Quacker - Hitos 1 y 2

Análisis de diseño (Hito 1)

Considerando la especificación de la aplicación a desarrollar en el documento "**Quacker.pdf**", realiza un análisis en el que reflejes:

- Qué clases deben existir en el modelo
- Qué relaciones existen entre esas clases y sus cardinalidades

Completa los siguientes ejercicios:

1. Redacta un documento con un epígrafe por cada clase identificada. En cada epígrafe:
    a. Lista todas las relaciones identificadas, describiéndolas verbalmente y justificando su necesidad
    b. Realiza una descripción de las tablas necesarias para reflejar este diseño en una base de datos relacional en el que incluyas:
        i. nombre de las tablas
        ii. nombre y tipo de dato de las columnas
        iii. restricciones importantes: restricciones de integridad (PK y FK), no nulo, longitud mínima, longitud máxima, valor mínimo, valor máximo... Deberás justificar las restricciones o la ausencia de ellas
        iv. reglas de integridad referencial (propagación, borrado, restricción). Deberás justificar las reglas planteadas
2. Crea un diagrama de clases UML que represente tu decisión de diseño haciendo uso de Draw.io.

Deberás producir: un fichero **.pdf** y un fichero **.drawio**

CRUD completo (Hito 2)

Completa los siguientes ejercicios:

1. Crea un proyecto de Laravel
2. Implementa el **CRUD completo** (incluyendo las vistas) de todas las clases identificadas en tu análisis de diseño. Utiliza las herramientas facilitadas por Laravel para migraciones, factorías y controladores

Importante: de momento ignora las relaciones, si quieres, puedes representar dichas relaciones mediante otro tipo de dato (por ejemplo, Quack-Usuario como Quack.nickname)

Importante: aunque no vayamos a usar muchas de estas vistas, sí es necesario que implementes todas

Importante: deberás ajustarte al patrón de diseño MVC y a la convención de nombrado RESTful

Deberás producir: un repositorio de GIT con una release claramente identificada. El **README.md** contendrá instrucciones de despliegue para que probar la aplicación no me haga perder 1 hora intentando ejecutarla. **Muy importante**: asegúrate de que el repositorio tenga visibilidad pública

Normas de entrega

Deberás entregar un fichero **nombre_grupo.zip** con la siguiente jerarquía de ficheros:

```
entrega
├── análisis.pdf
├── diagrama-uml.drawio
└── repositorio.txt

1 directory, 3 files
```

Siendo "**nombre_grupo**" el nombre elegido para tu grupo, en **snake_case**. Por ejemplo, si mi grupo se llama "Pato Cuáquero", crearé un fichero "**pato_cuáquero.zip**" que contenga la jerarquía de ficheros ya descrita, en la que:

- **entrega**: directorio de ficheros

- **análisis.pdf**: fichero PDF producido en el apartado Análisis de diseño
- **diagrama-uml.drawio**: fichero DRAWIO producido en el apartado Análisis de diseño
- **repositorio.txt**: fichero de texto que contenga únicamente un enlace al repositorio creado en el apartado CRUD completo

Cualquier desviación en la entrega (nombrado, extensión, contenido) será calificada con un 0 (cero)

Recursos de utilidad:

- Tipos de relaciones en diagramas UML
- Controladores de recursos en Laravel