|  |
| --- |
| OBLIGATORIO INFRAESTRUCTURA  23 de noviembre 2020  194592 - Facundo Rodriguez  196239 - Nicolas Arsel  156296 - Adrián Oses |

**PARTE A**

Utilizando el emulador Logic.ly, en su versión gratuita, se desarrollarán los

siguientes circuitos.

**1) Contador de 0 a 5 asincrónico, de uno en uno, ascendente, circular.**

Como el contador es asincrónico definimos la salida como el estado de salida, esto significa que cada salida **sn** será igual al estado **qn+1**, por ejemplo **s2 = q2\_n+1.**

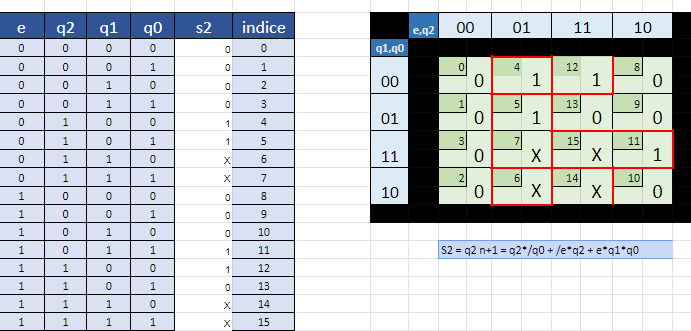
**Tabla de verdad**

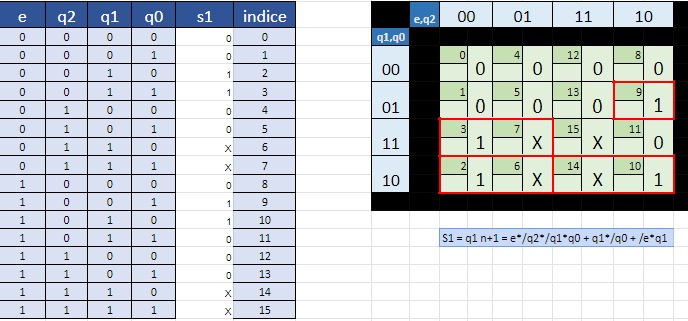
|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **e** | **q2\_n** | **q1\_n** | **q0\_n** | **q2\_n+1** | **q1\_n+1** | **q0\_n+1** | **s2** | **s1** | **s0** | **Valor representado** |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 |
| 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 2 |
| 0 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 3 |
| 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 4 |
| 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 5 |
| 0 | 1 | 1 | 0 | X | X | X | X | X | X |  |
| 0 | 1 | 1 | 1 | X | X | X | X | X | X |  |
| 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 |
| 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 2 |
| 1 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 3 |
| 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 4 |
| 1 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 5 |
| 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 1 | 0 | X | X | X | X | X | X |  |
| 1 | 1 | 1 | 1 | X | X | X | X | X | X |  |

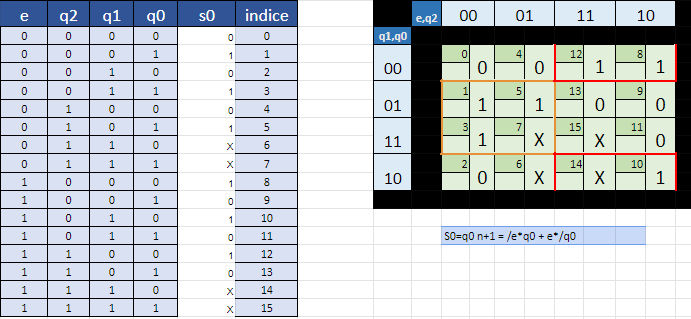
**Consideraciones:**

* El estado actual está conformado por q2\_n + q1\_n + q0\_n y la salida por s2 + s1 + s0
* En la tabla vemos cómo se representaría cada valor del 0 al 5 según la entrada y el estado en el cual se encuentre el contador.
* En caso de recibir como entrada 0, el estado actual y la salida quedan en el mismo valor.
* En caso de recibir como entrada 1, si el estado actual es menor que 5 incrementa el valor del estado y la salida en 1. Por otra parte si el estado actual representa a 5, el siguiente estado qn+1 será 0 al igual que las salidas

**Diagramas de Karnaugh**







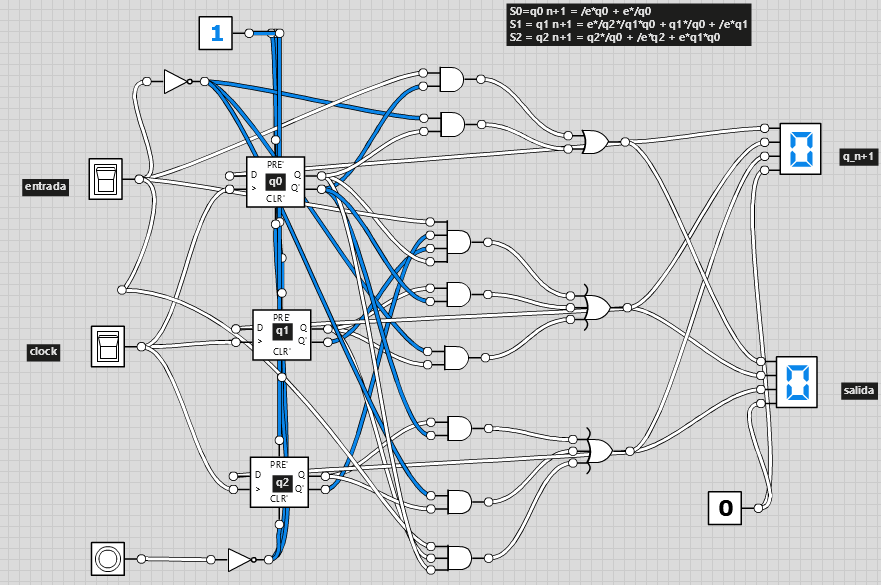
**Función simplificada**

S2 = q2 n+1 = q2\*/q0 + /e\*q2 + e\*q1\*q0

S1 = q1 n+1 = e\*/q2\*/q1\*q0 + q1\*/q0 + /e\*q1

S0 = q0 n+1 = /e\*q0 + e\*/q0

**Implementación en Logic.ly**



**2) Codificador con 8 entradas y 3 salidas. Código a elección.**

**Código elegido para la codificación**

|  |  |
| --- | --- |
| **Entrada** | **Salida** |
| e0 | 000 |
| e1 | 001 |
| e2 | 010 |
| e3 | 011 |
| e4 | 100 |
| e5 | 101 |
| e6 | 110 |
| e7 | 111 |

**Tabla de verdad simplificada**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **e7** | **e6** | **e5** | **e4** | **e3** | **e2** | **e1** | **e0** | **s2** | **s1** | **s0** |
| X | X | X | X | X | X | X | 1 | 0 | 0 | 0 |
| X | X | X | X | X | X | 1 | X | 0 | 0 | 1 |
| X | X | X | X | X | 1 | X | X | 0 | 1 | 0 |
| X | X | X | X | 1 | X | X | X | 0 | 1 | 1 |
| X | X | X | 1 | X | X | X | X | 1 | 0 | 0 |
| X | X | 1 | X | X | X | X | X | 1 | 0 | 1 |
| X | 1 | X | X | X | X | X | X | 1 | 1 | 0 |
| 1 | X | X | X | X | X | X | X | 1 | 1 | 1 |

**Consideraciones**

Para llegar a la función simplificada sólo lo hacemos por comprensión, no utilizamos el diagrama de Karnaug ya que el único caso que nos interesa para cada salida es saber que tecla se pulsó según el código que seleccionamos. Por ejemplo para **s2** sabemos que se pulsó e7 o e6 o e5 o e4, el resto de las entradas en realidad no nos interesan, son don’t care.

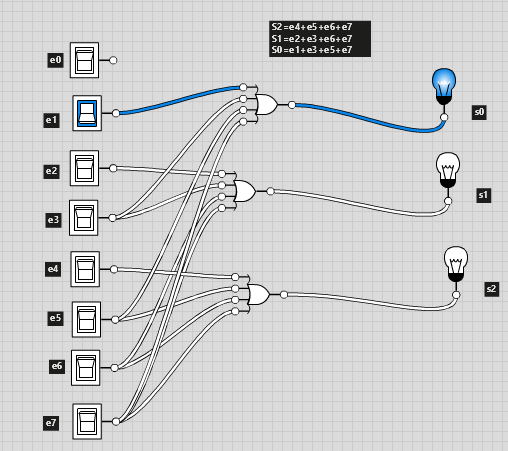
**Función simplificada**

S2 = e4+e5+e6+e7

S1 = e2+e3+e6+e7

S0 = e1+e3+e5+e7

**Implementación en Logic.ly**

****

**3) Decodificador que sea la inversa del anterior.**

Tabla de verdad

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **e2** | **e1** | **e0** | **s7** | **s6** | **s5** | **s4** | **s3** | **s2** | **s1** | **s0** |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Función simplificada**

Hallamos la siguiente función por comprensión:

s7 = e2.e1.e0

s6 = e2.e1./e0

s5 = e2./e1.e0

s4 = e2./e1./e0

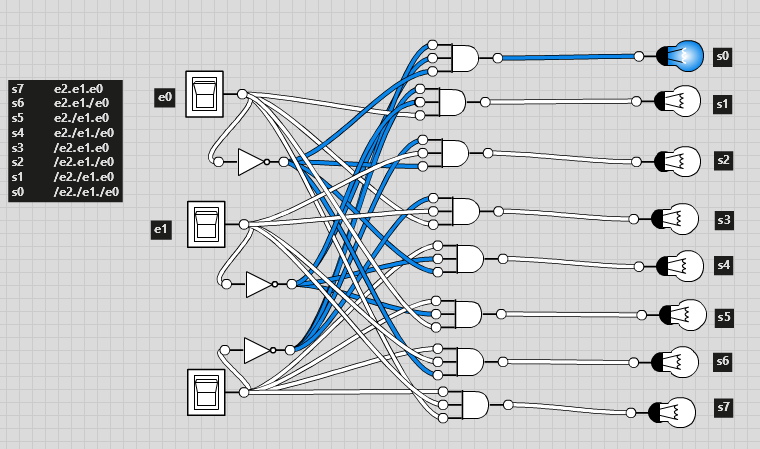
s3 = /e2.e1.e0

s2 = /e2.e1./e0

s1 = /e2./e1.e0

s0 = /e2./e1./e0

**Implementación en Logic.ly**

****

**PARTE B**

**Decisiones de diseño**

Todos los procesos deben tener por definición, un único recurso asignado y un usuario para su ejecución. Esto lo definimos así, porque entendemos que todos los procesos usan al menos un recurso. Un ejemplo de esto sería, el uso de RAM, Disco e implícitamente procesador y todos los sistemas operativos tienen al menos un usuario con permiso de ejecución.

Es importante mencionar que la relación entre proceso y recurso es 1 a 1. De haberlo relacionado con la tarea, deberíamos de haber implementado la lógica sobre qué recursos se pueden compartir simultáneamente, cuándo y cómo. Además de esto también deberíamos tener en cuenta las entradas prioritarias de los recursos de entrada y salida.. Esto agregaría complejidad a nuestro sistema y creemos está fuera del alcance.

**Procesos**

Principalmente está compuesto por un identificador, un nombre, y además tienen asociada una lista de tareas, un recurso, un usuario y un permiso. Esto último define qué tareas serán ejecutadas, el recurso que se utilizará y con qué permisos debe contar el usuario que ejecute el proceso. Esto es parte de la política de seguridad ya que mediante esta lógica establecemos que hay procesos que solo pueden ser ejecutados por un determinado grupo de usuarios, por ejemplo admin o root. Si el usuario pertenece al mismo grupo de permisos que tiene el proceso, este último se podrá ejecutar sin inconvenientes.

Para determinar la ejecución de los procesos usamos PEPS, es decir una cola en donde el primero que llega es el primero que se intentará ejecutar, siempre y cuando tenga recursos disponibles para hacerlo.

Una vez que el proceso termina su ejecución, es decir cuando todas las tareas que lo componen fueron realizadas, este sale de la lista de procesos liberando la memoria del mismo y el recurso asignado. Todo esto se simula mediante la interfaz de java List, en la cual se añaden todos los procesos a ejecutar y se van ejecutando según el orden de ingreso a la lista y la disponibilidad de los recursos necesarios para su ejecución.

Se definió que los recursos estén asociados al proceso y no a las tareas, de esa forma se evita que las tareas entren en conflicto por el uso de recursos.

**Tareas**

Se compone de un identificador y un nombre, esto último es a los efectos de simular la ejecución de una tarea, por ejemplo: leer archivo, escribir archivo, borrar archivo, etc.

**Recursos**

Se compone de un identificador, un nombre y una variable booleana que identifica si el recurso está en uso, es decir si un proceso lo tiene bloqueado. Una vez que el proceso que tiene bloqueado el recurso termina, se marca el mismo como que no está en uso mediante una variable boolean, entonces de esta forma se hace la devolución del mismo.

**Usuarios**

Se compone de un identificador, un nombre y además tiene asociado un permiso, es decir que el usuario pertenece a un determinado grupo de usuarios y puede ejecutar los procesos que tengan asignados ese tipo de permisos.

**Permisos**

Definimos tres tipos de permisos:

* **ADMIN**: puede ejecutar cualquier proceso.
* **USUARIO**: tiene limitaciones respecto del ADMIN, puede ejecutar procesos de USUARIO y de INVITADO.
* **INVITADO**: es el usuario que tiene más limitaciones, no puede ejecutar ningún proceso que pueda comprometer la seguridad del sistema. Solo puede ejecutar los procesos asignados a su tipo de permisos.

**Estados y ejecución de los procesos**

Para la ejecución y administración de los procesos utilizamos el modelo Roud Robin con 3 estados:

* **LISTO**: el proceso se crea y está pronto para ser ejecutado
* **EN\_EJECUCION**: el proceso se está ejecutando, es el estado previo a la asignación de un recurso.
* **BLOQUEADO**: los procesos en este estado, son los que tienen un recurso asignado y aún tienen tareas pendientes. Una vez terminada su ejecución, los mismos liberan el recurso y salen de la cola de ejecución.

Los demás estados SUSPENDIDO\_LISTO y SUSPENDIDO\_BLOQUEADO no son utilizados ya que no manejamos prioridades, por lo que no se tiene en cuenta el manejo de dispositivos E/S.

**Unidad de tiempo**

Se definió el *quantum* como la unidad de tiempo en la cual se realizará una tarea, corresponde a un segundo. Esto es a los efectos de poder determinar el timeout de un proceso. De esta forma se puede determinar cuánto tiempo se necesita para ejecutar un determinado proceso y agregamos un control que establece un máximo de tiempo de ejecución en base a una relación exponencial entre los *quantum* de las tareas por los procesos.

**Scheduling**

Elegimos una política de planificación de Round Robin, nuestro argumento en base a esta decisión es que nos parece la más completa para implementar, ya que su diagrama de flujo en base a estados de proceso, provee significativas ventajas en base a sus competidores.

Es fácil de implementar, lo cual hace que el desarrollo del sistema sea ligero, los algoritmos de cambios de estados son predecibles, lo cual conlleva un mejor manejo de errores y la sistematización de prioridades ofrecen un mejor control de ejecución.

**Interfaz de usuario**

Se desarrolló una interfaz de consola en donde se permite ingresar datos para simular la ejecución de procesos.

La interfaz permite:

1) Gestión de usuarios

2) Gestión de recursos

3) Gestión de procesos

4) Precargar lista

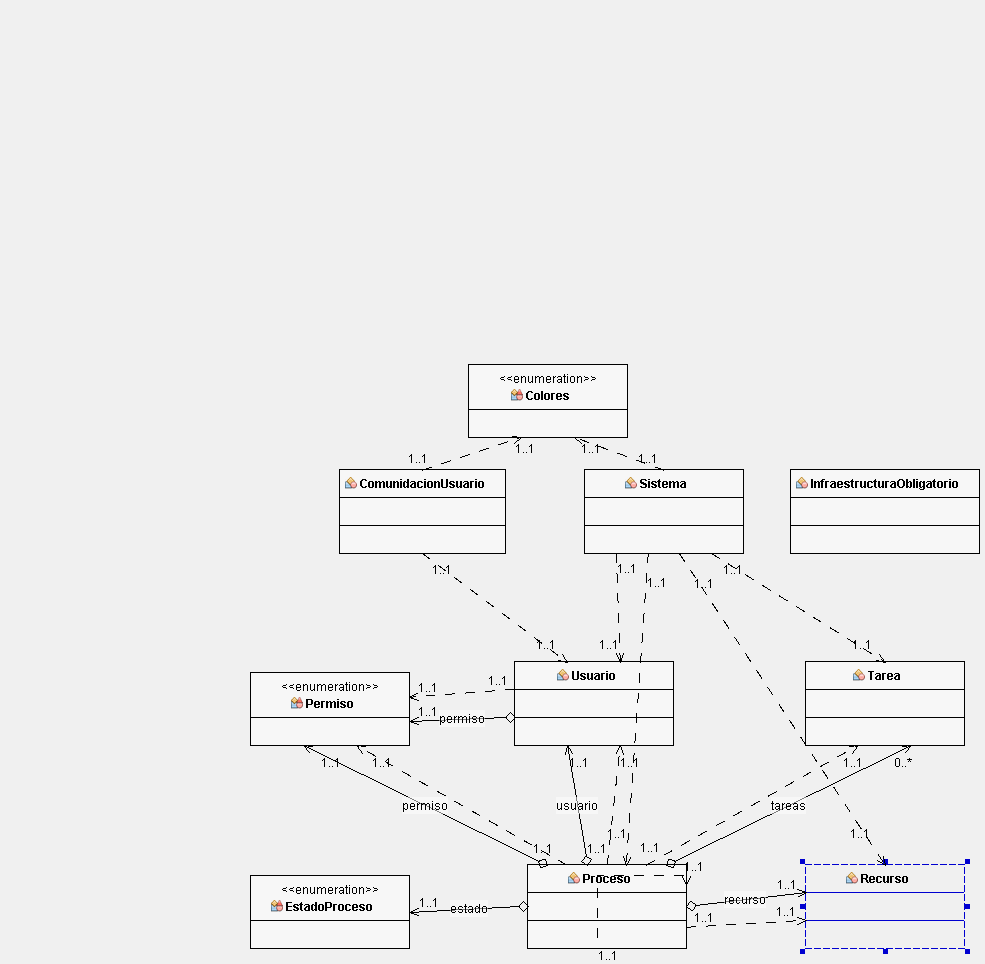
5) Ejecutar

6) Visualizar Datos

7) Salir

**Diagrama de clases**

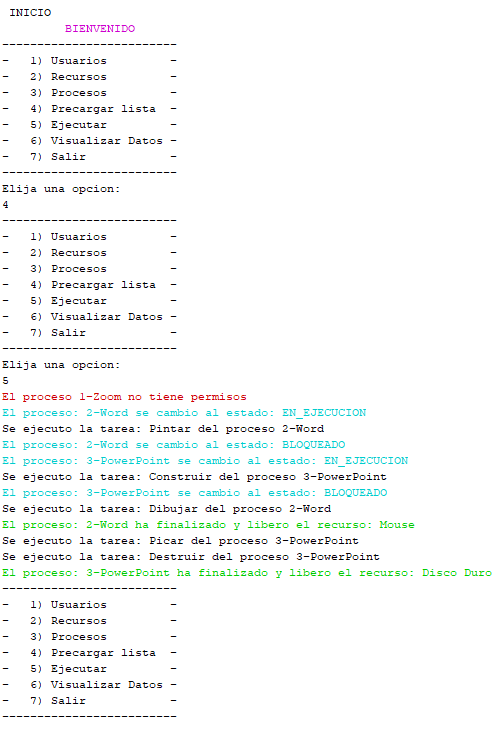
Para simplificarlo, en esta imagen, obviamos los atributos y propiedades

****

Se adjunta la versión completa del diagrama de clases.

**Pruebas**

Prueba con datos precargados:

****

**Clases de equivalencia**

1.1) variables de entradas

* Lista de usuarios
* Lista de recursos
* Lista de procesos
* Lista de tareas

|  |  |  |
| --- | --- | --- |
| **Entrada / Variable** | **Clases Válidas** | **Clases inválidas** |
| Usuario | Información válida (1) | El nombre del usuario no puede ser vacío (2), Error de validación de información en menú de navegación usuarios (3) |
| Recurso | Información válida (4) | El nombre del recurso no puede ser vacío (5), Error de validación de información en menú de navegación recursos (6) |
| Proceso | Despliegue el menú de Procesos, información válida. (7) | Para ingresar a esta opción, la lista de usuarios/recursos no puede ser vacía (8).Error de validación de información en menú de navegación procesos(9) |
| Tareas | Despliegue menú de tareas (10) | Error de validación de información menú tareas(11) |

**Caso de prueba alta de usuario (CPAU)**

|  |  |  |  |
| --- | --- | --- | --- |
| **Caso de prueba** | **Usuario** | **Resultado esperado** | **Clases de equivalencia** |
| CPAU1 | Accede al menú de usuarios, selecciona crear usuario, no ingresa el nombre de usuario | El sistema informa el error “El nombre del usuario no puede ser vacío” | 2 |
| CPAU2 | Accede al menú de usuarios, selecciona crear usuario, ingresa el nombre de usuario | El sistema despliega el menú de asignación de permisos | 1 |
| CPAU3 | El usuario selecciona una opción no válida en la asignación de permisos | Si el caracter ingresado es un texto, el sistema informa: “Debe ingresar un número” | 3 |
| CPAU4 | El usuario selecciona una opción no válida en la asignación de permisos | Si el caracter ingresado es un número, el sistema informa: “El numero ingresado debe estar entre 1 y n (tamaño de lista del menú de permisos)” | 3 |
| CPAU5 | El usuario selecciona correctamente un nivel de permisos para el usuario | El sistema informa “Se creó con éxito el usuario <Nombre del usuario>” | 1 |

**Caso de prueba alta de recurso (CPAR)**

|  |  |  |  |
| --- | --- | --- | --- |
| **Caso de prueba** | **Usuario** | **Resultado esperado** | **Clases de equivalencia** |
| CPAR1 | Accede al menú de recursos, selecciona crear recurso, no ingresa el nombre del recurso | El sistema informa el error “El nombre del recurso no puede ser vacío” | 5 |
| CPAR2 | Accede al menú de recursos, selecciona crear recurso, ingresa el nombre del recurso | El sistema informa “Se creó con éxito el recurso <Nombre recurso>” | 4 |
| CPAR3 | El usuario selecciona una opción no válida en el menú de navegación de recursos | Si el caracter ingresado es un texto, el sistema informa: “Debe ingresar un número” | 6 |
| CPAR4 | El usuario selecciona una opción no válida en el menú de navegación de recursos | Si el caracter ingresado es un número, el sistema informa: “El numero ingresado debe estar entre 1 y 3 (tamaño de lista de opciones de menú recursos)” | 6 |

# 

# **Caso de prueba alta de proceso (CPAP)**

|  |  |  |  |
| --- | --- | --- | --- |
| **Caso de prueba** | **Usuario** | **Resultado esperado** | **Clases de equivalencia** |
| CPAP1 | Accede al menú de Procesos, sin previamente haber dado de alta al menos un usuario | El sistema informa el error “Para ingresar a esta opción, la lista de usuarios no puede ser vacía” | 8 |
| CPAP2 | Accede al menú de Procesos, sin previamente haber dado de alta al menos un recurso | El sistema informa el error “Para ingresar a esta opción, la lista de recursos no puede ser vacía” | 8 |
| CPAP3 | Accede al menú de Procesos, habiendo previamente creado al menos un usuario y un recurso. | El sistema despliega el menú de opciones de proceso. | 7 |
| CPAP4 | Accede al menú de Procesos, habiendo previamente creado al menos un usuario y un recurso, seleccionando una opción inválida. | Si el caracter ingresado es un número, el sistema informa: “El numero ingresado debe estar entre 1 y 3 (tamaño de lista de opciones de menú procesos)” | 9 |
| CPAP5 | Accede al menú de Procesos, habiendo previamente creado al menos un usuario y un recurso, seleccionando una opción inválida. | Si el caracter ingresado es un texto, el sistema informa: “Debe ingresar un número” | 9 |
| CPAP6 | Accede al menú de Procesos, habiendo previamente creado al menos un usuario y un recurso, seleccionando la opción crear proceso. | Independientemente de si el usuario ingresa un nombre o no, el sistema despliega el menú de asignación de permisos. Los procesos se referencian por id. | 7 |
| CPAP7 | Accede al menú de Procesos, habiendo previamente creado al menos un usuario y un recurso, seleccionando la opción crear proceso, seleccionando una opción inválida. | Si el caracter ingresado es un número, el sistema informa: “El numero ingresado debe estar entre 1 y 3 (tamaño de lista de opciones de menú procesos-Asignación de permisos)” | 9 |
| CPAP8 | Accede al menú de Procesos, habiendo previamente creado al menos un usuario y un recurso, seleccionando la opción crear proceso, seleccionando una opción inválida. | Si el caracter ingresado es un texto, el sistema informa: “Debe ingresar un número” | 9 |
| CPAP9 | Accede al menú de Procesos, habiendo previamente creado al menos un usuario y un recurso, seleccionando la opción crear proceso, habiendo asignado correctamente un nivel de permisos. | El sistema despliega el menú de tareas | 7 |
| CPAP10 | Accede al menú de Procesos, habiendo previamente creado al menos un usuario y un recurso, seleccionando la opción crear proceso, habiendo asignado correctamente un nivel de permisos y habiendo creado al menos una tarea | El sistema despliega una lista de usuarios en base al nivel seleccionado de acceso. | 7 |
| CPAP11 | Accede al menú de Procesos, habiendo previamente creado al menos un usuario y un recurso, seleccionando la opción crear proceso, habiendo asignado correctamente un nivel de permisos y habiendo creado al menos una tarea, pero no existen usuarios creados para ese nivel de permisos. | El sistema despliega ”No existe ningún usuario con el permiso: <Nivel de permiso> asignado al proceso, Vuelva a intentarlo” | 9 |
| CPAP12 | Se ingresa información errónea en el menú de selección de usuarios para el proceso. | El sistema informa un error en base al tipo de información errónea | 9 |
| CPAP13 | Se selecciona un usuario para el proceso, de la lista de usuarios disponibles. | El sistema despliega el siguiente mensaje ”Se asignó correctamente el usuario: <Nombre de usuario>” y procede a desplegar el menú de asignación de recursos | 7 |
| CPAP14 | Se ingresa información errónea en el menú de selección de usuarios para el proceso. | El sistema informa un error en base al tipo de información errónea | 9 |
| CPAP15 | Se selecciona un recurso de la lista de recursos disponibles para el proceso. | El sistema informa:”Se asignó correctamente el recurso: <Nombre recurso>”  “Se creo con éxito el PROCESO: <Nombre proceso> con ESTADO: LISTO” y se despliega nuevamente el menú de procesos | 7 |

**Caso de prueba alta de tareas (CPAT)**

|  |  |  |  |
| --- | --- | --- | --- |
| **Caso de prueba** | **Usuario** | **Resultado esperado** | **Clases de equivalencia** |
| CPAT1 | Accede al menú de Tareas, a través del alta de proceso  seleccionando una opción inválida. | Si el caracter ingresado es un número, el sistema informa: “El numero ingresado debe estar entre 1 y 2” | 11 |
| CPAT2 | Accede al menú de Tareas, a través del alta de proceso  seleccionando una opción inválida. | Si el caracter ingresado es un texto, el sistema informa: “Debe ingresar un número” | 11 |
| CPAT3 | Accede al menú de Tareas, a través del alta de proceso ingresando la tarea | El sistema despliega el mensaje “Se agregó correctamente la tarea” y luego consulta si se quieren agregar más tareas | 10 |

# **Calidad de código**

Se siguieron los siguientes elementos básicos:

1. Comienzo y fin de estructuras
2. Agrupamiento de sentencias
3. Indentación, anidamiento
4. Nombres de elementos
5. Mayúsculas y minúsculas
6. Paréntesis
7. Espaciado de parámetros
8. Clases comienzan en mayúscula
9. Paquetes y métodos comienzan en minúscula
10. Uso de camelCase para diferenciar palabras
11. Constantes literales todo en mayúsculas
12. Llaves estilo Kernighan & Ritchie

# **Conclusiones**

**Trabajo en equipo**

Con base al grupo formado, tomamos decisiones en conjunto en base a la experiencia profesional y las materias ya cursadas, para dividir las tareas. El core del obligatorio se hizo en conjunto y cada decisión compleja fue tomada en una reunión virtual del equipo, pero luego cada uno en base a los puntos mencionados fue tomando y realizando las tareas correspondientes al rol asignado.

**Lecciones de materia**

Al intentar reproducir cómo funciona la planificación, preparación y ejecución de tareas de un sistema operativo, en java, nos vimos forzados a repasar los conceptos del curso. Esto provocó que se solidifican conceptos previamente adquiridos y generó la necesidad de aprender algunos nuevos.

Cuando pusimos en práctica estos conceptos, fue que logramos una mejor comprensión de éstos y entendimos la necesidad de desarrollar los algoritmos de planificación de ejecución de las cpu. También generamos discusiones sobre cómo los hubiéramos implementado nosotros y que le mejoraríamos a los ya implementados, intentando asociar experiencias reales de nuestro día a día en el uso de nuestros sistemas operativos, a los conceptos teóricos aprendidos en el curso.

NOTA: Puede encontrar el resumen del trabajo en <https://github.com/196239/Infraestructura-194592-196239-156296>, los commits no definen la carga de trabajo que tuvo caga compañero ya que, por lo general, nos reuniomos e hicimos el obligatorio en conjunto.