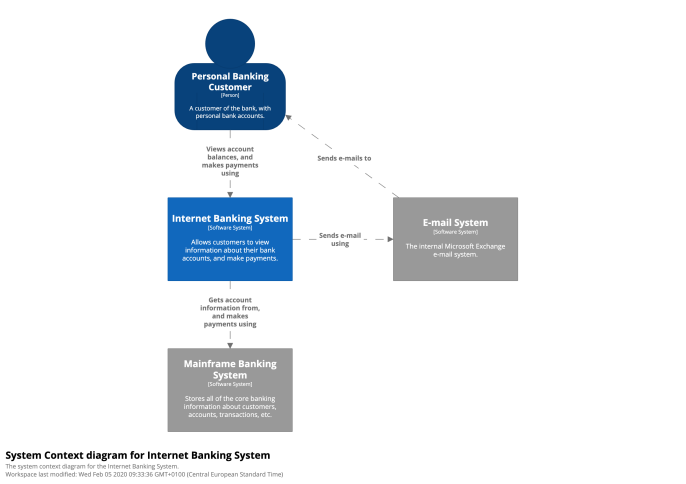
1. **INTRODUCCIÓN:** El presentedocumento de especificación de requerimientos de software para la marca tiene como objetivo y/o propósito detallar con diagramas y casos de usos el desarrollo del producto de software.
   1. **PROPÓSITO:** Es describir de forma clara, concisa, específica, detallada e
   2. inequívoca los requerimientos del software **TuMonedaEnNeiva.**
   3. **ÁMBITO DEL SISTEMA:** Detallar cual es el medio por el cual se va a mostrar o el medio que va a recibir nuestro producto de software ejemplo: Web, Web APP, servidor, móvil, etc.
   4. **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 |

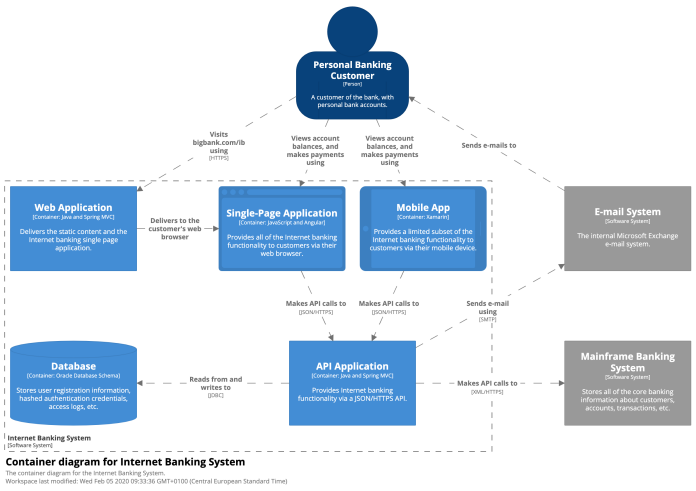
1. **DESCRIPCIÓN GENERAL:** Breve descripción del software a desarrollar
   1. **PERSPECTIVA DEL PRODUCTO:** Contestar la pregunta, ¿Que se espera que haga nuestro producto de software?
   2. **FUNCIONES DEL PRODUCTO:** Descripción del producto en base al Modelo C4 y detallar en casos de uso, como se representa a continuación.
      1. **Diagrama de contexto**



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

Un diagrama de Contexto del Sistema proporciona un punto de partida, mostrando cómo el sistema de software en el alcance encaja en el mundo que lo rodea.

* + 1. **Diagramas de contenedores**

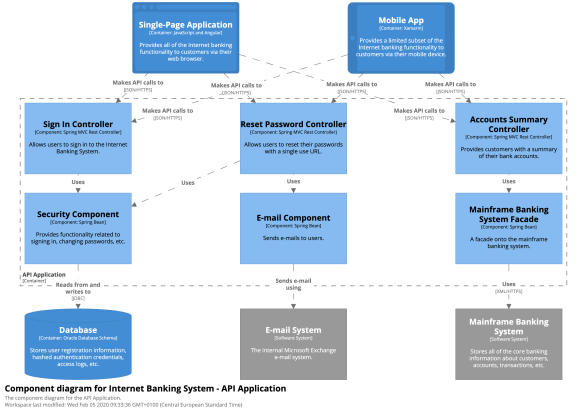


***Figura 2: Diagrama de contenedores***

Un diagrama de contenedor amplía el alcance del sistema de software y muestra los componentes básicos técnicos de alto nivel.

Es decir que pueden haber N cantidad de diagramas por contenedores dependiendo del modelo de negocio.

* + 1. **Diagrama de componentes**

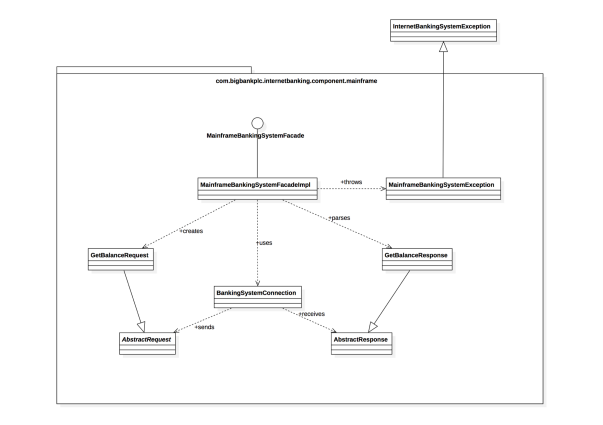


***Figura 3: Diagrama de componentes***

Un diagrama de componentes se acerca a un contenedor individual y muestra los componentes que contiene.

Es decir que pueden haber n cantidad de diagramas por componentes dependiendo del modelo de negocio.

* + 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 pueden 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**