UNAH

Universidad Nacional Autónoma de Honduras

Facultad de Ingeniería

Carrera de Ingeniería en sistemas

Sistemas Expertos

Número de cuenta: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Fecha: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Nombre: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Ejercicios de justificación:

A continuación, se le plantean ciertos casos que requieren de su opinión experta para realizar la implementación de servicios, en caso de no está de acuerdo exponga porque sería una mala decisión y cual debería de ser el caso correcto. En caso de que este bien la idea solo responda, con un “de acuerdo”.

1. Un Ingeniero backend está proponiendo hacer la implementación de un Queue storage para almacenar todas las informaciones de facturas que se generan en el negocio.
2. Un ingeniero frontend acaba de proponer almacenar la información de contacto de los usuarios en una base de datos tiempo real como las que proporciona firebase.
3. Un ingeniero de datos acaba de proponer usar function apps como herramienta de ETL, para mover los datos de las bases de origen al DW.
4. Su jefe acaba de proponer la idea de usar MongoDB para almacenar toda la información transaccional del negocio.
5. La compañía en la que trabaja acaba de vender el producto para 10 clientes, por lo que se ocupa levantar la infraestructura 10 veces. El Arquitecto de Software recomienda ir suscripción por suscripción configurando manualmente cada uno de los recursos para cada uno de los 10 clientes.
6. Un Junior no quiere crear muchos grupos de recursos en Azure por que le preocupa el presupuesto aprobado para el desarrollo del producto.
7. Un Ingeniero propone guardar todos los invoices generados en PDF en un Onedrive público y la UI se conectará directo al onedrive cuando el cliente quiera descargar sus invoices.

Diagrama de Arquitectura:

A continuación, una empresa emergente le presenta una idea de un producto que desean vender, primero quieren entender todos los recursos necesarios. Por lo que se le solicita a usted realizar un diagrama de arquitectura.

**Descripción General**

Una plataforma dedicada a la difusión y certificación de estándares de buenas prácticas en la industria del software, creada en colaboración con la Unión Americana de Software (UAS). Esta plataforma proporcionará acceso a nuevos estándares de gobernabilidad, materiales educativos, cursos de formación y certificaciones para empresas que deseen implementar y ser reconocidas por sus buenas prácticas.

**Funcionalidades Clave**

1. **Publicación de Estándares:**

• Una sección dedicada donde la Unión Americana de Software publica los nuevos estándares de gobernabilidad aprobados.

• Documentación detallada, guías y bases sobre cada nuevo estándar.

2. **Cursos y Materiales Educativos:**

• Ofrecimiento de cursos en línea diseñados para capacitar a los profesionales en la implementación de los nuevos estándares.

• Materiales educativos complementarios, como libros electrónicos, artículos, y estudios de caso.

3. **Certificación:**

• Proceso de certificación para empresas que buscan obtener el reconocimiento como implementadores de buenas prácticas.

• Evaluación y auditoría de las prácticas de las empresas para garantizar el cumplimiento de los estándares.

• Emisión de certificados oficiales y distintivos de reconocimiento.

4. **Plataforma de Comunidad:**

• Foro de discusión y grupos de trabajo para profesionales interesados en las mejores prácticas de gobernabilidad.

• Eventos en línea y seminarios web para el intercambio de conocimientos y experiencias.

5. **Seguimiento y Actualización:**

• Módulo de seguimiento para empresas certificadas para asegurar la continuidad y actualización de las buenas prácticas.

•Actualización continua de la plataforma con los últimos estándares y mejores prácticas reconocidas a nivel global.

Dado que el scope de este proyecto debe de ser seguro, le recomendamos pensar en mantener los modulos administrativos que usara UAS lo mas seguro que se pueda.

Refactorización de código:

import psycopg2

import requests

def connect\_to\_database():

try:

connection = psycopg2.connect(

host='localhost',

port='5432',

user='admin',

password='password123',

dbname='testdb'

)

cursor = connection.cursor()

cursor.execute("SELECT version();")

db\_version = cursor.fetchone()

print(f"Connected to database: {db\_version}")

cursor.close()

connection.close()

except Exception as error:

print(f"Error connecting to database: {error}")

def make\_api\_request(endpoint):

headers = {

'Authorization': 'Bearer sk\_test\_4eC39'

}

response = requests.get(endpoint, headers=headers)

if response.status\_code == 200:

return response.json()

else:

print(f"API request failed with status code {response.status\_code}")

return None

if \_\_name\_\_ == "\_\_main\_\_":

connect\_to\_database()

data = make\_api\_request('https://api.example.com/data')

if data:

print("API response data:", data)

Según el codigo proporcionado haga una propuesta de un archivo .env, especificando sus llaves y valores para hacer mas seguro la implementación del codigo anterior. Recuerde que estos valores .env tambien consideran valores que pueden cambiar entre ambientes como DEV, STG, PROD.

Restful API

Según la siguiente pantalla haga la propuesta de que endpoint se deberian de implementar

A screenshot of a computer

Description automatically generated

La definición de cada endpoint deberia de llevar:

* URI (use localhost Como domain base)
  + Si su endpoint ocupa parametros en el URI puede especificarlos mediante {nombre\_variable}
* HTTP Method
* Headers
* Payload

Pauta:

Ejercicios de justificación:

1. Las Queue storages son utilizes para almacenar mensajes que se son procesados por workers, aparte que los queue son almacenamiento cache. La informacion de factura se pueden almacenar una BD relacional ya que es logica muy de negocio y los PDFs en storage account.
2. Las base de datos de tiempo real son utilies para almacenar el status de alguna transaccion, notificaciones, etc. No se recomienda almacenar información sencible del negocio en ellas. La información de contacto como es tabular y delicada es mejor mantenerla en una base de datos tipo SQL.
3. “de acuerdo”
4. MongoDB es buena para almacenar información no estructurada. La manera mas efciente de almacenar la logica que es transaccional del negocio y que requiere normalemente de mucha integridad mediante los constraints que proporcionan las Llaves primarias, foraneas, etc es mejor usar una base de datos tipos SQL.
5. Hacerlo manualmente podria darnos el error humano de configurar algo mal. Para este tipo de caso de usos se recomienda usar mejor un IaC como lo es terraform.
6. Los grupos de recurso en Azure no tienen un impacto en el presupuesto o gastos operativos del mes. Igual, hay que crear los RG de la manera mas conveniente para la gestión de los proyectos.
7. Invoices es información sensible que no deberia de estar en repositorios publicos como lo es onedrive según la propuesta. Recomendamos usar mejor un storage account con el servcio de los blobs.

Diagrama de Arquitectura

A screenshot of a computer

Description automatically generated

.env

HOST=localhost

port=5432

user=admin

password=password123

dbname=testdb

SERVICE\_TOKEN=sk\_test\_4eC39

API=https://api.example.com

Restfulapi

A close-up of a sign

Description automatically generated

|  |  |
| --- | --- |
| Endpoint | Localhost/user |
| HTTP Method | GET |
| Header | Bearer jwt |
| Payload |  |



|  |  |
| --- | --- |
| Endpoint | Localhost/projects |
| HTTP Method | GET |
| Header | Bearer jwt |
| Payload |  |



|  |  |
| --- | --- |
| Endpoint | Localhost/priorities |
| HTTP Method | GET |
| Header | Bearer jwt |
| Payload |  |



|  |  |
| --- | --- |
| Endpoint | Localhost/project/{project\_id}/collaborators |
| HTTP Method | GET |
| Header | Bearer jwt |
| Payload |  |

A screenshot of a computer

Description automatically generated

|  |  |
| --- | --- |
| Endpoint | Localhost/project/{project\_id}/task |
| HTTP Method | POST |
| Header | Bearer jwt |
| Payload | {  “project\_id”: {project\_id}  , “priority\_id”: ##  ,”collaborator\_id”: ##  ,”deadline”: “####-##-##”  ,”description”: “”  } |

A group of white rectangular cards with red dots

Description automatically generated

|  |  |
| --- | --- |
| Endpoint | Localhost/project/{project\_id}/task |
| HTTP Method | GET |
| Header | Bearer jwt |
| Payload |  |



|  |  |
| --- | --- |
| Endpoint | Localhost/project/{project\_id}/task/{task\_id} |
| HTTP Method | DELETE |
| Header | Bearer jwt |
| Payload |  |



|  |  |
| --- | --- |
| Endpoint | Localhost/project/{project\_id}/task/{task\_id} |
| HTTP Method | PUT |
| Header | Bearer jwt |
| Payload | {  Status: “completed”  } |
|  |  |