



ESPE

UNIVERSIDAD DE LAS FUERZAS ARMADAS
INNOVACIÓN PARA LA EXCELENCIA

DEPARTAMENTO CIENCIAS DE LA COMPUTACIÓN

ARQUITECTURA DE SOFTWARE

NRC: 3895

INTEGRANTES: GUERRA LUCIANA
TIPANGUANO SAMANTHA
CAICEDO GABRIEL

LABORATORIO 2

CONSUMO DE APIS

Generador de cuentos con conversión
a voz usando APIs Públicas

SANGOLQUÍ – ECUADOR

24-01-2025

RESUMEN

El presente informe detalla el desarrollo de una aplicación web que permite la generación de cuentos literarios personalizados y su conversión a voz en tiempo real, utilizando exclusivamente APIs públicas. Se han empleado herramientas como Next.js, Django, REST y Bootstrap, cada una desempeñando un papel clave en la optimización del sistema. Estas tecnologías han sido seleccionadas por su capacidad para mejorar la eficiencia en la construcción de sistemas web escalables, seguros y de alto rendimiento.

INTRODUCCIÓN

El avance de la inteligencia artificial y las tecnologías de conversión de texto a voz ha permitido la creación de experiencias interactivas en el ámbito educativo y literario. En este contexto, se ha desarrollado una aplicación web full stack que permite la generación de cuentos personalizados y su conversión a voz en tiempo real, aprovechando exclusivamente APIs públicas.

El propósito de este documento es analizar las herramientas utilizadas en el desarrollo de este proyecto y su contribución en la optimización de la experiencia del usuario y la funcionalidad del sistema.

OBJETIVOS

Objetivo General

Desarrollar una aplicación web interactiva que permita la generación de cuentos personalizados mediante inteligencia artificial generativa y su conversión a voz en tiempo real utilizando APIs públicas, con el fin de mejorar la accesibilidad y fomentar el aprendizaje interactivo.

Objetivos Específicos.

- Implementar una API pública de generación de texto que permita la creación de cuentos originales con opciones de personalización.
- Integrar una API pública de conversión de texto a voz para ofrecer narraciones interactivas en diferentes voces y estilos.
- Diseñar una interfaz web amigable y accesible que facilite la interacción del usuario con los cuentos generados y la narración en tiempo real.

- Optimizar el rendimiento y escalabilidad de la aplicación mediante el consumo eficiente de APIs y la gestión de solicitudes concurrentes.

METODOLOGÍA

Para evaluar el impacto y la funcionalidad de cada herramienta, se realizó una revisión exhaustiva de sus características y beneficios. Se utilizó la metodología Scrum, una metodología ágil que permite dividir el trabajo en iteraciones cortas llamadas sprints, facilitando la entrega progresiva de funcionalidades y la mejora continua del producto (Schwaber & Sutherland, 2020). Scrum se basa en eventos estructurados, artefactos bien definidos y roles específicos que permiten maximizar la productividad y adaptabilidad del desarrollo.

Implementación de Scrum

Dado el tiempo limitado de **tres días** para la realización del taller, se estableció un plan de trabajo estructurado en fases cortas y reuniones de seguimiento. Scrum permite una adaptación dinámica a los requerimientos del proyecto, fomentando la colaboración y la entrega incremental del software. Para ello, se utilizaron los siguientes eventos clave:

- **Sprint Planning:** Definición de tareas y objetivos específicos para cada día de desarrollo.
- **Daily Scrum:** Reuniones diarias para revisar el progreso y ajustar tareas.
- **Sprint Review:** Evaluación del trabajo realizado y retroalimentación al final de cada jornada.
- **Sprint Retrospective:** Identificación de mejoras en el proceso para futuras iteraciones.

Plan de trabajo en tres días:

- **Día 1:**
 - Configuración del entorno de desarrollo.
 - Implementación del backend con Django y API REST.
 - Desarrollo inicial del frontend con Next.js y Bootstrap.
 - Primera reunión de seguimiento para evaluar avances.
- **Día 2:**
 - Integración de los módulos backend y frontend.
 - Implementación de estilos con Bootstrap y ajustes en UI/UX.
 - Pruebas iniciales de funcionalidad.
 - Segunda reunión de seguimiento y ajuste de tareas pendientes.
- **Día 3:**
 - Pruebas finales y depuración de errores.

- Documentación del proyecto y generación de informe.
- Presentación y demostración del sistema.

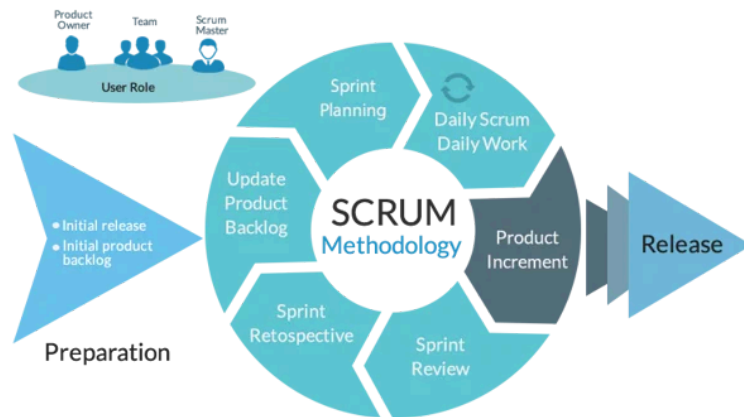


Figura 1. Proceso de metodología Scrum.

DESARROLLO

Arquitectura del sistema

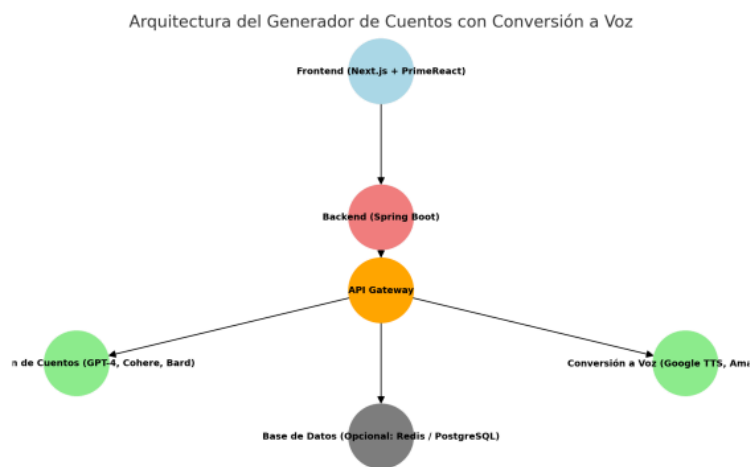


Figura 2. Diagrama de Arquitectura del Sistema.

Herramientas Utilizadas

1. **Next.js** : Next.js es un framework de React que permite construir aplicaciones web modernas, optimizadas y de alto rendimiento. Proporciona una estructura de desarrollo eficiente y herramientas avanzadas para la generación de páginas estáticas (SSG), renderizado del lado del servidor (SSR), y creación de APIs dentro del mismo proyecto.

Características:

- **Renderizado Híbrido:** Soporta SSR (Server-Side Rendering) y SSG (Static Site Generation) según la necesidad de cada página.
 - **Optimización Automática:** Maneja el código dividido (Code Splitting) para mejorar la carga de la aplicación.
 - **API Routes:** Permite crear endpoints en el mismo proyecto, evitando la necesidad de un backend separado.
 - **Soporte para TypeScript:** Permite trabajar con JavaScript o TypeScript sin configuración extra.
 - **Soporte para CSS y Tailwind CSS:** Permite usar estilos globales, módulos CSS, y frameworks como Tailwind CSS fácilmente.
2. **Django:** Django es un framework de desarrollo web en Python diseñado para construir aplicaciones robustas, seguras y escalables de manera rápida y eficiente. Sigue el principio “Don’t Repeat Yourself” (DRY), promoviendo el desarrollo limpio y reutilizable de código.

Características:

- **Arquitectura MVC/MVT:** Se basa en el patrón Modelo-Vista-Controlador (MVC), aunque en Django se llama Modelo-Vista-Template (MVT).
 - **ORM Poderoso:** Usa un Mapeo Objeto-Relacional (ORM) para manejar bases de datos sin escribir SQL manualmente.
 - **Seguridad Incorporada:** Protege contra ataques CSRF, XSS, inyección SQL y clickjacking.
 - **Admin Automático:** Incluye un panel de administración listo para usar basado en modelos de la base de datos.
 - **Desarrollo Rápido:** Permite crear aplicaciones funcionales en minutos, con una estructura clara.
3. **REST:** REST (Representational State Transfer) es un estilo de arquitectura para el diseño de servicios web. Define un conjunto de principios y restricciones que permiten a los sistemas comunicarse de manera eficiente utilizando HTTP como protocolo de transporte.

Características:

- **Basado en Recursos:** Todo en REST se trata como un recurso, identificado por una URL única.
- **Usa Métodos HTTP:** Utiliza los métodos estándar de HTTP (GET, POST, PUT, DELETE, etc.).

- Interfaz Uniforme: Define una estructura común para interactuar con la API.
- Sin Estado (Stateless): Cada solicitud HTTP debe contener toda la información necesaria para ser procesada.
- Soporta Representaciones Múltiples: Los recursos pueden ser representados en diferentes formatos (JSON, XML, HTML, etc.).

4. **BootStrap:** Bootstrap es un framework de código abierto desarrollado por Twitter que permite diseñar sitios web y aplicaciones responsivas de manera rápida y sencilla. Utiliza una combinación de HTML, CSS y JavaScript para proporcionar una colección de componentes pre-diseñados, como botones, formularios, tarjetas y sistemas de navegación.

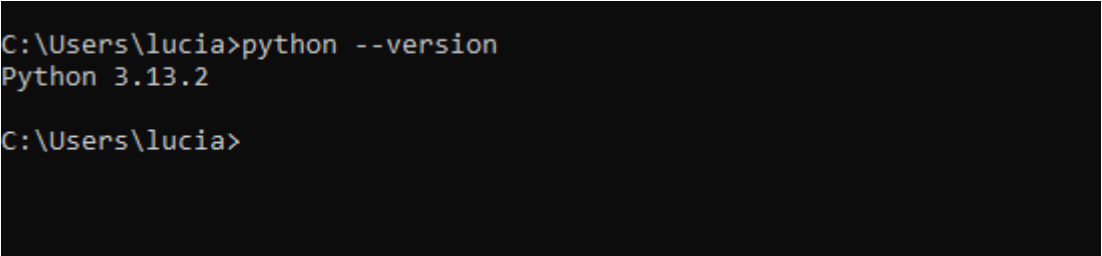
Características:

- Diseño Responsivo: Usa un sistema de grid flexible basado en columnas para adaptarse a cualquier pantalla.
- Componentes Listos para Usar: Botones, formularios, menús de navegación, modales, alertas y más.
- Compatibilidad con Navegadores: Funciona en Chrome, Firefox, Safari, Edge y otros.

DESARROLLO:

Instalación de Dependencias:

- Instalación de Python



```
C:\Users\lucia>python --version
Python 3.13.2

C:\Users\lucia>
```

- Creación de un entorno virtual

python -m venv venv

venv\Scripts\activate

```
C:\Users\lucia>python -m venv venv

C:\Users\lucia>venv\Scripts\activate

(venv) C:\Users\lucia>
```

- Instalación de Django

```
Símbolo del sistema
13_qbz5n2kfra8p0\LocalCache\local-packages\Python313\Scripts' which is not on PATH.
Consider adding this directory to PATH or, if you prefer to suppress this warning, use --no-warn-script-location.
WARNING: The script distro.exe is installed in 'C:\Users\lucia\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.13_qbz5n2kfra8p0\LocalCache\local-packages\Python313\Scripts' which is not on PATH.
Consider adding this directory to PATH or, if you prefer to suppress this warning, use --no-warn-script-location.
WARNING: The script normalizer.exe is installed in 'C:\Users\lucia\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.13_qbz5n2kfra8p0\LocalCache\local-packages\Python313\Scripts' which is not on PATH.
Consider adding this directory to PATH or, if you prefer to suppress this warning, use --no-warn-script-location.
WARNING: The scripts pyrsa-decrypt.exe, pyrsa-encrypt.exe, pyrsa-keygen.exe, pyrsa-priv2pub.exe, pyrsa-sign.exe and pyrsa-verify.exe are installed in 'C:\Users\lucia\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.13_qbz5n2kfra8p0\LocalCache\local-packages\Python313\Scripts' which is not on PATH.
Consider adding this directory to PATH or, if you prefer to suppress this warning, use --no-warn-script-location.
WARNING: The script httpx.exe is installed in 'C:\Users\lucia\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.13_qbz5n2kfra8p0\LocalCache\local-packages\Python313\Scripts' which is not on PATH.
Consider adding this directory to PATH or, if you prefer to suppress this warning, use --no-warn-script-location.
WARNING: The script openai.exe is installed in 'C:\Users\lucia\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.13_qbz5n2kfra8p0\LocalCache\local-packages\Python313\Scripts' which is not on PATH.
Consider adding this directory to PATH or, if you prefer to suppress this warning, use --no-warn-script-location.
Successfully installed annotated-types-0.7.0 anyio-4.8.0 cachetools-5.5.1 certifi-2025.1.31 charset-normalizer-3.4.1 distro-1.9.0 django-cors-headers-4.7.0 django-rest-framework-3.15.2 django-rest-framework-simplejwt-5.4.0 google-api-core-2.24.1 google-auth-2.38.0 google-cloud-texttospeech-2.24.0 googleapis-common-protos-1.66.0 grpcio-1.70.0 grpcio-status-1.70.0 h11-0.14.0 httpcore-1.0.7 httpx-0.28.1 idna-3.10 jiter-0.8.2 openai-1.61.1 proto-plus-1.26.0 protobuf-5.29.3 pyasn1-0.6.1 pyasn1-modules-0.4.1 pydantic-2.10.6 pydantic-core-2.27.2 pyjwt-2.10.1 requests-2.32.3 rsa-4.9 sniffio-1.3.1 tqdm-4.67.1 urllib3-2.3.0

[notice] A new release of pip is available: 24.3.1 -> 25.0
[notice] To update, run: C:\Users\lucia\AppData\Local\Microsoft\WindowsApps\PythonSoftwareFoundation.Python.3.13_qbz5n2kfra8p0\python.exe -m pip install --upgrade pip
```

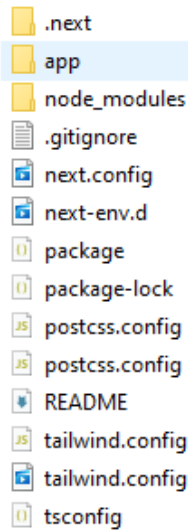
- Creación del Proyecto en Django

```
(venv) C:\Users\lucia>django-admin startproject backend
```

Estructura de la Carpeta Creada.

- Nombre
- api
 - backend
 - venv
 - .env
 - db.sqlite3
 - manage

Estructura de la Carpeta de Front



- .next
- app
- node_modules
- .gitignore
- next.config
- next-env.d
- package
- package-lock
- postcss.config
- postcss.config
- README
- tailwind.config
- tailwind.config
- tsconfig

Archivo setting:

```
settings.py x
Lab2 > Back > backend > settings.py > ...
1  from pathlib import Path
2  from datetime import timedelta
3  from dotenv import load_dotenv
4  import os
5
6  # Obtener la ruta base del proyecto
7  BASE_DIR = Path(__file__).resolve().parent.parent
8  dotenv_path = os.path.join(BASE_DIR, ".env")
9
10 # Forzar la carga manual del archivo `.env`
11 if os.path.exists(dotenv_path):
12     print(f"Cargando variables de entorno desde: {dotenv_path}") # ◆ Mensaje de depuración
13     load_dotenv(dotenv_path=dotenv_path, override=True) # ◆ Cargar variables de entorno
14 else:
15     print("⚠ ERROR: No se encontró el archivo .env en la raíz del backend.")
16
17 # Configuración de seguridad
18 SECRET_KEY = os.getenv("SECRET_KEY", "django-insecure-default-key")
19 DEBUG = os.getenv("DEBUG", "True") == "True"
20 ALLOWED_HOSTS = os.getenv("ALLOWED_HOSTS", "localhost,127.0.0.1").split(",")
21
22 # Configuración de Google Gemini API
23 GEMINI_API_KEY = os.getenv("GEMINI_API_KEY")
24 if not GEMINI_API_KEY:
25     raise ValueError("⚠ ERROR: La API Key de Google Gemini no está configurada. Revisa el archivo .env.")
26
27 print("◆ API Key de Gemini cargada correctamente:", GEMINI_API_KEY[:10] + "*****") # ◆ Debugging
28
29 # Google Cloud Credentials (Opcional)
30 GOOGLE_APPLICATION_CREDENTIALS = os.getenv("GOOGLE_APPLICATION_CREDENTIALS")
31
```


Clave de Api Bard

Google AI Studio

Obtén una clave de API

Get API key

Create Prompt

Stream Realtime

Starter Apps

Tune a Model

Library

Enable chat history

Prompt Gallery

API documentation

Developer forum

Changelog

Settings

-H 'Content-Type: application/json' \

-X POST \

-d '{

"contents": [{

"parts":[{"text": "Explain how AI works"}]

}]

}

'

Use code with caution.

Crear clave de API

Tus claves de API se enumeran a continuación. También puedes ver y administrar tu proyecto y tus claves de API en Google Cloud.

| Número del proyecto | Nombre del proyecto | Clave de API | Fecha de creación | Plan |
|---------------------|---------------------|--------------|-------------------|---|
| ...4426 | Gemini API | ...reHI | 10 feb 2025 | Sin costo Configurar facturación Ver datos de uso |

Recuerda usar las claves de API de forma segura. No las compartas ni las incorpores en código público. El uso de la API de Gemini desde un proyecto con la facturación habilitada está sujeto a los precios de pago por uso.

Activar Windows
Ve a Configuración para activar Windows.

Archivo urls.py

```
b2 > Back > backend > urls.py > ...
1  """
2  URL configuration for backend project.
3
4  The `urlpatterns` list routes URLs to views. For more information please see:
5  |   https://docs.djangoproject.com/en/5.1/topics/http/urls/
6  |   Examples:
7  |   Function views
8  |       1. Add an import:  from my_app import views
9  |       2. Add a URL to urlpatterns:  path('', views.home, name='home')
10 |   Class-based views
11 |       1. Add an import:  from other_app.views import Home
12 |       2. Add a URL to urlpatterns:  path('', Home.as_view(), name='home')
13 |   Including another URLconf
14 |       1. Import the include() function: from django.urls import include, path
15 |       2. Add a URL to urlpatterns:  path('blog/', include('blog.urls'))
16 |   """
17
18
19 from django.contrib import admin
20 from django.urls import path, include
21
22 urlpatterns = [
23     path('admin/', admin.site.urls),
24     path('api/', include('api.urls')), # Incluye las rutas de la app `api`
25 ]
```

Front: Page.tsx

```
page.tsx X
Lab2 > Front > app > page.tsx
1  | "use client";
2
3  | import { useState } from "react";
4  | import axios from "axios";
5
6  | export default function Home() {
7  |   const [prompt, setPrompt] = useState("");
8  |   const [story, setStory] = useState("");
9  |   const [audio, setAudio] = useState<string | null>(null);
10 |   const [loading, setLoading] = useState(false);
11 |   const [loadingAudio, setLoadingAudio] = useState(false);
12
13 |   // Función para generar el cuento
14 |   const generateStory = async () => {
15 |     setLoading(true);
16 |     try {
17 |       const response = await axios.post(`${process.env.BACKEND_URL}/api/generate/`, {
18 |         prompt,
19 |       });
20 |       setStory(response.data.story);
21 |       setAudio(null);
22 |     } catch (error) {
23 |       console.error("Error al generar la historia:", error);
24 |     }
25 |     setLoading(false);
26 |   };
27
28 |   // Función para convertir en voz
29 |   const convertToSpeech = async () => {
30 |     if (!story) return;
31
32 |     setLoadingAudio(true);
33 |   };
34 | }
```

Comando de ejecución del Back:

```
[notice] To update, run: C:\Users\lucia\AppData\Local\Microsoft\WindowsApps\PythonSoftwareF...
fra8p0\python.exe -m pip install --upgrade pip

(venv) C:\Users\lucia\Downloads\Lab2\Lab2\Back>
(venv) C:\Users\lucia\Downloads\Lab2\Lab2\Back>python manage.py runserver
Cargando variables de entorno desde: C:\Users\lucia\Downloads\Lab2\Lab2\Back\.env
[+] API Key de Gemini cargada correctamente: AIzaSyC0PM*****
[+] API Key de Gemini cargada correctamente: AIzaSyC0PM*****
Cargando variables de entorno desde: C:\Users\lucia\Downloads\Lab2\Lab2\Back\.env
[+] API Key de Gemini cargada correctamente: AIzaSyC0PM*****
[+] API Key de Gemini cargada correctamente: AIzaSyC0PM*****
Watching for file changes with StatReloader
Performing system checks...
```

Comando de ejecución de Front

```
added 175 packages, and audited 176 packages in 26s
40 packages are looking for funding
  run `npm fund` for details
found 0 vulnerabilities

C:\Users\lucia\Downloads\Lab2\Lab2\Front>npm run dev

> frontend@0.1.0 dev
> next dev --turbo

  ▲ Next.js 15.1.6 (Turbopack)
  - Local:      http://localhost:3000
  - Network:    http://10.40.38.199:3000

  Starting...
  Ready in 3.3s
```

RESULTADOS



CONCLUSIONES

El uso de herramientas especializadas en el desarrollo web permite optimizar procesos y mejorar la calidad del software. Next.js ofrece un entorno eficiente para la construcción de interfaces dinámicas, mientras que Django facilita la gestión de bases de datos y la seguridad de las aplicaciones. REST permite estructurar APIs escalables y eficientes, esenciales en aplicaciones modernas. Finalmente, Bootstrap agiliza el diseño responsivo y mejora la experiencia del usuario.

La combinación de estas tecnologías ha permitido construir un sistema que no solo genera cuentos personalizados, sino que también convierte el texto en audio en tiempo real, mejorando la

accesibilidad y fomentando el aprendizaje interactivo. La adopción de estas herramientas ha sido clave para garantizar la eficiencia y éxito del proyecto tecnológico.

REFERENCIAS

1. Next.js Documentation. (2024). Next.js - The React Framework. Recuperado de: <https://nextjs.org/docs>
2. Django Project. (2024). Django - The Web Framework for Perfectionists with Deadlines. Recuperado de: <https://www.djangoproject.com/>
3. Fielding, R. T. (2000). Architectural Styles and the Design of Network-based Software Architectures. University of California, Irvine.
4. Bootstrap Documentation. (2024). Bootstrap - Build fast, responsive sites. Recuperado de: <https://getbootstrap.com/docs/>
5. OpenAI API Documentation. (2024). GPT-4 and Text-to-Speech API. Recuperado de: <https://platform.openai.com/docs>.