**Desarrollo de la aplicación:**

**Estilo arquitectural:** Basado en Microservicios **Frontend:** React, node.js

**Cada microservicio del backend:** Python, FastAPI framework

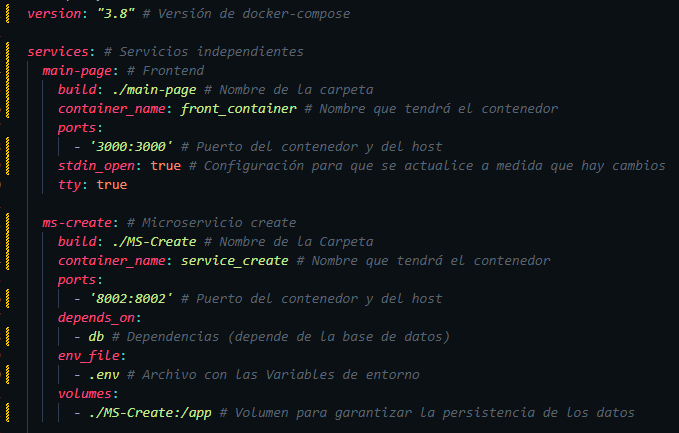
**Base de datos:** MySQL

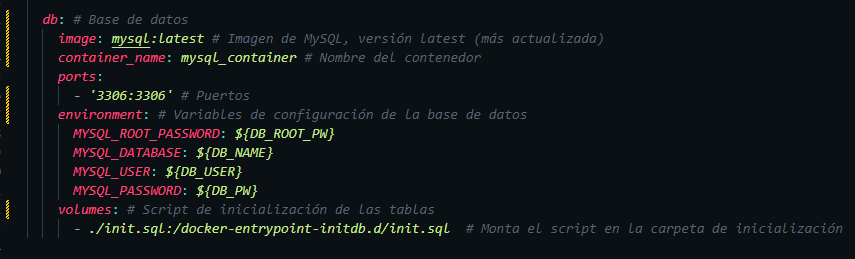
**Contenerización:** Docker & docker-compose

1. **Ir a Crear**
   1. Mencionar que se cumplieron las validaciones requeridas
   2. Mencionar que gracias a la comunicación del grupo, se hicieron validaciones extra en la mayoría de campos. Mencionar las siguientes:
      1. Un menor de edad no puede tener cédula: Cuando estén todos los campos llenos, digitar Cédula de ciudadanía en el Tipo y Fecha: 01/12/2005. Dará error porque no tiene los 18 años al enviar. Si pone 30/11/2005, ya tiene los 18 años y da bien.
      2. Tipo de archivo: No solamente se valida que no supere los 2 mb el archivo, si no que sea un formato de imagen válido: JPEG, PNG. Probar asignando un PDF al archivo (de menos de 2 mb). Saltará el error al enviar.
2. **Ir a Consultar**
   1. Mencionar que se validó que cuando la persona no existe envíe un mensaje (poner cualquier cédula que no exista)
   2. Mencionar que todos los inputs de este tipo están validados también, contra letras, menos de 10 digitos, etc.
   3. Digitar el documento creado anteriormente y buscar.
   4. Mencionar que se muestran todos los campos y la foto, y estos no son modificables.
3. **Ir a modificar**
   1. Ingresar el documento creado anteriormente
   2. Mencionar que si no se hace ningún cambio el sistema lo detecta. Enviar sin cambios.
   3. Mencionar que todas las validaciones anteriores son replicadas en el formulario del update. El numero de documento no se puede modificar.
   4. Cambiar la foto y el género y enviar
4. **Ir a eliminar**
   1. Igualmente, si la persona no existe, lo muestra.
   2. Ingresar el documento creado anteriormente y eliminar.
5. **Ir a registro**
   1. Mencionar que se puede filtrar por tipo o número de documento o fecha de registro (o combinar todas).
   2. Mencionar que muestra la información básica y cada ID tiene una lupa para ver la información detallada. Entrar a alguna lupa.
   3. Mostrar un Actualizar y un Buscar. Mostrar que en el Actualizar hay un campo que muestra los cambios realizados, mientras que en el Buscar no ya que no debería estar ahí.
6. **Abrir docker** 
   1. En **containers,** entrar al del proyecto (**finalproject**). Mostrar todos los contenedores corriendo independientemente. Para bajar algún contenedor, le dan al cuadrito (stop), y esperan a que baje. **NO ELIMINEN NADA**
   2. Mencionar que para proveer una capa más de seguridad, las credenciales y claves se manejan mediante variables de entorno. El contenedor no tiene especificadas ni contiene las claves directamente en la configuración. Esto da un plus de seguridad ya que enmascara cualquier tipo de credencial para la conexión.

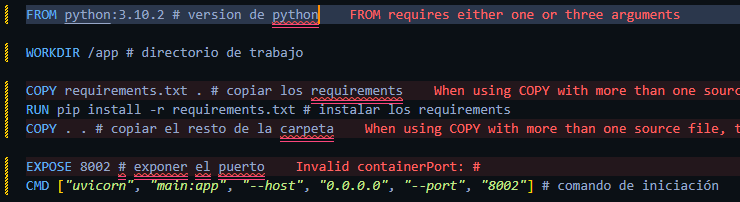
**FIN**

**Si les pide mostrar el docker-compose,**

****

****

**El dockerfile (está en cada carpeta),**

****

**Patrones de diseño MÁS usados:**

1. **CQRS: Separar las responsabilidades de consulta y escritura. Para esto son los microservicios.**
2. **Repositorio: Aislar la capa de datos del resto de la aplicación. Usado distribuyendo en un contenedor independiente la base de datos, gestionando el acceso a la misma, logrando así la persistencia de los datos y su independencia al resto de la aplicación.**

**Estilo arquitectural:**

**MICROSERVICIOS**

**Gestión de cambios:**

1. **La más importante: claridad de requisitos.** Primero que nada, nos enfocamos, a través de varias reuniones presenciales y virtuales, a conseguir la mejor claridad en los requisitos.
2. **Comunicación Interactiva:** Hablar sobre la importancia de la comunicación
3. **JIRA**

**Gestión de riesgos:**

1. **Falta de experiencia en contenerización**
2. **Problemas aislados en los microservicios**
3. **Limitaciones de Hardware**

**Metodología:**

**Agile, con Kanban**