

UNIVERSIDAD NACIONAL DE INGENIERÍA

FACULTAD DE INGENIERÍA INDUSTRIAL Y DE SISTEMAS



SW101-U INTRODUCCIÓN A LA INGENIERÍA EN SOFTWARE

“FRAMEWORKS”

PRÁCTICA N°4

GRUPO N°3:

- Castro Aguirre Lenin Giomar
- Eustaquio Avila Luis Gabriel
- Garay Yovera Alexis Yanpoll

Docente: Ing. Javier Sánchez Espinoza

19 de junio del 2024

Índice

Introducción	5
Objetivos	5
Capítulo 1: Aspectos Previos	6
¿Qué es un Framework?.....	6
¿Qué Constituye un Framework?.....	6
Librerías	6
Herramientas	6
Patrones.....	7
APIs (Application Programming Interfaces)	7
Plantillas (Templates)	7
Capítulo 2: Ventajas y Desventajas de los Frameworks	8
Estructura y Organización del Código Predeterminada.....	8
Reutilización del Código y Evitar Duplicidades.....	8
Agilidad y Rapidez en el Desarrollo.....	8
Minimización de Errores y Mayor Facilidad para Solucionarlos	8
Facilidad para Encontrar Librerías y Código Complementario	9
Colaboración Simplificada entre Desarrolladores	9
Mantenimiento Facilitado	9
Desventajas	9
Tiempo de Aprendizaje.....	9

	3
Versiones Inestables.....	10
Menor Rendimiento	10
Código sin Utilizar.....	10
Elección del Framework	11
Capítulo 3: Tipos de Frameworks.....	12
Frameworks de Back-End.....	12
Frameworks de Front-End	12
Frameworks de Machine Learning	12
Frameworks para Desarrollo de Juegos	13
Frameworks para Desarrollo Móvil.....	13
Capítulo 4: Criterios para la elección de un framework.....	14
Tipo de Proyecto	14
Ejemplo Práctico.....	14
Compatibilidad Tecnológica.....	15
Seguridad	15
Comunidad y Soporte	15
Retroalimentación y Experiencias Previas.....	15
Capítulo 5: Proyecto con el Framework Flask.....	16
Descripción del proyecto	16
Desarrollo.....	18
Frontend.....	18

Backend.....	21
Resultados	24
Conclusiones	27
Referencias	28

Introducción

Objetivos

En este trabajo monográfico se pretende conocer a profundidad acerca de una herramienta bastante utilizada actualmente por los programadores en cualquier ámbito del desarrollo de software, los frameworks. Consigo explicaremos su definición y los elementos que lo conforman. Asimismo, explicaremos las diferentes ventajas que nos aportan los frameworks, sin embargo, también posee algunas desventajas que también daremos a conocer.

Y para poder poner en práctica todo lo aprendido acerca del tema de los frameworks, realizaremos un programa como evidencia de esto. Este programa será una aplicación web que será desarrollado a través del framework Flask, el cual funciona en el lenguaje de programación Python.

Capítulo 1: Aspectos Previos

¿Qué es un Framework?

Un framework, que sería marco de trabajo traducido al español, es un modelo base que ofrece un conjunto de herramientas, bibliotecas y un conjunto estándar de las buenas prácticas al momento de programar, es decir, los frameworks son una base sobre la cual se pueden desarrollar softwares de manera más eficiente y organizada. Proporciona soluciones ya codificadas para tareas comunes y patrones recurrentes, liberando a los desarrolladores el tener que inventar código que ya ha sido creado. Esto les permite centrarse en las funcionalidades específicas y la lógica del proyecto, lo que hace que la elaboración del software sea más eficiente y mejorada.

¿Qué Constituye un Framework?

Librerías

Las librerías son una parte importante en el desarrollo de software, ya que estos hacen que se optimicen los procesos al momento de programar. Esto lo hacen porque ofrecen soluciones a problemas comunes, y esto es reutilizable por cualquiera que use dicha librería, en otras palabras, es una colección de código previamente desarrollado a problemas usuales.

Herramientas

Estas herramientas son software adicional que ayuda al desarrollo, incluyendo depuradores, editores de código e IDEs, sistemas de control de versiones, herramientas de prueba, sistemas de construcción y generadores de código. Dichas herramientas pueden estar integradas al framework o complementarlo de manera externa.

Patrones

Los patrones también son parte de los frameworks, ya que esto traen consigo beneficios como la reutilización, mantenibilidad, estandarización y flexibilidad. Entre ellos están los patrones creacionales, estructurales y de comportamiento, por ejemplo, el framework angular usa el patrón Modelo – Vista – ViewModel.

APIs (Application Programming Interfaces)

Según Amazon AWS (s.f.), las APIs o Interfaces de programación de aplicaciones son un conjunto de mecanismos conformados por protocolos que permite que dos componentes de software puedan comunicarse entre sí. En el caso de los frameworks, serían las interfaces que permitan que los desarrolladores puedan interactuar con el framework para acceder a sus distintas herramientas o funcionalidades que sean requeridas.

Plantillas (Templates)

Los templates o plantillas hacen referencia a una estructura previamente establecida de archivos que puede ser utilizada en un sitio web, en la que solo se tiene que modificar o adaptar ciertos parámetros de acuerdo con lo que necesitemos, por lo cual esta bastante relacionado a lo que es el tema de reutilización del software.

Capítulo 2: Ventajas y Desventajas de los Frameworks

Ventajas

Estructura y Organización del Código Predeterminada

Los frameworks proporcionan un esqueleto y una metodología de trabajo predefinida. Esto elimina la necesidad de analizar dónde colocar los diferentes archivos de una aplicación (recursos, controladores, vistas, modelos, etc.). Gracias a esta organización predeterminada, los desarrolladores pueden seguir un patrón uniforme que facilita el mantenimiento y la escalabilidad del proyecto.

Reutilización del Código y Evitar Duplicidades

En el desarrollo de aplicaciones, muchas funcionalidades suelen repetirse, como la conexión a bases de datos, la validación de formularios y la gestión de estilos. Los frameworks incluyen estas funcionalidades de manera predeterminada, lo que permite a los desarrolladores enfocarse en aspectos únicos de la aplicación en lugar de reinventar la rueda. Esto no solo ahorra tiempo, sino que también asegura la consistencia y la fiabilidad del código reutilizado.

Agilidad y Rapidez en el Desarrollo

La reutilización de código y las funcionalidades preconstruidas permiten un desarrollo más rápido y ágil. Al no tener que crear desde cero las funcionalidades básicas y repetitivas, los desarrolladores pueden dedicar más tiempo a mejorar y perfeccionar las características específicas de la aplicación. Esto resulta en un ciclo de desarrollo más eficiente y productivo.

Minimización de Errores y Mayor Facilidad para Solucionarlos

Al utilizar un framework, los desarrolladores se benefician del trabajo previo de otros programadores que ya han depurado y optimizado el código base. Esto reduce significativamente la probabilidad de errores y, en caso de que surjan problemas, es más

probable que ya exista una solución documentada por la comunidad. Esto contribuye a un desarrollo más seguro y estable.

Facilidad para Encontrar Librerías y Código Complementario

Los frameworks populares suelen tener un ecosistema robusto de herramientas y librerías que complementan su funcionalidad. Esto facilita la integración de nuevas características y soluciones sin tener que desarrollar todo desde cero. La disponibilidad de estas herramientas mejora la eficiencia y la capacidad de respuesta del equipo de desarrollo.

Colaboración Simplificada entre Desarrolladores

Una estructura de código uniforme facilita la colaboración entre desarrolladores. Ya sea dentro de un equipo o en una comunidad más amplia, como GitHub, la familiaridad con la estructura del framework permite a los desarrolladores entender y modificar el código más fácilmente. Esto mejora la eficiencia en proyectos colaborativos y reduce la curva de aprendizaje para nuevos miembros del equipo.

Mantenimiento Facilitado

Un enfoque uniforme y estructurado en el desarrollo facilita el mantenimiento y las actualizaciones de la aplicación. Cuando todos los miembros del equipo siguen las mismas pautas y utilizan las mismas herramientas, cualquier actualización o modificación se puede realizar de manera más rápida y con menor riesgo de introducir errores. Esto resulta en una gestión de proyectos más eficiente y menos costosa a largo plazo.

Desventajas

Tiempo de Aprendizaje

Antes de empezar a utilizar un framework, es necesario invertir tiempo en familiarizarse con su estructura, componentes y métodos de comunicación interna. Esta curva de aprendizaje puede ser significativa, especialmente para aquellos que son nuevos en el

framework o en el desarrollo de software en general. Durante este periodo de adaptación, es posible que la productividad disminuya, ya que los desarrolladores necesitan entender cómo funciona el framework y cómo integrarlo eficientemente en su flujo de trabajo.

Versiones Inestables

La popularidad de los frameworks implica que estén en constante actualización para incorporar nuevas tecnologías y políticas de seguridad. Sin embargo, esto también significa que las versiones más recientes pueden ser inestables y presentar problemas de compatibilidad con otras librerías o herramientas utilizadas en el proyecto. Utilizar una versión muy reciente del framework puede resultar en bugs no documentados y problemas de seguridad, lo que puede ralentizar el desarrollo y generar retrabajo.

Menor Rendimiento

Los frameworks, por su naturaleza, están diseñados para ser genéricos y cubrir una amplia variedad de casos de uso. Esta generalidad puede resultar en un mayor consumo de recursos comparado con una aplicación optimizada desarrollada desde cero. En aplicaciones muy exigentes en términos de rendimiento, el uso de un framework puede ser contraproducente, ya que puede introducir una sobrecarga innecesaria y limitar la capacidad de la aplicación para manejar grandes volúmenes de datos o altas tasas de solicitudes.

Código sin Utilizar

En proyectos pequeños o con requisitos específicos limitados, el uso de un framework completo puede ser excesivo. Estos frameworks suelen incluir una gran cantidad de funcionalidades y características que pueden no ser necesarias para el proyecto en cuestión. Esto resulta en un aumento del tamaño de la aplicación y un consumo innecesario de recursos, ya que se incluye mucho código que no se utilizará. En estos casos, desarrollar una solución a medida puede ser más eficiente y ligero.

Elección del Framework

Con una gran variedad de frameworks disponibles, cada uno con sus propias características y ventajas, elegir el más adecuado para un proyecto específico puede ser un desafío. La elección de un framework requiere una evaluación cuidadosa de factores como la comunidad de soporte, la documentación disponible, la frecuencia de actualizaciones y la compatibilidad con otras herramientas. Seleccionar un framework que luego queda desactualizado o es poco utilizado puede resultar en una pérdida de tiempo y recursos, ya que el desarrollador podría tener que migrar a otro framework o enfrentarse a una falta de soporte y actualizaciones.

Capítulo 3: Tipos de Frameworks

Frameworks de Back-End

Estos frameworks nos ayudan a desarrollar el back-end de una web, que sería el código que el usuario no puede ver ni interactuar porque está del lado del servidor. Estos frameworks ayudan a poder gestionar mejor las bases de datos, hacer la autenticación de usuarios y otras funcionalidades más. Las características en común de los distintos frameworks de este tipo son el MVC (Model-View-Controller), el ORM (Object-Relational-Mapping), etc. Algunos ejemplos de frameworks para el back-end serían Django en Python, Ruby on Rails en Ruby, Spring en Java, Flask en Python, etc.

Frameworks de Front-End

Los frameworks de front-end al contrario que los de back-end, nos ayudan a desarrollar el código que el usuario visualizará a través de una interfaz, y con la cual interaccionará. Una de sus características sería el Data Binding, el cuál vincula o sincroniza automáticamente la UI con el modelo de datos. Estos frameworks nos ayudarán a simplificar el código del lado del cliente de una aplicación, como menú de navegación, botones, imágenes, etc. Algunos ejemplos de estos serían: React en JavaScript, Angular en TypeScript, Vue.js en JavaScript, etc.

Frameworks de Machine Learning

Este tipo de frameworks son conjuntos ya establecidos con varias funcionalidades que nos ayudarán en el desarrollo y en el entrenamiento de los modelos de Machine Learning o Aprendizaje Automático, haciéndolo más rápido y eficiente. Sus características son el soporte para redes neuronales y los algoritmos ML. Algunos ejemplos serían TensorFlow en Python o C++, Scikit-learn en Python, PyTorch en Python, etc.

Frameworks para Desarrollo de Juegos

Estos frameworks han sido diseñados específicamente para el desarrollo de juegos debido a las herramientas especializadas que nos brindan, las cuales facilitan manejar los gráficos, físicas y otros aspectos de los juegos. Incluyen motores gráficos, motores de físicas, etc. Algunos ejemplos de estos serían Unreal Engine en C++, Unity en C++ y C#, Godot en GDScript, etc.

Frameworks para Desarrollo Móvil

Estos están diseñados para el desarrollo de aplicaciones móviles en los sistemas operativos de los móviles como iOS y Android, por lo que permiten poder utilizar herramientas y lenguajes específicos para estas plataformas. Dentro de sus características están la compatibilidad multiplataforma, el acceso a componentes nativos, etc. Algunos ejemplos de estos serían React Native en JavaScript, Flutter en Dart, Xamarin en C#, etc.

Capítulo 4: Criterios para la elección de un framework

Tipo de Proyecto

El tipo de proyecto en el que estás trabajando puede influir considerablemente en la elección del framework. Algunos de estos están específicamente diseñados para ciertos tipos de proyectos, como aplicaciones de alta carga de datos, aplicaciones de una sola página (SPA) o sitios de comercio electrónico. Es crucial considerar la complejidad, la funcionalidad y el propósito del proyecto para determinar cuál framework se ajustará mejor a tus necesidades.

Ejemplo Práctico

Supongamos que necesitas desarrollar una aplicación de comercio electrónico. El propósito del proyecto es vender productos en línea con una interfaz de usuario atractiva y una administración robusta del inventario y las órdenes. Los requisitos incluyen:

- Interfaz de usuario moderna y responsive.
- Sistema seguro de autenticación de usuarios.
- Gestión de productos y órdenes.
- Soporte para pagos en línea.
- Escalabilidad para manejar un creciente número de usuarios.

En este caso, un framework como Django podría ser adecuado debido a:

- Su fuerte soporte para el backend y administración de bases de datos.
- La inclusión de un sistema de autenticación robusto.
- La posibilidad de integrar fácilmente módulos de terceros para pagos en línea.
- Su capacidad de escalabilidad.

Compatibilidad Tecnológica

Asegúrate de que el framework que elijas sea adecuado para las tecnologías que planeas emplear en tu proyecto, incluyendo el lenguaje de programación, la base de datos y otros componentes esenciales. La falta de compatibilidad tecnológica puede ocasionar problemas de integración y restringir la flexibilidad de tu aplicación o proyecto.

Seguridad

La seguridad es un punto muy importante en cualquier proyecto de informática. Debido a esto , se recomienda elegir un framework que tenga medidas integradas de seguridad y que sea compatible con las prácticas recomendadas para proteger nuestra aplicación contra vulnerabilidades y ataques cibernéticos.

Comunidad y Soporte

Un Framework deberá constar de una comunidad activa y de un soporte sólido, ya que esta se encargará de proporcionarnos recursos adicionales, bibliotecas útiles, actualizaciones regulares , entre otras cosas. Debemos considerar el revisar los foros de desarrolladores, los repositorios de código y la frecuencia de las actualizaciones para evaluar adecuadamente el nivel de respaldo que ofrece.

Retroalimentación y Experiencias Previas

Al investigar y recopilar comentarios de otros desarrolladores que han usado el framework en proyectos similares, obtienes una visión clara de cómo funciona en la práctica. Las experiencias positivas te ofrecen confianza en sus capacidades, mientras que los comentarios negativos te alertan sobre posibles desafíos técnicos o limitaciones. Esto ayuda a tomar una decisión informada al seleccionar el framework más adecuado para el proyecto a desarrollar , optimizando así el proceso de desarrollo y reduciendo riesgos.

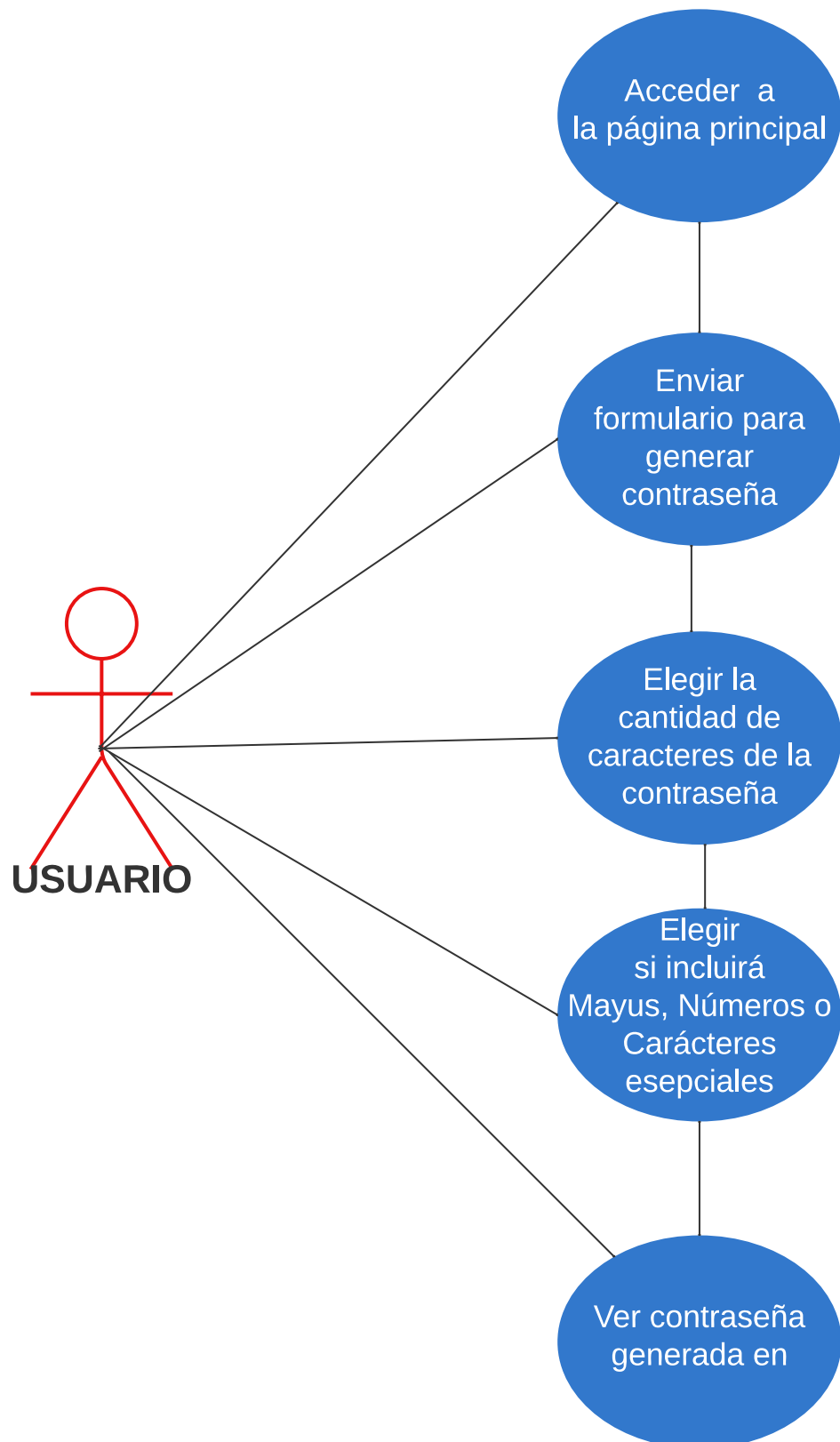
Capítulo 5: Proyecto con el Framework Flask

Descripción del proyecto

El proyecto realizado es un generador de contraseñas seguras. Este generador permite crear contraseñas aleatorias utilizando diferentes tipos de caracteres: números, letras minúsculas, letras mayúsculas y caracteres especiales. La seguridad de las contraseñas se garantiza debido a la aleatoriedad en la selección de estos caracteres, lo que hace que las contraseñas sean difíciles de descifrar. El usuario puede personalizar las contraseñas según lo que necesita, él especifica la longitud que requiere y elige los tipos de caracteres incluir.

El proyecto usa el framework Flask de Python , para proporcionar una interfaz web donde los usuarios pueden ingresar sus preferencias y recibir una contraseña generada automáticamente.

Diagrama de Casos de Uso



Desarrollo

Frontend

```

1 <!DOCTYPE html>
2 <html lang="es">
3 <head>
4     <meta charset="UTF-8">
5     <meta name="viewport" content="width=device-width, initial-scale=1.0">
6     <title>Generador de Contraseñas Seguras</title>
7     <link rel="stylesheet" href="/static/styles.css">
8 </head>

```

Se inicializa el *index.html*

```

1 <body>
2     <h1>Generador de Contraseñas Seguras</h1>
3     <form method="POST">
4         <label for="length">Longitud de la contraseña:</label>
5         <input type="number" name="length" id="length" value="8" min="4" max="32" required>
6         <br>
7         <div class="checkbox-container">
8             <input type="checkbox" name="uppercase" id="uppercase" {% if use_uppercase %}checked{% endif %}>
9             <label for="uppercase">Incluir mayúsculas</label>
10        </div>
11        <div class="checkbox-container">
12            <input type="checkbox" name="numbers" id="numbers" {% if use_numbers %}checked{% endif %}>
13            <label for="numbers">Incluir números</label>
14        </div>
15        <div class="checkbox-container">
16            <input type="checkbox" name="special" id="special" {% if use_special %}checked{% endif %}>
17            <label for="special">Incluir caracteres especiales</label>
18        </div>
19        <button type="submit">Generar Contraseña</button>
20    </form>
21    {% if password %}
22    <h2>Contraseña Generada:</h2>
23    <p>{{ password }}</p>
24    {% endif %}
25 </body>
26 </html>

```

En la parte del body se crea la interfaz para generar las contraseñas. Dentro de ella, hay un formulario que permite seleccionar los tipos de caracteres a usar (letras minúsculas, letras mayúsculas, números y caracteres especiales) y escoger la longitud de la contraseña. Además, se encuentra un botón que permitirá generar la contraseña. Y también se incluye una sección

que, si no detecta una contraseña generada, no muestra nada; pero si detecta una contraseña, la muestra.

```
1  body {
2    font-family: Arial, sans-serif;
3    text-align: center;
4    margin-top: 50px;
5  }
6
7  form {
8    display: inline-block;
9    text-align: left;
10   margin-bottom: 20px;
11 }
12
13 input[type="number"] {
14   width: 60px;
15   padding: 5px;
16   margin: 10px 0;
17 }
18
19 button {
20   display: block;
21   width: 100%;
22   padding: 10px 20px;
23   background-color: #4CAF50;
24   color: white;
25   border: none;
26   cursor: pointer;
27   margin-top: 10px;
28 }
29
30 button:hover {
31   background-color: #45a049;
32 }
33
34 h2 {
35   margin-top: 20px;
36 }
37
38 p {
39   font-size: 1.2em;
40   font-weight: bold;
41   word-break: break-all;
42 }
```

```
1  .checkbox-container {
2      display: flex;
3      align-items: center;
4      margin: 5px 0;
5  }
6
7  .checkbox-container input[type="checkbox"] {
8      display: none;
9  }
10
11 .checkbox-container label {
12     display: flex;
13     align-items: center;
14     cursor: pointer;
15 }
16
17 .checkbox-container label:before {
18     content: '';
19     display: inline-block;
20     width: 20px;
21     height: 20px;
22     margin-right: 10px;
23     border: 2px solid #4CAF50;
24     border-radius: 3px;
25     background-color: white;
26     transition: background-color 0.3s;
27 }
28
29 .checkbox-container input[type="checkbox"]:checked + label:before {
30     background-color: #4CAF50;
31 }
```

En el archivo styles.css se colocan los estilos que tendrán las diferentes etiquetas html y secciones.

Backend



```
1 from flask import Flask, render_template, request
2 import random
3 import string
4
5 app = Flask(__name__)
```

Este código inicia un proyecto importando las librerías necesarias, incluyendo el framework Flask. Se importa Flask para crear la instancia de la aplicación, `render_template` para renderizar archivos HTML y `request` para manejar solicitudes HTTP.

Además, se importan las bibliotecas de Python *random* y *string*. El primero se utiliza para generar valores aleatorios, necesarios para la creación de contraseñas, mientras que *string* proporciona constantes y clases útiles para manipular cadenas de texto.

Finalmente, se crea una instancia de la aplicación Flask con `app = Flask(__name__)`, que establece la base de la aplicación web. Esta instancia permitirá definir rutas y gestionar el ciclo de vida de la aplicación.

```
1 def generate_password(length, use_uppercase, use_numbers, use_special):
2     characters = string.ascii_lowercase
3     password = []
4     # Agrega al menos un carácter de cada tipo seleccionado
5     if use_uppercase:
6         characters += string.ascii_uppercase
7         password.append(random.choice(string.ascii_uppercase))
8     if use_numbers:
9         characters += string.digits
10        password.append(random.choice(string.digits))
11    if use_special:
12        characters += string.punctuation
13        password.append(random.choice(string.punctuation))
14    # Genera el resto de la contraseña
15    while len(password) < length:
16        password.append(random.choice(characters))
17    # Mezcla los caracteres para evitar patrones predecibles
18    random.shuffle(password)
19    # Convierte la lista de caracteres en una cadena
20    password = ''.join(password)
21
22    return password
```

Se define la función `generate_password`, encargada de generar contraseñas. Esta función toma como argumentos el tamaño de la contraseña y las variables booleanas que indican si deben incluirse mayúsculas, números y caracteres especiales.

La función comienza definiendo un conjunto básico de caracteres (variable `characters`) que contiene solo letras minúsculas. Luego, se inicializa una lista vacía `password` para construir la contraseña.

La función asegura que al menos un carácter de cada tipo seleccionado se incluya en la contraseña. Si `use_uppercase` es verdadero, se añaden las letras mayúsculas al conjunto de caracteres permitidos y se agrega una mayúscula a la lista `password`. Similarmente, si `use_numbers` es verdadero, se añaden los dígitos y se agrega un dígito a la lista `password`. Para `use_special`, se añaden los caracteres especiales y se agrega uno a la lista `password`.

Luego de ello, se completa la contraseña generando caracteres aleatorios del conjunto permitido hasta alcanzar la longitud deseada.

Para evitar patrones predecibles, se mezcla la lista password aleatoriamente. Finalmente, la lista se convierte en una cadena de texto y se retorna como la contraseña generada.

```

1  @app.route('/', methods=['GET', 'POST'])
2  def index():
3      #Se inicializa la contraseña
4      password = ''
5      use_uppercase = use_numbers = True
6      use_special = False
7      #Se verifica si se ha enviado un formulario
8      if request.method == 'POST':
9          length = int(request.form['length'])#Se obtiene la longitud de la
10                                             #contraseña
11          #Se verifica si se han seleccionado los checkbox
12          use_uppercase = 'uppercase' in request.form
13          use_numbers = 'numbers' in request.form
14          use_special = 'special' in request.form
15          #Se llama a la función que genera la contraseña
16          password = generate_password(length, use_uppercase, use_numbers, use_special)
17
18      return render_template('index.html', password=password, use_uppercase=use_uppercase,
19                             use_numbers=use_numbers, use_special=use_special)#Se renderiza index.html
20                                     #con la contraseña generada
21  if __name__ == '__main__':
22      app.run()

```

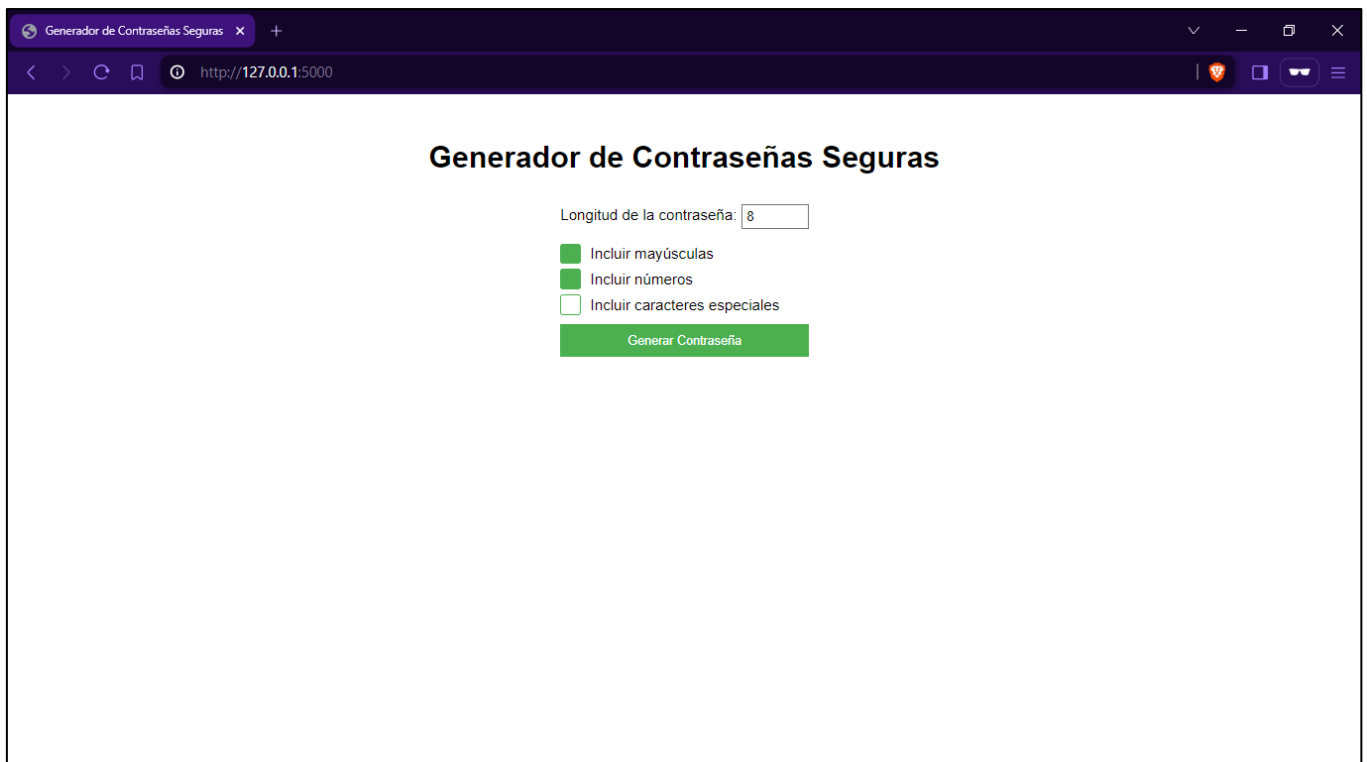
En esta parte se define la ruta principal de la aplicación Flask y maneja tanto solicitudes GET como POST. La función `index` se encarga de procesar el formulario para generar una contraseña.

Se inicializa la variable `password` como una cadena vacía, junto con las variables booleanas `use_uppercase` y `use_numbers` como `True` para que los checkbox estén marcados por defecto, igualmente la variable `use_special` como `False` para que no esté marcado (esto por el uso frecuente de los dos primeros tipos para contraseñas). Si se envía un formulario (método `POST`), se obtiene la longitud de la contraseña desde el formulario y se verifica si se han seleccionado las opciones de incluir mayúsculas, números y caracteres especiales.

Se llama a la función `generate_password` con los parámetros obtenidos para generar la contraseña. Finalmente, se renderiza la plantilla `index.html` y se pasa la contraseña generada para que se muestre en la página. La aplicación se ejecuta llamando a `app.run()` cuando el script se ejecuta directamente.

Resultados

Página por defecto.



Generador de Contraseñas Seguras

Longitud de la contraseña:

☒ Incluir mayúsculas
☒ Incluir números
☐ Incluir caracteres especiales

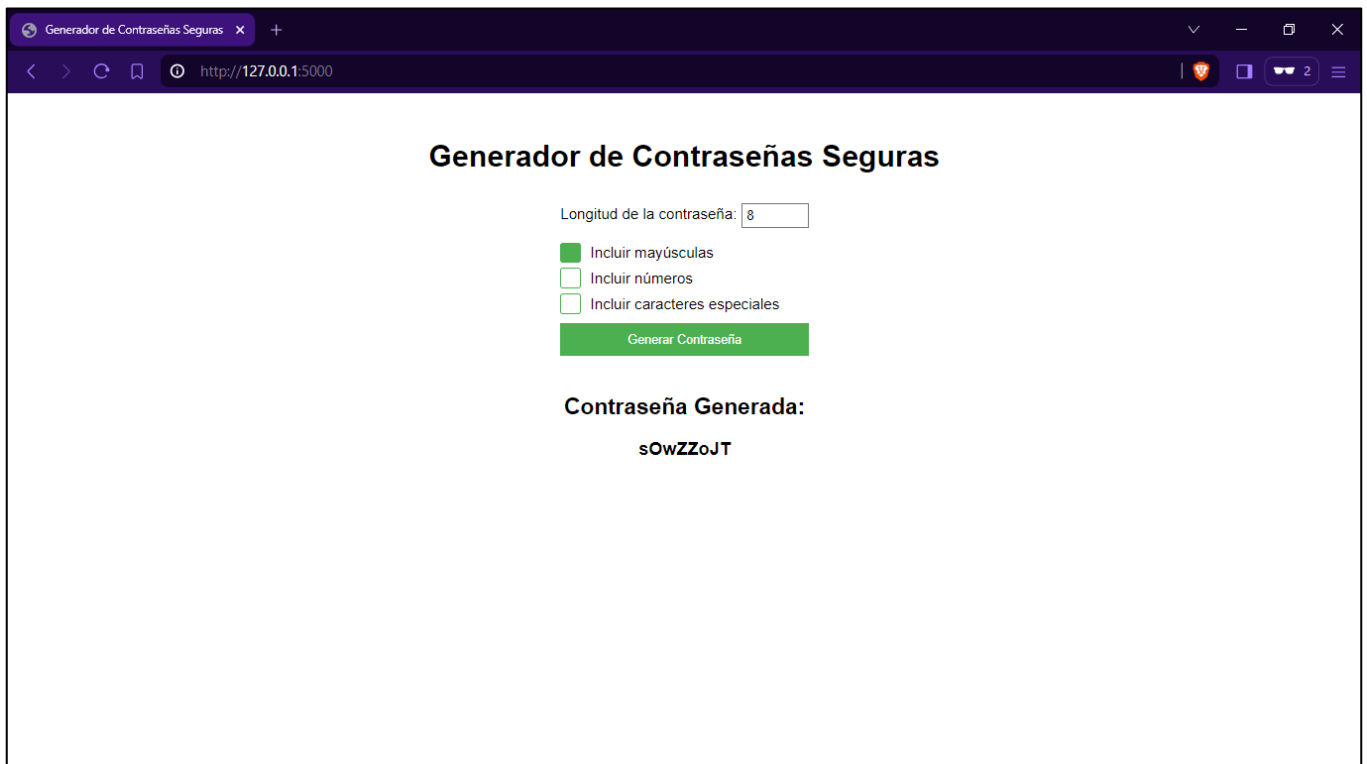
Resultados de una contraseña solo con minúsculas.



The screenshot shows a web browser window with the title 'Generador de Contraseñas Seguras'. The address bar shows 'http://127.0.0.1:5000'. The page content includes a form with the following elements:

- Longitud de la contraseña:
- ☐ Incluir mayúsculas
- ☐ Incluir números
- ☐ Incluir caracteres especiales
-
- Contraseña Generada:**
zgafizxw

Resultados de una contraseña con mayúsculas y minúsculas.



The screenshot shows the same web browser window as the previous one, but with the 'Incluir mayúsculas' checkbox checked. The generated password is now 'sOwZZoJT'.

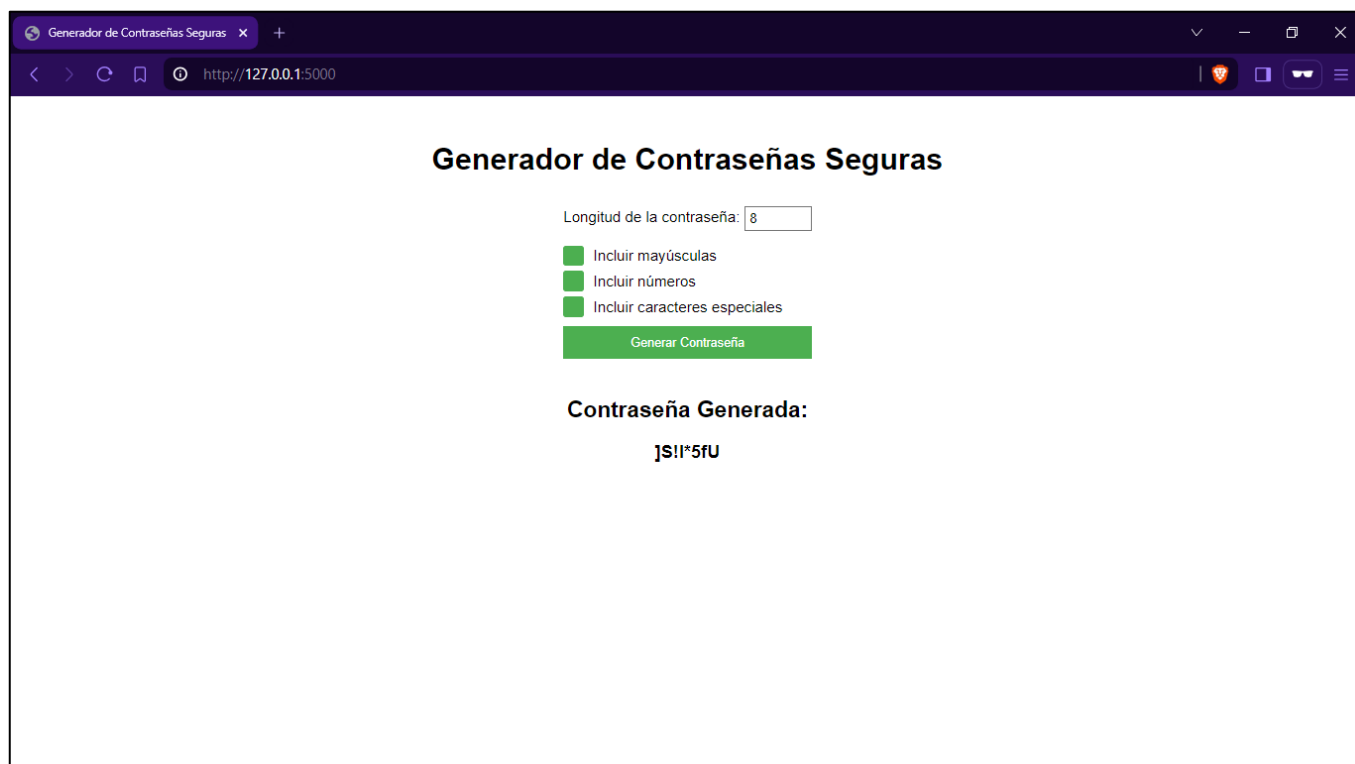
- Longitud de la contraseña:
- ☒ Incluir mayúsculas
- ☐ Incluir números
- ☐ Incluir caracteres especiales
-
- Contraseña Generada:**
sOwZZoJT

Resultados de una contraseña con mayúsculas, minúsculas y números.



The screenshot shows a web browser window with the title 'Generador de Contraseñas Seguras'. The address bar displays 'http://127.0.0.1:5000'. The page content includes a title 'Generador de Contraseñas Seguras', a label 'Longitud de la contraseña:' followed by a text input field containing '8', and three checkboxes: 'Incluir mayúsculas' (checked), 'Incluir números' (checked), and 'Incluir caracteres especiales' (unchecked). Below these is a green button labeled 'Generar Contraseña'. The generated password is displayed as 'Contraseña Generada: kAhFne98'.

Resultados de una contraseña con mayúsculas, minúsculas, números y caracteres especiales.



The screenshot shows the same web browser window as above, but with the 'Incluir caracteres especiales' checkbox now checked. The generated password is displayed as 'Contraseña Generada: j\$!l*5fU'.

Conclusiones

Los frameworks se han consolidado como herramientas esenciales en el desarrollo de software debido a su capacidad para estandarizar y agilizar el proceso de desarrollo. Facilitan la organización del código y la reutilización de componentes, lo que trae consigo un aumento de la eficiencia y una reducción en los costos de desarrollo. Entre las principales ventajas de utilizar frameworks destacan la estructuración predeterminada del código, la reutilización de este y la minimización de errores.

Sin embargo, los frameworks no están libres de desafíos, por ejemplo, el tiempo de aprendizaje puede ser considerable, sobre todo para aquellos que son nuevos en el framework o en el desarrollo de software en general. Y además de ello, tiene distintas desventajas como: versiones inestables, menor rendimiento, código sin utilizar, la elección del framework correcto, entre otros.

Y justamente en el último punto, la elección del framework, debe basarse en varios criterios, incluyendo la compatibilidad tecnológica, la seguridad, la comunidad de soporte y las experiencias previas de otros desarrolladores. Estos factores ayudan a asegurar que el framework elegido se adapte correctamente a las necesidades específicas del proyecto, optimizando el proceso de desarrollo y reduciendo riesgos.

Referencias

Amazon AWS (s.f.). *¿Qué es una interfaz de programación de aplicaciones (API)?*

Recuperado de: <https://aws.amazon.com/es/what-is/api/>

DECUBICA (s.f.). *¿Qué es un template y para qué sirve?* Recuperado de:

<https://www.decubica.com/disenio-web-usabilidad/que-es-un-template/>

Framework o librerías: ventajas y desventajas (2018). *¿Qué es un framework?* Recuperado

de: <https://www.tithink.com/es/2018/08/29/framework-o-librerias-ventajas-y-desventajas/>

GCFGGlobal (s.f.). *Plantillas web y frameworks*. Recuperado de:

<https://edu.gcfglobal.org/es/creacion-de-sitios-web/plantillas-web-y-frameworks/1/>

LetsTECH(2024). *¿Cómo elegir un Framework perfecto para tu proyecto?* Recuperado de:

<https://www.letstech.es/como-elegir-el-framework-perfecto-para-tu-proyecto/>

RedHat (2020). *¿Qué es una API y cómo funciona?* Recuperado de:

<https://www.redhat.com/es/topics/api/what-are-application-programming-interfaces>

Sommerville, I. (2011). *Software Engineering* (9th ed.). Addison-Wesley.