**Universidad Autónoma de Yucatán**

**Facultad de Matemáticas**

Construcción de software

5to semestre, grupo B

**Integrantes:**

Russel Bonilla Pech

Jaqueline Góngora Tun

Frida Pineda Alvarado

Víctor Rosado Koyoc

Luisa Villanueva Díaz

**Profesor:** Edwin León

**Fecha de entrega:** 15 de Noviembre de 2024

**Plan de Trabajo**

# **Descripción General**

El proyecto consiste en desarrollar un sistema de administración para una cafetería utilizando Java. Este sistema permitirá gestionar las actividades diarias de la cafetería, enfocándose en tres tipos de usuarios: **cliente, empleado y administrador.** Cada tipo de usuario tendrá funcionalidades específicas según su rol dentro de la organización.

El sistema tendrá una interfaz gráfica donde cada usuario podrá acceder a su respectivo panel de control y realizar las acciones correspondientes.

1. **Cliente**: Puede ver el menú actualizado, hacer pedidos, consultar el estado de su pedido y proporcionar retroalimentación sobre los productos, así como calificarlos.
2. **Empleado**: Gestiona los cobros de los pedidos que hacen los clientes. Puede registrar un pedido como pagado y confirmar la entrega.
3. **Administrador**: Tiene acceso completo para gestionar el menú, agregar o eliminar productos, ajustar precios, y generar reportes sobre el historial de pedidos y ventas. Además, puede revisar y gestionar las retroalimentaciones y calificaciones de los productos.

## **Supuestos y limitaciones**

El sistema debe estar funcionando para el día 3 de Diciembre del 2024.

## **Entregables del proyecto**

Los siguientes productos se producirán durante el proyecto:

* Diagrama de clases de la aplicación
* Código de la aplicación.
* Lista de todas las prácticas de construcción de software abordadas

#### 

# **Requisitos del sistema**

## **1. Gestión de Usuarios**

* El sistema debe permitir el inicio de sesión para tres tipos de usuarios: cliente, empleado y administrador.
* El sistema debe tener un mecanismo de autenticación para validar a los usuarios por tipo (empleado, administrador).
* El sistema debe mostrar diferentes interfaces para cada usuario según sus permisos.

## **2. Funcionalidades del Cliente**

* El cliente debe poder visualizar el menú de productos disponibles en la cafetería.
* El cliente debe poder seleccionar productos del menú para hacer un pedido.
* El cliente debe poder ver el total del pedido antes de confirmarlo.
* El sistema debe permitir al cliente confirmar y enviar el pedido a los empleados para su procesamiento.
* El cliente debe recibir una confirmación de que su pedido ha sido registrado exitosamente.
* El cliente debe poder proporcionar retroalimentación sobre los productos que ha pedido.
* El cliente debe poder calificar los productos en una escala de 1 a 5 estrellas.

## **3. Funcionalidades del Empleado**

* El empleado debe poder ver todos los pedidos pendientes realizados por los clientes.
* El empleado debe poder registrar el cobro de cada pedido.
* El empleado debe poder marcar un pedido como entregado una vez se haya realizado el cobro.
* El empleado debe poder ver el estado de los pedidos (pendiente, entregado, pagado).

## **4. Funcionalidades del Administrador**

* El administrador debe poder visualizar un historial completo de pedidos realizados por los clientes.
* El administrador debe poder agregar, editar y eliminar productos del menú.
* El administrador debe poder modificar el precio de los productos en el menú.
* El administrador debe tener acceso a reportes diarios, semanales y mensuales de las ventas.
* El administrador debe poder revisar y gestionar las retroalimentaciones y calificaciones de los productos.

# **Descripción de pruebas**

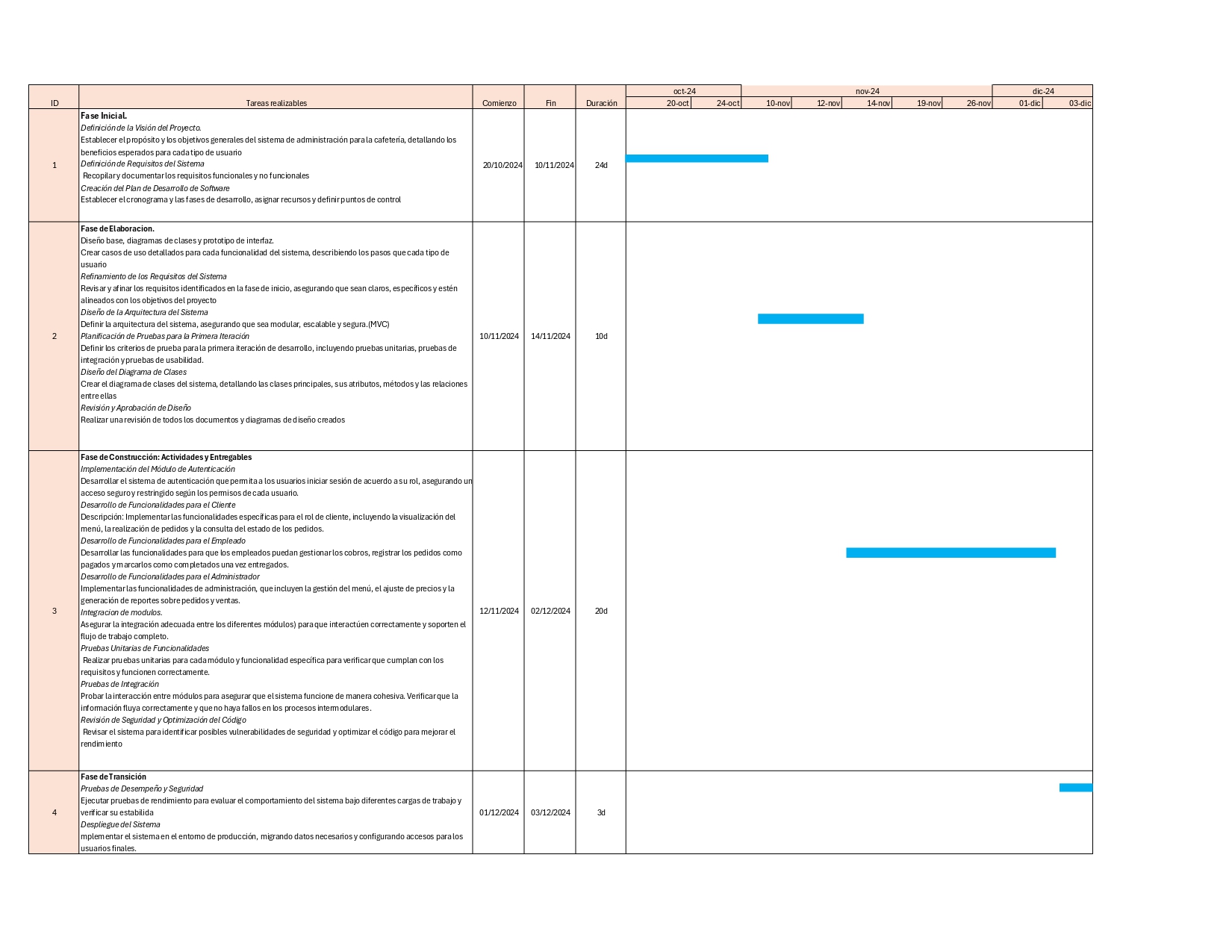
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Num. Prueba | Nombre | Descripción | Entrada | Resultado esperado |
| 1 | Autenticar de usuario válido | Un usuario con credenciales correctas puede autenticarse exitosamente en el sistema. | Usuario y contraseña válidos. | El sistema devuelve true. |
| 2 | Rechazar usuario inválido | El sistema debe rechazar usuarios con credenciales incorrectas. | Usuario válido, pero contraseña incorrecta. | El sistema devuelve false. |
| 3 | Realizar un pedido exitoso | El cliente puede realizar un pedido seleccionando productos. | Cliente y una lista de productos seleccionados. | Se genera un pedido válido que contiene los productos seleccionados y está asociado al cliente. |
| 4 | Calificar producto | Un cliente puede calificar un producto y el sistema actualiza la calificación promedio correctamente. | Producto específico y una calificación dada por el cliente. | La calificación promedio del producto se actualiza. Comentario registrado en la base de datos. |
| 5 | Modificar el precio de un producto | El administrador puede modificar el precio de un producto. | Producto existente y un nuevo precio. | El precio del producto se actualiza correctamente. |
| 6 | Actualizar estado de pedido | Un empleado puede cambiar el estado de un pedido a "Completado". | Pedido existente | El estado del pedido en la base de datos se actualiza a "Pagado". |
| 7 | Calcular total de un pedido | El total de un pedido se calcula correctamente. | Lista de productos. | El sistema devuelve el costo total de la compra. |
| 8 | Generar reporte | El administrador puede generar un reporte de ventas mensuales. | Mes del reporte requerido. | Reporte generado con: número total de pedidos, total de ingresos y productos más vendidos. |
| 9 | Agregar un nuevo producto | El administrador puede añadir productos al menú. | Producto a añadir. | El producto aparece en la lista de productos disponibles para los clientes. Registro en la base de datos exitoso. |
| 10 | Eliminar un producto existente | El administrador pueda eliminar un producto del menú | Producto a eliminar. | El producto deja de aparecer en el menú. Registro eliminado correctamente en la base de datos. |

# **Roles y responsabilidades**

La siguiente tabla muestra las funciones representadas en el diagrama del proyecto y de sus responsabilidades primarias.

|  |  |  |
| --- | --- | --- |
| Nombre | Rol | Descripción |
| Russel Bonilla | **Líder de Proyecto** | Responsable de coordinar y supervisar todas las actividades del proyecto, asegurando el cumplimiento de plazos y objetivos. |
| Jaqueline Gongora | **Especialista en Pruebas** | Se encarga de realizar pruebas del sistema para identificar y documentar errores. Garantiza que el sistema cumpla con los requisitos. |
| Frida Pineda | **Desarrollador de Funcionalidades del Sistema** | Encargado de implementar las funcionalidades relacionadas con la gestión de usuarios y la lógica de las operaciones de cliente, empleado y administrador. |
| Victor Rosado | **Desarrollador de Funcionalidades del Sistema** | Encargado de implementar las funcionalidades relacionadas con la gestión de usuarios y la lógica de las operaciones de cliente, empleado y administrador. |
| Luisa Villanueva | **Desarrollador de Interfaz Gráfica** | Responsable de la creación y diseño de la interfaz gráfica del sistema, asegurando que las vistas de cliente, empleado y administrador sean intuitivas y responsivas. |

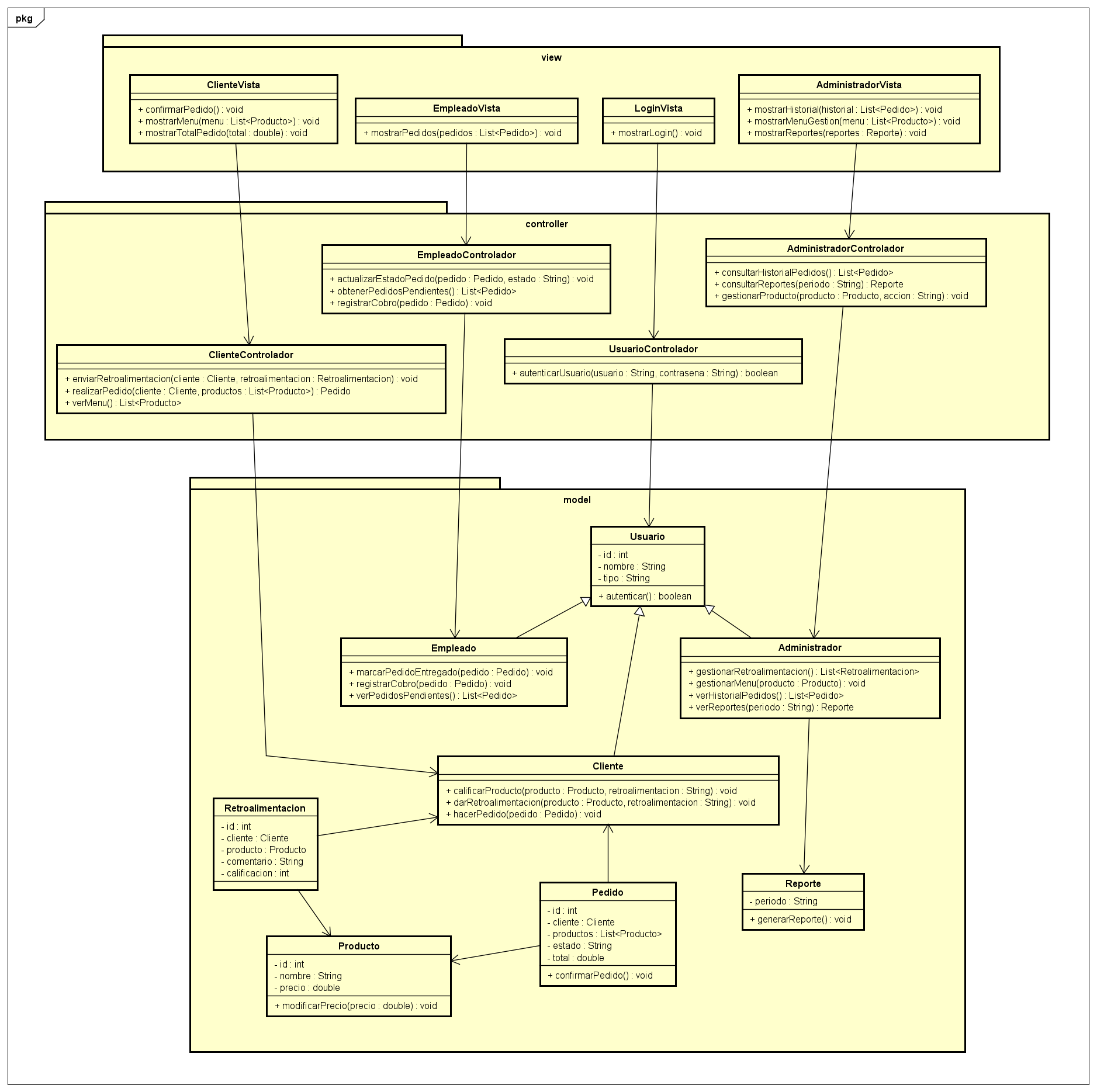
# **Cronograma**



**Link del cronograma:** [**Cronograma.pdf**](https://alumnosuady-my.sharepoint.com/:b:/g/personal/a19203622_alumnos_uady_mx/EakqXO-TwxxBmqy33gT6u88BKNpGArACHTLILWM6onHT8g?e=cVeEac)

**Repositorio de proyecto:** [**https://github.com/soy-russ-bp/prueba1**](https://github.com/soy-russ-bp/prueba1)

# **Diagrama de clases**

****

# **Checklist de prácticas de construcción**

|  |  |
| --- | --- |
| **Lineamiento de la asignatura** | **Cumple (Sí / No)** |
| ***Técnicas de construcción de variables y tipos de datos fundamentales*** | |
| Inicializar cada variable cuando se declara. |  |
| Idealmente, declarar y definir cada variable cerca de donde se usa. |  |
| Inicializar los datos de los miembros de una clase en su constructor. |  |
| Los nombres de las variables deben usar camelCase. |  |
| Usar calificadores de cálculos en nombres de variables. |  |
| Usar opuestos comunes en nombre de variables. Usar los opuestos con precisión. |  |
| Nombramiento de índices de bucle. Si tiene que usar i, j y k, no usar para otra cosa que no sean índices de bucle. |  |
| Si tiene varios bucles anidados, asignar nombres más largos a las variables de bucle para mejorar la legibilidad. |  |
| Pensar en un nombre mejor que “flag” para las variables de estado. Una bandera nunca debe tener “flag” en su nombre. |  |
| Utilizar nombres descriptivos para variables temporales en lugar de llamarlas como “temp” o “x”. |  |
| Tener en cuenta los nombres booleanos típicos (done, error, found, success). |  |
| Asignar nombres de variables booleanas que impliquen Verdadero o Falso. |  |
| Utilizar nombres de variables booleanos positivos. |  |
| Identificar constantes nombradas. Usar todas las letras mayúsculas, posiblemente con guiones bajos para separar las palabras, RECSMAX o RECS\_ MAX. |  |
| Diferenciar entre nombres de variables y nombres de rutinas. |  |
| Identificar parámetros de solo entrada (i*nput-only*). |  |
| Hacer que las conversiones de tipo sean obvias. |  |
| Revisar el desbordamiento de enteros. |  |
| Revisar si hay desbordamiento en resultados intermedios. |  |
| Evitar cadenas y caracteres mágicos |  |
| Estar atento a los errores off-by-one |  |
| ***Técnicas de organización de sentencias*** |  |
| Organizar el código para que las dependencias sean obvias. |  |
| Nombrar las rutinas para que las dependencias sean obvias. |  |
| Usar parámetros de rutina para hacer que las dependencias sean obvias. |  |
| ***Técnicas de construcción de estructuras de control de flujo*** |  |
| Escribir primero la ruta nominal a través del código; luego escribir los casos inusuales. |  |
| Asegurarse de bifurcar correctamente en igualdad. |  |
| Seguir la cláusula if con una declaración significativa. |  |
| Poner los casos más comunes primero. If-then-else. |  |
| Ordenar casos por frecuencia en las declaraciones case |  |
| ***Técnicas de construcción de procedimientos*** |  |
| Para nombrar una función, usar una descripción del valor devuelto. Por ejemplo, cos(), customerld.Next(), printer.IsReady0 y pen.CurrentColor(). |  |
| Para nombrar un procedimiento, utilizar un verbo fuerte seguido de un objeto. Por ejemplo: PrintDocument(), CalcMonthlyRevenues(),  CheckOrderInfo() y RepaginateDocument). |  |
| Poner las variables de estado o error al final en las rutinas. |  |
| No utilizar parámetros de rutina como variables de trabajo. |  |
| Usar parámetros con nombres significativos. |  |
| ***Técnicas de construcción de clases*** |  |
| Los nombres de las clases deben escribirse con PascalCase y deben ser sustantivos en singular. |  |
| Los nombres de las interfaces deben escribirse con PascalCase. |  |
| Preferir la herencia a la verificación extensiva de tipos. |  |
| Si varias clases comparten un comportamiento común, pero no datos, derivarlas de una clase base común que defina las rutinas comunes. |  |
| Si varias clases comparten datos y comportamientos comunes, heredar de una clase base común que define las rutinas y los datos comunes. |  |
| Heredar cuando desea que la clase base controle la interfaz; contener cuando desea controlar su interfaz. |  |
| Hacer cumplir la propiedad singleton mediante el uso de un constructor privado. |  |
| ***Técnicas de documentación de código*** |  |
| Usar espacios en blanco para separar elementos de una sentencia no muy relacionados o mostrar precedencia de operadores. |  |
| Uso de la sangría |  |
| Escribir comentarios que describan la intención del código. |  |
| Hacer que cada comentario cuente. No escribir más comentarios, hacer que el código sea más legible. |  |
| Centrar los comentarios de los párrafos en el por qué en lugar del cómo. |  |