

Desafío 2

Informe

Informática II (2025-1) – Universidad de Antioquia – Jose Valverde

* **Análisis del problema**

Este desafío, palabras más palabras menos, fue para diseñar un mercado de estadías diseñado para personas foráneas a la ciudad de Medellín, pero, mas allá de eso en términos de la programación, el desafío fue para poner en práctica los conocimientos sobre las clases, que partes está compuesta (constructor, destructor, getters, setters métodos) y como interactúan entre ellas

* **Soluciones que se consideraron**

El proyecto consta de las siguientes clases gira en torno a una clase llamada sistema que es el centro de la funcionalidad del programa, en esa clase sistema se reconocerá la identificación del usuario al tomar sus datos de autenticación que el usuario ingresa (numIdentidad y password/contraseña) y que rol tiene en el flujo de este programa, al reconocer ese rol entran en juego las clases “anfitrión” y “huésped” que el usuario va a acceder a esta dependiendo del rol que el sistema le haya reconocido (según la base de datos guardada), después de esto el usuario anfitrión puede revisar sus habitaciones y la reserva en cada una de las habitaciones, mientras que el huésped tendría como funciones crear la reserva, escoger el método de pago y al verificar el pago sobreescribirla en la base de datos, y cancelar esta misma.

Todo esto se explicará en detalle a través de la descripción de alto nivel.

* **Descripción del programa**

Para el desarrollo del desafío se utilizó este diagrama UML

Texto

El contenido generado por IA puede ser incorrecto.

Debido a que no todas las clases fueron desarrolladas en su totalidad (por lo que el programa no fue terminado) solo se explicará el flujo con el que quedó el código del desafío

**main.cpp**

El archivo principal (main.cpp) tiene el papel el flujo general del programa. Presenta un menú inicial donde el usuario puede iniciar sesión o salir.

**Inicio de sesión:**

* El usuario ingresa sus datos (el número de identidad y la contraseña) y se verifica si coincide en la base de datos de los usuarios (usuarios.txt).
* Según el tipo de usuario identificado (huésped o anfitrión), el sistema muestra el menú que le corresponde al rol el cual está registrado el usuario.
* Los menús permiten a los huéspedes crear/cancelar reservas y consultar habitaciones, mientras que los anfitriones pueden gestionar sus habitaciones y reservas asociadas.

**Lógica de interacción:**

* + Tras autenticación exitosa, se delega la lógica específica de cada rol a métodos de la clase Sistema y a los objetos correspondientes.
  + Se emplean menús y submenús para estructurar la interacción.

**sistema.h / sistema.cpp**

La clase “Sistema” centraliza la gestión de usuarios y la lógica de autenticación para acceder al programa.

**Validación y carga de usuarios:**

* Lee los datos de usuarios desde una base de datos .txt, valida los datos ingresados y luego determina el tipo de usuario (huésped o anfitrión).
* Carga los datos específicos del usuario autenticado, creando instancias de “Huesped” o “Anfitrion” según corresponda.

**Gestión de sesión:**

* Mantiene información del usuario actual y expone métodos para mostrar menús y delegar operaciones a los objetos de usuario.
* Provee métodos para mostrar menús diferenciados según el rol.

**anfitrion.h / anfitrion.cpp**

El objeto “Anfitrión” representa a los usuarios que pueden gestionar habitaciones y reservas.

**Gestión de habitaciones propias:**

* Permite consultar todas las habitaciones del anfitrión, accediendo a la base de datos de habitaciones y filtrando por su identificador.
* Por cada habitación, ofrece un submenú para ver, agregar o cancelar reservas por esa habitación, interactuando con el módulo de habitaciones y reservas.

**Encapsulamiento:**

* Almacena información relevante del anfitrión (a través de sus getters) y referencia a sus habitaciones y reservas.

**habitaciones.h / habitaciones.cpp**

La clase “habitación” modela cada una de las habitaciones ofertadas en el sistema.

**Carga y filtrado:**

* El sistema puede cargar todas las habitaciones desde un archivo y filtrarlas por el número de identificación del anfitrión, de esta forma se consiguen también los datos del anfitrión que están guardados en la base de datos.
* Permite mostrar información detallada de cada habitación, incluyendo amenidades y precios.

**Gestión de reservas:**

* Implementa métodos para mostrar, agregar y cancelar reservas asociadas a la habitación.
* La relación con reservas es dinámica y se apoya en la persistencia de datos en archivos.

**reservas.h / reservas.cpp**

El objeto “Reserva” representa una reserva realizada por un huésped para una habitación.

**Atributos y persistencia:**

* Almacena los datos esenciales de la reserva (identificadores, fechas, estado, monto).
* Provee métodos para cargar, mostrar, cancelar y actualizar reservas, con persistencia en archivos de texto.

**Integración:**

* Es utilizada tanto por huéspedes como anfitriones para consultar y gestionar el estado de las reservas.