UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

ESTRUCTURA DE DATOS

CATEDRÁTICO: ING. EDGAR RENE ORNELIS HOILS

TUTOR ACADÉMICO: ELIAN SAUL ESTRADA URBINA



**MANUAL TÉCNICO RENTA DE ACTIVOS**

ENNER ESAÍ MENDIZABAL CASTRO

CARNÉ: 202302220

SECCIÓN: A

GUATEMALA, 16 DE DICIEMBRE DEL 2,024

# ÍNDICE

[ÍNDICE 1](#_Toc165147344)

[INTRODUCCIÓN 2](#_Toc165147345)

[OBJETIVOS 2](#_Toc165147346)

[1. GENERAL 2](#_Toc165147347)

[2. ESPECÍFICOS 2](#_Toc165147348)

[ALCANCES DEL SISTEMA 2](#_Toc165147349)

[ESPECIFICACIÓN TÉCNICA 3](#_Toc165147350)

[● REQUISITOS DE HARDWARE 3](#_Toc165147351)

[● REQUISITOS DE SOFTWARE 3](#_Toc165147352)

[DESCRIPCIÓN DE LA SOLUCIÓN 4](#_Toc165147353)

[LÓGICA DEL PROGRAMA 7](#_Toc165147354)

# INTRODUCCIÓN

Este documento técnico tiene como objetivo proporcionar una descripción detallada de la arquitectura, diseño e implementación del programa Renta de Activos. Se presentarán las soluciones técnicas empleadas, los tipos de datos abstractos utilizados y la lógica de programación subyacente a cada una de sus funcionalidades. Con esto, se busca facilitar la comprensión del sistema a desarrolladores interesados en replicar o extender sus capacidades.

# OBJETIVOS

## GENERAL

* 1. Documentar de manera exhaustiva las soluciones técnicas adoptadas en la implementación del programa Renta de Activos.

## ESPECÍFICOS

* 1. Detallar los Tipos de Datos Abstractos (TDA) utilizados en la construcción del programa y su justificación.
  2. Describir la lógica de implementación de las estructuras de datos y algoritmos empleados, así como su relación con las funcionalidades del sistema.

# ALCANCES DEL SISTEMA

Este manual se centra en la arquitectura interna del programa Renta de Activos, proporcionando una visión detallada de su diseño y desarrollo. Se abordan aspectos como:

* Estructura: Organización general del código y relación entre los diferentes módulos.
* Algoritmos: Descripción de los algoritmos utilizados para resolver los problemas planteados.
* Tipos de datos: Definición y uso de los TDA empleados para representar la información.
* Tecnologías: Herramientas y lenguajes de programación utilizados en el desarrollo.

Este documento está dirigido a desarrolladores con conocimientos básicos de programación y un interés particular en la arquitectura de software y el diseño de algoritmos.

# ESPECIFICACIÓN TÉCNICA

## REQUISITOS DE HARDWARE

* + Resolución mínima de 1024x768
  + 8 GB de memoria RAM
  + Dispositivos de entrada y salida: Pantalla, Ratón y Teclado.
  + 3.5GB de almacenamiento libre en disco

## REQUISITOS DE SOFTWARE

* + Sistema operativo: Windows, macOS o Linux
  + Compilador de C++/C. (Si se tiene Clion, no es necesario)
  + Entorno de desarrollo. Se sugiere utilizar un entorno de desarrollo integrado (IDE) como Clion, aunque cualquier editor de texto con soporte para C++ puede ser utilizado.

# DESCRIPCIÓN DE LA SOLUCIÓN

* **Creación del diseño del Frontend:** Para la creación del frontend se utilizó REACT, pero este requiere de JavaScript, CSS y HTML para estructurar la página web, así que para resolver este problema y facilitar el proceso de la creación del programa, se usó bootstrap, el cual prestó velocidad y agilidad al trabajo de diseño en gran medida.
* **Validación de la contraseña:** Para esta aplicación web, era necesario que se validara que la contraseña tuviera al menos 1 letra mayúscula, 1 letra minúscula, 1 número, 1 carácter especial y que tuviera por lo menos 7 caracteres. Para solucionar esto, lo que se llevaron a cabo fueron variables que contaban cada uno de los tipos de caracteres de la contraseña, y, si estos poseían más de uno entonces, se podría usar la contraseña.
* **Relleno obligatorio de los campos:** Para los campos de texto, algunos eran opcionales, pero otros no, por lo que para que se rellenaran los campos de texto obligatoriamente se usó una función de bootstrap que se llama “requered”, la cual solicitaba que se ingresara algún dato dentro del campo, sino no se podría pulsar el botón, apareciendo un mensaje sobre el campo de texto que dictamina que se ingrese algo.
* **Manejo de Usuarios y publicaciones:** Para el manejo de los usuarios se usó como “servidor” un arreglo dentro del backend, el cual permitía que se fueran ingresando objetos de tipo Json que representaban los usuarios. De la misma manera se trabajó con las publicaciones, únicamente variando la estructura del Json, dado a que son objetos distintos.
* **Manejo del administrador:** Así como hay usuarios, también debía haber un administrador, el cual tendría opciones especiales al ingresar a su modulo. Para distinguir entre el usuario administrador y el usuario común y corriente, se usó únicamente un condicional, el cual, al reconocer que el carné era el del administrador, puesto a que este carnet y contraseña ya estaban previamente establecidos, redirigiría el usuario a la página a la que debería de ir.
* **Tabla de usuarios y publicaciones:** Para la realización de estas tablas, las cuales permitirían la visualización de toda la información de cada una de las publicaciones y usuarios, así como su eliminación, simplemente se llamó una función del backend la cual entregaría todo el arreglo que se usaba como “servidor”. Posteriormente, se usó toda la información se este para crear la tabla y acceder a los datos por medio de un índice que proporcionaría al programa el conocimiento de la ubicación de la posición de la información de cada uno de los objetos de tipo Json.
* **Eliminación de usuario y publicación:** Para la realización de una eliminación de una publicación o de un usuario se creó, dentro del backend, una función de tipo “delete”, la cual permitiría que, con el índice que obtendría el programa para identificar la posición de cada una de las publicaciones y usuarios, pueda eliminarlo.
* **Código único de publicación:** Para las publicaciones, era necesario crear un ID para cada una, dado a que se precisaba poder hallar cada una de estas. Por tal motivo se recurrió a la utilización de una variable dentro del backend, la cual aumentaría de uno en uno y se colocaría a cada una de las nuevas publicaciones, de esta manera sería capaz de evitar que dos publicaciones tengan el mismo ID.
* **Pestaña Inicio:** La pestaña inicio es la pestaña en la que se muestran todas las publicaciones que se han creado en la red social, para que se pudiera mostrar en orden de la más reciente a la más antigua simplemente se usó una función del backend que entregaría el arreglo con todos los objetos de tipo de Json que guardan la información de las publicaciones, para que de este modo se obtenga la información de estos, y se pueda presentar de forma reversa para que se muestren en orden de los más nuevos a los más antiguos.
* **Creación de las publicaciones:** Para la creación de las publicaciones se usó una cookie la cual guarda la información del usuario al momento del registro, para que de esta forma la publicación tenga la información del usuario que la creó si es que no se escoge la opción de anónimo, la cual se creó con un condicional que al reconocer que esta seleccionada esa opción, simplemente no coloca la información personal de quien lo público. Ahora para la colocación de la imagen, se insertó un buscador de archivos con el cual seleccionaría la imagen, la cual a posterior se guardaría en base 64 dentro del Json. Por último, para guardar la fecha de la creación de la publicación, se usó la función date() y se le dio el formato deseado.
* **Pestaña de tendencias:** En esta pestaña de muestran las 10 publicaciones con más me gustas dentro de toda la aplicación, y para esto se realizó, dentro del backend, una función, la cual tomaría el arreglo que guardaría las publicaciones, las ordenaría y entregaría al frontend únicamente las 10 publicaciones con más me gustas para que se muestren.
* **Pestaña para editar perfil de usuario:** Para editar los datos del usuario se creó una pestaña, muy similar a la pantalla de registro, la cual solicitaría los datos del usuario para que sean actualizados, todos menos el carné dado a que ese se usa como identificador. Luego se guarda dentro del arreglo dentro del backend con la posición que se obtendría por medio del identificador que es el carné.

# LÓGICA DEL PROGRAMA