Cloud Computing
MiConservatorio

Carlos Santiago Sánchez Muñoz

2 de febrero de 2021

Carlossamu7/CC1-Conservatorio

Descripción del problema

- Una escuela de música o conservatorio privado tiene todo su registro de alumnos, profesores, horarios y otros en papel.

- Quiere actualizarse para realizar dichas tareas computacionalmente.
  - Ahorro de trabajo.
  - Se evita que la pérdida de un papel suponga un fallo grave.

¡Vamos a dar una solución *cloud* al problema!

Historias de usuario - Administrador

[HU1] Como administrador quiero dar de alta una asignatura.

[HU2] Como administrador quiero modificar una asignatura.

[HU3] Como administrador quiero borrar una asignatura.

[HU11] Como administrador quiero saber en el número de alumnos y asignaturas del conservatorio.

[HU12] Como administrador quiero saber las asignaturas que imparte un profesor.

[HU13] Como administrador quiero saber el horario completo de un profesor.

[HU14] Como administrador quiero saber las aulas que usa un profesor/alumno.

[HU15] Como administrador quiero dar de alta un alumno

[HU16] Como administrador quiero obtener un listado de los alumnos y su información.

[HU17] Como administrador quiero obtener un listado de las asignaturas y su información.

Historias de usuario - Alumno

[HU4] Como alumno quiero matricularme de ciertas asignaturas.

[HU5] Como alumno quiero desmatricularme de ciertas asignaturas.

[HU6] Como alumno quiero consultar mis asignaturas matriculadas.

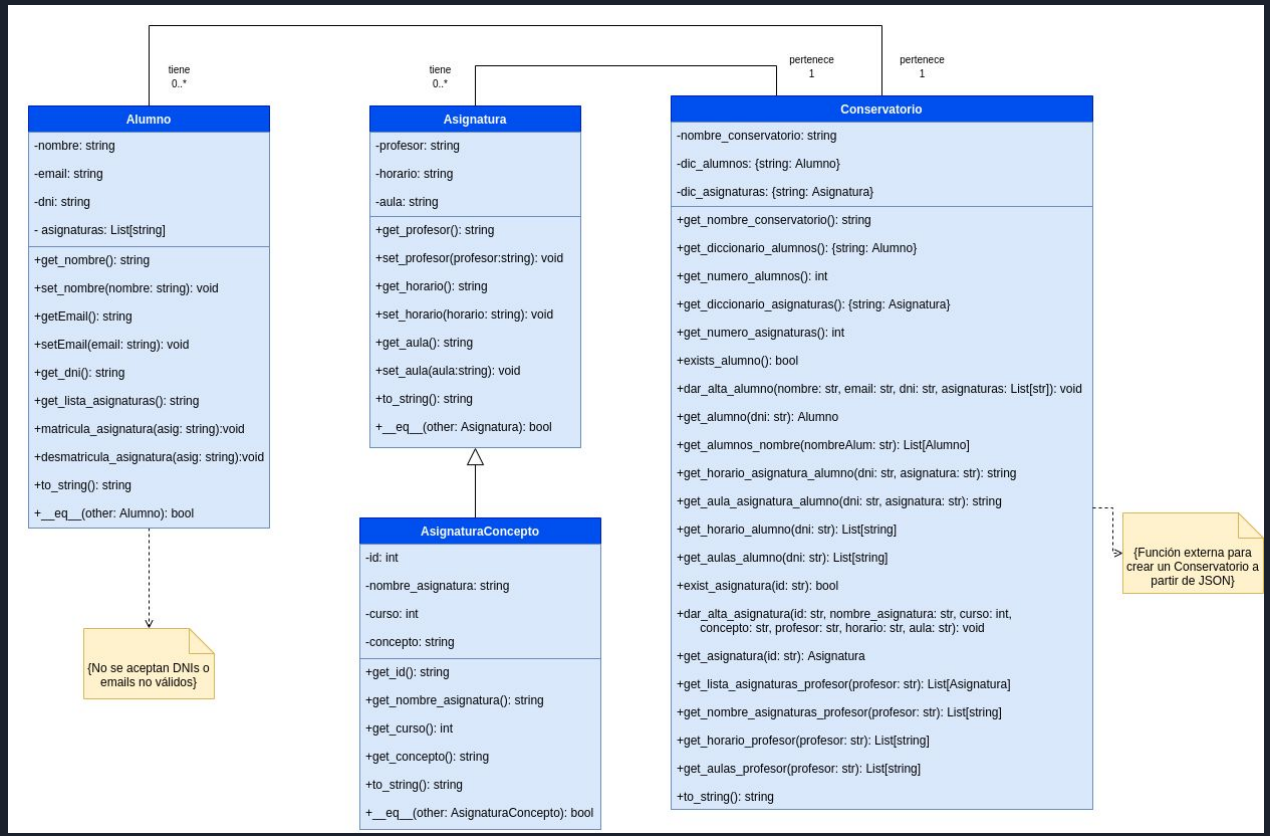
[HU7] Como alumno quiero modificar la dirección de correo con la que el centro se pone en contacto conmigo.

[HU8] Como alumno quiero consultar el horario de una asignatura.

[HU9] Como alumno quiero consultar el aula de una asignatura.

[HU10] Como alumno quiero saber mi horario completo.

Clases y estructura del proyecto



Arquitectura

- Gestiones registradas de forma automática y sin esperas ➡ ~~Dirigida por eventos~~
- Gestión independiente de diferentes tareas ➡ ~~Monolítica~~
- Añadir nuevas funcionalidades como una biblioteca ➡ ~~Microkernel~~
- Modularidad, escalabilidad, facilidad de integración de nuevos elementos

  ➡ **Arquitectura de microservicios** ✅

Test

Robot: poca flexibilidad y sin bucles anidados

Ampliamente aceptada aunque no viene integrado

Nose2: Excesivas dependencias y *plugins*

unittest

Intuitivo como framework y al combinarlo con *asserts* es cómodo.

Contenerización

- Imagen base

```
carlos@carlos-Aspire-VN7-571G:~/Documentos/CC1-Conservatorio/docs/Dockerfiles/Ubuntu20$ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
ubuntu20            latest              b80fe059178a        34 seconds ago      401MB
ubuntu18            latest              545f2586a486        17 minutes ago      500MB
fedora_python       latest              66b4e403706c        43 minutes ago      389MB
python3.9-slim      latest              881d272dc076        About an hour ago   115MB
python3.8-slim      latest              8452547a8670        About an hour ago   113MB
alpine_python       latest              4f0ced2f7337        2 hours ago         96.6MB
```

- Instalar *build-essentials*

- Instalar las dependencia de *requirements.txt*

- Ejecutar la aplicación como un usuario no privilegiado

carlossamu7/cc1-conservatorio

Integración continua

Usa docker y es similar a travis

Compilador se cuelga y tuvo inactividad

Interfaz simple, atención al cliente de calidad, dockers...

Automatización excesiva

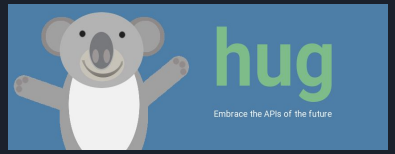
Gratuita + integraciones ilimitadas

Interfaz pobre y poca documentación

Popular, lleva tiempo, fácil configuración, conectividad GitHub

Solo ofrece soporte para proyectos de Github

Microservicios

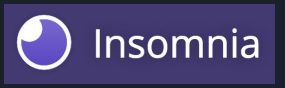


APIs fáciles de entender y mantener. Buenas prácticas

Nacido de una fusión y su comunidad está creciendo. Minimalista, rápido y fiable

Framework MVT para desarrollar proyectos robustos rápidamente

Sencillo, flexible, ligero, fácil de aprender, buenas prácticas, código elegante...



Composición de servicios

- Servidor de la API
- Contenedor de datos
  ➡Trabajo Futuro
- Cliente desde dentro de la composición.
- Test con *GitHub Actions*: wget de las rutas más relevantes

docker-compose.yml

```
version: '3'
services:
  server:
    build:
      context: .
      dockerfile: execute.Dockerfile
    restart: always
    ports:
    - "80:80"

  client:
    build:
      context: .
      dockerfile: client.Dockerfile
    depends_on:
      - server
    ports:
    - "8001:8001"
```

Gracias por su atención

¿PREGUNTAS?