Desarrollo de Aplicación

Contenido

[Capa de Datos 2](#_Toc22913663)

[Capa de procesamiento 2](#_Toc22913664)

# Capa de Datos

Se requiere desarrollar una aplicación que actúe como directorio telefónico.

Los datos involucrados son los siguientes:

|  |  |  |
| --- | --- | --- |
| Dato | Contenido | Especificación |
| Número de teléfono | Es el número de teléfono del contacto. | **telefono**  No se puede omitir.  Solo dígitos.  Longitud igual a 10. |
| Correo electrónico | Es el correo electrónico del contacto. | **correo**  No se puede omitir.  Debe tener un formato de correo válido.  Debe estar completamente en minúsculas.  Longitud máxima de 45 caracteres. |
| Nombre completo | Es el nombre completo de la persona. | **nombre**  No se puede omitir.  Solo permite letras, espacios y puntos.  Al menos tiene un espacio intermedio, entre palabras.  Longitud mínima 10 caracteres, y longitud máxima 40 caracteres. |
| Fecha de registro | Fecha y hora en que se registró el contacto. Se asigna automáticamente por el programa. | **fecharegistro**  No se puede omitir.  Es un dato fecha/hora. |

En el caso de Python:

1. Los datos se almacenan en archivos de texto plano (**PyContactos.txt**), en formato JSON, que componen el repositorio de datos.
2. Al cargar la aplicación, la totalidad de los datos se des-serializan y de almacenan en una lista, y ahí se manejan.
3. Los datos se manejan como objetos, no como datos.
4. Al cerrar la aplicación, se actualiza el repositorio físicamente. Los datos deben serializarse.

# Capa de procesamiento

1. Se debe declarar una clase llamada Contacto.
2. La clase tendrá una propiedad por cada uno de los datos.
3. Se agregará una propiedad adicional que se llame **ContactoEsValido**. Valdrá **True**, si los datos son completos, correctos y válidos, y **False**, si no. Esta propiedad se actualiza al aplicar **ValidarContacto()**.
4. La clase tendrá los siguientes métodos, siguiendo la metodología REST:
   1. **GuardarContacto()**. Agrega un contacto al repositorio.
   2. **ModificarContacto()**. Modifica los datos de un contacto ya registrado previamente, y del cual se proporciona el número de teléfono.
   3. **EliminarContacto()**. Elimina un contacto previamente registrado.
   4. **LeerContacto()**. Recupera la información de un contacto en específico, a partir del número de teléfono.
   5. **LeerContactos()**. Recupera la información de todos los contactos en el repositorio.
   6. **ValidarContacto()**. Verifica que los datos contenidos en las propiedades sean completos, correctos y válidos.

Capa de presentación

La capa de presentación se encarga de asegurar que los datos requeridos para almacenamiento y procesamiento están completos, correctos y válidos.

Se espera que la interfaz de la oportunidad de manejar las siguientes acciones:

1. Ver todos los contactos.
2. Buscar un contacto existente.
3. Agregar un contacto.
4. Modificar un contacto existente.
5. Eliminar un contacto existente.
6. Actualizar repositorio.
7. Salir de la aplicación.

Se debe tener una interfaz que solo permita datos válidos.

import datetime

# Declaración de clase Contacto

class Contacto:

# Defino un método constructor, que permite crear una clase usando argumentos.

def \_\_init\_\_(self, telefono, correo, nombre, fecharegistro=datetime.datetime.now()):

self.telefono = telefono

self.correo = correo

self.nombre = nombre

self.fecharegistro = fecharegistro

# Declarando una lista vacía

misContactos=[]

# Almaceno dos objetos en una lista

misContactos.append(Contacto("8119999999","Felipe","felipe@gmail.com"))

misContactos.append(Contacto("8119999998","Martha","martha@gmail.com"))

# Lectura de los objetos.

for contactos in misContactos:

print(contactos.nombre)