Ejercicios GraphQL

1. Dado el siguiente schema:

```
type Query {
  movies: [Movie]                # All movies
  movie(unit: String!): Movie    # Get a specific movie
  actors: [Actor]                # All actors
}

type Movie {
  name: String!                  # Name, e.g. “Back to the Future”
  releaseDate: String            # Release Date, e.g. “36-12-1985”
  director: Director
  actors: [Actor]

}

type Director {
  fullName: String!              # Director’s name, e.g. “R.Zemeckis”
  age: Int                       # Director’s age, e.g., 67
}

type Actor {
  fullName: String!              # Actor’s name, e.g. “Michael Fox”
  age: Int                       # Actor’s age, e.g. 58
  appearsIn: [Movie]             # List of films where he/she appears
}
```

escribir queries para consultar la siguiente información:
    a. Obtener nombre y edad de todos los actores.
    b. Para la película “Titanic”, obtener nombre y fecha de estreno.
    c. Para todas las películas, consultar su nombre, nombre del director, y nombre y edad de cada actor/actriz que participa en ella.
2. Modelar un schema para construir un API dados los siguientes requerimientos funcionales: Se quiere implementar una app para envío de correos electrónicos. A tener en cuenta:
    a. La app tendrá usuarios. Cada Usuario puede contar con múltiples Cuentas de email.
    b. Cada Cuenta tiene una dirección, un Inbox y borradores.
    c. Cada correo tiene un usuario remitente, un asunto y un cuerpo/body del mensaje.
    d. Un Usuario puede enviar un correo si cuenta con la dirección del destinatario. (recuerde que aquí se está **enviando** información al servidor).

    Puede probar escribiendo algunas queries sobre el esquema definido.
3. Vemos que GraphQL no tiene un wildcard (*) para consultar todos los fields de un objeto, evitando nombrarlos uno por uno (como por ejemplo, el * de SQL). Por qué cree que ocurre esto?
4. Piense en posibles desventajas que tiene GraphQL.