Práctica 1 – Django, MySQL y Git con XAMPP

Curso: Ingeniería para la Web
Docente: Jorge
Estudiante: ____
Fecha: ____

💡 Objetivo

Configurar un entorno de desarrollo completo en **Windows Server** utilizando **Django** y **MySQL** con **XAMPP**, e integrar **control de versiones con Git** en Visual Studio Code.

🧰 Requisitos

- Python 3.12 o superior
- XAMPP para Windows
- Visual Studio Code
- Git instalado (https://git-scm.com/downloads)
- Navegador web

🗜️ Parte 1: Instalación de XAMPP

1. Descarga **XAMPP** desde apachefriends.org.
2. Instálalo en `C:\xampp` (no en Archivos de programa).
3. Selecciona los módulos: **Apache**, **MySQL**, **phpMyAdmin**.
4. Inicia el Panel de Control y presiona **Start** en Apache y MySQL.
5. Verifica en el navegador: http://localhost

(Insertar captura de XAMPP corriendo)

🧰 Parte 2: Configurar MySQL

1. Abre phpMyAdmin: http://localhost/phpmyadmin
2. Crea la base de datos `django_db` y un usuario `django_user` con todos los permisos.

SQL:

```
CREATE USER 'django_user'@'localhost' IDENTIFIED BY 'clave_segura';
GRANT ALL PRIVILEGES ON django_db.* TO 'django_user'@'localhost';
FLUSH PRIVILEGES;
```

🧹Parte 3: Crear entorno Django

1. Abre **VS Code como administrador**.
2. Crea la carpeta del proyecto:

```
C:\jorge\Doc\proyectos
```

3. En la terminal:

```
python -m venv venv
venv\Scripts\activate
pip install django mysqlclient
django-admin startproject webapp .
```

⚙️Parte 4: Configurar la base de datos

Edita `webapp/settings.py`:

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': 'django_db',
        'USER': 'django_user',
        'PASSWORD': 'clave_segura',
        'HOST': '127.0.0.1',
        'PORT': '3307',
    }
}
```

🧱Parte 5: Migraciones y superusuario

```
python manage.py migrate
python manage.py createsuperuser
```

Ejemplo:

```
Usuario: JorgeDjango25
Contraseña: ********
```

Parte 6: Crear app de usuarios

```
python manage.py startapp cuentas
```

Agrega `'cuentas'` a `INSTALLED_APPS`.

`cuentas/urls.py`:

```
from django.urls import path
from . import views

urlpatterns = [
    path('login/', views.login_view, name='login'),
    path('bienvenida/', views.bienvenida, name='bienvenida'),
    path('logout/', views.logout_view, name='logout'),
]
```

`webapp/urls.py`:

```
from django.contrib import admin
from django.urls import path, include
from django.shortcuts import redirect

urlpatterns = [
    path('', lambda request: redirect('login')),
    path('admin/', admin.site.urls),
    path('', include('cuentas.urls')),
]
```

Parte 7: Crear vistas (views.py)

```
from django.shortcuts import render, redirect
from django.contrib.auth import authenticate, login, logout
from django.contrib.auth.decorators import login_required
from django.contrib import messages

def login_view(request):
    if request.method == 'POST':
        username = request.POST.get('username')
        password = request.POST.get('password')
        user = authenticate(request, username=username, password=password)
        if user is not None:
            login(request, user)
            return redirect('bienvenida')
        else:
            messages.error(request, 'Usuario o contraseña incorrectos')
```

```
    return render(request, 'cuentas/login.html')

@login_required
def bienvenida(request):
    return render(request, 'cuentas/bienvenida.html', {'usuario':
request.user})

def logout_view(request):
    logout(request)
    return render(request, 'cuentas/logout.html')
```

🕐Parte 8: Plantillas con Bootstrap 5

base.html – plantilla principal con barra de navegación.
login.html, **bienvenida.html**, **logout.html** – formularios y vistas estilizadas.

(Usar código de la práctica anterior)

🧰Parte 9: Prueba final

1. Ejecuta:

   ```
   python manage.py runserver
   ```

2. Abre http://127.0.0.1:8000
3. Inicia sesión y verifica funcionamiento de login/logout.

🔦Parte 10: Control de versiones con Git en Visual Studio Code

Basado en la guía: rogerdudler.github.io/git-guide/index.es.html

🕰️1. Configurar Git por primera vez

Abre la terminal y configura tu usuario:

```
git config --global user.name "TuNombre"
git config --global user.email "tuemail@ejemplo.com"
```

Verifica la configuración:

```
git config --list
```

🕰️2. Inicializar repositorio Git

En la carpeta del proyecto (donde está `manage.py`):

```
git init
```

Esto creará una carpeta oculta `.git`.

🕰️3. Crear archivo `.gitignore`

Crea un archivo en la raíz del proyecto con este contenido:

```
venv/
__pycache__/
*.pyc
*.db
.DS_Store
.env
```

Esto evita subir archivos innecesarios.

🕰️4. Agregar y confirmar cambios

```
git add .
git commit -m "Primer commit: Proyecto Django conectado a MySQL"
```

🕰️5. Conectar con GitHub

1. Crea un repositorio nuevo en GitHub (sin README ni .gitignore).
2. Copia la URL HTTPS del repositorio.
3. En la terminal VS Code:

```
git remote add origin https://github.com/tu_usuario/tu_repositorio.git
git branch -M main
git push -u origin main
```

🕰️6. Actualizar el repositorio

Cuando modifiques tu proyecto:

```
git add .
git commit -m "Actualización de vistas o plantillas"
git push
```

🕰️7. Ver estado y log

Ver archivos pendientes:

```
git status
```

Ver historial de commits:

```
git log --oneline
```

🕰️8. Recomendaciones para el aula

- Realiza commits pequeños y frecuentes.
- Usa mensajes descriptivos.
- Verifica siempre con `git status` antes de hacer push.
- Evita subir archivos de entorno (`venv`, `__pycache__`, `db.sqlite3`).

🔗Resultado final

Los estudiantes tendrán: - Entorno Django funcionando con MySQL. - Proyecto controlado por Git desde Visual Studio Code. - Repositorio remoto en GitHub con código limpio y documentado.