U N I V E R S I D A D C E N T R A L “ M A R T A A B R E U ” D E L A S V I L L A S
F A C U L T A D D E M A T E M Á T I C A F Í S I C A Y C O M P U T A C I Ó N

S I S T E M A A U T O M A T I Z A D O D E E C O S I S T E M A S A C U Á T I C O S

T r a b a j o d e D i p l o m a

A u t o r: L i e n R o d r í g u e z L ó p e z

T u t o r e s: D r a . G l a d y s C a s a s C a r d o s o

D r . R o l a n d o C á r d e n a s O r t i z

S a n t a C l a r a

2 0 1 1

DICTAMEN

Hago constar que el presente trabajo de diploma fue realizado en la Universidad Central "Marta Abreu" de Las Villas como parte de la culminación de estudios de la especialidad de Ingeniería Informática, 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 publicados sin autorización de la Universidad.

Firma del Autor

Los abajo firmantes certificamos que el presente trabajo ha sido realizado según acuerdo 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 Jefe de Departamento donde se defiende el trabajo

Firma del Tutor

PENSAMIENTO

Con orden y tiempo se encuentra el secreto de hacerlo todo, y de hacerlo bien.

Pitágoras

DEDICATORIA

AGRADECIMIENTOS

A mis padres y abuelos, sin ellos nada tendría sentido

Al Ninja y el Muke, gracias por toda la ayuda

A Leidy por siempre estar donde la necesito

A mis tutores Gladita y Rolando por todo su apoyo

A mis profesores por darnos lo mejor de si

Al grupo de Bioinformática.

A Mayrene y Daile por su colaboración

Resumen

Los ecosistemas acuáticos se ven grandemente afectados actualmente a causa del trabajo humano, el desarrollo turístico, la contaminación ambiental, la pesca masiva y el cambio climático global. En el Centro de Investigación de Ecosistemas Costeros (CIEC) en Cayo Coco se lleva a cabo un estudio para preservar las comunidades de estas aguas. Las herramientas disponibles en el centro no satisfacen sus necesidades en cuanto a precisión. Este trabajo propone la implementación de un sistema automatizado para simplificar el trabajo de los biofísicos que laboran en este lugar. Para ello se realizaron mediciones en el área de Laguna Larga sobre las comunidades de pastos marinos que habitan esta zona aplicando técnicas experimentales que aportaron los valores para la Base de Datos elaborada en el framework de desarrollo Django y para el sitio web capaz de realizar el cálculo correspondiente a la inhibición de la fotosíntesis por el ultravioleta. De esta forma se facilita el trabajo para el desarrollo científico técnico de estos especialistas en Bioinformática en nuestro país.

A B S T R A C T

A q u a t i c e c o s y s t e m s a r e g r e a t l y a f f e c t e d n o w a d a y s b e c a u s e o f h u m a n l a b o r , t o u r i s m d e v e l o p m e n t , p o l l u t i o n , o v e r - f i s h i n g a n d g l o b a l c l i m a t e c h a n g e . A s t u d y t o p r e s e r v e t h e c o m m u n i t i e s t h a t i n h a b i t t h o s e w a t e r s i s c a r r i e d o u t a t t h e C e n t e r f o r C o a s t a l E c o s y s t e m R e s e a r c h (C I E C) i n C a y o C o c o . T h e t o o l s a v a i l a b l e i n t h e f a c i l i t y d i d n o t m e e t t h e i r n e e d s a c c o r d i n g t o a c c u r a c y . T h i s p a p e r p r o p o s e s t h e i m p l e m e n t a t i o n o f a n a u t o m a t e d s y s t e m t o s i m p l i f y t h e w o r k o f b i o p h y s i c i s t s w o r k i n g i n t h i s p l a c e . T a k i n g t h i s i s s u e i n t o a c c o u n t , m e a s u r e m e n t s w e r e p e r f o r m e d i n L a g u n a L a r g a r e g i o n , s p e c i f i c a l l y o n s e a g r a s s c o m m u n i t i e s , u s i n g e x p e r i m e n t a l t e c h n i q u e s w h i c h p r o v i d e d v a l u e s f o r t h e d a t a b a s e d e v e l o p e d i n D j a n g o , a f r a m e w o r k o f d e v e l o p m e n t a n d a l s o f o r t h e w e b s i t e c a p a b l e o f c a l c u l a t i n g t h e p h o t o s y n t h e s i s i n h i b i t i o n b y u l t r a v i o l e t . T h i s r e s e a r c h w i l l f a c i l i t a t e t h e w o r k a i m e d t o s c i e n t i f i c a n d t e c h n i c a l d e v e l o p m e n t o f t h e s e B i o i n f o r m a t i c s s p e c i a l i s t s i n o u r c o u n t r y .

Tabla de Contenidos

INTRODUCCIÓN... 1
.....**Error! Bookmark not defined.**

ANTECEDENTES Y PLANTEAMIENTO DEL PROBLEMA... 1
.....**Error! Bookmark not defined.**

OBJETIVOS GENERALES... 3
.....**Error! Bookmark not defined.**

OBJETIVOS ESPECIFICOS... 3
.....**Error! Bookmark not defined.**

ESTRUCTURA DEL TRABAJO... 3
.....**Error! Bookmark not defined.**

2 ANÁLISIS, DISEÑO E IMPLEMENTACION DEL SISTEMA... ... ...
.....**Error! Bookmark not defined.**

3 MANUAL DE USUARIO... ... ...
.....**Error! Bookmark not defined.**

Introducción 10

Antecedentes y planteamiento del problema 10

Objetivo General: 12

Estructura del trabajo: 13

Capítulo 1 Marco Teórico 14

1.1 Introducción a los términos biofísicos 14

1.2 Propagación de la luz en los ecosistemas acuáticos 16

1.2.1 Modelo para el cálculo de la tasa de fotosíntesis 17

1.3 Herramientas computacionales usadas en el sistema 18

1.3.1 HTML 18

1.3.2 CSS 19

1.3.3 JavaScript 19

1.3.4 AJAX 20

1.3.5 Django 21

1.3.6 Python 24

1.3.7 Jquery 25

1.3.8 Patrón de arquitectura Modelo-Vista-Controlador (MVC) 26

1.3 Consideraciones finales 28

Capítulo 2 Análisis, diseño e implementación del sistema 29

2.1 Etapa de Análisis 29

2.1.1 Objetivos del Análisis 30

2.2 Etapa de Diseño del software 31

2.2.1 Diseño de Bases de Datos 32
2.2.2 Diseño del sitio web 35
2.3 Etapa de implementación del sistema 35
2.4 Actores y Casos de Uso del sistema 36
2.4.1 Casos de Uso 36
2.4.2 Actores del sistema 38
2.5 Requisitos Funcionales del Sistema 38
2.6 Requisitos no funcionales del sistema 40
2.7 Arquitectura cliente-servidor 41
2.8 Herramientas utilizadas en el desarrollo del diseño 43
2.8.1 UML 43
2.8.2 Visual paradigm 45
2.8.3 Editor TopStyle_v4 45
2.8.4 SQLite 46
2.9 Diagrama de Despliegue 48
2.10 Consideraciones finales 48
Capítulo 3 Manual de usuario 49
3.1 Manual de usuario 49
3.1.1 Características generales del sistema 49
3.1.2 Requerimientos mínimos 49
3.1.3 Instalación 50
3.1.4 Descripción de la página inicial del sitio 50
3.3 Comparación con el trabajo anterior 53
CONCLUSIONES 59
RECOMENDACIONES 60
REFERENCIAS BIBLIOGRAFICAS 61

Introducción

Los ecosistemas acuáticos incluyen las aguas de los océanos y las aguas continentales dulces o saladas. Los tipos de ecosistemas acuáticos más importantes son los ecosistemas marinos y los ecosistemas costeros. Los manglares, corales, marismas, playas, lagunas y praderas submarinas son los diferentes tipos de ecosistemas costeros existentes. Cada día factores ambientales como los cambios producidos en las temperaturas de los océanos y factores sociales tales como la urbanización, la pesca masiva, la contaminación ambiental y el desarrollo turístico contribuyen a la pérdida de superficie oceánica y que se afecten de cierta manera estos ecosistemas. No se debe olvidar que estos son fuente de energía, alimento, medicina y recreación. Además, protegen las costas de las tormentas, el oleaje, de las inundaciones y la erosión; purifican el agua, mantienen la diversidad y capturan el bióxido de carbono del ambiente, entre otras muchas funciones vitales para el equilibrio ecológico global. Todas estas razones son motivos más que suficientes para conservarlos.[1]

Antecedentes y planteamiento del problema

El proceso de fotosíntesis es uno de los más importantes en toda la biosfera. En este se combina el dióxido de carbono (CO_2) atmosférico con agua, bajo la acción de la luz solar, para dar lugar a compuestos orgánicos y oxígeno que se libera a la atmósfera:

$$CO_2 + H_2O \xrightarrow{\text{luz}} CH_2O + O_2 \qquad (1)$$

Los compuestos orgánicos formados constituyen alimento para las especies que realizan la fotosíntesis, las cuales son llamadas productores primarios (fitoplancton, algas, plantas superiores). Estas especies a su vez son alimento para los animales y el hombre, de modo que este proceso constituye la base de la cadena o ensamblaje alimentario. Además, este proceso tiene implicaciones climatológicas. Para ello recordemos que hoy día sufrimos un incremento del llamado efecto invernadero, que no es más que el hecho de que la atmósfera absorbe más radiación (proveniente del Sol) que la que deja escapar al espacio (irradiada por la Tierra), contribuyendo al calentamiento del planeta. Esto es

debido a que los gases de efecto invernadero absorben más la radiación visible (que es la que más emite el Sol) y menos la radiación infrarroja (que es la que más emite la Tierra). Precisamente el CO_2 es uno de los gases que contribuyen al efecto invernadero y el proceso de fotosíntesis consume este gas, como se puede ver en la ecuación (1). Además la fotosíntesis libera O_2 a la atmósfera, como se ve en los productos de la mencionada ecuación. Todo lo anterior evidencia la enorme importancia de la fotosíntesis para la vida en la Tierra.

La determinación experimental de la eficiencia de la fotosíntesis por lo general implica traslado hasta la zona de investigación, colecta de las muestras y posterior procesamiento a nivel de laboratorio para la medición de las tasas de fotosíntesis. Esto implica cuantiosos gastos de recursos. Por esa razón, algunos investigadores han desarrollado modelos matemáticos de fotosíntesis, los cuales en principio permiten hacer una estimación de la eficiencia fotosintética sin tener que trasladarse al área de estudio. Uno de los investigadores más destacados en esta área es el ambientalista norteamericano Patrick J Neale (Smithsonian Environmental Research Centre) que ha elaborado varios modelos matemáticos para el cálculo de las tasas de fotosíntesis.[2] El Departamento de Física de nuestra universidad, en conjunto con el Grupo de Bioinformática, ha intercambiado algunas ideas con este investigador, precisamente uno de los creadores del modelo matemático de fotosíntesis que utilizamos en esta tesis. Este y otros modelos hoy día se utilizan para predecir la eficiencia de la fotosíntesis fundamentalmente en el Océano del Sur, que rodea la Antártida. El énfasis en esta área se debe al agujero sobre la capa de ozono que se forma sobre el Polo Sur durante el invierno, lo cual implica una mayor incidencia de ultravioleta solar. Es importante destacar que este último fenómeno ocasiona difusión de ozono desde los trópicos hacia los polos, por lo que todo el planeta resulta afectado.

Considerando todo lo anterior, nuestro propósito con este trabajo es la confección de un software que permita la realización de cálculos teóricos con el fin de ahorrar recursos, reducir tiempo de trabajo, lograr la predicción en lugares de difícil acceso y disminuir el número de mediciones de las tasas de fotosíntesis. Este software podría ser utilizado por varias instituciones ambientalistas de nuestro país, siendo la primera el Centro de

Investigaciones de Ecosistemas Costeros (CIEC) de Cayo Coco, Ciego de Ávila, con el que se mantienen activos vínculos de investigación. Los investigadores de este centro han realizado numerosas determinaciones experimentales de tasas de fotosíntesis en diversas áreas, y serían muy beneficiados al poder contar con un software que les permita hacer las estimaciones teóricas sin necesidad de invertir cuantiosos recursos materiales al visitar cada una de las áreas que son de su interés.

Por tanto, para lograr un mayor desarrollo en la comunidad científica relacionada con el medio ambiente en nuestro país, y para maximizar el tiempo de las investigaciones nos trazamos como objetivos para este trabajo:

Objetivo General:

- Automatizar un sistema basado en un modelo físico-matemático de fotosíntesis que permita estimar las tasas de fotosíntesis en diversas latitudes geográficas y a cualquier profundidad en un ecosistema acuático.

Como objetivos específicos se proponen:

- Diseñar una Base de Datos para almacenar los valores correspondientes a las características del modelo.
- Diseñar un sitio web que realice el cálculo del modelo.
- Implementar el sitio web

Para dar cumplimiento a estos objetivos fue necesario plantearse y solucionar algunas tareas de investigación, entre las que se encuentra estudiar todos los elementos de los ecosistemas acuáticos para comprender el procesamiento que realizan los especialistas en el tema.

Después de una amplia revisión de la literatura se formula la siguiente hipótesis de investigación:

"El análisis, diseño e implementación de un sistema para automatizar un modelo físico-matemático de fotosíntesis, le facilita a los biólogos la estimación de las tasas de fotosíntesis en diversas latitudes geográficas y a cualquier profundidad en un ecosistema acuático"

Estructura del trabajo:

El trabajo confeccionado consta de tres capítulos:

El capítulo 1 es el marco teórico. En él se abordaran los conceptos biofísicos fundamentales para un mejor entendimiento del sistema y una detallada explicación del modelo que se utilizará para el desarrollo del mismo. Se presentan luego las herramientas computacionales utilizadas para la elaboración del software.

El capítulo 2 trata sobre el análisis, diseño e implementación del sistema donde se presentan los actores del sistema así como sus casos de uso. Además se muestra el diseño de la base de datos, las herramientas utilizadas en la etapa del diseño, así como la última etapa de implementación del software y los requisitos funcionales y no funcionales para la creación del mismo.

El capítulo 3 se dedica al manual de usuario. En él se muestra el manual de usuario y un ejemplo para un caso especifico del proceso automatizado, cómo se resolvía antes de la confección del software y como se realiza el mismo cálculo luego de ya implementado este.

La tesis culmina con las conclusiones, recomendaciones y referencias bibliográficas.

Capítulo 1 Marco Teórico

En este capítulo se tratan los conceptos fundamentales para una mejor manipulación del sistema, las herramientas que fueron utilizadas para la elaboración del mismo. Se explica además el modelo para el cálculo de las tasas de fotosíntesis, así como las herramientas utilizadas durante el proceso de creación del software.

1.1 Introducción a los términos biofísicos

Existen dos grandes tipos de radiaciones: las compuestas por partículas y las ondas electromagnéticas. Las radiaciones particuladas están formadas por protones, electrones, neutrones y otras partículas. Las ondas electromagnéticas tienen diversas bandas de frecuencia y energía, teniéndose en orden ascendente las ondas de radio y televisión, la luz (infrarrojo, visible y ultravioleta), los rayos X y los rayos gamma, siendo estos últimos los más energéticos.

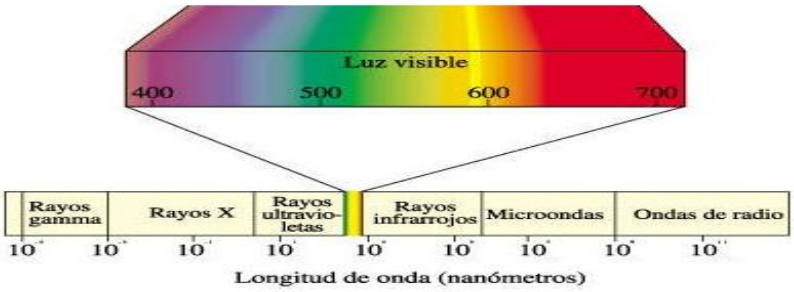


Figura 1.1 **Espectro electromagnético**

El Sol emite constantemente radiaciones electromagnéticas a la Tierra como son la luz ultravioleta (UV) y la luz visible. La luz ultravioleta comprende en el espectro de longitudes de onda al rango de 100 a 400 nm resultando nociva a los seres vivos debido a que es capaz de causar daños severos a nivel molecular. Por otro lado, la luz visible abarca el rango de 400 a 700 nm considerándose beneficiosa, esta región espectral corresponde al rango de luz visible para el ojo humano y además es utilizada por las algas y plantas terrestres como fuente de energía a través del proceso denominado fotosíntesis en el cual la energía luminosa se convierte en energía química dando lugar a su vez a la producción de casi toda la materia orgánica que conocemos. Por esa razón a la luz visible también se le designa también con la sigla PAR (del inglés *Photosynthetically Active Radiation*).

Las ondas electromagnéticas están formadas por un conjunto de cuantos de energía llamados fotones. La relación entre la energía E_f de los fotones y la frecuencia f de la onda es:

$$E_f = hf = \frac{hc}{\lambda} \qquad (1.1)$$

donde h es la constante de Planck, c la velocidad de propagación de la onda en el vacío, y λ es la longitud de onda. Por lo general la acción biológica de las ondas electromagnéticas está asociada a la absorción de fotones por parte de las moléculas biológicas. Para que este proceso ocurra, debe cumplirse la regla cuántica:

$$\Delta E = E_f \qquad (1.2)$$

Donde ΔE es el salto energético que dará la molécula biológica. En general el efecto biológico de los fotones se incrementa medida que aumenta su frecuencia, no sólo porque es mayor su energía, sino porque la probabilidad de cumplimiento de la regla cuántica aumenta (o sea, aumenta la probabilidad de absorción del fotón por la molécula biológica). Por otro lado, las ondas electromagnéticas que nos llegan del Sol son fundamentalmente luz (ultravioleta, visible e infrarroja). La luz visible por lo general es beneficiosa para la fotosíntesis, mientras que la ultravioleta por lo general la inhibe.

Cuando se considera la dosimetría de radiaciones se debe tener en cuenta dos conceptos básicos:

a) La dosis o fluencia, cuyo símbolo es D o F. Esta es la energía suministrada por la radiación por unidad de área de tejido biológico. Se da en J/m^2 en el Sistema Internacional de Unidades (SI).

b) La velocidad de suministro de la dosis, o irradiancia, cuyo símbolo es E. Esta es la energía suministrada por la radiación por unidad de área de tejido biológico y por unidad de tiempo. Se da en W/m^2 en el Sistema Internacional de Unidades.

Las radiaciones ultravioletas del Sol pueden causar algunos daños en el aparato fotosintético de las células, por ejemplo, pueden dañar la enzima Rubisco, que juega un importante rol en la realización de una de las fases de la fotosíntesis. Sin embargo, las células tienen mecanismos (reacciones bioquímicas) que permiten reparar esos daños. Ahora bien, cuando la velocidad de dosis o irradiancia E es alta, las células no tienen la posibilidad de reparar casi ninguno de los daños que las radiaciones van causando, por lo que el efecto radioactivo es acumulativo, y entonces en un modelo matemático se debe usar la dosis o fluencia F. Por otro lado si la velocidad de dosis o irradiancia E es baja, las células tienen la posibilidad de reparar una fracción de los daños que las radiaciones van causando, por lo que el efecto radioactivo no es acumulativo, y entonces en un modelo matemático se debe usar la velocidad de dosis o irradiancia E. Este último es el caso que nos ocupa: en este trabajo se programó un modelo matemático para estimar las tasas de fotosíntesis de organismos unicelulares acuáticos (fitoplancton) sometidos a la irradiancia solar visible y ultravioleta.

1.2 Propagación de la luz en los ecosistemas acuáticos

La irradiancia espectral $E(\lambda)$, o intensidad de la luz en el agua oceánica, decae exponencialmente con la profundidad z según la ley de Lambert-Beer de la Óptica:

$$E(\lambda,z) = E(\lambda,0^-).e^{-K(\lambda).z} \qquad (1.3)$$

La coordenada 0^- es la posición justamente debajo de la superficie acuática, mientras que la 0^+ es la que está justo encima. La relación entre las irradiancias en estas posiciones es:

$$E(\lambda,0^-) = (1-A)E(\lambda,0^+) \qquad (1.4)$$

Donde A es el albedo o poder reflectante del agua, o sea, la fracción de irradiancia que es reflejada y que por tanto no penetra en la columna de agua. Por lo general A ≈ 0,07 para el agua oceánica. Por otro lado, los coeficientes de atenuación de la luz K(λ) son los que determinan el tipo óptico del agua, su valor crece desde el tipo I hacia el III. En nuestro trabajo tomamos los valores iniciales de K(λ) suministrados en [3], pero optimizados mediante interpolación lineal en [4]. N. Jerlov, inicialmente definió la existencia de tres grandes tipos de agua oceánica:

- I (claras u oligotróficas),
- II (intermedias o mesotróficas) y
- III (turbias o eutróficas).

1.2.1 Modelo para el cálculo de la tasa de fotosíntesis

La capa superior del océano, en que la irradiancia es suficiente para hacer posible la fotosíntesis, es llamada zona fótica. Hoy día alcanza hasta 200 metros de profundidad en las aguas más claras. Por otro lado, la circulación oceánica hace que la parte superior de la zona fótica se mantenga bien mezclada, por lo que las propiedades físicas tales como densidad y temperatura son constantes. La profundidad de esta capa mezclada depende de la localidad, pero por lo general es de decenas de metros. La biota viviente en la capa mezclada del océano es resistente a las radiaciones. Estos organismos deberían tener buenas capacidades de reparación, de modo que empleamos el llamado modelo E que aparece en [5] para estimar las tasas de fotosíntesis en la capa mezclada del océano.

La tasa de fotosíntesis P en este modelo se calcula mediante:

$$P = P_{Pot}\left(\frac{1}{1+E^{*}\ inh}\right) \qquad (1.5)$$

Donde E^{*}_{inh} es la irradiancia inhibitoria adimensional, dada por la radiación ultravioleta UV, mientras que P_{pot} es la tasa de fotosíntesis en ausencia de fotoinhibición por el UV, calculada mediante:

$$P_{pot} = P_s(1 - e^{-E_{PAR}/E_S}) \qquad (1.6)$$

En la expresión anterior, Ps es la tasa de saturación de la fotosíntesis en ausencia de inhibición; E_{PAR} (en $W.m^{-2}$) es la irradiancia de la luz visible, mientras que E_s (en $W.m^{-2}$) es un parámetro del modelo que representa el valor de irradiancia de E_{PAR} para el que la fotosíntesis alcanza el 63% de su máximo valor posible. Mientras menos sea el valor de E_S, mayor es la eficiencia del aparato fotosintético, pues se alcanza el 63% de la tasa máxima de fotosíntesis con una irradiancia menor. El valor de E_S para una gran cantidad de especies fluctúa entre 15 y 25 W/m^2, aunque existen especies para los que E_S cae fuera de ese rango. La irradiancia inhibitoria adimensional de radiación ultravioleta está dada por:

$$E^*inh = \sum_{200nm}^{400nm} E(\lambda)\,\varepsilon_E(\lambda)\Delta\lambda \qquad (1.7)$$

Donde $\varepsilon(\lambda)$. (en $W^{-1}.m^2$) son los pesos biológicos que cuantifican la efectividad o impacto biológico de la exposición espectral E(λ) (en $W^{-2}.m^{-2}.nm$), es decir, representa la inhibición biológica efectiva de la fotosíntesis causada por la radiación ultravioleta de longitud de onda λ.

Sustituyendo la ecuación 1.6 en la 1.5 y normalizando respecto a Ps se obtiene que:

$$\frac{P}{Ps} = \frac{1-e^{-E_{PAR}/E_S}}{1+E^*_{inh}} \qquad (1.8)$$

Ahora queda claro que la tasa de fotosíntesis es una combinación de dos factores: el numerador en la ecuación anterior favorece la fotosíntesis (pues la irradiancia de la radiación fotosintéticamente activa PAR está allí), mientras que el denominador la inhibe, debido a la presencia del factor inhibitorio de irradiancia ultravioleta.

1.3 Herramientas computacionales usadas en el sistema

1.3.1 HTML

HTML, siglas de HyperText Markup Language (Lenguaje de Marcado de Hipertexto), es el lenguaje de marcado predominante para la elaboración de páginas web. Se usa para

describir la estructura y el contenido en forma de texto, así como para complementar el texto con objetos tales como imágenes. HTML se escribe en forma de «etiquetas», rodeadas por corchetes angulares (<,>). HTML también puede describir, hasta un cierto punto, la apariencia de un documento, y puede incluir un script (por ejemplo Javascript), el cual puede afectar el comportamiento de navegadores web y otros procesadores de HTML.[6]

1.3.2 CSS

Las Hojas de Estilo en Cascada (Cascading Style Sheets), son un mecanismo simple que describe cómo se va a mostrar un documento en la pantalla, o cómo se va a imprimir, o incluso cómo va a ser pronunciada la información presente en ese documento a través de un dispositivo de lectura. Esta forma de descripción de estilos ofrece a los desarrolladores el control total sobre estilo y formato de sus documentos.

CSS se utiliza para dar estilo a documentos HTML y XML, separando el contenido de la presentación. Los *Estilos* definen la forma de mostrar los elementos HTML y XML. CSS permite a los desarrolladores Web controlar el estilo y el formato de múltiples páginas Web al mismo tiempo. Cualquier cambio en el estilo marcado para un elemento en la CSS afectará a todas las páginas vinculadas a esa CSS en las que aparezca ese elemento.

1.3.3 JavaScript

JavaScript es un lenguaje de programación interpretado, dialecto del estándar ECMAScript. Se define como orientado a objetos, basado en prototipos, imperativo, débilmente tipado y dinámico.

Se utiliza principalmente en su forma del lado del cliente (client-side), implementado como parte de un navegador web permitiendo mejoras en la interfaz de usuario y páginas web dinámicas, aunque existe una forma de JavaScript del lado del servidor (Server-side JavaScript o SSJS). Su uso en aplicaciones externas a la web, por ejemplo en documentos PDF, aplicaciones de escritorio (mayoritariamente widgets) es también significativo.

JavaScript se diseñó con una sintaxis similar al C, aunque adopta nombres y convenciones del lenguaje de programación Java. Sin embargo Java y JavaScript no están relacionados y tienen semánticas y propósitos diferentes.

Todos los navegadores modernos interpretan el código JavaScript integrado en las páginas web. Para interactuar con una página web se provee al lenguaje JavaScript de una implementación del Document Object Model (DOM).

Tradicionalmente se venía utilizando en páginas web HTML para realizar operaciones y únicamente en el marco de la aplicación cliente, sin acceso a funciones del servidor. JavaScript se interpreta en el agente de usuario, al mismo tiempo que las sentencias van descargándose junto con el código HTML. [7]

1.3.4 AJAX

Ajax, siglas de Asynchronous JavaScript and XML, es un término que describe un nuevo acercamiento a usar un conjunto de tecnologías existentes juntas, incluyendo las siguientes: HTML o XHTML, hojas de estilo (Cascading Style Sheets o css), Javascript, el DOM (Document Object Model), XML, XSLT, y el objeto XMLHttpRequest.

Cuando se combinan estas tecnologías en el modelo Ajax, las aplicaciones funcionan mucho más rápido, ya que las interfaces de usuario se pueden actualizar por partes sin tener que actualizar toda la página completa.

Es una técnica de desarrollo web para crear aplicaciones interactivas o RIA (*Rich Internet Applications*). Estas aplicaciones se ejecutan en el cliente, es decir, en el navegador de los usuarios mientras se mantiene la comunicación con el servidor en segundo plano. De esta forma es posible realizar cambios sobre las páginas sin necesidad de recargarlas, lo que significa aumentar la interactividad, velocidad y usabilidad en las aplicaciones.

Ajax es una tecnología asíncrona, en el sentido de que los datos adicionales se requieren al servidor y se cargan en segundo plano sin interferir con la visualización ni el comportamiento de la página. Javascrip es el lenguaje interpretado (scripting language) en el que normalmente se efectúan las funciones de llamada de Ajax mientras que el acceso a los datos se realiza mediante XML HttpRequest, objeto disponible en los

navegadores actuales. En cualquier caso, no es necesario que el contenido asíncrono esté formateado en XML. Ajax es una técnica válida para múltiples plataformas y utilizable en muchos sistemas operativos y navegadores, dado que está basado en estándares abiertos como JavaScript y Document Object Model (DOM).

Tecnologías incluidas en Ajax

Ajax es una combinación de cuatro tecnologías ya existentes:

- XHTML (o HTML) y hojas de estilo en cascada (CSS) para el diseño que acompaña a la información.
- Document Object Model (DOM) accedido con un lenguaje de scripting por parte del usuario, especialmente implementaciones ECMAScript como JavaScript y JSScript, para mostrar e interactuar dinámicamente con la información presentada.
- El objeto XML HttpRequest para intercambiar datos de forma asíncrona con el servidor web. En algunos frameworks y en algunas situaciones concretas, se usa un objeto `iframe` en lugar del XMLHttpRequest para realizar dichos intercambios.
- XML es el formato usado generalmente para la transferencia de datos solicitados al servidor, aunque cualquier formato puede funcionar, incluyendo HTML preformateado, texto plano, JSON y hasta EBML.

Como el DHTML, LAMP o SPA, Ajax no constituye una tecnología en sí, sino que es un término que engloba a un grupo de éstas que trabajan conjuntamente. [8]

1.3.5 Django

Django es un framework de desarrollo web de código abierto, escrito en Python, que cumple en cierta medida el paradigma del Modelo Vista Controlador. Fue desarrollado en origen para gestionar varias páginas orientadas a noticias de la World Company de Lawrence, Kansas, y fue liberada al público bajo una licencia BSD en julio de 2005; el framework fue nombrado en alusión al guitarrista de jazz gitano Django Reinhardt En Junio del 2008 fue anunciado que la recién formada Django Software Foundation se hará

cargo de Django en el futuro. La meta fundamental de Django es facilitar la creación de sitios web complejos. Django pone énfasis en el re-uso, la conectividad y extensibilidad de componentes, del desarrollo rápido y del principio de DRY (del inglés *Don't Repeat Yourself*). Python es usado en todas las partes del framework, incluso en configuraciones, archivos, y en los modelos de datos.

- Los orígenes de Django en la administración de páginas de noticias son evidentes en su diseño, ya que proporciona una serie de características que facilitan el desarrollo rápido de páginas orientadas a contenidos. Por ejemplo, en lugar de requerir que los desarrolladores escriban controladores y vistas para las áreas de administración de la página, Django proporciona una aplicación incorporada para administrar los contenidos, que puede incluirse como parte de cualquier página hecha con Django y que puede administrar varias páginas hechas con Django a partir de una misma instalación; la aplicación administrativa permite la creación, actualización y eliminación de objetos de contenido, llevando un registro de todas las acciones realizadas sobre cada uno, y proporciona una interfaz para administrar los usuarios y los grupos de usuarios (incluyendo una asignación detallada de permisos).

Otras características de Django son:

- Un mapeador objeto-relacional.
- Aplicaciones "enchufables" que pueden instalarse en cualquier página gestionada con Django.
- Una API de base de datos robusta.
- Un sistema incorporado de "vistas genéricas" que ahorra tener que escribir la lógica de ciertas tareas comunes.
- Un sistema extensible de plantillas basado en etiquetas, con herencia de plantillas.
- Un despachador de URLs basado en expresiones regulares.
- Aunque Django está fuertemente inspirado en la filosofía de desarrollo Modelo Vista Controlador, sus desarrolladores declaran públicamente que no se sienten

especialmente atados a observar estrictamente ningún paradigma particular, y en cambio prefieren hacer "lo que les parece correcto". Como resultado, por ejemplo, lo que se llamaría "controlador" en un "verdadero" framework MVC se llama en Django "vista", y lo que se llamaría "vista" se llama "plantilla".

- Gracias al poder de las capas mediator y foundation, Django permite que los desarrolladores se dediquen a construir los objetos Entity y la lógica de presentación y control para ellos.
- Aquí se maneja la interacción entre el usuario y el computador. En Django, ésta tarea la realizan el template engine y el template loader que toman la información y la presentan al usuario (vía HTML, por ejemplo). El sistema de configuración de URLs es también parte de la capa de presentación.

Control

En esta capa reside el programa o la lógica de aplicación en sí. En Django son representados por las views y manipulators. La capa de presentación depende de ésta y a su vez ésta lo hace de la capa de dominio.

Mediator

Es el encargado de manejar la interacción entre el subsistema Entity y foundation. Aquí se realiza el mapeo objeto-relacional a cargo del motor de Django.

Entity

El subsistema entity maneja los objetos de negocio. El mapeo objeto-relacional de Django permite escribir objetos de tipo entity de una forma fácil y estándar.

Foundation

La principal tarea del subsistema foundation es la de manejar a bajo nivel el trabajo con la base de datos. Se provee soporte a nivel de foundation para varias bases de datos y otras están en etapa de prueba.

Soporte de bases de datos

Respecto a la base de datos, la recomendada es PostgreSQL, pero también son soportadas MySQL y SQLite 3. Se encuentra en desarrollo un adaptador para Microsoft SQL Server. Una vez creados los data models, Django proporciona una abstracción de la base de datos a través de su API que permite crear, recuperar, actualizar y borrar objetos. También es posible que el usuario ejecute sus propias consultas SQL directamente. En el modelo de datos de Django, una clase representa una tabla en la base de datos y las instancias de esta serán las filas en la tabla.

Soporte de servidores Web

Como mencionamos en los requisitos, Django incluye un servidor web liviano para realizar pruebas y trabajar en la etapa de desarrollo. En la etapa de producción, sin embargo, se recomienda Apache 2 con mod_python. Aunque Django soporta la especificación WSGI, por lo que puede correr sobre una gran variedad de servidores como FastCGI o SCGI en Apache u otros servidores (particularmente Lighttpd).

Requerimientos

Django requiere Python 2.5 o superior. No se necesitan otras bibliotecas de Python para poder obtener una funcionalidad básica. En un entorno de desarrollo –especialmente si queremos experimentar con Django— no necesitamos un web server instalado, ya que Django trae su propio servidor liviano para éste propósito, con la restricción de solo permitir un usuario a la vez..

- Base de datos PostgreSQL, MySQL, Oracle o SQLite.[9]

1.3.6 Python

Python es un lenguaje de programación de alto nivel cuya filosofía hace hincapié en una sintaxis muy limpia y que favorezca un código legible.

Se trata de un lenguaje de programación multiparadigma ya que soporta orientación a objetos, programación imperativa y, en menor medida, programación funcional. Es un lenguaje interpretado, usa tipado dinámico, es fuertemente tipado y multiplataforma.

Es administrado por la Python Software Foundation. Posee una licencia de código abierto, denominada Python Software Foundation License, que es compatible con la

Licencia pública general de GNU a partir de la versión 2.1.1, e incompatible en ciertas versiones anteriores.

Python es un lenguaje de programación multiparadigma. Esto significa que más que forzar a los programadores a adoptar un estilo particular de programación, permite varios estilos: programación orientada a objetos, programación imperativa y programación funcional. Otros paradigmas están soportados mediante el uso de extensiones.

Python usa tipado dinámico y conteo de referencias para la administración de memoria.

Una característica importante de Python es la resolución dinámica de nombres; es decir, lo que enlaza un método y un nombre de variable durante la ejecución del programa (también llamado ligadura dinámica de métodos).

Otro objetivo del diseño del lenguaje es la facilidad de extensión. Se pueden escribir nuevos módulos fácilmente en C o C++. Python puede incluirse en aplicaciones que necesitan una interfaz programable.

Aunque la programación en Python podría considerarse en algunas situaciones hostil a la programación funcional tradicional del Lisp, existen bastantes analogías entre Python y los lenguajes minimalistas de la familia Lisp como puede ser Scheme.[10]

1.3.7 Jquery

jQuery es una biblioteca o framework de JavaScript, creada inicialmente por John Resig, que permite simplificar la manera de interactuar con los documentos HTML, manipular el árbol DOM, manejar eventos, desarrollar animaciones y agregar interacción con la técnica AJAX a páginas web. Fue presentada el 14 de enero de 2006 en el BarCamp NYC. **jQuery** es software libre y de código abierto, posee un doble licenciamiento bajo la Licencia MIT y la Licencia Pública General de GNU v2, permitiendo su uso en proyectos libres y privativos. jQuery, al igual que otras bibliotecas, ofrece una serie de funcionalidades basadas en JavaScript que de otra manera requerirían de mucho más código, es decir, con las funciones propias de esta biblioteca se logran grandes resultados en menos tiempo y espacio.

Las empresas Microsoft y Nokia anunciaron que incluirán la biblioteca en sus plataformas.Microsoft la añadirá en su IDE Visual Studio y la usará junto con los frameworks ASP.NET AJAX y ASP.NET MVC, mientras que Nokia los integrará con su plataforma Web Run-Time.

Características

- Selección de elementos DOM.
- Interactividad y modificaciones del árbol DOM, incluyendo soporte para CSS 1-3 y un plugin básico de XPath.
- Eventos.
- Manipulación de la hoja de estilos CSS.
- Efectos y animaciones.
- Animaciones personalizadas.
- AJAX.
- Soporta extensiones.
- Utilidades varias como obtener información del navegador, operar con objetos y vectores, funciones como trim() (elimina los espacios en blanco del principio y final de una cadena de caracteres), etc.
- Compatible con los navegadores Mozilla Firefox 2.0+, Internet Explorer 6+, Safari 3+, Opera 10.6+ y Google Chrome 8+.

1.3.8 Patrón de arquitectura Modelo-Vista-Controlador (MVC)

Modelo Vista Controlador (MVC) es un patrón de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos. El patrón de llamada y retorno MVC (según CMU), se ve frecuentemente en aplicaciones web, donde la vista es la página HTML y el código que provee de datos dinámicos a la página. El modelo es el Sistema de Gestión de Base de Datos y la Lógica de negocio, y el controlador es el responsable de recibir los eventos de entrada desde la vista.

- **Modelo**: Esta es la representación específica de la información con la cual el sistema opera. En resumen, el modelo se limita a lo relativo de la *vista* y su *controlador* facilitando las presentaciones visuales complejas. El sistema también puede operar con más datos no relativos a la presentación, haciendo uso integrado de otras lógicas de negocio y de datos afines con el sistema modelado.
- **Vista**: Este presenta el modelo en un formato adecuado para interactuar, usualmente la interfaz de usuario.
- **Controlador**: Este responde a eventos, usualmente acciones del usuario, e invoca peticiones al modelo y, probablemente, a la vista.

Muchos de los sistemas informáticos utilizan un Sistema de Gestión de Base de Datos para gestionar los datos: en líneas generales del **MVC** corresponde al modelo. La unión entre capa de presentación y capa de negocio conocido en el paradigma de la Programación por capas representaría la integración entre **Vista** y su correspondiente **Controlador** de eventos y acceso a datos, MVC no pretende discriminar entre capa de negocio y capa de presentación pero si pretende separar la capa visual gráfica de su correspondiente programación y acceso a datos, algo que mejora el desarrollo y mantenimiento de la Vista y el Controlador en paralelo, ya que ambos cumplen ciclos de vida muy distintos entre sí.

Aunque se pueden encontrar diferentes implementaciones de **MVC**, el flujo que sigue el control generalmente es el siguiente:

1. El usuario interactúa con la interfaz de usuario de alguna forma (por ejemplo, el usuario pulsa un botón, enlace, etc.)
2. El controlador recibe (por parte de los objetos de la interfaz-vista) la notificación de la acción solicitada por el usuario. El controlador gestiona el evento que llega, frecuentemente a través de un gestor de eventos (handler) o callback.
3. El controlador accede al modelo, actualizándolo, posiblemente modificándolo de forma adecuada a la acción solicitada por el usuario (por ejemplo, el controlador actualiza el carro de la compra del usuario). Los controladores complejos están a

menudo estructurados usando un patrón de comando que encapsula las acciones y simplifica su extensión.

4. El controlador delega a los objetos de la vista la tarea de desplegar la interfaz de usuario. La vista obtiene sus datos del modelo para generar la interfaz apropiada para el usuario donde se reflejan los cambios en el modelo (por ejemplo, produce un listado del contenido del carro de la compra). El modelo no debe tener conocimiento directo sobre la vista. Sin embargo, se podría utilizar el patrón Observador para proveer cierta indirección entre el modelo y la vista, permitiendo al modelo notificar a los interesados de cualquier cambio. Un objeto vista puede registrarse con el modelo y esperar a los cambios, pero aun así el modelo en sí mismo sigue sin saber nada de la vista. El controlador no pasa objetos de dominio (el modelo) a la vista aunque puede dar la orden a la vista para que se actualice. Nota: En algunas implementaciones la vista no tiene acceso directo al modelo, dejando que el controlador envíe los datos del modelo a la vista.
5. La interfaz de usuario espera nuevas interacciones del usuario, comenzando el ciclo nuevamente.

Este patrón de diseño utilizado está estrechamente relacionado con la arquitectura cliente servidor elegida para el sistema. En el patrón de diseño el elemento vista corresponde al usuario, el elemento modelo a la base de datos y el controlador a la interfaz web.[11]

1.1 Consideraciones finales

En este capítulo se describen los fundamentos matemáticos del modelo matemático para el cálculo de las tasas de fotosíntesis en ecosistemas acuáticos, así como las herramientas computacionales fundamentales que se utilizaron en la creación del software elaborado.

Capítulo 2 Análisis, diseño e implementación del sistema

En este capítulo se explicarán brevemente las fases de análisis, diseño e implementación, los actores y casos de uso del sistema. Se mostrará la ingeniería del software y el diseño de la base de datos correspondiente, la captura de los requisitos funcionales y no funcionales, los diagramas de actividad, despliegue, así como las herramientas propias utilizadas en el diseño.

2.1 Etapa de Análisis

Durante esta etapa, se analizan los requisitos que se describen en la captura de requisitos, refinándolos y estructurándolos. El objetivo de hacerlo es conseguir una comprensión más precisa de los requisitos y una descripción de los mismos que sea fácil de mantener y que ayude a la estructuración del sistema completo.[11]

La función del Análisis puede ser dar soporte a las actividades del negocio, o desarrollar un producto que pueda venderse para generar beneficios. Para conseguir este objetivo, un Sistema basado en computadoras hace uso de seis elementos fundamentales:

- Software, que son Programas de computadora, con estructuras de datos y su documentación que hacen efectiva la logística metodología o controles de requerimientos del Programa.
- Hardware, dispositivos electrónicos y electromecánicos, que proporcionan capacidad de cálculos y funciones rápidas, exactas y efectivas (Computadoras, Censores, maquinarias, bombas, lectores, etc.), que proporcionan una función externa dentro de los Sistemas.
- Personal, son los operadores o usuarios directos de las herramientas del Sistema.
- Base de Datos, una gran colección de informaciones organizadas y enlazadas al Sistema a las que se accede por medio del Software.

- Documentación, Manuales, formularios, y otra información descriptiva que detalla o da instrucciones sobre el empleo y operación del Programa.
- Procedimientos, o pasos que definen el uso específico de cada uno de los elementos o componentes del Sistema y las reglas de su manejo y mantenimiento.

Un Análisis de Sistema se lleva a cabo teniendo en cuenta los siguientes objetivos en mente:

- Identificar las necesidades del Cliente.
- Evaluar qué conceptos tiene el cliente del sistema para establecer su viabilidad.
- Realizar un Análisis Técnico y económico.
- Asignar funciones al Hardware, Software, personal, base de datos, y otros elementos del Sistema.
- Establecer las restricciones de presupuestos y planificación temporal.
- Crear una definición del sistema que forme el fundamento de todo el trabajo de Ingeniería.

Para lograr estos objetivos se requiere tener un gran conocimiento y dominio del Hardware y el Software, así como de la Ingeniería humana (Manejo y Administración de personal), y administración de base de datos.

2.1.1 Objetivos del Análisis

Identificación de Necesidades

Es el primer paso del análisis del sistema, en este proceso el Analista se reúne con el cliente y/o usuario (un representante institucional, departamental o cliente particular), e identifican las metas globales, se analizan las perspectivas del cliente, sus necesidades y requerimientos, sobre la planificación temporal y presupuestal, líneas de mercadeo y otros puntos que puedan ayudar a la identificación y desarrollo del proyecto.

Algunos autores suelen llamar a esta parte ***Análisis de Requisitos*** y lo dividen en cinco partes:

- Reconocimiento del problema.
- Evaluación y Síntesis.
- Modelado.
- Especificación.
- Revisión
- Antes de su reunión con el analista, el cliente prepara un documento conceptual del proyecto, aunque se recomienda que este se elabore durante la comunicación cliente – analista, ya que de hacerlo el cliente solo de todas maneras tendría que ser modificado, durante la identificación de las necesidades. Las especificaciones de los requisitos del software se producen en la terminación de la tarea del análisis.
- En conclusión un proyecto de desarrollo de un sistema comprende varios componentes o pasos llevados a cabo durante la etapa del análisis, el cual ayuda a traducir las necesidades del cliente en un modelo de Sistema que utiliza uno más de los componentes: Software, hardware, personas, base de datos, documentación y procedimientos.

2.2 Etapa de Diseño del software

La importancia del Diseño del Software se puede definir en una sola palabra ***Calidad***, dentro del diseño es donde se fomenta la calidad del Proyecto. El Diseño es la única manera de materializar con precisión los requerimientos del cliente. El Diseño del Software es un proceso y un modelado a la vez. El proceso de Diseño es un conjunto de pasos repetitivos que permiten al diseñador describir todos los aspectos del Sistema a construir. A lo largo del diseño se evalúa la calidad del desarrollo del proyecto con un conjunto de revisiones técnicas:

- El diseño debe implementar todos los requisitos explícitos contenidos en el modelo de análisis y debe acumular todos los requisitos implícitos que desea el cliente.

- El Diseño debe proporcionar una completa idea de lo que es el Software, enfocando los dominios de datos, funcional y comportamiento desde el punto de vista de la Implementación.
- Para evaluar la calidad de una presentación del diseño, se deben establecer criterios técnicos para un buen diseño como son:
- Un diseño debe presentar una organización jerárquica que haga un uso inteligente del control entre los componentes del software.
- El diseño debe ser modular, es decir, se debe hacer una partición lógica del Software en elementos que realicen funciones y subfunciones especificas.
- Un diseño debe contener abstracciones de datos y procedimientos.
- Debe producir módulos que presenten características de funcionamiento independiente.
- Debe conducir a interfaces que reduzcan la complejidad de las conexiones entre los módulos y el entorno exterior.
- Debe producir un diseño usando un método que pudiera repetirse según la información obtenida durante el análisis de requisitos de Software.

Estos criterios no se consiguen por casualidad. El proceso de Diseño del Software exige buena calidad a través de la aplicación de principios fundamentales de diseño, metodología sistemática y una revisión exhaustiva.[12]

2.2.1 Diseño de Bases de Datos

El diseño es el proceso mediante el cual se debe asegurar o garantizar antes de construirlo que un objeto trabaje bien, o sea, cumpla los requerimientos o funcionalidades que se espera del mismo. Mediante el diseño se obtiene un esquema del objeto a construir especificado en un determinado lenguaje de modelación. Esencialmente, en el diseño se expresan las ideas en papeles antes de construir realmente un objeto, y quizás también experimentar trasladando partes y piezas para ver como ajustan o lucen.

Una base de datos, se debe diseñar antes de construirla, de llenarla de datos y asociarla a las aplicaciones. El proceso de diseño de una base de datos es el conjunto de etapas necesarias para obtener un modelo de los datos de un dominio de aplicación. El diseño correcto de la base de datos es de suma importancia ya que todas las aplicaciones que utilicen la base de datos dependerán completamente de la estructura de la misma.

Los objetivos del diseño de la base de datos son:

- soportar los requisitos de procesamiento y objetivos de rendimiento como tiempo de respuesta, tiempo de procesamiento, espacio de almacenamiento, entre otros.
- conseguir un esquema flexible de BD, es decir tal que sea posible modificarlo fácilmente (como consecuencia de cambios en los requisitos del sistema) una vez implementada la BD.

Para construir una Base de Datos se deben tener en cuenta:

1-Captura de Requisitos: Es la recolección de información acerca de la naturaleza de los datos, propiedades requeridas, y cualquier necesidad especializada tal como respuestas de salida esperada. Este proceso se realiza mediante entrevistas a los clientes y empleados de compañía, lo cual permite obtener una mejor idea de lo que necesitan exactamente.

2-Diseño conceptual: En este paso se crea el modelo de la base de datos en un lenguaje de alto nivel generalmente grafico, que sirve también como medio de intercambio de criterios con los usuarios potenciales de la base de tatos. En este paso se pueden utilizar herramientas graficas. Este paso incluye creación de tablas, campos dentro de las tablas, y las interrelaciones entre las tablas. Este paso también incluye normalización

3-Diseño lógico: En el diseño lógico se crean comandos del lenguaje de base de datos para generar las definiciones de las tablas. Algunas herramientas de las que se pueden utilizar para el diseño conceptual permiten la generación de script del lenguaje de definición de datos. Estos script pueden ser muy genéricos por lo que debe comprobarlos antes de ejecutarlos en cualquier SGBD (Sistema Gestor de Bases de Datos).

4-Diseño Físico: Aquí se determina la organización de archivos y las estructuras de almacenamiento interno.

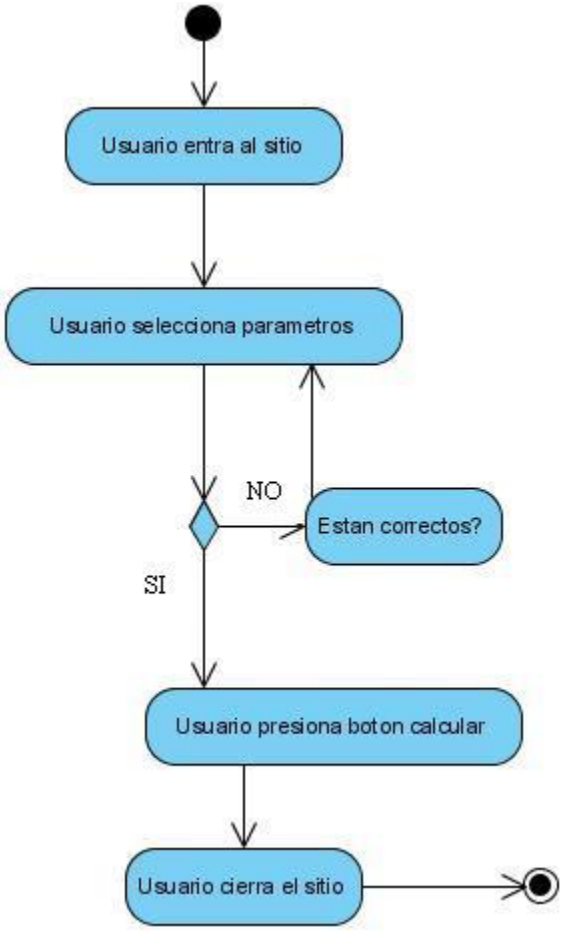


Figura 2.1 Diagrama de actividad

En esta figura se muestra la secuencia de actividades realizadas por el actor usuario al interactuar con el sistema, de entrada a salida.

Clases ¿Por qué no se pone un diagrama de clases?

La Base de Datos consta de cuatro clases, ninguna de ellas relacionadas, una clase representa una tabla de la Base de Datos y las instancias de esta serán las filas en la tabla:

- 1-Atenuación: En esta clase implementamos los coeficientes de atenuación (Kd) en dependencia del tipo de agua y del rango de longitud de onda en el que estemos trabajando.

- 2-Factor: En esta clase agregamos los factores de corrección según el hemisferio donde trabajemos y en la latitud dada (en grados).
- 3-Radiación: En esta clase agregamos las irradiancias, ya sean de la luz visible o la ultravioleta también dependiendo de la longitud de onda.
- 4-Galería: En esta clase se muestran imágenes que hacen referencia a los ecosistemas costeros existentes.

2.2.2 Diseño del sitio web

La tecnología y 1os efectos visuales constituyen la base de la pirámide del desafío Web; ambos son necesarios y tienen que estar directamente relacionados con la finalidad del sitio. En lugar de implementar primero y hacer preguntas después, es preferible analizar la finalidad del sitio y, posteriormente, determinar cómo conseguir cualquier objetivo definido. En primer lugar, hay que considerar la finalidad del sitio. Antes de construir un sitio se debe definir cuidadosamente el asunto del que se va a ocupar o el objetivo que esté tratando de conseguir. Conocer bien la finalidad reducir el riesgo de que el proyecto falle y ayudar a determinar si es razonable, incluso, la construcción del propio sitio. Después de determinar el objetivo o 1os objetivos del sitio, deberá elaborarse una especificación. La especificación incluirá todos 1os requisitos del sitio y estimar la audiencia de forma muy cuidadosa. A continuación, debería realizar un diseño del sitio. El desafío puede incluir tanto el prototipo técnico como el visual. Una vez finalizado el desafío, el sitio debería implantarse y probarse.

Los buenos sitios Web son aquellos realmente provechosos para sus usuarios. Puede pensarse en este concepto como en una combinación de utilidad y facilidad de empleo.

La utilidad comprende todas las funciones del sitio que satisfacen las necesidades del usuario. La facilidad de empleo es la facilidad de la que dispone el usuario para manejar las funciones del sitio con el fin de conseguir un determinado objetivo.[6]

2.3 Etapa de implementación del sistema

En la implementación empezamos con el resultado del diseño e implementamos el sistema en términos de componentes, es decir, ficheros de código fuente, scripts, ficheros de código binario, ejecutables y similares.

Afortunadamente la mayor parte de la arquitectura del sistema se captura durante el diseño, siendo el propósito principal de la implementación el desarrollar la arquitectura y el sistema como un todo. De forma más específica, los propósitos de la implementación son:

- Planificar las integraciones de sistema necesarias en cada iteración. Seguimos para ello un enfoque incremental, lo que da lugar a un sistema que se implementa en una sucesión de pasos pequeños y manejables.
- Distribuir el sistema asignando componentes ejecutables a nodos en el diagrama de despliegue. Esto se basa fundamentalmente en las clases activas encontradas durante el diseño.
- Implementar las clases y subsistemas encontrados durante el diseño. En particular, las clases se implementan como componentes de fichero que contienen código fuente.
- Probar los componentes individualmente, y a continuación integrarlos compilándolos y enlazándolos en uno o más ejecutables, antes de ser enviados para ser integrados y llevar a cabo las comprobaciones de sistema.[11]

2.4 Actores y Casos de Uso del sistema

Este epígrafe comenta los casos de uso y actores del sistema elaborado.

2.4.1 Casos de Uso

Un caso de uso es un fragmento de funcionalidad del sistema que proporciona al usuario un resultado importante. Los casos de uso representan los requisitos funcionales. Todos los casos de uso juntos constituyen el **modelo de casos de uso** el cual describe la funcionalidad total del sistema los casos de uso no son solo una herramienta para especificar los requisitos de un sistema. También guían su diseño, implementación, y

prueba; esto es, guían el proceso de desarrollo. Basándose en el modelo de casos de uso, los desarrolladores crean una serie de modelos de diseño e implementación que llevan a cabo los casos de uso. Los desarrolladores revisan cada uno de los sucesivos modelos para que sean conformes al modelo de casos de uso. Los ingenieros de prueba prueban la implementación para garantizar que los componentes del modelo de implementación implementan correctamente los casos de uso. De este modo, los casos de uso no solo inician el proceso de desarrollo sino que le proporcionan un hilo conductor. Un caso de uso es una forma en que un actor usa el sistema. Los casos de uso son fragmentos de funcionalidad que el sistema ofrece para aportar un resultado de valor observable para sus actores. De manera específica, un caso de uso especifica una secuencia de acciones que el sistema puede llevar a cabo interactuando con sus actores,

Por tanto, un caso de uso es una especificación. Especifica el comportamiento de cosas dinámicas, en este caso, de instancia de los casos de uso. Un caso de uso representa un rol que un actor juega accediendo a él. [11]

La figura 2.2 muestra los casos de uso del sistema elaborado.

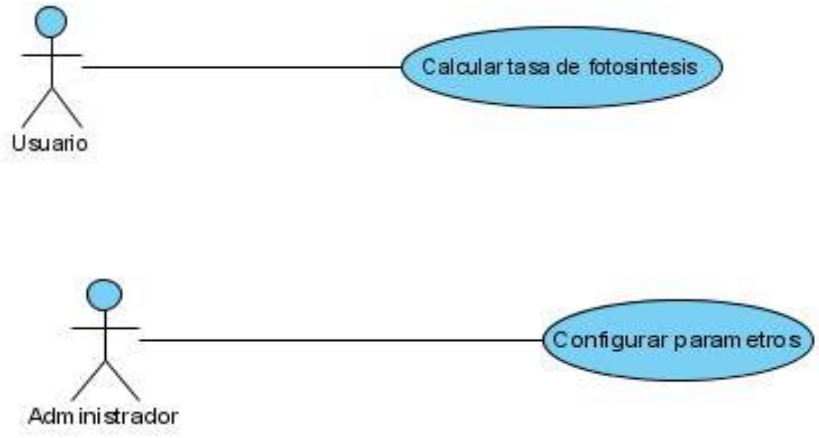


Figura 2.2 Casos de uso del sistema

Caso de uso: (Calcular tasa de fotosíntesis) realizado por el actor usuario.

Curso normal de los eventos: Calcular tasa de fotosíntesis	
Acción del actor	Respuesta del sistema

1 Escribe login y contraseña.	2 Si están correctas se establece la conexión con la base de datos y entra al sitio
3Entra al sitio	
4 Selecciona el tipo de agua con la que va a trabajar. 5 Selecciona a la profundidad que va a trabajar. 6 Selecciona latitud y hemisferio 7 Selecciona irradiancia superficial	8 Si están correctos los datos los muestra la página web.
9 presiona el botón calcular	10 Se muestra el resultado
11 cierra el sitio	
Flujos Alternativos	
Acción 2 si no están correctos los datos	Se muestra un mensaje de error y se introduce un nuevo dato, en caso de estar incorrecto se muestra nuevamente el mensaje de error.

2.4.2 Actores del sistema

Por definición tendremos que un actor es una persona (usuario) o sistema que interactúa con el sistema en desarrollo, con el propósito de lograr un beneficio.

Un actor puede desempeñar varios roles. Cada rol estará representado por un caso de uso al que accede el actor.

- Actor usuario: Es el que utiliza el software, el que selecciona los datos a su interés para realizar el cálculo.
- Actor administrador: Es el encargado de supervisar que el software este funcionando correctamente. Es el que introduce los datos al sistema y a su vez es el que modifica los mismos..

2.5 Requisitos Funcionales del Sistema

La técnica inmediata para identificar los requisitos del sistema se basa en los casos de uso. Cada usuario quiere que el sistema haga algo para él o ella, es decir, que lleve a cabo

ciertos casos de uso. Para el usuario, un caso de uso es un modo de utilizar el sistema. En consecuencia, si los analistas pueden describir todos los casos de uso que necesita el usuario, entonces saben lo que debe hacer el sistema. La captura de los casos de uso que realmente se quieren para el sistema, como aquellos que soportaran el negocio y que el usuario piensa que le permiten trabajar "cómodamente", requiere que conozcamos en profundidad las necesidades del usuario y del cliente. Para hacerlo tenemos que comprender el contexto del sistema, entrevistar a los usuarios, discutir propuestas. [11]

Requisitos funcionales:

- El sistema debe permitir que el usuario seleccione el tipo de agua en el que desea trabajar.
- El sistema debe permitir que el usuario introduzca a la profundidad con la que quiere trabajar.
- El sistema debe permitir que el usuario seleccione el hemisferio con el que quiere trabajar.
- El sistema debe permitir que el usuario seleccione en los grados que quiere trabajar.
- El sistema debe permitir el cálculo de la tasa de fotosíntesis
- El sistema permitirá la autenticación del usuario administrador
- El sistema permitirá que el usuario administrador introduzca los valores de los coeficientes de atenuación por tipo de agua.
- El sistema permitirá que el usuario administrador modifique los valores de los coeficientes de atenuación.
- El sistema permitirá que el usuario administrador pueda introducir los valores correspondientes a las irradiancias tanto de la luz visible como de las longitudes del ultravioleta.
- El sistema permitirá que el usuario administrador pueda modificar los valores correspondientes a las irradiancias tanto de la luz visible como de las longitudes del ultravioleta.
- El sistema permitirá que el usuario administrador introduzca los factores de corrección respecto a las latitudes y hemisferios.

- El sistema permitirá que el usuario administrador modifique los factores de corrección respecto a las latitudes y hemisferios.
- El sistema presentará una ventana de error cuando se le introduzca un valor no establecido en un rango dado.

2.6 Requisitos no funcionales del sistema

Los requisitos no funcionales especifican propiedades del sistema, como restricciones del entorno o de la implementación, rendimiento, dependencias de la plataforma, facilidad de mantenimiento, extensibilidad, y fiabilidad. La fiabilidad hace referencia a características como la disponibilidad, exactitud, tiempo medio entre fallos, defectos por miles de líneas de código (KLDC), y defectos por clase. Un requisito de rendimiento impone condiciones sobre los requisitos funcionales como la velocidad, rendimiento, tiempo de respuesta, y uso de memoria.

Requisitos no funcionales

- El sistema debe estar escrito en Python.
- El sistema debe estar hecho en el framework Django.
- El sistema debe contar con una interfaz web.
- El sistema es fiable
- El sistema debe contar con autentificación para modificar los datos.
- El sistema debe responder en un máximo de 5 segundos ante cualquier actividad que no pueda realizar.

Requisitos Adicionales

Los requisitos adiciónales son fundamentalmente requisitos no funcionales que no pueden asociarse a ningún caso de uso en concreto en cambio cada uno de estos requisitos tiene impacto en varios casos de uso o en ninguno. Algunos ejemplos son el rendimiento, las interfaces y los requisitos de diseño físico, así como las restricciones arquitectónicas, de diseño y de implementación.

Los requisitos adiciónales se capturan de forma muy parecida a como se hacía en la especificación de requisitos tradicional, es decir, como una lista de requisitos. Después se utilizan durante el análisis y el diseño.

Un requisito de interfaz especifica la interfaz con un elemento externo con el cual debe interactuar el sistema, o que establece restricciones condicionantes en formatos, tiempos, u otros factores de relevancia en esa interacción

Un requisito físico especifica una característica física que debe poseer un sistema, como su material, forma, tamaño o peso.

Requisitos de plataforma Hardware:

Una restricción de diseño limita el diseño de un sistema, como lo hacen las restricciones de extensibilidad y mantenibilidad, o las restricciones relativas a la reutilización de sistemas heredados o partes esenciales de los mismos.

Una restricción de implementación especifica o limita la codificación o construcción de un sistema. Son ejemplos los estándares requeridos, las normas de codificación, los lenguajes de programación, políticas para la integridad de la base de datos, limitaciones de recurso, y entornos operativos.

2.7 Arquitectura cliente-servidor

Cuando se modela la arquitectura de un sistema se capturan las decisiones sobre los requisitos del sistema, sus elementos lógicos y físicos. Al modelar aspectos estructurales y de comportamiento, también se modelan las diferentes vistas

La arquitectura está condicionada por los casos de uso que queremos que soporte el sistema; los casos de uso son directores de la arquitectura.

La arquitectura del software abarca decisiones importantes sobre:

- La organización del sistema software.
- Los elementos estructurales que compondrán el sistema y sus interfaces, junto con sus comportamientos, tal y como se especifican en las colaboraciones entre estos elementos.

- Se necesita una arquitectura para:
- Comprender el sistema.
- Organizar el desarrollo
- Fomentar la reutilización
- Hacer evolucionar el sistema

En este caso la arquitectura seleccionada para la realización del software es la *arquitectura cliente-servidor*.

En la arquitectura cliente/servidor (C/S), un subsistema, el **servidor**, proporciona servicios a instancias de los demás subsistemas llamados **clientes**.

La petición de un servicio se realiza por lo general, por un mecanismo de llamada a procedimiento remoto. El flujo de control en los clientes y el servidor es independiente, a excepción de la sincronización para administrar las peticiones o para recibir los resultados.

Aunque esta idea se puede aplicar a programas que se ejecutan sobre una sola computadora es más ventajosa en un sistema operativo multiusuario distribuido a través de una red de computadoras.

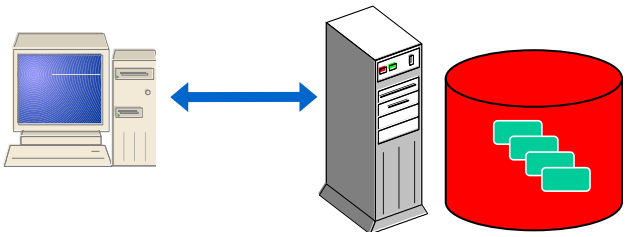


Figura 2.3 Arquitectura Cliente-Servidor

En esta arquitectura la capacidad de proceso está repartida entre los clientes y los servidores, aunque son más importantes las ventajas de tipo organizativo debidas a la centralización de la gestión de la información y la separación de responsabilidades, lo que facilita y clarifica el diseño del sistema.

La separación entre cliente y servidor es una separación de tipo lógico, donde el servidor no se ejecuta necesariamente sobre una sola máquina ni es necesariamente un sólo

programa. Los tipos específicos de servidores incluyen los servidores web, los servidores de archivo, los servidores del correo, etc. Mientras que sus propósitos varían de unos servicios a otros, la arquitectura básica seguirá siendo la misma.

En especial para la navegación por Internet, un cliente puede tener acceso a datos de miles de servidores diferentes.

Las arquitecturas cliente/servidor son apropiadas para los sistemas distribuidos que manejan grandes cantidades de datos.[11]

Ventajas

- Centralización del control: los accesos, recursos y la integridad de los datos son controlados por el servidor de forma que un programa cliente defectuoso o no autorizado no pueda dañar el sistema. Esta centralización también facilita la tarea de poner al día datos u otros recursos (mejor que en las redes P2P).
- Escalabilidad: se puede aumentar la capacidad de clientes y servidores por separado. Cualquier elemento puede ser aumentado (o mejorado) en cualquier momento, o se pueden añadir nuevos nodos a la red (clientes y/o servidores).
- Fácil mantenimiento: al estar distribuidas las funciones y responsabilidades entre varios ordenadores independientes, es posible reemplazar, reparar, actualizar, o incluso trasladar un servidor, mientras que sus clientes no se verán afectados por ese cambio (o se afectarán mínimamente). Esta independencia de los cambios también se conoce como encapsulación.
- Existen tecnologías, suficientemente desarrolladas, diseñadas para el paradigma de C/S que aseguran la seguridad en las transacciones, la amigabilidad de la interfaz, y la facilidad de empleo.

2.8 Herramientas utilizadas en el desarrollo del diseño

2.8.1 UML

El Lenguaje Unificado de Modelado (Unified Modeling Language, UML) es un lenguaje estándar para escribir planos de software. UML puede utilizarse para visualizar,

especificar, construir y documentar los artefactos de un sistema que involucra una gran cantidad de software. UML es apropiado para modelar desde sistemas de información en empresas hasta aplicaciones distribuidas basadas en la Web, e incluso para sistemas empotrados de tiempo real muy exigentes. Es un lenguaje muy expresivo, que cubre todas las vistas necesarias para desarrollar y luego desplegar tales sistemas. Aunque sea expresivo, UML no es difícil de aprender ni de utilizar. Aprender a aplicar UML de modo eficaz comienza por crear un modelo conceptual del lenguaje, lo cual requiere aprender tres elementos principales: los bloques básicos de construcción de UML, las reglas que dictan cómo pueden combinarse esos bloques y algunos mecanismos comunes que se aplican a lo largo del lenguaje. UML es sólo un lenguaje y por tanto es tan sólo una parte de un método de desarrollo de software. UML es independiente del proceso, aunque para utilizarlo óptimamente se debe a usar en un proceso que fue dirigido por los casos de uso, centrado en la arquitectura, iterativo e incremental.

UML no es un lenguaje de programación visual, pero sus modelos pueden conectarse de forma directa a una gran variedad de lenguajes de programación. Esto significa que es posible establecer correspondencias desde un modelo UML a un lenguaje de programación como Java, C++ o Visual Basic, o incluso a tablas en una base de datos relaciona o al almacenamiento persistente en una base de datos orientada a objetos. Las cosas que se expresan mejor gráficamente también se representan gráficamente en UML, mientras que las cosas que se expresan mejor textualmente se plasman con el lenguaje de programación. Esta correspondencia permite ingeniería directa: la generación de código a partir de un modelo UML en un lenguaje de programación. Lo contrario también es posible: se puede reconstruir un modelo en UML a partir de una implementación. La ingeniería inversa no es magia. A menos que se codifique esa información en la implementación, la información se pierde cuando se pasa de los modelos al código. La ingeniería inversa requiere, por lo tanto, herramientas que la soporten e intervención humana. La combinación de estas dos vías de generación de código y de ingeniería inversa produce una ingeniería *"de ida y vuelta"*, entendiendo por esto la posibilidad de trabajar en una vista gráfica o textual, mientras las herramientas mantienen la consistencia entre las dos vistas.

Además de esta correspondencia directa, UML es lo suficientemente expresivo y no ambiguo como para permitir la ejecución directa de modelos, la simulación de sistemas y la instrumentación de sistemas en ejecución.[13]

2.8.2 Visual paradigm

Visual Paradigm para UML es una herramienta UML profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. El software de modelado UML ayuda a una más rápida construcción de aplicaciones de calidad, mejores y a un menor coste. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación. [11]

Características

- Producto de calidad.
- Soporta aplicaciones web.
- Las imágenes y reportes generados, no son de muy buena calidad.
- Varios idiomas.
- Generación de código para Java y exportación como HTML.
- Fácil de instalar y actualizar.
- Compatibilidad entre ediciones.

2.8.3 Editor TopStyle_v4

TopStyle es un editor de hojas de estilo de gran alcance para la creación de sitios compatibles con los estándares. Le permite editar el código HTML, XHTML y CSS en un solo programa. Ofrece muchas características únicas, incluyendo la opción de actualizar sus documentos HTML mediante la sustitución de marcas obsoletas con un estilo equivalente. También puede convertir HTML a XHTML y comprobar su sintaxis CSS contra varios navegadores, con un lado a ver el lado. Una característica colores armoniosos, le permite crear fácilmente sistemas de agradable color para su sitio y más.

TopStyle Pro ofrece una gestión completa del sitio con el construido en el Explorador de archivos, bibliotecas de clips y administrador de recursos.

TopStyle contiene poderosas herramientas para la creación de sitios web, compatible con los estándares. TopStyle te permite guardar el documento con codificación ANSI (el TopStyle 3.x por defecto), o con codificación Unicode (UTF-8 o UTF-16). Al abrir un documento existente, TopStyle detectará automáticamente la codificación para el documento que está intentando abrir. El nuevo Panel del Explorador de FTP que permite editar documentos en línea a través de FTP. Al guardar el documento, es cargado automáticamente a su sitio FTP. TopStyle viene con una nueva barra de herramientas HTML configurable por el usuario que se muestra sobre el editor. Basta con arrastrar y soltar cualquier fragmento de esta barra de herramientas HTML en su documento. TopStyle incluye nuevas definiciones CSS para IE8 (Microsoft Internet Explorer 8), FF3 (Mozilla Firefox 3), y SF3 (Apple Safari 3).

Características principales:

- Vista previa de CSS mientras se escribe
- Fácil de crear esquemas de color agradable para su sitio
- Estilo Checker valida la sintaxis CSS contra varios navegadores
- Estilo de actualización rápidamente reemplaza todos los anticuados código HTML

2.8.4 SQLite

SQLite es un sistema de gestión de bases de datos relacional compatible con ACID, contenida en una relativamente pequeña biblioteca en C. SQLite es un proyecto de dominio público creado por D. Richard Hipp.

A diferencia de los sistemas de gestión de bases de datos cliente-servidor, el motor de SQLite no es un proceso independiente con el que el programa principal se comunica. En lugar de eso, la biblioteca SQLite se enlaza con el programa pasando a ser parte integral del mismo. El programa utiliza la funcionalidad de SQLite a través de llamadas simples a subrutinas y funciones. Esto reduce la latencia en el acceso a la base de datos, debido a que las llamadas a funciones son más eficientes que la comunicación entre procesos. El

conjunto de la base de datos (definiciones, tablas, índices, y los propios datos), son guardados como un sólo fichero estándar en la máquina host. Este diseño simple se logra bloqueando todo el fichero de base de datos al principio de cada transacción.

En su versión 3, **SQLite** permite bases de datos de hasta 2 Terabytes de tamaño, y también permite la inclusión de campos tipo BLOB.

El autor de SQLite ofrece formación, contratos de soporte técnico y características adicionales como compresión y cifrado.

La biblioteca implementa la mayor parte del estándar SQL-92, incluyendo transacciones de base de datos atómicas, consistencia de base de datos, aislamiento, y durabilidad (ACID), triggers y la mayor parte de las consultas complejas.

SQLite usa un sistema de tipos inusual. En lugar de asignar un tipo a una columna como en la mayor parte de los sistemas de bases de datos SQL, los tipos se asignan a los valores individuales. Por ejemplo, se puede insertar un *string* en una columna de tipo entero (a pesar de que SQLite tratará en primera instancia de convertir la cadena en un entero). Algunos usuarios consideran esto como una innovación que hace que la base de datos sea mucho más útil, sobre todo al ser utilizada desde un lenguaje de scripting de tipos dinámicos. Otros usuarios lo ven como un gran inconveniente, ya que la técnica no es portable a otras bases de datos SQL. SQLite no trataba de transformar los datos al tipo de la columna hasta la versión 3.

Varios procesos o hilos pueden acceder a la misma base de datos sin problemas. Varios accesos de lectura pueden ser servidos en paralelo. Un acceso de escritura sólo puede ser servido si no se está sirviendo ningún otro acceso concurrentemente. En caso contrario, el acceso de escritura falla devolviendo un código de error (o puede automáticamente reintentarse hasta que expira un timeout configurable). Esta situación de acceso concurrente podría cambiar cuando se está trabajando con tablas temporales. Sin embargo, podría producirse un deadlock debido al multithread

Existe un programa independiente de nombre sqlite que puede ser utilizado para consultar y gestionar los ficheros de base de datos SQLite. También sirve como ejemplo para la escritura de aplicaciones utilizando la biblioteca SQLite.

2.9 Diagrama de Despliegue

Un diagrama de despliegue modela la arquitectura en tiempo de ejecución de un sistema. Esto muestra la configuración de los elementos de hardware (nodos) y muestra cómo los elementos y artefactos del software se trazan en esos nodos.

Nodo

Un Nodo es un elemento de hardware o software. Esto se muestra con la forma de una caja en tres dimensiones, como se muestra a continuación.

Figura 2.4 Diagrama de despliegue

2.10 Consideraciones finales

En este capítulo se explicaron las fases de análisis, diseño e implementación, los actores y casos de uso del sistema elaborado. Se mostraron los aspectos de ingeniería del software y el diseño de la base de datos correspondientes, la captura de los requisitos funcionales y no funcionales, los diagramas de actividad, despliegue, así como las herramientas que se utilizaron en el diseño.

Capítulo 3 Manual de usuario. Validación inicial

En este capítulo se mostrará el manual de usuario del software desarrollado para una mejor manipulación por parte de los usuarios. Se presentará una muestra del proceso llevado a cabo por los usuarios antes de la automatización del sistema y el mismo ejemplo luego de ser automatizado.

3.1 Manual de usuario

El sistema automatizado de ecosistemas acuáticos está diseñado para una mayor comodidad de la comunidad científica de nuestro país. Con este sistema se pretende ahorrar cuantiosos recursos materiales y facilitar el trabajo a esas personas que día a día hacen posible que nuestras aguas sean más útiles para la vida.

3.1.1 Características generales del sistema

El sistema automatizado de ecosistemas acuáticos cuenta con una interfaz web para cada una de las funcionalidades del mismo, con un ambiente agradable y fácil de trabajar para aquellas personas que no son especialistas en el campo de la computación. Este sistema permite modificar cualquiera de sus campos en caso de que se extienda un poco más el área de investigación. Incluye una galería de imágenes donde ilustra a través de fotos los diferentes tipos de ecosistemas existentes. Permite el cálculo de la tasa de fotosíntesis con rapidez y precisión.

3.1.2 Requerimientos mínimos

El sistema debe ser instalado a través de Python 2.6. Para su correcta utilización se necesita incorporarle algunas bibliotecas para su funcionamiento. Se requiere un servidor web y uno para la base de datos pero ambos los trae incorporado el framework Django.

Los usuarios pueden acceder al sistema desde cualquier tipo de navegador, preferiblemente Internet Explorer o Mozilla Firefox.

3.1.3 Instalación

El proceso de instalación es el siguiente:

- Instalar Python 2.6
- Instalar PIL 1.1.7
- Instalar pysqlite-2.6.0
- Copiar la instalación de Django en la carpeta de instalación del Python
- Copiar en el cmd la dirección donde está el proyecto
- Copiar en consola el comando **manage.py runserver** para que esté funcionando el Django.
- Abrir cualquier explorador y asignarle el http con la dirección del localhost.

3.1.4 Descripción de la página inicial del sitio

La interfaz que muestra la página principal muestra una breve explicación de los ecosistemas acuáticos y su clasificación. A demás el menú correspondiente con los campos: **Inicio**, **Función**, **Imágenes** y **Administración**.

En el primero: **Inicio** es la presentación del software y la que permite acceder a las otras tres.

En la segunda: **Función**, se realiza el cálculo de la tasa de fotosíntesis, en esta página se permite la selección por parte del usuario de los datos con los cuales quiere trabajar. Este selecciona una profundidad, una irradiancia superficial(ES), un hemisferio y una latitud, pulsando el botón calcular se muestra el resultado del cálculo.

En la tercera: **Imágenes**, se muestra una galería de imágenes sobre los diversos ecosistemas acuáticos existentes.

En la cuarta: **Administración**, sólo puede accederse por el administrador porque es la página donde se entran los datos propios del sistema, se pueden modificar los existentes, así como agregarles otros nuevos.

A continuación se muestran las páginas anteriormente descritas:

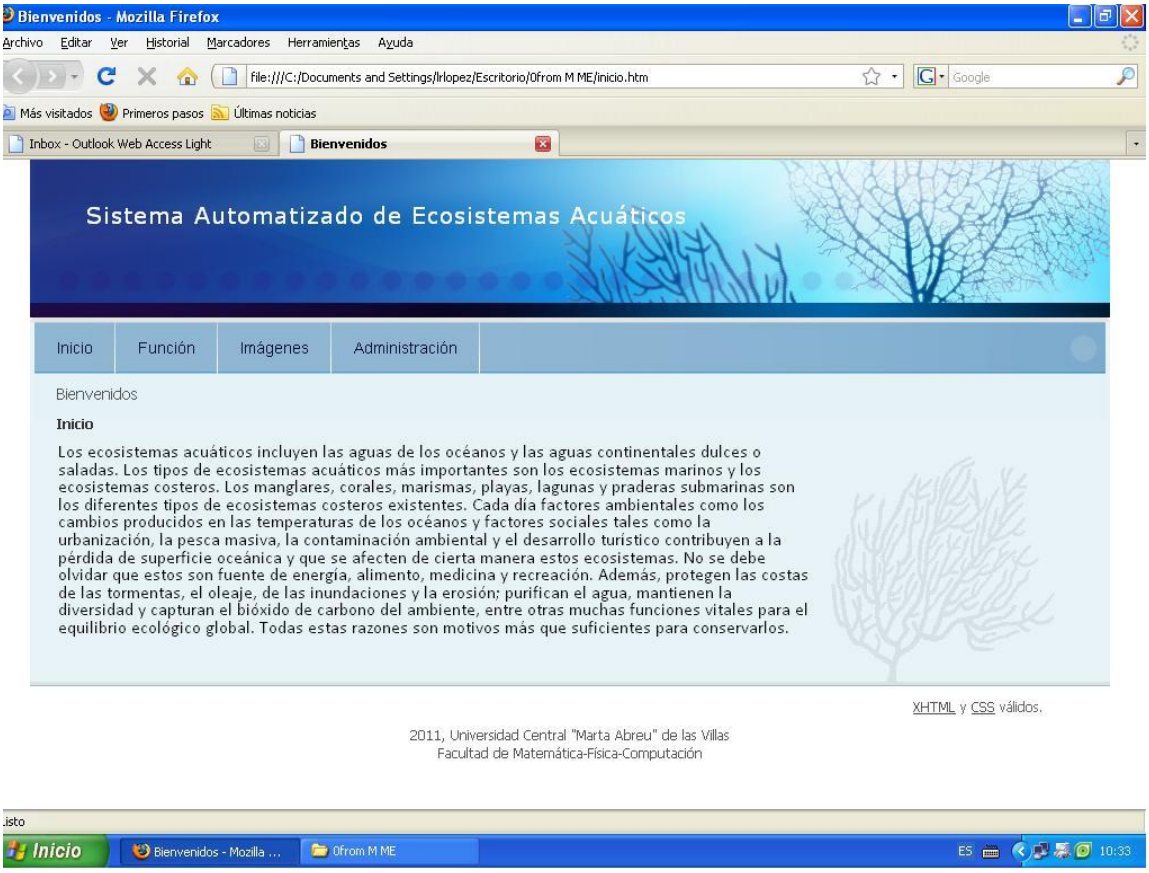


Figura 3.1 Página de inicio del Sistema Automatizado de Ecosistemas Acuáticos

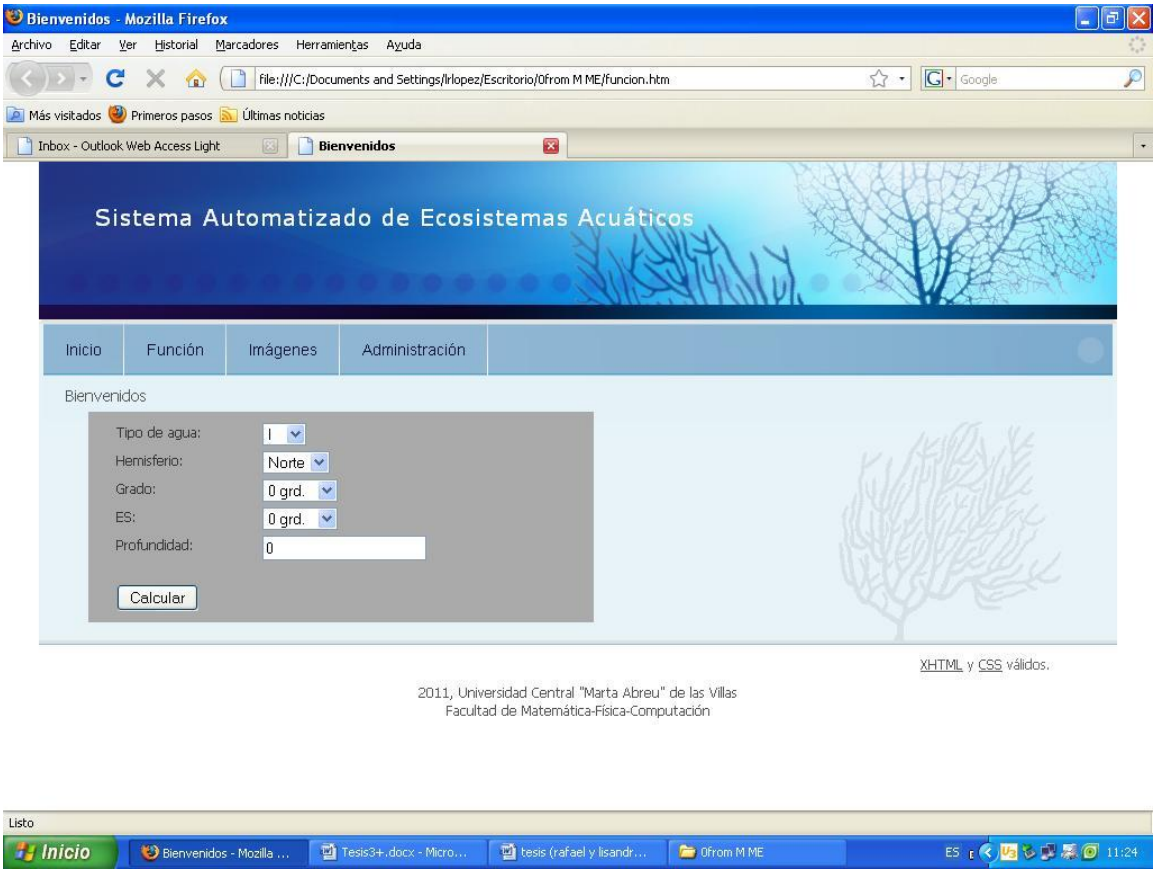


Figura 3.2 Página de Función del Sistema Automatizado de Ecosistemas Acuáticos

3.3 Validación inicial. Comparación con el trabajo anterior

En este epígrafe se muestra una validación sencilla al comparar los resultados del software propuesto con la hoja de cálculo que utilizan los biólogos con el mismo propósito.

Este ejemplo corresponde al cálculo de la tasa de fotosíntesis para el tipo de agua I, con una Es de 7, con un factor de corrección de 0.2, para el hemisferio nortea una latitud de 40 grados.

Como primer paso se calcula la ecuación del epígrafe 1.2.1

$$P = P_{Pot}\left(\frac{1}{1+E^{*}inh}\right) \qquad (1.5)$$

Sustituyendo:

$$P_{pot} = P_S(1 - e^{-E_{PAR}/E_S}) \qquad (1.6)$$

Susttituyendo:

$$Einh = \sum_{200nm}^{400nm} E(\lambda)\,\varepsilon_E(\lambda)\Delta\lambda$$

Se obtiene:

$$\frac{P}{Ps} = \frac{1-e^{-E_{PAR}/E_S}}{1+E^{*}_{inh}}$$

Este proceso resultaba engorroso calcularlo no sólo manualmente sino incluso con la herramienta Microsoft Excel. Se procedía de la siguiente forma:

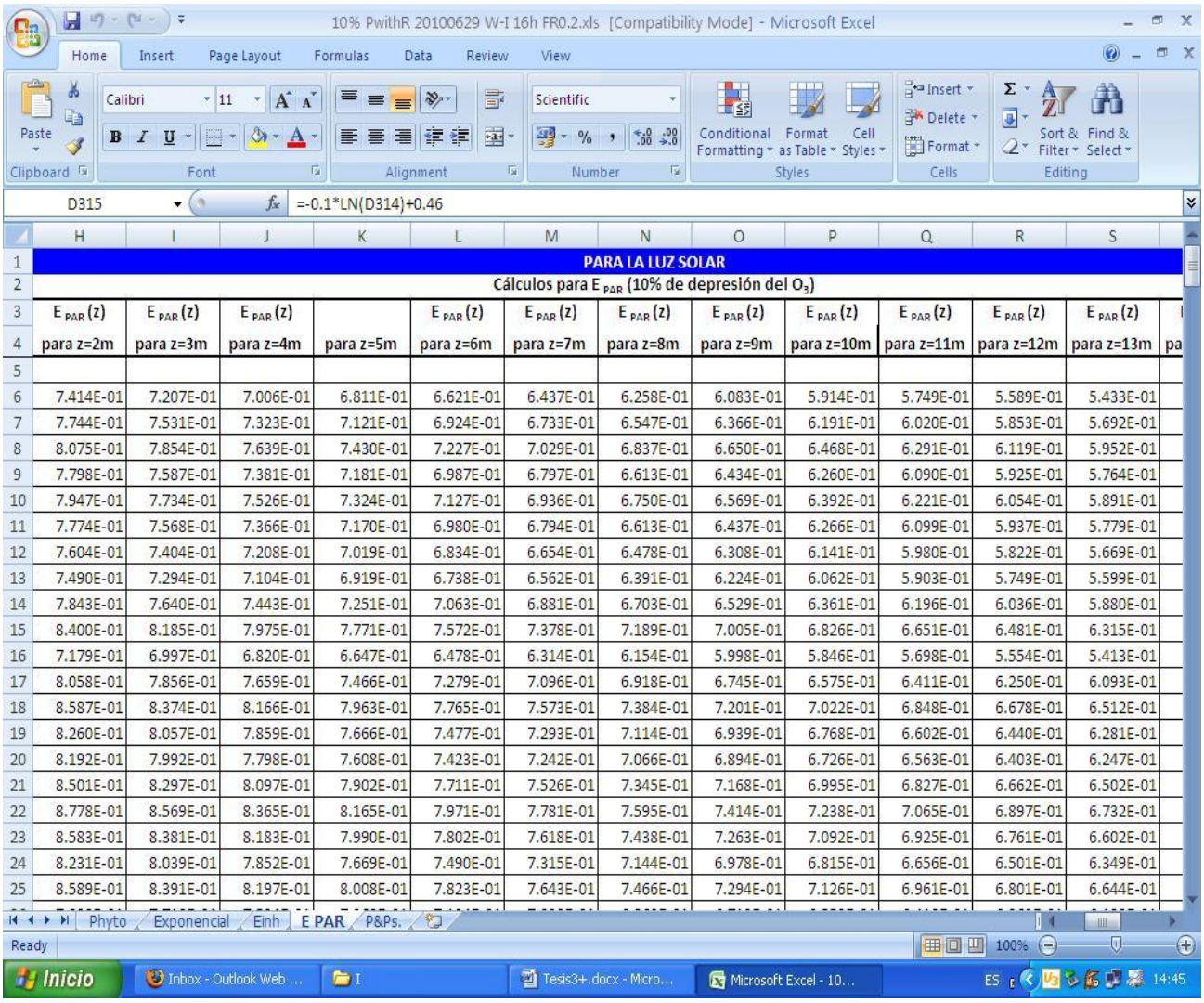


Figura 3.3 Hoja de cálculo de Microsoft Excel para calcular las irradiancias de la luz visible, a veces hasta 200 m de profundidad.

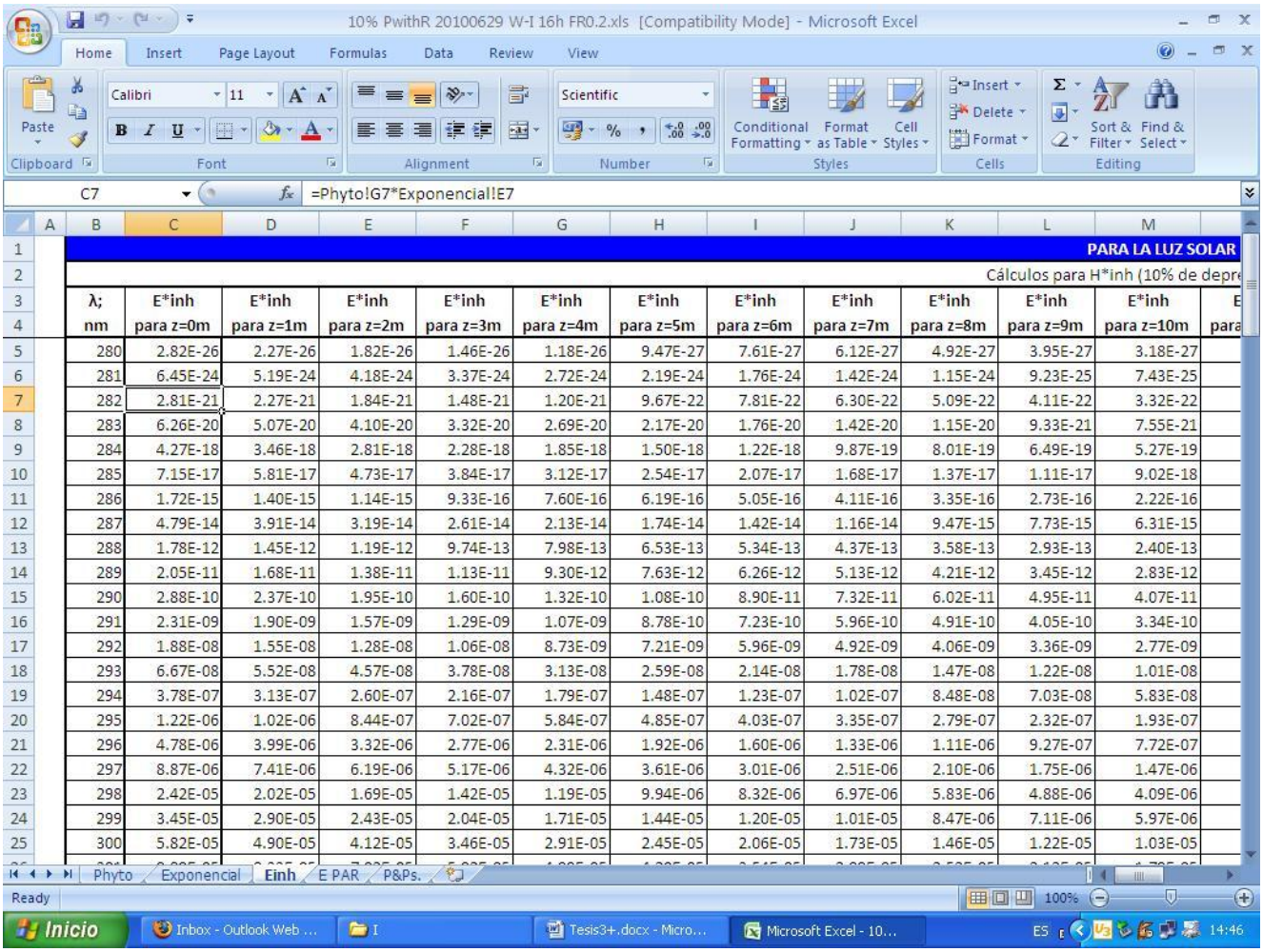
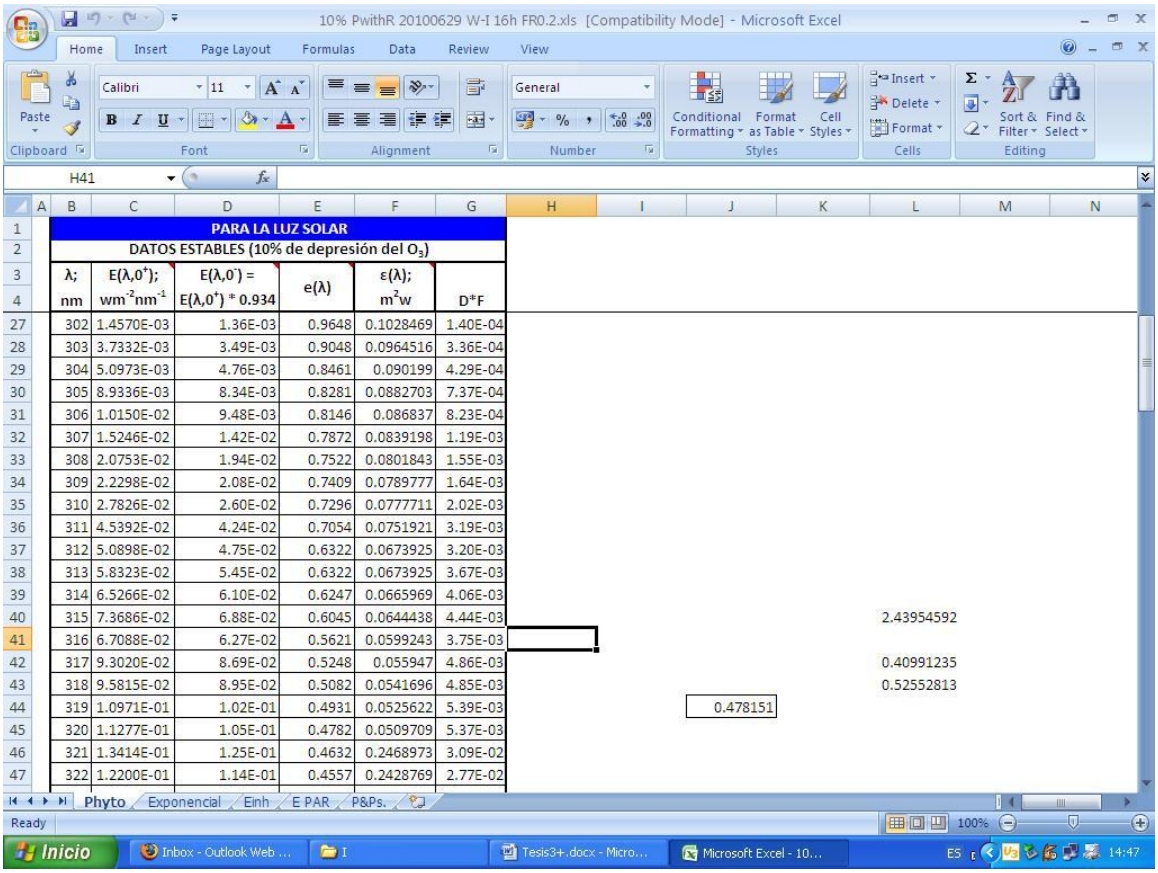


Figura 3.4 Hoja de cálculo de Microsoft Excel para calcular las irradiancias del ultravioleta.



	λ; nm	E(λ,0+); wm-2nm-1	E(λ,0-) = E(λ,0+) * 0.934	e(λ)	ε(λ); m2w	D*F
27	302	1.4570E-03	1.36E-03	0.9648	0.1028469	1.40E-04
28	303	3.7332E-03	3.49E-03	0.9048	0.0964516	3.36E-04
29	304	5.0973E-03	4.76E-03	0.8461	0.090199	4.29E-04
30	305	8.9336E-03	8.34E-03	0.8281	0.0882703	7.37E-04
31	306	1.0150E-02	9.48E-03	0.8146	0.086837	8.23E-04
32	307	1.5246E-02	1.42E-02	0.7872	0.0839198	1.19E-03
33	308	2.0753E-02	1.94E-02	0.7522	0.0801843	1.55E-03
34	309	2.2298E-02	2.08E-02	0.7409	0.0789777	1.64E-03
35	310	2.7826E-02	2.60E-02	0.7296	0.0777711	2.02E-03
36	311	4.5392E-02	4.24E-02	0.7054	0.0751921	3.19E-03
37	312	5.0898E-02	4.75E-02	0.6322	0.0673925	3.20E-03
38	313	5.8323E-02	5.45E-02	0.6322	0.0673925	3.67E-03
39	314	6.5266E-02	6.10E-02	0.6247	0.0665969	4.06E-03
40	315	7.3686E-02	6.88E-02	0.6045	0.0644438	4.44E-03
41	316	6.7088E-02	6.27E-02	0.5621	0.0599243	3.75E-03
42	317	9.3020E-02	8.69E-02	0.5248	0.055947	4.86E-03
43	318	9.5815E-02	8.95E-02	0.5082	0.0541696	4.85E-03
44	319	1.0971E-01	1.02E-01	0.4931	0.0525622	5.39E-03
45	320	1.1277E-01	1.05E-01	0.4782	0.0509709	5.37E-03
46	321	1.3414E-01	1.25E-01	0.4632	0.2468973	3.09E-02
47	322	1.2200E-01	1.14E-01	0.4557	0.2428769	2.77E-02

Figura 3.5 Hoja de cálculo de Microsoft Excel para calcular los espectros de acción biológica

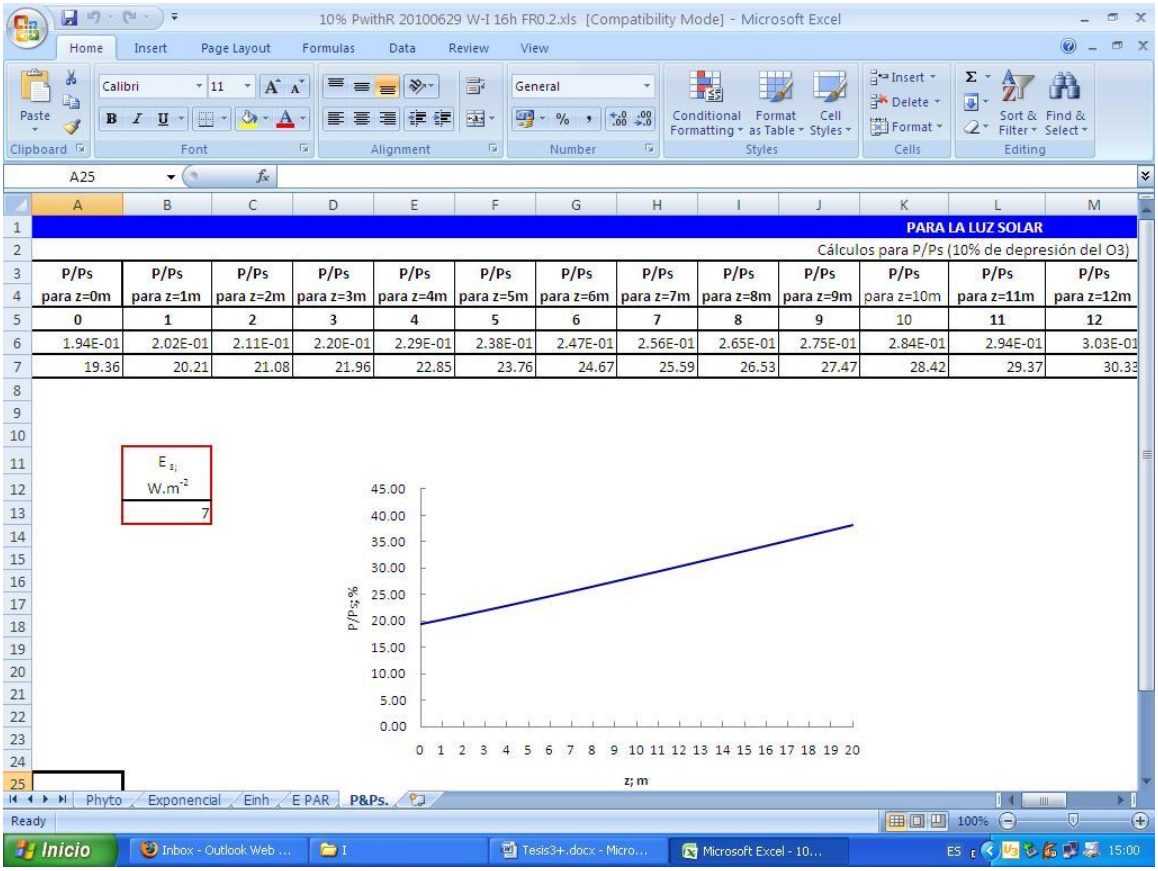


Figura 3.6 Hoja de cálculo de Microsoft Excel. Finalmente aquí se acumulaba toda la información en fracciones y se calculaba la tasa de fotosíntesis.

Como se puede apreciar este proceso resultaba muy difícil para un usuario inexperto en computación, como pudiera serlo un especialista en biología u otra área relacionada con estas investigaciones. Este trabajo realiza la automatización de este proceso.

A continuación se mostrara el mismo ejemplo pero esta vez ejecutado desde Django donde puede apreciarse a demás de rapidez menor riesgo de equivocación y una interfaz más sencilla y útil.

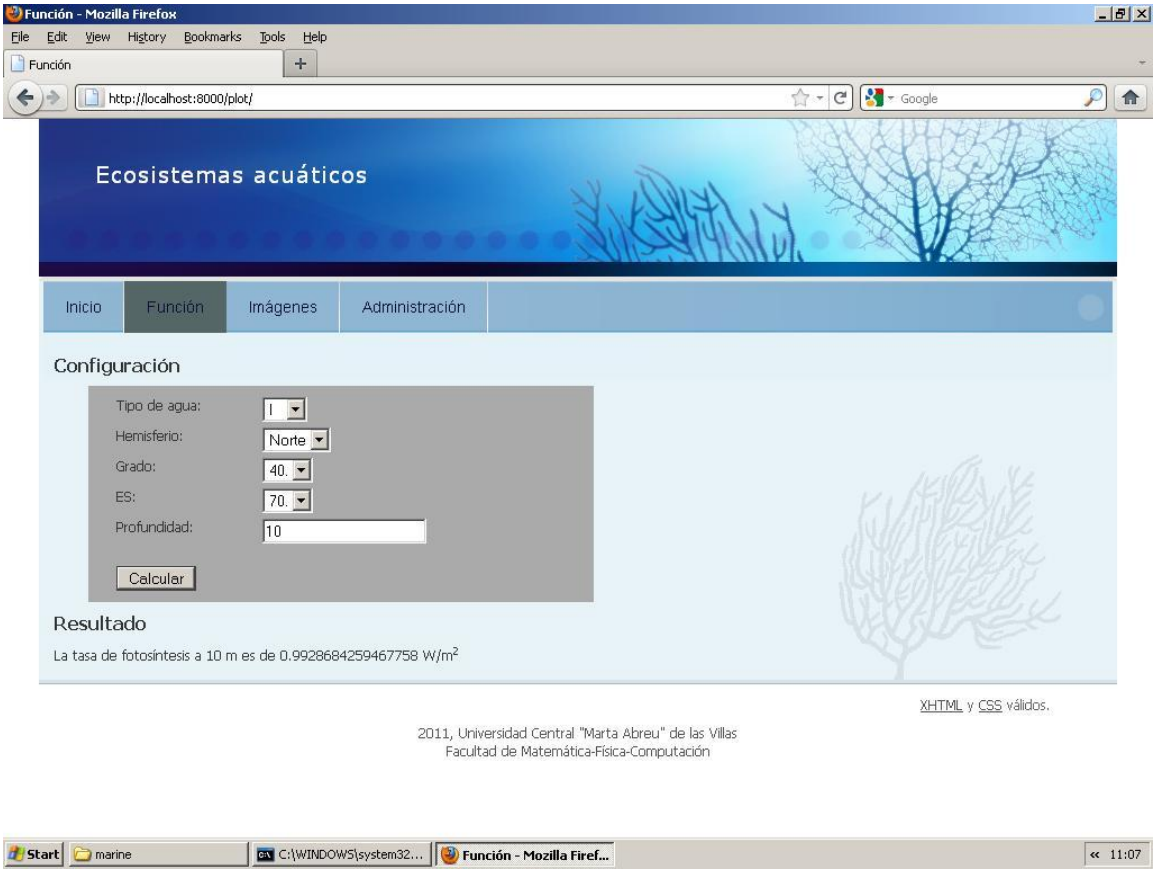


Figura 3.7 Página de Función de Ecosistemas Acuáticos.(Se calcula la tasa de fotosíntesis para los valores antes establecidos).

Como puede apreciarse este proceso tan complicado se logró automatizar y pasó a ser una operación sencilla con el nuevo software. Los resultaros coincidieron en todos los ejemplos probados.

3.4 Consideraciones finales

En este capítulo se muestra y explica el manual de usuario para este software, como los requisitos mínimos para la instalación del mismo. Se establece una comparación entre el proceso desarrollado por los biofísicos para calcular la tasa de fotosíntesis a través del Microsoft Excel y el mismo cálculo mediante el sistema desarrollado en Django.

CONCLUSIONES

Durante la etapa de creación e implementación de este software se obtuvo como resultados fundamentales:

- Se creó un software en Python que calcula la tasa de la fotosíntesis a diversas profundidades y a una latitud dada, cumpliendo así el principal objetivo de este trabajo.
- Se diseñó e implementó una base de datos que almacena los valores correspondientes a las características del modelo.
- Se diseñó e implementó un sitio web para realizar los cálculos del modelo.
- Se realizó una validación inicial sencilla que mostró la superioridad del software elaborado con respecto a la forma en la que los especialistas realizaban sus cálculos.

RECOMENDACIONES

Este no es un trabajo que concluye, debido a eso se realizan las siguientes recomendaciones:

- Incluir en el software desarrollado una biblioteca que permita la creación de gráficos para visualizar los resultados obtenidos mediante el cálculo.
- Agregar otros modelos para el cálculo de la tasa de fotosíntesis con otras características que también es utilizado por los biofísicos.

REFERENCIAS BIBLIOGRAFICAS

1. A. Day, T. and P.J. Neale, *Effects of UV-B Radiation on Terrestrial and Aquatic Primary Producers*. Annual Review of Ecology and Systematics, 2002. **Vol. 33**: p. 371-396
2. Cullen, J. and P. Neale, *Ultraviolet radiation, ozone depletion, and marine photosynthesis*. Photosynthesis Research, 1994. **39**(3): p. 303-320.
3. Jerlov, N.G., *Optical Classification of Ocean Water*. Physical Aspects of Light in the Sea. 1964, Hawaii. pp. 45–49.
4. Peñate, L., R. Cardenas, and O. Martin, *Short-term effects of gamma ray bursts on oceanic photosynthesis*. Astrophys Space Sci, 2010. **330**: p. 211-217.
5. Fritz, J., *Response of Antarctic phytoplankton to solar UVR exposure: inhibition and recovery of photosynthesis in coastal and pelagic assemblages*. Marine Ecology Progress Series, 2008. **365**: p. p.1-16.
6. Powell, T.A., *Diseño de Sitios Web*: Osborne MacGraw-Hill.
7. Weiss, A., *JavaScript Tutorial for Programmers* 1998.
8. CRANE, D., E. PASCARELLO, and D. JAMES, *Ajax in Action*. 2005, Greenwich: Manning.
9. Holovaty, A. and J. Kaplan-Moss, *The Definitive Guide to Django: Web Development Done Right*. 2008, New York: Copyright.
10. Brueck, D. and S. Tanner, *Python*. 2001, New York: Hungry Minds.
11. Jacobson, I., G. Booch, and J. Rumbaugh *El proceso unificado de desarrollode sofware*. 2000 España: PEARSON EDUCACIÓN, S, A.
12. Kendall, K. and J. Kendall, *Análisis y Diseño de Sistemas de Información* 2009, España: Pearson Educación.
13. Booch, G., J. Rumbaugh, and I. Jacobson, *El lenguaje unificado de modelado*, ed. T.A.-W.O. Technology. 2000, España.