1. **INTRODUCCIÓN:** El presente documento de Especificación de Requerimientos de Software (ERS) para el sistema ERP Tributario tiene como objetivo y/o propósito detallar, mediante una estructura formal, los requerimientos funcionales y no funcionales que guiarán el desarrollo del producto de software. Este documento servirá como base para el diseño, construcción, validación y entrega del sistema, incluyendo diagramas, casos de uso y descripciones técnicas necesarias para su correcta implementación.
   1. **PROPÓSITO:** Es describir de forma clara, concisa, específica, detallada e inequívoca los requerimientos del software ERP TRIBUTARIO NEIVA, diseñado como un MVP que permita automatizar procesos tributarios dentro de la Alcaldía de Neiva, garantizando eficiencia, transparencia y facilidad de acceso tanto para la administración pública como para los contribuyentes.
   2. **ÁMBITO DEL SISTEMA:** El sistema **ERP TRIBUTARIO NEIVA** será desarrollado como un **MVP (Producto Mínimo Viable)** enfocado en su funcionamiento en dispositivos **móviles** y **plataformas web**. La solución será accesible desde navegadores modernos y dispositivos móviles, permitiendo a los usuarios interactuar con el sistema desde cualquier entorno compatible.

El desarrollo contempla:

* **Aplicación web híbrida** construida con **Ionic + React**, lo cual facilita su ejecución tanto en navegadores como en dispositivos móviles Android o iOS.
* **Backend** implementado en **Spring Boot**, encargado de gestionar la lógica de negocio, el procesamiento de datos tributarios y la conexión con la base de datos.
* **Base de Datos MySQL**, utilizada para almacenar información estructurada sobre contribuyentes, obligaciones tributarias, facturas y reportes.

Este MVP tendrá un enfoque académico, con funcionalidades diseñadas para simular el comportamiento de un sistema real, sin necesidad de conexión permanente a internet, pero con capacidad de sincronización si se requiere.

* 1. **DEFINICIONES, ACRÓNIMOS Y ABREVIATURAS:** Describir el glosario utilizado en todo el documento, para que este sea entendible en cualquier área, ejemplo:

|  |  |
| --- | --- |
| SGBD | Sistema gestor de Base de datos. |
| IEEE | Institute of Electrical and Electronics Engineers |
| UML | Lenguaje unificado para modelamiento de sistemas |
| JAVASCRIPT | JavaScript es un lenguaje de programación interpretado, dialecto del estándar ECMAScript. |

* 1. **REFERENCIAS:** Se listan a continuación otros documentos a los que se hace referencia desde éste**:**

|  |  |  |
| --- | --- | --- |
| **# TÍTULO** | **NÚMERO** | **FECHA** |
| 1 IEEE Guide for Software Requirements Specification IEEE | Std 830-84 | 1994 |
| 2 OMG Unified Modeling Language Specification Version 1.4 | formal/2001-09-67 | 2001 |
| 3 C4 model Simon Brown | C4 | 2011 |

1. **DESCRIPCIÓN GENERAL:** El sistema **ERP TRIBUTARIO NEIVA** es una solución tecnológica diseñada para optimizar y automatizar los procesos de gestión tributaria local de la Alcaldía de Neiva. Actualmente, la administración tributaria municipal enfrenta retos como la generación manual de facturas, el seguimiento ineficiente de pagos y la falta de un sistema digital centralizado que permita la gestión efectiva de contribuyentes y obligaciones fiscales.

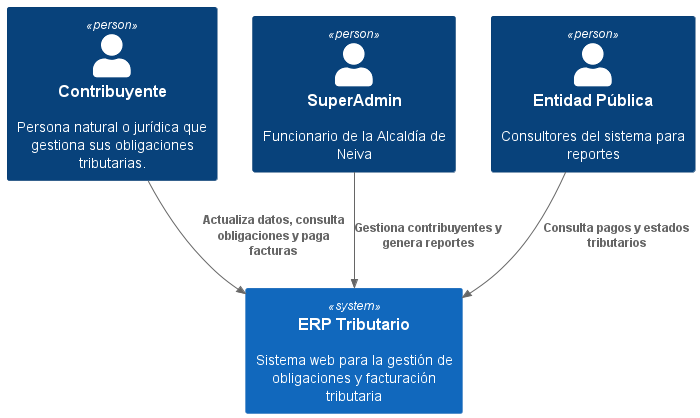
Este proyecto propone el desarrollo de un **MVP (Producto Mínimo Viable)** accesible desde dispositivos **móviles** y plataformas **web**, permitiendo a los distintos actores (superadministradores, entidades públicas y contribuyentes) interactuar con el sistema desde entornos multiplataforma. El sistema facilitará el registro y clasificación de contribuyentes, la gestión de obligaciones tributarias, la emisión de facturas electrónicas simplificadas, la consulta de estados de pago, la generación de reportes y la autogestión de información por parte de los usuarios.

El enfoque del proyecto está orientado a ofrecer una herramienta intuitiva, segura y funcional, construida con tecnologías modernas como **Ionic React** para el frontend, **Spring Boot** para el backend y **MySQL** como sistema de gestión de bases de datos.

* 1. **PERSPECTIVA DEL PRODUCTO:** Este producto de software tiene como propósito principal automatizar la gestión tributaria de la Alcaldía de Neiva, eliminando procesos manuales y facilitando el acceso, actualización y consulta de la información tanto para la administración pública como para los contribuyentes.

Se espera que el sistema permita:

* Registrar contribuyentes (naturales y jurídicos).
* Asignar y gestionar obligaciones tributarias según el perfil del contribuyente.
* Generar facturas electrónicas simplificadas.
* Permitir autogestión por parte de los contribuyentes.
* Visualizar el estado de pagos.
* Emitir reportes básicos sobre recaudos y morosidad.
  1. **FUNCIONES DEL PRODUCTO:** El sistema **ERP TRIBUTARIO NEIVA**, basado en el Modelo C4, se estructura en tres niveles: **contexto**, **contenedor** y **componente**. A nivel de contexto, se describe cómo interactúan los actores principales (como contribuyentes y funcionarios públicos) con el sistema desde plataformas móviles y web. En el nivel de contenedor, se detallan los principales elementos tecnológicos, como la aplicación móvil/web, el backend con servicios REST y la base de datos. En el nivel de componente, se identifican los módulos funcionales del sistema, como la gestión de usuarios, la generación de facturas, la consulta de obligaciones tributarias y la visualización de reportes.
     1. **Diagrama de contexto**



***Figura 1: Diagrama de contexto***

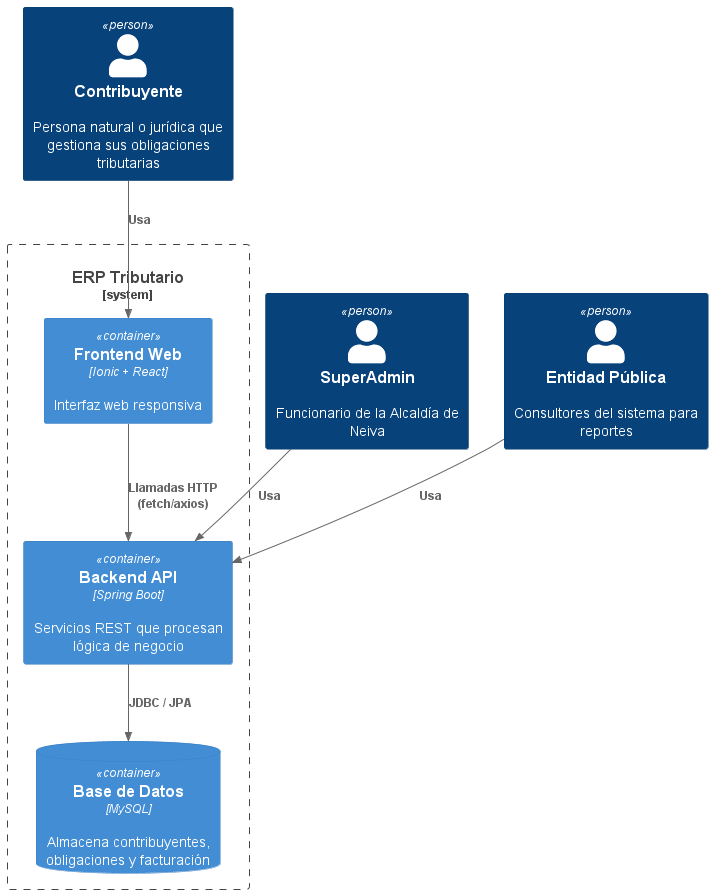
Este diagrama de contexto muestra una visión general del sistema ERP Tributario, representando los actores externos que interactúan con él:

Contribuyente: Persona natural o jurídica que utiliza el sistema para consultar obligaciones tributarias, actualizar datos personales y realizar pagos.

SuperAdmin: Funcionario de la Alcaldía que administra la información del sistema, incluyendo la gestión de contribuyentes y la generación de reportes.

Entidad Pública: Organismos o entidades externas que consultan el sistema para verificar pagos y el estado tributario de los contribuyentes.

Cada actor interactúa directamente con el sistema ERP, destacando las principales funcionalidades según el tipo de usuario.

* + 1. **Diagramas de contenedores**

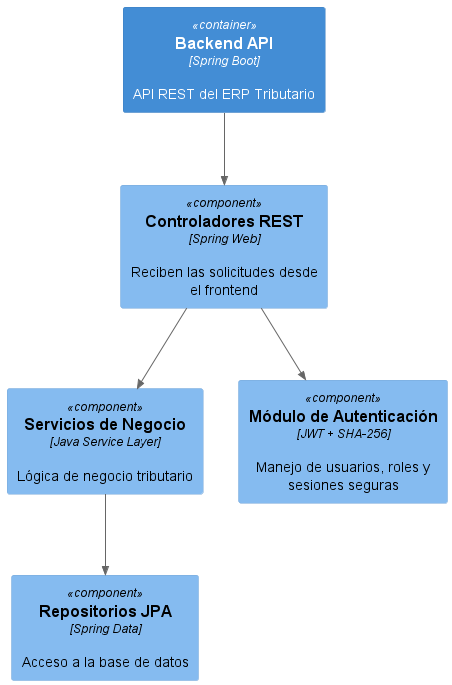
***Figura 2: Diagrama de contenedores***

Un diagrama de contenedor descompone el sistema ERP Tributario en sus principales contenedores de software y muestra cómo interactúan entre sí y con los usuarios:

Actores Externos:

* Contribuyente: Usa la interfaz web para consultar y gestionar sus obligaciones tributarias.
* SuperAdmin: Accede directamente al backend para tareas de gestión y administración.
* Entidad Pública: Consulta datos tributarios a través del backend.

Contenedores del Sistema:

* Frontend Web (Ionic + React): Interfaz web responsiva usada por los contribuyentes. Realiza llamadas HTTP al backend (usando fetch o axios).
* Backend API (Spring Boot): Contiene la lógica de negocio. Expone servicios REST para manejar las operaciones del sistema.
* Base de Datos (MySQL): Almacena información relacionada con contribuyentes, sus obligaciones y facturación. Es accedida desde el backend mediante JDBC o JPA.
  + 1. **Diagrama de componentes**

***Figura 3: Diagrama de componentes***

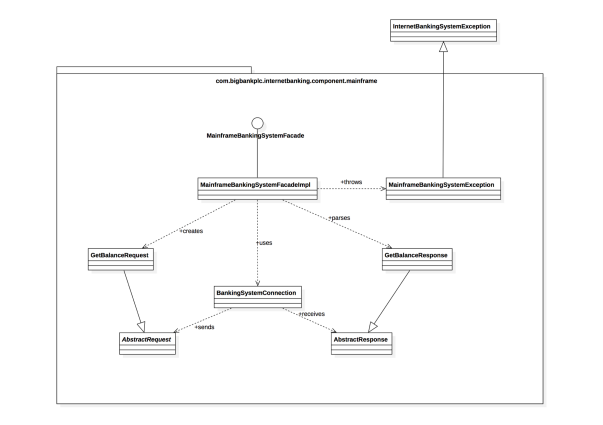
Un diagrama de componentes detalla los componentes internos del contenedor Backend API desarrollado en Spring Boot, mostrando cómo se organiza y se comunica cada módulo:

Componentes:

* Controladores REST (Spring Web): Reciben las solicitudes del frontend (clientes) y actúan como la puerta de entrada al backend.
* Servicios de Negocio: Contienen la lógica de negocio tributario, como validaciones, cálculos y reglas.
* Repositorios JPA (Spring Data): Proveen el acceso a la base de datos mediante operaciones CRUD, utilizando JPA.
* Módulo de Autenticación (JWT): Encargado del manejo de usuarios, control de roles y sesiones seguras usando tokens JWT.

Flujo de interacción:

* El Backend API se conecta con los Controladores REST.
* Los controladores interactúan con los Servicios de Negocio.
* Los servicios, a su vez, acceden a los Repositorios JPA para obtener o modificar datos.
* Los controladores también se comunican con el Módulo de Autenticación para validar sesiones y permisos.
  + 1. **Diagrama de clases**



***Figura 4: Diagrama de clases***

Se puede usar un diagrama de código (por ejemplo, clase UML) para ampliar un componente individual y mostrar cómo se implementa ese componente.

Es decir que puede haber n cantidad de diagramas por componentes dependiendo del modelo de negocio y por cada diagrama de clases debe hacer un Caso de uso.

* + - 1. **HU**

En la HU, es la descripción de una acción o actividad. Una HU se usa en la descripción de las actividades que deberá realizar alguien o algo para llevar a cabo algún proceso, ver **Anexo 1- HU.**

* 1. **CARACTERÍSTICAS DE LOS USUARIOS**

En este literal se describe qué conocimientos básicos debe tener el usuario final o cliente para poder usar nuestro sistema, se pueden especificar los roles si es necesario, permisos y demás.

* 1. **RESTRICCIONES**
     1. **Políticas de la empresa:** Son todas las restricciones que salen del modelo de negocio**,** como se observa en el siguiente ejemplo:

Nota: si no presenta escribir “No se contemplan”

* + 1. **Limitaciones del hardware:** son las características mínimas que debe tener el hardware, para poder instalar o correr nuestro software ejemplo: Linux, CPU 16, RAM 128 GB, etc. Si no se presentan escribir “No se contemplan”.
    2. **Operaciones paralelas:** La computación paralela es una forma de cómputo en la que se hace uso de 2 o más procesadores para resolver una tarea. La técnica se basa en el principio según el cual, algunas tareas se pueden dividir en partes más pequeñas que pueden ser resueltas simultáneamente, ejemplo:

Nota: si no presenta escribir “No se contemplan”.

* + 1. **Funciones de auditoría:** Las funciones de auditoría se instauran para controlar y verificar los sistemas informáticos internos, Analizar los riesgos del entorno informático, sistemas operativos, redes y telecomunicaciones, estas funciones suelen estar basadas en comprobaciones del software. si no presenta escribir “No se contemplan”
    2. **Funciones de control:** Las funciones de control permiten tener garantías sobre el software y sus datos, ejemplo:

Nota: si no presenta escribir “No se contemplan”.

* + 1. **Lenguajes de programación:** Son todos los lenguajes de programación, frameworks y/o herramientas tecnológicas usadas en la elaboración del sistema, ejemplo:
* JavaScript
* PHP
* MySQL
* Angular
* TypeScript

Nota: si no presenta escribir “No se contemplan”.

* + 1. **Protocolos de comunicación:** Los protocolos de comunicaciones entre los diferentes nodos de la infraestructura hardware de soporte serán los siguientes, ejemplo:
* HTTPS para conexiones con el servidor web.
* TCP/IP a nivel físico.
* Ethernet 802.3 a nivel eléctrico.
  + 1. **consideraciones de seguridad:** Son todas las consideraciones y atributos de seguridad dentro de nuestro software, deben también estar incluidos en los casos de uso, como por ejemplo:
* Contraseña robusta.
* Logueo de 2 pasos.
* Bloqueos de usuarios por IP.

Nota: si no presenta escribir “No se contemplan”.

* + 1. **Criticalidad de la aplicación:** En este literal se debe instaurar o explicar qué debe hacer la aplicación o el software en caso de que se presente algún problema o falle, ejemplo:
  1. **SUPOSICIONES Y DEPENDENCIAS:** En la mayoría de los casos nuestras aplicaciones pueden estar expuestas para su consumo desde otros software, así mismo, nuestro software puede tener dependencias internas o consumos de otros sistemas, como por ejemplo:
* Nuestro sistema expone el total de usuarios nuevos, para que sirva de estadística para un app de marketing.
* Nuestro producto consume una API orientada a las finanzas para poder realizar las conversiones de moneda.
  1. **REQUERIMIENTOS FUTUROS:** Son todos los requerimientos que no se tendrán en cuenta en esta versión del proyecto o producto.

1. **INTERFACES EXTERNAS:** Son todas las interfaces que nuestro sistema debe proveer o en su defecto usar para la comunicación con el mismo o con software externos, estas interfaces deben ser a nivel de usuario, hardware, software y de comunicación.
   1. **Interfaces de Usuario**
   2. **Interfaces Hardware**
   3. **Interfaces Software**
   4. **Interfaces de Comunicación**
2. **REQUERIMIENTOS DE RENDIMIENTO:** requerimientos orientados al rendimiento y/o performance de nuestra aplicación

* Número de usuarios conectados en simultáneo.
* Número de peticiones al servidor.
* Número de descargas en simultáneo.
* Número de subidas de archivos en simultáneo.

1. **REQUERIMIENTOS ESPECÍFICOS:** En este apartado se presentan los requisitos funcionales que deberán ser satisfechos por el sistema, Todos los requisitos aquí expuestos son esenciales, es decir, no será aceptable un sistema que no satisfaga alguno de los requisitos aquí presentados. **HU**
2. **CRITERIOS DE ACEPTACIÓN:** Son las condiciones que un producto de software debe satisfacer para ser aceptado por un usuario, cliente o stakeholder, en ese orden de ideas se deben agrupar todos los requisitos tanto funcionales como no funcionales expuestos en este documento a manera de check en el **HU**