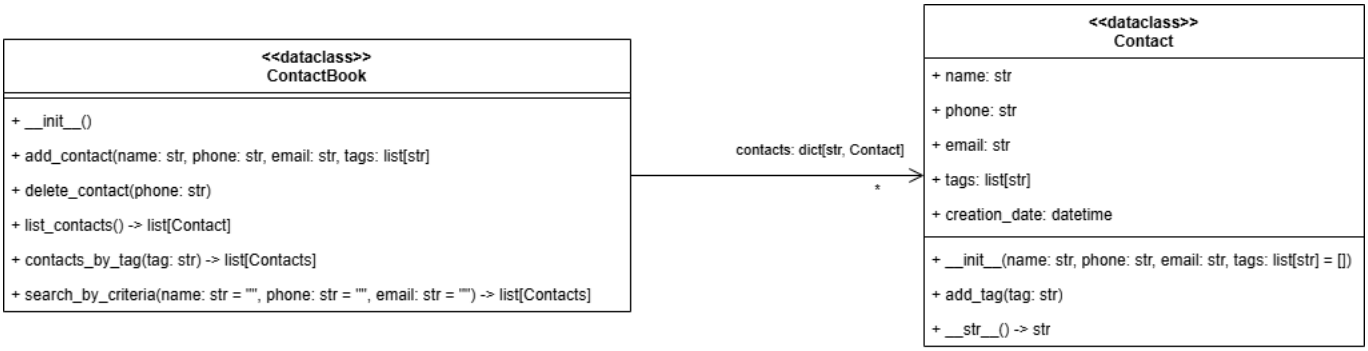
Contact Book

Esta es una aplicación de práctica para conceptos de POO en Python. La aplicación es una libreta de contactos simple con las siguientes características:

- **Agregar un nuevo contacto:** Permite a los usuarios agregar un nuevo contacto especificando el nombre, número de teléfono, correo electrónico y etiquetas (por ejemplo, amigo, compañero de trabajo, etc.). El sistema asigna un número de teléfono único a cada contacto.
- **Listar todos los contactos:** Muestra todos los contactos almacenados actualmente en la libreta de contactos.
- **Buscar por múltiples criterios:** Proporciona la funcionalidad para buscar contactos utilizando combinaciones de nombre, número de teléfono y/o dirección de correo electrónico.
- **Eliminar un contacto:** Los usuarios pueden eliminar un contacto de la libreta de contactos proporcionando el número de teléfono del contacto.
- **Buscar por etiqueta:** Filtra contactos basados en una etiqueta específica asignada a ellos (por ejemplo, trabajo, amigo, etc.).

El modelo de dominio para la aplicación consta de las siguientes clases:



Tu tarea es implementar el modelo en el archivo `contactbook/model/contacts.py`. Considera las siguientes pautas:

1. Clase `Contact`

- La clase debe ser implementada como un `dataclass`.
- La clase debe tener los siguientes atributos:
  - `name` de tipo `str` inicializado con un parámetro en el constructor.
  - `phone` de tipo `str` inicializado con un parámetro en el constructor.
  - `email` de tipo `str` inicializado con un parámetro en el constructor.
  - `tags` de tipo `list[str]` inicializado con un parámetro en el constructor, pero con una lista vacía como valor predeterminado.
  - `creation_date` de tipo `datetime`. Este atributo **no** se inicializa con un parámetro en el constructor, pero debe establecerse en la fecha y hora actuales cuando se crea el objeto.

    > **Consejo:** Puedes usar la función `field` del módulo `dataclass` y la función `datetime.now()` del módulo `datetime`.

- La clase debe tener un método de instancia `add_tag` que reciba un parámetro `tag` de tipo `str` y lo agregue al atributo `tags`. El método no debe agregar la etiqueta si ya existe en la lista.
- La clase debe tener un método de instancia `__str__` que devuelva un `str` con el siguiente formato:

```
Name: {name}
Phone: {phone}
Email: {email}
Tags: {tags}
Created on: {creation_date}
```

Donde `{name}` debe ser reemplazado por el valor del atributo `name`, `{phone}` por el valor del atributo `phone`, `{email}` por el valor del atributo `email`, `{tags}` por el valor del atributo `tags` separado por comas, y `{creation_date}` por el valor del atributo `creation_date`.

2. Clase `ContactBook`

- La clase debe ser implementada como un `dataclass`.
- La clase debe tener un atributo de instancia `contacts` de tipo `dict[str, Contact]` que no se inicializa con un parámetro en el constructor, sino con un diccionario vacío como valor predeterminado.
- La clase debe tener un método `add_contact` que reciba los siguientes parámetros:
  - `name` de tipo `str`.
  - `phone` de tipo `str`.
  - `email` de tipo `str`.
  - `tags` de tipo `list[str]`.

  El método debe crear un nuevo objeto `Contact` con los parámetros proporcionados y agregarlo al diccionario `contacts` usando el `phone` como clave.
- La clase debe tener un método `delete_contact` que reciba un parámetro `phone` de tipo `str` y elimine el contacto con el número de teléfono proporcionado del diccionario `contacts`.
- La clase debe tener un método `list_contacts` que devuelva una `list[Contact]` con todos los contactos en el diccionario `contacts`.
- La clase debe tener un método `contacts_by_tag` que reciba un parámetro `tag` de tipo `str` y devuelva una `list[Contact]` con todos los contactos que tengan la etiqueta proporcionada en su atributo `tags`.
- La clase debe tener un método `search_by_criteria` que reciba los siguientes parámetros:
  - `name` de tipo `str`.
  - `phone` de tipo `str`.
  - `email` de tipo `str`.

  El método debe devolver una `list[Contact]` con todos los contactos que coincidan con los criterios proporcionados. Si un parámetro es una cadena vacía, no debe ser considerado en la búsqueda. Por

ejemplo, si el parámetro `name` es una cadena vacía, el método debe devolver todos los contactos que coincidan con los criterios de `phone` y `email`, pero no con el criterio de `name`. La búsqueda debe ignorar el caso de las cadenas al compararlas y debe considerar una coincidencia si el criterio de búsqueda es una subcadena del atributo del contacto.