



Servicio de Taxis

Programación Orientada a Objetos

Equipo 9:

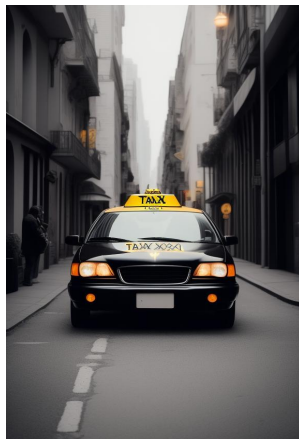
Gordillo Solis Ivan Javier
Legorreta Flores Karla Valeria
Oropeza Hernández Bruno Gabriel

Descripción del problema

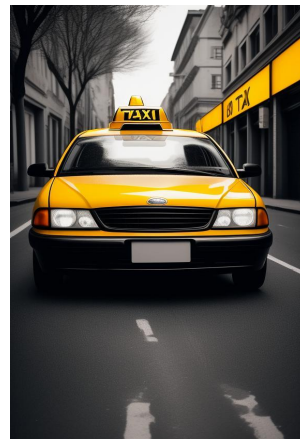
Para establecer un sistema de taxis se requieren de tres tipos de unidades diferentes:



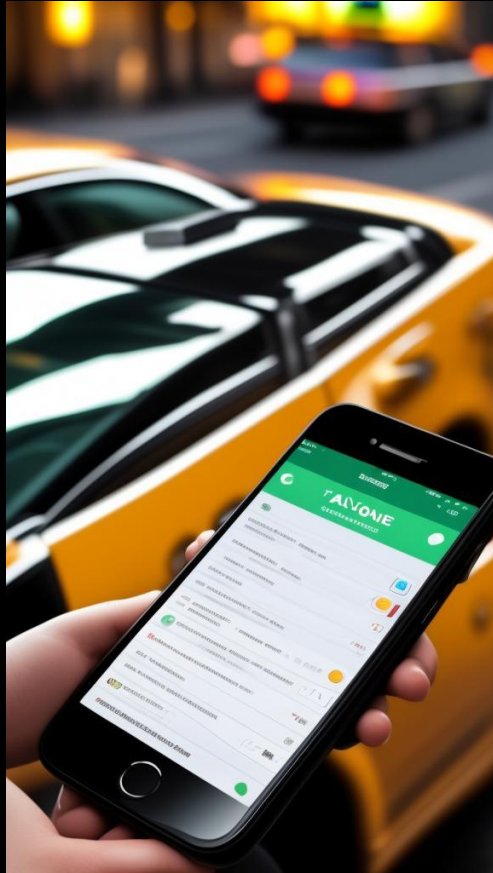
Diamond



Black



Express



El sistema de taxis deberá de cumplir con lo siguiente:

Para que el usuario pueda hacer una reservación en el sistema, deberá de estar dado de alta

- El usuario proporcionará su nombre y los datos de su tarjeta de crédito
- El usuario podrá acceder a promociones y descuento únicos si ha hecho reservaciones al menos 5 veces en el lapso de una semana

Reservaciones:

- El cliente hará la reservación
- Origen y destino marcado por el cliente
- La fecha y hora del viaje
- Tipo de unidad solicitado

Abstracción

SISTEMA DE TAXIS

Mundo real



Lo que necesitamos en programación

Características y comportamientos

- Gestionar las reservaciones
- Gestionar la base de datos
- Registro de rutas establecidas

UNIDADES

Mundo real



Lo que necesitamos en programación

Características y comportamientos

- Las unidades necesitan ir a mantenimiento por lo tanto no están disponibles para dar servicio
- La tarifa es diferente en cada unidad

USUARIO

Mundo real



Lo que necesitamos en programación

Características y comportamientos

- Lleva un registro de los viajes para ser considerado a ofertas
- También se lleva un registro de su información como nombre e información de pago (Tarjeta de crédito)
- El usuario es el único que hace reservaciones

MENU INTERACTIVO

Mundo real



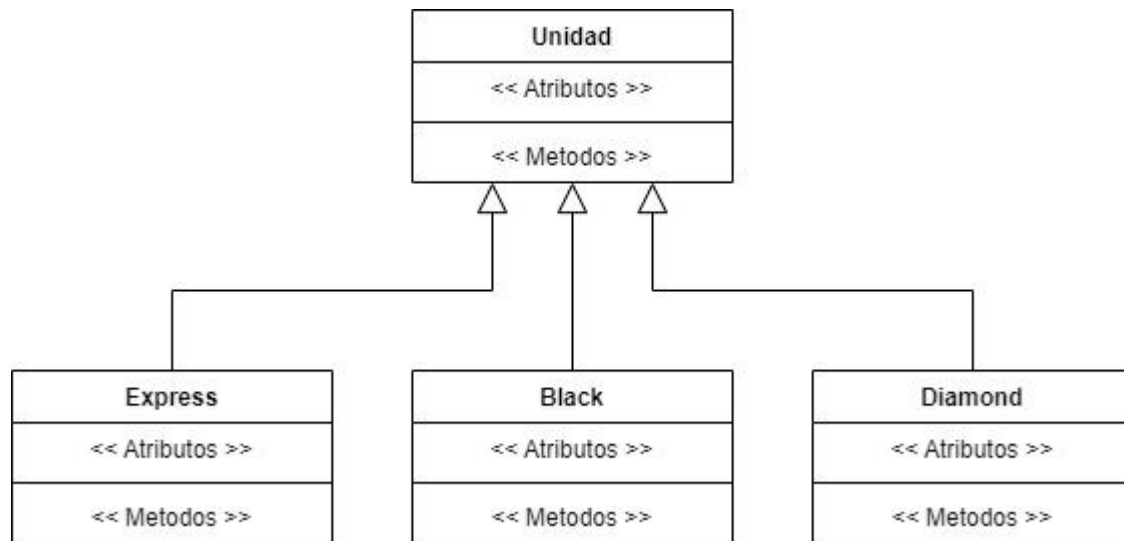
Lo que necesitamos en programación

Características y comportamientos

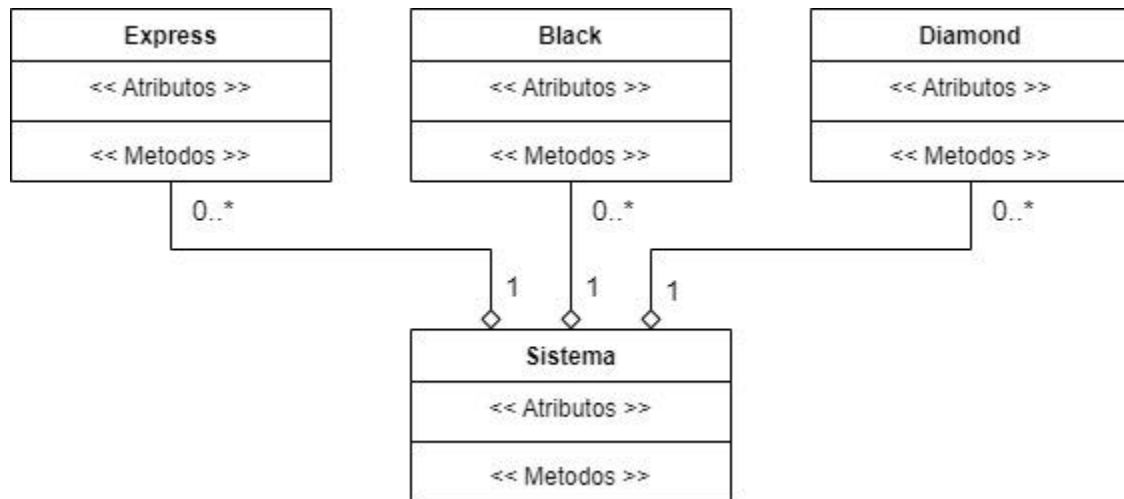
- Establecer una comunicación sencilla entre usuario-sistema
- Permitir que se accedan a las funcionalidades que brinda el sistema

Diseño

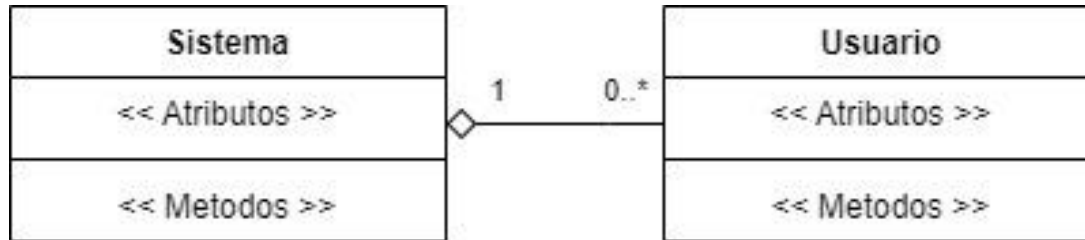
Herencia unidad-unidades



Agregacion unidades – sistema



Agregación sistema-usuario



Composición sistema-menú

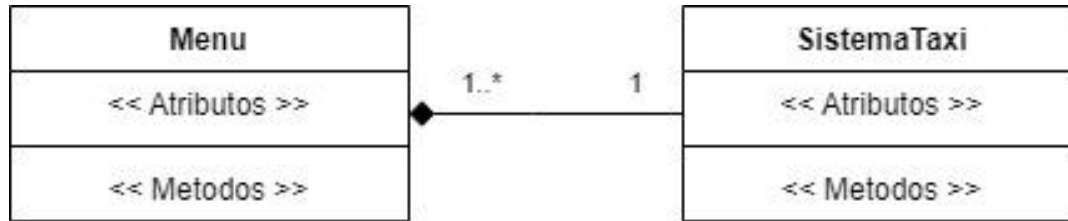
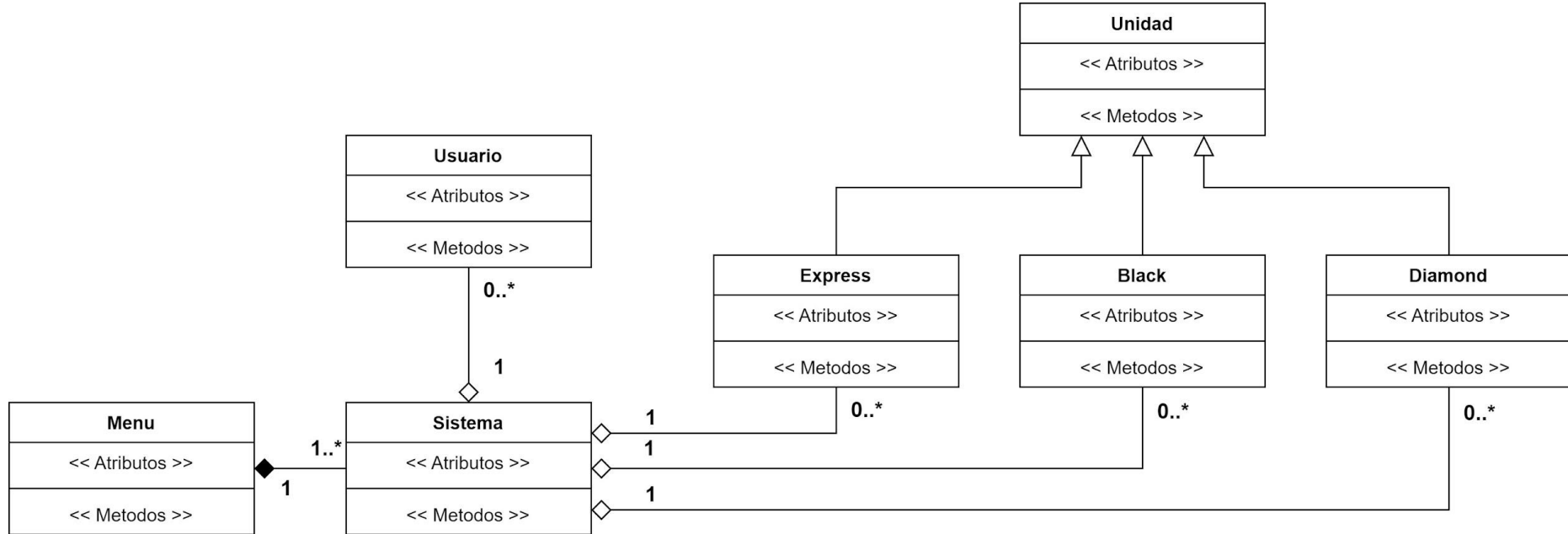
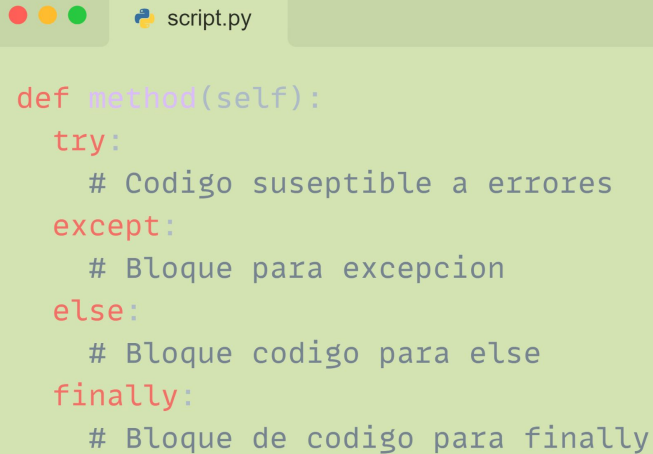


Diagrama UML



Manejo de excepciones

A screenshot of a Python script editor window. The window has a title bar with three colored circles (red, yellow, green) and a tab labeled 'script.py'. The code inside is a Python function definition for 'method(self)'. It uses a try-except-else-finally block to handle exceptions. The 'try' block contains a comment '#Codigo suseptible a errores'. The 'except' block contains a comment '#Bloque para excepcion'. The 'else' block contains a comment '#Bloque codigo para else'. The 'finally' block contains a comment '#Bloque de codigo para finally'.

```
def method(self):  
    try:  
        #Codigo suseptible a errores  
    except:  
        #Bloque para excepcion  
    else:  
        #Bloque codigo para else  
    finally:  
        #Bloque de codigo para finally
```


Tipo de excepciones manejadas

script.py

```
except IndexError:
```

script.py

```
except FileNotFoundError:
```

script.py

```
except Exception as e:
```

script.py

```
except ArithmeticError:
```

script.py

```
except KeyError:
```

Inicialización

```
def load_database_usuarios(self):  
    try:  
        with open('database_usuarios', "rb") as file:  
            self.__usuarios = load(file)  
    except FileNotFoundError:  
        self.__usuarios = {}  
        print('Base de datos de Usuarios no encontrada, se ha inicializado con valores vacíos.')  
    else:  
        print('Base de datos de usuarios cargado con éxito!')
```

Procesamiento

```
script.py

def delete_reservacion(self, reservacion):
    try:
        self.__reservaciones.pop(int(reservacion))
    except IndexError:
        print('No hay reservaciones por eliminar o ingreso un número fuera de rango')
```

```
script.py

def sacar_unidad_taller(self, no_eco):
    try:
        unidad = self.__unidades[no_eco]
        unidad.mantenimiento_finalizado()
    except KeyError:
        pass
```