
Facens

Algoritmos e Programação em Python

Aplicativo para Reservas de Quartos

Gabriel Nunes de Moraes Ghirardelli
João Henrique Crivelli Rodrigues de Souza
Leonardo Costa Teixeira

Proposta

Aplicativo para Reservas de Quartos

- Possibilitar ao usuário o cadastro completo de hotéis
- Possibilitar ao usuário a criação e cancelamento de reservas nos hotéis cadastrados
- Realizar leitura e escrita de dados em sistema de arquivo, para persistência de informações.
- Considerar problemáticas envolvendo a edição e exclusão de hotéis e manipulação de reservas
- Apresentar sistema para visualização de quartos reservados e disponíveis de cada hotel cadastrado através de gráficos.

Estruturação

O sistema é composto majoritariamente por 4 classes, as quais são:

01

Storage

02

Hotel

03

Hotels

04

System

Storage

Gerencia a leitura e escrita de dados em formato JSON.

Hotel

Representa um hotel, com atributos coerentes e métodos para manipulação de informações.

Hotels

Realiza o gerenciamento de uma lista de objetos da classe Hotel, além de aplicar métodos de armazenamento de informações em JSON.

System

Responsável por todas as interações com o usuário e coordenação de funcionalidades do sistema através de menus e sistema de seleção.

Storage

Métodos

- **__init__(self, path):**
  - Construtor que inicializa a classe com o caminho do arquivo onde os dados serão manipulados.
- **read_from_json(self):**
  - Lê os dados do arquivo JSON especificado e retorna um dicionário com esses dados.
- **save_to_json(self, data_dict):**
  - Recebe um dicionário como entrada e escreve esses dados no arquivo JSON especificado.

Parâmetros

- **path:**
  - Representa o caminho do arquivo onde os dados serão manipulados.

Hotel

Métodos

- **rooms_list, rooms_list_to_str, rooms_per_status, show_infos:**
  - Manipulações internas e exibição de dados.

- **reservate_room, change_num_rooms, update_hotel, make_reservation, cancel_reservation:**
  - Alteração de dados de hotéis e reservas.

Parâmetros

- **cnpj, name, address, city, state, country, num_rooms:**
  - Armazenam as informações base do hotel.

- **reservations:**
  - Utilizado para realizar o controle de quartos disponíveis.

Hotels

Métodos

- **load_hotels, save_hotels, rooms_per_status, show_infos:**
  - Interface com a classe Storage, para persistência de dados.

- **add_hotel_to_hotels, update_hotel_in_hotels, delete_hotel_from_hotels, create_room_reservation, delete_room_reservation:**
  - Criação, leitura, alteração e escrita de dados de hotéis e reservas.

- **plot_rooms_per_hotel, plot_occupation_per_hotel:**
  - Exibição de dados e estatísticas de quartos e reservas dos hotéis.

Parâmetros

- **path:**
  - Representa o caminho do arquivo onde os dados serão lidos e escritos.

- **hotels:**
  - Lista de objetos da classe Hotel.

System

Métodos

- **start, show_menu, show_sub_menu, display_title, display_menu, check_command, func_loop, run_command, wait:**
  - Exibição de menus, validação de inputs, manutenção de loops e controle de fluxo do sistema.

- **input_hotel_info, print_hotel_infos, select_hotel, select_room, create_hotel, list_hotels, update_hotel, delete_hotel, create_reservation, delete_reservation, show_reservation_infos:**
  - Execução e controle das funcionalidades e sistemas de exibição e leitura de dados.

Parâmetros

- **options_menu, options_menu_hotel, options_menu_reservation, options_return, options_confirmation,:**
  - Parâmetros fixos utilizados para exibição de menus e controle de opções.

- **hotels_list:**
  - Instância da classe Hotels.

Conceitos aplicados

Conceitos aplicados

Alguns dos principais conceitos aplicados durante a construção do projeto foram:

Encapsulamento

- Ao manter as funções dentro das classes necessárias, todas implementações foram condensadas em unidades coesas. Isso ajuda a controlar o acesso aos dados e funcionalidades, evitando alterações não autorizadas.

Modularidade

- A divisão do sistema em classes separadas promove a modularidade, o que significa que cada classe pode ser manipulada de forma independente, facilitando a testagem de cada componente, a reutilização do código e a colaboração entre diferentes membros da equipe.

Agradecemos a atenção!