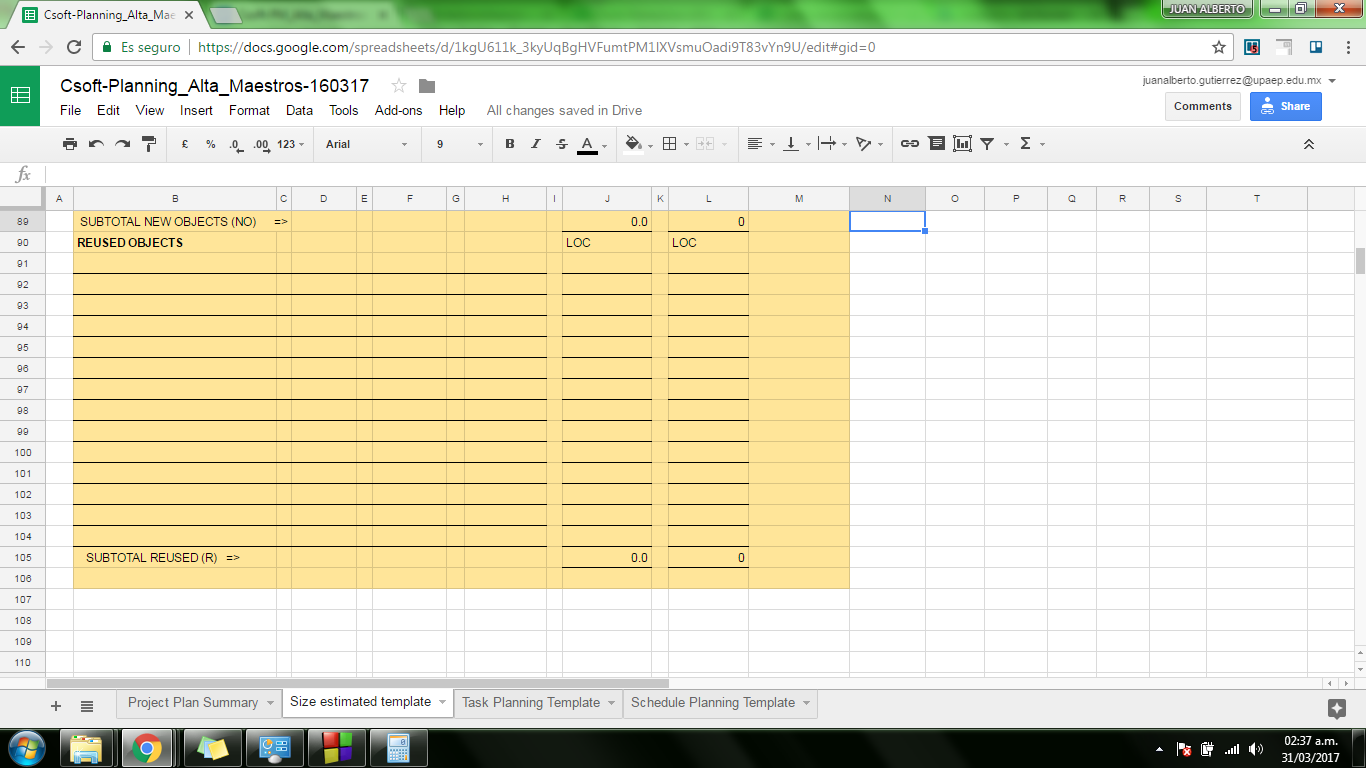
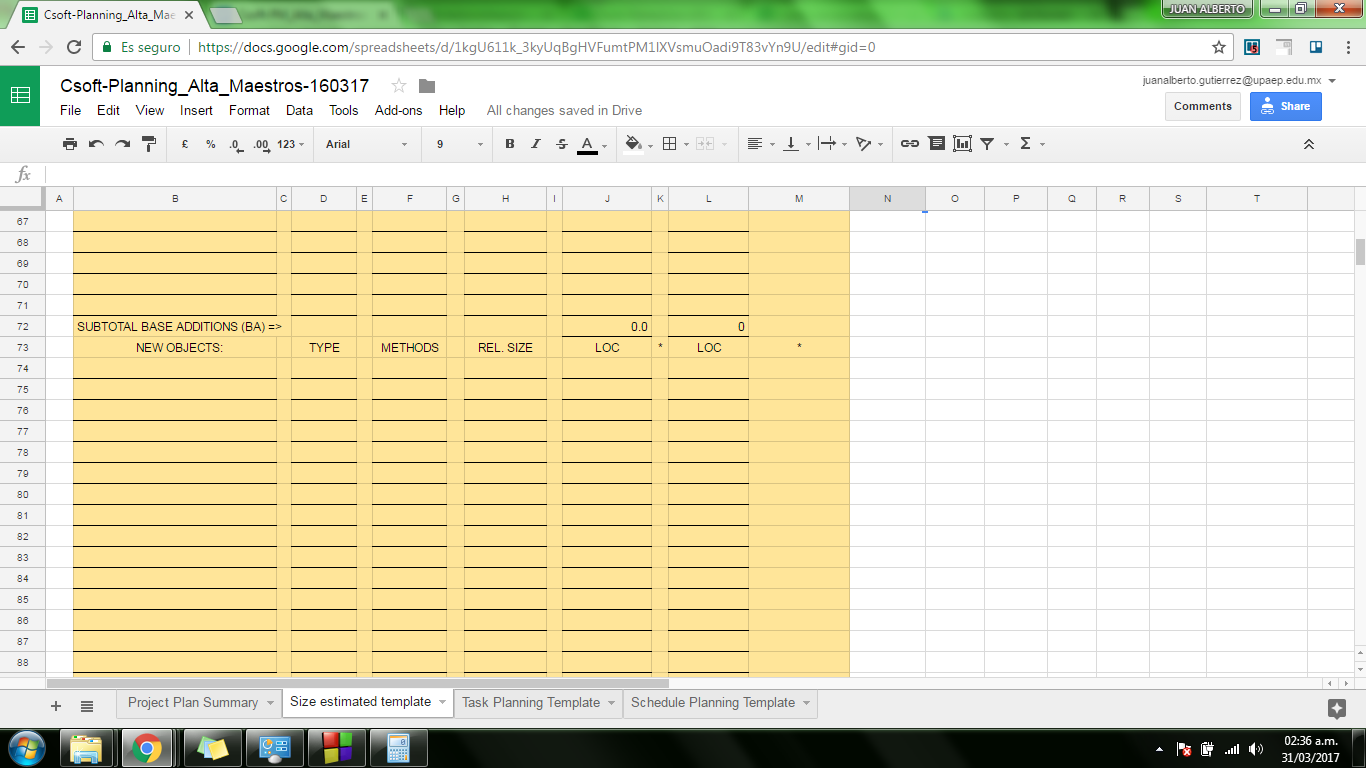
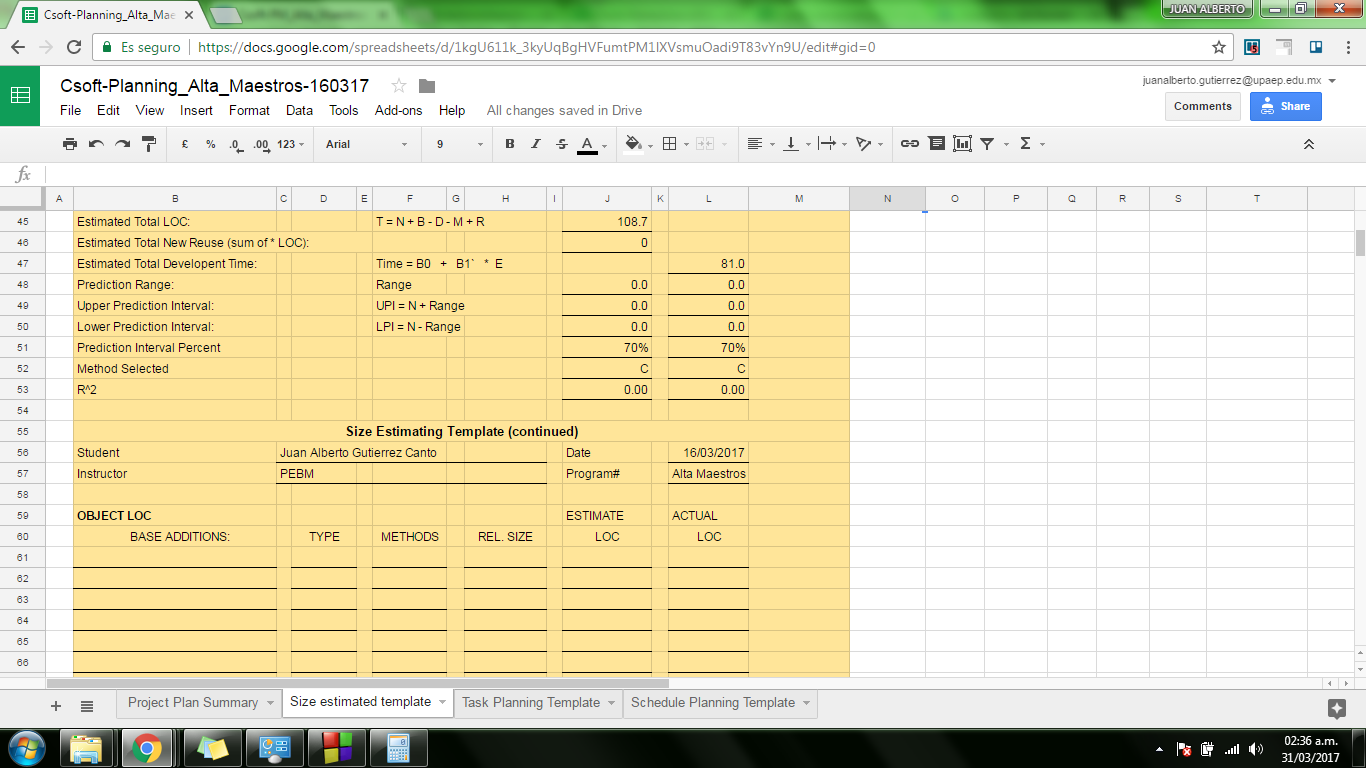
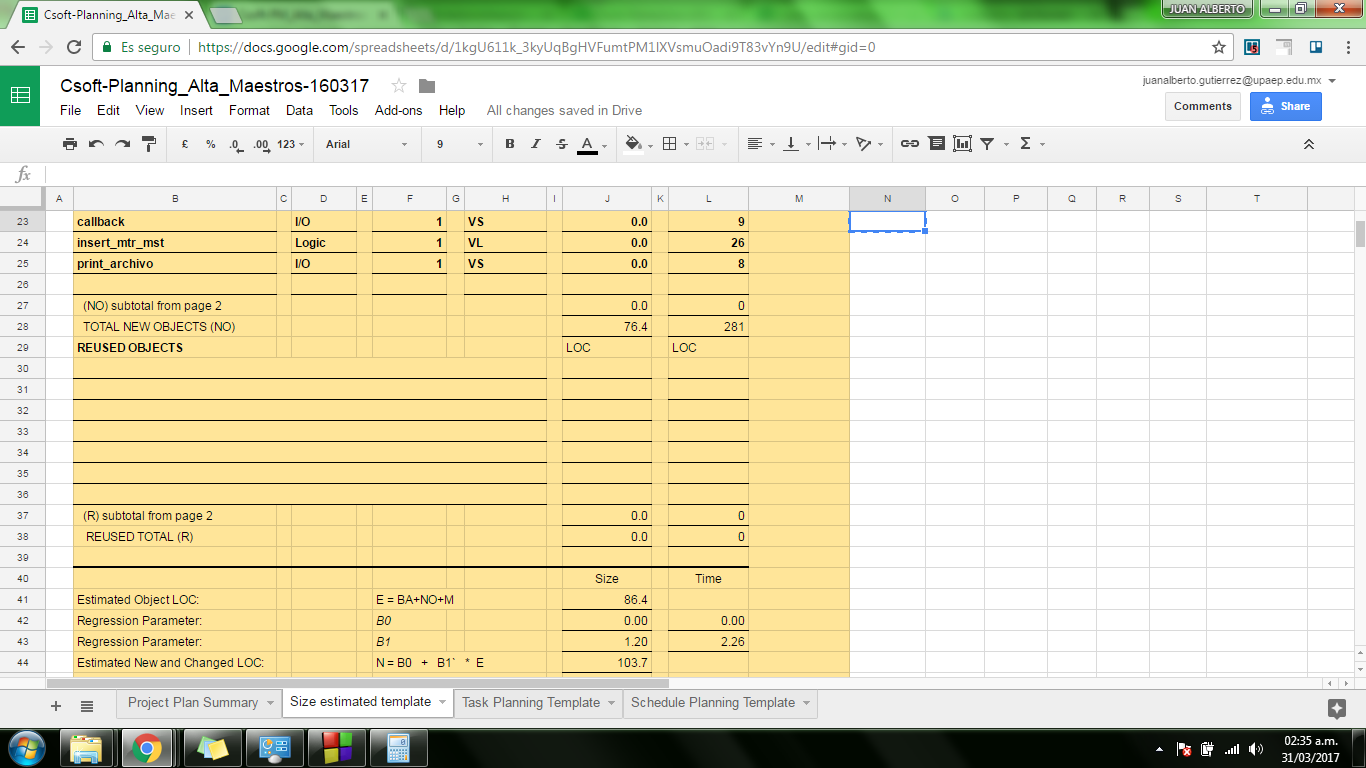
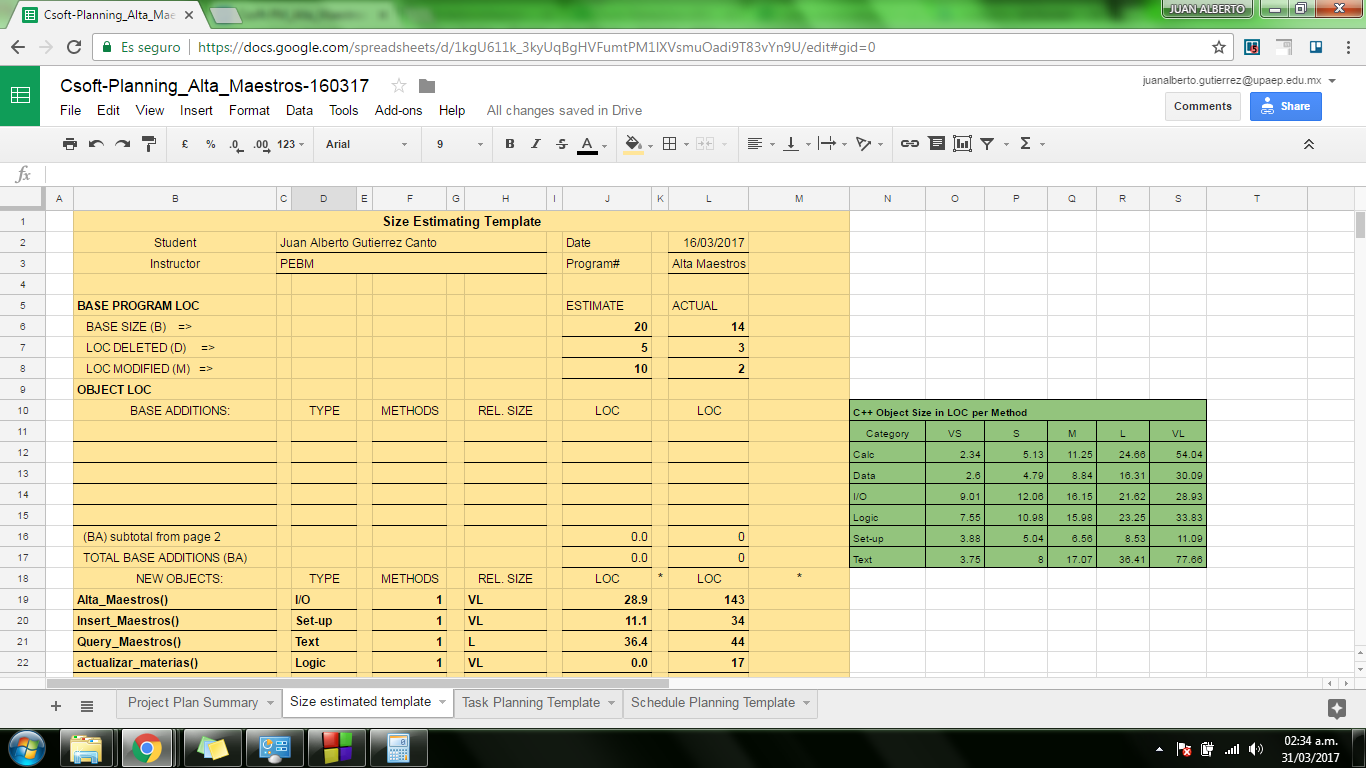
# Template de pruebas

|  |  |
| --- | --- |
| Nombre de la Prueba | Prueba de Alta de Maestros |
| Caso de Uso | 3.0 |
| Objetivo de la Prueba | Verificar que los datos de los maestros sean correctos en la base de datos. |
| Descripción de la Prueba | Se ingresará a sistema como Administrador y se seleccionará la opción de “Alta Maestros” |
| Test | 1. Direccionarse a la opción de Alta Maestros. 2. Ingresar Datos del Maestro. 3. Se le asigna materias para impartir. 4. Se le asigna materias para algún semestre. 5. Se confirma alta de maestro. 6. Se puede seguir agregando maestros (regresar al paso 1). |
| Resultados Esperados | Mensaje de Éxito:  Datos Enviados: (Se mostrará los datos almacenados en variables locales)  Nombres  Apellido paterno  Apellido materno  Fecha de nacimiento  Dirección  Correo electrónico  Teléfono  Datos Almacenados: (Se hará consulta a DB)  Nombres  Apellido paterno  Apellido materno  Fecha de nacimiento  Dirección  Correo electrónico  Teléfono  Mensaje de Error:  Datos Enviados: (Se mostrará los datos almacenados en variables locales)  Nombres  Apellido paterno  Apellido materno  Fecha de nacimiento  Dirección  Correo electrónico  Teléfono  “Error de consulta el Maestro 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 validación de la fecha  Se agregó limpieza de buffer para los arreglos char  Se validó más de dos nombre y dirección con espacios |
| Responsable de prueba | JAGC |
| Comentarios | Falta pantallas de las correcciones. |

# IX.- PIP

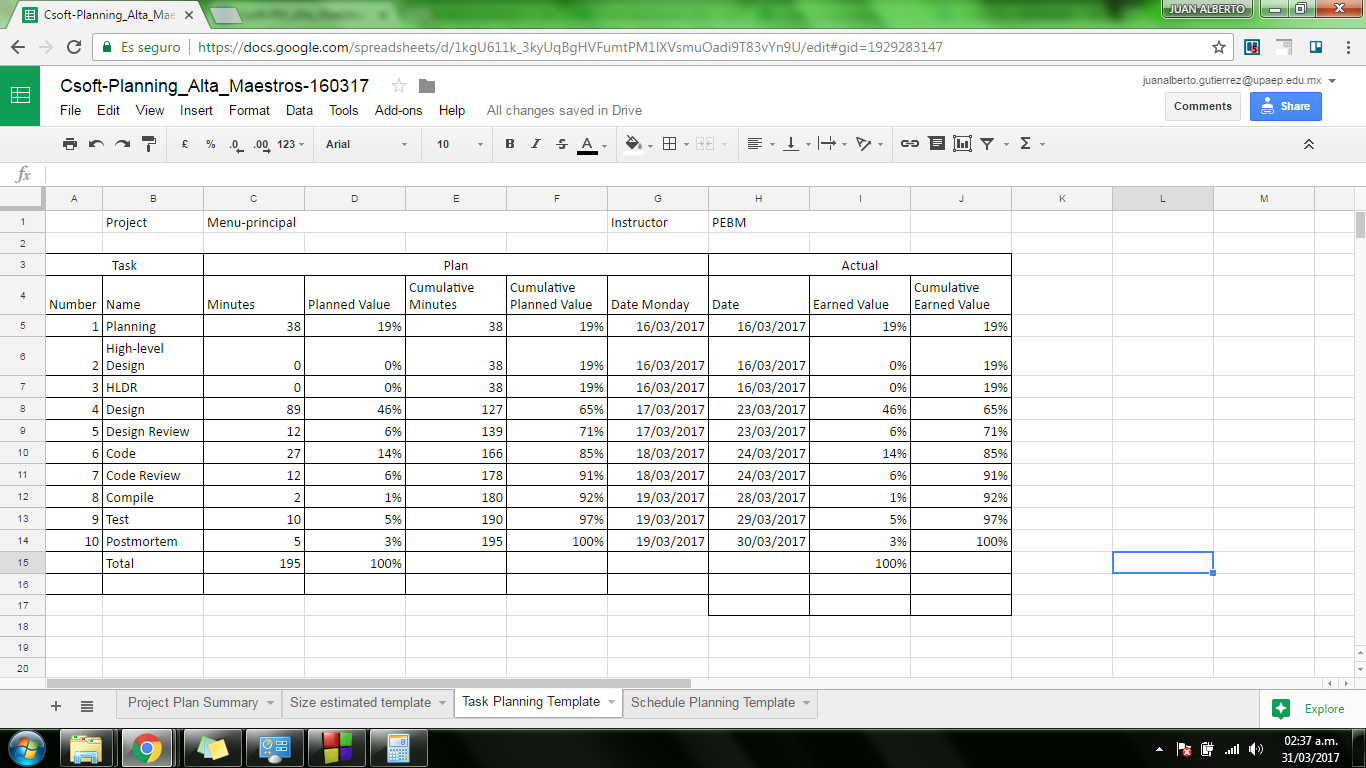
|  |  |  |
| --- | --- | --- |
| Menú Principal | | |
| No. PIP | Descripción del Problema | Descripción de la propuesta. |
| 2 | No se sabía el uso de sqlite3 | Preparar una asesoría antes de empezar el módulo. |
|  |  |  |
| Nombre Módulo | | |
| No. PIP | Descripción del Problema | Descripción de la propuesta. |

# X.- Size Estimating Template



# XI.- Task and Schedule

Task



Schedule

