



Universidad Autónoma de San Luis Potosí
Facultad de Ingeniería
Área de Ciencias de la Computación

Monitoreo de los biorepositorios del Laboratorio de Genómica Viral y Humana mediante un sistema web

Trabajo recepcional

Para obtener el grado de:
Ingeniería en informática

Presenta:
Julio César Álvarez Charqueño

Asesor:
Dr. Juan Carlos Cuevas Tello
Co-Asesor:
Dr. Christian Alberto García Sepúlveda

San Luis Potosí, S. L. P.

Febrero del 2023



Agradecimientos

Gracias a mis asesores el Dr. Juan Carlos y el Dr. Christian A. por darme su confianza para realizar este trabajo, por la gran paciencia que me tuvieron y a su disponibilidad para atender mis dudas. Al Dr. Juan Carlos por sus consejos académicos y profesionales que me guiaron en todo el proceso de este trabajo.

Gracias a mi pareja Adela Anai Rocha por todo su apoyo, consejos y cariño que me brindó cuando más lo necesité y por creer en mí para culminar este trabajo.

Gracias a mis padres Javier Manuel y Raquel por todo su amor incondicional, apoyo y por su confianza ya que sin ellos no hubiera logrado llegar a este punto de mi vida.

Gracias a mis hermanos Javier, Esther, Gabriel, Martín, Raquel y Gloria por ser un ejemplo a seguir que me motiva a seguir adelante.

Gracias a mis amigos por sus ánimos y consejos. A Alberto Villanueva que siempre me apoyo y aconsejó durante mis estudios.

Gracias a mis profesores no solo por sus enseñanzas si no por la motivación que me brindaron. Al profesor Raymundo González por facilitarme el espacio para realizar pruebas al sistema hecho para este trabajo.

Índice general

Resumen	4
Introducción	5
Planteamiento del problema	7
Hipótesis	7
Justificación	7
Objetivos	8
Estado del arte	9
Metodología	11
Sitio web	12
Estructura	13
1. Análisis y diseño del sistema web	15
1.1. Lenguajes	17
1.2. Frameworks y entornos de desarrollo	21
2. Análisis y diseño de la base de datos	24
2.1. Modelo relacional y no relacional	26
2.2. Gestores de bases de datos	31
3. Implementación del sistema de monitoreo	36
3.1. Requerimientos	36
3.2. Modelado	39
3.2.1. Casos de uso	39
3.2.2. Actividades	45
3.2.3. Clases	47
3.3. Diseño de arquitectura	48
3.3.1. Patrón arquitectónico	48

3.4. Diseño lógico de Base de datos	50
3.4.1. Entidad-relación	50
3.4.2. Esquema relacional	50
3.4.3. Normalización	53
3.5. Ciberseguridad	57
4. Pruebas y resultados del sistema de monitoreo	58
Conclusiones	77
Bibliografía	78
A. Documento ERS	82
B. Manual del programador	93

Resumen

El monitoreo y registro manual de la medición de los sensores presentes en áreas y biorepositorios permite realizar análisis con el fin de detectar anomalías que puedan ocasionar daños en los bioespecímenes. El estudio de los bioespecímenes permite generar medicamentos y tratamientos que mitiguen o curan afectaciones. De ahí la importancia de mostrar y analizar las temperaturas de forma remota a través de Internet. En esta investigación se muestra el diseño y el desarrollo de un sistema remoto basado en web para el monitoreo de biorepositorios ubicados en distintas áreas en el Laboratorio de Genómica Viral y Humana (LGVH). Para el desarrollo del sistema web se utiliza HTML, CSS y JavaScript en la parte frontal. Para la parte trasera, se utiliza el framework Django con Python y la base de datos se implementa utilizando MySQL. Al sistema web se le realizaron pruebas de caja negra y blanca para garantizar el cumplimiento de los requerimientos del cliente y la funcionalidad del sistema. El sistema propuesto permite recolectar, mostrar y crear gráficas para analizar las temperaturas de áreas y biorepositorios. El análisis y el diseño del sistema propuesto es la base para un sistema de monitoreo web inteligente que permita la predicción de fallas.

Palabras clave: Caja negra, Caja blanca, Django, CSS, HTML, JavaScript, Python, Biorepositorio, MySQL, Bioespecímen, Monitoreo, Temperaturas, Sistema web.

Introducción

La investigación médica siempre esta en busca de soluciones a las enfermedades humanas realizando gran cantidad de estudios y experimentos sobre muestras de bioespecímenes obtenidos de pacientes infectados y no infectados, estas son recolectadas y almacenadas con la finalidad de tener múltiples registros para futuros usos. Gracias al almacenamiento de grandes cantidades de bioespecímenes se ha permitido el desarrollo de mejores medicamentos, tratamientos y técnicas moleculares de alta resolución. Posteriormente se aplican en la realización de estudios en diferentes áreas como epidemiología molecular, diagnóstico comercial y vigilancia epidemiológica rutinaria tanto de virus, hongos y bacterias [12]. Un ejemplo de ello es la organización American Type Culture Collection (ATCC) que recopila, almacena y brinda bioespecímenes para fines de investigación [12].

Los biorepositorios son conjuntos de registros de bioespecímenes almacenados en refrigeradores o congeladores que los mantienen en un proceso de criopreservación congelándolos a muy bajas temperaturas lo que disminuye sus funciones vitales (tanto célula como organismo), todo esto para conservarlos en vida suspendida por largo tiempo [12]. Gracias a ello, los registros se han convertido en una parte fundamental en los avances científicos [12].

Los resultados que se han obtenido gracias a las buenas condiciones de las muestras ha generado un interés en mantener una revisión constante y ardua de las infraestructuras que envuelven a los bioespecímenes. Factores como fallo en los sensores de temperatura, fallo mecánico, fallo eléctrico, factores ambientales o alguna falla que pueda presentar el área o biorepositorio representan el daño o perdida total de las muestras, es por ello que algunos laboratorios u organismos (como el ATCC) mantienen un constante monitoreo del estado y funcionamiento de sus biorepositorios y entornos.

Un sistema de monitoreo brinda seguimiento a variables que necesiten estar bajo supervisión con la finalidad de conocer los estado o condiciones que se vayan presentando a lo largo del tiempo. Gracias a que el monitoreo automático se apoya de sensores y microcontroladores los sistemas pueden analizar las condiciones en las que se encuentran los biorepositorios de manera remota.

En la Facultad de Medicina de la Universidad Autónoma de San Luis Potosí se encuentra el Laboratorio de Genómica Viral Humana (LGVH) el cual posee biorepositorios de sangre entera, suero, ADN viral, ARN y ADN humano, células mononucleares periféricas y partículas virales de pacientes infectados que a lo largo de los años se ha ido creando, los cuales, están conformados por:

- Colección de ADN Genómico Mexicano - Conjunto de referencia Mestizo v5.0
- Colección de ADN Genómico Mexicano - Conjunto de referencia TNK Amerindios v2.0
- Colección de líneas celulares linfoblastoides B mexicanas - v1.0
- Colección de células mononucleares mexicanas - v1.0
- Colección de aislamientos virales mexicanos - v1.0

Los biorepositorios se encuentran distribuidos en las distintas áreas del LGVH y tanto áreas como biorepositorios tienen instalados sensores de temperaturas que permiten al personal del laboratorio saber si éstas se encuentran en los rangos adecuados.

Diariamente el personal anota manualmente en un documento las temperaturas que muestran los sensores, creando registros que se utilizan para hacer análisis y detectar posibles afectaciones que pongan en riesgo la integridad de los bioespecímenes, tales como: temperaturas fuera de los rangos, fallas en los sensores, posibles fallas en los biorepositorios o fallas eléctricas.

Planteamiento del problema

Debido al método que se utiliza para hacer los registros, algunas situaciones como días inhábiles y las distintas actividades del personal no permiten la toma de registros de manera constante, lo que puede llegar a generar un mal análisis poniendo en riesgo el bienestar de los bioespecímenes.

Hipótesis

Ante la problemática que se ha mostrado, la propuesta para dar solución es utilizar herramientas digitales que permitan monitorear las temperaturas que presenten las áreas y biorepositorios de forma remota. La implementación de un sistema informático permitirá al personal de LGVH visualizar desde una página web las temperaturas que muestran los sensores que oscilan desde los -70°C hasta los 26°C , además de esto, se podrán consultar los históricos de las temperaturas, crear estadísticas y generar gráficas que le permitan al personal hacer un análisis.

Justificación

Dada la importancia que representan los bioespecímenes en las investigaciones que realiza el LGVH, mantenerlos en condiciones óptimas ha resultado ser un tema de gran consideración. Debido a las distintas circunstancias que pueden presentarse en el LGVH, la dinámica para monitorear las temperaturas de áreas y biorepositorios que realiza el personal se ve afectada, dejando sin monitoreo posibles condiciones adversas como temperaturas de áreas (Tabla 1) y/o biorepositorios (Tabla 2) fuera de los rangos permitidos. Es por ello que la implementación de un sistema web de monitoreo automático con acceso remoto permita al LGVH mantener monitoreos constantes de las temperaturas, asegurando el bienestar de los bioespecímenes.

Tabla 1: *Rango de temperaturas que deben de presentar las áreas del LGVH.*

Áreas	Rango
RT-PCR ambiental	20°C a 26°C
Biología celular ambiental	20°C a 26°C
BSL3 ambiental	20°C a 23°C
Laboratorio principal ambiental	20°C a 26°C

Tabla 2: *Rango de temperaturas que deben de presentar los biorepositorios del LGVH.*

Biorepositorios	Rango
BSL2 Ultra No.2	-60°C a -70°C
Refrigerador RT-PCR	0°C a 4°C
Refrigerador de biología celular	0°C a 4°C
Refrigerador BSL3	0°C a 4°C
BSL3 Ultra No.2	-60°C a -70°C
BSL3 Ultra No.1	-60°C a -70°C
Congelador vertical del laboraorio principal	-15°C a -20°C
Refrigerador del laboratorio principal	0°C a 4°C
BSL2 Ultra No.1	-60°C a -70°C
Congelador horizontal del laboraorio principal	-20°C a -30°C

Objetivos

- General: Desarrollar e implementar un sistema web de monitoreo que permita registrar y mostrar las temperaturas que presentan las áreas y biorepositorios.
- Específicos:
 - Crear un sistema web que lea los reportes generados por los sensores y los almacene en una base de datos.
 - Crear gráficas de línea y caja de las temperaturas.
 - Visualización de históricos.
 - Crear estadísticas de las temperaturas.

- Crear un administrador para el sistema.

Estado del arte

En la actualidad los temas sobre monitoreo de temperaturas no son algo nuevo pero al tratarse de monitoreo de biorepositorios la literatura no es muy basta, aún así, existe literatura en otros ámbitos que sirven como referencia para poder culminar el sistema requerido. En la Tabla 3 se muestra una comparativa entre documentos con relación en el contexto de esta tesis, dando una mayor comprensión de la importancia que implica este sistema.

El primer trabajo relacionado es un desarrollo que se implementa en un supermercado ubicado en Córdoba (España) sobre los refrigeradores de productos lácteos, charcutería y productos congelados [28].

En el segundo trabajo se desarrolla un circuito que se instala en un frigorífico que almacena muestras de sangre. El circuito monitorea la temperatura la cual debe de permanecer en 4 °C, lo que evita la escasez de suministros de sangre [15].

El tercer trabajo tuvo como incentivo los cortes de energía que afectaban el aire acondicionado de una sala de servidores. Se implementa un monitoreo de temperaturas basado en web de la sala de servidores para evitar su sobrecalentamiento, permitiendo al usuario un monitoreo a través de su sitio utilizando una computadora o dispositivo móvil [31].

Un cuarto trabajo relacionado tiene como finalidad conservar los especímenes de los biorepositorio en condiciones adecuadas de humedad y temperatura en el biobanco de la Universidad Autónoma de Bucaramanga (UNAB) incluyendo el área donde están ubicados [17]

Tabla 3: *Análisis de trabajos relacionados. Las columnas muestran las características que contiene cada trabajo.*

Título	Página web	IoT	Mensaje de alerta	Base de datos	Análisis de datos
Monitoreo de temperaturas en gabinetes refrigerados minoristas aplicando IoT sobre hardware y software de código abierto [28].			✓	✓	✓
Sistema inalámbrico de Monitoreo de Temperatura para Banco de Sangre usando Zigbee [15].	✓	✓		✓	
Un sistema de monitoreo de temperaturas basado en la web para el colegio de artes y letras [31].	✓	✓	✓	✓	✓
Prototipo del sistema informático para monitoreo de condiciones ambientales en el bio-banco de la UNAB empleando hardware y software libre [17].	✓	✓	✓	✓	✓

IoT = Internet of Things por su siglas en ingles.

Metodología

El diseño desarrollado busca satisfacer la problemática planteada en base a los objetivos y antecedentes propuestos utilizando herramientas informáticas como el proceso metodológico de cascada (Figura 1) que divide el desarrollo en fases.

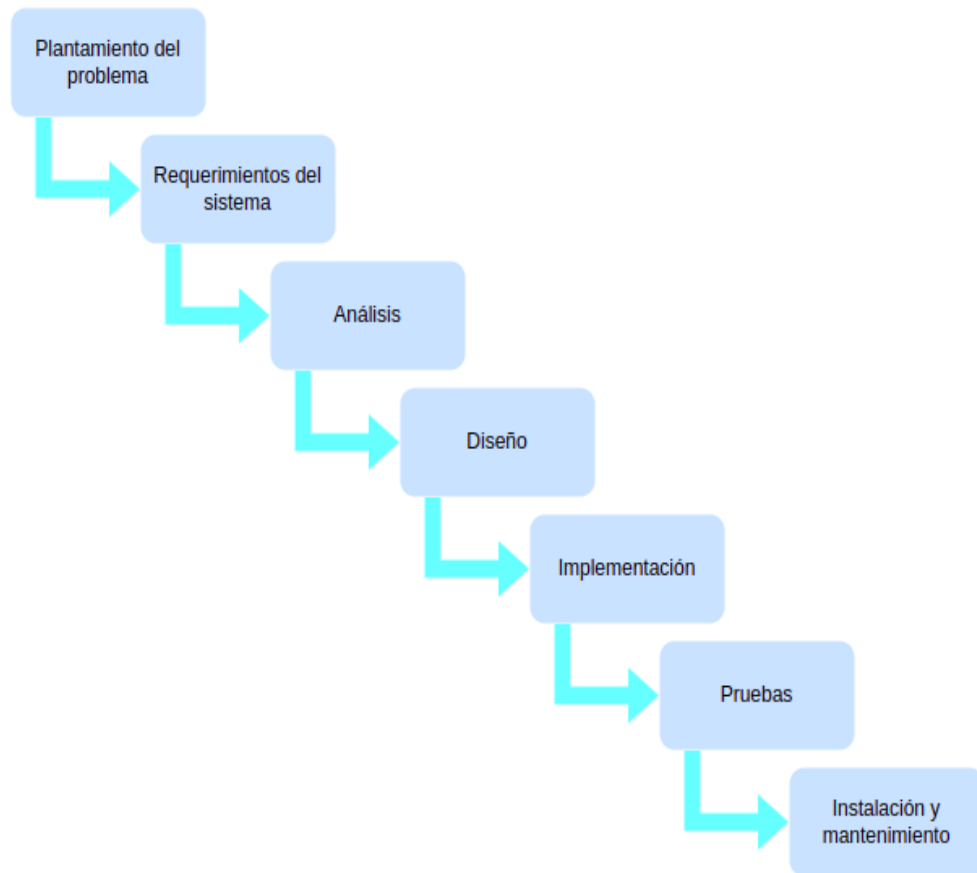


Figura 1: *Modelo de cascada del sistema web a desarrollar.*

El LGVH cuenta con un diseño de interfaces que muestran el contenido, la estructura y la funcionalidad de sus elementos; a ese diseño se le ajustaron algunos detalles visuales y se le complementaron con elementos dichos en los requerimientos obtenidos. En la Figura 2 se muestra el esquema del LGVH de las áreas (rojo difuminado) y los biorepositorios (rojo) que están siendo monitoreados por un sensor. Los sensores que utiliza el LGVH generan un reporte en Excel con la información y temperaturas del área o biorepositorio que están sensando, dichos reportes son leídos por el sistema web de monitoreo para obtener las temperaturas e ingresarlas dentro de la base de datos que será utilizada para mostrar los datos en el sistema.

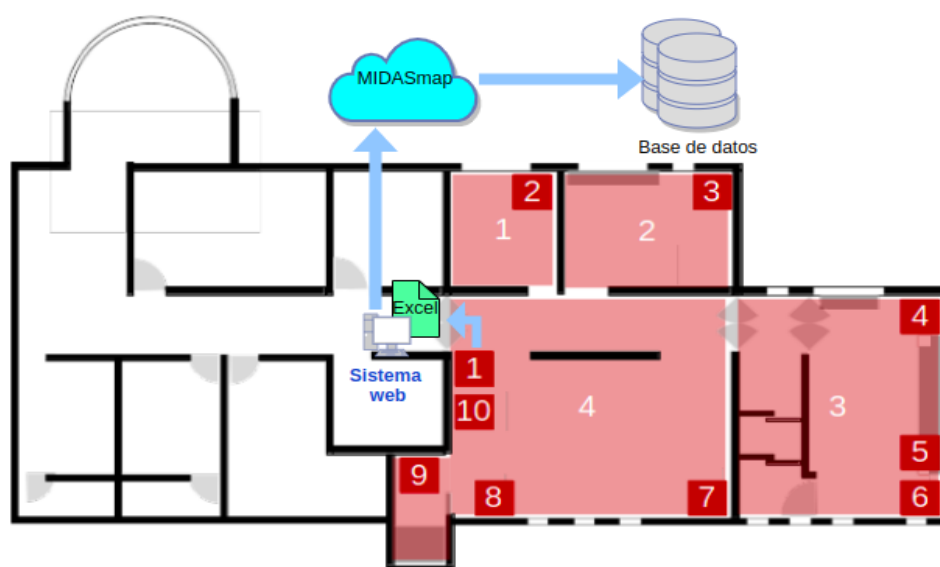


Figura 2: Esquema del LGVH que muestra la distribución de las áreas (Tabla 1) y biorepositorios (Tabla 2), cada uno con un sensor para medir su temperatura.

Sitio web

MIDASmap (Mexican Infectious Disease Analysis and Surveillance mapping application) es un sistema de monitoreo epidemiológico basado en web que permite visualizar emergencias y diseminación de los brotes sanitarios a

autoridades y público general [3]. Cuenta con una lista de módulos integrados con distintos propósitos ubicados en la sección de herramientas, aquí se encontrará el sistema web de monitoreo de biorepositorios.

Las herramientas para el desarrollo del sistema se presentan en el Capítulo 2 y posteriormente se tratará la elección de la base de datos revisada en el Capítulo 3. El sistema se encuentra dividido en dos partes, trasera (servidor) y frontal (cliente), con la finalidad de tener una mejor estructura en el sistema:

- **Parte trasera:**

Se encarga de leer los reportes generados por los sensores, la conexión con la base de datos, el funcionamiento interno del sistema web, el análisis de datos para la construcción de gráficas y estadísticas.

- **Parte frontal:**

Permitirá al usuario la visualización e interacción con el sistema de monitoreo por medio de interfaces.

Estructura

Esta tesis esta conformada por 4 Capítulos donde se explica todo el análisis y desarrollo que se ha realizado para lograr los objetivos planteados.

- **Análisis y diseño del sistema web**

Se da una introducción al Internet, las páginas web y todo lo relacionado a los inicios de las comunicaciones y cómo el desarrollo de ello ha generado las herramientas para la construcción de este sistema web, además, se analizan herramientas informáticas que permiten elegir las indicadas para el desarrollo de este sistema.

- **Análisis y diseño de la base de datos**

El almacenamiento de datos dentro de este sistema es una parte importante de mencionar ya que es donde se almacenan las temperaturas. La introducción a ¿qué es? y motivaciones para el uso de herramientas en bases de datos asegurará que exista una comprensión de su importancia.

- **Implementación del sistema de monitoreo**

En este capítulo se plasma el desarrollo del sistema de monitoreo y todo lo que conlleva, como el diseño de esquemas y figuras de las estructuras internas previo a su implementación en código.

- **Pruebas y resultados del sistema de monitoreo**

Se realizan las pruebas necesarias para comprobar que el producto tiene una funcionalidad adecuada, marcando la pauta para su implementación en el LGVH. Aquí se verán todos los resultados obtenidos.

Capítulo 1

Análisis y diseño del sistema web

La tecnología se ha convertido en parte de nuestra vida cotidiana que día a día nos va demostrando su importancia, especialmente, el Internet. Cuando un usuario entra a Internet desde un dispositivo móvil o computadora se encuentra con un sin fin de interfaces con variedad de diseños y estilos conocidas comúnmente como páginas web. Las páginas web ayudan a la interacción del usuario con una empresa, negocio, herramienta o individuo que ofrezca o permita un servicio de interés como puede ser entretenimiento, laboral, aprendizaje o como forma de comunicación entre personas al rededor del mundo.

Nacido en 1968, gracias a la agencia ARPA (Agencia de Proyectos de Investigación Avanzados de Defensa, por sus siglas en inglés DARPA) el Internet vio la luz al inicio de un proyecto que requería que una red de ordenadores pudiera sobrevivir a cualquier catástrofe, la cual fue llamada ARPANET (Advanced Research Projects Agency Network) [2]. Al dejar de ser uso exclusivo militar y académico [2] el Internet llegó a dar un giro enorme al estilo de vida de las personas, las cuales acceden por medio de dispositivos electrónicos como celulares, computadoras y tabletas que se interconectan en la red de Internet, dicha red cuenta con protocolos, normas y lenguajes que utilizan los dispositivos para comunicarse entre si (Figura 1.1).

Pero, ¿cómo es que alguien sin conocimientos de computación llegó a poder interactuar con el Internet? ¡Bueno, eso fue gracias a las páginas web!. En 1992 un científico británico llamado Tim Berners-Lee trabajador del CERN (sigla en francés Conseil Européen pour la Recherche Nucléaire, con tra-

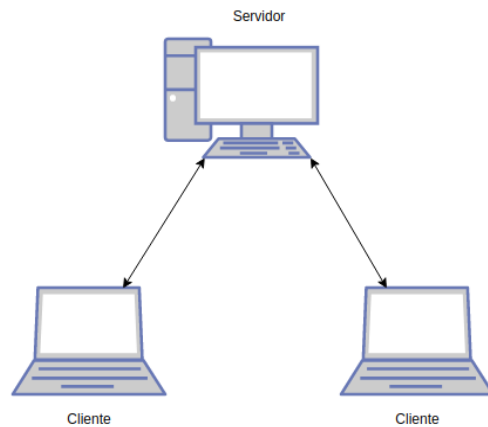


Figura 1.1: *Solicitud de recursos e información de ordenador a ordenador, uno sirviendo como servidor y los otros como clientes [20].*

ducción al español como Consejo Europeo para la Investigación Nuclear) de Suiza crea un sistema que representa información multimedia en Internet llamándolas páginas web, este sistema generó gran interés que en 1993 se puso a disposición el primer lector de páginas llamado World Wide Web (WWW) [2].

Gracias a que los sitios web muestran la información de forma multimedia y en lenguaje natural debido a WWW, los usuarios pueden entender e interactuar con ellos sin la necesidad de conocimientos sobre lenguajes de computación o ser un experto informático.

El WWW o Web es un servicio de Internet que muestra texto, imágenes, sonidos, gráficos, enlaces, servicios o páginas Web, con ayuda de lectores como Internet Explorer, Netscape, Opera, entre otros, permitiendo leer el contenido que hay en una página gracias al conjunto de protocolos como el HTTP (HyperText Transmission Protocol) y FTP (File Transfer Protocol), con estos protocolos la información puede ser compartida a través de la red [2].

Hoy en día la gran mayoría de las personas tiene acceso al mundo digital, cientos de miles navegan por Internet diariamente lo que lo convierte en un mercado enorme lleno de oportunidades. Las empresas y negocios cada vez más se están digitalizando haciendo uso de una página web como medio para ofrecer sus productos y servicios, aumentando la posibilidad de ventas y dar a conocer su marca en cualquier parte del mundo. Gracias a esto,

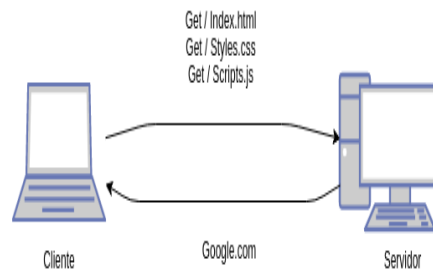


Figura 1.2: *El cliente le solicita al servidor los elementos que construyen una página web, al recibir la petición el servidor le regresa esos elementos como respuesta al cliente y este la renderiza para mostrar la página.*

los expertos informáticos y computólogos se han puesto manos a la obra en desarrollar métodos más eficientes de comunicación entre dispositivos y mejorar sus herramientas para la construcción de páginas web.

1.1. Lenguajes

Para el desarrollo de sitios web existen lenguajes (programación, estilos, etc) y herramientas que permiten a los programadores crear la puerta a todo entusiasta por explorar nuevas rutas en Internet. Los lenguajes de programación le permiten al programador controlar el comportamiento lógico y físico de una computadora mediante algoritmos, gracias a esto, una página web puede tener diferentes características de interfaz y usabilidad, todo ello regido por los requerimientos y diseño que se hayan establecido. Cada lenguaje tiene su función dentro de una página web, como proporcionar estética o una funcionalidad sencilla, esto es muy considerado ya que si el sitio es engorroso, complicado y poco entendible el usuario no accederá a la página, por esta razón, actualmente el desarrollo de un sitio se divide en dos partes, lado del cliente y lado del servidor:

- Cliente: Su ejecución es en el navegador y se encarga de permitir al usuario interacción con el sistema. Los lenguajes más utilizados son HTML, CSS y JavaScript (Tabla 1.1).

- **Servidor:** Manejan la parte lógica de la página y el acceso a la base de datos. Los lenguajes más utilizados son PHP, Python, Java, Ruby y C# (Tabla 1.1).

La división puede involucrar múltiples razones como mejoras de tiempos de desarrollo, especialización, mayor control, escalabilidad, mantenimiento, etc.

Algunos de los lenguajes más utilizados en el desarrollo web son los siguientes (la tabla 1.1 muestra el logo y una sintáxis básica de los lenguajes a mencionar):

HTML HyperText Markup Language es un lenguaje para la creación de documentos web. Este lenguaje se encuentra en toda la Internet y es el encargado de dar estructura a las página web gracias al uso de etiquetas que permiten incluir imágenes, enlaces a otros sitios, multimedia, entre otras [33].

CSS Cascading Style Sheets es un lenguaje de estilos que tienen el objetivo de definir la apariencia de los elementos que se encuentran en la página web. Se implementa en HTML y permite darle estilos a grupos de etiquetas sin necesidad de hacerlo individualmente.

JavaScript Es un lenguaje de programación desarrollado por Netscape que permite al usuario interactuar con la páginas web [25]. La finalidad de JavaScript es su incorporación al lenguaje HTML ya que no necesita compilarse gracias a que el navegador se encarga de traducir su código, permitiendo hacer programas que se ejecuten directamente en el navegador [25].

PHP Del acrónimo recursivo Hypertext Preprocessor es un lenguaje de código abierto que se utiliza junto a HTML, y al igual que JavaScript, puede ejecutarse en el navegador, aunque su enfoque principal es la programación de scripts (la ejecución de sus instrucciones permiten realizar alguna tarea [5]) del lado del servidor [5].

Python Es un lenguaje de programación de alto nivel que es interpretado, multiplataforma y multipropósito el cual cuenta con licencia Open Source [9].




Java Lenguaje de programación que se utiliza para el desarrollo de aplicaciones para todo ámbito. En la actualidad muchas aplicaciones y sitios web





dependen de él para su funcionamiento [14].


C# Es un lenguaje de programación orientado a objetos que permite desarrollar una variedad de aplicaciones que son ejecutadas en el ecosistema .NET [6].

Ruby Lenguaje de programación orientado a objetos, interpretado y reflexivo. Tiene su distribución bajo la licencia de software libre. Su sintaxis se familiariza con la de Python y Perl [22].

Tabla 1.1: *Algunos lenguajes junto a su logo y sintaxis para el desarrollo de páginas web.*

Lenguaje	Sintaxis
	<pre><BODY> <input name="N1"> <input name="N2"> </BODY></pre>
	<pre>button{ font-size: 20px; font-family: 'Arial'; }</pre>
	<pre>function Show() { alert('Sumar'); }</pre>

	<pre> if(\$_POST) { \$n1 = \$_POST['N1']; \$n2 = \$_POST['N2']; } </pre>
	<pre> def Suma(request): n1 = request.POST['N1'] n2 = request.POST['N2'] </pre>
	<pre> If(request.getParameter("suma")) { int n1 = Integer.parseInt(request.getParameter("N1")); } </pre>
	<pre> Suma(FormCollection form) { string n1 = form["N1"]; string n2 = form["N2"]; } </pre>

	<pre>def Suma n1 = params[:N1] n2 = params[:N2] end</pre>
---	---

Cada lenguaje tiene cualidades que los hacen mejor opción para desarrollar que otros y usos que puede aplicarse en distintas áreas, algunos con diferencias como los orientados a objetos, sintaxis más sencillas, documentaciones más amplias y/o con grandes comunidades de usuarios.

La selección de los lenguajes para el desarrollo de este sistema no tiene como justificación la lista de lenguajes vista anteriormente (ya que su propósito es solo dar a conocer algunas tecnologías utilizadas para el desarrollo web) si no a la experiencia en su uso.

Para el desarrollo del lado del cliente se ha considerado el uso de HTML, CSS y JavaScript debido a que se tratan de los lenguajes base para el desarrollo de este lado, agregando además, su gran popularidad, rápido desarrollo, mucha documentación y a la gran capacidad de estructurar, dar estilos y crear dinamismo a una página web.

Para desarrollar el lado del servidor se ha seleccionado Python debido a su buena comunidad activa de desarrolladores, documentación, sintaxis sencilla y rápido desarrollo, además, cuenta con una sencillez para utilizar tecnologías de inteligencia artificial que este sistema pretende implementar en un futuro.

1.2. Frameworks y entornos de desarrollo

Para el desarrollo de un sistema web no necesariamente se debe de tener una súper herramienta que nos permita comenzar a trabajar, basta con algún editor de texto para eso, pero esta opción no es recomendable cuando involucramos sistemas grandes ya que puede volverse tedioso y poco escalable, ante estas circunstancias se utilizan frameworks e IDEs que permiten un mejor desarrollo web.

Un framework engloba conceptos, reglas y módulos que sirven como base para la organización y desarrollo de software. El framework ofrece una in-

fraestructura de programación para construir aplicaciones sin redundancia de código y fácil mantenimiento [11]. Existen frameworks para los diferentes lenguajes mencionados con anterioridad (Tabla 1.1) que permiten crear una página web de una manera sencilla. Para el caso de PHP se encuentra Laravel, para Ruby se usa Ruby on Rails, Python cuenta Django, Java tiene Spring y C# cuenta con .NET los cuales usados dentro de un IDE brindan una gran herramienta que facilita la construcción de la página web, teniendo en cuenta lo dicho, se ha considerado el framework Django.

Django es un framework de alto nivel que permite el desarrollo web de manera rápida y práctica, creado por Adrian Holovaty periodista y desarrollador web se vio forzado a implementar herramientas para un rápido desarrollo de aplicaciones por las necesidades de su trabajo de crear aplicaciones enteras en poco tiempo, esto dio nacimiento a Django [11]. En 2005 se libera el framework como software de código abierto, gracias a esto, alrededor del mundo se han generado miles de usuarios y colaboradores que permiten que Django sea un proyecto estable y maduro.

Con el lenguaje y el framework establecidos, ahora es turno de pensar donde se impregnará el vestigio de dicha obra de arte, lo que llevó a considerar el uso de un IDE.



Figura 1.3: *Iconos de los IDEs de NetBeans, Visual Studios, PyCharm y Komodo.*

Un IDE (por sus siglas en ingles Integrated Development Environment) es una aplicación informática que facilita la creación de proyectos, depuración de código, marcado de sintaxis erróneo y más. Muchos lenguajes de programación tienen IDEs que permiten su uso de una manera rápida y sencilla,

como NetBeans donde comúnmente se implementa Java y PHP, Dev-C implementa a C y C++, PyCharm implementa a Python, C# usa Visual Basic, Aptana Studio implementa a Ruby, Perl utiliza a Komodo, etc (figura 1.3). Y aunque otros IDEs pueden permitir la implementación de Python como en el caso de NetBeans, se ha optado por el uso de PyCharm gracias a que cuenta con un buen soporte para Django lo que permite el desarrollo de la parte frontal y trasera.

PyCharm es un IDE desarrollado por la compañía jetBrains que proporciona un entorno multiplataforma con versiones para los sistemas operativos Windows, macOS y Linux para el desarrollo de aplicaciones en el lenguaje Python. Brinda análisis de código, depurador gráfico, integración con sistemas de control de versiones (VCS), probador de unidades y soporta el desarrollo web con Django y ciencia de datos usando Anaconda [30].

Capítulo 2

Análisis y diseño de la base de datos

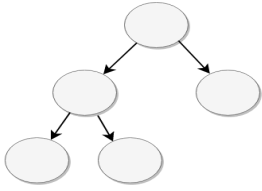
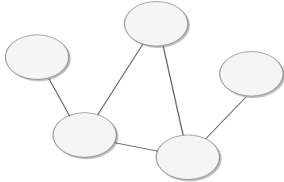
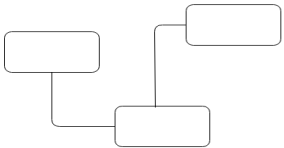
Muchos sistemas informáticos actuales tienen como requisito fundamental implementar una Base de Datos (abreviada en español como BD) ya que necesitan almacenar datos creados por el sistema o ingresada por sus usuarios, dichos datos se guardan en la BD con la finalidad de llevar un registro de actividades, usuarios o alguna otra necesidad del sistema web. La información que se almacena no se pierde si el sistema no esta en uso o inactivo ya que los datos están almacenados en algún servidor del cual la página hace uso.

Una BD es una colección de archivos relacionados que almacenan una representación abstracta de una problemática en forma de datos o información dentro de un computador [10]. Cabe mencionar que existen distintas definiciones hechas por expertos que pueden ir desde algo sencillo y general hasta algo más detallado que incluye las características que deben de cumplir las BD para ser llamadas así, pero este documento no busca a profundidad todo lo que involucra una BD, si no que busca lo esencial para el desarrollo de este sistema.

Para poder desarrollar una BD de manera eficaz se debe de tomar en cuenta el diseño y la lógica de negocios del sistema con enfoque en los requerimientos, todo esto tiene la finalidad de apoyar la elección respecto a la forma y herramientas que son requeridas para construir dicha BD. Existen diferentes modelos de BD que sirven para la descripción de los datos, sus relaciones y sus restricciones [16] con distintos diseños para dar solución a

las necesidades que requieren los sistemas actuales, incluyendo este. En la Tabla 2.1 se muestran algunos ejemplos de ellos.

Tabla 2.1: *Ejemplos de modelos de bases de datos.*

Módelo	Descripción	Esquema
Jerárquico	La estructura que organiza los datos tiene forma de árbol de manera jerárquica. Un nodo padre puede tener varios hijos pero los hijos no pueden tener varios padres [16].	
Red	A comparación del modelo Jerárquico, en este modelo los hijos pueden tener múltiples padres, permitiendo una relación de muchos a muchos lo que remueve la jerarquía [16].	
Relacional	El conjunto de datos que se almacenan (no importa el orden como en el jerárquico o red) llevan consigo una relación que permite su recuperación y almacenamiento a manera de consultas. Por lo regular se representan en tablas con filas y columnas [16].	
Plano	Toda la información esta almacenada en una sola tabla o archivo. Su estructura es simple y se maneja por renglones y/o columnas.	Datos del usuario 1. Datos del usuario 2. Datos del usuario 3. Datos del usuario 4.

La reinención de nuevos modelos se ha impulsado gracias al rápido desa-

rollo tecnológico y las exigencias de la sociedad moderna. En un futuro (no muy lejano) saldrán a la luz modelos más optimizados que harán las cosas más sencillas para el manejo de datos, y a pesar de que los modelos mencionados (y no mencionados) tienen mucho tema por delante, para este documento solo serán mencionados los modelos relacionales y los llamados no relacionales (abarcen múltiples modelos en este grupo) ya que son los más usados a la fecha de escrito este documento.

2.1. Modelo relacional y no relacional

Existen herramientas o lenguajes que permiten la implementación e interacción con un modelo en una BD. No todos los modelos se trabajan igual, en el caso del modelo plano (Tabla 2.1) solo basta con leer el archivo donde se encuentren los datos usando algunas líneas de código, este ejemplo es simple porque la estructura del modelo no requiere de más, pero en el caso del modelo relacional, es otra historia.

El modelo relacional (Tabla 2.1) utiliza un Lenguaje de Consultas Estructuradas (Structure Query Language, por sus siglas en inglés SQL) que tiene la finalidad de ser una interfaz para el uso de comandos (también llamados sentencias) que permiten crear tablas, modificarlas y eliminarlas, además de la inserción, consulta, modificación y eliminación de los registros de las tablas. Una tabla puede relacionarse con otra utilizando identificadores que son un conjunto de caracteres alfanuméricos de cualquier longitud, únicos para cada registro de la tabla.

Los modelos relacionales se representan con un esquema de tablas (Figura 2.1) que cuentan con atributos y uniones de diferentes significados que representan el tipo de relación entre tablas, en el caso de la tabla ‘Banda’ y tabla ‘Álbum’ tienen una relación llamada ‘uno a muchos’, representada por una línea que en sus extremos tienen simbologías que indican el tipo de relación que hay entre ellas, que en este caso, significa que una banda puede tener uno o muchos álbumes creados pero un álbum solo puede pertenecer a una sola banda. Estos diseños ayudan a que las tablas se conecten y muestren el tipo de relación que hay entre ellas.

La estructura de los modelos no relacionales es de forma distinta debido a que no cuentan con el lenguaje SQL (llamados noSQL), no tienen identificadores para sus tuplas y a pesar de que se expongan como un modelo que no es recomendable usar puede que sea todo lo contrario ya que ellos nacieron

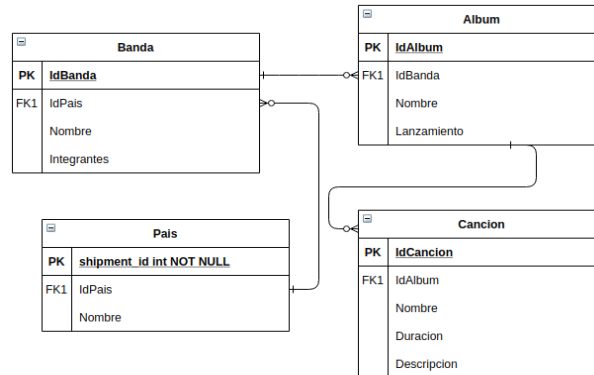


Figura 2.1: En las tablas se observan los atributos asignados. Las abreviaturas *PK* y *FK* se refieren a identificadores llamados llaves que hacen la relación entre tablas y que ayuda a las sentencias *SQL* a acceder a esas relaciones.

gracias a las deficiencias que tienen los modelos relacionales. Existe una variedad de modelos NoSQL [23] con diferentes estructuras para poder manejar una BD, cada una de ellas utiliza distintas técnicas de manejo de datos que pueden adaptarse a los requerimientos de un sistema. Algunos ejemplos son:

- El modelo orientado a grafos permite representar los datos en forma de vértices que están conectados por una o varias aristas, indicando así la relación que tienen los datos (Figura 2.2).
- Un modelo clave/valor almacena los datos con una clave única asignada, tanto la clave y los datos pueden ser cualquier cosa ya sea de manera compleja o sencilla, se podría decir que este modelo es similar al plano (Figura 2.1) con la excepción de que el plano no tiene necesidad de tener una clave para cada valor (Figura 2.2).
- Los almacenes de documentos contienen en sus registros archivos de texto con estructuras JSON o XML (entre otras) pero con libertad esquemática, a los registros se accede gracias a una clave única (Figura 2.2).

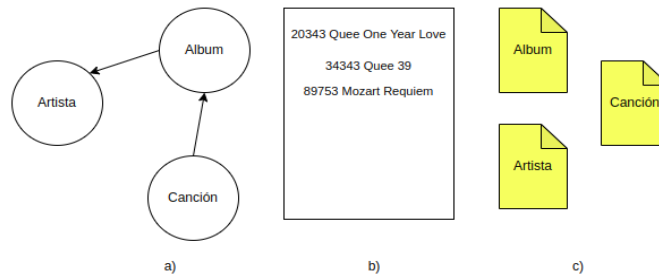


Figura 2.2: *a) Modelo de gráfico, b) modelo clave/valor y c) modelo almacenes de documentos.*

Con los modelos presentados, el panorama para una selección adecuada se extiende lo que genera preguntas sobre que diferencias hay entre ellos. La tabla 2.2 muestra algunas diferencias entre modelos.

Tabla 2.2: *Comparación entre modelo relacional y no relacional.*

	Ventajas	Desventajas
Relacional	<p>Estándar: Cuentan con el lenguaje SQL que facilita la creación de tablas y la gestión de los datos dentro de ellas.</p> <p>Atomicidad: Indica que se debe de seguir la regla de ‘se hace o no se hace’. Cuando se hace una modificación a la BD como insertar, modificar, o borrar, si en algún punto de esa acción falla algo toda la acción se mostrara como fallida.</p> <p>Sencilles: Gracias a su semejanza con el lenguaje humano la comprensión de las sentencias escritas no requieren de una gran experiencia para poderlas comprender.</p> <p>Madurez: Por llevar más tiempo en uso y haber generado buena aceptación por los desarrolladores ha generado que la comunidad de usuarios sea extensa al igual que la documentación para implementarla.</p> <p>Portabilidad: Tiene usabilidad en PC, servidores, dispositivos móviles y microcontroladores.</p>	<p>Estructura: Si en caso de que sea requerido una nueva modificación en los registros o la estructura de la BD y no sea aceptable por la estructura actual, posiblemente deba de reestructurarse toda la BD para que los cambios sean bien aceptados.</p> <p>Crecimiento: Cuando la cantidad de datos aumenta el mantenimiento y almacenamiento puede convertirse en algo muy costoso.</p> <p>Desempeño: La respuesta de las consultas se hacen lentas cuando la cantidad de datos es muy grande.</p>


No Relacional	<p>Crecimiento: Tiene una gran escalabilidad y no presenta mayor problema al extender la BD, todo esto sin afectar el funcionamiento lo que permite soportar una gran cantidad de datos (Big Data).</p> <p>Datos: Soportan cualquier tipo de datos, formato, estructura, tamaño, extensión, origen, etc.</p> <p>Versatilidad: Puede agregarse un nuevo dato o cambiarse la entrada de los datos sin afectar a la estructura de BD y sin necesidad de parar la producción del sistema.</p> <p>Costos: Tiene un costo menor ya que no se requieren de muchos recursos para su implementación, por lo regular los datos se alojan en clusters dentro de maquinas de bajo costo por lo cual no requiere de un servidor que suelen ser de mayor costo.</p>	<p>Atomicidad: Ya que no todas las BD tiene esta propiedad los datos entre nodos pueden ser inconsistentes.</p> <p>Joven: Su tiempo de vida no es tan extenso por lo cual no existe mucha documentación ni comunidad.</p> <p>Estándar: Los motores que se utilizan para estas BD no están estandarizados, por lo cual cada motor puede que tenga su propio lenguaje.</p> <p>Traducción: No todas las sentencias de SQL se pueden traducir a noSQL</p>
---------------	---	---

Para este trabajo se ha optado utilizar el modelo relacional debido a su madurez, su estándar, la experiencia en el manejo del modelo, a su adaptación a la lógica de negocios que presenta este sistema y a las comparativas de la Tabla 2.2.

2.2. Gestores de bases de datos

Los sistemas gestores de base de datos (SGBD, por sus siglas en español) fungen como interfaz para la manipulación de una BD, con esta interfaz se pueden realizar operaciones como crear, borrar y modificar una BDs, también crear tablas y/o insertar, consultar, modificar y borrar datos dentro de las mismas. Existen variedad de SGBD que pueden administrar de manera adecuada una BD, pero para este documento mencionaremos los más utilizados actualmente:

Tabla 2.3: *Ejemplo de algunos manejadores de bases de datos.*

Logo	Descripción	Usuarios
	MySQL es un SGBD multiusuario y multihilo que cuenta con dos tipos de licencia, de código abierto y comercial que está gestionada por Oracle. Su modelo es cliente-servidor, cuenta con sistema de producción crítica y gran carga lo que le permite integrarse a softwares de despliegue masivo (Tabla 2.2).	YouTube, PayPal, Tesla, Github [27].

	<p>PostgreSQL es un sistema de BD objeto-relacional open source que usa y extiende el lenguaje SQL combinado con muchas características que almacenan y escalan de forma segura las cargas de trabajo de datos más complicadas [13]. Nació en 1986 como parte del proyecto POSTGRES en la Universidad de California en Berkeley y tiene más de 30 años de desarrollo activo en la plataforma central [13].</p>	<p>The Oxford University Computing Services, Skype, Yahoo, Fujitsu, Red Hat, Sun Microsystems [13].</p>
	<p>Creado por Microsoft, SQL Server es un sistema de BD diseñado para un entorno empresarial. El lenguaje de desarrollo utilizado (por línea de comandos o mediante la interfaz gráfica de Management Studio) es Transact-SQL (TSQL), una implementación del estándar ANSI del lenguaje SQL, usado para manipular y recuperar datos (DML), crear tablas y definir relaciones entre ellas (DDL) [8].</p>	<p>Unibanco, Hiscox, Baltika [8].</p>

	Oracle database es un SGDB de tipo objeto-relacional desarrollado por la corporación Oracle [18].	Cern openlab, drop-tank y Panasonic [18].
	MongoDB es una BD NoSQL de documentos que ofrece una gran escalabilidad y flexibilidad, y un modelo de consultas e indexación avanzado [7].	Ebay, Google, SEGA y Adobe [7].
	Apache Cassandra es una BD NoSQL distribuida de código abierto, presenta un modelo columnar que fue diseñada para manejar cantidades grandes de datos en diferentes servidores [24].	Activision, Apple, Bestbuy [24].

La sintaxis y el manejo de la BD en los gestores mencionados (excepto mongo y cassandra) son muy similares entre ellas gracias a que utilizan SQL. La sintaxis SQL se muestra con el caso:

```
INSERT INTO nombre_tabla (columna1, columna2, columna3,..)
VALUES (valor1, valor2, valor3, ..)
```

dicha sentencia requiere de especificaciones para poder efectuarse, como el nombre de la tabla donde se ingresarán los datos, los nombres de los atributos y los valores que se almacenarán en cada atributo. En el siguiente ejemplo se tiene una tabla llamada ‘pacientes’ (Tabla 2.4) en una BD de un consultorio médico, al llegar a una consulta los pacientes son registrados en el sistema colocando sus datos en los campos que se encuentran en la interfaz, paso seguido el encargado de los registros presiona el botón ‘registrar’ lo que hace una sentencia INSERT que recibe los datos y guardar la información:

```
INSERT INTO pacientes (Nombre, Edad, Peso, Sintomas)
VALUES (‘Angel’, 29, 76, ‘fiebre y tos’)
```

Tabla 2.4: *Tabla sin registros*

Nombre	Edad	Peso	Síntomas

Tabla 2.5: *Tabla con registros*

Nombre	Edad	Peso	Síntomas
Angel	29	76	Fiebre y tos

el nuevo registro se muestra en la tabla de ‘pacientes’ (Tabla 2.5).

INSERT pertenece a una variedad de sentencias que maneja SQL con diferentes propósitos y sintaxis que realiza más de una acción a la vez lo que permite un manejo de datos más sencillo, y aunque cada manejador tiene sintaxis diferentes (algunos similares) la lógica para el uso de las sentencias es la misma.

Para el caso de algunos de los SGBD de modelos NoSQL, sus datos son almacenados en documentos similares a un JSON, permitiendo a sus atributos cambiar entre documentos y su estructura tenga posibilidad de cambiarse en un futuro [23]. El siguiente ejemplo es de una sentencia para la inserción del registro del paciente:

```
db.pacientes.insert({
  Nombre: "Angel",
  Edad: 29,
  Peso: 76,
  Sintomas: "Fiebre y tos"
})
```

una vez registrado los datos, el archivo muestra ese dato de forma:

```
{"Nombre": "Angel", "Edad": 29, "Peso": 76, "Sintomas": "Fiebre y tos"}
```

Para administrar los registros de temperaturas del sistema se elige el manejador MySQL por ser open source, experiencia personal en su manejo,

compatibilidad con el framework Django (visto en el Capítulo 1), gran comunidad, documentación y sencillez.

Capítulo 3

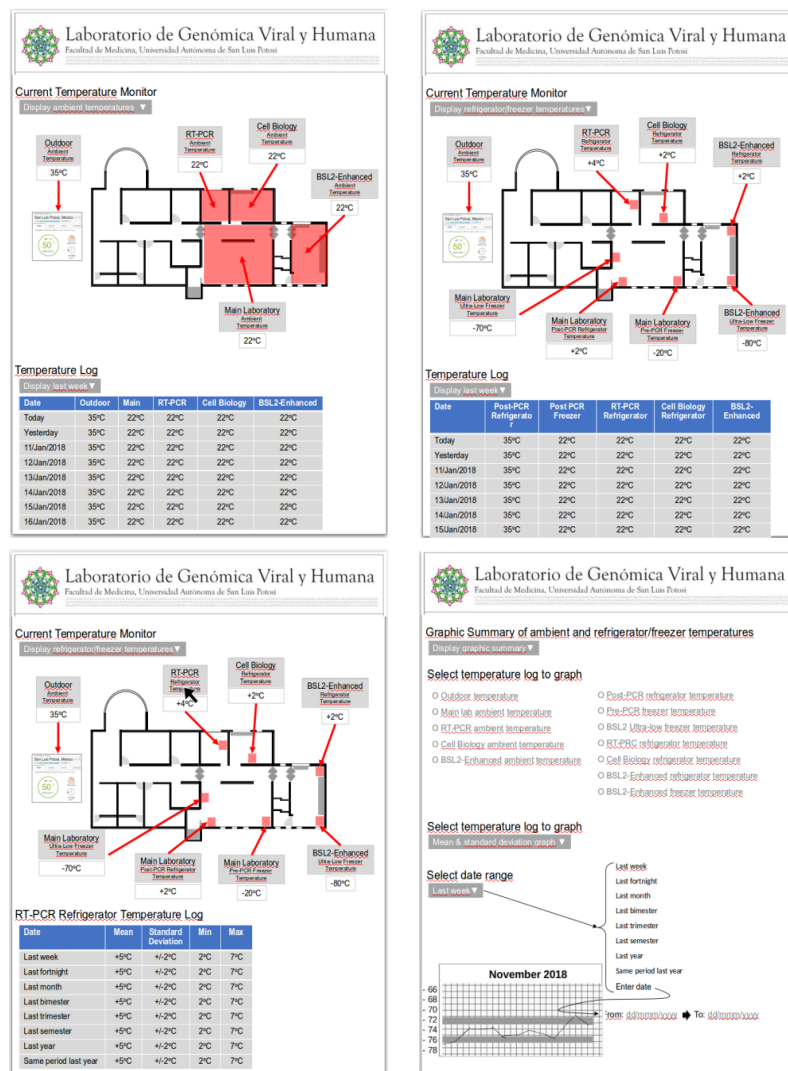
Implementación del sistema de monitoreo

Con el análisis visto en los capítulos anteriores y la selección de los elementos que se encuentran involucrados para el desarrollo de este sistema se procede a crear los diseños tanto de funcionalidad y estructura necesarios para cumplir el objetivo de este sistema, usando algunos métodos de forma directa (sin llegar a explicaciones profundas de los mismos) se pretende convertir los requerimientos en elementos más técnicos que sirvan como guía para la construcción del sistema de monitoreo. Cabe destacar que la elección de herramientas para el desarrollo de un sistema se realiza comúnmente después de analizar las necesidades del sistema, sin embargo, el orden puede variar dependiendo de los intereses y experiencia del desarrollador.

3.1. Requerimientos

Cuando se plantea un problema existen situaciones, variables o datos que permiten entender la problemática y dar una solución. Toda la información recolectada puede involucrar características y necesidades específicas requeridas que son proporcionadas por el cliente para el desarrollo del sistema. Para el caso de este trabajo los requerimientos fueron proporcionados por el LGVH donde se muestran las necesidades y los motivos por los que se debe implementar un sistema de monitoreo de biorepositorios, gracias al diseño previo de las interfaces del sistema (Tabla 3.1) los requerimientos solo fueron

Tabla 3.1: Interfaces del sistema que incluye los elementos que permiten al usuario registrar, monitorear y analizar las temperaturas de las áreas y biorepositorios del laboratorio.



complementarios lo que genera un mayor entendimiento de sus necesidades. Los requerimientos tomados son los siguientes:

1. Los registros de las temperaturas climáticas deberán obtenerse de Internet.

2. Las temperaturas se deberán consultar por fechas y mostrarlas en una tabla.
3. Realizar una identificación de temperaturas anormales mediante colores en las tablas de temperaturas.
4. Crear una tabla con las características estadísticas de la media, desviación estandar, máximo y mínimo de un área y biorepositorio seleccionado.
5. Crear histogramas de las temperaturas de un área o biorepositorio a partir de una fecha específica.
6. Crear gráficas de cajas de un área o biorepositorio donde se muestre los valores cuartiles, mediana, máximos y mínimos de un periodo de tiempo de tres años apartir de una fecha específica.
7. El sistema tomará los reportes generados por el Temperature Logger Elitech de todas las áreas y biorepositorios para extraer los registros de las temperaturas. Los lapsos de registro de las temperaturas serán de una hora.
8. El sistema de monitoreo se implementará en MIDASmap.
9. El encargado del LGVH será el único con accesos al sistema.

Los requerimientos se dividen en dos categorías, funcional (RF) y no funcional (RNF), el primero de ellos evaluá el comportamiento del sistema respecto a la acción que pueda hacer un usuario u otro sistema en él, de inicio a fin mientras que el segundo no hace referencia a las funciones específicas, sino a propiedades externas al sistema [32], por ejemplo:

- Extensibilidad: Facilidad de crecimiento del sistema.
- Escalabilidad: Cantidad que soporta el sistema a una carga de trabajo.
- Mantenibilidad: Facilidad de dar mantenimiento al producto.
- Seguridad: Protección de datos del sistema, software o uso indebido del producto.
- Usabilidad: Facilidad de uso del producto.

Los requerimientos proporcionados entran en la categoría RF ya que su enfoque va orientado a la funcionalidad del sistema, y aunque no se hayan proporcionado los RNF, se han considerado para el desarrollo de este sistema junto a otros aspectos redactados en el documento que se encuentra en el Apéndice A sobre Especificaciones de Requisitos de Software (ERS) necesarios para el desarrollo de este sistema.

3.2. Modelado

Los diagramas UML (Unified Modeling Language) permiten representar esquemas o procesos de software que ayudan al análisis y entendimiento del comportamiento de múltiples acciones que debe de realizar el sistema [21] basados en los requerimientos, los diagramas de estados, clases, componentes y casos de uso permiten representar funcionalidades y estructuras que el sistema va a tener. De los diagramas UML existentes se han considerado los casos de uso, de actividad y clases ya que se adaptan a los requerimientos vistos previamente.

3.2.1. Casos de uso

Un caso de uso plasma la interacción de un usuario o sistema que utiliza un sistema computacional [21] usando secuencia de acciones para llegar a un resultado. En la Tabla 3.2 se muestran unas plantillas con información acerca de cada caso de uso (encerrado en un círculo u óvalo) de la Figura 3.1, esto permite ver las acciones que los actores (usuarios u otros sistemas) del sistema van a realizar.

Tabla 3.2: *Plantillas de descripción*

Nombre:	Autenticación
Descripción:	Ingreso al sistema con usuario y contraseña.
Actores:	Administrador.
Pre-Condiciones:	El actor este registrado en el sistema.
Flujo:	<u>Actor</u> : El actor solicita el acceso al sistema. <u>Sistema</u> : Comprueba el registro del actor.
Flujo alternativo:	Mensaje de error cuando el usuario y/o contraseña son inválidos.
Pos-Condiciones:	Despliega la interfaz del sistema.

Nombre:	Registrar Temperaturas.
Descripción:	Registra las temperaturas de áreas, biorepositorios y climática.
Actores:	Administrador. Sistema de registro.
Pre-Condiciones:	El actor Administrador debe de estar autenticado. El sistema debe de esperar una hora.
Flujo normal:	<u>Actor</u> : - Ingresa a la interfaz administrador, agrega las temperaturas y da clic en guardar. - Lee los reportes producidos por el Temperature Logger Elitech cada hora y registra las temperaturas. <u>Sistema</u> : - Comprueba que los campos tengan datos y estos sean correctos.
Flujo alternativo:	Mensaje de error cuando faltan campos por llenar.
Pos-Condiciones:	Nada.

Nombre:	Actualizar Temperaturas.
Descripción:	Actualiza las temperaturas de áreas, biorepositorios y climática.
Actores:	Administrador.
Pre-Condiciones:	El actor debe de estar autenticado.
Flujo normal:	<u>Actor:</u> Selecciona un registro, hace los cambios del registro y presiona "Guardar". <u>Sistema:</u> Comprueba que todos los campos tengan datos y estos sean correctos.
Flujo alternativo:	Mensaje de error cuando faltan campos por llenar.
Pos-Condiciones:	Nada.

Nombre:	Eliminar Temperaturas.
Descripción:	Eliminar las temperaturas de áreas y biorepositorios.
Actores:	Administrador.
Pre-Condiciones:	El actor debe de estar autenticado.
Flujo normal:	<u>Actor:</u> Selecciona un registro y pulsa el botón eliminar. <u>Sistema:</u> Espera a la aprobación del actor para borrar el registro.
Flujo alternativo:	Mensaje para aprobar la baja del registro.
Pos-Condiciones:	Nada.

Nombre:	Consultar Temperaturas.
Descripción:	Consulta las temperaturas del laboratorio al seleccionar una fecha.
Actores:	Administrador.
Pre-Condiciones:	El actor debe de estar autenticado.
Flujo normal:	<u>Actor:</u> - Selecciona un rango de fecha para ver esos registros. - Colocar un rango de fecha específico para ver esos registros. <u>Sistema:</u> Busca los registros y los devuelve.
Flujo alternativo:	Mensaje de error de fechas específicas incorrectas.
Pos-Condiciones:	Llenar la tabla con las temperaturas devueltas.

Nombre:	Consultar estadísticas de temperaturas.
Descripción:	Calcula y muestra la media, desviación estándar (DS), máximo y mínimo de varios lapsos de tiempo de un área o biorepositorio.
Actores:	Administrador.
Pre-Condiciones:	El actor debe de estar autenticado.
Flujo normal:	<u>Actor:</u> - Pulsa el cuadro de temperaturas de un área o biorepositorio ubicado en el croquis del laboratorio. - Puede seleccionar un rango de fechas a calcular de la opción seleccionada. <u>Sistema:</u> Calcula la media, DS, máximo y mínimo de todos los registros.
Flujo alternativo:	Selección de rango: Mensaje de error de fechas específicas incorrectas.
Pos-Condiciones:	Crear una tabla con los datos calculados.

Nombre:	Consultar históricos de temperaturas.
Descripción:	Construye un histograma o una gráfica de cajas del área o biorepositorio de interes.
Actores:	Administrador.
Pre-Condiciones:	El actor debe de estar autenticado.
Flujo normal:	<u>Actor:</u> Selecciona el área o biorepositorio, selecciona el tipo de gráfica, elige el rango de fechas y presiona el botón update. <u>Sistema:</u> Obtiene las opciones elegidas y construye la gráfica.
Flujo alternativo:	Ninguno.
Pos-Condiciones:	Mostrar la gráfica creada.

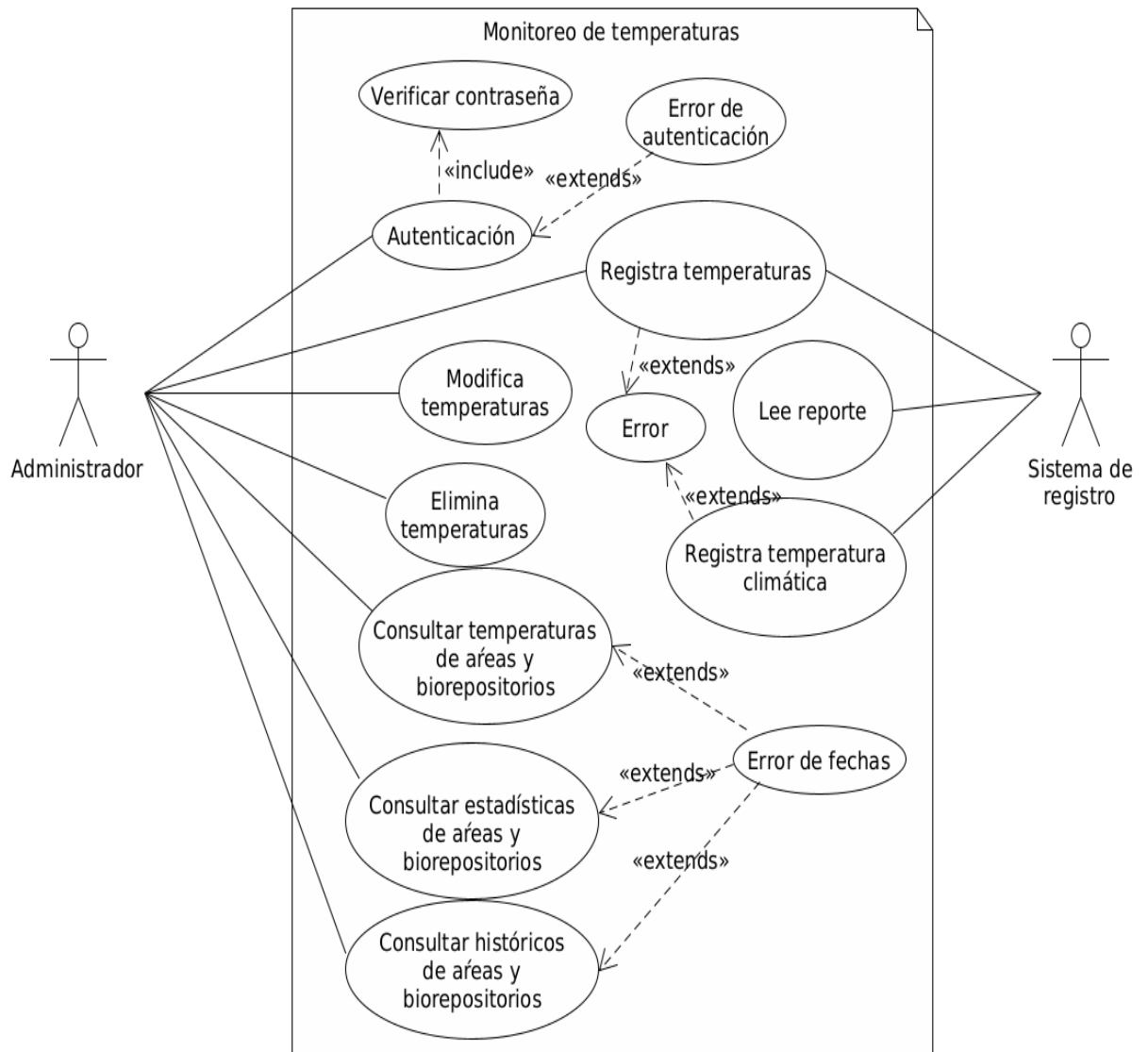
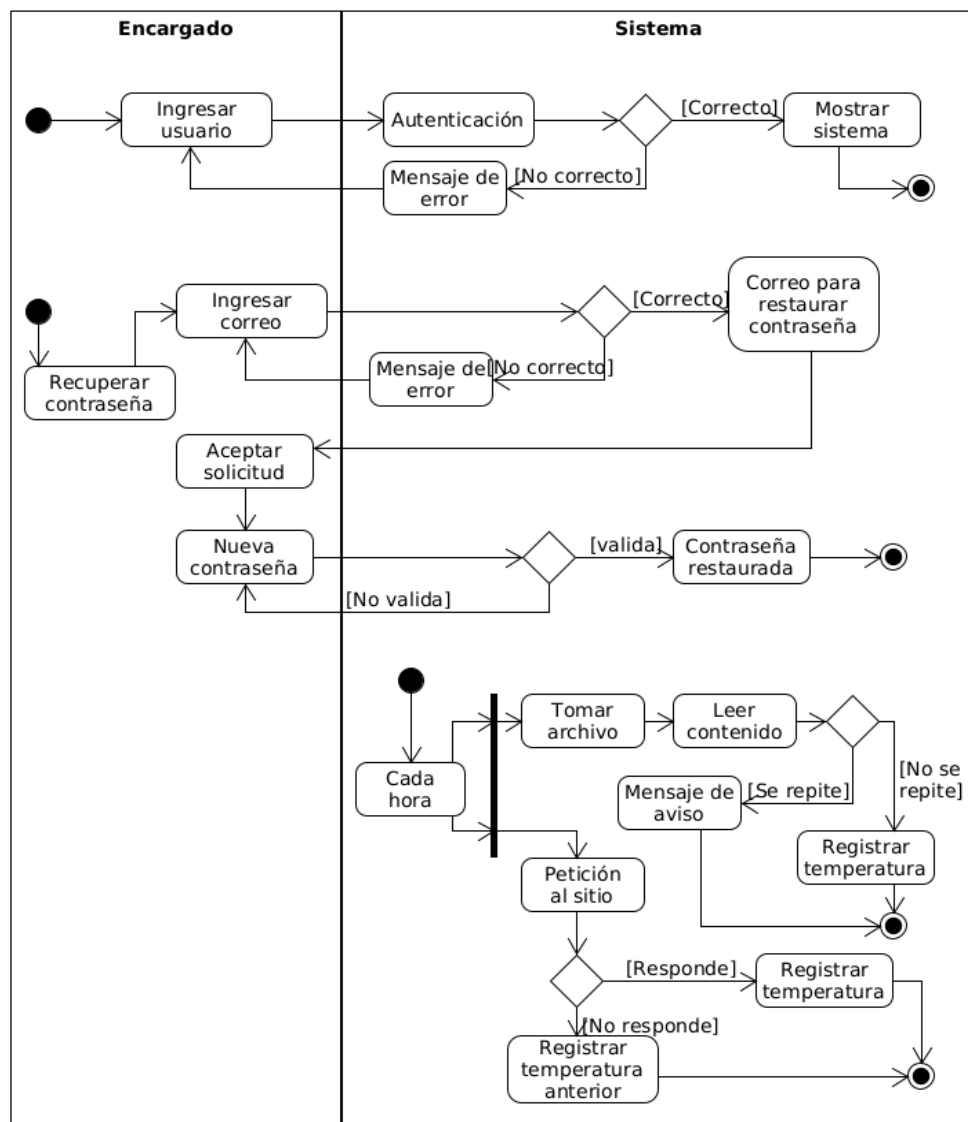


Figura 3.1: *Diagrama de casos de uso*

3.2.2. Actividades

Muestra el flujo de actividades de un sistema que incluye la acción de todos los actores involucrados de forma secuencial o paralela [21]. En la Figura 3.2 se muestra toda la actividad que el sistema hace internamente cuando el usuario realiza una acción, como presionar un botón para entrar al sistema una vez colocada su contraseña.



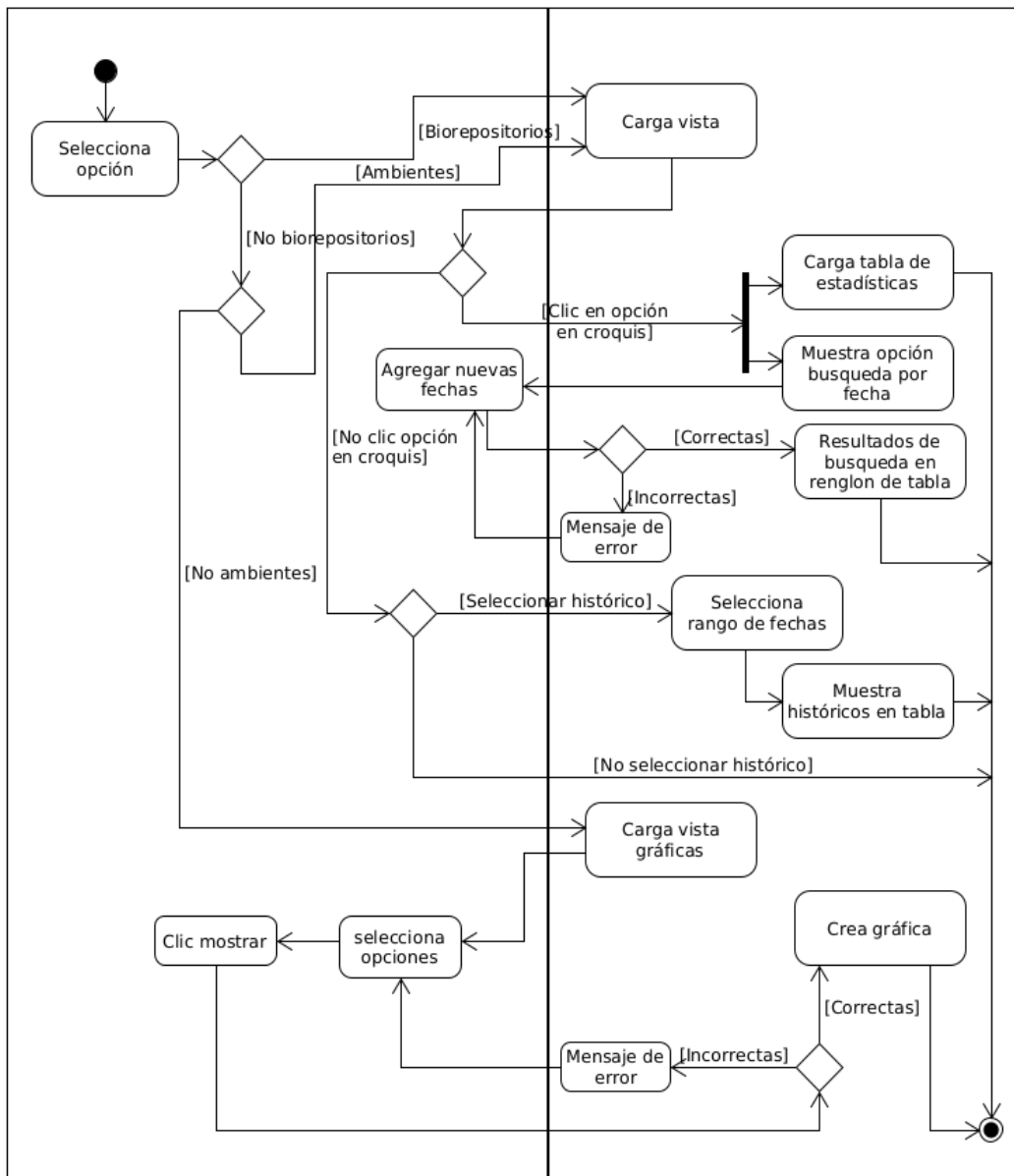


Figura 3.2: *Diagrama de actividades*

3.2.3. Clases

Muestra la estructura, comportamiento, características y relaciones entre elementos de los objetos dentro del sistema [21]. En la Figura 3.3 los objetos son representados por una caja (excepto Interface) y las relaciones con flechas. Una ejemplo de relación se muestra entre el objeto Ambiente y Temperatura, llamada asociación unidireccional donde un objeto esta dentro de otro.

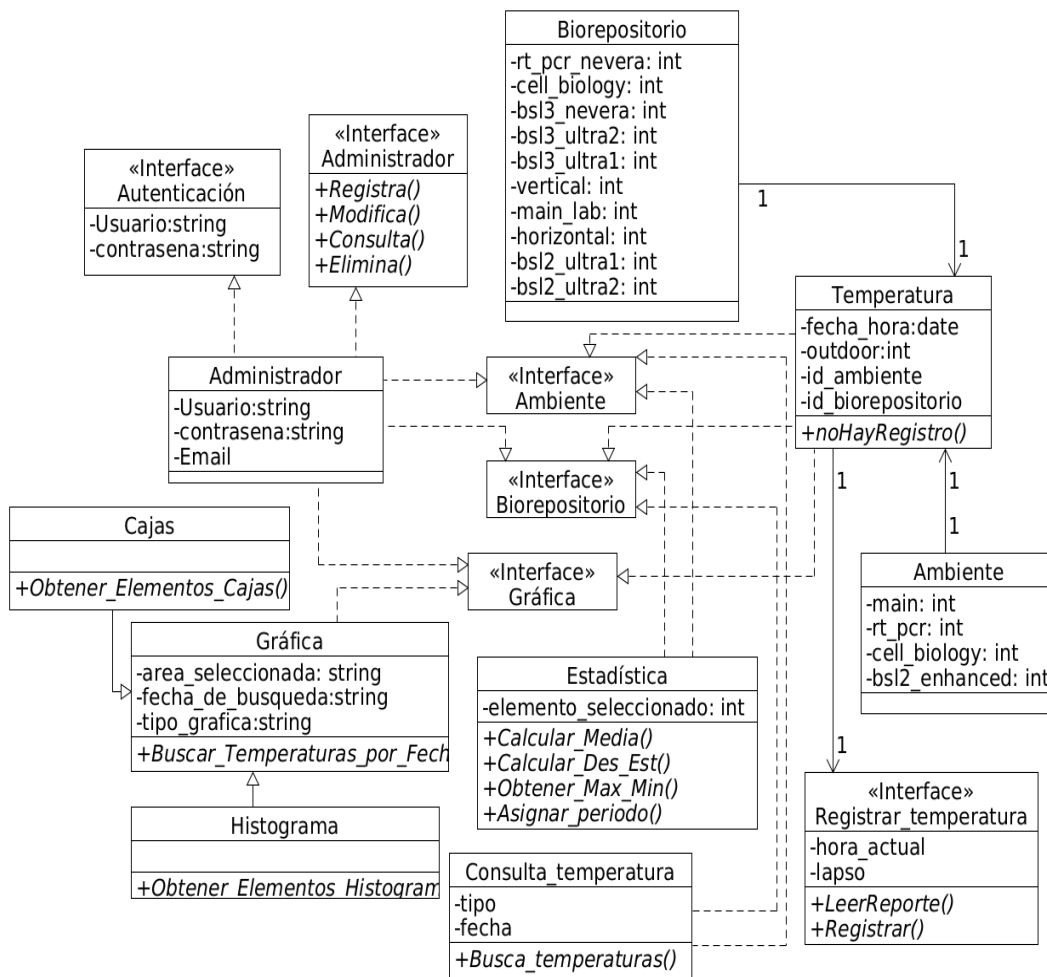


Figura 3.3: Diagrama de clases

3.3. Diseño de arquitectura

El proceso de desarrollo de un sistema lleva distintas etapas para llegar a su culminación, la elección de una arquitectura para el sistema se encuentra entre esas etapas y es importante porque brinda organización al sistema en base a los requerimientos, no considerar una arquitectura tiene una similitud a construir una casa sin planos ya que las arquitecturas permiten representar la estructura de los componentes y datos que manejará el sistema creando un panorama de lo que llegará a ser. Las arquitecturas han sido de gran ayuda para los nuevos sistemas porque permiten utilizar soluciones que ya han sido comprobadas por expertos en software [26], es por ello que se ha considerado implementarlas en este trabajo.

3.3.1. Patrón arquitectónico

Los patrones arquitectónicos permiten dar una descripción del desarrollo de un sistema de forma generalizada que puede ser implementada en sistemas web [32]. Existen diferentes tipos de arquitecturas como la de tubería y filtro, cliente servidor o MVC (Modelo Vista Controlador) [32] pero la arquitectura de capas se ha elegido debido a que permite separar el sistema para un desarrollo independiente de cada capa, comúnmente en tres capas o niveles, el uso de ella es muy común en desarrollo web [19] y además es una arquitectura que es utilizable en sistemas de monitoreo [4].

Como se muestra en la Figura 3.4 el nivel o capa superior se encuentra todo lo relacionado a las interfaces e interacción con el usuario, el nivel intermedio trabaja la lógica junto con el procesamiento de los datos causados por acciones recibidas del nivel superior y el nivel inferior gestiona los datos. Esta arquitectura permite la escalabilidad, la redundancia y el cambio de capas (mientras sean equivalentes) lo que es funcional en sistemas donde se considera la disponibilidad [32], la cual es un RNF importante para este sistema.

La arquitectura expuesta cubre la mayoría de los requerimientos mencionados anteriormente con la excepción del RF registro de temperaturas, es por ello que se ha optado por utilizar una arquitectura híbrida entre la de capas y la de repositorio, esta última usa los datos generados por algún componente externo (como los sensores o temperaturas del clima) almacenados en un punto central [32] que son obtenidos por otro sistema para su uso (Figura 3.5).

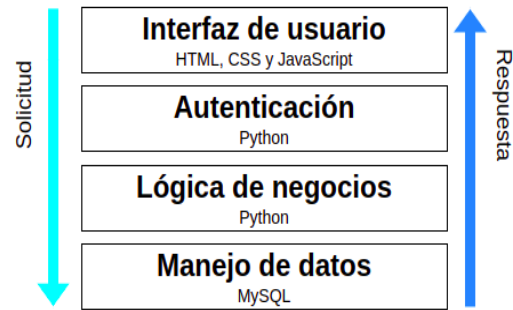


Figura 3.4: Las flechas muestran el flujo de forma descendente-ascendente, la cual debe de ser estricta y los nivel muestran el o los lenguajes utilizados en ellos.

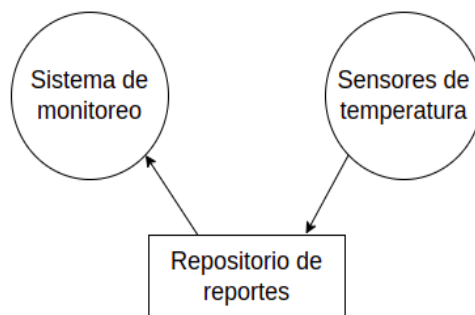


Figura 3.5: Los sensores generan los reportes y los almacenan en un repositorio para que el sistema de monitoreo pueda obtenerlos.

3.4. Diseño lógico de Base de datos

Crear un diseño para la BD permite hacer una definición de la estructura de los datos con base en los requerimientos y diseños que se han presentado previamente, en dicha estructura se representan las tablas junto a sus atributos y el tipo de relaciones que tienen con otras tablas (similar al diagrama de clases) lo que permite ver el panorama de nuestra BD.

3.4.1. Entidad-relación

El modelo entidad-relación permite representar de forma esquemática una estructura de BD [29]. En la Figura 3.6 las tablas se representan con rectángulos unidos a óvalos que representan los atributos de tabla. Las tablas se encuentran relacionadas por medio de líneas y rombos que dictan el tipo de relación que presentan las tablas. La relación que presenta la tabla Temperatura y la tabla Ambiente es de uno a uno, donde una Temperatura solo puede tener un Ambiente y viceversa.

3.4.2. Esquema relacional

El esquema relacional muestra las relaciones entre tablas de una forma similar al diagrama de clases (subsección 3.2.3) el cual, a diferencia del esquema entidad-relación, las entidades se muestran como tablas que permiten conocer una mayor información acerca de sus estructuras internas y sus atributos. En la tabla Temperatura de la Figura 3.7 se muestran el tipo de dato de los atributos y cuales de ellos son usados como identificadores.

A simple vista los esquemas muestran una estructura de BD sólida y viable, sin embargo, dichos esquemas no sean los más eficiente para gestionar la BD ya que pueden presentarse redundancias y dependencias incoherentes poniendo en riesgo los datos y que la BD sea inflexible. Para comprobar que los esquemas son confiables y óptimos para el manejo de datos del sistema de monitoreo se utiliza el proceso de normalización.

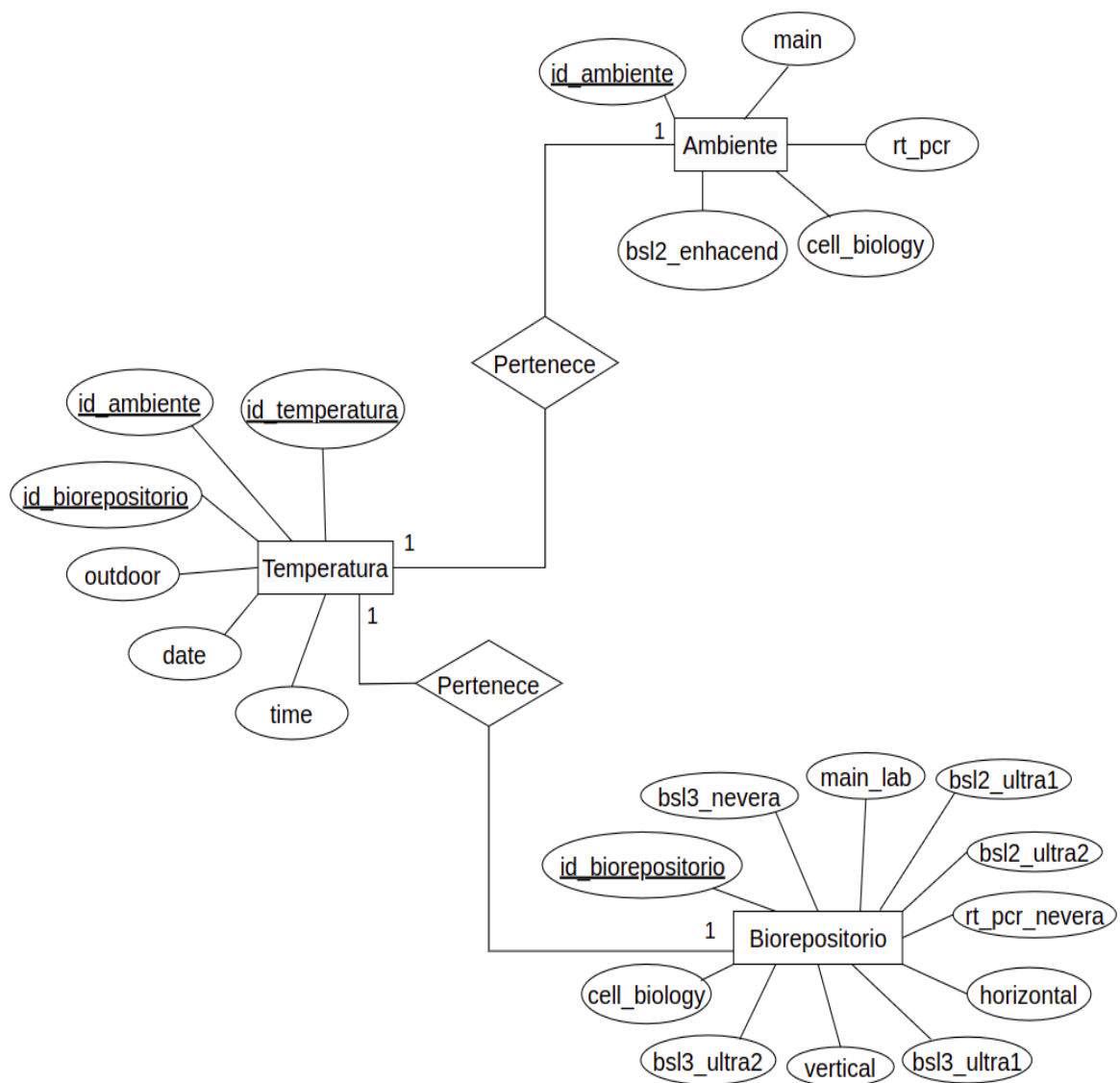


Figura 3.6: *Esquema entidad-relación.*

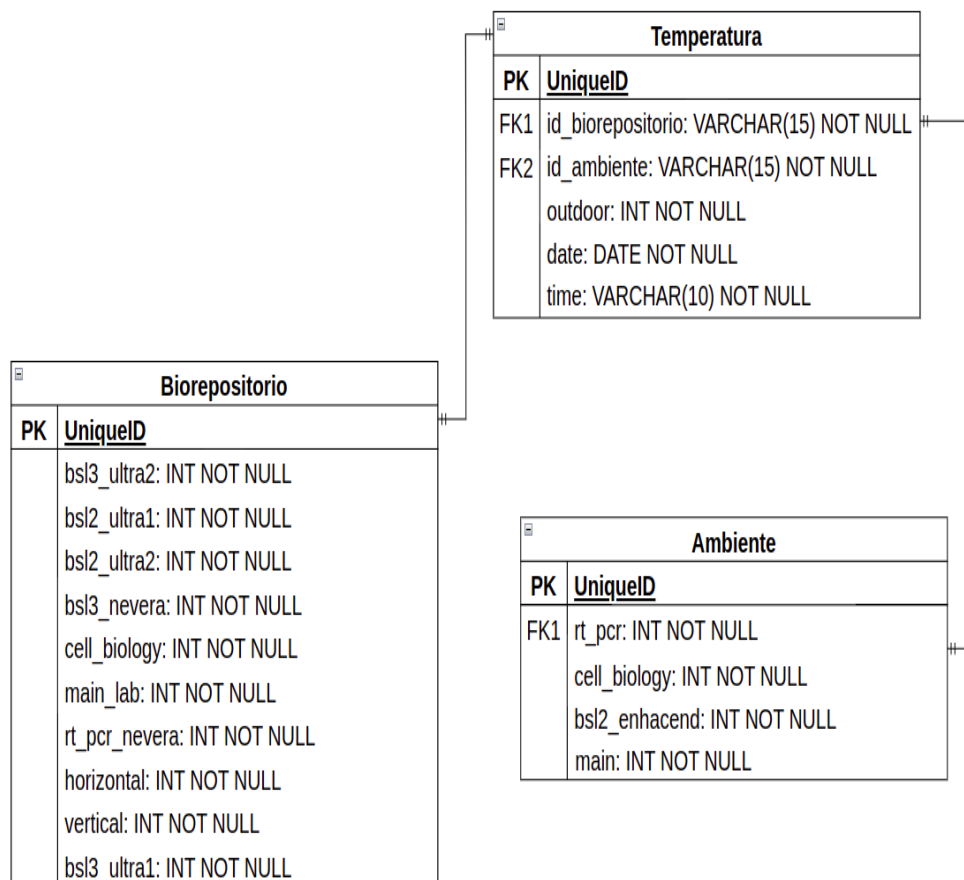


Figura 3.7: *Esquema relacional.*

3.4.3. Normalización

Para poder normalizar el esquema relacional se utilizan formas de normalización que permiten optimizar el esquema haciéndolo más fiable, eficiente y un manejo de la información mas rápido, además evita la redundancia dentro del esquema y la protección de la integridad de los datos [29]. Antes de aplicar las formas hay que tener en cuenta que las tablas deben de tener algunas condiciones como un nombre único, los datos de cada columna deben de ser del mismo tipo y no deben de existir columnas repetidas. En este caso ya se han preparado las tablas que a continuación se muestran para normalizar.

Tabla 3.3: *Registros de la tabla temperatura*

id	idAmb	idBiorepo	Date	Time	Outdoor
abcde1	abc1	xyz1	26-04-2022	09:08:07	30°C
abcde2	abc2	xyz2	26-04-2022	10:08:07	31°C
abcde3	abc3	xyz3	26-04-2022	11:08:07	30°C

Tabla 3.4: *Registros de la tabla Ambiente*

id	Main	RT-PCR	Cell Biology	BSL2-Enhanced
abc1	25°C	21°C	25°C	23°C
abc2	21°C	24°C	24°C	23°C
abc3	22°C	23°C	25°C	23°C

La primera forma indica que *Los atributos deben de tener atomicidad y tener un identificador único* [29].

- Tabla 3.3: Su identificador es el id y los atributos (a excepción del Outdoor) tiene atomicidad ya que solo manejan un único valor en cada celda. El atributo Outdoor presenta más de un valor por lo cual en el registro solo deben de ingresarse números sin la unidad de temperatura. Un caso similar estaría presente si el Date y Time estuvieran en un solo atributo.
- Tabla 3.4: Su identificador es el id y los atributos presentan un caso similar al atributo Outdoor.

Tabla 3.5: *Registros en la tabla biorepositorio*

Id	bsl3 ultra2	bsl2 ultra1	bsl2 ultra2	bsl3 neve- ra	cell bio- logy	main lab	rt pcr neve- ra	hori- zon- tal	verti- cal	bsl3 ul- tra1
xyz1	-62°C	-63°C	-67°C	2°C	0°C	4°C	1°C	- 26°C	- 18°C	- 69°C
xyz2	-62°C	-63°C	-67°C	2°C	2°C	3°C	2°C	- 24°C	- 19°C	- 69°C
xyz3	-62°C	-61°C	-68°C	4°C	1°C	4°C	1°C	- 22°C	- 18°C	- 69°C

- Tabla 3.5: Presenta el mismo caso que la tabla ambiente, por lo cual, se ha dado la misma solución.

La segunda forma indica *Se haya cumplido la regla uno y que todos los atributos que no son clave primaria tiene una dependencia funcional respecto a la clave o claves con las que cuenta el esquema*, es decir, que todos los atributos deben de tener relación directa con la llave o llaves primaria(s) [29].

- Tabla 3.3: Todos los atributos si tienen dependencia de la clave primaria ya que en su registro no existe una llave que haga referencia a los mismos atributos de la tabla. Para el caso de las llaves secundarias la regla no aplica.
- Tabla 3.4: Sus registros son de temperaturas específicas de la tabla, por lo cual, tienen dependencia a su Id lo que permite cumplir esta forma.
- Tabla 3.5: Tiene el mismo caso que la tabla ambiente.

La tercera forma indica *Todos los atributos que no son llave primaria no tienen una dependencia transitiva a esta*, a partir de un atributo X que tenga dependencia a la llave primaria otro atributo Y tenga dependencia al atributo X [29].

- Tabla 3.3: No tiene una variable que le haga trascendencia a otra. Para este caso la trascendencia hubiera existido si la tabla Ambiente y Biorepositorio estuvieran unidas en una sola y en esa misma tabla existieran los atributo ambiente y biorepositorio que provocan la trascendencia.
- Tabla 3.4: La regla no aplica ya que no presenta trascendencia de atributos.
- Tabla 3.5: El mismo caso de la tabla Ambiente.

Debido a que la BD del sistema de monitoreo contiene un número reducido de tablas y las formas revisadas demostraron que el manejo de datos es adecuado, no se continua con otras formas ya que la tercera forma ha llegado a los resultados requeridos para el sistema. Las Figuras 3.6, 3.7 y 3.8 muestran los resultados de la normalización.

Tabla 3.6: *Tabla temperatura normalizada*

id	idAmb	idBiorepo	Fecha	Outdoor
abcde1	abc1	xyz1	26-04-2022 09:08:07	30
abcde2	abc2	xyz2	26-04-2022 10:08:07	31
abcde3	abc3	xyz3	26-04-2022 11:08:07	30

Tabla 3.7: *Tabla Ambiente normalizada*

id	Main	RT-PCR	Cell Biology	BSL2-Enhanced
abc1	25	21	25	23
abc2	21	24	24	23
abc3	22	23	25	23

Tabla 3.8: *Tabla biorepositorio normalizada*

Id	bsl3 ultra2	bsl2 ultra1	bsl2 ultra2	bsl3 neve- ra	cell bio- logy	main lab	rt pcr neve- ra	hori- zon- tal	verti- cal	bsl3 ul- tra1
xyz1	-62	-63	-67	2	0	4	1	-26	-18	-69
xyz2	-62	-63	-67	2	2	3	2	-24	-19	-69
xyz3	-62	-61	-68	4	1	4	1	-22	-18	-69

3.5. Ciberseguridad

La seguridad es un aspecto importante a tener en cuenta en el desarrollo de este sistema web, esto se debe a que las solicitudes son a través de Internet. Si bien, a simple vista Internet puede parecer seguro pero siempre existe la posibilidad de no serlo, en él se ocultan peligros acechando a atacar sistemas desprevenidos y de baja seguridad, ya sean malwares o ciberataques que pueden ocasionar el acceso no autorizado a datos privados o afectaciones al sistema web, solo por mencionar algunos.

Para proteger el sistema de monitoreo se han tomado en cuenta algunas de las amenazas más comunes y conocidos en tema de páginas web y BD que se muestran en la Tabla 3.9, para las cuales, Django ya cuenta con funciones de seguridad para evitar dichas amenazas [11, 1], algunas de estas ya se encuentran implementadas y otras han requerido una implementación manual.

Para el caso de acceso al sistema, se ha considerado implementar una autenticación, autorizaciones y control de accesos con la finalidad de que el LGVH sea el único en acceder al sistema de monitoreo.

Tabla 3.9: *Algunas amenazas que pueden llegar a afectar al sistema de monitoreo.*

Amenaza	Descripción
Falsificación de solicitudes en sitios cruzados (CSRF)	Uso mal intencionado de las credenciales de un usuario sin la autorización de dicho usuario.
Secuencias de comandos en sitios cruzados (XSS)	Inserción de código HTML no perteneciente a la página web original, robo de información de cookies y sesiones.
Inyección SQL	Inserción de fragmentos SQL en la BD, lo cual puede provocar la pérdida o extracción de datos.

Esta implementación de seguridad tiene la finalidad de mantener la integridad del sistema de monitoreo ante posibles ataques cibernéticos.

Capítulo 4

Pruebas y resultados del sistema de monitoreo

Realizar pruebas a un sistema permite garantizar el cumplimiento de los requerimientos dados por el cliente, dichas pruebas también pueden encontrar defectos, anomalías o elementos no funcionales que pueden convertirse en problemas para el sistema, por este motivo, se utilizan pruebas de validación que se enfocan en demostrar que el sistema hace lo que se espera. Para realizar las pruebas al sistema se consideraron los requerimientos vistos en el Capítulo 3 y el documento ERS del Apéndice A, se utilizaron registros de temperaturas ficticios con fecha de inicio del 26-03-2017 con tiempo 00:08:07 y fecha final del 26-03-2022 con tiempo 00:08:07 dentro del máximo y mínimo permitido para cada área y biorepositorio, agregando algunas excepciones para crear una simulación más real. También se incluyeron registros posteriores de forma automática extraídos de los reportes de sensores y tomados del pronóstico climático.

El método de prueba utilizado a continuación es de caja negra; este método permite verificar el funcionamiento del sistema usando datos o acciones de entrada al sistema que permita el recibir una salida, esto comprueba que el módulo encargado del requerimiento hace o no lo esperado. Los documentos que muestran las Tablas 4.1, 4.2, 4.3, 4.4, 4.5, 4.6 contienen una estructura de reporte que muestra la información de las pruebas realizadas.

Tabla 4.1: *Caso de prueba del primer requerimiento. El elemento de color azul con datos climáticos es el resultado esperado.*

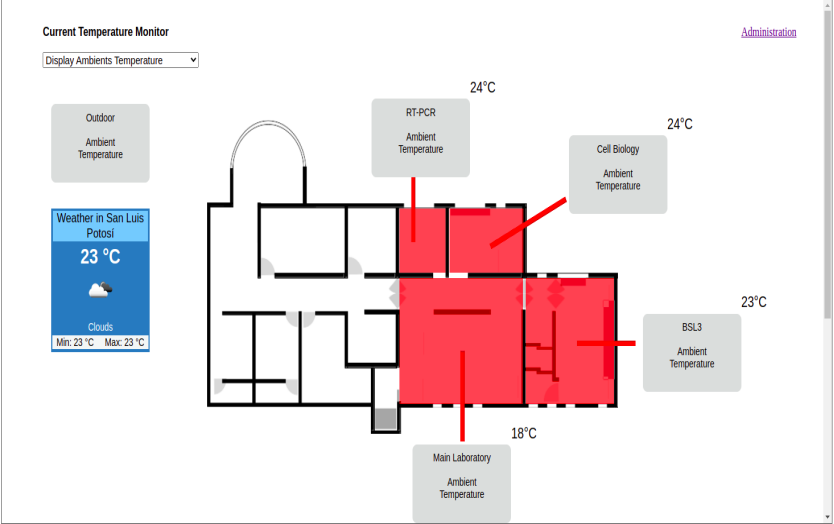
Ambiente:	Localhost.	
Autor	Julio César Álvarez Charqueño	
Req. 1	Propósito	Realizar la petición a la página de clima y el resultado se muestre en la interfaz de Ambiente.
	Acciones	Cargar la página o seleccionar la opción del selector Ambiente o Biorepositorios.
	datos de entrada	Ninguno
	Salida esperada	Muestre un elemento con la información climatológica.
	Salida obtenida	Elemento con la información climatológica.
Evidencia		

Tabla 4.2: Caso de prueba del segundo requerimiento. El selector encima de la tabla permite seleccionar distintos periodos de búsqueda por fechas, además existe una opción que permite la búsqueda entre dos fechas específicas elegidas por el usuario.

Ambiente:	Localhost.	
Autor	Julio César Álvarez Charqueño	
Req. 2	Propósito:	Comprobar que las consultas por fechas sean las adecuadas según la opción de búsqueda del selector.
	Acciones	Seleccionar alguna opción del selector que tiene como opción predeterminada Display last week.
	datos de entrada	La opción seleccionada del selector.
	Salida esperada	Muestre una tabla con las fechas y temperaturas de todas las áreas o biorepositorios ordenadas por fecha del registro más reciente hasta la opción seleccionada del selector.
	Salida obtenida	Tabla con las fechas y temperaturas de todas las áreas o biorepositorios ordenadas por fecha del registro más reciente hasta la opción seleccionada del selector.

Min: 17°C Max: 17°C

<

Tabla 4.3: Caso de prueba del tercer requerimiento. El usuario no puede interactuar con dichos colores ya que el sistema identifica las temperaturas automáticamente.

Ambiente:	Localhost.																																																																																																																																					
Autor	Julio César Álvarez Charqueño																																																																																																																																					
Req. 3	Propósito:	Verificar que en la tabla de temperaturas exista una marca roja para temperaturas mayores y una azul para menores al rango aceptado por las áreas y biorepositorios.																																																																																																																																				
	Acciones	Al elegir la opción pantalla Ambiente o Biorepositorio del selector o elegir una opción de consulta por fecha del selector de la tabla.																																																																																																																																				
	datos de entrada	Los registros de las temperaturas de áreas o biorepositorios.																																																																																																																																				
	Salida esperada	Muestre una marca roja o azul en temperaturas que están fuera del limite permitido.																																																																																																																																				
	Salida obtenida	Marca roja o azul en temperaturas que están fuera del limite permitido.																																																																																																																																				
Evidencia	<div><div><div>Clouds</div><div>Min: 21 °C Max: 21 °C</div></div><div><div><div>-27°C</div><div>Main Laboratory Horizontal Freezer Temperature</div></div><div><div>-69°C</div><div>BSL2 #1 Ultra-Low Freezer Temperature</div></div><div><div>3°C</div><div>Main Laboratory Fridge Temperature</div></div><div><div>-17°C</div><div>Main Laboratory Vertical Freezer Temperature</div></div><div><div>-68°C</div><div>BSL3 #1 Ultra-Low Freezer Temperature</div></div></div><div><div>Display last week</div><div><table><tr><th>Date</th><th>Time</th><th>BSL2 #2 Ultra-Low</th><th>RT-PCR Fridge</th><th>Cell Biology Fridge</th><th>BSL3 Fridge</th><th>BSL3 #2 Ultra-Low</th><th>BSL3 #1 Ultra-Low</th><th>Vertical Freezer</th><th>Main lab Fridge</th><th>BSL2 #1 Ultra-Low</th><th>Horizontal Ultra-Low</th></tr><tr><td>June 26, 2022</td><td>23:17:10</td><td>-59°C</td><td>-1°C</td><td>4°C</td><td>0°C</td><td>-62°C</td><td>-68°C</td><td>-17°C</td><td>3°C</td><td>-69°C</td><td>-27°C</td></tr><tr><td>June 26, 2022</td><td>22:17:10</td><td>-58°C</td><td>0°C</td><td>5°C</td><td>3°C</td><td>-66°C</td><td>-60°C</td><td>-15°C</td><td>0°C</td><td>-67°C</td><td>-24°C</td></tr><tr><td>June 26, 2022</td><td>21:17:10</td><td>-71°C</td><td>5°C</td><td>0°C</td><td>2°C</td><td>-71°C</td><td>-70°C</td><td>-20°C</td><td>0°C</td><td>-58°C</td><td>-19°C</td></tr><tr><td>June 26, 2022</td><td>20:17:10</td><td>-71°C</td><td>3°C</td><td>0°C</td><td>0°C</td><td>-58°C</td><td>-59°C</td><td>-17°C</td><td>2°C</td><td>-69°C</td><td>-22°C</td></tr><tr><td>June 26, 2022</td><td>19:17:10</td><td>-68°C</td><td>3°C</td><td>0°C</td><td>3°C</td><td>-59°C</td><td>-60°C</td><td>-13°C</td><td>4°C</td><td>-62°C</td><td>-26°C</td></tr><tr><td>June 26, 2022</td><td>18:17:10</td><td>-64°C</td><td>3°C</td><td>4°C</td><td>0°C</td><td>-60°C</td><td>-67°C</td><td>-15°C</td><td>1°C</td><td>-70°C</td><td>-18°C</td></tr><tr><td>June 26, 2022</td><td>17:17:10</td><td>-58°C</td><td>5°C</td><td>4°C</td><td>2°C</td><td>-64°C</td><td>-68°C</td><td>-16°C</td><td>3°C</td><td>-61°C</td><td>-23°C</td></tr><tr><td>June 26, 2022</td><td>16:17:10</td><td>-67°C</td><td>2°C</td><td>2°C</td><td>1°C</td><td>-67°C</td><td>-67°C</td><td>-21°C</td><td>2°C</td><td>-61°C</td><td>-27°C</td></tr><tr><td>June 26, 2022</td><td>15:17:10</td><td>-62°C</td><td>0°C</td><td>-1°C</td><td>0°C</td><td>-71°C</td><td>-60°C</td><td>-21°C</td><td>0°C</td><td>-71°C</td><td>-22°C</td></tr><tr><td>June 26, 2022</td><td>14:17:10</td><td>-62°C</td><td>3°C</td><td>5°C</td><td>5°C</td><td>-60°C</td><td>-70°C</td><td>-13°C</td><td>3°C</td><td>-71°C</td><td>-30°C</td></tr></table></div></div></div>		Date	Time	BSL2 #2 Ultra-Low	RT-PCR Fridge	Cell Biology Fridge	BSL3 Fridge	BSL3 #2 Ultra-Low	BSL3 #1 Ultra-Low	Vertical Freezer	Main lab Fridge	BSL2 #1 Ultra-Low	Horizontal Ultra-Low	June 26, 2022	23:17:10	-59°C	-1°C	4°C	0°C	-62°C	-68°C	-17°C	3°C	-69°C	-27°C	June 26, 2022	22:17:10	-58°C	0°C	5°C	3°C	-66°C	-60°C	-15°C	0°C	-67°C	-24°C	June 26, 2022	21:17:10	-71°C	5°C	0°C	2°C	-71°C	-70°C	-20°C	0°C	-58°C	-19°C	June 26, 2022	20:17:10	-71°C	3°C	0°C	0°C	-58°C	-59°C	-17°C	2°C	-69°C	-22°C	June 26, 2022	19:17:10	-68°C	3°C	0°C	3°C	-59°C	-60°C	-13°C	4°C	-62°C	-26°C	June 26, 2022	18:17:10	-64°C	3°C	4°C	0°C	-60°C	-67°C	-15°C	1°C	-70°C	-18°C	June 26, 2022	17:17:10	-58°C	5°C	4°C	2°C	-64°C	-68°C	-16°C	3°C	-61°C	-23°C	June 26, 2022	16:17:10	-67°C	2°C	2°C	1°C	-67°C	-67°C	-21°C	2°C	-61°C	-27°C	June 26, 2022	15:17:10	-62°C	0°C	-1°C	0°C	-71°C	-60°C	-21°C	0°C	-71°C	-22°C	June 26, 2022	14:17:10	-62°C	3°C	5°C	5°C	-60°C	-70°C	-13°C	3°C	-71°C	-30°C
Date	Time	BSL2 #2 Ultra-Low	RT-PCR Fridge	Cell Biology Fridge	BSL3 Fridge	BSL3 #2 Ultra-Low	BSL3 #1 Ultra-Low	Vertical Freezer	Main lab Fridge	BSL2 #1 Ultra-Low	Horizontal Ultra-Low																																																																																																																											
June 26, 2022	23:17:10	-59°C	-1°C	4°C	0°C	-62°C	-68°C	-17°C	3°C	-69°C	-27°C																																																																																																																											
June 26, 2022	22:17:10	-58°C	0°C	5°C	3°C	-66°C	-60°C	-15°C	0°C	-67°C	-24°C																																																																																																																											
June 26, 2022	21:17:10	-71°C	5°C	0°C	2°C	-71°C	-70°C	-20°C	0°C	-58°C	-19°C																																																																																																																											
June 26, 2022	20:17:10	-71°C	3°C	0°C	0°C	-58°C	-59°C	-17°C	2°C	-69°C	-22°C																																																																																																																											
June 26, 2022	19:17:10	-68°C	3°C	0°C	3°C	-59°C	-60°C	-13°C	4°C	-62°C	-26°C																																																																																																																											
June 26, 2022	18:17:10	-64°C	3°C	4°C	0°C	-60°C	-67°C	-15°C	1°C	-70°C	-18°C																																																																																																																											
June 26, 2022	17:17:10	-58°C	5°C	4°C	2°C	-64°C	-68°C	-16°C	3°C	-61°C	-23°C																																																																																																																											
June 26, 2022	16:17:10	-67°C	2°C	2°C	1°C	-67°C	-67°C	-21°C	2°C	-61°C	-27°C																																																																																																																											
June 26, 2022	15:17:10	-62°C	0°C	-1°C	0°C	-71°C	-60°C	-21°C	0°C	-71°C	-22°C																																																																																																																											
June 26, 2022	14:17:10	-62°C	3°C	5°C	5°C	-60°C	-70°C	-13°C	3°C	-71°C	-30°C																																																																																																																											

Tabla 4.4: Caso de prueba del cuarto requerimiento. Los campos debajo de la tabla permiten al usuario calcular una nueva estadística en base a un rango de fecha. El boton encima de la tabla permite mostrar la tabla de temperaturas.

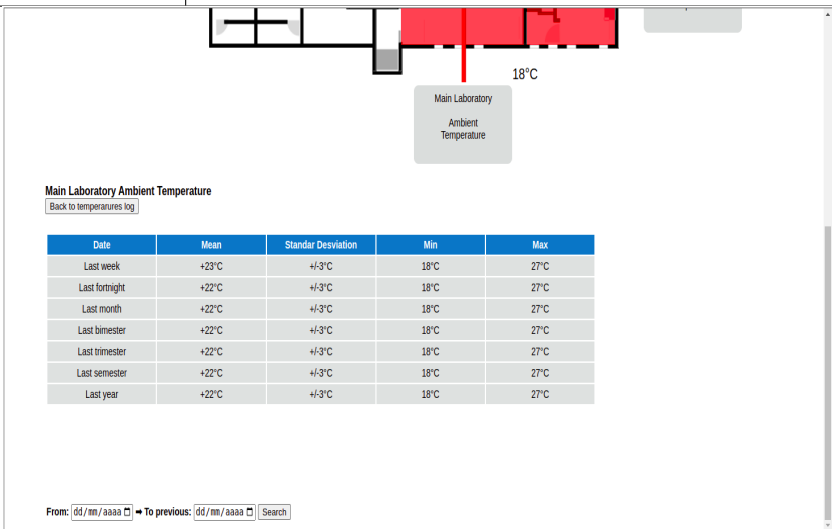
Ambiente:	Localhost.	
Autor	Julio César Álvarez Charqueño	
Req. 4	Propósito:	Verificar que la tabla muestre los cálculos correctos.
	Acciones	Dar clic en recuadro gris asignado a una área o biorepositorio.
	Datos de entrada	Opción seleccionada.
	Salida esperada	Muestre una tabla con las estadísticas con siete diferentes lapsos de fecha del área o biorepositorio seleccionado.
	Salida obtenida	Tabla con las estadísticas con siete diferentes lapsos de fecha del área o biorepositorio seleccionado.
Evidencia	<div></div>	

Tabla 4.5: Caso de prueba del quinto requerimiento. Evidencia incompleta debido a la cantidad de datos en la gráfica, la parte faltante muestra las altas temperaturas similar a las que se muestran.

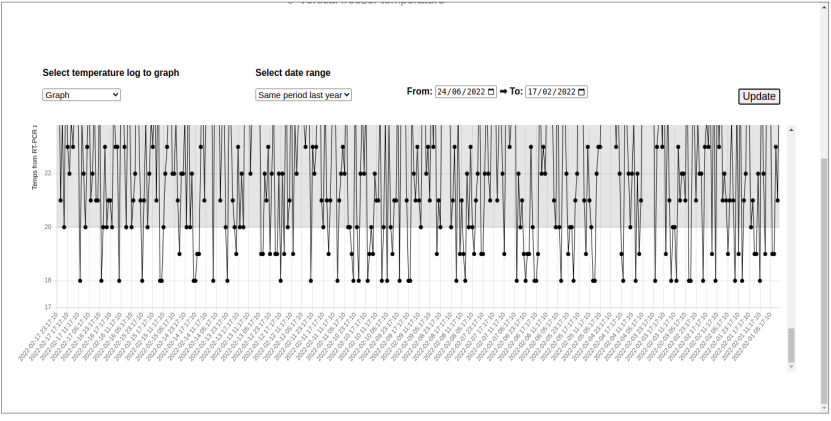
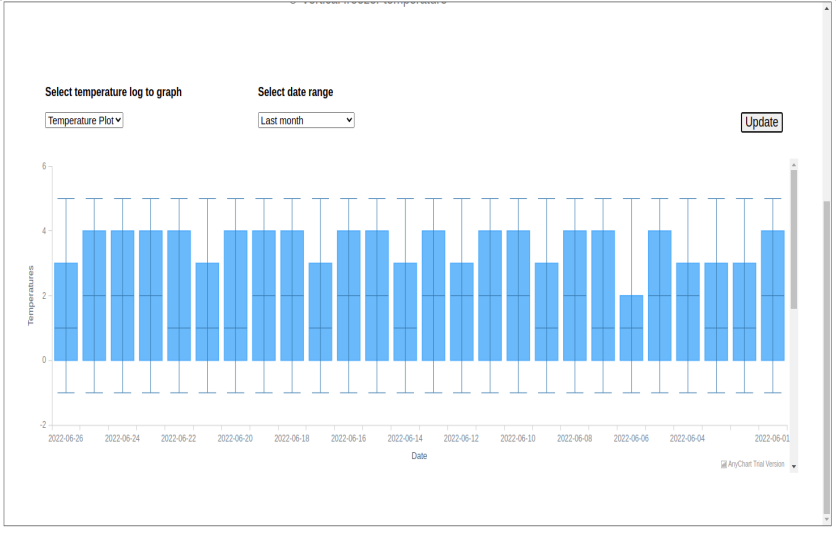
Ambiente:	Localhost.	
Autor	Julio César Álvarez Charqueño	
Req. 5	Propósito:	Que la información de la gráfica lineal sea correcta.
	Acciones	Seleccionar un registro de temperatura y la opción <i>graphic</i> del selector tipo de gráfica, elegir un rango de fecha y dar clic en el botón actualizar.
	Datos de entrada	Opciones seleccionadas tanto del área o biorepositorio, tipo de gráfica y rango de fecha.
	Salida esperada	Muestre una o varias gráficas (según el rango de fechas) de linea con fecha y hora de los registros dentro del rango de tiempo seleccionado, la o las gráficas por mes, área marcada con el máximo y mínimo de temperaturas y nombre del área o biorepositorio seleccionado.
	Salida obtenida	Una o varias gráficas (según el rango de fechas) de linea con fecha y hora de los registros dentro del rango de tiempo seleccionado, la o las gráficas por mes, área marcada con el máximo y mínimo de temperaturas y nombre del área o biorepositorio seleccionado.
Evidencia		

Tabla 4.6: Caso de prueba del sexto requerimiento. Las gráficas se construyen por mes y se muestra de forma descendente al igual que las gráficas de línea del quinto requerimiento.

Ambiente:	Localhost.	
Autor	Julio César Álvarez Charqueño	
Req. 6	Propósito:	Representar los datos mediante una gráfica de caja de manera correcta.
	Acciones	Seleccionar un registro de temperatura, seleccionar la opción Diagrama de temperatura del selector Tipo de gráfica, elegir un rango de fecha y dar clic en el botón actualizar.
	Datos de entrada	Opciones seleccionadas tanto del área o bio-repositorio, tipo de gráfica y rango de fecha.
	Salida esperada	Muestre una o varias gráficas (según el rango de fechas) de caja con la información de las temperaturas.
	Salida obtenida	Una o varias gráficas (según el rango de fechas) de caja con la información de las temperaturas.
Evidencia		

Para estas pruebas no se ha considerado el requerimiento ocho y nueve debido a que el primero es una implementación y el segundo es sobre seguridad.

Las pruebas de caja negra se aplican a partir de los RF dados por el cliente limitando su alcance en otros aspectos del sistema por lo que pueden dejar márgenes a errores producidos durante el desarrollo, es por ello que se realizaron pruebas de caja blanca.

Las pruebas de caja blanca permiten verificar la ejecución de al menos una vez de todos los caminos de cada módulo que involucra a los requerimientos. En el camino se revisan las decisiones lógicas, los ciclos y las estructuras de los datos, todo ello tiene como fin comprobar que el flujo interno del sistema no tenga fallas o código poco adecuado causante de errores. Para implementar estas pruebas se utiliza el método de camino básico que permite medir la complejidad lógica del diseño apoyándose en diagramas de flujo y grafos de flujo dirigidos [26].

La Figura 4.1 se toma como ejemplo para evaluar la complejidad ciclomática del módulo, esta evaluación da a conocer el número de pruebas que deben realizarse para comprobar su correcto funcionamiento y, además, el número de caminos independientes [26]. La fórmula que se utiliza para dicho cálculo es:

$$V(G) = a - n + 2$$

donde a es el número de aristas y n es el número de nodos que muestra el diagrama de grafo. El siguiente cálculo pertenece a la Figura 4.1 el cuál

$$V(G) = 4 - 4 + 2 = 2$$

dicta que el requerimiento necesita dos pruebas y tiene dos caminos independientes.

Camino1 : A, B, D

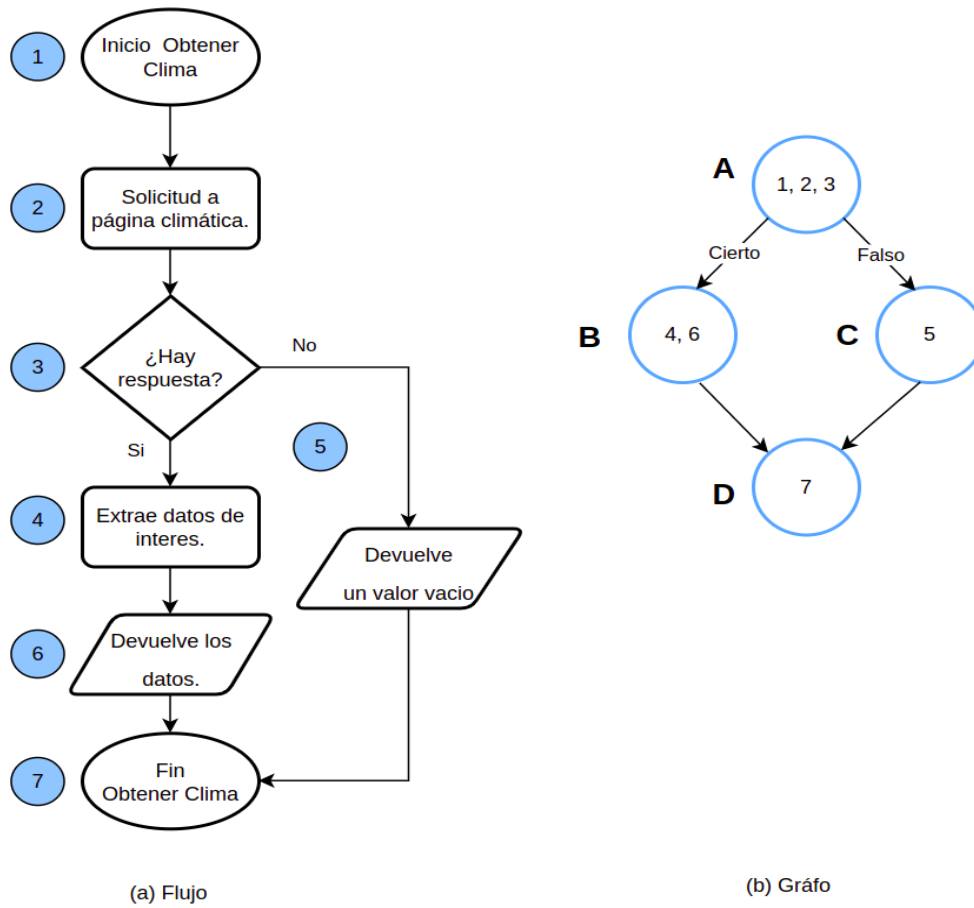


Figura 4.1: Diagrama de flujo y grafo del primer requerimiento. Su código fuente se encuentra en la sección 3.2 del apéndice B.

Camino2 : A, C, D

Esta metodología se ha implementado en cada requerimiento (excepto el ocho y nueve) que ha dado como resultado los registros de la Tabla 4.7 y las Figuras 4.1, 4.2, 4.3, 4.4, 4.5, 4.6, 4.8, 4.10.

Tabla 4.7: *Cálculo de la Complejidad Ciclomática (CC) y obtención de los Caminos Independientes (CI) de cada grafo de flujo.*

Fig	CC	CI
4.2	$V(G) = 5-5+2 = 2$	1: A,B,D 2: A,C,D
4.3	$V(G) = 8-7+2 = 3$	1: A,B,D 2: A,C,E,D 3: A,C,F,D
4.5	$V(G) = 25-19+2 = 8$	1: A,B,D,H,J 2: A,B,E,K,H,J 3: A,B,E,L,H,J 4: A,C,F,M,P,Q,R,S,J 5: A,C,F,M,P,R,S,J 6: A,C,F,N,P,R,S,J 7: A,C,G,Ñ,O,S,J 8: A,C,G,O,S,J
4.8	$V(G) = 21-17+2 = 6$	1: A,B,D,F,G,I,J,L,M,N,Ñ,O,P,Q 2: A,B,D,F,G,I,J,L,M,Ñ,O,P,Q 3: A,B,D,F,G,I,J,O,P,Q 4: A,B,D,F,H,I,J,O,P,Q 5: A,B,E,F,H,I,J,O,P,Q 6: A,C,Q
4.8	$V(G) = 33-26+2 = 9$	1: A,B,D,F,G,I,K,R,S,T,U,V,W,Y,Z,a,X,b,c,d,e,P,Q 2: A,B,D,F,G,I,K,R,S,T,U,V,W,Y,a,X,b,c,d,e,P,Q 3: A,B,D,F,G,I,K,R,S,T,U,V,X,b,c,d,e,P,Q 4: A,B,D,F,G,I,K,R,S,T,U,V,X,b,d,e,P,Q 5: A,B,D,F,G,I,K,R,S,T,e,P,Q 6: A,B,D,F,G,I,K,R,T,e,P,Q 7: A,B,D,F,H,I,K,R,T,e,P,Q 8: A,B,E,F,H,I,K,R,T,e,P,Q 9: A,C,Q
4.10	$V(G) = 12-10+2 = 4$	1: A,B,C,E,G,H,I,J,D 2: A,B,C,E,G,H,J,D 3: A,B,C,F,G,H,J,D 4: A,B,D

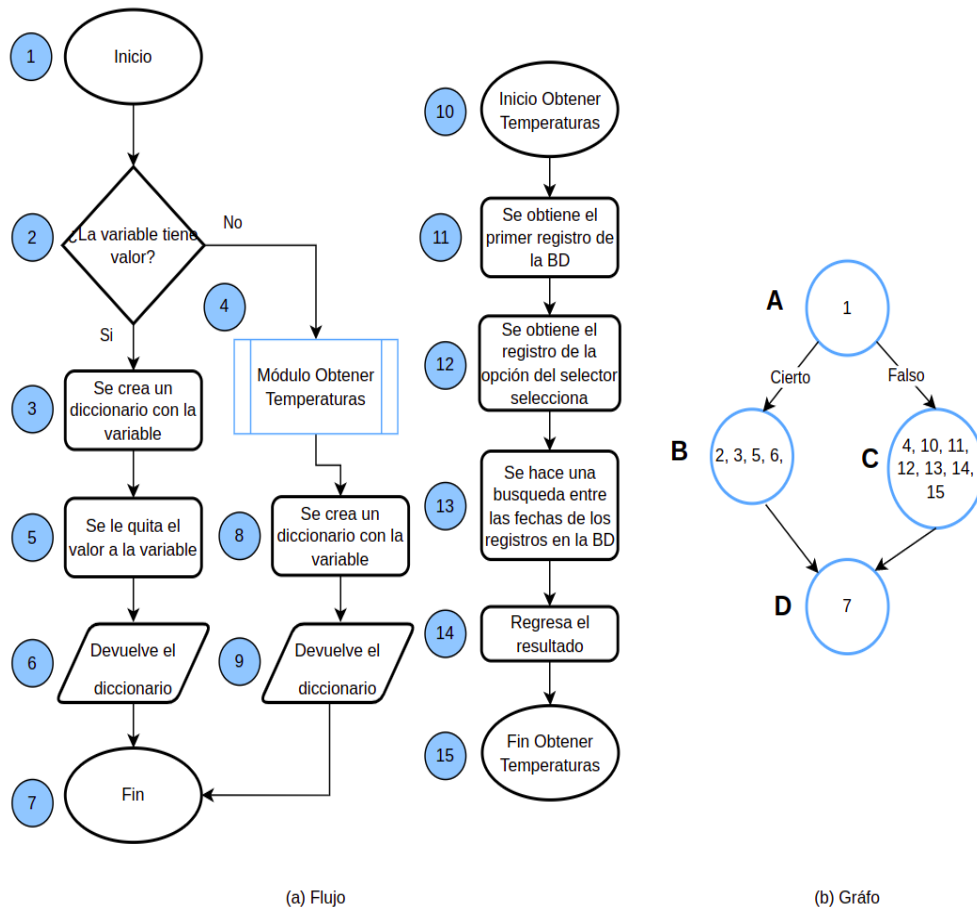


Figura 4.2: Diagrama de flujo y grafo del segundo requerimiento. Su código fuente se encuentra en la sección 4.2 y 5.2 del apéndice B.

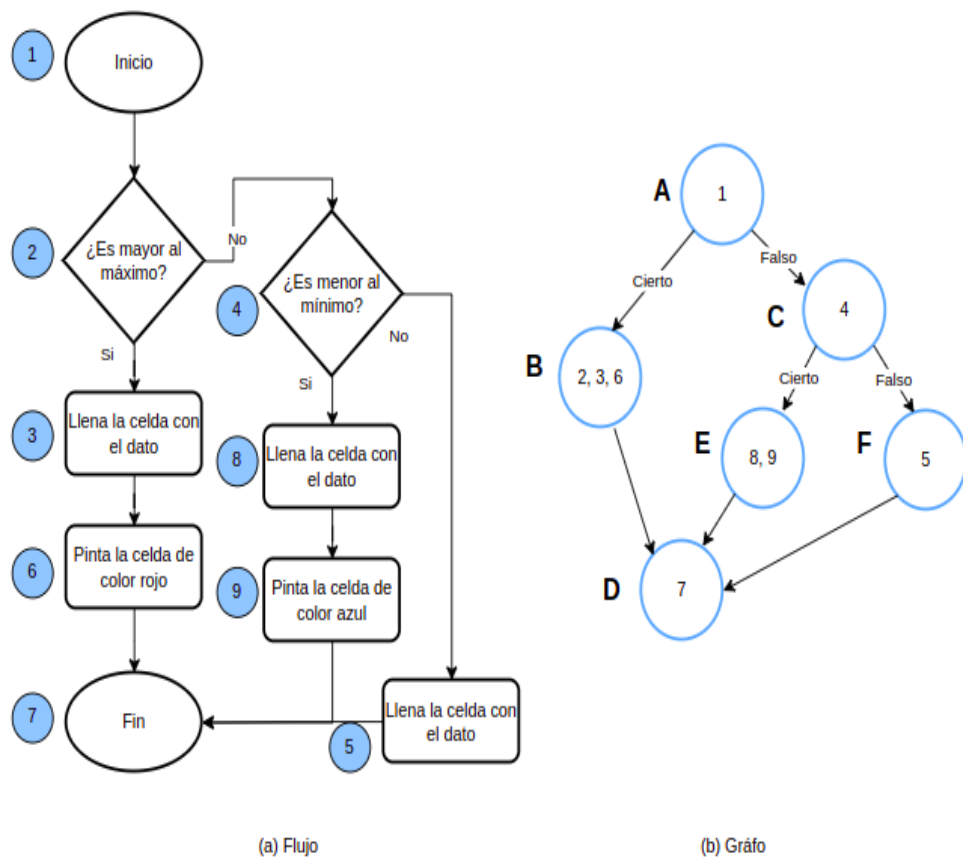


Figura 4.3: Diagrama de flujo y grafo del tercer requerimiento. Su código fuente se encuentra en la sección 3.6 del apéndice B.

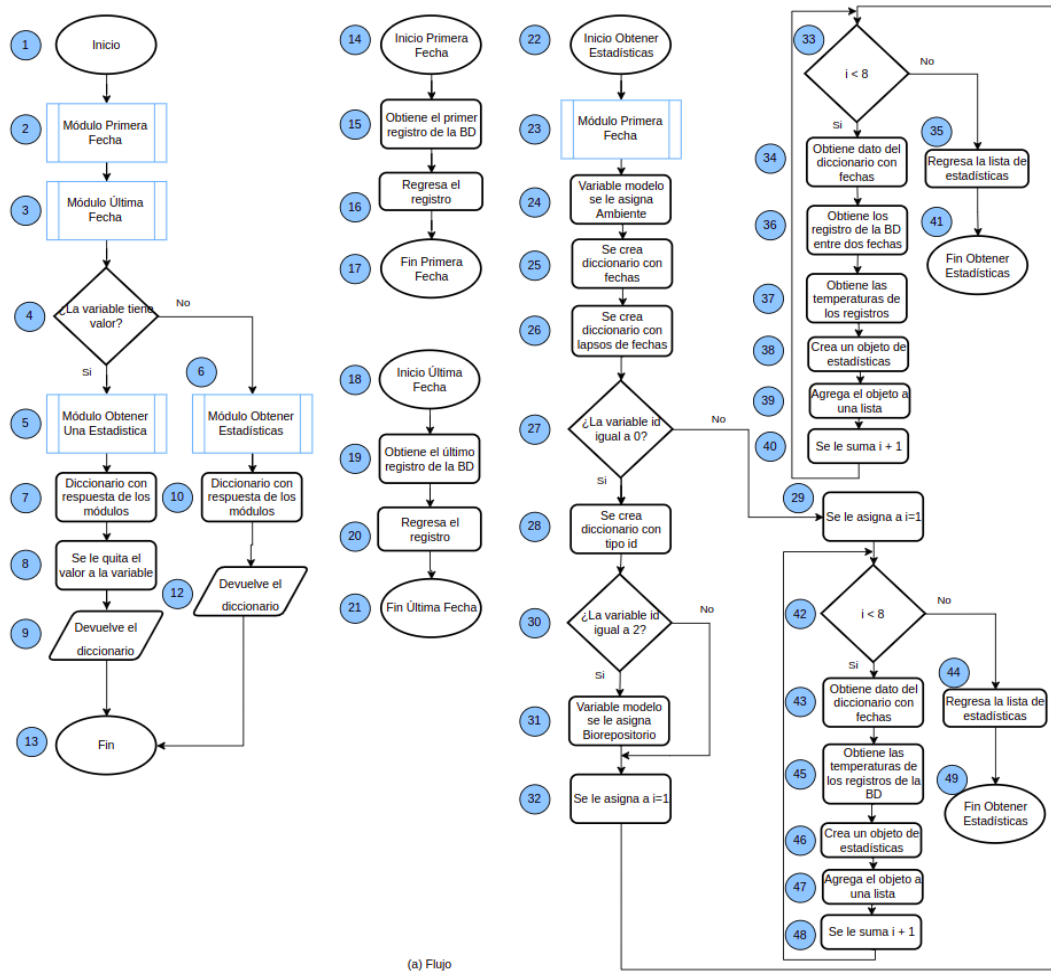


Figura 4.4: Primera parte del diagrama de flujo del cuarto requerimiento.
Su código fuente se encuentra en la sección 6 del apéndice B.

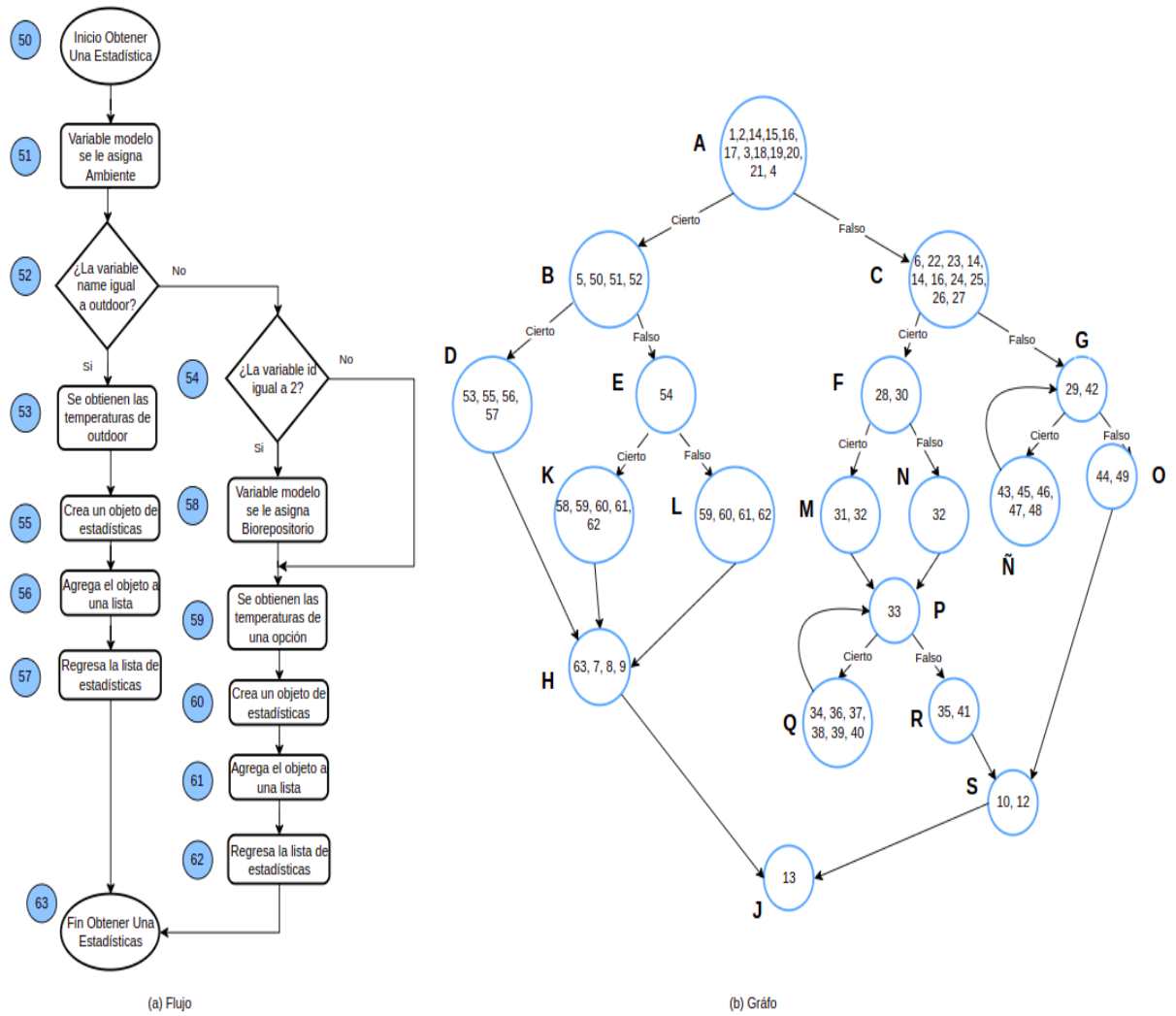


Figura 4.5: Segunda parte del diagrama de flujo y grafo del cuarto requerimiento. Su código fuente se encuentra en la sección 6 del apéndice B.

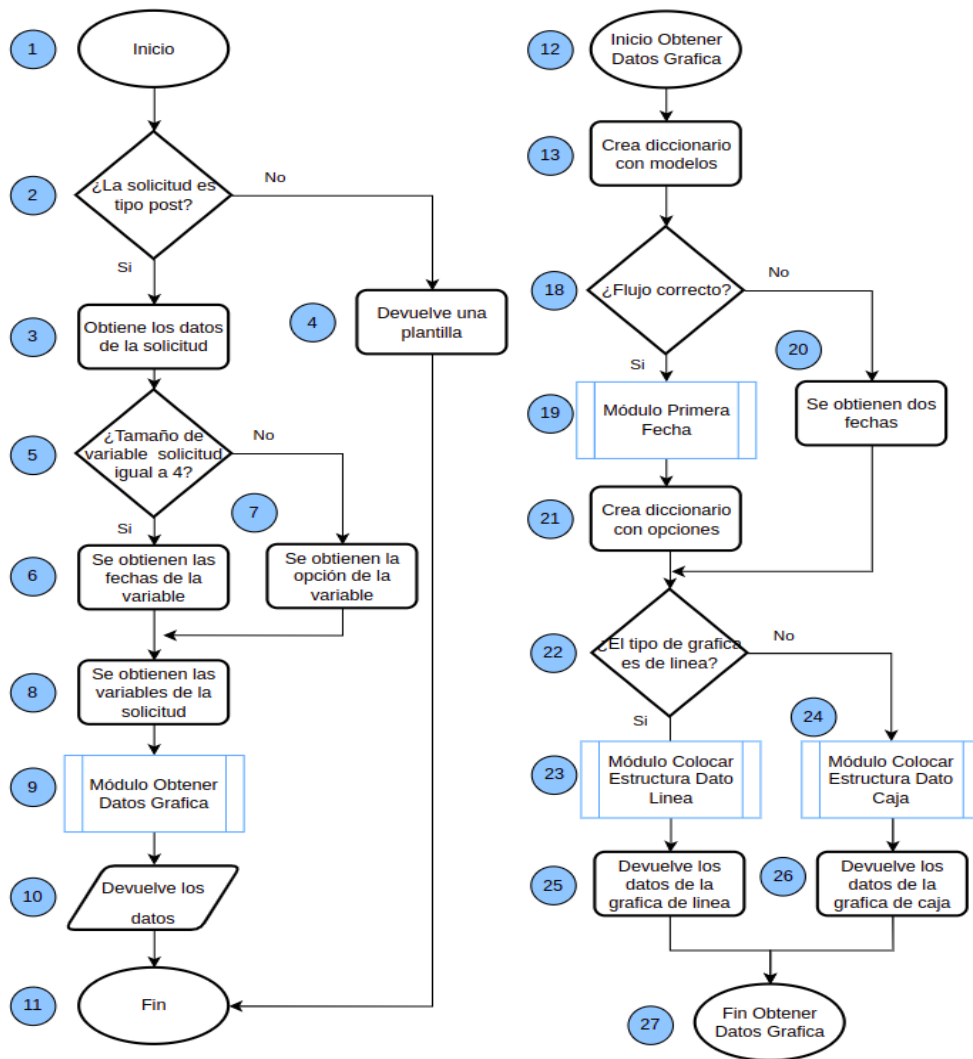


Figura 4.6: Sección en común del diagrama de flujo que comparte el quinto y sexto requerimiento. Su código fuente se encuentra en la sección 7.2 y 7.3 del apéndice B.

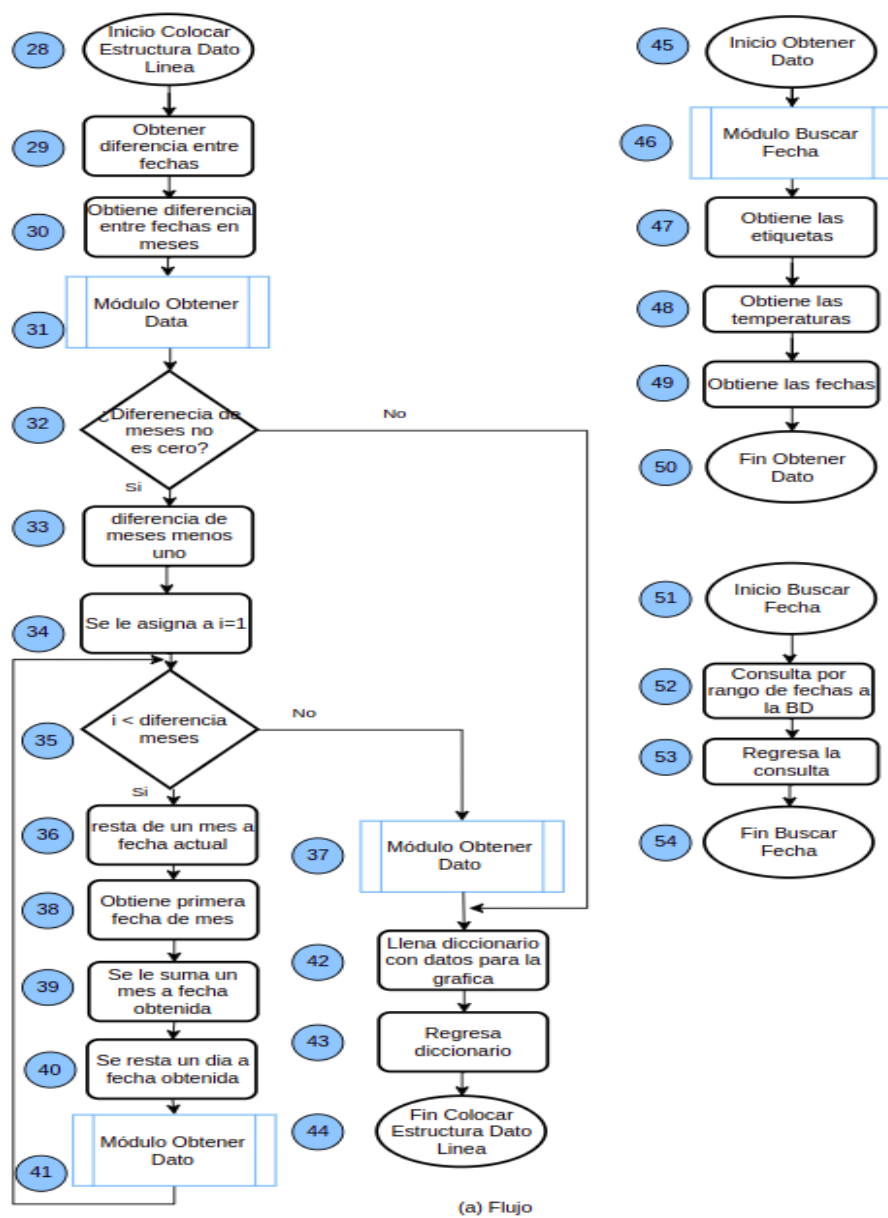


Figura 4.7: Diagrama de flujo del quinto requerimiento. Su código fuente se encuentra en la sección 7.4 y 7.5 del apéndice B.

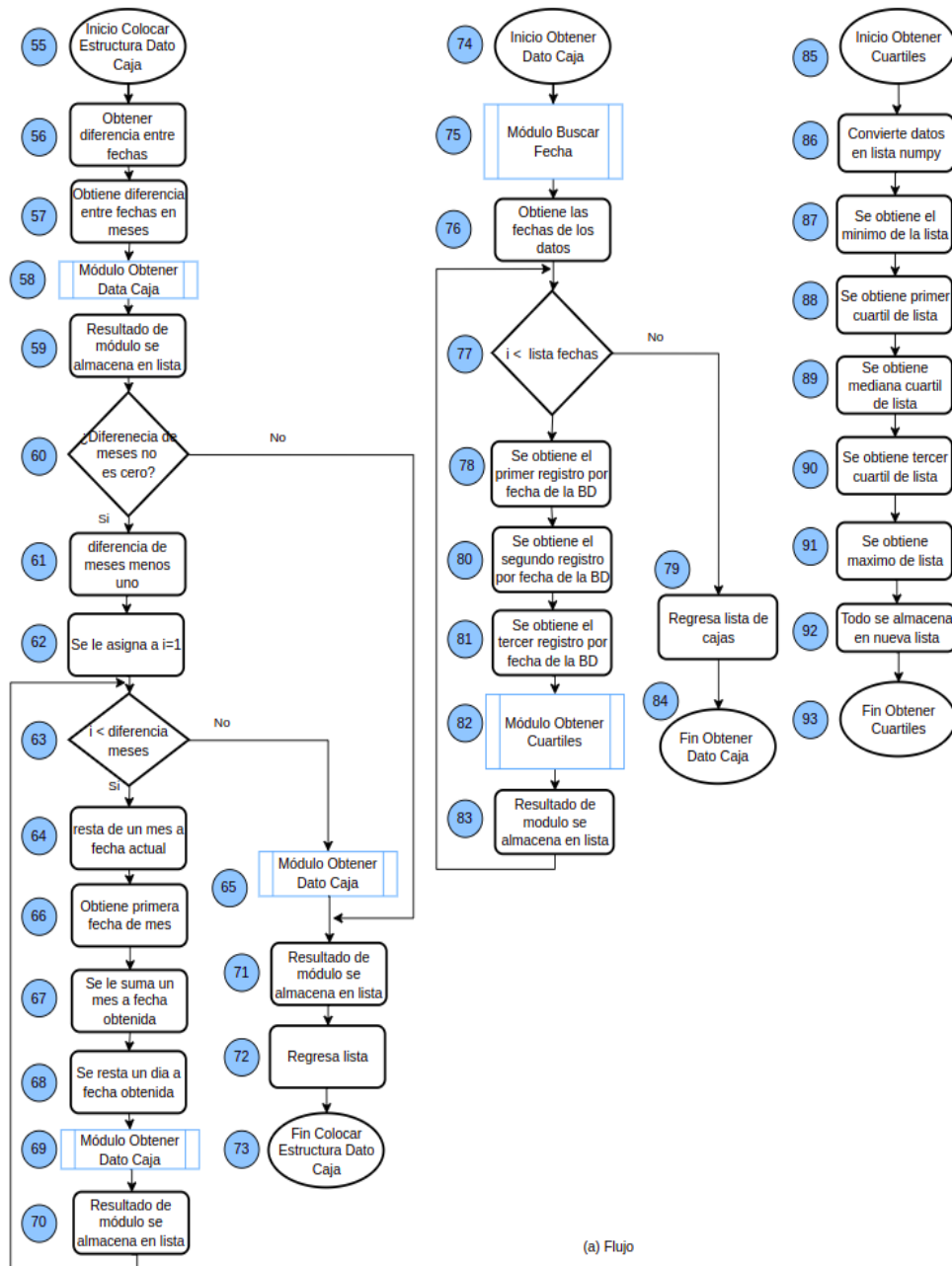


Figura 4.8: Diagrama de flujo del sexto requerimiento. Su código fuente se encuentra en la sección 7.6, 7.7 y 7.8 del apéndice B.

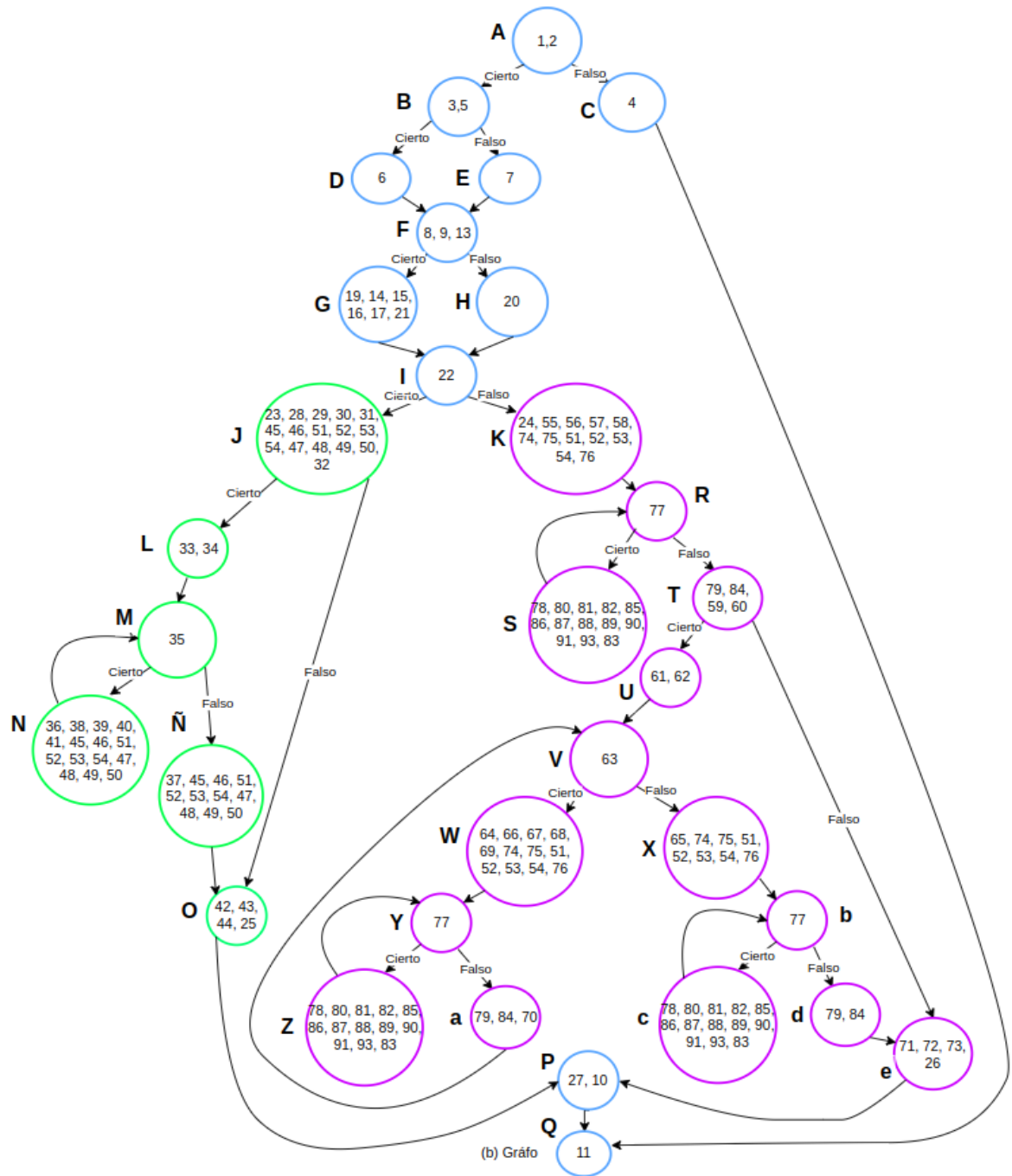


Figura 4.9: Grafo de flujo del quinto y sexto requerimiento. La sección en color verde representa el quinto requerimiento y la morada al sexto.

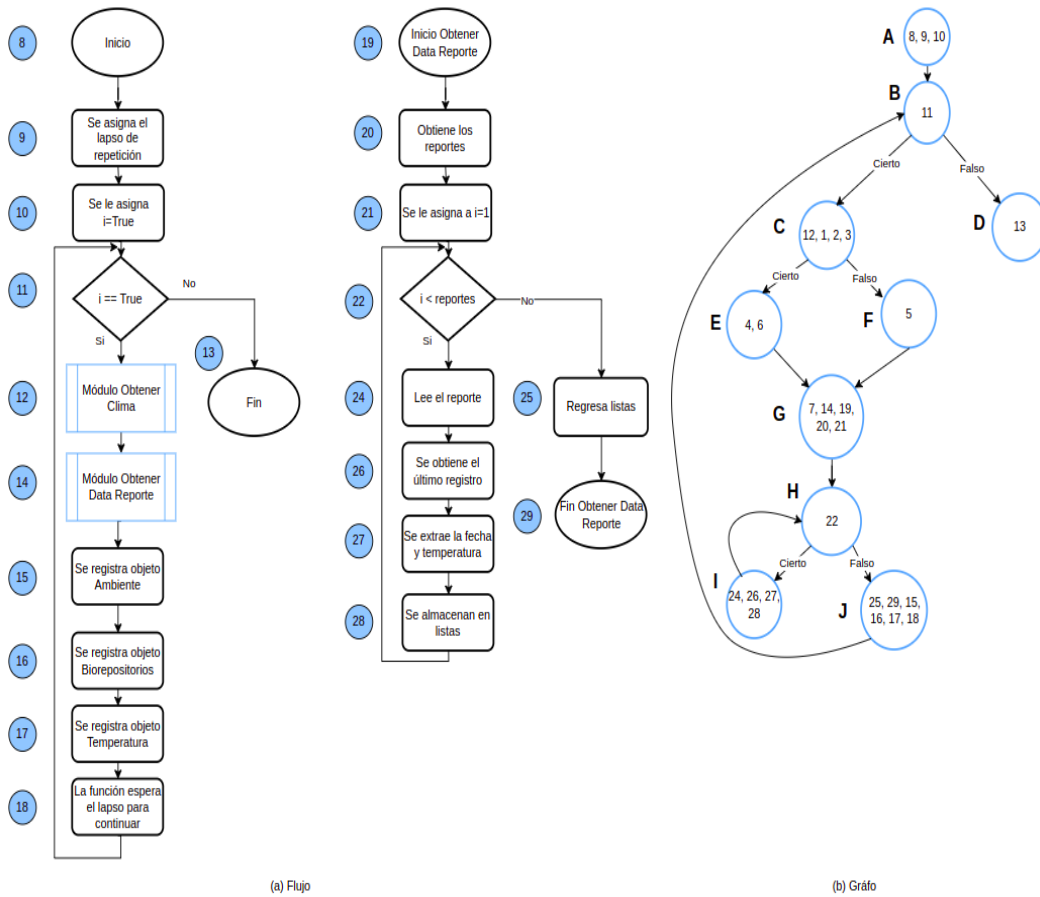


Figura 4.10: Diagrama de flujo y grafo del séptimo requerimiento. Su código fuente se encuentra en la sección 3.1 y 3.3 del apéndice B.

Conclusiones

En el presente trabajo se expuso el análisis y el proceso necesario para el desarrollo del sistema web de monitoreo de biorepositorios que involucran técnicas y tecnologías utilizadas.

Se le realizaron pruebas de caja negra al sistema que demostraron el cumplimiento de los requerimientos funcionales establecidos por el Laboratorio de Genómica Viral y Humana. Las pruebas de camino básico demostraron que los procesos estructurales internos cuentan con un correcto funcionamiento ya que todas sus instrucciones se ejecutan sin errores, también esta prueba permitió dejar en evidencia la complejidad de las instrucciones dentro del sistema lo que resulta de gran utilidad al momento de realizar mantenimientos. Las pruebas dejan en evidencia el cumplimiento de los requerimientos y de los objetivos de este trabajo, lo que permite concluir que el sistema es funcional y puede ser implementado para su uso.

Implementar un modelado con casos de uso, diagrama de actividades, diagrama de clases, patrones arquitectónicos, diagrama entidad-relación, diagrama relacional y normalización permitió construir una estructura con la cuál el mantenimiento, la extensibilidad y la implementación de nuevos módulos pueda realizarse con mayor facilidad, esto también facilitó su implementación en código. Todo el análisis y modelado se trabajó así debido a que este trabajo es la primera fase de un proyecto de mayor expansión.

Actualmente (tiempo en el que se escribió este trabajo) el Laboratorio de Genómica Vital y Humana se encuentra en una reestructuración por motivos de la actualización del manejo de bioseguridad nivel 2 al de nivel 3, por lo cuál, no fue posible probar el sistema en el entorno real. Con los resultados obtenidos de las pruebas (hechas en este documento) se espera que el sistema

brinde la solución a la problemática planteada y no solo sea una herramienta para registrar y consultar temperaturas, si no que se convierta en una herramienta de mejora para esta nueva fase del laboratorio.

Trabajo futuro

Algunos módulos a implementar durante el desarrollo de este trabajo se consideraron para las siguientes fases del proyecto debido a que abordan temas que se encuentran fuera de los objetivos presentes en este trabajo.

Como parte del trabajo futuro se están desarrollando diferentes módulos y funcionalidades que permitan el crecimiento y la mejorar del sistema actual:

- Derivado de los cambios en los diseños institucionales de las páginas web y a la reestructuración del laboratorio, el sistema web se debe adaptar a nuevas interfaces de usuario.
- Enviar mensajes de alarma por correo electrónico o por mensaje de texto cuando una temperatura sobrepase el rango permitido.
- Realizar respaldos de la BD.
- Hacer pruebas de seguridad.
- Una vez que se tiene una base de datos con información histórica, desarrollar un modelo de inteligencia artificial para la predicción de fallas en sensores y biorepositorios con la finalidad de mantener en buen estado los equipos.

Bibliografía

- [1] <https://www.djangoproject.com/>.
- [2] B. Albert Bernaus, J. *Nuevo Curso de Diseño y programación para Internet a fondo*. INFORBOOKS, S.L., España, 2001.
- [3] Octavio Medina y Marco Govea Anahí Martínez. <https://acominf.github.io/midasmap/>, 2016.
- [4] Andrei Bragarenco, Galina Marusic, and Ciufudean Calin. Layered architecture approach of the sensor software component stack for the internet of things applications. *WSEAS Transactions on Computer Research*, Volume 7:124–135, 01 2020.
- [5] L. M. Cabezas Granado. *Manual Imprescindible de PHP 5*. Grupo Anaya S.A, Madrid España, 2004.
- [6] Fco. J. Ceballos Sierra. *Microsoft C # Curso de programación, 2da edición*. Alfaomega Grupo Editor. S.A de C.V, México D.F., 2013.
- [7] Kristina Chodorow. *MongoDB: The Definitive Guide, Second Edition*. O'Reilly Media, Inc, Sebastopol, CA, USA, 2013.
- [8] Robin Dewson. *Beginning SQL server 2005 for developers: From Novice to Professional*. Apress, California, USA, 2006.
- [9] A. Fernández Montoro. *Python 3 al descubierto*. Alfaomega Grupo Editor S.A de C.V, D.F. México, 2013.
- [10] S. Garcia M. *Bases de Datos. Desde Chen hasta Codd con ORACLE*. RA-MA, MADRID, 2001.

- [11] S. Garcia M. *La guía definitiva de Django: Desarrolla aplicaciones web de forma rápida y sencilla*. Django Software Corporation, Celayita Polotitlán., México, 2015.
- [12] Christian A. García Sepúlveda. Estudio de la diversidad genómica kir y de sus interacciones con hla en infecciones naturales y experimentales por hiv-1.
- [13] W. Jason Gilmore and Robert H. Treat. *Beginning PHP and PostgreSQL 8: From Novice to Professional*. Apress, California, USA, 2006.
- [14] David. Hugues. *Fundamental of Computer Science Using Java*. Jones and Bartlett, Sudbury Massachusetts., 2002.
- [15] Lau Hui, Fauzan Khairi Che Harun, Yusmeeraz Yusof, and Norlaili Safri. Wireless temperature monitoring system for blood bank using zigbee. *Jurnal Teknologi*, 61, 03 2013.
- [16] Luis .H Ibáñez. *Base de Datos*. RA-MA Editorial, Paracuellos de Jarama, Madrid, 2012.
- [17] G. Jiménez. *Prototipo del sistema informático para monitoreo de condiciones ambientales en el biobanco de la UNAB empleando hardware y software libre*. PhD thesis, Universidad Autónoma de Bucaramanga., 2016.
- [18] David .C Kreines. *Oracle Database administration: the essencial reference*. O'Reilly, Sebastopol, CA, USA, 1999.
- [19] Bimal Kumar. Layered architecture for mobile web based applications. *Asia-Pacific World Congress on Computer Science and Engineering, APWC on CSE 2014*, 03 2015.
- [20] Michael V. Mannino. *Database design, application development, and administration*. The McGraw-Hill, New York, USA, 2001.
- [21] Kendall Scott Martin Fowler. *UML Gota a gota*. S.A. ALHAMBRA MEXICANA, MÉXICO, 1999.
- [22] Yukihiro Matsumoto. *Ruby in a nutshell: a desktop quick reference*. O'Reilly, Sebastopol, CA, USA, 2002.

- [23] Kaufmann Michael Meier Andreas. *SQL and NoSQL Databases*. Springer Vieweg, Wiesbaden, Germany, 2019.
- [24] Nishant Neeraj. *Mastering Apache Cassandra. Second Edition*. Pacht Publishing Ltd, Birmingham B3 2PB, UK, 2015.
- [25] C. Orós Cabello, J. *Diseño de páginas Web con XHTML, JavaScript y CSS*. Alfaomega Grupo Editor S.A de C.V, México, 2006.
- [26] Roger .S Pressman. *Ingeniería de software. Un enfoque práctico*. McGraw-Hill Interamericana Editores, S.A. de C.V., México, D. F., 2010.
- [27] César Pérez. *MySQL para windows y linux*. Alfaomega Grupo Editor, S.A. de C.V., D.F, México, 2008.
- [28] José Ramírez-Faz, Luis Manuel Fernández-Ahumada, Elvira Fernández-Ahumada, and Rafael López-Luque. Monitoring of temperature in retail refrigerated cabinets applying iot over open-source hardware and software. *Sensors*, 20(3), 2020.
- [29] Catherine M. Ricardo. *Bases de datos*. McGraw-Hill Interamericana Editores, S.A. de C.V., México, D. F., 2009.
- [30] Valentin Kipyatkov Sergey Dmitriev and Eugene Belyaev. blog.jetbrains.com, 2000.
- [31] Rigoberto Solorio. *A WEB-BASED TEMPERATURE MONITORING SYSTEM FOR THE COLLEGE OF ARTS AND LETTERS*. PhD thesis, California State University - San Bernardino, 2015.
- [32] Ian Sommerville. *Ingeniería de Software*. Pearson Educación de México, S.A. de C.V., Naucalpan de Juárez, Estado de México, 2011.
- [33] R. Soriano Momparler. *HTML 4. Diseño y creación de páginas web*. Alfaomega Grupo Editor S.A de C.V, México, 2002.

Apéndice A

Documento ERS



ERS (SRS por sus siglas en inglés) Especificación de Requisitos de Software

Sistema Web de Monitoreo de Biorepositorios

Versión 1.0

Encargado: Julio César Álvarez Charqueño

Fecha: 6/01/23

Índice general

1. Introducción

- 1. 1 Propósito del documento.
- 1. 2 Ámbito del sistema.
- 1. 3 Definiciones, siglas y abreviaciones.
- 1. 4 Referencias.
- 1. 5 Vista global.

2. Descripción global

- 2. 1 Perspectiva del producto.
- 2. 2 Funciones del producto.
- 2. 3 Características del usuario.
- 2. 4 Restricciones.
- 2. 5 Suposiciones y dependencias.
- 2. 6 Requerimientos futuros.

3. Requerimientos específicos

- 3. 1 Interfaces externas.
- 3. 2 Requerimientos funcionales.
- 3. 3 Requerimientos no funcionales.
- 3. 4 Restricciones de diseño.
- 3. 5 Requerimientos de licencias.
- 3. 6 Otros requerimientos.

1. Introducción

En este documento se planteará y analizarán los requerimientos y especificaciones del proyecto Sistema Web de Monitoreo de Biorepositorios de manera global que se desarrolla para el Laboratorio de Genómica Viral y Humana.

1.1 Propósito del documento

Lo que se verá en este documento es una descripción de los distintos requerimientos involucrados en el Sistema Web de Monitoreo de Biorepositorios. También se tiene la finalidad de describir la funcionalidad en general que presentará el sistema al termino de su desarrollo.

1.2 Ámbito del sistema

Se han considerado los siguientes puntos:

- El sistema será incluido dentro del sistema MIDASmap.
- El sistema servirá para revisar y registrar las temperaturas de los sensores que presentan las áreas y biorepositorios.
- El sistema servirá para consultar históricos de temperaturas.
- El sistema servirá para crear estadísticas de las temperaturas registradas.
- El sistema servirá para crear gráficas de linea y caja usando los registros de temperaturas.
- El sistema podrá leer los reportes creados por los sensores de áreas y biorepositorios.
- El sistema permitirá al personal monitorear las temperaturas de áreas y biorepositorios sin la necesidad de hacerlo de forma manual en el laboratorio.
- El sistema permitirá modificaciones e implementaciones nuevas.
- Se consideran otros requerimientos como compatibilidad, seguridad, usabilidad, extensibilidad, sistemas operativos, restricciones de diseño y licencias.

1.3 Definiciones, siglas y abreviaciones

- LGVH – Laboratorio de Genómica Viral y Humana.
- SWMB – Sistema Web de Monitoreo de Biorepositorios.
- MIDASmap – Mexican Infectious Disease Analysis and Surveillance mapping application
- BD – Base de datos.
- IDE – Integrated Development Environment.
- W3C – World Wide Web Consortium.

1.4 Referencias

- IEEE-STD-830-1998: Especificaciones de los requisitos del software.
- Véase documento Bibliografía
- <https://www.mysql.com/about/legal/licensing/oem/>
- <https://www.gnu.org/licenses/>
- <https://www.apache.org/licenses/LICENSE-2.0>
- <https://docs.python.org/3/license.html>
- <https://docs.djangoproject.com/es/4.1/faq/general/how-is-django-licensed>

1.5 Visión general

Este documento tiene el objetivo de mostrar las especificaciones del sistema que involucran a los requerimientos funcionales dados por el cliente (el laboratorio) y no funcionales los cuales incluyen aspectos como la seguridad del sistema, su compatibilidad con algún hardware y software, la extensibilidad que permita el crecimiento del sistema, su usabilidad y su funcionalidad.

2. Descripción global

Esta sección pretende mostrar los factores que afectan al contexto de este sistema como la interacción con otros sistemas, la dependencia a otros sistemas y la disponibilidad del cliente.

2.1 Perspectiva del producto

Este sistema será un módulo que formará parte de un sistema mayor llamado MIDASmap y se encontrará en la sección de herramientas. Para que el sistema tenga un funcionamiento esperado el entorno donde este MIDASmap o donde será alojado deberán de poseer los requisitos dictados por SWMB. Los reportes generados por los sensores deberán de permitir el acceso al SWMB para su uso, además, deberán de estar en un formato compatible.

2.2 Funciones del producto

- Registro automático de temperaturas.
- Consulta de los registros de las temperaturas.
- Creación de gráficas de linea y caja de las temperaturas.
- Cálculo de estadísticas de las temperaturas.

2.3 Características del usuario

Encargado del laboratorio; usuario con nivel de educación superior (Doctorado) con la capacidad intermedia del uso de equipos de cómputo y con conocimiento de tecnologías informáticas.

2.4 Restricciones

- Uso de autenticaciones para evitar el acceso a usuarios no autorizados.
- No se cuenta con especificaciones de hardware y software.
- No se conoce a profundidad cómo funcionan los sensores y dónde almacenan sus reportes.
- El laboratorio se está reestructurando.

2.5 Suposiciones y dependencias

El servidor donde se aloje deberá contar con la seguridad para proteger los datos que maneje el sistema. El sistema operativo del servidor deberá tener compatibilidad con todas las especificaciones que pueda tener el sistema de

monitoreo.

2.6 Requerimientos futuros

Para la siguiente fase o versión de este proyecto se deberán realizar cambios en las interfaces actuales para adaptarse a las nuevas necesidades del laboratorio; implementar el módulo de mensajes de alarma cuando una temperatura este fuera del rango permitido; realizar respaldos de la base de datos; desarrollar un módulo de inteligencia artificial para la predicción de fallas en sensores y biorepositorios.

3. Requerimientos Específicos

En esta sección se revisarán a detalle los requerimientos del SWMB.

3.1 Interfaces externas

Se utilizará como base el diseño de interfaces proporcionado por el LGVH las cuales deberán ser compatibles con Google Chrome (108.0.5359.124+), Firefox (108.0+), Safari (14.0.2+) y con los sistemas operativos Microsoft Windows, Mac OS X y Linux Ubuntu.

Puntos a considerar para las interfaces:

- **Login**
 - En el centro de la pantalla se encontrarán dos elementos que el usuario deberá de llenar para poder ingresar al sistema.
 - Debajo; se encontrará una liga con la leyenda *I forgot my password* que permitirá al usuario crear una nueva contraseña.
 - Debajo; se encontrará un botón con el texto Login que validara el acceso al usuario.
- **Reset Password**
 - En el centro de la pantalla se encontrará un breve instructivo de qué hacer para restaurar la contraseña.
 - Debajo; se encontrará un elemento donde se ingresarán los datos solicitados en el instructivo.
 - Debajo; un botón con el texto *send email* que mandará la información solicitada.
- **Password reset sent**

- En el centro de la pantalla se mostrarán unas instrucciones con información respecto al correo enviado.
- **Password reset confirm**
 - En el centro de la pantalla se mostrará una instrucción a realizar.
 - Debajo; se encontrarán dos elementos que el usuario deberá de llenar para poder modificar su contraseña.
 - Debajo; se encontrará un botón con el texto *Reset password* para cambiar la contraseña.
- **Password reset complete**
 - En el centro de la pantalla se mostrará una instrucción a realizar.
 - Debajo; se encontrará una liga con el texto *Login* la que te envía a la interfaz de login.
- **Interfaz base**
 - En la esquina superior izquierda se encontrará un selector para seleccionar la opción que se desea mostrar.
 - Esquina superior derecha, se encontrarán tres ligas, con el texto *Administration* para la *administración* de las temperaturas, con *Help* para visualizar el manual de usuario y con el texto *Logout* para salir de la sesión.
- **Seleccionar la opción del selector Display Ambient Temperatures**
 - Se mostrará un croquis del laboratorio con las áreas marcadas en color rojo en el centro de la pantalla.
 - Se mostrarán figuras de color gris que tiene como texto el nombre del área y la temperatura que presentan, dichas figuras se asociarán a cada parte marcada del croquis.
 - Se mostrará del lado izquierdo del croquis un elemento que muestre el pronóstico del clima con su respectiva figura color gris encima del elemento, con el texto *Outdoor Ambient Temperature*.
 - Debajo del croquis se encontrará un selector y una tabla que muestra un histórico de las temperaturas ya registradas según la opción seleccionada del selector. El selector tendrá una opción para elegir un rango de fechas, esta opción mostrará del lado derecho del selector dos elementos para elegir fechas y un botón con el texto *Search* para realizar la búsqueda de temperaturas por fechas.
 - Si el usuario da clic en alguna figura gris, se mostrará, debajo del

croquis, una tabla con distintos lapsos de tiempo del cálculo de las estadísticas del área perteneciente a la figura. Debajo de la tabla creada se mostrarán dos elementos para elegir fecha y un botón a su lado derecho con el texto *Search* que tiene como intención calcular las estadísticas de un rango de fechas específicas. Encima de la tabla se mostrará un botón con el texto *Back to temperature log* que permitirá volver a la tabla de históricos, encima del botón un texto que indicará el área a la que pertenecen las estadísticas.

- **Seleccionar la opción del selector Display Biorepository Temperatures**
 - Se mostrará una estructura y una funcionalidad similar a la vista en *Display Ambient Temperatures* pero para los biorepositorios.
- **Seleccionar la opción del selector Display Graphic Summary**
 - Se mostrará el texto *Select temperature log to graph*.
 - Debajo; se mostrará una lista de tres columnas con todas las áreas y biorepositorios del laboratorio con la opción de que el usuario pueda marcar un elemento de interés.
 - Debajo; se mostrará el texto *Select temperature log to graph* y a su derecha el texto *Select date range*.
 - Debajo; se mostrarán un selector que permitirá al usuario elegir el tipo de gráfica a construir y a su derecha otro selector que le permitirá elegir un rango de fechas. A su derecha se mostrará un botón con el texto *Update* que al dar clic le permitirá crear la gráfica debajo de los elementos mencionados.

■ Administration

- Se mostrará una interfaz que permitirá registrar nuevas temperaturas, modificar y eliminar las temperaturas que se encuentran en la base de datos.

3.2 Requerimientos funcionales

Se muestran los requerimientos funcionales para el SWMB dictados por el LGVH. Se organizan en dos partes; primero por lo que debe hacer el sistema y el segundo organizado por tipo de usuario.

El sistema deberá:

FR1 Obtener la temperatura climática de Internet.

- FR2 Leer los reportes generados por los sensores para extraer las temperaturas y almacenarlas en la BD.
- FR3 Identificar temperaturas fuera del rango permitido y marcar con azul las bajas y con rojo las altas.

El personal del laboratorio podrá:

- FR1 Realizar consultas de los históricos de las temperaturas que se muestren en una tabla.
- FR2 Crear una tabla con las características estadísticas de la media, desviación estándar, máximo y mínimo de un área o biorepositorio de interés.
- FR3 Crear una gráfica lineal de las temperaturas de un área o biorepositorio a partir de una fecha específica.
- FR4 Crear una gráfica de cajas de un área o biorepositorio que muestre los valores cuartiles, mediana, máximos y mínimos de un periodo de tiempo de tres años a partir de una fecha específica.

3.3 Requerimientos no funcionales

Estos requerimientos no son dados por el cliente ya que son propiedades externas al sistema en si.

QR1 Disponibilidad

- Es de suma importancia el acceso a los datos en todo momento.

QR2 Seguridad

- El acceso al SWMB será exclusivo del LGVH por lo cuál es importante el uso de usuarios, contraseña, claves y la implementación de tecnologías que permitan la integridad de los datos y el bienestar del sistema.

QR3 Extensibilidad

- El SWMB es parte de un proyecto de mayor extensión por lo cuál este deberá permitir la implementación de nuevos desarrollos y módulos.

QR4 Mantenibilidad

- Contar con la facilidad de identificar y corregir defectos que puedan presentarse y la adaptación a entornos cambiantes.

QR5 Usabilidad

- El uso de manuales, capacitaciones, tutoriales y el sistema sea sencillo e intuitivo permitirán al usuario un mejor entendimiento de su manejo.

3.4 Restricciones de diseño

El sistema será desarrollado en un entorno local para posteriormente alojarse en un servidor.

- **Tecnologías**

- Frontend será desarrollado utilizando HTML, CSS y JavaScript.
- Backend se desarrollara usando Python y MySQL.
- Se utilizará el IDE PyCharm para su desarrollo.

- **Tipo de arquitectura**

- La arquitectura en capas permite el desarrollo independiente y asignación de tareas específicas en cada capa, esta se implementará de forma híbrida con la arquitectura de repositorios.
- La arquitectura de repositorio será implementada para la obtención de los reportes.

- **Hardware y software del equipo del cliente**

Debido a la reestructuración del LGVH no se cuenta con las especificaciones que tiene el servidor, es por ello que se sugieren algunos aspectos.

- **Software**

- El equipo deberá contar con las configuraciones necesarias para el uso de las tecnologías como Python, HTML, CSS, JavaScript y MySQL.
- El equipo necesitará la instalación del framework Django junto a otras instalaciones que sean requeridas por el sistema.

- **Hardware**

- El equipo deberá de contar con conexión a Internet.
- El equipo deberá de contar con medios de almacenamiento como discos duros.

3.5 Requerimientos de licencias

Para el desarrollo del SWMB serán necesarias licencias de programas y tecnologías como:

- MySQL – Tiene una licencia dual GPLv2/Comercial. Se deberá de usar bajo licencia GPLv2 ya que es de código abierto y libre uso.
- Apache-2.0 – Tiene una licencia Apache gratuita y software libre. Tiene compatibilidad con software basado en GPL.

- Python – Tiene una licencia PSF que lo vuelve software libre y es compatible con la licencia GPL.
- Django – Cuenta con una licencia BSD de 3 clausulas que le permite ser software libre.
- HTML y CSS – Son estándares creados por W3C.
- JavaScript – Es de uso gratuito ajustado al estándar ECMAScript.

3.6 Otros requerimientos

Estos requerimientos son aquellos que tienen mayor probabilidad de ser agregados o modificados a lo largo del desarrollo.

- Incluir un módulo de registro manual de temperaturas (independiente al que se realiza en el sistema de administración).
- El nuevo registro de temperaturas se actualice sin necesidad de refrescar la página o cambiar de opción en el selector de secciones.

Apéndice B

Manual del programador



Sistema Web de Monitoreo de Biorepositorios

Desarrollador: Julio César Álvarez Charqueño

Fecha: 6/01/23

Índice general

1. Introducción

2. Librerías

3. General

- 3. 1 Registro automático
- 3. 2 Pronóstico del climática
- 3. 3 Leer reportes
- 3. 4 Base del sistema
- 3. 5 Procesar Ajax
- 3. 6 Colores de celda

4. Sección Ambientes

- 4. 1 Sección Ambientes
- 4. 2 Datos para tabla Ambientes

5. Sección Biorepositorios

- 5. 1 Sección Biorepositorios
- 5. 2 Datos para tabla Biorepositorios

6. Estadísticas

- 6. 1 Datos para tabla Biorepositorios
- 6. 2 Calcular estadísticas
- 6. 3 Calcular estadística extra

7. Sección Gráficas

- 7. 1 Sección Gráficas
- 7. 2 Datos para Gráficas
- 7. 3 Obtener Gráfica

- 7. 4 Estructura de linea
- 7. 5 Datos
- 7. 6 Estructura de caja
- 7. 7 Datos de caja
- 7. 8 Obtener cuartiles.

1. Introducción

El presente manual tiene como finalidad explicar cada parte de código relacionado a las secciones principales del Sistema Web de Monitoreo de Biorepositorios con la finalidad de que el desarrollador pueda dar mantenimiento, mejoras o pueda adaptar nuevas funciones. Cada parte del código presentará una descripción general de su función y con una serie de puntos que explican partes del código, esas partes están marcadas con la siguiente señalítica:

$< - - [N$

2. Librerías

Para el desarrollo y funcionamiento del sistema se ha utilizado Conda para crear un entorno de desarrollo con la versión 3.8.5 de Python el cuál ya cuenta con librerías instaladas, por ello solo se instalaron los paquetes extra decorator 4.0.2 y Django 3.1.1,

3. General

Esta sección muestra las funciones y el código que se utiliza como base de las tres secciones que maneja la página y código que tiene otros fines no tienen relación directa con las funciones de alguna sección.

3.1 Registro automático

La función *RegisterTemperatures* hace el registro cada hora de todas las temperaturas. Se utiliza dentro del hilo *t*.

- 1. Variable global.
- 2. Calcula una cantidad de segundos.
- 3. Manda llamar a *GetWeather*.
- 4. Manda llamar a *GetDataReport*.
- 5. Duerme a la función el total de segundos calculados.
- 6. Crea la variable de hilo con la función.

```
def RegisterTemperatures():  
    global WEATHER <--[ 1  
    lapse = 60 #minutos  
    total = 60 * lapse <--[ 2  
    while True:
```

```

WEATHER = GetWeather() <--[ 3
GetDataReport() <--[ 4
time.sleep(total) <--[ 5

t = threading.Thread(target=RegisterTemperatures) <--[ 6
t.start()

```

3.2 Pronóstico del climática

La función *GetWeather* realiza una solicitud a una página del clima.

- 1. Solicita la petición.
- 2. Se extraen los datos del archivo JSON respuesta.
- 3. Regresa el objeto con datos de interés.

```

def GetWeather():
    response = requests.get(config('URLWEATHER')) <--[ 1
    if response:
        data = {key:value for key, value in
            response.json().items()} <--[ 2
        infoTemp = {
            'main': int(data['main']['temp'] - 273.15),
            'city': data['name'],
            'img': config('IMGWEATHER') +
            str(data['weather'][0]['icon']) + ".png",
            'text': data['weather'][0]['main'],
            'min': int(data['main']['temp_min'] - 273.15),
            'max': int(data['main']['temp_max'] - 273.15)
        }
        return infoTemp <--[ 3
    return None

```

3.3 Leer reportes

La función *GetDataReport* lee los reporte generados por los sensores.

- 1. Dirección de los reportes
- 2. Obtiene la lista de reportes del directorio.
- 3. Convierte el reporte de tipo Excel a un data frame.

- 4. Obtiene el último registro de temperatura.
- 5. Obtiene la fecha del registro.
- 6. Obtiene la temperatura del registro.

```
def GetDataReport():
    url = config('URLREPORTS') <--[ 1
    reports = os.listdir(url) <--[ 2
    for report in reports:
        df = pd.DataFrame(pd.read_excel(url + report)) <--[ 3
        lastRecord = df.iloc[-1:] <--[ 4
        date = lastRecord.iloc[0]['Unnamed: 1'].split(' ') <--[ 5
        temp = lastRecord.iloc[0]['Unnamed: 2'] <--[ 6
```

3.4 Base del sistema

La función *Base* renderiza una plantilla que se mantiene constante en pantalla y sirve como base para las secciones.

- 1. Renderiza el HTML

```
def Base(request):
    return render(request, 'base.html') <--[ 1
```

3.5 Procesar Ajax

La función *processAjaxSearch* se encarga de recibir y procesar los Ajax de los POST hechos por las búsquedas por fecha que hace el usuario.

- 1. Evalúa si petición es de tipo POST.
- 2. Obtiene los registros solicitados por medio del método de clase.
- 3. Asigna los registros a la variable global.
- 4. Asigna un String a la variable global.

```
def processAjaxSearch(request):
    global AJAXDATA, DATESTATISTIC
    if request.method == 'POST': <--[ 1
        data = OBJTEMP.Search_Date(request.POST['ini'],
                                   request.POST['end']) <--[ 2
        AJAXDATA = data <--[ 3
        DATESTATISTIC = request.POST['end'] + ' to ' +
        request.POST['ini'] <--[ 4
    return render(request, 'base.html')
```

3.6 Colores de celda

El código de abajo se encarga de elegir el color que debe tener una celda según el valor de la temperatura recibida. Este código es parte de la plantilla *tableBiorepositories.html* y *tableAmbients.html*.

- 1. Condición para evaluar si cambia a color rojo.
- 2. Condición para evaluar si cambia a color azul.
- 3. Condición que no le pone color.

```
{% if dates.IDAMB.CELL_BIO_LAB|add:"0" > 26 %} <--[ 1
    <td style="background:#de5a5a;">
        {{dates.IDAMB.CELL_BIO_LAB}}°C</td>
{% elif dates.IDAMB.CELL_BIO_LAB|add:"0" < 20 %} <--[ 2
    <td style="background:#7bc5f2;">
        {{dates.IDAMB.CELL_BIO_LAB}}°C</td>
{%else%} <--[ 3
    <td>{{dates.IDAMB.CELL_BIO_LAB}}°C</td>
{%endif%}
```

4. Sección Ambientes

Esta sección muestra el código que se utiliza para el manejo de todas las funcionalidades de ambientes.

4.1 Datos de ambiente

La función *getAmbiente* obtiene y pasa los datos a la plantilla *ambient.html* para mostrar las temperaturas actuales de las áreas y limitar las fechas en el selector de fechas.

- 1. Se crea el objeto de la clase Temperatures.
- 2. Llama a la función del objeto.
- 3. Llama a la función del objeto.
- 4. Se crea un objeto con el primer registro, con el último y con la información del clima.
- 5. Le pasa el objeto a la plantilla *ambient.html*.
- 6. Consulta el registro más reciente.
- 7. Consulta el último registro.

```

OBJTEMP = Temperatures.temperature_class() <--[ 1
def getAmbiente(request):
    first = OBJTEMP.First_Date() <--[ 2
    last = OBJTEMP.Last_Date() <--[ 3
    context = {'firstDate':first, 'lastDate': last,
               'weather': WEATHER}<--[ 4
    return render(request, 'ambient.html', context) <--[ 5

def First_Date(self):
    data = Temperature.objects.order_by('-DATE', '-TIME').first() <--[ 6
    return data

def Last_Date(self):
    data = Temperature.objects.order_by('-DATE', '-TIME').last() <--[ 7
    return data

```

4.2 Datos para tabla Ambientes

La función *TableAmbient* recibe un ajax con las opción elegida por el usuario para buscar los registros de temperaturas.

- 1. Crea un objeto con los registros asignados a la variable global.
- 2. Borra los registros de la variable global.
- 3. Le pasa el objeto a la plantilla *tableAmbients.html*.
- 4. Llama la función de la clase.
- 5. Le pasa el objeto a la plantilla *tableAmbients.html*.
- 6. Ordena por fecha los registros y devuelve el registro con la fecha más actual.
- 7. Se crea un diccionario con las opciones de los rangos de fechas.
- 8. Se asigna una fecha anterior a la actual según la opción recibida.
- 9. Obtiene la fecha del registro actual.
- 10. Obtiene los registros que se encuentran entre la fecha actual y la anterior.

```

def TableAmbient(request, option):
    global AJAXDATA

```

```

if AJAXDATA:
    context = {'temps': AJAXDATA} <--[ 1
    AJAXDATA = None <--[ 2
    return render(request, 'tableAmbients.html', context) <--[ 3
show = OBJTEMP.Get_Temps(option) <--[ 4
context = {'temps': show}
return render(request, 'tableAmbients.html', context) <--[ 5

def Get_Temps(self, option):
    current = Temperature.objects.order_by('-DATE', '-TIME').first()
<--[ 6
    date_range = { <--[ 7
        '1': current.DATE - datetime.timedelta(days=7),
        '2': current.DATE - relativedelta(months=2),
        '3': current.DATE - relativedelta(months=3),
        '4': current.DATE - relativedelta(months=6),
        '5': current.DATE - relativedelta(years=1)
    }
    old = str( date_range[option] ) <--[ 8
    current = str(current.DATE) <--[ 9
    data = Temperature.objects.filter(DATE__range=(old, current))
        .order_by('-DATE', '-TIME') <--[ 10
    return data

```

5. Sección Biorepositorios

Esta sección muestra el código que se utiliza para el manejo de todas las funcionalidades respecto a los biorepositorios.

5.1 Datos de biorepositorio

La función *getBiorepository* obtiene y pasa los datos a la plantilla *biorepository.html* para mostrar las temperaturas actuales de las áreas y limitar las fechas en el selector de fechas.

- 1. Se crea el objeto de la clase Temperatures.
- 2. Llama a la función del objeto.
- 3. Llama a la función del objeto.
- 4. Se crea un objeto con el primer registro, con el último y con la información del clima.

- 5. Le pasa el objeto a la plantilla *biorepository.html*.
- 6. Consulta el registro más reciente.
- 7. Consulta el último registro.

```
OBJTEMP = Temperatures.temperature_class() <--[ 1
def getBiorepository(request):
    first = OBJTEMP.First_Date() <--[ 2
    last = OBJTEMP.Last_Date() <--[ 3
    context = {'firstDate': first, 'lastDate': last,
               'weather': WEATHER}<--[ 4
    return render(request, 'biorepository.html', context) <--[ 5

def First_Date(self):
    data = Temperature.objects.order_by('-DATE', '-TIME').first() <--[ 6
    return data

def Last_Date(self):
    data = Temperature.objects.order_by('-DATE', '-TIME').last() <--[ 7
    return data
```

5.2 Datos para tabla Biorepositorios

La función *TableBiorepository* recibe un ajax con las opción elegida por el usuario para buscar los registros de temperaturas.

- 1. Crea un objeto con los registros asignados a la variable global.
- 2. Borra los registros de la variable global.
- 3. Le pasa el objeto a la plantilla *tableBiorepositories.html*.
- 4. Llama la función de la clase.
- 5. Le pasa el objeto a la plantilla *tableBiorepositories.html*.
- 6. Ordena por fecha los registros y devuelve el registro con la fecha más actual.
- 7. Se crea un diccionario con las opciones de los rangos de fechas.
- 8. Se asigna una fecha anterior a la actual según la opción recibida.
- 9. Obtiene la fecha del registro actual.

- 10. Obtiene los registros que se encuentran entre la fecha actual y la anterior.

```
def TableBiorepository(request, option):
    global AJAXDATA
    if AJAXDATA:
        context = {'temps': AJAXDATA} <--[ 1
        AJAXDATA = None <--[ 2
        return render(request, 'tableBiorepositories.html', context)<--[ 3
    show = OBJTEMP.Get_Temps(option) <--[ 4
    context = {'temps': show}
    return render(request, 'tableBiorepositories.html', context)<--[ 5

def Get_Temps(self, option):
    current = Temperature.objects.order_by('-DATE', '-TIME').first()<--[ 6
    date_range = { <--[ 7
        '1': current.DATE - datetime.timedelta(days=7),
        '2': current.DATE - relativedelta(months=2),
        '3': current.DATE - relativedelta(months=3),
        '4': current.DATE - relativedelta(months=6),
        '5': current.DATE - relativedelta(years=1)
    }
    old = str( date_range[option] ) <--[ 8
    current = str(current.DATE) <--[ 9
    data = Temperature.objects.filter(DATE__range=(old, current))
        .order_by('-DATE', '-TIME') <--[ 10
    return data
```

6. Estadísticas

Esta sección muestra el código que se utiliza para calcular las estadísticas de los registros de las secciones Ambientes y Biorepositorios.

6.1 Datos para tabla Estadísticas

La función *TableStatistics* recibe un ajax con las opción elegida por el usuario para crear las estadísticas.

- 1. Llama a la función de la clase.
- 2. Le pasa el objeto a la plantilla *tableStatistics.html*.

- 3. Divide el String de la variable *option* para obtener los parámetros.
- 4. Se asigna el nombre del área o biorepositorio.
- 5. Se asigna la sección con la que se trabaja.
- 6. Crea una lista con el resultado de la llamada a la función de clase, el nombre y la sección.
- 7. Le pasa el objeto a la plantilla *tableStatistics.html*.

```
def TableStatistics(request, option):
    global AJAXDATA, STATEMPS, DATESTATISTIC
    first = OBJTEMP.First_Date()
    last = OBJTEMP.Last_Date()
    if AJAXDATA:
        temps = OBJTEMP.Get_One_Statistic(
            AJAXDATA, STATEMPS[0], STATEMPS[1], STATEMPS[2],
            DATESTATISTIC)<--[ 1
        context = {'statistics': temps, 'firstDate': first,
            'lastDate': last}
        AJAXDATA = None
        STATEMPS = None
        DATESTATISTIC = None
        return render(request, 'tableStatistics.html', context)<--[ 2
    param = option.split(':') <--[ 3
    name = param[0] <--[ 4
    id = param[1] <--[ 5
    STATEMPS = [ OBJTEMP.Get_Statistics(name, int(id)), name, id ]<--[ 6
    context = {'statistics': STATEMPS[0], 'firstDate':first,
        'lastDate': last}
    return render(request, 'tableStatistics.html', context)<--[ 7
```

6.2 Calcular Estadísticas

La función *Get Statistics* calcula las estadísticas desviación estándar, media, máximo y mínimo de cada rango de temperatura.

- 1. Asigna el modelo de base de datos con el que se va a trabajar.
- 2. Cero para realizar los cálculos para las temperaturas climáticas.
- 3. Asigna una fecha anterior a la actual.

- 4. Obtiene los registros que se encuentran entre la fecha actual y la anterior.
- 5. Crea un objeto con los cálculos estadísticos.
- 6. Agrega el objeto a una lista.
- 7. Diccionario con los identificadores de sección.
- 8. Obtiene solo los registros de la sección seleccionada.

```
def Get_Statistics(self, option, id):
    current = self.First_Date().DATE
    model = Ambient <--[ 1
    statistics = []
    date_range = {
        1: current - datetime.timedelta(days=7),
        2: current - datetime.timedelta(days=15),
        3: current - relativedelta(months=1),
        4: current - relativedelta(months=2),
        5: current - relativedelta(months=3),
        6: current - relativedelta(months=6),
        7: current - relativedelta(year=1)
    }
    dictionary_name_date = {
        1: 'Last week', 2: 'Last fortnight', 3: 'Last month',
        4: 'Last bimester', 5: 'Last trimester', 6: 'Last semester',
        7: 'Last year'}
    if id == 0: <--[ 2
        for i in range(1, 8):
            old = date_range[i] <--[ 3
            temps = Temperature.objects.filter(DATE__range=(str(old),
                str(current))).order_by('-DATE',
                '-TIME').values_list(option)<--[ 4
            a = Statistics(dictionary_name_date[i], np.amax(temps),
                np.amin(temps), round(np.mean(temps)),
                round(np.std(temps))) <--[ 5
            statistics.append(a)<--[ 6
        return statistics
    id_model = {1: 'IDAMB', 2: 'IDBIO'}<--[ 7
    if id==2:
```

```

        model = Biorepositorie
    for i in range(1,8):
        old = date_range[i]
        data = Temperature.objects.filter(
            DATE__range=(str(old), str(current))).values_list(
                id_model[id] ).order_by(
                    '-DATE', '-TIME')
        temps = model.objects.filter(id__in=data)
        .values_list(option)<--[ 8
        a = Statistics(dictionary_name_date[i], np.amax(temps),
            np.amin(temps), round(np.mean(temps)),
            round(np.std(temps)))
        statistics.append(a)
    return statistics

```

6.3 Calcular Estadística extra

La función *Get One Statistic* calcula una estadísticas que se agrega a las ya calculadas.

- 1. Obtiene los registros del área *OUTDOOR*.
- 2. Se agrega el nuevo cálculo a las estadísticas.

```

def Get_One_Statistic(self, data, others, name, id, date):
    model = Ambient
    if name == 'OUTDOOR':
        temps = data.values_list('OUTDOOR')<--[ 1
        s = Statistics(date, np.amax(temps), np.amin(temps),
            round(np.mean(temps)),round(np.std(temps)))
        others.append(s)<--[ 2
        return others
    if id == '2':
        model = Biorepositorie
        temps = model.objects.filter(id__in=data).values_list(name)
        s = Statistics(date, np.amax(temps), np.amin(temps),
            round(np.mean(temps)), round(np.std(temps)))
        others.append(s)
    return others

```

7. Sección Gráficas

Esta sección muestra el código que se utiliza para obtener la información que

presentarán las gráficas.

7.1 Sección para Gráficas

La función *getGraphics* obtiene y pasa los datos a la plantilla *graphic.html* para mostrar los elementos de interfaz y limitar las fechas en el selector de fechas.

- 1. Le pasa el objeto a la plantilla *graphic.html*.

```
def getGraphics(request):
    first = OBJTEMP.First_Date()
    last = OBJTEMP.Last_Date()
    context = {'firstDate': first, 'lastDate': last}
    return render(request, 'graphic.html', context) <--[ 1
```

7.2 Datos para Gráficas

La función *TableGraphics* recibe un ajax con las opción elegida por el usuario para construir la gráfica.

- 1. Extrae la información recibida del POST.
- 2. Asigna el tipo de gráfica.
- 3. Asigna un rango de fechas.
- 4. Asigna una opción de fecha.
- 5. Divide la sección del nombre de un área o biorepositorio.
- 6. Asigna el nombre.
- 7. Asigna la sección.
- 8. Llama a la función de clase.
- 9. Le pasa el objeto a la plantilla *graphic.html*.

```
def TableGraphics(request):
    global DATAGRAPHSICS
    if request.method == 'POST':
        post = request.POST['graph']
        post = post.split('/') <--[ 1
        typeGraphic = post[1] <--[ 2
        if len(post) == 4:
```

```

        dateRange = post[2] + ',' + post[3] <--[ 3
    else:
        dateRange = post[2] <--[ 4
    option = post[0].split(':') <--[ 5
    value = option[0] <--[ 6
    model = option[1] <--[ 7
    DATAGRAPHICS = OBJTEMP.getDataGraphic(model, value,
        dateRange, typeGraphic) <--[ 8
    return render(request, 'tableGraphics.html',
        DATAGRAPHICS) <--[ 9
return render(request, 'tableGraphics.html', DATAGRAPHICS)

```

7.3 Obtener Gráfica

La función *getDataGraphic* brinda los datos del tipo de gráfica seleccionada.

- 1. Obtiene la sección.
- 2. Asigna la la fecha final.
- 3. El *except* entra a una parte que permite utilizar las fechas elegidas por el usuario.
- 4. Separa el String de fechas.
- 5. Convierte el String a Date.
- 6. Llama a la función de clase.
- 7. Regresa un objeto con la información de la gráfica.

```

def getDataGraphic(self, typeModel, name, dateRange, typeGraphic):
    options = {'1':Ambient, '2': Biorepositorie, '0':Temperature}
    model = options[typeModel] <--[ 1
    try:
        current = self.First_Date().DATE
        date_range = {
            '1': current - datetime.timedelta(days=7),
            '2': current - relativedelta(months=1),
        }
        last = date_range[dateRange] <--[ 2
    except: <--[ 3
        dates = dateRange.split(',') <---[ 4

```

```

        current = datetime.datetime.strptime(dates[0],
        '%Y-%m-%d').date() <---[ 5
        last = datetime.datetime.strptime(dates[1],
        '%Y-%m-%d').date()
    if typeGraphic == 'line':
        dicc = self.setStructureDataLine(current, last,
        model, name) <---[6
        return {'data': dicc['data'], 'labels': dicc['labels'],
        'dates': dicc['dates'], 'option': name} <---[ 7
    else:
        data = self.setStructureDataBox(current, last,
        model, name)
        return {'data': data, 'labels': ['labels'],
        'dates': ['dates'], 'option': name}

```

7.4 Estructura de linea

La función *setStructureDataLine* estructura los datos para adaptarlos a la forma de una gráfica de linea.

- 1. Obtiene la diferencia en meses entre las dos fechas.
- 2. Llama a la función propia de la clase.
- 3. Se resta un mes a la fecha.
- 4. Coloca en el primer día la fecha.
- 5. Le suma un mesa a la fecha.
- 6. Le resta un día a la fecha.
- 7. Crea un diccionario.
- 8. Asigna un elemento al diccionario.

```

def setStructureDataLine(self, current, last, model, name):
    labels = []
    temps = []
    dates = []
    delta = relativedelta(current, last)
    res_months = delta.months + (delta.years * 12) <-- 1
    self.getData(current, current.replace(day=1), labels, temps,

```

```

        dates, model, name) <--[ 2
if (res_months != 0):
    for i in range(1, res_months):
        d1 = current - relativedelta(months=i ) <--[ 3
        ini = d1.replace(day=1) <--[ 4
        d2 = ini + relativedelta(months=1) <--[ 5
        end = d2 - datetime.timedelta(days=1) <--[ 6
        self.getData(end, ini, labels, temps, dates,
            model, name)
    self.getData(last, last.replace(day=1), labels, temps,
        dates, model, name)
dicc = dict() <--[ 7
dicc['data'] = temps <--[ 8
dicc['labels'] = labels
dicc['dates'] = dates
return dicc

```

7.5 Datos

La función *getData* realiza las búsquedas de los registros y las etiquetas necesarias para la gráfica.

- 1. Obtiene las fechas de cada registro.
- 2. Lo agrega a una lista.
- 3. Crea una lista con las temperaturas de cada fecha y la agrega a la lista.
- 4. Crea una lista con el nombre de los meses de la consulta.

```

def getData(self, currentDate, lastDate, labels, temps,
    dates, model, name):
    first = self.Search_Date(str(currentDate), lastDate)
    concatenate = ["{} {}".format(b_, a_) for a_, b_ in
        zip(list(first.values_list('TIME', flat=True)),
            list(first.values_list('DATE', flat=True)))] <--[ 1
    labels.append(concatenate) <--[ 2
    temps.append(list(model.objects.filter(id__in=first).
        values_list(name, flat=True))) <--[ 3
    dates.append(currentDate.strftime("%B")) <--[ 4

```

7.6 Estructura de caja

La función *setStructureDataBox* estructura los datos para adaptarlos a la forma de una gráfica de caja.

- 1. Llama a la función propia de la clase.
- 2. Guarda en una lista la información de *self.getDataBox*.

```
def setStructureDataBox(self, current, last, model, name):
    delta = relativedelta(current, last)
    res_months = delta.months + (delta.years * 12)
    allData = []
    infoBox = self.getDataBox(current, current.replace(day=1),
                              model, name) <--[ 1
    allData.append(infoBox) <--[ 2
    if (res_months != 0):
        for i in range(1, res_months):
            d1 = current - relativedelta(months=i)
            ini = d1.replace(day=1)
            d2 = ini + relativedelta(months=1)
            end = d2 - datetime.timedelta(days=1)
            infoBox = self.getDataBox(end, ini, model, name)
            allData.append(infoBox)
    infoBox = self.getDataBox(last, last.replace(day=1),
                              model, name)
    allData.append(infoBox)
    return allData
```

7.7 Datos de caja

La función *getDataBox* busca las fechas uno, dos y tres años atras de cada fecha dentro del rango.

- 1. Crea una lista de fechas.
- 2. Ordena la lista.
- 3. Obtiene el registro de la fecha un año atras.
- 4. Llama a la función propia de la clase.
- 5. Guarda en una lista la información de *self.getQuartiles*


```

def getDataBox(self, currentDate, lastDate, model, name):
    data = self.Search_Date(str(currentDate), lastDate)
    data = list(data.values_list('DATE', flat=True)) <--[ 1
    newData = sorted(set(data), key=data.index) <--[ 2
    allBoxInfo = []
    for i in newData:
        firstDate = Temperature.objects.filter(
            DATE=i - relativedelta(years=1) ) <--[ 3
        secondDate = Temperature.objects.filter(
            DATE=i - relativedelta(years=2) )
        thirdDate = Temperature.objects.filter(
            DATE=i - relativedelta(years=3) )
        boxInfo = self.getQuartiles(
            list(model.objects.filter(id__in=firstDate).
                values_list(name, flat=True)) +
            list(model.objects.filter(id__in=secondDate).
                values_list(name, flat=True)) +
            list(model.objects.filter(id__in=thirdDate).
                values_list(name, flat=True)),
            i.strftime("%Y-%m-%d")
        ) <--[ 4
        allBoxInfo.append(boxInfo) <--[ 5
    return allBoxInfo

```

7.8 Obtener cuartiles

La función *getQuartiles* calcula los cuartiles, mediana, maximos y minimos de cada fecha dentro del rango de fechas

- 1. Convierte la data a un arreglo de Numpy.
- 2. Obtiene el valor mínimo.
- 3. Obtiene el primer cuartil (q1).
- 4. Calcula la mediana.
- 5. Obtiene el tercer cuartil (q3).
- 6. Obtiene el valor máximo.
- 7. Crea una lista con los valores obtenidos.

```

def getQuartiles(self, data, date):
    lista = np.array(data) <--[ 1
    min = np.amin(lista) <--[ 2
    q1 = np.percentile(lista, 75) <--[ 3
    mediana = np.percentile(lista, 50) <--[ 4
    q3 = np.percentile(lista, 25) <--[ 5
    max = np.amax(lista) <--[ 6
    boxInfo = [date, min, q1, mediana, q3, max] <--[ 7
    return boxInfo

```