

Extensión de la app web a Azure

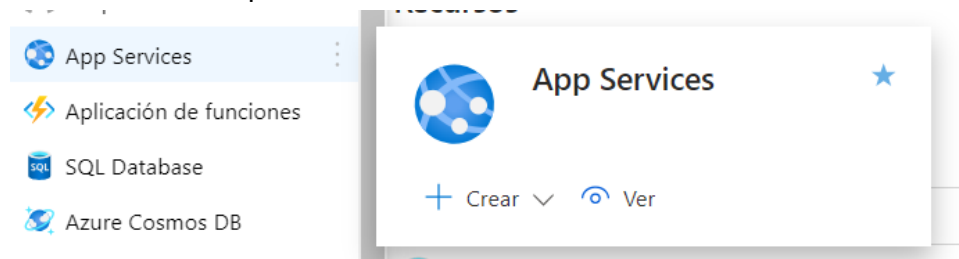
Se ha seguido el siguiente tutorial para extender la aplicación web Django a Azure:

<https://learn.microsoft.com/en-us/azure/app-service/quickstart-python?tabs=flask%2Cwindows%2Cazure-cli%2Cazure-cli-deploy%2Cdeploy-instructions-azportal%2Cterminal-bash%2Cdeploy-instructions-zip-azcli>

Los requisitos previos para realizarlo es tener una cuenta creada en Azure y realizaremos todo el proceso por medio de Azure portal.

<https://portal.azure.com/>

Comenzamos creando un aplicación web en Azure:



Hemos rellenado la configuración de la aplicación web con la siguiente información:

Detalles del proyecto

Seleccione una suscripción para administrar los recursos implementados y los costos. Use los grupos de recursos como carpetas para organizar y administrar todos los recursos.

Suscripción *	<input type="text" value="Azure for Students"/>
Grupo de recursos *	<input type="text" value="(Nuevo) hads2324-django-python"/>

[Crear nuevo](#)

Detalles de instancia

Nombre *	<input type="text" value="SitioDeFilmsG4"/> .azurewebsites.net
Publicar *	<input checked="" type="radio"/> Código <input type="radio"/> Container <input type="radio"/> Aplicación web estática
Pila del entorno en tiempo de ejecución *	<input type="text" value="Python 3.11"/>
Sistema operativo *	<input checked="" type="radio"/> Linux <input type="radio"/> Windows
Región *	<input type="text" value="East US"/>

i ¿No encuentra su plan de App Service? Pruebe otra región o seleccione su App Service Environment.

Planes de precios

El plan de tarifa de App Service determina la ubicación, las características, los costos y los recursos del proceso asociados a la aplicación. [Más información](#)

Plan de Linux (East US) * ⓘ

(Nuevo) ASP-hads2324django-python-bea5

[Crear nuevo](#)

Plan de precios

Básico B1 (Total de ACU: 100, 1.75 GB de memoria, 1 vCPU)

[Explorar planes de precios](#)

[Revisar y crear](#)

[< Anterior](#)

[Siguiente: Base de datos >](#)

✓ Se completó la implementación



Nombre de implementación: Microsoft.Web-WebApp-Portal-3f030...

Hora de inicio: 8/4/2024, 12:22:27

Suscripción: [Azure for Students](#)

Id. de correlación: ed880f84-124d-4e5b-9519-93ec783846ed

Grupo de recursos: [hads2324-django-python](#)



A continuación debemos implementar nuestro código del proyecto a Azure. Se llevará a cabo usando Local Git y Azure portal.

Primero, debemos acceder al recurso creado "SitioDeFilmsG4" y acceder a su centro de implementación, seleccionar *Github*, rellenar los campos necesarios y guardar. Cabe destacar que hemos creado otro repositorio solo con el proyecto de Lab5.

[Configuración](#)

[Registros](#)

[Credenciales de FTPS](#)

Permite implementar y compilar código a partir del proveedor de compilación y código fuente preferidos. [Más información](#)

Origen

GitHub

[Desconectar](#)

GitHub

Sesión iniciada como

xiiomara19

Organización

xiiomara19

Repositorio

HADS-Lab5

Rama

[main](#)

Compilación

Proveedor de compilaci...

Acciones de GitHub

Pila del entorno en tiem...

Python

Versión

Python 3.11

Comenzará el proceso de build y deploy desde *Github*. En Azure portal lo veremos así:



Centro de implementación

Registros de implementación

[Ver registros](#)

Última implementación



En curso, compilando la aplicación
[Actualizar](#)

Proveedor de implementación

GitHubAction

y en Github así:

← Build and deploy Python app to Azure Web App - SitioDeFilmsG4

🔔 Add or update the Azure App Service build and deployment workflow config #1

[Cancel workflow](#)

🕒 Latest #2 ▾



🏠 Summary

Jobs

✅ build

🔔 deploy

Run details

🔧 Usage

📄 Workflow file

Re-run triggered 2 minutes ago

👤 xiomara19 🔗 6629853 main

Status
In progress

Total duration
-

Artifacts
-

main_sitiodefيلمsg4.yml
on: push



🔗 - +

Se implementa correctamente pero al acceder el dominio nos informa de lo siguiente:

DisallowedHost at /

Invalid HTTP_HOST header: 'sitiodefيلمsg4.azurewebsites.net'. You may need to add 'sitiodefيلمsg4.azurewebsites.net' to ALLOWED_HOSTS.

Se añade a setting.py lo mencionado, *sitiodefيلمsg4.azurewebsites.net*. Sin embargo, no funciona correctamente ya que salta el error 403 y es que hace falta crear un array de CSRF_TRUSTED_ORIGINS.

Forbidden (403)

CSRF verification failed. Request aborted.

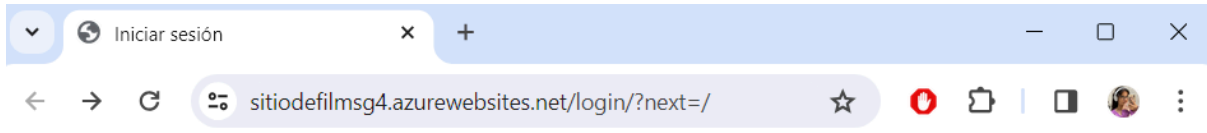
```
ALLOWED_HOSTS = ['sitiodefيلمsg4.azurewebsites.net']

CSRF_TRUSTED_ORIGINS = [
    'https://' + os.environ['WEBSITE_HOSTNAME'],
    'https://localhost',
    'https://127.0.0.1'
]
```

NOTA. Se ha añadido * en el apartado ALLOWED_HOSTS ya que no detectaba el dominio.

```
28     ALLOWED_HOSTS = [  
-     'sitiodefيلمsg4.azurewebsites.net'  
29     +     '*'  
30 ]
```

Por último, se vuelve a construir e implementar el workflow. De esta manera conseguiremos acceder a nuestra aplicación web desde el dominio creado.



The screenshot shows a web browser window with the title 'Iniciar sesión'. The address bar displays 'sitiodefيلمsg4.azurewebsites.net/login/?next=/' with a star icon for bookmarks and a red circle icon for notifications. The main content area has the heading 'INICIAR SESIÓN' in bold. Below the heading are two input fields: 'Username...' with a person icon and 'Password...' with a key icon. A 'Login' button is centered below the fields. At the bottom, there is a link: 'Don't have an account? [Sign Up](#)'.

Sin embargo, podemos observar que no cargan los estilos. Por ello tenemos que añadir el middleware *whitenoise* en requirements.txt. Con este middleware conseguimos implementar nuestros propios estilos en la aplicación.

```
whitenoise~=5.3.0
```

Por último, hemos indicado en urls.py la dirección a la carpeta static para que Azure pueda encontrar los archivos .css.

```
from django.conf import settings  
from django.conf.urls.static import static
```

```
urlpatterns = [  
    path('admin/', admin.site.urls),  
    path('register/', views.registerPage, name='register'),  
    path('login/', views.loginPage, name='login'),  
    path('', views.homePage, name='home'),  
    path('logout/', views.logoutPage, name='logout'),  
    path('films/', views.filmsPage, name='films'),  
    path('vote/', views.votePage, name='vote'),  
    path('votes/', views.votesPage, name='votes')  
] + static(settings.STATIC_URL, document_root=settings.STATIC_ROOT)
```

Tras hacer esto debería de aparecer la página de login, register y home con los estilos establecidos pero eso no ocurre así y no hemos comprendido las razones. Sin embargo, para las páginas films, navegation, votes y vote si se muestra el estilo implementado y es que los estilos se han desarrollado en los html directamente por medio de un link o en cada elemento definiéndolo en las clases.

```
{% load static %}
<link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@4.0.0/dist/css/bootstrap.min.css" integrity="sha384-IyeS5QzscvHycRNdsMwR10/1cKj10eakKoa3WYbTW6qoXWYmjlBn6uwwZC/Bh6I" crossorigin="anonymous">

<div class="container">
  <div class="row justify-content-center">
    <form method="post" action="{% url 'votes' %}" style="width: 100%; border: 1px solid #ccc; padding: 10px;">
      <label for="testua">Elige una película:</label>
      <div class="input-group mb-3">
        <select class="custom-select" name="film">
          {% for p in Pelicula %}
            <option value="{{ p.title }}">{{ p.title }}</option>
          {% endfor %}
        </select>
        <div class="input-group-append">
          <button class="btn btn-primary" type="submit" name="vote">Mostrar aficionados</button>
        </div>
      </div>
    </form>
  </div>
</div>
```

Se puede visualizar nuestra aplicación de sitios de films en el siguiente dominio.

<https://sitiodefilmmsg4.azurewebsites.net/>