Implementación de un Sistema CRUD para Gestión de Películas Usando Backend y Frontend

Anthony Cahui, Jose Vilca Escuela Profesional de Ing. De Sistemas, Universidad Nacional de San Agustín de Arequipa
acahuib@unsa.edu.pe
jvilcamam@unsa.edu.pe

Resumen:

Este artículo detalla el desarrollo de un sistema CRUD (Create, Read, Update, Delete) para la gestión de una base de datos de películas en una empresa cinematográfica, utilizando Django como framework backend y Vue.js como framework frontend. Se presentan los beneficios de la integración de estas tecnologías y se discuten las metodologías y resultados obtenidos.

I. INTRODUCCIÓN

En la era digital, la gestión eficiente de datos es crucial para el éxito de cualquier organización. Las empresas cinematográficas, en particular, deben manejar grandes volúmenes de información sobre sus películas, desde detalles técnicos hasta información de distribución. Para abordar este desafío, se requiere un sistema robusto y flexible que permita la creación, lectura, actualización y eliminación (CRUD) de registros de manera eficiente.

La integración de Django para el backend y Vue.js para el frontend proporciona una arquitectura robusta y escalable para aplicaciones web modernas. Django maneja las operaciones del lado del servidor, incluyendo la lógica de negocio y la interacción con la base de datos, mientras que Vue.js se encarga de la interfaz de usuario, proporcionando una experiencia interactiva y responsiva para el usuario final.

En este artículo, se presenta la implementación de un sistema CRUD para la gestión de películas en una empresa cinematográfica, utilizando Django como backend y Vue.js como frontend. El sistema permite a los usuarios realizar operaciones básicas de gestión de datos sobre un conjunto de películas, mejorando la eficiencia y precisión del manejo de información. A través de esta implementación, se destacan las ventajas y desafíos de utilizar estas tecnologías en conjunto. Donde como adicional se mostrará la configuración sobre google cloud básica para poder ser subida a la web.

II. DJANGO

A continuación, se presentará la definición de Django y también algunas características sobre este.

A. Definición

Django, un framework de desarrollo web de alto nivel para el lenguaje de programación Python, ha ganado popularidad por su simplicidad y su capacidad para desarrollar aplicaciones web rápidas y seguras. Django sigue el principio de "No te repitas" (DRY), lo que permite a los desarrolladores construir aplicaciones complejas con menos código y esfuerzo [1]. Vue.js es conocido por su enfoque en la reactividad y la modularidad, lo que facilita la creación de componentes reutilizables y la gestión del estado de la aplicación [2].

B. Características

Entre sus características destacadas se incluyen su enfoque en la seguridad, ofreciendo protecciones integradas contra inyección SQL, ataques XSS y CSRF. Además, Django es altamente escalable, permitiendo el desarrollo de aplicaciones desde proyectos pequeños hasta sistemas empresariales complejos. Su ORM facilita la interacción con bases de datos usando objetos Python, y su sistema de plantillas permite separar la lógica de negocio de la presentación, promoviendo un diseño organizado. El sistema de administración automáticamente generado facilita la gestión de contenidos y usuarios, y su documentación extensa proporciona un valioso recurso para desarrolladores.

C. Objetivos

Django tiene como principal objetivo facilitar el desarrollo rápido de aplicaciones web, permitiendo llevar ideas del concepto a la finalización de manera eficiente. Busca promover un diseño limpio y pragmático mediante la reutilización de código y la automatización de tareas comunes . La seguridad es un objetivo fundamental, proporcionando una base segura con protecciones integradas contra amenazas comunes. Además, Django está diseñado para ser escalable y flexible, adaptándose a proyectos de diferentes tamaños y complejidades. Finalmente, Django ofrece una herramienta completa con características integradas que permiten desarrollar aplicaciones web completas sin depender de herramientas externas.

III. VUE.JS

En esta sección se presentará el concepto de vue.js ademas de sus ventajas frente a otros competidores.

A. Definición

Vue.js es un framework progresivo de JavaScript utilizado para construir interfaces de usuario. A diferencia de otros frameworks monolíticos, Vue está diseñado desde cero para ser adoptado de manera incremental. Su núcleo se enfoca en la capa de vista solamente, lo que facilita la integración con otras bibliotecas o proyectos existentes. Además, Vue es perfectamente capaz de impulsar aplicaciones de una sola página (SPA) cuando se combina con herramientas modernas y bibliotecas de apoyo[4].

B. Ventajas

Vue.js se destaca por ser ligero, lo que contribuye a tiempos de carga rápidos. Su sistema de reactividad altamente optimizado asegura que las actualizaciones en la interfaz de usuario sean eficientes, mejorando la experiencia general. La flexibilidad de Vue.js permite su uso en una amplia gama de proyectos, desde aplicaciones pequeñas hasta sistemas grandes y complejos. Además, se integra bien con herramientas modernas y tiene un ecosistema rico con herramientas como Vue CLI, Vue Router y Vuex que facilitan el desarrollo de aplicaciones completas.

IV. CRUD

En esta sección vemos las principales definiciones de CRUD, por último los objetivos.

A. Definición

CRUD es un conjunto de operaciones esenciales que permite la manipulación de datos en una base de datos. Estas operaciones son:

Create (Crear): Proceso de añadir nuevos registros a la base de datos. Esta operación implica insertar datos específicos en las tablas de la base de datos, estableciendo así un nuevo conjunto de datos almacenados.

Read (Leer): Proceso de recuperar datos existentes de la base de datos. Esta operación permite a los usuarios o aplicaciones consultar la base de datos para obtener información relevante sin modificar los datos.

Update (Actualizar): Proceso de modificar datos existentes en la base de datos. Esta operación permite la modificación de registros específicos para reflejar cambios o correcciones en los datos almacenados.

Delete (Eliminar): Proceso de eliminar datos existentes de la base de datos. Esta operación implica la eliminación de registros específicos, liberando así el espacio ocupado y manteniendo la base de datos actualizada.

B. Objetivos

El principal objetivo de CRUD es proporcionar un marco estructurado y sistemático para la gestión de datos en aplicaciones.

Garantiza que los datos almacenados sean consistentes y puedan ser mantenidos fácilmente. Al permitir las cuatro operaciones básicas, CRUD asegura que los datos puedan ser creados, leídos, actualizados y eliminados de manera coherente y controlada.

Permite la implementación de mecanismos de control de acceso, asegurando que solo los usuarios autorizados puedan realizar operaciones específicas en la base de datos. Esto es crucial para la protección de datos sensibles y la prevención de accesos no autorizados.

V. GOOGLE CLOUD

A.Definición

Google Cloud Platform (GCP) proporciona varios servicios de computación, como Google Compute Engine y Google Kubernetes Engine, que permiten a los desarrolladores implementar aplicaciones web de manera eficiente. Compute Engine ofrece máquinas virtuales escalables, mientras que Kubernetes Engine facilita la gestión de contenedores, optimizando el rendimiento y la escalabilidad de las aplicaciones.

B. Ventajas

Una de las principales ventajas de Google Cloud es su capacidad para escalar recursos según las necesidades de la aplicación. Esto permite manejar picos de tráfico sin comprometer el rendimiento.

Google Cloud proporciona un conjunto robusto de medidas de seguridad, incluyendo cifrado de datos, autenticación de usuarios y cumplimiento con normativas

VI. METODOLOGÍA

En la siguiente sección se mostrarán 4 casos relacionados con el trabajo.

A. CASO 1:
   Según Grinberg[3], este realizó un sistema CRUD para la gestión de películas utilizando Flask como framework de backend y React para el frontend. La aplicación permite a los usuarios realizar operaciones CRUD en una base de datos de películas, mejorando la experiencia del usuario mediante una interfaz interactiva y reactiva. El uso de Flask y React ofrece una solución ligera y flexible para el desarrollo rápido de aplicaciones web.

B. CASO 2:
   En el trabajo de Vidal-Silva et. al. [7], presenta la experiencia académica, tanto de profesores como estudiantes, en un curso de último año para el desarrollo de sistemas de información web mediante el uso del marco de trabajo Django del lenguaje de programación Python. Django facilita el desarrollo de aplicaciones web con el uso del patrón de desarrollo Modelo-Plantilla-Vista (MTV, en inglés). Por la simplicidad sintáctica y práctica de Python y Django, los estudiantes de una carrera del área de administración y ciencias de la información fueron capaces de desarrollar prototipos de sistemas de información. Las experiencias de los estudiantes permiten remarcar la relevancia de las primeras fases del ciclo de desarrollo de software, requerimientos y

diseño. Los proyectos que se presentan son sistemas web prototipo con opciones para crear, leer, actualizar y eliminar (CRUD, en inglés) registros de la base de datos.

VII. IMPLEMENTACIÓN DEL CRUD

Para la implementación se dividió en dos partes, el backend y el frontend, donde en el primero se realizó la definición de las variables. Esto se realizó primero con la creación del proyecto de Django. Además de haberle hecho sus migraciones.

```
linux@linux-ThinkPad-T470:~/LabFinal$ cd backend/
linux@linux-ThinkPad-T470:~/LabFinal/backend$ ls
app  backend  db.sqlite3  manage.py
```

Luego se tiene dentro de la app un archivo de nombre serializers.py que importa el módulo rest_framework que se utiliza para definir los serializers. Y movie del módulo models que es el modelo Django que representa una película.

```python
from rest_framework import serializers
from .models import Movie

class MovieSerializer(serializers.ModelSerializer):
    class Meta:
        model = Movie
        fields = '__all__'
```

Luego se tiene el modelo a usar que es models.py, iniciando las definiciones de la clase movie.

```python
from django.db import models

class Movie(models.Model):
    name = models.CharField(max_length=255)
    description = models.TextField()
    image = models.URLField()
    rating = models.IntegerField(default=0)
    genres = models.JSONField()
    in_theaters = models.BooleanField(default=False)

    def __str__(self):
        return self.name
```

Se tiene la clase de urls.py que ayuda definir la ubicación del url de las películas.

```python
from django.urls import path, include
from rest_framework.routers import DefaultRouter
from .views import MovieViewSet

router = DefaultRouter()
router.register(r'movies', MovieViewSet)

urlpatterns = [
    path('api/', include(router.urls)),
]
```

Luego para poder que nuestra carpeta de backend reconozca a la app y algunas importaciones, se le modifica el archivo settings.py.

```python
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'rest_framework',
    'app',
    'corsheaders',
]
```

Para la carpeta de frontend se creará un proyecto, y a este se le descargara con node.js usando npm, que descargara la carpeta node_modules.

```
linux@linux-ThinkPad-T470:~/LabFinal/frontend$ ls
CHECKLIST.md  package.json       public      style.css          yarn.lock
index.html    package-lock.json  README.md   tailwind.config.js
node_modules  postcss.config.js  src         vite.config.mjs
```

Luego dentro de src tenemos las funciones principales para configurar el frontend.

```
linux@linux-ThinkPad-T470:~/LabFinal/frontend$ cd src/
linux@linux-ThinkPad-T470:~/LabFinal/frontend/src$ ls
api.js  App.vue  main.js  movies.json
```

En api.js tenemos un módulo de JavaScript que utiliza la biblioteca axios para hacer peticiones HTTP a un servidor backend que expone una API para gestionar películas. Está diseñado para interactuar con un backend que ejecuta un servidor Django REST framework. Donde este contiene métodos para realizar operaciones CRUD llamando a nuestro servidor anterior usando su url.

```javascript
import axios from 'axios';

const API_URL = 'http://localhost:8000/api/movies/';

export const api = {
  getMovies() {
    return axios.get(API_URL);
  },
  createMovie(movie) {
    return axios.post(API_URL, movie);
  },
  updateMovie(id, movie) {
    return axios.put(`${API_URL}${id}/`, movie);
  },
  deleteMovie(id) {
    return axios.delete(`${API_URL}${id}/`);
  },
};
```

Luego tenemos el archivo App.vue, donde se estará utilizando una sintaxis de <script setup> siendo un componente de vue.js, donde se establece el estado inicial y las variables reactivas necesarias para un componente de Vue.js que maneja la lista de películas y un formulario para agregar o editar películas. Las variables movies y showMovieForm se usan para gestionar la lista de películas y la visibilidad del formulario,

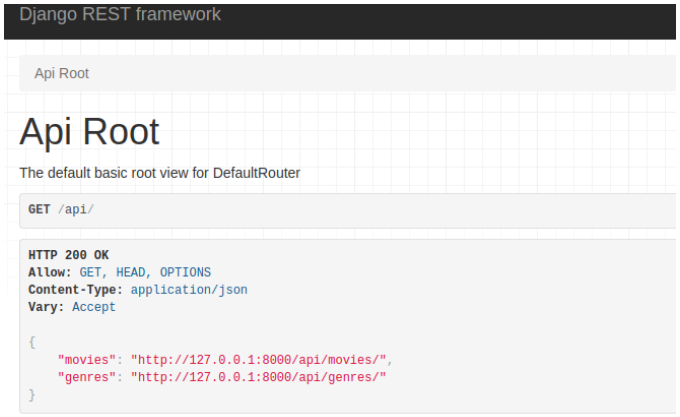
respectivamente. El objeto form contiene los datos del formulario de la película, y errors se utiliza para almacenar los mensajes de error relacionados con la validación del formulario.

```
<script setup>
import { computed, reactive, ref, onMounted } from "vue";
import { StarIcon, TrashIcon, PencilIcon } from "@heroicons/vue/24/solid";
import { api } from "./api";

const movies = ref([]);
const showMovieForm = ref(false);
const form = reactive({
  id: null,
  name: '',
  description: '',
  image: '',
  inTheaters: false,
  genres: [],
});
const errors = reactive({
  name: null,
  description: null,
  image: null,
  inTheaters: null,
  genres: null,
});
const validations = reactive({
```

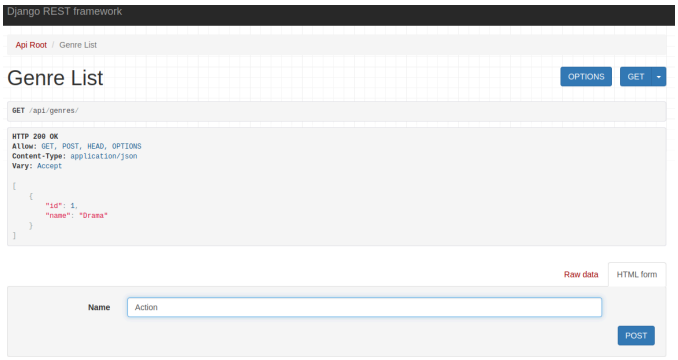
Donde dentro del archivo se observa más funciones dedicadas al crud de la administración de las películas, además de que el mismo archivo contiene otra sección llamada <template>.

Se tiene como principal url en django se tiene.

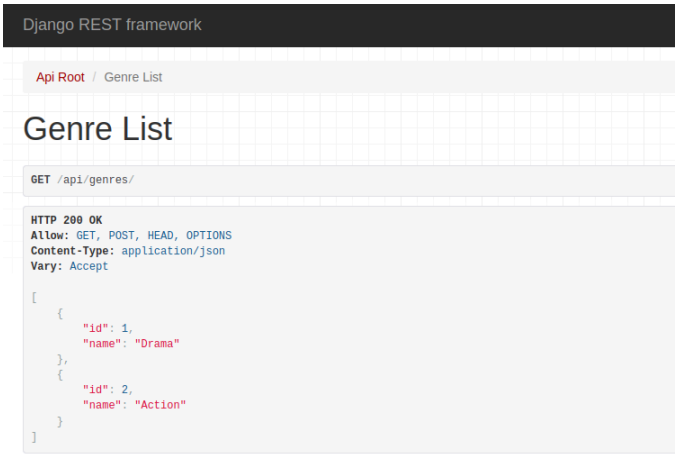
```
linux@linux-ThinkPad-T470:~/LabFinal/backend$ python3 manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).
July 22, 2024 - 03:26:13
Django version 5.0.6, using settings 'backend.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CONTROL-C.

[22/Jul/2024 03:26:21] "GET /api/ HTTP/1.1" 200 5395
```



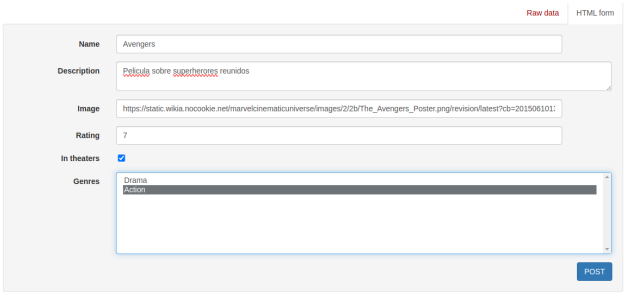
Para realizar la prueba del backend con django, se tiene como primera pantalla, la de Genre.



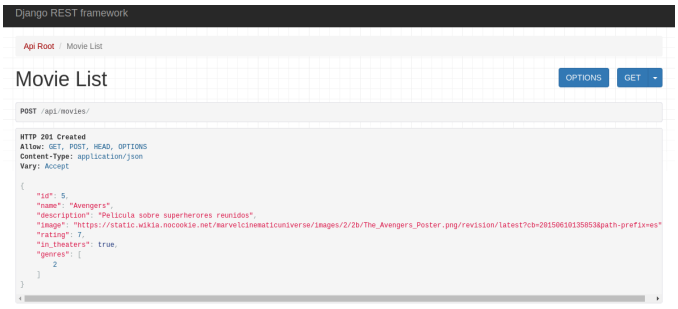
Y se tiene como opción la lista de Géneros para poder crear más y actualizarla en la base de datos.



También está la pantalla de agregar películas.



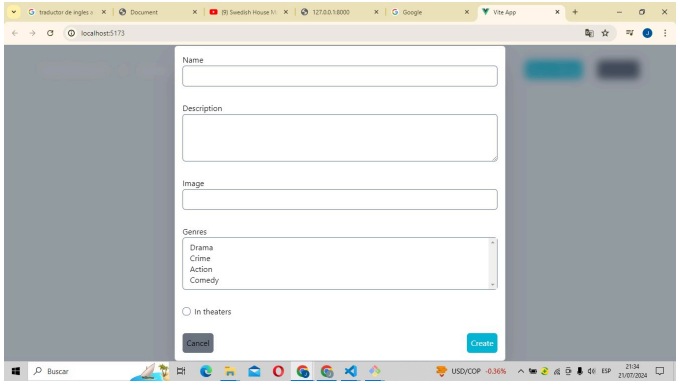
Además de tener también su propia lista.



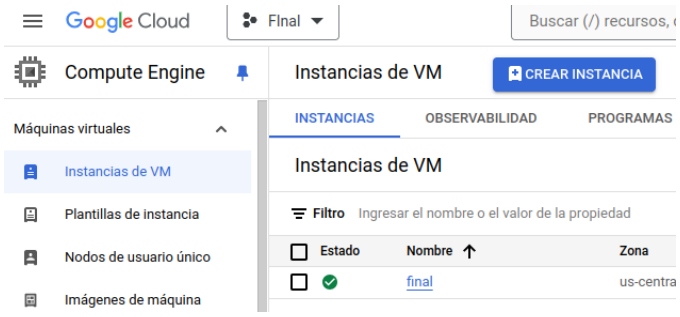
Para el frontend también está su ejecución, que se tiene que realizar después del backend, por si solo no funciona.

```
linux@linux-ThinkPad-T470:~/LabFinal/frontend$ npm run dev

> challenge-1@0.0.0 dev
> vite

Port 5173 is in use, trying another one...

  VITE v5.2.11  ready in 293 ms

  ➜  Local:   http://localhost:5174/
  ➜  Network: use --host to expose
  ➜  press h + enter to show help
```

Y su pantalla para poder agregar películas con su propio css.



Para poder implementar esto a una web ya donde cualquier persona pueda ingresar, se usará una web gratuita que es google cloud[6].



Se creará un instancia y en esta se ira a la opción de ssh donde nos abrirá a una opción similar a un cmd, donde tendremos que clonar los objetos del repositorio.

```
cahuianthony6@final:~$ ls
cahuianthony6@final:~$ git clone https://github.com/acahuib/LabFinal.git
Cloning into 'LabFinal'...
remote: Enumerating objects: 25952, done.
remote: Total 25952 (delta 0), reused 0 (delta 0), pack-reused 25952
Receiving objects: 100% (25952/25952), 38.34 MiB | 18.47 MiB/s, done.
Resolving deltas: 100% (9018/9018), done.
Updating files: 100% (16141/16141), done.
cahuianthony6@final:~$ ls
LabFinal
cahuianthony6@final:~$ cd LabFinal/
cahuianthony6@final:~/LabFinal$ ls
README.md  backend  frontend
cahuianthony6@final:~/LabFinal$
```

Luego se activará el entorno virtual y si falta alguna descarga se descargara lo que te pide, para que pueda arrancar correctamente.

Además se instancian los valores de la dirección a usar y que esté conectado con el django y vue.js.

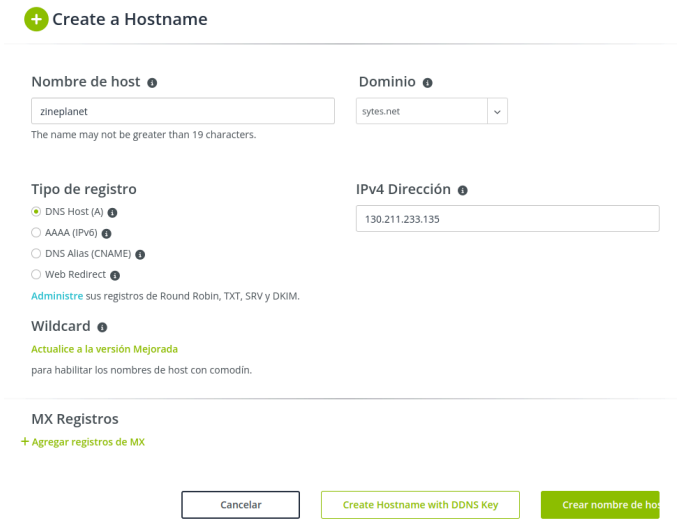
```
 GNU nano 4.8                    /etc/nginx/sites-available/myproject
server {
    listen 80;
    server_name 130.211.233.135 zineplanet.sytes.net;

    # Configuración para servir la aplicación Vue.js
    location / {
        root /home/cahuianthony6/LabFinal/frontend/dist;
        try_files $uri $uri/ /index.html;
    }

    # Configuración para servir la API Django
    location /api/ {
        proxy_pass http://unix:/home/cahuianthony6/LabFinal/myproject.sock;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
    }
}
```

Donde al terminar se coloca ‘sudo ln -s /etc/nginx/sites-available/myproject /etc/nginx/sites-enabled/’ para poder guardarlo y se quiere hacer modificaciones en un futuro se usa ’sudo systemctl restart nginx’.

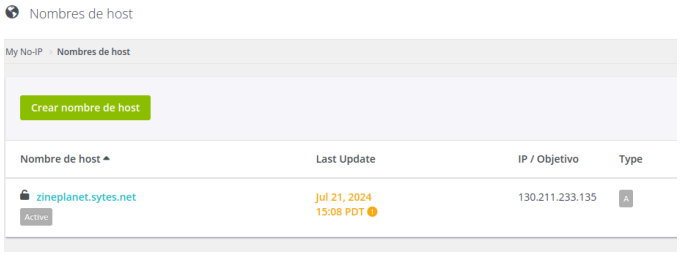
Con esto se tendría solo que ingresar la dirección ip externa ofrecida por google cloud para poder ingresar a la web.

Como opcional para poder tener un dominio con el nombre que queramos se ingresa a una página de nombre noip[5].

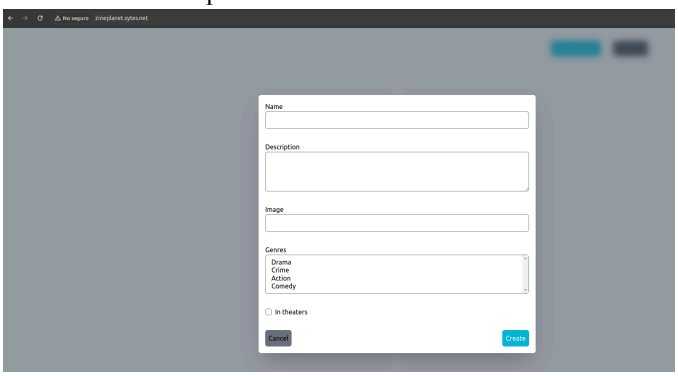
Luego de ingresar nuestro correo simplemente se da la opción de crear un dominio con los siguiente parámetros.



Y teniendo ya así un dominio propio que se le agregaría solo el nombre al ssh del cloud.



Donde para poder ver su ejecución se escribe la url creada, y nos mostrará la pantalla del frontend.



VIII. CONCLUSIONES

La implementación de un sistema CRUD para la gestión de películas, utilizando una arquitectura basada en backend y frontend, y desplegado en un servidor en la nube, demuestra ser una solución robusta y eficiente para la gestión de aplicaciones web.

La utilización de servicios en la nube para el backend, como Google Cloud o AWS, proporciona una infraestructura escalable y de alto rendimiento. Esto permite manejar grandes volúmenes de datos y transacciones, asegurando una experiencia de usuario fluida y rápida. La capacidad de escalar automáticamente según la demanda garantiza que el sistema pueda adaptarse a picos de tráfico sin comprometer su desempeño.

La implementación de un sistema CRUD utilizando tecnologías modernas y servicios en la nube facilita la integración con otras aplicaciones y servicios. Esto permite la expansión y adición de nuevas funcionalidades con relativa facilidad, asegurando que el sistema pueda evolucionar según las necesidades del negocio o las demandas del mercado.

REFERENCIAS

[1] A. Holovaty and J. Kaplan-Moss, (2009). "The Django Book," 2nd ed., Apress.

[2] Vue.js. (2023). "The Progressive JavaScript Framework," Recuperado de: https://vuejs.org/.

[3] Grinberg, M. (2018). Flask Web Development: Developing Web Applications with Python. O'Reilly Media.

[4] Vue.js (2014). The Progressive JavaScript Framework. Recuperdao de: https://vuejs.org

[5] NoIp. 'Servicio de DNS Administrativo'. REcuperado de: https://www.noip.com/es-MX

[6] Google Cloud. Documentation. Ingresado en Julio 21, 2024, recuperado de: https://cloud.google.com/docs/

[7] Vidal-Silva, C. L., Sánchez-Ortiz, A., Serrano, J., & Rubio, J. M. (2021). Experiencia académica en desarrollo rápido de sistemas de información web con Python y Django. Formación universitaria, 14(5), 85-94.

[8] LInk de GitHub: https://github.com/acahuib/LabFinal

Avance de grupo
Cahui Benegas Anthony Ronaldo: 100%
VIlca Mamani Jose Armando: 100%