# Proyecto biUNestar – Documentación Técnica

Integrantes:

Pablo Rosales Obando

Jaime Javier Quiñones Segura

Omar Andrés Quiñones Mejía

## 1. Introducción

El proyecto biUNestar es una aplicación web desarrollada con el framework Django, cuyo objetivo es ayudar a los estudiantes universitarios a organizar y mantener hábitos saludables relacionados con sueño, actividad física, hidratación y bienestar emocional.
El sistema permite registrar hábitos, visualizar reportes semanales y recibir recordatorios personalizados, promoviendo así la constancia y la autoevaluación.

## 2. Contexto y motivación

El estrés académico y la falta de estructura en la vida cotidiana son factores que deterioran la salud y el rendimiento de los estudiantes.
biUNestar busca combatir estos problemas ofreciendo una herramienta intuitiva que ayude al usuario a estructurar sus rutinas, medir su progreso y mantener constancia.

Los principales usuarios del sistema son:

Estudiantes: registran hábitos y consultan su progreso.

Administradores: gestionan contenidos de apoyo y supervisan usuarios.

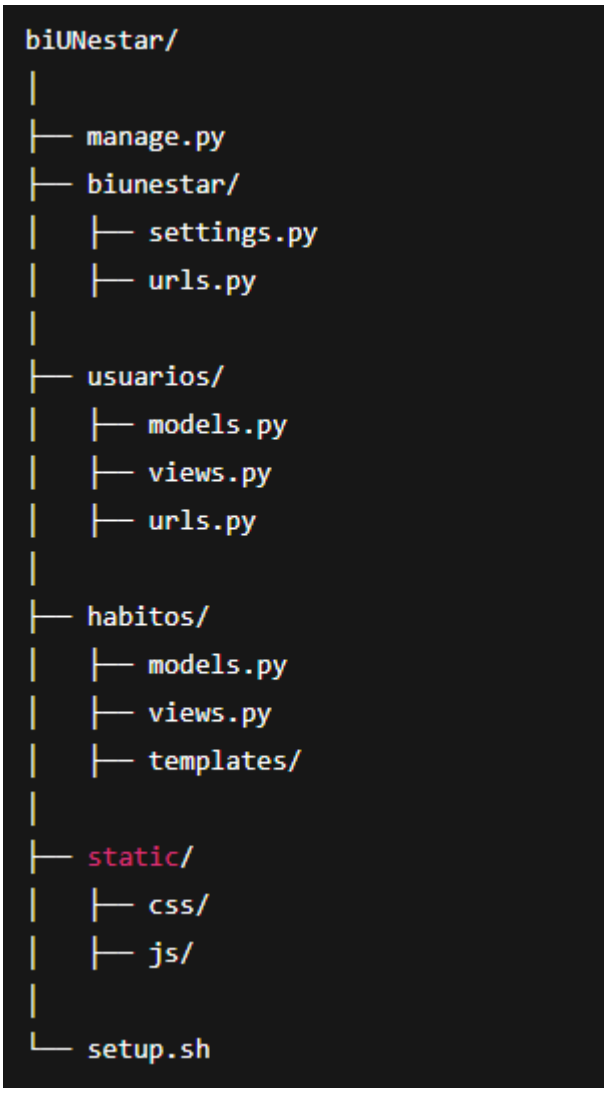
## 3. Requisitos del sistema revisar

https://github.com/davidRamirezLopez/AltF4/blob/main/Documentaci%C3%B3n/Proyecto/Entrega_03.pdf

## 4. Diseño y arquitectura

El sistema está implementado bajo el framework Django, que sigue el patrón Modelo-Vista-Controlador (MVC).
Se utiliza el patrón Singleton para la conexión a la base de datos, asegurando que solo exista una instancia activa.
La arquitectura modular permite separar claramente las funciones de cada componente.

Estructura general:




5. Fragmentos de código y explicación

5.1 Configuración general (settings.py)

```python
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'usuarios',
    'habitos',
]

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': BASE_DIR / 'db.sqlite3',
    }
}
```

 Este archivo define las configuraciones globales del proyecto.

INSTALLED_APPS registra los módulos activos del sistema (como usuarios y habitos).

DATABASES configura la conexión a la base de datos (en este caso, SQLite3).
Permite centralizar la administración del proyecto y preparar el entorno de ejecución.

5.2 Modelos de datos (models.py)

```python
from django.contrib.auth.models import User
from django.db import models

class Perfil(models.Model):
    usuario = models.OneToOneField(User, on_delete=models.CASCADE)
    edad = models.IntegerField(null=True)
    carrera = models.CharField(max_length=100)
    objetivos = models.TextField(blank=True)

    def __str__(self):
        return self.usuario.username
```

 Define la estructura de la tabla Perfil asociada a cada usuario.

OneToOneField vincula el modelo con la cuenta base del usuario (User).

Incluye información complementaria como edad, carrera y objetivos personales.

 El modelo Habito guarda los registros diarios del usuario.
Cada hábito está asociado a un tipo (sueño, agua, ejercicio) y almacena un valor cuantitativo.

5.3 Vistas principales (views.py)

```python
from django.shortcuts import render, redirect
from django.contrib.auth import authenticate, login
from django.contrib import messages

def login_usuario(request):
    if request.method == 'POST':
        username = request.POST['username']
        password = request.POST['password']
        user = authenticate(request, username=username, password=password)
        if user:
            login(request, user)
            return redirect('dashboard')
        else:
            messages.error(request, "Credenciales incorrectas.")
    return render(request, 'login.html')
```

Controla la lógica de inicio de sesión:

Verifica las credenciales del usuario con authenticate().

Si son correctas, inicia sesión con login() y redirige al panel principal.

En caso de error, muestra un mensaje usando el sistema de alertas de Django

5.4 Enrutamiento (urls.py)

```python
from django.contrib import admin
from django.urls import path, include

urlpatterns = [
    path('admin/', admin.site.urls),
    path('usuarios/', include('usuarios.urls')),
    path('habitos/', include('habitos.urls')),
]
```

Define las rutas principales del sistema.
Cada módulo tiene su propio archivo urls.py, lo que permite mantener la arquitectura modular y limpia.
El include() enlaza las rutas de cada aplicación secundaria.

5.5 Patrón Singleton (database.py)

```python
class DatabaseConnection:
    _instance = None

    def __new__(cls):
        if cls._instance is None:
            cls._instance = super(DatabaseConnection, cls).__new__(cls)
            cls._instance.connection = sqlite3.connect('biunestar.db')
        return cls._instance

    def get_connection(self):
        return self.connection
```

 implementa el patrón Singleton, garantizando que exista una única conexión activa con la base de datos. Previene errores por múltiples conexiones y optimiza el acceso a los datos.

5.6 Script de automatización (setup.sh)

```bash
#!/bin/bash
echo "Inicializando entorno virtual..."
python -m venv venv
source venv/bin/activate

echo "Instalando dependencias..."
pip install -r requirements.txt

echo "Migrando base de datos..."
python manage.py makemigrations
python manage.py migrate

echo "Levantando el servidor..."
python manage.py runserver
```

 Automatiza el despliegue del sistema:

Crea el entorno virtual.

Instala dependencias.

Ejecuta migraciones de base de datos.

Inicia el servidor automáticamente.

Esto facilita la instalación del proyecto en nuevos entornos.

5.7 Interfaz de usuario (login.html)

```html
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <title>Iniciar sesión - biUNestar</title>
    <link rel="stylesheet" href="{% static 'css/style.css' %}">
</head>
<body>
    <div class="login-container">
        <h1>Bienvenido a biUNestar</h1>
        <form method="post">
            {% csrf_token %}
            <input type="text" name="username" placeholder="Usuario" required>
            <input type="password" name="password" placeholder="Contraseña" required>
            <button type="submit">Ingresar</button>
        </form>
    </div>
</body>
</html>
```

 Plantilla HTML que define la interfaz del login.
Utiliza Django Template Language (DTL) para insertar el token de seguridad {% csrf_token %} y acceder a archivos estáticos.

5.8 Estilos principales (style.css)

```css
body {
    background-color: #f7f9fa;
    font-family: 'Inter', sans-serif;
    display: flex;
    justify-content: center;
    align-items: center;
    height: 100vh;
}

.login-container {
    background: white;
    padding: 2rem;
    border-radius: 12px;
    box-shadow: 0 0 10px rgba(0,0,0,0.1);
    text-align: center;
}
```

 Define el estilo base de la aplicación: colores suaves, estructura centrada y diseño minimalista para el formulario de inicio de sesión.

6. Gestión del desarrollo

El proyecto se gestiona mediante GitHub (control de versiones) y Trello (seguimiento de tareas).
Cada integrante se especializa en diferentes áreas: backend, frontend y documentación.
El trabajo colaborativo ha permitido cumplir con los requisitos funcionales principales.

7. Conclusiones y próximos pasos

Se ha construido una versión funcional inicial con autenticación, registro de hábitos y estructura modular.

El código cumple principios de diseño y buenas prácticas de Django.

Los siguientes pasos incluyen:

Implementar reportes visuales con gráficas.

Añadir sistema de recordatorios automáticos.

Mejorar la usabilidad y desplegar el sistema en producción.