

**Autor**: Alain Daniel Rodríguez Domínguez

**Tutores**: Lic. Alejandro Cespón

Dr.C.Orlando Miguel Saucedo Castillo

**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 Ciencias de la 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

**Tutors**: B. A. Alejandro Cespón

Dr.C.Orlando Miguel Saucedo Castillo

**Bachelor in Computer Science**

**Computer Science 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 (debido a su incapacidad para obtener nutrientes de células muertas y su dependencia de plantas de maíz) 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 (Góngora-Canul et al., 2022; 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.

*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, 2024).

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 y el pronóstico de la misma?

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 requisitos del sistema basados en los aspectos técnicos del negocio, 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 que resultan relevantes al desarrollo del sistema propuesto, así como los conceptos, enfoques, tecnologías y herramientas empleadas para la implementación de este.

## Aspectos técnicos del negocio y requisitos del sistema

El sistema a desarrollar debe ser capaz de almacenar, recuperar y eliminar información relevante a la investigación sobre la incidencia de *Phyllachora maydis*; en particular, de los registros diarios de la información meteorológica dados por las estaciones correspondientes, con la fecha de observación, la temperatura máxima, mínima y media en grados Celsius, la humedad máxima, mínima y media en porciento, la cantidad de horas con humedad relativa mayor que el 90% (y dentro de ese período, los valores de temperatura máxima, media y mínima), la precipitación del día anterior en milímetros y la velocidad del viento en metros por segundo. De los terrenos de cultivo se debe guardar, con información provista por las entidades de producción, el nombre de la forma productiva correspondiente, la denominación del cultivar, el tipo de suelo, la fecha de siembra, si la semilla tiene tratamiento químico o no, el área sembrada en hectáreas, el tipo de fertilizante y su dosis (en kilogramos por hectárea), el marco de siembra y el sistema de riego. De cada estación meteorológica se tendrá su código, nombre y municipio donde está ubicada.

Otra funcionalidad del software será emitir pronósticos sobre la incidencia de la enfermedad fúngica mancha de asfalto sobre la base de condiciones climáticas. La información especificada por este pronóstico incluye la unidad de producción y el cultivar en riesgo de afectación, el período climático que resultó favorable para la aparición de la enfermedad y que por tanto motivó la emisión del pronóstico, el plazo de predicción de los primeros síntomas de la mancha de asfalto, el tipo de mensaje de predicción, así como el valor total de “grados días” o suma térmica en grados Celsius.

Según las indicaciones directas del cliente experto, el Dr.C.Orlando Miguel Saucedo Castillo, investigador titular del Centro de Investigaciones Agropecuarias (CIAP) de la Universidad Central "Marta Abreu" de Las Villas, a partir del día de siembra se comienza el cálculo diario de la suma térmica. Cuando el acumulado alcance 700 °C, se comienza a tomar un período de 7 días. Si en esos días se registra una temperatura media diaria entre 17 °C y 22 °C, una máxima menor que 30 °C, mínima superior a 15 °C, humedad relativa diaria mayor o igual a 90% por más de 7 horas, y precipitación mayor que 25 mm, se da lugar a que el día siguiente sea crítico. Es decir, se toma los días 1 al 7 después de la suma térmica de 700 °C, y si se cumplen los parámetros meteorológicos anteriores el día 8 es crítico y después se va analizando los siguientes 7 días corriendo un día hacia adelante, del 2 al 8, y así sucesivamente. Cuando se obtenga 5 días críticos se emite un pronóstico de alerta, y cuando se llegue a 6 días críticos se da uno de peligro máximo (no tienen que ser estrictamente consecutivos, pues si uno o dos días subsiguientes no resultan críticos, se saltan; pero si se pasan de dos días seguidos que no sean críticos, se reinicia la cuenta). La acumulación de calor (grados-días) se calcula restando 10 °C del valor medio de las temperaturas máxima y mínima; pero si la temperatura máxima es mayor que 30 °C, se reemplaza por 30 °C; y si la mínima es menor que 10 °C, se reemplaza por 10 °C.

## 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. 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 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 ofrecen funcionalidades que combinan el acceso al servidor con el soporte *offline*; se pueden instalar en el dispositivo del usuario y, hasta cierto punto, usarse sin conexión a Internet. También incluyen potencialmente 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).

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 y las redes celulares (Mwamba Nyabuto et al., 2024).

Esta arquitectura cuenta con varias ventajas importantes: la centralización del control (el acceso y los recursos son 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 organizado (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 Meta, 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 en inglés). Fue propuesta por Roy Fielding en 2000 y se ha convertido en un estándar *de facto* para las interacciones de servicios web (Liu et al., 2022).

Las APIs de tipo REST se adhieren al intercambio de datos mediante protocolos estandarizados, e incluyen entre sus principios fundamentales la ausencia de estado (el servidor no almacena información sobre el estado del cliente), la cacheabilidad (las respuestas del servidor deben ser compatibles con el uso de caché), y una interfaz uniforme que simplifique las interacciones cliente-servidor. Usan extremos o *endpoints*, que son funcionalidades concretas implementadas de un proceso de negocios, y son invocadas con ayuda de una dirección URL (*Uniform Resource Locator,* localizador uniforme de recursos en inglés). Las solicitudes del cliente, mediante las cuales este puede acceder a recursos de datos a ser creados, eliminados, actualizados o recuperados, son enviadas a uno de los *endpoints* con una ruta de recurso y un método, que comúnmente es POST (crear), GET (recuperar), PUT (actualizar) o DELETE (eliminar) (Ehsan et al., 2022; Kim et al., 2023).

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, entre otros.

En el contexto del desarrollo de aplicaciones web, un *framework* es una plataforma de software que provee a los desarrolladores herramientas preconstruidas para la implementación de sus proyectos. Un *framework* extiende las características normalmente presentes en el lenguaje sobre el cual se basan y añaden sus propias clases, funciones y capacidades, con el objetivo de simplificar el trabajo de producción y mantenimiento de código (Mendez, 2014).

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 en ingés) constituye el esqueleto del desarrollo web, pues provee la estructura básica y el contenido de una página web, definiendo elementos como tablas, formularios, listas y títulos, e identifica dónde las secciones empiezan y terminan; todo ello mediante etiquetas (*tags*).

CSS (*Cascading Style Sheets*, hojas de estilo en cascada) permite dar estilos y aspectos diferentes a los elementos definidos por HTML, creando reglas sobre el color, la fuente y la forma de mostrar las páginas web. También determina cuándo dichas reglas deben ser usadas, sobre la base de información como el tipo de dispositivo conectado, o en respuesta a una acción del usuario (Mendez, 2014).

JavaScript fue concebido originalmente como un lenguaje simple del lado del cliente para especificar instrucciones que añadieran funcionalidad a las páginas web. Sin embargo, su evolución gradual lo ha transformado en un lenguaje más robusto y versátil, capaz de abordar desafíos de programación complejos, y el surgimiento de *frameworks* actuales para JavaScript ha facilitado su transformación en un lenguaje de programación que puede emplearse tanto en el desarrollo *back-end* como en el *front-end* (USA and Shukla, 2023).

HTML, CSS y JavaScript son esenciales para el desarrollo del lado del cliente 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 codificación de datos estandarizado para su transmisión entre sistemas, que se ha hecho muy popular en la industria del desarrollo de software para la lectura y escritura de datos en disco, así como en la comunicación a través de una red. JSON usa la sintaxis de JavaScript para representar datos estructurados de forma concisa, legible por máquinas y humanos, con especificaciones de lenguaje e implementación bastante simples (Stanek and Killough, 2023).

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 en inglés), 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 ofrece gran flexibilidad a los desarrolladores (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 notablemente 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 (Kumar and 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 popular paquete de Django usado para construir APIs, usado en la implementación de numerosos servicios web actuales. Añade a Django útiles componentes preconstruidos, como las *views* REST basadas en clases, *viewsets* y serializadores, de forma que la creación de las interfaces que el servidor ofrecerá a las aplicaciones del lado del cliente se vuelve más concisa, simple y mantenible para el desarrollador (Gagliardi, 2021).

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

## Bases de datos

Las bases de datos son colecciones organizadas de información que se almacenan y gestionan en un sistema informático. Están diseñadas para gestionar eficazmente grandes volúmenes de datos estructurados o no estructurados, proporcionan una forma de almacenar y acceder a los datos para su procesamiento, y son por lo general el elemento central del lado del servidorde un sistema de software.

Una clase de gran importancia dentro de las bases de datos es las relacionales. Estas se basan en el álgebra relacional, una teoría que utiliza estructuras algebraicas con una semántica bien fundamentada para modelar datos y definir consultas sobre ellos. Los datos se almacenan como tablas bidimensionales con filas y columnas, y los valores de los datos deben pertenecer a uno de los tipos definidos en el sistema. En la actualidad estas siguen siendo las bases de datos más utilizadas (Fernández Iglesias, 2024).

Las bases de datos relacionales son propicias 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 una aplicación que sirve para administrar bases de datos e interactuar con ellas de manera eficiente y segura. Proporciona una interfaz para crear, modificar, almacenar y recuperar datos de la base, y gestiona aspectos como la integridad de los datos, la seguridad, la concurrencia y la recuperación en caso de fallos (Fernández Iglesias, 2024).

### 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*, muy usado 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, 2024).

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).

Dos de las principales estrategias de autenticación en sistemas web son la basada en sesión y la basada en *tokens.* El primero es el más tradicional, y fue dominante en el desarrollo web de días pasados. Consiste en el seguimiento por parte del servidor de una sesión abierta por el cliente y finalmente cerrada al terminar su actividad. El segundo usa *tokens*, unidades de información dadas por el servidor al cliente que deben ser usadas en las solicitudes que este hace al servidor, de forma que el *token* sirve como clave secreta temporal (Balaj, 2017).

Como tipo de *token* para autenticación destaca *JSON Web Token* (JWT), que representa un medio ampliamente adoptado de intercambio seguro de información entre entidades como objetos JSON (véase el epígrafe 1.4.2) que facilitan la confidencialidad de alto nivel. Los JWTs están diseñados para permitir mecanismos de firmado, lo cual se puede lograr empleando criptografía de clave secreta o encriptación de par de llaves pública-privada, protegiendo los datos en la transmisión. Con tales protocolos independientes de almacenamiento de estados de sesión, JWT es adecuado para escenarios como APIs REST y constituye una opción aceptable para la funcionalidad de autenticación en el sistema requerido (Dimitrijevic et al., 2024).

### 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 las técnicas provistas 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 Django REST Framework, a diferencia del *TokenAuthentication* preconstruido, la autenticación JWT no necesita usar una base de datos para validar un *token*. Un paquete para JWT es djangorestframework-simplejwt. En cuanto al aspecto de la autorización, los permisos en Django REST Framework siempre son definidos como una lista de clases de permisos. Antes de ejecutar el cuerpo principal de una *view* cada permiso en la lista es chequeado; y si algún permiso falla, una excepción es elevada, el cuerpo de la *view* no se correrá, y se retornará una respuesta 403 Forbidden o 401 Unauthorized (Django REST Framework Documentation, 2024).

La combinación de NextAuth.js y las funcionalidades basadas en JWT y clases de permisos de Django REST Framework ofrece un esquema de autenticación y autorización adecuado para el sistema requerido.

## Conclusiones parciales del capítulo

En este capítulo se especificó los requisitos que ha de cumplir la aplicación concebida como respuesta al problema de investigación en el entorno de las facetas técnicas del mismo, y se describió las características más relevantes de las tecnologías de desarrollo web seleccionadas para ser usadas en la implementación del sistema; y así se argumentó que una aplicación web de tipo SPA con la típica arquitectura cliente-servidor, hecha con HTML, CSS y JavaScript mediante las plataformas React, Next.js y Tailwind en el *front-end*, y Django y Django REST como *frameworks* basados en Python en el *back-end*, así como una combinación de NextAuth y las facilidades de Django REST para autenticación y autorización, constituye una solución viable y adecuada al problema planteado.

# capítulo 2. descripción de la propuesta técnica implementada.

# capítulo 3. validación y prueba de los resultados.

# conclusiones

# recomendaciones

# referencias bibliográficas

Abirami, N., Lavanya, S., Madhanghi, A., 2019. A Detailed Study of Client-Server and its Architecture (Short Paper). Sri Krishna Arts and Science College, Kuniyamuthur, Coimbatore.

Ahyar Muawwal, 2024. The Implementation of PWA (Progressive Web App) Technology in Enhancing Website Performance & Mobile Accessibility: The Implementation of PWA (Progressive Web App) Technology in Enhancing Website Performance & Mobile Accessibility. Bul. Pos Dan Telekomun. 22. https://doi.org/10.17933/bpostel.v22i1.395

Balaj, Y., 2017. Token-Based vs Session-Based Authentication: A Survey. University of Prishtina “Hasan Prishtina.”

Díaz-Morales, F., De León-García De Alba, C., Nava-Díaz, C., Mendoza-Castillo, M.D.C., 2018. Inducción de resistencia a Puccinia sorghi y complejo mancha de asfalto (Phyllachora maydis y otros) en maíz (Zea mays). Rev. Mex. Fitopatol. Mex. J. Phytopathol. 37. https://doi.org/10.18781/R.MEX.FIT.1807-6

Dimitrijevic, N., Zdravkovic, N., Bogdanovic, M., Mesterovic, A., 2024. Advanced Security Mechanisms in the Spring Framework: JWT, OAuth, LDAP and Keycloak.

Django REST Framework Documentation, 2024. Authentication - Django REST framework [WWW Document]. URL https://www.django-rest-framework.org/api-guide/authentication/ (accessed 9.12.24).

Ehsan, A., Abuhaliqa, M.A.M.E., Catal, C., Mishra, D., 2022. RESTful API Testing Methodologies: Rationale, Challenges, and Solution Directions. Appl. Sci. 12, 4369. https://doi.org/10.3390/app12094369

European and Mediterranean Plant Protection Organization, 2024. Phyllachora maydis [WWW Document]. URL https://www.eppo.int/ACTIVITIES/plant\_quarantine/alert\_list\_fungi/https%3A%2F%2Fwww.eppo.int%2FACTIVITIES%2Fplant\_quarantine%2Falert\_list\_fungi%2Fphyllachora\_maydis (accessed 8.19.24).

Fernández Iglesias, M.J., 2024. Pequeña introducción a las bases de datos. Universidade de Vigo.

Ferreira, F., Borges, H.S., Valente, M.T., 2024. Refactoring react-based Web apps. J. Syst. Softw. 215, 112105. https://doi.org/10.1016/j.jss.2024.112105

Gagliardi, V., 2021. Decoupled Django: understand and build decoupled Django architectures for Javascript front-ends. Apress, Place of publication not identified.

Góngora-Canul, C., Jiménez-Beitia, F., Puerto-Hernández, C., Carolina Avellaneda, M., Kleczewski, N., Telenko, D.E.P., Shim, S., Solórzano, J.E., Goodwin, S.B., Scofield, S.R., Cruz, C.D., 2022. Induction of Phyllachora maydis (Maubl.) signs on corn leaves. Res. Sq. 12.

Hernández Cruz, L.M., González Novelo, M.J., Cab Chan, J.R., Mex Álvarez, D.C., 2023. Implementación de la aplicación web BITA en Google Compute Engine. Multidiscip. Ing. 7, 107–117. https://doi.org/10.29105/mdi.v7i10.228

Kim, M., Sinha, S., Orso, A., 2023. Adaptive REST API Testing with Reinforcement Learning.

Kowalczyk, K., Szandala, T., 2024. Enhancing SEO in Single-Page Web Applications in Contrast With Multi-Page Applications. IEEE Access 12, 11597–11614. https://doi.org/10.1109/ACCESS.2024.3355740

Kralina, H., Popova, A., 2024. NOWADAYS TRENDS IN WEB DEVELOPMENT, in: EDUCATION AND SCIENCE OF TODAY: INTERSECTORAL ISSUES AND DEVELOPMENT OF SCIENCES. Presented at the EDUCATION AND SCIENCE OF TODAY: INTERSECTORAL ISSUES AND DEVELOPMENT OF SCIENCES, European Scientific Platform. https://doi.org/10.36074/logos-29.03.2024.068

Kumar, M., Nandal, D.R., 2024. Python’s Role in Accelerating Web Application Development with Django. Int. Res. J. Adv. Eng. Manag. IRJAEM 2, 2092–2105. https://doi.org/10.47392/IRJAEM.2024.0307

Liu, Yi, Li, Y., Deng, G., Liu, Yang, Wan, R., Wu, R., Ji, D., Xu, S., Bao, M., 2022. Morest: model-based RESTful API testing with execution feedback, in: Proceedings of the 44th International Conference on Software Engineering. Presented at the ICSE ’22: 44th International Conference on Software Engineering, ACM, Pittsburgh Pennsylvania, pp. 1406–1417. https://doi.org/10.1145/3510003.3510133

Llamuca-Quinaloa, J., Vera-Vincent, Y., Tapia-Cerda, V., 2021. Análisis comparativo para medir la eficiencia de desempeño entre una aplicación web tradicional y una aplicación web progresiva. TecnoLógicas 24, e1892. https://doi.org/10.22430/22565337.1892

Mendez, M., 2014. The Missing Link - An Introduction to Web Development and Programming. Open SUNY Textbooks.

Mwamba Nyabuto, G., Mony, V., Mbugua, S., 2024. Architectural Review of Client-Server Models. Int. J. Sci. Res. Eng. Trends 10, 139–143.

NextAuth Documentation, 2024. Introduction | NextAuth.js [WWW Document]. URL https://next-auth.js.org/getting-started/introduction (accessed 9.6.24).

Nguyen, T., 2023. What are APIs? A computer scientist explains the data sockets that make digital life possible [WWW Document]. The Conversation. URL http://theconversation.com/what-are-apis-a-computer-scientist-explains-the-data-sockets-that-make-digital-life-possible-213042 (accessed 9.6.24).

Olanrewaju, R.F., Khan, B.U.I., Morshidi, M.A., Anwar, F., Kiah, M.L.B.M., 2021. A Frictionless and Secure User Authentication in Web-Based Premium Applications. IEEE Access 9, 129240–129255. https://doi.org/10.1109/ACCESS.2021.3110310

Panwar, V., 2024. Web Evolution to Revolution: Navigating the Future of Web Application Development. Int. J. Comput. Trends Technol. 72, 34–40. https://doi.org/10.14445/22312803/IJCTT-V72I2P107

Patel, V., 2023. Analyzing the Impact of Next.JS on Site Performance and SEO. Int. J. Comput. Appl. Technol. Res. https://doi.org/10.7753/IJCATR1210.1004

Ramírez-Galvis, J.P., 2023. Generación de un servicio RSS con Django. https://doi.org/10.13140/RG.2.2.15992.62723

Satter, A., Tabassum, A., Ishrat, J.E., 2023. Software Evolution of Next.js and Angular. Int. J. Eng. Manuf. 13, 20–33. https://doi.org/10.5815/ijem.2023.04.03

Stanek, J., Killough, D., 2023. Synthesizing JSON Schema Transformers. Univ. Wis. 1, 7.

The PostgreSQL Global Development Group, 2024. PostgreSQL: About [WWW Document]. URL https://www.postgresql.org/about/ (accessed 8.20.24).

USA, Shukla, A., 2023. Modern JavaScript Frameworks and JavaScript’s Future as a FullStack Programming Language. J. Artif. Intell. Cloud Comput. 1–5. https://doi.org/10.47363/JAICC/2023(2)144

Valdez Ocampo, F.D., Huerta Maciel, Á.M., Mongelós Barrios, C.A., Sánchez Jara, R., Ruiz Díaz Lovera, E.M.D., Sanchez Gonzalez, M.A., 2024. Evaluación de cultivares de maíz (Zea maíz L.) sembrados en diferentes arreglos espaciales. Rev. Alfa 8, 363–375. https://doi.org/10.33996/revistaalfa.v8i23.269

Vargas Zermeño, E., 2024. Criterios de implementación de Tailwind CSS en desarrollos frontend. Cuad. Téc. Univ. DGTIC 2. https://doi.org/10.22201/dgtic.ctud.2024.2.3.63

Venkata Koteswara Rao Ballamudi, Karu Lal, Harshith Desamsetti, Sreekanth Dekkati, 2021. Getting Started Modern Web Development with Next.js: An Indispensable React Framework. Digit. Sustain. Rev. 1.

Vinces-Tachong, R.E., Vélez-Ruiz, M.C., Gaibor-Fernández, R.R., Herrera-Eguez, F.E., 2022. Implementación del procesamiento de imágenes para la evaluación de la mancha de asfalto (Phyllachora maydis) en maíz (Zea mays). Rev. TERRA Latinoam. 40. https://doi.org/10.28940/terra.v40i0.1066