

**Autor**: Alain Daniel Rodríguez Domínguez

**Tutores**: Lic. Alejandro Cespón

Orlando M. Saucedo

**Sistema para el pronóstico de la enfermedad fúngica mancha de asfalto (Phyllachora maydis) en el cultivo del maíz basado en observaciones climáticas.**

**Ciencia de la Computación**

**Departamento de Computación**

, Cuba, junio 2019

**Licenciatura en Ciencia de la Computación**

Autor: Roberto Carlos Vázquez Basulto

**BLOQUE PARA EL RECONOCIMIENTO DE EMOCIONES FACIALES DE LOS ESTUDIANTES EN MOODLE**

, Cuba, junio 2019



**Author**: Alain Daniel Rodríguez Domínguez

**Tutor**: B. A. Alejandro Cespón

Orlando M. Saucedo

**Bachelor in Computer Science**

**Informatics Department**

**System for the forecasting of the tar spot fungal disease (Phyllachora maydis) in corn growing, based on weather observations.**

Este documento es Propiedad Patrimonial de la Universidad Central “Marta Abreu” de Las Villas, y se encuentra depositado en los fondos de la Biblioteca Universitaria “Chiqui Gómez Lubian” subordinada a la Dirección de Información Científico Técnica de la mencionada casa de altos estudios.

Se autoriza su utilización bajo la licencia siguiente:

**Atribución- No Comercial- Compartir Igual**



Para cualquier información contacte con:

Dirección de Información Científico Técnica. Universidad Central “Marta Abreu” de Las Villas. Carretera a Camajuaní. Km 5½. Santa Clara. Villa Clara. Cuba. CP. 54 830

Teléfonos.: +53 01 42281503-14190

2024.



Hago constar que el presente trabajo fue realizado en la Universidad Central “Marta Abreu” de Las Villas como parte de la culminación de los estudios de la especialidad de Ciencia de la Computación, autorizando a que el mismo sea utilizado por la institución, para los fines que estime conveniente, tanto de forma parcial como total y que además no podrá ser presentado en eventos ni publicado sin la autorización de la Universidad.

\_\_\_\_\_\_\_\_\_\_\_\_\_

Firma del autor

Los abajo firmantes, certificamos que el presente trabajo ha sido realizado según acuerdos de la dirección de nuestro centro y el mismo cumple con los requisitos que debe tener un trabajo de esta envergadura referido a la temática señalada.

\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Firma del tutor Firma del tutor Firma del jefe del Dpto

# AGRADECIMIENTOS

# resumen

La Facultad de Ciencias Agropecuarias de la Universidad Central de Las Villas “Marta Abreu” requiere de un sistema que apoye la investigación de la incidencia de la enfermedad fúngica mancha de asfalto (*Phyllachora maydis*) en el cultivo del maíz (*Zea mays*) mediante el registro y análisis de observaciones meteorológicas y el pronóstico de la afectación basado en dichas mediciones. Para ello se propone e implementa una aplicación web construida con los *framewoks* Next.js (que se sustenta en React, una biblioteca de JavaScript) para el contenido y la funcionalidad, y Tailwind (basado en el lenguaje CSS) para la estilización en el *front-end*, y con Django y Django REST Framework (que usan el lenguaje de programación Python) en el *back-end*, así como un servidor PostgreSQL para la gestión de la base de datos. Se implementó además un sistema de credenciales para la autenticación de los usuarios autorizados a modificar la información manejada. La aplicación web resultante permite la introducción, recuperación y eliminación de datos sobre las observaciones, las estaciones meteorológicas y las unidades de cultivo del maíz. Posibilita la emisión automática de pronósticos de advertencia cuando el riesgo de aparición y desarrollo de la enfermedad en los sembrados monitoreados lo haga necesario.

**Palabras clave:** maíz, *Phyllachora maydis*, aplicación web, Django, Next.js

# abstract

The Faculty of Agropecuary Sciences of the Central University of Las Villas “Marta Abreu” needs a system to support the investigation of the incidence of the tar spot fungal disease (*Phyllachora maydis*) in the growing of corn (*Zea mays*) through the recording and analysis of weather observations and the forecasting of possible affectations on the basis of those records. To that effect, the author implements and proposes a web application built with the framewoks Next.js (which is built on the JavaScript library React) for content and functionality, and Tailwind (based on the CSS language) for styling in the front-end, and with Django and Django REST Framework (which use programming language Python) in the back-end. Furthermore, a PostgreSQL server was used for database management and a system of credentials was implemented for the authentication of the users that are allowed to make changes to the information in the system. The resulting web application makes it possible to input, retrieve and delete data corresponding to weather observations, meteorological stations and corn-growing units. It is able to issue warnings when the estimated risk of presence and development of the fungal disease in the investigated corn fields becomes high.

**Key words:** corn, *Phyllachora maydis*, web application, Django, Next.js

# contenidos

**Lista de figuras:**

**Lista de tablas:**

# introducción

El maíz (de nombre científico *Zea mays*) se destaca como un cultivo de interés para pequeños, medianos y grandes productores debido a su versatilidad y al aprovechamiento de su cosecha en diversos ámbitos, incluyendo su creciente uso como cobertura vegetal de suelo en países tropicales y subtropicales. Esta tendencia refleja la importancia que el maíz ha adquirido en la agricultura moderna, no solo como fuente de alimento, sino también como recurso para mejorar la salud del suelo y promover prácticas agrícolas sostenibles. Su capacidad para adaptarse a diferentes condiciones climáticas y su considerable papel en la seguridad alimentaria lo hacen un cultivo esencial en la producción agrícola a nivel mundial (Valdez Ocampo et al., 2024).

Estos hechos justifican el interés de los investigadores en la protección de la salud de los cultivos de maíz, pues esta planta no está exenta del peligro de amenazas en el ámbito fitosanitario. Las plagas tienen el potencial de disminuir sustancialmente el volumen y/o la calidad del maíz producido, así como la capacidad de la planta para realizar su función de cobertura vegetal; por tanto, los estudios dirigidos a entender mejor las enfermedades que afectan al maíz, para poder así combatirlas mejor, son una necesidad para la correcta explotación de estos cultivos. Entre las más importantes de ellas están el carbón común (*Ustilago maydis*) que afecta principalmente a la mazorca; el carbón de la espiga (*Sporisorium reilianum*), que afecta la espiga y la mazorca; la roya común (*Puccinia sorghi*) y el complejo mancha de asfalto (Díaz-Morales et al., 2018).

Esta última es provocada por el hongo ascomyceto *Phyllachora maydis*. La presencia de este reduce el rendimiento y en algunos casos puede llegar a causar la muerte de plantas de maíz. A nivel de América latina se han documentado pérdidas de rendimiento de hasta el 46%. La mancha de asfalto se caracteriza por la presencia de estromas negros en las hojas. Estas manchas son producto de los cuerpos fructíferos del hongo, un biótrofo obligado que comienza infectando la parte baja de la planta, aunque puede infectar vainas o mazorcas. Una vez establecido, se distribuye a través de la hoja provocando manchas que llegan a unirse, causando posteriormente necrosis, senescencia prematura y muerte (Vinces-Tachong et al., 2022).

La afectación de las plantas de maíz por la mancha de asfalto está muy relacionada con las condiciones climáticas en las que se encuentre. La velocidad del viento que dispersa las ascosporas, los valores entre los cuales varían la temperatura y la humedad relativa, etcétera, son factores que influyen en el riesgo de que la enfermedad incida en los cultivos, razón por la cual el registro diario de observaciones meteorológicas es una parte clave del estudio de la misma (European and Mediterranean Plant Protection Organization, n.d.).

*Phyllachora maydis* fue descrito inicialmente en 1904 en México e incide en diversos países de América Central y Suramérica. En 2015 fue registrado por primera vez en los Estados Unidos, en Illinois e Indiana, y después se propagó a otros estados (European and Mediterranean Plant Protection Organization, n.d.).

Esta enfermedad del maíz resulta de interés a la Facultad de Ciencias Agropecuarias de la Universidad Central de Las Villas “Marta Abreu”. Sin embargo, todo estudio moderno, en cualquier campo de investigación, debe auxiliarse de las tecnologías de la información recientes, y los estudios fitosanitarios del maíz no son una excepción, sino que ya existen precedentes para ello, como el trabajo de (Vinces-Tachong et al., 2022), que describe la implementación del procesamiento de imágenes para la evaluación de la presencia de la misma enfermedad que concierne a la presente investigación. El uso de soportes digitales aventaja notablemente a los métodos tradicionales de manejo de datos basados en papel manuscrito o impreso, ya que disminuye los riesgos de redundancia de la información, aumenta abrumadoramente la facilidad de copiado y modificación de la misma, reduce notablemente el volumen del material requerido para almacenarla, prescinde de los costos continuos de nuevas hojas y tinta y, además de todo lo anterior, contribuye al prestigio de la universidad como centro interesado por la modernidad tecnológica y la digitalización. Adicionalmente, la automatización del procesamiento de datos para realizar predicciones sobre la incidencia de la enfermedad ahorraría esfuerzo humano experto en el pronóstico de las afectaciones de la misma. Por otra parte, de estar asentado en la web (*online*), el sistema que dé respuesta a esta necesidad podrá eliminar las distancias entre los expertos, ya que sería accesible desde cualquier computadora o dispositivo móvil conectado a Internet, sin importar dónde se encuentre. Estas técnicas se perfilan en el panorama de la tecnología como un frente de innovación; el desarrollo de soluciones web avanza a un ritmo impresionante, impulsado por una alta demanda y sustentado por multitud de opciones y herramientas en constante evolución que compiten entre sí y se ofrecen como alternativas efectivas para el desarrollador y le permiten satisfacer cada vez mejor las necesidades de los usuarios finales (Kralina and Popova, 2024). Todo ello constituye la **justificación de la investigación** realizada por el autor.

De las consideraciones anteriores se deduce el siguiente **problema de investigación:** ¿Cómo desarrollar un sistema informático que auxilie a los investigadores asociados a la Facultad de Ciencias Agropecuarias de la mencionada Universidad en el manejo de datos relevantes para el estudio de la incidencia de la mancha de asfalto en los cultivos de maíz?

Para abordar de manera efectiva esta problemática se plantea tres **preguntas de investigación** esenciales:

1. ¿Qué requisitos debe cumplir el sistema a desarrollar?
2. ¿Cuáles son las mejores prácticas y tecnologías actuales para el desarrollo de sistemas informáticos basados en actividad en línea (aplicaciones web) para el manejo de datos?
3. ¿Cómo implementar una aplicación web que responda a las necesidades planteadas por el problema que motiva su realización?
4. ¿Cómo validar la propuesta técnica resultante?

Considerando el problema y las preguntas de investigación formuladas, se proponen los siguientes objetivos generales y específicos:

**Objetivo general:** desarrollar una aplicación web que permita a los investigadores asociados a la Facultad de Ciencias Agropecuarias el manejo de datos de observaciones relacionadas con la incidencia de la enfermedad fúngica mancha de asfalto en el cultivo de maíz en las unidades estudiadas, y emita pronósticos automáticos de riesgo de afectación por dicha enfermedad.

**Objetivos específicos:**

1. Definir los requisitos que debe cumplir el sistema.
2. Asimilar prácticas y tecnologías actuales idóneas para el desarrollo de aplicaciones web de manejo de datos.
3. Implementar la aplicación web que cumpla con los requisitos determinados.
4. Realizar pruebas de software a la aplicación realizada.

**Estructura del trabajo:** el presente informe se organiza en una introducción, tres capítulos, conclusiones, recomendaciones y referencias bibliográficas.

En el capítulo 1 se aborda los conceptos fundamentales del negocio y los requisitos del sistema, así como las tecnologías relacionadas con el desarrollo de aplicaciones web modernas en el entorno de la solución del problema.

En el capítulo 2 se describe el análisis técnico y los elementos más relevantes de la implementación.

En el capítulo 3 se describen las técnicas de pruebas y las pruebas realizadas al software.

# capítulo 1. marco teórico y conceptual

El presente capítulo cubre los fundamentos técnicos del problema cuya solución se persigue (la investigación sobre la incidencia de la enfermedad fúngica mancha de asfalto y su pronóstico basado en condiciones meteorológicas) relevantes al desarrollo del sistema propuesto, así como los conceptos, enfoques, tecnologías y herramientas empleadas para la implementación de este.

## Conceptos y requisitos del negocio

## Aplicaciones web

Las aplicaciones web son sistemas accesibles mediante un navegador a través de una red local o Internet, utilizados para la obtención y el procesamiento de datos, que dependen de un servidor físico o virtual *online* para el cumplimiento de sus funcionalidades (Hernández Cruz et al., 2023; Llamuca-Quinaloa et al., 2021).

El campo de desarrollo de aplicaciones web ha experimentado una evolución pronunciadamente rápida desde su nacimiento. En contraste con los sitios web o *websites* estáticos de los días pasados del Internet, que ofrecían contenido sin características interactivas, las *web apps* de hoy son integrales en experiencia digital, brindando servicios dinámicos e inmersivos que compiten con los tradicionales programas de escritorio (Panwar, 2024).

### Ventajas y clasificación

Una aplicación web disfruta de ciertas ventajas sobre otras clases de sistemas. El mismo hecho de residir en un servidor remoto alivia al dispositivo del usuario de las cargas de almacenamiento y procesamiento que tendría que asumir el ordenador en el caso de una aplicación de escritorio ubicada localmente. Además, ofrece una flexibilidad geográfica y facilidades de sincronización superiores, pues un usuario puede acceder a sus servicios desde cualquier lugar con acceso libre al Internet, y si emplea diferentes dispositivos no necesita ocuparse de mantener la consistencia de la información guardada en ambos, pues el servidor al que accede mantiene sus datos de aplicación en un respaldo unificado. Por la parte del desarrollador, las aplicaciones web son altamente compatibles y su mantenimiento y actualización son simples, ya que requieren un único desarrollo para funcionar en diferentes sistemas operativos y no hay conflictos de versiones, dado que todos los usuarios utilizan la misma versión, evitando así problemas de incompatibilidad (Sotolongo Martínez, n.d.). Todas estas características favorables respaldan la elección de una aplicación web como el tipo de sistema a implementar para dar solución al problema abordado por el presente trabajo.

Independientemente de las características esenciales que comparten, las aplicaciones web pueden ser asociadas a diferentes clases. Las tres principales son *Single Page Application (SPA)* o aplicaciones de una sola página, *Multi-Page Application (MPA)* o aplicaciones de múltiples páginas y *Progressive Web Application (PWA)* o aplicaciones web progresivas.

En las SPA la transición entre vistas de la aplicación web no provoca una nueva carga de la página completa, lo cual elimina la necesidad de cargar cada subpágina desde el servidor. Esto se debe a que la SPA carga un único archivo HTML (véase el epígrafe 1.4.1) y después actualiza el contenido de este documento según se requiera. Todo el contenido de la página es cargado simultáneamente, mientras que los datos necesarios son renderizados del lado del cliente (véase el epígrafe 1.2.2), lo cual favorece el rendimiento de la página y ofrece una mejor experiencia de usuario a lo largo de su interacción con la aplicación; sin embargo, la carga inicial toma más tiempo, pues se debe tomar toda la página de la primera vez.

Las MPA representan un enfoque más tradicional, que se basa en la generación dinámica de múltiples archivos HTML generados dinámicamente del lado del servidor. Cada transición entre subpáginas requiere una nueva solicitud de página para tomar el archivo HTML del servidor, lo cual, al contrario de lo que sucede en las SPA, causa demoras al cargar y mostrar el contenido durante la interacción con el usuario (Kowalczyk and Szandala, 2024).

Las PWA buscan que los usuarios de la aplicación web tengan una experiencia similar a la de una aplicación nativa. Ofrecen funcionalidades que combinan el acceso al servidor con el soporte *offline*; se pueden instalar en el dispositivo del usuario y hasta cierto punto se pueden usar sin conexión a Internet. También pueden incluir notificaciones emergentes y acceso al hardware del dispositivo. Este enfoque fue desarrollado por Google y otros actores del campo tecnológico para simplificar la creación de aplicaciones multi-plataforma, y es una opción adecuada cuando se desee crear aplicaciones web que se sientan como aplicaciones nativas pero que sean más fáciles de desarrollar y mantener que las aplicaciones nativas tradicionales (Ahyar Muawwal, 2024; Sotolongo Martínez, n.d.).

Para dar solución al problema de la presente investigación se desarrolla una aplicación SPA, con su moderno y eficiente enfoque; con ello se evita las dificultades de rendimiento de las MPA sin necesidad de entrar en las complicaciones de implementación de versiones del lado del cliente específicas para diferentes plataformas que implican las PWA.

### Arquitectura cliente-servidor

El modelo abstracto cliente-servidor es una arquitectura de sistemas distribuidos en los que una o más computadoras clientes solicitan recursos de un servidor remoto (que es una computadora física o virtual, o un grupo de ellas) a través de una red. El cliente proporciona interfaces adecuadas a los usuarios mediante las cuales pueden pedir recursos del servidor, mientras que este recibe una o múltiples solicitudes (*requests)*, las procesa, y devuelve una respuesta *(response)* al cliente. Este enfoque es el fundamento mismo del Internet, los sistemas bancarios, las redes celulares, etcétera (Mwamba Nyabuto et al., 2024).

Esta arquitectura cuenta con varias ventajar importantes: la centralización del control (el acceso y los recursos controlados por el servidor asignado, lo cual reduce el riesgo de uso indebido de los datos y posibilita la actualización unificada de los mismos); el manejo apropiado (todos los datos son almacenados en el mismo sitio, lo que facilita su administración); facilidad de respaldo y recuperación de los datos por parte de los usuarios; accesibilidad (el servidor puede brindar servicios a los clientes con independencia de la ubicación de estos); etcétera (Abirami et al., 2019).

El contexto de las tecnologías, herramientas y desarrollo de la parte del cliente se conoce como *front-end*; y el del lado del servidor, como *back-end*.

### API y API REST

Una API (del inglés *Application Programming Interface*, interfaz de programación de aplicaciones) es un protocolo de comunicación entre aplicaciones, un convenio que especifica las formas mediante las cuales un sistema o un componente de un sistema puede acceder a funcionalidades de otro. Las APIs vinculan una amplia variedad de softwares y les proveen la comunicación entre sí, por lo que están presentes a lo largo del mundo digital, desde las redes sociales hasta las transacciones de comercio electrónico. Para los desarrolladores de software es importante tener acceso a las APIs de los servicios relevantes a su trabajo, que pueden ser creadas por ellos mismos para regular la interacción entre partes de sus propios productos, o provistas por compañías que dominan el panorama tecnológico, como Facebook, Microsoft y OpenAI (Nguyen, 2023).

Una clase de APIs de singular importancia es conocida como REST (del inglés *REpresentational State Transfer*, transferencia de estado representacional). Esta es un estándar que guía el diseño y desarrollo de procesos que permiten la interacción con datos almacenados en un servidor. Una API se considera de tipo REST (*RESTful* en inglés) si cumple general o totalmente con los siguientes principios:

* Arquitectura cliente-servidor (véase el epígrafe 1.2.2)
* Ausencia de estado (la comunicación cliente-servidor no se apoya en el almacenamiento del contexto del cliente en el servidor entre solicitudes. Cada solicitud de cualquier cliente contiene toda la información necesaria para efectuar la solicitud).
* Uso del caché (las respuestas a las solicitudes deben ser cacheables por el cliente, en pro de la escalabilidad y el rendimiento del sistema).
* Sistema por capas (el cliente no puede determinar si está conectado directamente al extremo del servidor o a un intermediario; las capas del sistema están compartimentadas).
* Código por demanda (los servidores pueden extender o adaptar temporalmente la funcionalidad de un cliente al transferir código ejecutable).
* Interfaz uniforme: los servidores mantienen una uniformidad razonable entre las APIs que ofrecen, de tal forma que los desarrolladores estén familiarizados con la generalidad de ellas al conocer solo una (Abba, 2022; Visual Paradigm, n.d.).

Una API REST es, por tanto, una opción adecuada para la comunicación entre el servidor y el cliente en el sistema propuesto como solución al problema.

## Conceptos de lenguaje de programación y framework

Un lenguaje de programación es un lenguaje formal diseñado para especificar instrucciones ejecutables por una computadora digital. Los desarrolladores de hoy cuentan con una variedad enorme de lenguajes de programación para seleccionar, los cuales se distinguen entre sí por sus propósitos o subdominios a los que están dedicados, los paradigmas que siguen para construir programas, el nivel de abstracción que presentan con respecto al código directamente ejecutable por un procesador, etcétera.

En el contexto del desarrollo de aplicaciones web, un *framework* es una plataforma de software que provee a los desarrolladores herramientas preconstruidas y bibliotecas para simplificar el desarrollo de las aplicaciones. *Frameworks* para *front-end* y *back-end* promueven además el uso de buenas prácticas y procesos estandarizados, permitiendo a los desarrolladores mejorar la calidad del código y su mantenibilidad (B. Patel, 2023).

En el desarrollo del sistema propuesto como solución al problema de esta investigación, se hace uso de varios lenguajes de programación y *frameworks*, los cuales serán especificados en los epígrafes subsiguientes.

## Tecnologías del lado del cliente y desarrollo front-end

### HTML, CSS y JavaScript

Todo el desarrollo web moderno del lado del cliente está construido sobre tres lenguajes fundamentales: HTML, CSS y JavaScript.

HTML (del inglés *HyperText Markup Language*, lenguaje de marcado de hipertexto) es un lenguaje relativamente simple que permite a los desarrolladores crear la estructura básica de un sitio web. HTML no usa funciones o procedimientos en la forma usual de un lenguaje de programación algorítmico, sino que emplea etiquetas (*tags*) para marcar diferentes tipos de contenido en una página web.

CSS (*Cascading Style Sheets*, hojas de estilo en cascada) permite dar estilos y aspectos diferentes a los elementos definidos por HTML, ajustando propiedades como el color, el tamaño, el tipo de letra, márgenes, rellenos, etcétera. También ayuda a los sitios web a adaptarse a diferentes tamaños de dispositivos y tamaños de pantalla.

JavaScript es el más complejo de esta trilogía. Es un lenguaje completo con plenas capacidades algorítmicas (puede ser usado para codificar cualquier procedimiento computable), y por tanto es usado para dar funcionalidad al lado del cliente. JavaScript es considerado el lenguaje de programación más usado en el mundo (Austin Boot Camps, n.d.).

HTML, CSS y JavaScript son esenciales para el desarrollo de cualquier aplicación web, y por tanto están plenamente integrados en el sistema propuesto por la presente investigación.

### JSON

JSON (*JavaScript Object Notation*, notación de objetos de JavaScript en inglés) es un formato de datos creado para aplicaciones de JavaScript. Su principal propósito es facilitar el intercambio de datos de forma clara y eficiente. Para ello usa pares de clave (*key*) y valor (*value*) al estilo de un objeto nativo de JavaScript (de ahí su nombre).

JSON es ampliamente usado en el desarrollo web para el traspaso de información entre el cliente y el servidor, debido a las ventajas que ofrece: intercambio compacto de los datos, independencia del lenguaje que lo use, compatibilidad prácticamente universal, legibilidad por humanos y facilidad de análisis (*parsing*) (Soyak, 2024).

Por las razones anteriores, JSON es escogido como el formato de intercambio de datos entre el cliente y el servidor en la aplicación web desarrollada.

### React y Next.js

React es una biblioteca de JavaScript desarrollada y mantenida por Facebook (Meta) para la construcción de interfaces de usuario (UI, siglas de *User Interface* en inglés) de aplicaciones web. Con ella, los desarrolladores pueden adoptar un enfoque declarativo y eficiente para la creación de elementos de UI capaces de ser actualizados dinámicamente sobre la base de cambios en los datos o interacciones del usuario.

React usa el modelo virtual DOM *(Document Object Model)* para renderizar y acualizar los elementos UI eficientemente. La estructura y el comportamiento de estos es descrita con JSX (*JavaScript XML)*, un lenguaje específico al dominio que combina sintaxis de JavaScript con XML. La estructura elemental usada por React es el *component* (componente), que permite el desarrollo modular de unidades reusables que se adaptan mediante *props* (propiedades) ajustables a los diferentes usos del mismo componente (Ferreira et al., 2024).

Next.js es un popular *framework* de React para aplicaciones web, que ofrece como ventajas una experiencia de usuario mejorada, rendimiento superior y eficiencia de SEO (*Search Engine Optimization*, optimización de motores de búsqueda), y simplificación del desarrollo. La evolución de Next.js ha sido estable, y su utilidad para la implementación de aplicaciones con React ha sido abundantemente demostrada por la experiencia de la comunidad de desarrolladores (Patel, 2023; Satter et al., 2023; Venkata Koteswara Rao Ballamudi et al., 2021).

Por tanto, la aplicación implementada usa React con Next.js para el *front-end.*

### Tailwind

Tailwind es un *framework* de CSS usado en el desarrollo de *front-end* para estilizar las interfaces gráficas de usuario. Se destaca por su popularidad, la claridad de su documentación y su eficiencia de almacenamiento en la compilación de archivos. Trabaja mediante la inserción de *utility classes* predefinidas (y ampliables mediante personalización) directamente dentro de los elementos HTML, lo cual, mediante la combinación de aquellas, ofrece gran flexibilidad a los desarrolladores (Fagner, 2021; Vargas Zermeño, 2024).

Debido a lo anterior, Tailwind fue seleccionado como herramienta de estilo para la aplicación web desarrollada.

## Tecnologías del lado del servidor y desarrollo back-end

### Python

Python, desarrollado por Guido van Rossum, fue introducido en 1991. Es un lenguaje de programación de propósito general de alto nivel e interpretado. Su metodología y características están diseñadas para ayudar a los programadores a escribir código conciso y fácilmente legible para proyectos de cualquier envergadura. Su sintaxis es clara y simple y se apoya fuertemente en la indentación. Python usa recolección de basura para mejorar el uso de memoria y tiene tipado dinámico. Aunque concebido principalmente como un lenguaje orientado a objeto, es compatible también con los paradigmas funcional y estructurado (particularmente procedural). Viene con una biblioteca estándar particularmente amplia. Es uno de los lenguajes de programación más populares del mundo, y la extensísima comunidad de desarrolladores que lo usan lo ha enriquecido con diversas bibliotecas disponibles de forma gratuita (Manoj Kumar and Dr Rainu Nandal, 2024). Todas las ventajas anteriores fundamentan la elección de Python como lenguaje de programación adecuado para la implementación del *back-end* del sistema.

### Django

Django es un *framework* para *back-end* escrito en Python que goza de amplia aceptación en la comunidad de desarrolladores por su enfoque en la eficiencia y la simplicidad. Al seguir el patrón MTV (*Model-View-Template*), Django proporciona una estructura sólida para la implementación de aplicaciones web escalables y de alto rendimiento. Ofrece por defecto una gran variedad de herramientas listas para usar, y se destaca por su soporte para bases de datos, administración de usuarios, seguridad y gestión de formularios (Ramírez-Galvis, 2023).

Estas características justifican la adopción de Django como tecnología de base para el desarrollo del lado del servidor en la aplicación web propuesta.

### Django REST Framework

Django REST Framework es un paquete construido sobre Django para la creación de APIs REST en sistemas web. Añade al desarrollo *back-end* con Django simplicidad y velocidad, cuenta con una documentación robusta, es extendible y personalizable, y ofrece un buen soporte para la autenticación y la administración de permisos. Entre sus características más destacadas están las facilidades de serialización (conversión de tipos de datos complejos a tipos nativos de Python que pueden ser renderizados como JSON o XML), el uso de *ViewSet* y *Routers*, paginación, etcétera (Hayat Maki, 2023; SkillHats, n.d.).

Todo lo anterior fundamenta el uso de Django REST Framework en el desarrollo de la solución web propuesta.

## Bases de datos

Una base de datos es una colección organizada de datos, almacenados y accedidos electrónicamente. Las bases de datos son usadas para almacenar y administrar grandes cantidades de datos estructurados o no estructurados, y pueden ser empleadas para apoyar un amplio rango de actividades de diversos tipos de sistemas. Hay diferentes clases de bases de datos, incluyendo las relacionales, orientadas a objetos y NoSQL (Vieedy, 2023).

Las bases de datos relacionales son adecuadas para el almacenamiento en tablas con esquemas rígidos (cada una de las instancias guardadas en una tabla tiene que contener los mismos campos con los mismos tipos de datos) que se vinculan unas con otras mediante campos en común (Mitrani, 2019). Esta clase de bases de datos es propicia para el manejo de la información con la que ha de trabajar el sistema que dé solución al problema de investigación, debido a que se manejará instancias con las mismas estructuras de datos agrupables en relaciones o tablas.

### Sistema gestor de bases de datos

Un sistema gestor de bases de datos es un software que permite a los usuarios crear y administrar bases de datos, así como insertar, proteger, leer, actualizar y eliminar datos en ellas; y sirve como interfaz entre las bases de datos y las aplicaciones, asegurando que la información se organiza consistentemente y permanece accesible (Yasar and Mullins, n.d.).

### PostgreSQL

PostgreSQL es un sistema gestor de bases de datos objeto-relacional y de código abierto, que usa y extiende el lenguaje SQL (*Standard Query Language*, el más extendido e influyente en la historia de las bases de datos) combinado con diversas características que almacenan y escalan con seguridad el trabajo con grandes volúmenes de datos. PostgreSQL ha obtenido una sólida reputación por su arquitectura, confiabilidad, integridad de los datos, robustez de funcionalidad, y extensibilidad, y la dedicación de la comunidad de código abierto que lo respalda. Funciona en todos los principales sistemas operativos (The PostgreSQL Global Development Group, n.d.).

Por tales razones, PostgreSQL es una opción adecuada para suplir las necesidades de manejo de base de datos de la aplicación web propuesta.

## Autenticación y autorización

La autenticación es el proceso de verificación de la identidad de un usuario mediante algún tipo de credenciales, como un par de nombre de usuario y contraseña. La autorización es la concesión o negación de acceso a recursos o acciones específicas dentro de un sistema.

Estos conceptos son capas esenciales de seguridad para los sistemas basados en Internet. A lo largo de los años, muchos esfuerzos han sido realizados en aras de propiciar entornos seguros mediante el uso de mecanismos de autenticación, y se considera que en general las formas convencionales de verificación de identidad ofrecen un buen acercamiento y permiten a los usuarios acceder a aplicaciones web seguras (Olanrewaju et al., 2021).

Los dos enfoques principales usados por las aplicaciones web para administrar y recordar las identidades de los usuarios son el basado en sesión y el basado en *tokens*. En el primero, cuando un usuario se autentica, el servidor crea una sesión y la rastrea, le asigna un identificador y lo envía de vuelta al cliente. En el segundo, el servidor provee los detalles del usuario y los encripta en un *token*; de forma que el servidor no tiene que guardar información alguna subsiguiente sobre una sesión, sino que es la responsabilidad del cliente incluir el *token* en cada solicitud al servidor.

Un tipo especial de *token* usado para el manejo de autorizaciones es JWT (*JSON Web Token*), que tiene una estructura bien definida consistente en tres partes, llamadas *Header, Payload* y *Signature*.

El propósito principal de JWT es asegurar la integridad de los datos. Incluso si un actor malicioso consigue acceder al JWT de un usuario, no puede manipular los datos sin la clave secreta.

Este método de autenticación es, pues, adecuado para la implementación requerida.

### NextAuth y Django REST Framework

NextAuth.js es una solución de autenticación de código abierto para aplicaciones hechas con el *framework* Next.js, tratado en el epígrafe 1.4.3. Es un servicio flexible, fácil de usar, compatible con cualquier sistema de bases de datos, y prestigioso en la comunidad de desarrollo web con Next.js (NextAuth Documentation, 2024).

Django REST Framework, cubierto en el epígrafe 1.5.3, ofrece un rango de métodos de autenticación para salvaguardar las aplicaciones de accesos no autorizados. Entre esas técnicas se destaca la autenticación basada en *tokens* como un acercamiento popular y robusto, amoldado a las necesidades de las aplicaciones web modernas, que fue abordado en el epígrafe 1.7. En cuanto al aspecto de la autorización, los permisos en Django REST Framework controlan las acciones que un usuario tiene derecho a realizar sobre recursos específicos. El *framework* provee varias clases predefinidas para la definición de permisos, como IsAuthenticated, IsAdminUser, y AllowAny, pero también permite la creacicón de clases personalizadas (Perez, 2024).

La combinación de NextAuth.js y la funcionalidad de autenticación basada en *tokens* ofrece un esquema de autenticación y autorización adecuado para el sistema requerido.

## Conclusiones parciales del capítulo

# capítulo 2. descripción de la propuesta técnica implementada.

# capítulo 3. validación y prueba de los resultados.

# conclusiones

# recomendaciones

# referencias bibliográficas