Universidad Tecnológica de La Habana

"José Antonio Echeverría"

Trabajo de Curso

Tema 2:

"Gestión y Almacenamiento de Viviendas Afectadas

por eventos meteorológicos"

GAVA

Autor: Brian Serrano Alfonso

Tutor: Fermín Rivas Sotomayor

Diseño y Programación Orientada a Objetos

Año académico: 1ro

Carrera: Ing. Informática

RESUMEN

Un evento meteorológico deja a su paso un gran numero de afectaciones al fondo habitacional y personas damnificadas. Ante esto, equipos recorren las zonas afectadas recogiendo las evaluaciones hechas, en una ficha técnica. Generalmente, estos planes no se cumplen correctamente. Al final, terminan quedando muy pocas fichas completas, la mayoría incorrectas y con el personal insuficiente para realizar el levantamiento de daños. Esto dificulta el avance en el trabajo de los Consejos de Administración y el Ministerio de la Construcción. La mayor parte de estas problemáticas pueden ser resueltas mediante un sistema digital de introducción, almacenamiento y gestión de fichas técnicas de la vivienda, que también se encargue de los cálculos pertinentes para obtener la cubicación real necesaria para que la vivienda sea reconstruida de forma óptima. Es en este punto cuando la creación del sistema de Gestión y Almacenamiento de Viviendas Afectadas (GAVA) cobra sentido y entra en acción.

Palabras Clave: GAVA, Ficha Técnica, Afectación, Vivienda, Construcción.

ÍNDICE

INTRODUCCIÓN

El Ministerio de la Construcción (MICONS) ha encargado el desarrollo de un sistema informático capaz de digitalizar la información sobre las afectaciones al fondo habitacional como consecuencia de los eventos meteorológicos Para cumplir con la problemática propuesta se decidió el desarrollo del sistema informático GAVA.

El objetivo principal de dicho sistema sería digitalizar la información de las viviendas afectadas y los daños provocados por un evento meteorológico, facilitando el almacenamiento seguro de dichos datos. Además de permitir automatizar el cálculo de la cuantificación de daños y evitar el error humano.

A corto plazo se desea un almacenamiento digital eficiente de la información, posibilitando visibilizarla y administrarla en cualquier momento y de forma organizada. A mediano plazo se aspira a mejorar la organización del sistema de almacenamiento de afectaciones nacionales. A largo plazo se busca una mejora en la capacidad de gestión por parte del MICONS, las entidades administrativas territoriales y empresas responsables, conllevando un desarrollo exponencial de la reacción de dichas entidades ante fenómenos meteorológicos para disminuir el número de afectaciones a las viviendas y sus habitantes. Así como un mejor empleo de los fondos gubernamentales respecto a la gestión de los materiales de la construcción destinados a la reparación del fondo habitacional afectado.

La presente documentación trata el predesarrollo, desarrollo y posdesarrollo del sistema informático GAVA. En este trabajo se presenta un análisis detallado del sistema, el plan de trabajo aplicado en su realización, las herramientas y componentes necesarios para el correcto funcionamiento de la capa lógica del programa, así como las consideraciones tomadas en cuenta en busca de la comodidad y facilidad de uso por parte del usuario.

CAPÍTULO 1: Predesarrollo

1.1 Descripción del problema

Se propone que este sistema deba permitir almacenar de forma digital los datos obtenidos por los equipos de expertos en la ficha técnica de daños ocasionados (FT), con respecto a la información de la vivienda, el núcleo familiar que la conforma y las afectaciones provocadas al inmueble y el mobiliario.

Al crear la FT digital de la vivienda afectada, se debe realizar la cubicación y valoración de los recursos materiales necesarios, que se suministrarán al damnificado para la reparación del inmueble afectado. Para esto se debe contar con una lista de materiales de la construcción con su respectivo precio por unidad.

Tanto la FT como sus respectivas cubicaciones por material deben almacenarse para su posterior entrega a los organismos correspondientes. Para facilitar el proceso de búsqueda y administración de las FT, afectaciones y materiales. Cada uno debe contar con un código único que facilite su identificación.

El sistema debe permitir el almacenamiento y administración de la información recogida y tener la facilidad de que el usuario reciba reportes con respecto a los datos almacenados.

1.2 Consideraciones

Para la realización del proyecto se tuvo en cuenta la Programación Orientada a Objetos (POO), tomando como lenguaje de programación a Java, en el entorno de desarrollo Eclipse en su versión 4.3.0 con el nivel de compilación 1.6. Se empleó la biblioteca Java Swing para el desarrollo de la interfaz gráfica, la api JCalendar para la interfaz de la fecha de levantamiento y el JUnit 4 para la realización de pruebas de funcionamiento. Para la creación del diagrama de clases UML se usó el software Enterprise Architect 8.0. Para el diseño de imágenes como el icono de GAVA, el logotipo del MICONS y los vectores

pertenecientes a cada botón de selección de pantalla, se empleó la plataforma Medibang Paint Pro Studio.

Para crear una aplicación fiel a la realidad, fue necesaria la realización de una investigación respecto a las FT, las Oficinas de Trámites (OT), dictámenes de vivienda, materiales de la construcción, entre otros. Tras dicha investigación, se logró moldear el sistema en función del realismo y la practicidad de los profesionales encargados de la gestión de las fichas técnicas y cubicaciones.

De esta forma, realizando una investigación en documentos oficiales como la Gaceta Oficial y la Constitución de la República, así como un trabajo de tesis avalado con una temática similar al problema propuesto, se realizaron numerosas consideraciones al respecto, como es el caso de:

- ✓ La fecha de levantamiento. Como la gestión de las fichas se centra en la reconstrucción de las viviendas afectadas de forma rápida y eficaz, no tiene sentido la introducción de viviendas cuyo levantamiento fuese realizado hace más de diez años, o en algún momento futuro.
- ✓ La documentación legal, y las tipologías habitacional y constructiva, presentan opciones de selección provenientes de legislaciones y tipos impuestos por el MICONS o la Constitución de la República. Igualmente, en previsión de cambios futuros posibles, está permitido configurar dichas listas.
- ✓ Uno de los requisitos para que una vivienda sea considerada digna es que su área sea mayor de $25m^2$ [1], por tanto, se decidió que las dimensiones de la vivienda no podían ser menor de 5 metros o superar los 150 metros, con opción configurable.
- ✓ Podemos considerar que las afectaciones que puede sufrir una vivienda se dividen en dos categorías: de techo o de pared, y a su vez, estas pueden ser parciales o totales [2].
- ✓ Una vivienda se considera afectada cuando presenta al menos una afectación, si una vivienda no presenta ningún tipo de afectación, no se podrá crear su FT.
- ✓ El jefe de núcleo debe ser una persona mayor de edad. Una persona es considerada mayor de edad cuando supera los 18 años de edad, con respecto a trámites nacionales [1].

- ✓ En el caso del núcleo familiar, un menor de edad no debería vivir solo en una vivienda, por lo que será sumado al total de habitantes, su tutor legal, y será considerado este, como el jefe de núcleo.
- ✓ Se extrajeron de la Resolución 41 de la Gaceta Oficial los principales materiales y sus datos técnicos para su futura implementación en el sistema [3].

1.2.1 Validaciones

Al realizar el análisis de la problemática, se llega a la necesidad de aplicar posibles validaciones para la entrada de datos, entre las que podríamos encontrar:

- ✓ El número del carné de identidad del jefe de núcleo: Debe ser validado antes de ser obtenido por el sistema, el carné debe tener por obligación 11 caracteres numéricos. A través de la comprobación de sus primeros 7 caracteres, relacionados con la fecha de nacimiento (1 y 2 son el año, 3 y 4 son el mes, 5 y 6 son el día, y 7 es el siglo). En este caso comprobaremos si los valores del día, mes, año y siglo son posibles y viables. También se tendrá en cuenta que el nuevo carné ingresado no haya sido introducido previamente en otra ficha técnica, o que sea de un menor de edad.
- ✓ Para la documentación legal y tipologías habitacional y constructiva, el usuario solo puede seleccionar aquellas opciones ya preestablecidas, aunque se podrán agregar o eliminar opciones, preparando al sistema para futuras situaciones.
- ✓ Algo similar sucede con el material de las afectaciones. Solo pueden ser seleccionados los respectivos materiales de techo o pared ya existentes, aunque se podrá agregar o eliminar según sea necesario.
- ✓ Solo podrá ser introducida una fecha de levantamiento que se halle en el rango entre el día de hoy y hace exactamente diez años.
- ✓ Las dimensiones de la vivienda deben ser entre 5 y 150 metros, con posibilidad de configurarse.
- ✓ Las dimensiones de una afectación deben ser mayores que 0.
- ✓ Las dimensiones de una afectación de pared deben ser menores que el largo máximo posible de una pared, que el sistema tenga contemplado.

- ✓ Las dimensiones de una afectación de techo total serán iguales que las dimensiones de la vivienda.
- ✓ Las dimensiones de una afectación de techo parcial deben ser menores que las dimensiones de la vivienda.

La mayor parte de estas validaciones, como límites de caracteres o elementos que funcionan como opción en una lista-categorizador podrán ser modificados.

1.2.2 Usuarios

El desarrollo de GAVA tiene como fin fundamental ayudar a los individuos encargados de almacenar la información de la FT. Los cuáles serán los usuarios principales de este sistema informático.

El sistema contempla como usuarios objetivo a aquellos profesionales dedicados a ingresar u obtener los datos almacenados de las FT, con el conocimiento necesario para realizarlo correctamente, pertenecientes al MICONS, el Consejo de Administración de su respectivo municipio, las OT o la Defensa Civil.

GAVA no necesita ni debe tener implementado un sistema de seguridad de Inicio de Sesión, ni los usuarios que acceden deberían estar divididos en jerarquías de restricciones.

1.3 Plan de trabajo

En la próxima tabla se presenta el plan de trabajo a seguir para la realización de GAVA, dicho plan puede verse modificado según las necesidades durante el proceso de desarrollo. Como el sistema solo tiene un desarrollador, se omitirán las columnas de Responsable y Probador. Se toma como fecha de inicio el 15 de octubre de 2022, y a partir de ese punto se realizará un conteo por semanas.

No.	Tarea	Fecha
1	Estudiar el enunciado y Tarjetas CRC	Semana 1
2	Crear resumen de funcionalidades	Semana 1
3	Prototipo de diagrama de clases	Semana 1
4	Crear las clases en Java	Semana 2
5	Programar funcionalidades básicas	Semana 2
6	Prototipo de interfaz por consola de prueba	Semana 2-3-4
7	Prototipo finalizado de lógica del sistema con validaciones	Semana 3-4
8	Integrar lógica e interfaz por consola	Semana 4-5
9	Probar el programa por consola	Semana 5
10	Arreglar errores de lógica	Semana 6
11	Arte conceptual de la interfaz gráfica	Semana 6
12	Iniciar la interfaz gráfica	Semana 6-7
13	Pantallas de Inicio y las dos primeras de formulario Nueva Ficha	Semana 7
14	Validaciones de las dos primeras pantallas de formulario Nueva Ficha	Semana 7-8
15	Dos últimas pantallas de formulario Nueva Ficha y sus validaciones	Semana 8
16	Pantalla de Lista de Viviendas y sus funcionalidades	Semana 9-10
17	Pantallas de Reportes y Ajustes	Semana 10
18	Diseño de las imágenes a implementar y mejora de la UI y UX	Semana 11
19	Pruebas finales y corrección de errores	Semana 11-12
20	Terminar el programa	Semana 12

La anterior tabla constituye el plan del primer corte del proyecto. Debido a que la primera parte del plan no se cumplió correctamente, es necesario realizar un nuevo plan de trabajo acorde a lo ya hecho y lo que aún falta por desarrollar o implementar.

Teniendo en cuenta que ya estar'ia conformado el prototipo de la interfaz gráfica, y también creada, implementada y probada la lógica del sistema; se iniciará el plan a partir de este punto. Se tomará como nueva fecha de inicio el 16 de noviembre de 2022, realizando el mismo conteo por semana. La próxima tabla ejemplifica el nuevo plan:

No.	Tarea	Fecha
1	Implementar la pantalla de Inicio	Semana 1
2	Implementar las cuatro pantallas del formulario Nueva Ficha	Semana 2-3
3	Validar por interfaz las entradas de Nueva Ficha	Semana 4-5
4	Integrar la lógica en Nueva Ficha	Semana 5
5	Implementar la pantalla Lista de Fichas	Semana 6
6	Realizar el tratamiento de excepciones en Nueva Ficha	Semana 6
7	Diseñar e implementar las pantallas de Reportes y Ajustes	Semana 7-8
8	Diseñar e implementar las imágenes e iconos	Semana 8
9	Mejora de la UI y UX	Semana 8
10	Pruebas finales y correcciones	Semana 9
11	Terminar el programa	Semana 9

Se considera que, a partir de los avances realizados respecto al primer plan de trabajo, y el correcto cumplimiento del nuevo plan; se ha logrado cumplir con las metas propuestas y tener como resultado un sistema consistente y óptimo para su correcto funcionamiento y objetivo.

1.4 Tarjetas CRC

Las tarjetas CRC (siglas de Clase-Responsabilidad-Colaboración) constituyen una herramienta usada como metodología para el diseño de POO. Cada tarjeta representa una clase que compone al sistema. Cada tarjeta está constituida por tres zonas: el nombre de la clase, las responsabilidades (objetivos) de dicha clase y los colaboradores, es decir, aquellas otras clases que ayudan a conseguir cumplir con sus responsabilidades.

El empleo de esta herramienta posibilita representar correctamente un sistema orientado a objetos. Las tarjetas CRC son un puente de comunicación entre integrantes de un equipo, en el caso de ser un grupo de varias personas. Además, permite ver las clases no solo como repositorios de datos, sino también como entidades individuales con comportamiento propio.

ConsejoAdmin	Colaboradores
✓ Creación de fichas técnicas ✓ Creación y almacenamiento de materiales de la construcción ✓ Almacenamiento y gestión global de plantillas. ✓ Mostrar reportes.	✓ Material ✓ Plantilla ✓ Ficha

Cubicacion	Colaboradores
✓ Calcular la cantidad de material requerido y el costo final.	✓ Material

Plantilla	Colaboradores
✓ Almacenar una ficha ✓ Almacenar las cubicaciones de dicha ficha	✓ Ficha ✓ Cubicacion

Ficha	Colaboradores
✓ Manipulación de los datos de una ficha ✓ Creación y almacenamiento de un dictamen, núcleo familiar, afectaciones al inmueble y elementos afectados.	✓ Dictamen ✓ NucleoFamiliar ✓ Afectacion ✓ ElementosAfectados

NucleoFamiliar	Colaboradores
✓ Ingresar, almacenar y gestionar el total de individuos que viven en una vivienda, así como cuántos de ellos son ancianos, niños o embarazadas. ✓ Insertar y almacenar el número de carné de identidad del jefe de núcleo.	

ElementoAfectado	Colaboradores
✓ Almacenar los datos de un elemento afectado	

Dictamen	Colaboradores
✓ Insertar, almacenar y gestionar los datos estructurales, físicos y legales de la vivienda. ✓ Obtener el área de la vivienda.	

Afectacion	Colaboradores
✓ Ingresar, almacenar y gestionar los datos de una afectación al inmueble. ✓ Almacenar la cubicación correspondiente a la afectación.	✓ Cubicacion

AfectTecho	Colaboradores

✓ Clasificar una afectación según la superficie, en este caso el techo.	

AfectPared	Colaboradores
✓ Clasificar una afectación según la superficie, en este caso una pared.	

Material	Colaboradores
✓ Almacenar los datos de un material	

CAPÍTULO 2: Desarrollo

2.1 Clases e interfaces

En la POO, una clase es una plantilla para crear objetos, que va a definir las características y comportamientos de dichos objetos. Una clase está constituida por atributos, que son las características del objeto a crear; y los métodos, que son las operaciones que dicho objeto podrá emplear. [4]

Una interfaz constituye una recopilación de operaciones que define un conjunto uniforme de comportamientos. Las interfaces son similares a una clase, excepto que una clase puede tener una instancia de su tipo, y una interfaz debe tener, como mínimo, una clase para ser implementada. [4]

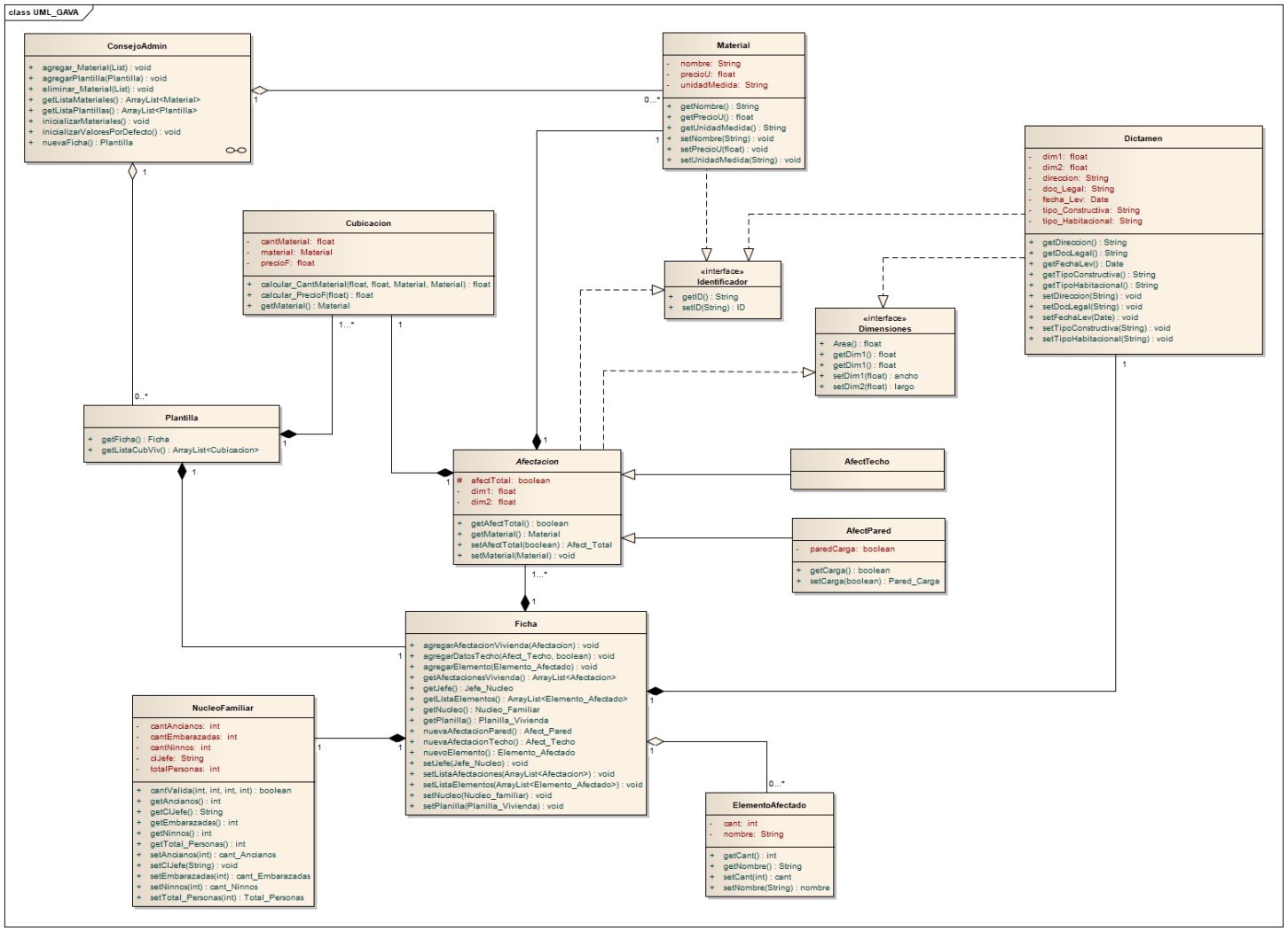
2.1.1 Diagrama de clases UML

El Lenguaje Unificado de Modelado (UML, por sus siglas en inglés) ayuda a modelar sistemas informáticos de diversas formas. El UML se estableció como un modelo estandarizado para describir un enfoque de POO. Uno de los tipos más populares y útiles en el UML es el diagrama de clases (DC). [5]

Un DC es un tipo de diagrama de estructura que describe los componentes que deben estar presentes en el sistema que se está modelando. Los diversos componentes de un DC representan las clases que se programan y la interacción entre ellas.

La representación de una clase en el diagrama consiste en un rectángulo de tres filas. La fila superior muestra el nombre de la clase, la central contiene sus atributos y la inferior expresa los métodos que puede emplear. De esta forma, en el diagrama las clases (e interfaces) se agrupan para mostrar la relación que existe entre cada una de ellas.

A continuación, se presenta el diagrama de clases del sistema GAVA:



En el sistema GAVA se emplean varios tipos de relaciones o interacciones entre las clases, a continuación, se presentará cada tipo empleado, sus características y razón de uso:

- Herencia: Se emplea cuando es necesario que una subclase o clase derivada reciba las funcionalidades y atributos de una superclase o clase principal. En GAVA se emplea en el caso de las afectaciones de techo (*AfectTecho*) y de pared (*AfectPared*), que heredarían de *Afectacion*.
- Agregación: Se emplea cuando una clase es contenida en una clase contenedora (CC), y la destrucción de la contenedora no conlleva a la destrucción de la clase contenida. En este caso, la CC es responsable de agregar sus componentes. Y la

CC puede existir sin necesidad de contener ninguna instancia de la clase contenida. En GAVA se emplea en casos como la relación entre los materiales (*Material*) y las plantillas (*Plantilla*) con el Consejo de Administración (*ConsejoAdmin*); así como los elementos afectados (*ElementoAfectado*) y la ficha técnica (*Ficha*).

- Composición: Al igual que la agregación, se emplea cuando una clase es contenida en una CC, pero en este caso, la destrucción de la contenedora conlleva a la destrucción de la contenida. Una vez exista la CC, automáticamente existirá la contenida, y si una de las dos desaparece (ya sea el Todo o la Parte), el otro también lo hará. En GAVA es el tipo de relación más recurrente; ya que una plantilla (*Plantilla*) está compuesta por una FT (*Ficha*) y sus cubicaciones (*Cubicacion*); una FT (*Ficha*) está compuesta por un dictamen de la vivienda (*Dictamen*), un núcleo familiar (*NucleoFamiliar*) y una o varias afectaciones (*Afectacion*); una afectación (*Afectacion*) está compuesta por el material predominante (*Material*) y la cubicación resultante (*Cubicacion*).
- Relación clase-interfaz: Cuando una clase implementa una interfaz, se obliga a la clase a implementar los métodos declarado en la interfaz, por ejemplo, en el sistema GAVA, tenemos que: las clases *Dictamen* y *Afectacion* implementan de la interfaz Dimensiones, ya que en Dimensiones se declaran métodos como obtener o colocar las dimensiones de largo y ancho, así como obtener el área; como tanto en el dictamen como en una afectación son necesarios estos procesos, se emplea la relación entre las clases y la interfaz. Algo similar sucede con la interfaz Identificador, donde se obtiene un código que permite identificar entidades de forma única, como los materiales (*Material*), dictámenes (*Dictamen*) y afectaciones (*Afectacion*) requieren de dicha identificación, implementan la interfaz Identificador.

2.1.2 Objetivos, atributos y funciones

A continuación, se explicarán los objetivos de cada una de las clases e interfaces que integran el sistema lógico de GAVA, así como los atributos presentes en cada clase; y

las funciones que deberán cumplir para un correcto funcionamiento del sistema y sus componentes.

Clases:

- *ConsejoAdmin*:

Objetivos:

- ✓ Control del sistema lógico
- ✓ Creación de fichas técnicas
- ✓ Creación y almacenamiento de materiales de la construcción
- ✓ Almacenamiento y gestión de las plantillas.
- ✓ Mostrar reportes.
- ✓ Administrar las configuraciones

Atributos:

- ✓ *listaMateriales*: Lista de arreglos de materiales de construcción.
- ✓ *listaPlantillas*: Lista de arreglos de las plantillas de viviendas.
- ✓ *reporte*: Instancia de la clase Reporte encargado de mostrarlos.
- ✓ *configuracion*: Instancia de la clase Configuracion encargada de modificar los ajustes.

Funciones:

- *Getters* (se encargan de obtener…):
  - ✓ *getListaMateriales*: la lista de arreglos de los materiales de la construcción.
  - ✓ *getListaPlantillas*: la lista de arreglos global de las plantillas de las viviendas.

- *Adders* (se encargan de agregar un objeto a una lista):
  - ✓ *agregarPlanilla*: agrega una planilla de vivienda a la lista de arreglos global de las planillas de viviendas.

- *Creators* (se encarga de crear un nuevo objeto):
  - ✓ *nuevaFicha*: Crea una nueva ficha técnica vacía.

- Otros:
  - ✓ *inicializarMateriales*: Crea los materiales de construcción creados por defecto y los almacena en la lista de arreglos de materiales de la construcción.
  - ✓ *inicializarValoresPorDefecto*: Pone valores predefinidos para los valores que empleará el sistema.
  - ✓ *definirMaterialNec*: Tomando como parámetro el nombre de un material devuelve el material requerido.

- Reportes:
  - ✓ *buscarViviendaMayorHab*: Llama y muestra el método de igual nombre de la clase Reporte, encargado de buscar el mayor número de habitantes de una vivienda, y las viviendas que tengan esta cantidad.
  - ✓ *buscarDatosMaterialSelec*: Llama y muestra el método de igual nombre de la clase Reporte, encargado de buscar los datos del material requerido por el usuario.
  - ✓ *buscarMaterialMasCant*: Llama y muestra el método de igual nombre de la clase Reporte, encargado de buscar cuál es el material con mayor cantidad de afectación, y dicho valor.
  - ✓ *buscarTotalFondos*: Llama y muestra el método *calcularTotalFondos* de la clase Reporte, encargado de calcular la suma total de todas las cubicaciones.
  - ✓ *buscarPromedioCub*: Llama y muestra el método *calcularPromedioCub*, encargado de calcular el promedio de todas las cubicaciones.
  - ✓ *buscarVivPrioridad*: Llama y muestra el resultado del método de igual nombre, encargado de devolver la vivienda con mayor prioridad a ser reparada.

- *Ficha*:

Objetivos:

- ✓ Manipulación de los datos de una ficha
- ✓ Creación y almacenamiento de un dictamen, núcleo familiar, afectaciones al inmueble y elementos afectados.

Atributos:

- ✓ *dictamen*: Un dictamen de vivienda.
- ✓ *nucleo*: Un núcleo familiar.
- ✓ *AfectacionesVivienda*: Una lista de arreglos de las afectaciones de la vivienda.
- ✓ *listaElementos*: Una lista de arreglos de los elementos afectados de la vivienda.

Funciones:

- *Getters*(se encarga de obtener…):
  - ✓ *getDictamen*: El dictamen de vivienda.
  - ✓ *getNucleo*: El núcleo familiar
  - ✓ *getAfectacionesVivienda*: La lista de arreglos de las afectaciones de la vivienda
  - ✓ *getListaElementos*: La lista de arreglos de las afectaciones de la vivienda.

- *Setters* (se encarga de insertar o modificar):
  - ✓ *setListaElementos*: La lista de arreglos de los elementos afectados.
  - ✓ *setListaAfectaciones*: La lista de arreglos de las afectaciones de la vivienda

- *Adders*(se encargan de agregar un objeto a una lista):
  - ✓ *agregarAfectacionVivienda*: Una afectación a la lista de arreglos de afectaciones de la vivienda.
  - ✓ *agregarElemento*: Un elemento a la lista de arreglos de elementos afectados.

- *Creators* (se encarga de crear un nuevo objeto):
  - ✓ *nuevaAfectacionPared*: Afectación de pared
  - ✓ *nuevaAfectacionTecho*: Afectación de techo
  - ✓ *nuevaListaElementos*: Lista de arreglos de elementos
  - ✓ *nuevoElemento*: Elemento afectado
  - ✓ *crearPlantilla*: Plantilla donde se contenga la ficha en sí y una lista de sus cubicaciones.

- Otros:
  - ✓ *agregarDatosTecho*: Tomando como parámetro la afectación de techo en cuestión y si es parcial o total; en caso de ser total, iguala las dimensiones de la afectación a las dimensiones de la vivienda.

- *Dictamen*:

Objetivos:

- ✓ Insertar, almacenar y gestionar los datos estructurales, físicos y legales de la vivienda.
- ✓ Obtener el área de la vivienda.

Atributos:

- ✓ *direccion*: De tipo *String*, constituye la dirección de la vivienda.
- ✓ *docLegal*: De tipo *String*, constituye el documento legal de la vivienda.
- ✓ *fechaLev*: De tipo *Date*, constituye la fecha del levantamiento de la vivienda.
- ✓ *tipoHabitacional*: De tipo *String*, constituye la tipología habitacional de la vivienda.
- ✓ *tipoConstructiva*: De tipo *String*, constituye la tipología constructiva de la vivienda.
- ✓ *ID*: De tipo *String*, constituye el código de identificación de la vivienda.
- ✓ *dim1*: De tipo *float*, constituye la una de las dimensiones de la vivienda (largo).

- ✓ *dim2*: De tipo *float*, constituye la otra dimensión de la vivienda (ancho).

Funciones:

- *Getters* (se encargan de obtener…):
  - ✓ *getDireccion*: la dirección de la vivienda
  - ✓ *getDocLegal*: el documento legal de la vivienda
  - ✓ *getFechaLev*: la fecha del levantamiento de la vivienda
  - ✓ *getTipoHabitacional*: la tipología habitacional de la vivienda
  - ✓ *getTipoConstructiva*: la tipología constructiva de la vivienda
  - ✓ *getID*: el identificador del dictamen
  - ✓ *getDim1*: el largo de la vivienda
  - ✓ *getDim2*: el ancho de la vivienda

- *Setters* (se encargan de insertar o modificar…):
  - ✓ *setDireccion*: la dirección de la vivienda
  - ✓ *setDocLegal*: el documento legal de la vivienda.
  - ✓ *setFechaLev*: la fecha del levantamiento de la vivienda.
  - ✓ *setTipoHabitacional*: la tipología habitacional de la vivienda.
  - ✓ *setTipoConstructiva*: la tipología constructiva de la vivienda.
  - ✓ *setDim1*: el largo de la vivienda.
  - ✓ *setDim2*: el ancho de la vivienda.

- Otros:
  - ✓ *Area*: Método polimórfico de la interface Dimensiones. Calcula el área de la vivienda multiplicando el largo y ancho de esta.

- *NucleoFamiliar*:

Objetivos:

- ✓ Ingresar, almacenar y gestionar el total de individuos que viven en una vivienda, así como cuántos de ellos son ancianos, niños o embarazadas.
- ✓ Insertar y almacenar el número de carné de identidad del jefe de núcleo.

Atributos:

- ✓ *numCI*: De tipo *String*, es el número de carné de identidad del jefe de núcleo.
- ✓ *totalPersonas*: De tipo *int*, constituye la cantidad de personas que viven en una vivienda.

- ✓ *cantAncianos*: De tipo *int*, constituye la cantidad de ancianos que viven en una vivienda.
- ✓ *cantNinnos*: De tipo *int*, constituye la cantidad de niños que viven en una vivienda.
- ✓ *cantEmbarazadas*: De tipo *int*, constituye la cantidad de embarazadas que viven en una vivienda.

Funciones:

- *Getters* (se encargan de obtener…):
  - ✓ *getNumCI*: Obtener el número de carné de identidad.
  - ✓ *getTotalPersonas*: total de habitantes
  - ✓
  - ✓ *getCantAncianos*: cantidad de ancianos
  - ✓ *getCantNinnos*: cantidad de niños
  - ✓ *getCantEmbarazadas*: cantidad de embarazadas

- *Setters* (se encargan de insertar o modificar…):
  - ✓ *setNumCI*: Insertar o modificar el número de carné de identidad.
  - ✓ *setTotalPersonas*: total de habitantes.
  - ✓ *setCantAncianos*: cantidad de ancianos.
  - ✓ *setCantNinnos*: cantidad de niños.
  - ✓ *setCantEmbarazadas*: cantidad de embarazadas.

- *Afectacion* (abstracta):

Objetivos:

- ✓ Ingresar, almacenar y gestionar los datos de una afectación al inmueble.
- ✓ Almacenar la cubicación correspondiente a la afectación.

Atributos:

- ✓ *afectTotal*: De tipo boolean, constituye el tipo de deterioro provocado por la afectación, si es parcial o total.
- ✓ *ID:* De tipo String, constituye el código de identificación de la afectación.
- ✓ *dim1*: De tipo float, constituye el largo de la superficie afectada.
- ✓ *dim2*: De tipo float, constituye el ancho de la superficie afectada.
- ✓ *material*: De tipo Material, es el material de la estructura afectada.
- ✓ *CubicacionesAfect*: Lista de arreglos de la cubicación de la afectación.

Funciones:

- *Getters* (se encargan de obtener…):
  - ✓ *getAfectTotal*: tipo de deterioro
  - ✓ *getID*: código de identificación de la afectación.

- ✓ *getDim1*: largo de la superficie deteriorada.
- ✓ *getDim2*: ancho de la superficie deteriorada.
- ✓ *getMaterial*: Material de la estructura afectada.
- ✓ *getCubicacionesVivienda*: lista de arreglos de la cubicación.

- *Setters* (se encargan de insertar o modificar…):
  - ✓ *setAfectTotal*: tipo de deterioro
  - ✓ *setDim1*: largo de la superficie deteriorada
  - ✓ *setDim2*: ancho de la superficie deteriorada.
  - ✓ *setMaterial*: Material de la estructura afectada.
  - ✓ *setCubicacionesVivienda*: lista de arreglos de la cubicación.

- *Adders*(se encargan de agregar un objeto a una lista):
  - ✓ *agregarCubicacion*: agrega una cubicación a la lista de arreglos de cubicaciones de la afectación.
- Otros:
  - ✓ *Area*: Método polimórfico de la interface Dimensiones. Calcula el área de la afectación multiplicando el largo y ancho de la superficie deteriorada.

- *AfectTecho* (hereda de *Afectacion*):

Objetivos:

- ✓ Clasificar una afectación según la superficie, en este caso el techo.

Atributos:

- ✓ Se heredan los atributos de *Afectacion*

Funciones:

- ✓ *Area*: Método polimórfico de la interface Dimensiones. Calcula el área de la afectación de techo, multiplicando el largo y ancho de la superficie deteriorada.

- *AfectPared* (hereda de *Afectacion*):

Objetivos:

- ✓ Clasificar una afectación según la superficie, en este caso el techo.

Atributos:

- ✓ Se heredan los atributos de *Afectacion*.

- ✓ carga: De tipo *boolean*, constituye el tipo de pared, si es de carga o no.

Funciones:

- ✓ *getCarga*: Obtiene si la pared es de carga o no.
- ✓ *setCarga*: Modifica si la pared es de carga o no.

- *Material*

Objetivos:

- ✓ Ingresar, almacenar y gestionar los datos de un material.

Atributos:

- ✓ *nombre*: De tipo *String*, es el nombre del material.
- ✓ *precioU*: De tipo *double*, es el precio de cada unidad del material.
- ✓ *unidadMedida*: De tipo String*,* es el tipo de unidad de medida del material.
- ✓ *ID*: De tipo *String*, es el código de identificación del material.

Funciones:

- *Getters* (se encargan de obtener…):
  - ✓ *getID*: código de identificación del material.
  - ✓ *getNombre*: nombre del material.
  - ✓ *getPrecioU*: precio de cada unidad del material.
  - ✓ *getUnidadMedida*: tipo de unidad de medida.

- *Setters* (se encargan de insertar o modificar…):
  - ✓ *setNombre*: nombre del material
  - ✓ *setPrecioU*: precio de cada unidad del material.
  - ✓ *setUnidadMedida*: tipo de unidad de medida.

- *Cubicacion*:

Objetivos:

- ✓ Calcular la cantidad de material requerido y el costo final.

Atributos:

- ✓ *material*: De tipo *Material*, es el material de la superficie afectada.
- ✓ *cantMaterialP*: De tipo *int*, es la cantidad de material requerido
- ✓ *precioMatP*: De tipo *double*, es el costo final de la cubicación.

Funciones:

- *Getters* (se encargan de obtener…):
  - ✓ *getMaterial*: material de la superficie afectada.
  - ✓ *getCantMaterialP*: cantidad de material requerido.
  - ✓ *getPrecioMatP*: costo final de la cubicación.

- *Setters* (se encargan de insertar o modificar…):
  - ✓ *setMaterial*: el material de la afectación.

- Otros:
  - ✓ *calcularCantMaterialP*: Calcula la cantidad de material requerido, a través de la multiplicación de la capacidad de dicho material en $1m^2$ y el área de la afectación.
  - ✓ *calcularPrecioMatP*: Calcula el costo final de la cubicación, a través de la multiplicación del precio unitario del material en cuestión por la cantidad de material requerido calculado previamente.

- *Reporte*:

Objetivos:

- ✓ Desarrollo de los reportes necesarios.

Atributos:

- ✓ *admin*: Instancia de la clase *ConsejoAdmin*.

Funciones:

- ✓ *calcularTotalFondos*: Calcula el total de fondos sumando el valor de todas las cubicaciones presentes en la lista de cubicaciones del *ConsejoAdmin*.
- ✓ *calcularPromedioFondos*: Calcula el promedio
- ✓ *buscarViviendaMayorHab*: Busca el mayor número de habitantes en una vivienda y busca todas las viviendas que tengan ese valor mayor de habitantes.
- ✓ *buscarMaterialMasCant*: Buscar el material con mayor cantidad requerida.
- ✓ *buscarDatosMaterialSelec*: Buscar los fondos totales y la cantidad requerida por un material seleccionado.
- ✓ *buscarViviendaPrioridad*: Empleando un algoritmo que depende de: el porciento de afectación de la vivienda, el número de habitantes vulnerables y el total de personas.

- *ViviendaMayorHab* (dentro de la clase Reporte):

Objetivos:

- ✓ Almacenar y devolver la mayor cantidad de habitantes y las viviendas con esa cantidad.

Atributos:

- ✓ ids: Lista de arreglos donde se almacenan las viviendas con la mayor cantidad de habitantes.
- ✓ cantHab: De tipo int, es la mayor cantidad de habitantes.

Funciones:

- ✓ *getIds*: Obtener la lista de arreglos de las viviendas con mayor cantidad de habitantes.
- ✓ *getCantHab*: Obtener la mayor cantidad de habitantes.

- *MaterialMasCant* (dentro de la clase Reporte):

Objetivos:

- ✓ Almacenar y devolver la mayor cantidad requerida de un material y los nombres de los materiales con este valor.

Atributos:

- ✓ *nombres*: Lista de arreglos donde se almacenan los nombres de los materiales con mayor cantidad.
- ✓ *fondos*: De tipo *double*, mayor cantidad requerida de un material.

Funciones:

- ✓ *getNombres*: Obtener la lista de arreglos de los nombres de los materiales con mayor fondo.
- ✓ *getFondos*: Obtener el mayor fondo.

- *DatosMaterialSelec* (dentro de la clase Reporte):

Objetivos:

- ✓ Almacenar y devolver los datos de un material requerido

Atributos:

- ✓ *nombre*: De tipo *String*, nombre del material.
- ✓ *fondo*: De tipo *double*, es el fondo total dedicado al material
- ✓ *cantidad*: De tipo *int*, es la cantidad de material total obtenida de las cubicaciones.
- ✓ *unidadMedida*: De tipo *String*, es la unidad de medida del material

Funciones:

- ✓ *getNombre*: Obtener el nombre del material
- ✓ *getFondo*: Obtener el fondo del material.
- ✓ *getCantidad*: Obtener la cantidad de material total
- ✓ *getUnidadMedida*: Obtener la unidad de medida del material.

<u>Interfaces:</u>

- Identificador

Objetivos:

- ✓ Permitir una correcta identificación de las entidades necesarias.

Funciones:

- ✓ *getID*: Obtener el código de identificación.

- Dimensiones

Objetivos:

- ✓ Permitir las mediciones de las entidades necesarias.
- ✓ Calcular el área.

Funciones:

- *Getter* (se encarga de obtener…):
  - ✓ *getDim1*: la primera dimensión.
  - ✓ *getDim2*: la segunda dimensión.
- *Setter* (se encarga de insertar o modificar):
  - ✓ *setDim1*: la primera dimensión.
  - ✓ *setDim2*: la segunda dimensión.

- Otros:
  - ✓ *Area*: Calcular el área de la entidad.

2.2 Tipos de lista implementados

El tipo de lista que se implementará en el sistema será *ArrayList*. De las que se han estudiado, es la más factible y eficiente, ya que permite que el acceso a un elemento de la lista sea inmediato a través del método *get*(). Además, la lista puede crecer y disminuir como sea necesario, a diferencia de un *Array* que debe tener un tamaño predefinido. El *ArrayList* será el único tipo de lista empleado en cada una de las posibles situaciones del proyecto.

2.3 Capas organizativas

2.3.1 Interfaz

- *main*: Contiene la clase *Main* encargada de inicializar los datos y correr la pantalla.
  - *Main*
- *prototipos*: Contiene la primera interfaz de prueba, que trabaja por consola. Actualmente no está funcional.
  - *InterfazV*
- *grafica*: Contiene las clases encargadas de la visualización y composición de la pantalla. Así como la pantalla en sí.
  - *Pantalla*
  - *Menu*
  - *Composicion*

2.3.2 Logica

- *logica*: Contiene el sistema lógico principal y funcional de GAVA.

  Clases:

  - *ConsejoAdmin*
  - *Ficha*

- *Plantilla*
- *Dictamen*
- *NucleoFamiliar*
- *Afectacion*
- *AfectTecho*
- *AfectPared*
- *Material*
- *Cubicacion*
- *Reporte*
- *Configuracion*

Interfaces:

- *Identificador*
- *Dimensiones*

- *utiles*: Contiene métodos útiles, valores iniciales y validaciones necesarias para el funcionamiento eficiente del sistema lógico. También presenta los métodos que permiten buscar y ordenar la tabla de las fichas.
  - *Utiles*
  - *Validaciones*
  - *ValorPorDefecto*
  - *AccionTabla*

2.3.3 Recursos

La carpeta Recursos no está constituido por paquetes, sino por otras carpetas: la carpeta de Imágenes donde se encuentra el logotipo del Ministerio de la Construcción que se emplea para el fondo de la pantalla de Inicio del programa; y la carpeta de Iconos, donde encontraremos los iconos de los botones del menú, así como el icono de GAVA.

2.4 Mecanismos de Validación

Para que los datos ingresados por el usuario se confirmen que son correctos, se debe recurrir a numerosos mecanismos de validación. Obtenidos con el apoyo visual de la interfaz y el procesamiento de los datos por parte de la lógica de GAVA usando como referencia los métodos validadores que se hallan en la clase Validaciones del paquete *utiles*. De esta forma, si el usuario introduce información que el sistema considera

inválida, se le notificará con un mensaje emergente, generalmente explicando de forma específica la situación causante de la invalidez.

2.4.1 Validaciones por campo

- *Dictamen*
  - ✓ Dirección: No debe ser mayor de 200 caracteres. Aunque se puede modificar en Ajustes. El usuario no puede ingresar más caracteres que los definidos.
  - ✓ Fecha de Levantamiento: Debe ser una fecha en el rango entre hoy y hace exactamente 10 años. El usuario no puede seleccionar en el calendario una fecha fuera de este rango.
  - ✓ Documento legal, tipología habitacional y tipología constructiva: Debe ser uno de las opciones *String* que tiene la lista. Gracias a la interfaz, se le obliga al usuario a seleccionar una de las opciones a través de *ComboBox*. En Ajustes se pueden agregar o eliminar opciones en estas listas.
  - ✓ Dimensiones: En una vivienda se tiene que las dimensiones deben presentar una longitud entre 3 y 150 metros. Aunque se puede modificar en Ajustes.

- *NucleoFamiliar*
  - ✓ Carné de identidad del jefe de núcleo: Por obligación debe presentar 11 caracteres numéricos. Comprobar que la fecha de nacimiento presentado en los 6 primeros dígitos sea válida, así como que el jefe sea mayor de edad.
  - ✓ Total de habitantes: No puede ser menor que 0, se tiene como máximo 15 habitantes. Aunque puede modificarse en Ajustes
  - ✓ Vulnerables: La suma de ancianos, embarazadas y niños de la vivienda no puede superar el valor máximo del total de habitantes.
  - ✓ Niños: No puede haber solo niños como habitantes en la vivienda, por lo que automáticamente se le agrega a ese total, uno más.

- *Afectacion*
  - ✓ Cantidad de afectaciones: Por obligación la vivienda debe presentar como mínimo 1 afectación. Solo puede haber una afectación de techo, en el caso de pared pueden ser hasta 20, pero dicho valor puede modificarse.
  - ✓ Dimensiones: Pueden ser entre 0.1 y 150 metros. Aunque estos límites pueden modificarse en Ajustes
  - ✓ Materiales: Debe elegirse una de las opciones de la lista de materiales disponibles para el tipo de afectación. En Ajustes se pueden agregar o eliminar materiales.
  - ✓ Afectación de techo: Si una afectación de techo es total, se le define automáticamente como dimensiones, las dimensiones de la vivienda.

- *ElementoAfectado*
  - ✓ Nombre: El nombre del elemento solo puede presentar caracteres alfabéticos. Y no ser mayor de 20 caracteres, aunque es modificable
  - ✓ Cantidad: una vez se desea agregar un elemento, su cantidad debe ser mayor que cero. Su límite máximo es 15, aunque se puede modificar.

- *Configuracion*
  - ✓ En este ámbito solo validaremos que los nuevos datos introducidos por el usuario sean correctos y válidos para realizar la modificación. Dígase que sean del mismo tipo.
  - ✓ En el caso de crear un material, este debe cumplir con las respectivas validaciones de su clase.
  - ✓ En el caso de crear documentación legal o una tipología, la nueva opción debe cumplir unos límites obligatorios de caracteres.

- *Material*
  - ✓ ID: El código de un material debe ser un número mayor o igual que 8, menor o igual que 12.
  - ✓ Nombre: El nombre del material debe cumplir con un límite de caracteres, y solo implementar caracteres alfabéticos.
  - ✓ Precio: Debe ser un valor mayor que 0.
  - ✓ Unidad de medida: Debe ser una de las que se encuentre en la lista, que no es modificable.
  - ✓ Cantidad necesaria por metro cuadrado: Debe ser mayor que 0

2.4.2 Tratamiento de excepciones

Cuando el usuario inserta un valor o una opción, llegará al método *set()* de dicho atributo. Dentro del método *set()* se tendrá una condicional donde, al pasar por un método de validación (que se encontraría en la clase *Validaciones* del paquete *utiles*); si el valor insertado cumple con los requisitos del método de validación para ser válido, entonces el valor introducido será almacenado en el atributo requerido. Sin embargo, si no cumple con los requisitos de la validación, retornará que es erróneo y en el método *set()* del atributo se enviará una excepción. En otros casos, cuando el método de validación es lo suficientemente complejo, él mismo se encarga de lanzar una excepción, generalmente presenta varias, según el tipo de error que tenga el valor introducido.

Para tratar las excepciones se usará el try/catch en la interfaz gráfica. En los formularios de Nueva Ficha, dichos try/catch se encuentran en los botones de Siguiente para pasar a la próxima pantalla. Dentro de este bloque se intentaba que los atributos del objeto creado tomarán los valores introducidos, y si hallaba una excepción, el catch se encargaba de tomarlo y hacer aparecer una ventana emergente de Error o Advertencia, avisándole al usuario de lo acontecido.

Si alguna casilla de una pantalla del formulario se mantiene vacía para cuando se toque el botón de Siguiente, saltará un mensaje de Error confirmando de que existen casillas sin completar, así como los bordes de dicha casilla se marcarán en rojo hasta que se vuelva a introducir un nuevo valor.

Generalmente cuando se encuentra una excepción relacionada con uno de los datos ingresados, se le anunciará con un mensaje de Advertencia al usuario del dato que está incorrecto, así como cuál es su error.

2.5 Diseño de Interfaz

Para el desarrollo de la interfaz gráfica de GAVA se tuvieron en cuenta numerosas consideraciones en aspectos diversos como teoría del color, principios y patrones de diseño

2.5.1 Teoría del color

El color predominante y característico de la interfaz de GAVA es el color verde mate, y sus respectivas tonalidades. Esta decisión se basa directamente en el color del logotipo del Ministerio de la Construcción de Cuba.

El empleo del color verde inevitablemente conlleva a la asociación por parte del usuario con la naturaleza, relacionándose íntimamente con los fenómenos naturales que provocan las afectaciones en las viviendas que se registrarán en el sistema. Como cualidades positivas tenemos que es un color que apacigua y tranquiliza.

El verde se asocia con el pensamiento estratégico, el optimismo y la seguridad. Es percibido como un color natural y saludable, sinónimo de frescura y aire limpio.

También se presenta, aunque en menor medida, el blanco y el gris. Se tuvo en cuenta evitar colores brillantes sobre fondos opacos, por el contrario, los fondos varían entre verde manzana mate y blanco, y las letras negras. Mientras que el menú presenta el mismo color verde, pero con dos tonalidades un poco más oscuras.

2.5.2 Principios generales de diseño

- Consistencia: GAVA será construido sobre las bases de la experiencia del mundo real del usuario, asimilando el registro de fichas técnicas de forma más dinámica y conocida. El sistema debe ser funcionalmente simple y consistente. Todos los componentes tendrán similar apariencia y comportamiento. La terminología sería la misma en los menús. Siempre se usará el mismo estilo para mostrar los mensajes.

- Control de la aplicación: El usuario siempre llevará el control de los procesos, dígase, desplazarse libremente en el sistema y siguiendo flujos de pantallas determinados. Gracias al diseño del Menú Principal basado en pestañas, el usuario tendrá visibles todas las opciones disponibles.

- Retroalimentación: Se debe poner al tanto al usuario de los procesos que ocurran en tiempo real. Ejemplo: si el usuario introduce un valor incorrecto o vacío se le informa; cuando la ficha está creada se le muestra un mensaje al usuario de Ficha Registrada con Éxito; el usuario puede siempre revisar la lista de Fichas para confirmar el registro. Además, los diferentes botones cambiarán su color y el de su texto según si el cursor entra o sale de este, así como si se presiona. Durante el proceso de registro de ficha, se mostrará una barra que representa el progreso de registro.

- Rectificabilidad: Esto constituye que las acciones del usuario deben ser reversibles y corregibles. Siempre se podrá cambiar los datos de la ficha que se está registrando, reiniciarlos, cambiar entre pantallas con libertad, y eliminar alguna ficha registrada.

- Estética: La aplicación debe ser visualmente atractiva para el usuario, combinando el poder funcional con buena apariencia.

- Sencillez: GAVA debe ser comprensible, de fácil aprendizaje y uso. Se utilizará un lenguaje familiar para el usuario, reduciendo la carga visual innecesaria, en busca de un diseño basado en el minimalismo.

- Claridad: El sistema será visual y lingüísticamente claro, los elementos visuales deben comprenderse con facilidad y que los textos no sean ambiguos.

Estos principios generales fueron extraídos de [6].

2.6 Reportes

Además de presentar las FT y sus datos en una tabla, GAVA presenta otros reportes, relacionados con el ámbito financiero, económico y administrativo, estos serían:

- Buscar las viviendas con el mayor número de habitantes. Este reporte brinda uno de los factores a considerar para priorizar la reparación de aquellas viviendas con el mayor número de personas afectadas.
- Buscar todos los datos administrativos y económicos de un material específico deseado. Entre estos datos podemos encontrar los fondos totales dedicados a dicho material, así como la cantidad requerida del material.
- Buscar el material con la mayor cantidad requerida. Así el Ministerio de la Construcción y los Consejos de Administración tendrán en cuenta cuales son los materiales más requeridos para la reparación de las viviendas afectadas, y priorizar el aumento de unidades de dicho material.
- Buscar el total de fondos requeridos para la reconstrucción de las viviendas afectadas. Así se tendrá noción de cuanto se debe dedicar para esta causa.
- Obtener, a través de un algoritmo, la vivienda que debería ser priorizada para reconstruir primero de la lista de fichas.

2.7 Filosofía de trabajo

La principal filosofía laboral aplicada ha sido la filosofía Kaizen, proveniente del mundo empresarial japonés. Se basa en la estrategia de la mejora continua para la calidad total. Esta filosofía se refiere a un sistema de mejora progresiva en el que las pequeñas pero constantes mejoras de los procesos y organizaciones, acumulan tras de sí grandes beneficios a largo plazo. Todos sus fundamentos se podrían resumir en una frase: “Hoy se trabaja mejor que ayer, pero peor que mañana. [7]

Otras filosofías consideradas son que se debe trabajar no solo por los resultados económicos, sino principalmente por la satisfacción consecuente de alcanzar la concentración en determinadas tareas creativas que potencian la capacidad de raciocinio lógico, así como la relación mantenida con otros individuos mediante estas tareas, ampliando el entorno social. Los resultados del trabajo permiten sentir que los conocimientos y tiempo de preparación han decantado en una labor útil para la sociedad.

3.1 Casos de prueba

En ingeniería de software, un caso de prueba es un conjunto de condiciones o variables bajo las cuales se determinará si una aplicación, un sistema informático o una característica o comportamiento de estos resulta o no aceptable. [8]

Con el propósito de comprobar que todos los requisitos de un sistema son revisados, debe existir al menos un caso de prueba para cada requisito. En el caso de GAVA, serán revisados aquellos métodos más relevantes de las clases esenciales relacionadas con la lógica del modelo de "negocio" del sistema

Existe una gran variedad de casos de prueba para determinar que un requisito es completamente satisfactorio. Para probar el sistema GAVA se emplearán casos de prueba de tipo caja blanca y caso de uso.

3.1.1 Caja Blanca

Un caso de prueba de tipo caja blanca es una técnica de prueba que evalúa el código y la estructura interna de un programa. Implican la observación de la estructura del código. Este tipo de caso de prueba permite descubrir errores como aquellos errores lógicos que resultan del diseño e implementación de funciones, condiciones o controles externos al programa, también se descubren errores de diseño que ocurren por la posible diferencia entre el flujo lógico del programa y la implementación real. [8]

- Consejo_Admin
  - ✓ definirMaterialNec

Caso de Prueba			
Método	definir_MaterialNec		
Desarrollador	Brian Serrano Alfonso		
Probador	Brian Serrano Alfonso		
Fecha	10/9/2022		
Combinaciones de valores de entrada			

CP	Variable	Valor	Técnica de diseño empleada	Resultados esperados	Resultados reales
1	String nombreMaterial	Madera	Bucle	Madera	Madera
2	String nombreMaterial	null	Bucle	Exception	Exception

- ✓ getListaMateriales

Caso de Prueba					
Método	getListaMateriales				
Desarrollador	Brian Serrano Alfonso				
Probador	Brian Serrano Alfonso				
Fecha	10/9/2022				
Combinaciones de valores de entrada			Técnica de diseño empleada	Resultados esperados	Resultados reales
CP	Variable	Valor			
1	Material clavo	ID=123123132 Nombre=Clavo PrecioU=1 UnidadMedida=U MatCantNec=1	Prueba de condición	Material clavo	Material clavo

- Ficha

- ✓ AgregarElemento

Caso de Prueba	
Método	AgregarElemento
Desarrollador	Brian Serrano Alfonso

Probador	Brian Serrano Alfonso				
Fecha	10/9/2022				
Combinaciones de valores de entrada			Técnica de diseño empleada	Resultados esperados	Resultados reales
CP	Variable	Valor			
1	ListaAfectaciones.size(), elemento	0, televisor, 2	Condición	1	1
	ListaAfectaciones.size(),Elemento1, elemento2	0, televisor, 2, lavamanos,1	Condición	2	2

- ✓ AgregarAfectacion

Caso de Prueba					
Método	AgregarAfectacion				
Desarrollador	Brian Serrano Alfonso				
Probador	Brian Serrano Alfonso				
Fecha	10/9/2022				
Combinaciones de valores de entrada			Técnica de diseño empleada	Resultados esperados	Resultados reales
CP	Variable	Valor			
1	AfectacionVivienda.size(), afectacion	0,false,12,21, madera	Condición	0	0

- Planilla_Vivienda

- ✓ getTipoHabitacional

Caso de Prueba

Método	getTipoHab
Desarrollador	Brian Serrano Alfonso
Probador	Brian Serrano Alfonso
Fecha	10/9/2022

Combinaciones de valores de entrada			Técnica de diseño empleada	Resultados esperados	Resultados reales
CP	Variable	Valor			
1	String tipoHab	Casa	Condición	Casa	Casa
2	String tipoHab	Apartamento	Condición	Apartamento	Apartamento
3	String tipoHab	Trapiche	Condición	Exception	Exception

- ✓ Area

Caso de Prueba	
Método	Area
Desarrollador	Brian Serrano Alfonso
Probador	Brian Serrano Alfonso
Fecha	10/9/2022

Combinaciones de valores de entrada			Técnica de diseño empleada	Resultados esperados	Resultados reales
CP	Variable	Valor			
1	Dim1, Dim2	26, 31	Condición	806	806
2	DIm1, Dim2	0, -5	Condición	Exception	Exception
3	Dim1, Dim2	350,4500	Condición	Exception	Exception

- NucleoFamiliar

- ✓ getTotal

Caso de Prueba					
Método	getTotal				
Desarrollador	Brian Serrano Alfonso				
Probador	Brian Serrano Alfonso				
Fecha	10/9/2022				
Combinaciones de valores de entrada			Técnica de diseño empleada	Resultados esperados	Resultados reales
CP	Variable	Valor			
1	TotalPersonas, cantAncianos, cantEmbarazadas, cantNinnos	13,2,3,4	Condición	13	13
2	TotalPersonas, cantAncianos, cantEmbarazadas, cantNinnos	13,26,3,1	Condición	Exception	Exception
3	TotalPersonas, cantAncianos, cantEmbarazadas, cantNinnos	0,0,0,0	Condición	Exception	Exception

- ✓ getAncianos

Caso de Prueba					
Método					
Desarrollador	Brian Serrano Alfonso				
Probador	Brian Serrano Alfonso				
Fecha	10/9/2022				
Combinaciones de valores de entrada			Técnica de diseño empleada	Resultados esperados	Resultados reales
CP	Variable	Valor			
1	TotalPersonas, cantAncianos, cantEmbarazadas, cantNinnos	13, 2, 4, 1	Condición	2	2

	TotalPersonas, cantAncianos, cantEmbarazadas, cantNinnos	13, 23,2,1	Condición	Exception	Exception
	TotalPersonas, cantAncianos, cantEmbarazadas, cantNinnos	13, 0, 2, 3	Condición	Exception	Exception

- ElementoAfectado

  ✓ getElemento

Caso de Prueba					
Método	getElemento				
Desarrollador	Brian Serrano Alfonso				
Probador	Brian Serrano Alfonso				
Fecha	10/9/2022				
Combinaciones de valores de entrada			Técnica de diseño empleada	Resultados esperados	Resultados reales
CP	Variable	Valor			
1	elemento	Televisor,2	Condición	Televisor	Televisor
2	Elemento	Null,2	Condición	Exception	Exception
3	elemento	Televisor, 0	Condición	Exception	Exception

  ✓ getCantidad

Caso de Prueba	
Método	10/9/2022
Desarrollador	Brian Serrano Alfonso
Probador	Brian Serrano Alfonso
Fecha	10/9/2022

Combinaciones de valores de entrada			Técnica de diseño empleada	Resultados esperados	Resultados reales
CP	Variable	Valor			
1	elemento	Televisor,2	Condición	2	2
2	Elemento	Null,2	Condición	Exception	Exception
3	elemento	Televisor, 0	Condición	Exception	Exception

- Afect_Pared

  ✓ getDim1

Caso de Prueba	
Método	getDim1
Desarrollador	Brian Serrano Alfonso
Probador	Brian Serrano Alfonso
Fecha	10/9/2022

Combinaciones de valores de entrada			Técnica de diseño empleada	Resultados esperados	Resultados reales
CP	Variable	Valor			
1	Dim1	13	Condición	13	13
2	Dim1	132132	Condición	Exception	Exception
3	Dim1	-232	Condición	Exception	Exception

  ✓ getAfectTotal

Caso de Prueba	
Método	getAfectTotal
Desarrollador	Brian Serrano Alfonso
Probador	Brian Serrano Alfonso
Fecha	10/9/2022

Combinaciones de valores de entrada			Técnica de diseño empleada	Resultados esperados	Resultados reales
CP	Variable	Valor			
1	AfectTotal	true	Condición	true	true
2	AfectTotal	False	Condición	false	false
3	AfectTotal	null	Condición	Exception	Exception

- Afect_Techo

  ✓ getDIm2

Caso de Prueba	
Método	getDim2
Desarrollador	Brian Serrano Alfonso
Probador	Brian Serrano Alfonso
Fecha	10/9/2022

Combinaciones de valores de entrada			Técnica de diseño empleada	Resultados esperados	Resultados reales
CP	Variable	Valor			
1	Dim2	13	Condición	13	13
2	Dim2	132132	Condición	Exception	Exception
3	Dim2	-232	Condición	Exception	Exception

  ✓ getAfectTotal

Caso de Prueba	
Método	getAfectTotal
Desarrollador	Brian Serrano Alfonso
Probador	Brian Serrano Alfonso
Fecha	10/9/2022

Combinaciones de valores de entrada			Técnica de diseño empleada	Resultados esperados CP	Resultados reales
CP	Variable	Valor			
1	AfectTotal	true	Condición	1	
2	AfectTotal	False	Condición	2	
3	AfectTotal	null	Condición	3	

- Cubicacion

✓ CalcularCantMaterial

Caso de Prueba	
Método	Calcular_CantMaterialNec
Desarrollador	Brian Serrano Alfonso
Probador	Brian Serrano Alfonso
Fecha	10/9/2022

Combinaciones de valores de entrada			Técnica de diseño empleada	Resultados esperados	Resultados reales
CP	Variable	Valor			
1	Área,nombreMaterial	12, Madera	Condición	13	13
2	Área,material	-2, Madera	Condición	Exception	Exception
3	Área,material	12, “”	Condición	Exception	Exception

✓ CalcularPrecio

Caso de Prueba	
Método	Calcular_PrecioMatP()
Desarrollador	Brian Serrano Alfonso
Probador	Brian Serrano Alfonso
Fecha	10/9/2022

Combinaciones de valores de entrada			Técnica de diseño empleada	Resultados esperados	Resultados reales
CP	Variable	Valor			
1	precioU, cantMaterialP	15, 13	Condición	195	195
2	precioU, cantMaterialP	0,13	Condición	Exception	Exception

- Material

  ✓ getUnidadMedida

Caso de Prueba	
Método	getUnidadMedida
Desarrollador	Brian Serrano Alfonso
Probador	Brian Serrano Alfonso
Fecha	10/9/2022

Combinaciones de valores de entrada			Técnica de diseño empleada	Resultados esperados	Resultados reales
CP	Variable	Valor			
1	unidadMedida	U	Condición	U	U
2	unidadMedida	m^3	Condición	m^3	m^3
3	unidadMedida	pastelito	Condición	Exception	Exception

  ✓ getPrecioU

Caso de Prueba	
Método	getPrecioU
Desarrollador	Brian Serrano Alfonso
Probador	Brian Serrano Alfonso
Fecha	10/9/2022

Combinaciones de valores de entrada			Técnica de diseño empleada	Resultados esperados	Resultados reales
CP	Variable	Valor			
1	precioU	2.00	Condición	2.00	2.00
2	precioU	-2.00	Condición	Exception	Exception

- Reporte

  - ✓ TotalFondos

Caso de Prueba					
Método	calcularTotalFondos				
Desarrollador	Brian Serrano Alfonso				
Probador	Brian Serrano Alfonso				
Fecha	10/9/2022				
Combinaciones de valores de entrada			Técnica de diseño empleada	Resultados esperados	Resultados reales
CP	Variable	Valor			
1	ListaMateriales.size(), precioMat1, precioMat2, precioMat3	3, 234, 232, 452	Condición	918	918
2	ListaMateriales.size(), precioMat1, precioMat2, precioMat3	0, 0, 0, 0	Condición	Exception	Exception
3	ListaMateriales.size(), precioMat1, precioMat2, precioMat3	3, 0, 12, 344	Condición	Exception	Exception

- ✓ PromedioFondos

Caso de Prueba					
Método	calcularPromedioFondos				
Desarrollador	Brian Serrano Alfonso				
Probador	Brian Serrano Alfonso				
Fecha	10/9/2022				
Combinaciones de valores de entrada			Técnica de diseño empleada	Resultados esperados	Resultados reales
CP	Nombres de variables de entrada (V1,V2,…,Vn)	Para cada variable de entrada: variable=valor			
1	ListaMateriales.size(), precioMat1, precioMat2, precioMat3	3, 234, 232, 452	Condición	306	306
2	ListaMateriales.size(), precioMat1, precioMat2, precioMat3	0, 0, 0, 0	Condición	Exception	Exception
3	ListaMateriales.size(), precioMat1, precioMat2, precioMat3	3, 0, 12, 344	Condición	Exception	Exception

3.1.2 Casos de Uso

Un caso de uso es la descripción de un conjunto de interacciones que inicia un actor principal con el sistema. Sirve para especificar la comunicación y el comportamiento del sistema mediante su interacción con el usuario u otro sistema. Esta técnica de caso de prueba tiene éxito en sistemas interactivos. Permite centrarse en las necesidades del usuario y lo que espera lograr al usar el sistema. [8]

Identificador	Nombre HU	Prioridad	Puntos estimados
HU-DM	HU-DefinirMaterial		
Usuario	Usuario1		
Descripción	Al insertar el nombre de un material, retorna dicho material		
Programador responsable	Brian Serrano		

Escenario	Condición	Resultado esperado
Retorno de material	nombreMaterial aparece en la búsqueda en la lista de materiales	material
No retorno del material	nombreMaterial no aparezca en la búsqueda en la lista de materiales	null

Identificador	Nombre HU	Prioridad	Puntos estimados
HU-TH	HU-TipoHab		
Usuario	Usuario2		
Descripción	Obtiene la tipología habitacional de la vivienda		
Programador responsable	Brian Serrano		

Escenario	Condición	Resultado esperado
Obtener tipología	tipoHab debe coincidir con algún valor del Enum TipoHab	tipoHab
NO obtener tipología	tipoHab no coincide con ningún valor del Enum TipoHab	Exception

Identificador	Nombre HU	Prioridad	Puntos estimados
HU-D1	HU-getDim1		
Usuario	Usuario3		
Descripción	Obtiene el largo de la vivienda		
Programador responsable	Brian Serrano		

Escenario	Condición	Resultado esperado
Obtener largo	0 < dim1 < 200	Dim1
No obtener largo	Dim < 0 o DIm1 > 200	Exception

Caso de Prueba						
Historia de usuario	HU-DM					
Desarrollador	Brian Serrano Alfonso					
Probador	Brian Serrano Alfonso					
Fecha	10/9/2022					
Combinaciones de valores de entrada				Técnica de diseño empleada	Resultados esperados	Resultados reales
CP	Escenarios	Nombre de variables de entrada	Valor de variables de entrada			
1	Retornar material	nombreMaterial	“Madera”	Partición equivalente	Material madera	Material madera
2	No retorna material	nombreMaterial	“Panqueque”	Partición equivalente	null	null

Caso de Prueba						
Historia de usuario		HU-TH				
Desarrollador		Brian Serrano Alfonso				
Probador		Brian Serrano Alfonso				
Fecha		10/9/2022				
Combinaciones de valores de entrada				Técnica de diseño empleada	Resultados esperados	Resultados reales
CP	Escenarios	Nombre de variables de entrada	Valor de variables de entrada			
1	Obtener tipologia	tipoHab	“Casa”	Partición equivalente	“Casa”	“Casa”
2	No obtener tipologia	tipoHab	“Trapiche”	Partición equivalente	null	null

Caso de Prueba						
Historia de usuario		HU-D1				
Desarrollador		Brian Serrano Alfonso				
Probador		Brian Serrano Alfonso				
Fecha		10/9/2022				
Combinaciones de valores de entrada				Técnica de diseño empleada	Resultados esperados	Resultados reales
CP	Escenarios	Nombre de variables de entrada	Valor de variables de entrada			
1	Obtener largo	Dim1	34	Partición equivalente	34	34
2	No obtener largo	Dim1	2343243	Partición equivalente	null	null

CONCLUSIONES

Con la finalización del desarrollo del sistema informático GAVA podemos llegar a la conclusión de que el sistema logró cumplir con su objetivo inicial y principal: convertirse en un programa capaz de digitalizar la información de las viviendas afectadas y los daños provocados por un evento meteorológico, facilitando el almacenamiento seguro de dichos datos. Además de permitir automatizar el cálculo de la cuantificación de daños y evitar el error humano. También se logró cumplir el objetivo a corto plazo, que era que GAVA almacenara de forma eficiente la información necesaria, posibilitando visibilizarla y administrarla en cualquier momento y de forma organizada.

La interfaz gráfica de GAVA fue creada a partir de los principios y patrones de diseño, logrando crear un programa eficiente e intuitivo. La interfaz fue diseñada y construida teniendo como objetivo principal lograr una optimización máxima de la experiencia de usuario.

GAVA constituyó el debut del desarrollador en el mundo de la POO, permitiéndole profundizar en el amplio espectro de conocimientos relacionados; siendo de esta forma, el punto de partida para futuros proyectos,

RECOMENDACIONES

GAVA fue creada en función de solucionar la problemática que fue propuesta. Por tanto, es de conocimiento general el hecho de que el estado actual del sistema no le permite constituir una aplicación capaz de usarse en situaciones de la vida real.

De esta forma, se recomienda, en un futuro, implementar nuevas funcionalidades y realizar modificaciones en función de la realidad objetiva de nuestra sociedad actual. Entre estas modificaciones podríamos agregar: la implementación de un sistema de base de datos, facilitando así el eficiente almacenamiento por tiempo indefinido y seguridad de los datos como las plantillas de las viviendas, los materiales de la construcción, y los datos con los que trabaja el sistema.

También se recomienda separar GAVA en dos aplicaciones con funciones más específicas. Dichas aplicaciones serían:

- EDEN (Evaluación de Daños por Eventos Naturales):
  - ✓ Objetivo: Crear la FT y enviarla a un servidor dedicado de las OT.
  - ✓ Soporte: Dispositivos móviles Android.
  - ✓ Usuarios: Expertos encargados de realizar la evaluación inicial de los daños a la vivienda y crear la respectiva FT.

- GAVA (Gestión y Almacenamiento de Viviendas Afectadas):
  - ✓ Objetivo: Calcular y almacenar las cubicaciones de la FT y almacenar dicha FT y su lista de cubicaciones en una plantilla. Almacenar esta plantilla en la base de datos. Permitir visualizar y gestionar las plantillas almacenadas y mostrar reportes financieros y económicos respecto a estas.
  - ✓ Soporte: PC.
  - ✓ Usuarios: Expertos encargados almacenar y gestionar las plantillas de la vivienda. Trabajadores de las OT, Consejos de Administración, Defensa Civil o MICONS.

BIBLIOGRAFÍA

[1] Asamblea Nacional del Poder Popular (2019) Constitución de la República de Cuba. Editora Política. La Habana, Cuba.

[2] Novoa Rodríguez, Leonardo (2018, junio) Módulo Otorgamiento de subsidios a la población cubana para la construcción de viviendas de la plataforma Bienestar. Tesis Ing. en Ciencias Informáticas. UCI. La Habana.

[3] Resolución 41. Gaceta Oficial de la República de Cuba. ANEXOS I, II y II. La Habana. 20 de abril de 2022.

[4] Beck, Kent; Cunningham, Ward. A Laboratory for Teaching Object-Oriented Thinking. Actas de la Conferencia OOPSLA'89 del 1 al 6 de octubre de 1989, Nueva Orleans, Luisiana.

[5] La guía sencilla para la diagramación de UML. Microsoft 365 Team. 24 de septiembre de 2019.

[6] Tidwell, Jenifer (2010, diciembre) Designing Interfaces, Second Edition. O'Reilly Media. Canadá.

[7] Álvarez Arteaga, Areli (2020, septiembre 23). "¿Qué es y en qué consiste la filosofía Kaizen? Pasos y ejemplos". Lean Construction México

[8] Departamento de Informática, Universidad de Valladolid: Casos de Uso (PDF). Archivado desde el original el 5 de julio de 2016. Consultado el 21 de noviembre de 2022.

García Borroto, Milton; Pérez Lovelle, Sonia; Porras Nodarse Cynthia; Padrón Prado, Jorge & Nodarse Cañizares, Camilo J. (2020, junio 26) Introducción a la Programación Orientada a Objetos (POO) en JAVA. La Habana.

Stricker, Scott (2002, mayo 28) Java programming for C/C++ developers. IBM Corporation. US.

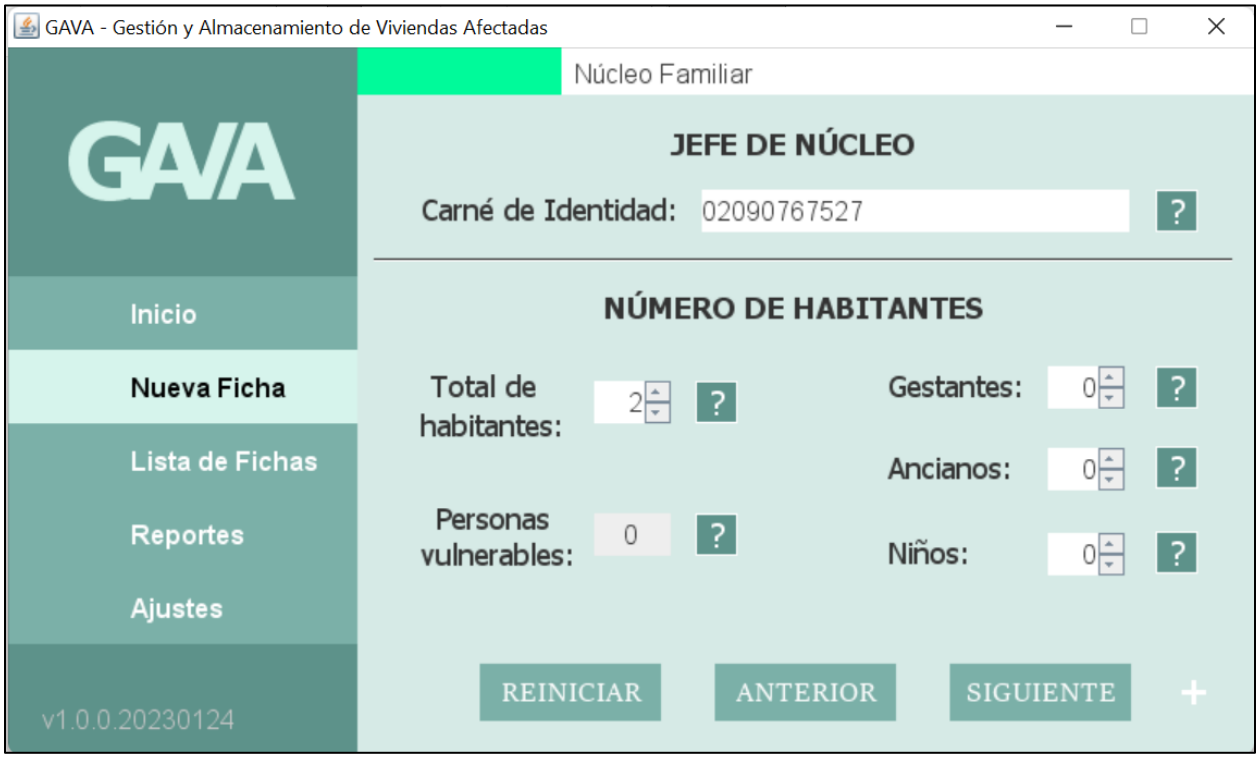
Suárez Rivas, Ronald. Papeles y tempestades después del huracán. Periódico Granma versión web: 13 de octubre de 2022.

ANEXOS

Anexo 1: Logotipo del MICONS empleado en GAVA

Anexo 2: Pantalla de Inicio

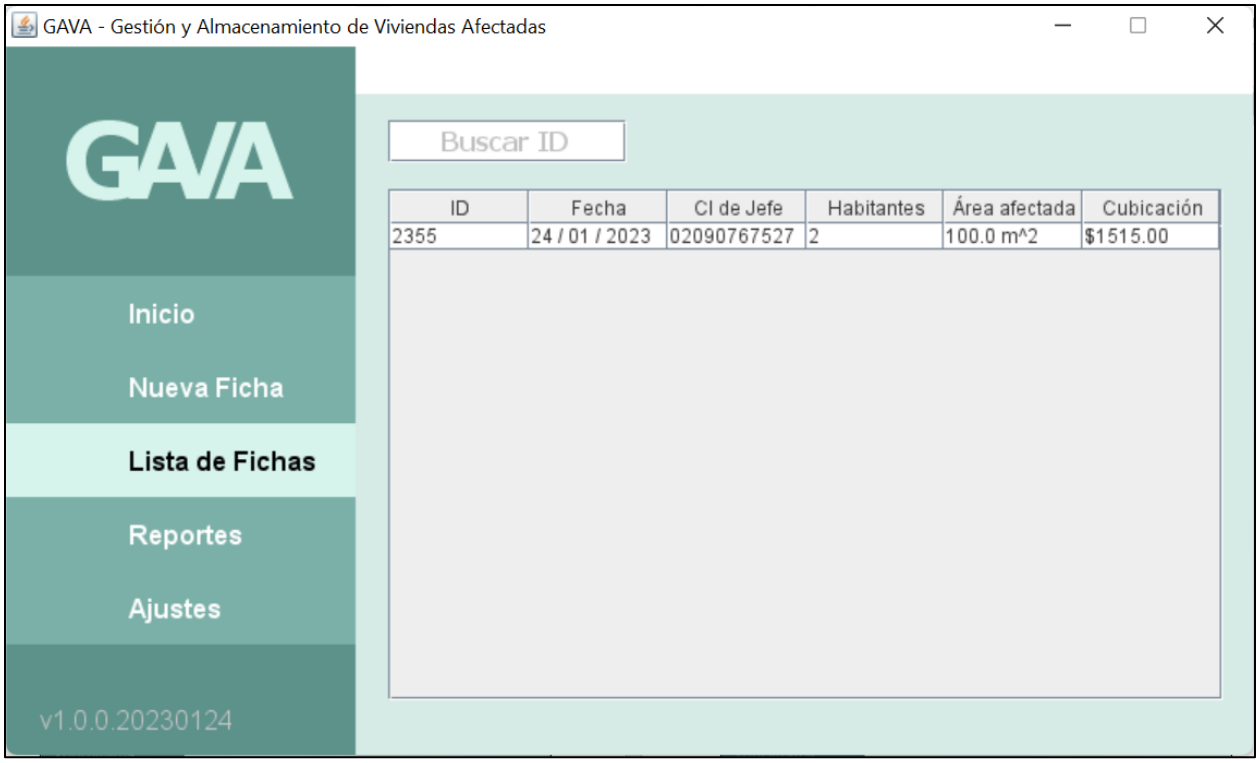
Anexo 3: Pantalla de Nueva Ficha 1: Dictamen



Anexo 4: Pantalla de Nueva Ficha 2: Núcleo Familiar

Anexo 5: Pantalla de Nueva Ficha 3: Afectaciones

Anexo 6: Pantalla de Nueva Ficha 4: Elementos Afectados



Anexo 7: Pantalla de Lista de Fichas