**PROGRAMA INGENIERÍA DE SISTEMAS**

**PROYECTO DE AULA**

**NOMBRE DEL PPROYECTO:** SISTEMA DISTRIBUIDO CON CLIENTE MOVIL PARA EL MANEJO DE TRANSACIONES ENTRE CUENTAS BANCARIAS

**DESCRIPCIÓN DEL PROBLEMA:** Un banco requiere de un sistema que les permita a sus clientes realizar transferencias ente cuentas y tener acceso a su historial de transacciones bancarias desde sus dispositivos móviles. El inconveniente es que es una entidad pequeña con una infraestructura pequeña por lo que piden usar solo una maquina física para implantar el servidor, pero sin tener la base de datos en el mismo lugar, de modo tal, en un futuro, se puedan crear más servidores que se conecten fácilmente a la base de datos.

**OBJETIVO GENERAL:** Construir un sistema distribuido que permita la realización de transferencias entre cuentas bancarias desde dispositivos móviles contemplando las limitaciones de infraestructura de la entidad bancaria.

**OBJETIVOS ESPECÍFICOS:**

* Analizar los requerimientos a satisfacer por parte del sistema distribuido a construir.
* Diseñar los componentes del sistema distribuido, tales como la base de datos, API y la aplicación móvil en base a las necesidades de la entidad bancaria.
* Construir la arquitectura distribuida para realizar transferencias bancarias utilizando los respectivos componentes.
* Realizar pruebas para la funcionalidad del sistema distribuido.

**TEMÁTICAS:**

1. **Cloud Computing**: Es el suministro de servicios informáticos (incluidos servidores, almacenamiento, bases de datos, redes, software, análisis e inteligencia) a través de internet (la “nube”), cuyo objetivo es ofrecer una innovación más rápida, recursos flexibles y economías de escala (Microsoft, 2021).
2. **Bases de datos**: Es una colección organizada de información estructurada, o datos, típicamente almacenados electrónicamente en un sistema de computadora. Una base de datos es usualmente controlada por un sistema de gestión de base de datos (DBMS). En conjunto, los datos y el DBMS, junto con las aplicaciones que están asociados con ellos, se conocen como un sistema de base de datos (Oracle Mexico, 2021).
3. **Servicios web**: Son aplicaciones modulares autocontenidas que puede describir, publicar, localizar e invocar a través de una red. Un escenario de servicios web típico es una aplicación empresarial que solicita un servicio de otra aplicación existente. La solicitud se procesa a través de una dirección web determinada usando mensajes SOAP a través de un transporte HTTP. El servicio recibe la solicitud, la procesa y devuelve una respuesta (IBM, 2021).
4. **Dispositivos móviles**: Son aparatos de tamaño pequeño (generalmente) que cuentan con características tales como capacidades especiales de procesamiento, conexión permanente o intermitente a una red, memoria limitada, diseños específicos para una función principal y versatilidad para el desarrollo de otras funciones. Tanto su posesión como su operación se asocian al uso individual de una persona, quien puede configurarlo según sus criterios y preferencias.
5. **Requerimientos del negocio**: Son especificaciones de alto nivel, establecidas por la organización que perciben necesidades o detectan oportunidades. Incluyen los objetivos del negocio, la visión del producto y el alcance del proyecto. Define el por qué y para qué se construye un sistema (Ferreira, 2015).
6. **Pruebas de seguridad**: Son pruebas en un sistema que evidencian las vulnerabilidades más típicas existentes en las aplicaciones, con el propósito de evitar que cualquier persona pueda aprovecharse de ellas para ingresar a un sistema.

**COMPETENCIAS PARA DESARROLLAR:**

1. Manejo de servicios de cloud computing
2. Desarrollo de bases de datos relacionales
3. Desarrollo de servicios web/APIs
4. Desarrollo de aplicaciones móviles con Android Studio

**CONOCIMIENTOS PREVIOS REQUERIDOS:**

1. Lenguaje SQL
2. Conectividad y red
3. Spring Boot
4. Android Studio
5. Conexión a maquinas remotas

**RECURSOS A NIVEL HARDWARE Y SOFTWARE:**

Para la realización de este proyecto se definen los siguientes recursos, los cuales se emplearán en el diseño, desarrollo y verificación de los componentes del sistema distribuido:

|  |  |  |
| --- | --- | --- |
| **Icono** | **Nombre del recurso** | **Descripción** |
| Cloud Computing, servicios de alojamiento y APIs | Google Cloud | [Google Cloud](https://cloud.google.com/) | Consiste en un conjunto de recursos físicos, como computadoras y unidades de disco duro, y recursos virtuales como VM, que se encuentran en los centros de datos de Google de todo el mundo. |
|  | [MySQL 5.7](https://dev.mysql.com/doc/refman/5.7/en/) | Es un sistema de bases de datos relacional desarrollado bajo licencia dual por Oracle Corporation. |
|  | [MySQL Workbench 8.0](https://dev.mysql.com/doc/workbench/en/) | Es un entorno gráfico de MySQL que permite modelar, generar y gestionar bases de datos. |
|  | [Java 1.8](https://docs.oracle.com/javase/8/docs/api/) | Es el estándar global para desarrollar y distribuir aplicaciones móviles y embebidas, juegos, contenido basado en web y software de empresa. |
|  | [Spring Boot – Spring Tools 4.2](https://docs.spring.io/spring-boot/docs/current/reference/htmlsingle/) | Es una herramienta que facilita el desarrollo de aplicaciones web y microservicios con el Framework Spring, de una manera más fácil y veloz. |
| Eclipse IDE 2021-03 R para Windows - Descargar | [Eclipse IDE for Enterprise Java Developers - 2020-09](https://www.eclipse.org/downloads/download.php?file=/technology/epp/downloads/release/2018-12/R/eclipse-jee-2018-12-R-win32-x86_64.zip) | Es una plataforma de software compuesto por un conjunto de herramientas de programación de código abierto multiplataforma para desarrollar aplicaciones. |
|  | [Android Studio 4.2](https://developer.android.com/studio) | Es el entorno de desarrollo integrado (IDE) oficial para el desarrollo de apps para Android y está basado en IntelliJ IDEA. |
|  | [Retrofit](https://square.github.io/retrofit/) | Librería que funciona como un cliente HTTP seguro para la comunicación con Spring Boot |
|  | [Postman 8.5](https://www.postman.com/downloads/) | Es una plataforma API para construir y utilizar APIs. |
| PC Icon - Free Icons | Equipos personales | Son los equipos que van a servir como:   * Equipos de desarrollo. * Clientes. * Servidores. |
|  | [Github – Github Desktop](https://github.com/lfvelascot/SISTEMAS_DISTRIBUIDOS) | Es un repositorio de proyectos mediante el sistema de control de versiones Git. Se utiliza principalmente para la creación de código fuente de programas de ordenador. |

Ilustración 1 tabla de recursos de software y hardware

**Nota:**  por medio de los nombres de cada uno de los recursos de software se puede tener acceso a documentación y archivos usados.

**DESCRIPCIÓN DE LA PRÁCTICA (PROCEDIMIENTOS, EJERCICIOS Y METODOLOGÍA):**

Para el desarrollo de este proyecto se contemplan las siguientes actividades, procedimientos y metodologías:

* Metodología: se hará uso de la metodología en Cascada propuesta por Winston W. Royce en 1970.
* Modelado de la arquitectura del sistema distribuido: se contempla el diseño del sistema como una actividad de vital importancia ya que, por medio de esta se definen los componentes que tendrá el sistema, la forma en que se van a comunicar los componentes entre si y los recursos de software y hardware que soportaran cada uno de los componentes.
* Definición de recursos de software y hardware a emplear: en esta actividad se definirán las tecnologías que se emplearán en el desarrollo del proyecto, además de identificar la documentación del software a emplear con el fin de acceder de forma rápida a estos en el momento que se requiera.
* Definición de requerimientos funcionales y no funcionales del sistema distribuido: con base a la descripción del problema, se definen los requerimientos a cumplir por parte del sistema distribuido.
* Instalación de componentes de software a emplear: con los recursos definidos, instalan los recursos de software identificados.
* Levantamiento de la instancia de VM: esta actividad contempla la configuración de la instancia de VM en Google Cloud.
* Instalación y configuración de MySQL en la instancia VM: con la instancia de la VM en Google cloud, se deben instalar los recursos requeridos en la instancia para el correcto funcionamiento de MySQL.
* Diseño e implementación de la base de datos: se definirán las tablas con los datos requeridos para la naturaleza del proyecto.
* Configuración de la API en Spring Boot: con la base de datos construida, se levantarán los componentes de la API en base a la misma estructura definida de la base de datos.
* Conexión de la base de datos con la API: esta actividad comprobara que la API pueda acceder correctamente a los datos almacenados en la base de datos.
* Diseño de interfaces graficas de la aplicación: en esta actividad se definirán las pantallas a las cuales el usuario tendrá acceso, teniendo presente los servicios que podrá consumir la API.
* Implementación de conectividad entre la aplicación móvil y la API: con las pantallas diseñadas, se va a implementar la librería que permita la conectividad con la API en Spring Boot.
* Pruebas de funcionamiento de los componentes y del sistema distribuido: se realizarán pruebas individuales del funcionamiento de cada uno de los componentes, al tener todos los componentes funcionando correctamente, se probará el correcto funcionamiento del sistema distribuido.
* Pruebas de seguridad: se realizarán pruebas relacionadas con la seguridad de cada uno de los componentes.
* Mantenimiento y mejora: se realizarán mejoras a los componentes teniendo en cuenta la información obtenida de las pruebas de funcionamiento y seguridad.

Para el desarrollo de este proyecto se implementará la metodología en cascada, la cual es útil en el desarrollo de sistemas, de modo tal se definan los elementos relacionados con los requerimientos, diseño e implementación de cada uno de los componentes, para realizar la verificación de cada uno de los componentes y luego una verificación del sistema distribuido en general. A continuación, se especifica cada una de las fases y los productos o resultados que se esperan de cada fase.

**METODOLOGÍA: METODOLOGIA EN CASCADA**

El modelo o metodología en cascada es la propuesta de un enfoque metodológico que consiste en ordenar de forma lineal las distintas fases que se deben seguir al momento de desarrollar software (Royce, 1970).

1. **Análisis**: Consiste en la fase de análisis primario, que contempla el estado inicial del proyecto, los recursos disponibles, los requerimientos de negocio (objetivos del proyecto, justificaciones y estado final del proyecto) y el tiempo disponible para su desarrollo.

En esta fase se obtendrá como resultado las tablas de requerimientos funcionales y no funcionales del sistema distribuido.

1. **Diseño de componentes**: Consiste en analizar los recursos disponibles del proyecto y determinar cuales se utilizarán teniendo en cuenta los requerimientos de este. Luego, se utilizan dichos recursos para construir los componentes necesarios del sistema.

En esta fase se obtendrán los siguientes productos:

* + Modelo relacional de la base de datos
  + Listado de servicios de la API Rest
  + Diseño de interfaces graficas de la aplicación móvil

1. **Implementación o desarrollo ingenieril**: Consiste en acoplar los componentes para realizar la aplicación funcional del sistema en el contexto del proyecto.

En esta fase se obtendrán los siguientes productos:

* + Instancia de VM con la base de datos MySQL creada y configurada
  + API Rest en Spring Boot configurada
  + Aplicación móvil desarrollada en Android Estudio.

1. **Verificación**: Consiste en realizar las pruebas de funcionamiento sobre el sistema en cuestión, con el fin de detectar y corregir errores, implementar o inhabilitar funciones y optimizar el sistema (de ser posible).

En esta fase se obtendrán los siguientes productos:

* + Pruebas realizadas a la API Rest en Postman
  + Pruebas funcionales de la aplicación móvil

1. **Mantenimiento**: Consiste en verificar el sistema funcional sobre el contexto del proyecto para verificar que funcione correctamente, y de ser necesario, corregir errores y vulnerabilidades.

En esta fase se obtendrán los siguientes productos:

* + Pruebas de seguridad y conectividad a los componentes
  + Revisión de componentes

Gráfico, Diagrama, Gráfico de cajas y bigotes

Descripción generada automáticamente

Ilustración 2 estructura de la metodología en cascada

En la ilustración 2 se puede ver como se estructura esta metodología, de modo tal, al momento de finalizar todas las fases, se pueda realizar el mantenimiento del sistema distribuido y se vuelva a alguna de las fases anteriores para realizar las correcciones necesarias. Si se vuelve a una fase, esto demandara que se realicen correcciones en las fases que la proceden, esto con el fin de arreglar los errores ocasionados en la fase en la que se detectó el error o fallo inicialmente.

**IMPLEMENTACION DE LA METODOLOGIA**

**FASE 1: ANÁLISIS**

A continuación de demostraran los productos y evidencia de actividades desempeñadas en cada una de las fases especificadas por la metodología que se va a emplear para el desarrollo de este proyecto.

***Definición de requerimientos***

A continuación, se definen los requerimientos funcionales y no funcionales del sistema, contemplando las necesidades expuestas por el cliente en las cuales se definen las actividades a realizar por el usuario, se da un vistazo a los componentes que se deben implementar y se definen los ítems a cumplir con el desarrollo del proyecto.

|  |  |  |
| --- | --- | --- |
| **REQUERIMIENTOS FUNCIONALES BANCAPP** | | |
| Cod | Nombre | Descripción |
| RW001 | Ingreso a la aplicación móvil | La aplicación debe permitirles a los usuarios ingresar a ella mediante las credenciales correspondientes, y en base a ellas determinar si ingresan como clientes o como administrador. |
| RW002 | Transferencia entre cuentas | La aplicación debe permitirles a los clientes del banco realizar trasferencias entre cuentas de los otros clientes del banco y notificar si se ha presentado algún inconveniente en el proceso. |
| RW003 | Ver historial de transacciones | La aplicación debe permitirles a los clientes del banco revisar los movimientos y transacciones (consignación, retiro, transferencia) realizados. |
| RW004 | Cambio de datos personales y credenciales de usuario | La aplicación debe permitirles a los usuarios cambiar cierta información personal y sus credenciales de acceso, esto último como medida de seguridad. |

Ilustración 3 tabla de requerimientos funcionales

|  |  |  |
| --- | --- | --- |
| **REQUERIMIENTOS NO FUNCIONALES BANCAPP** | | |
| Cod | Nombre | Descripción |
| RW00X | Privacidad de datos empresariales | Este requerimiento consta de asegurar el acceso a la información asociada a cada uno de los componentes, es decir, permitir a los clientes de la entidad bancaria acceder de forma segura a la información de sus transacciones y la ejecución de estas. |
| RW007 | Almacenamiento en base de datos | Se requiere que la aplicación móvil posea una base de datos la cual asegure el correcto indexado de los datos, los cuales serán utilizados por ella. |
| RW008 | Acceso remoto a la información | Se requiere que se tenga acceso a las funcionalidades de forma remota, siempre y cuando, los usuarios tengan acceso a Internet. |
| RW09 | Interfaz de usuario simple y fluida | Debido a que se requiere una gestión ágil para la información, no se requiere dinamismo en la presentación de la información para cada uno de los usuarios, ya que lo importante aquí es tener acceso rápido a las funcionalidades que requiera cada usuario. |
| RW010 | Seguridad de la aplicación móvil | Debido al manejo de información personal, como lo son los datos de cada uno de los usuarios y de los empleados de las empresas, se requiere que la aplicación móvil cuente con seguridad frente a vulnerabilidades que afecten su funcionamiento |
| RW011 | Seguridad de la base de datos | Se requiere algún sistema de seguridad para la base de datos y los datos que esta contiene debido a la importancia de la información que se manejara. |
| RW012 | Limitación en infraestructura | Los componentes de base de datos y servidor deben encontrase distribuidos, pero solo se cuenta con un equipo |
| RW013 | Panel de ayuda de aplicación móvil | Para que los usuarios de la aplicación móvil puedan resolver dudas de forma inmediata frente a su uso, la aplicación requiere tener un espacio con ítems relacionados al manejo de cada una de las funcionalidades de la aplicación. |

Ilustración 4 tabla de requerimientos no funcionales

**FASE 2: DISEÑO**

Contemplando los objetivos y la descripción del problema, se definió el siguiente diseño del sistema distribuido, en el cual se definen los componentes y herramientas de software requeridas para el desarrollo, implementación y puesta en marcha de todo el sistema, en la siguiente imagen se podrán ver la forma en que el sistema se encontrara distribuido y la forma en que cada uno de los componentes se conectara y comunicaran entre sí, de modo tal se definirán también los protocolos y actividades de comunicación:

Diagrama

Descripción generada automáticamente

Ilustración 5 Diseño de la arquitectura distribuida

En la ilustración 5 se detalla un sistema distribuido que utiliza una arquitectura cliente-servidor, en la cual se tendrá del lado del servidor los componentes de la base de datos y la API que hará uso de esta, esta API será la que tendrá comunicación con el cliente que en este caso será una Aplicación móvil. Esta arquitectura y las tecnologías usadas garantizan que se podrá en un futuro incluir un nuevo cliente como lo sería un cliente web, y escalar en servicios la API existente.

* **Base de datos**: contemplando las limitaciones de infraestructura y la necesidad de tener los componentes de base de datos y servidor o API separados, teniendo en cuenta que solo existe un equipo físico, por esto se define que la base de datos MySQL se encontrara en una instancia de VM (Virtual Machine) en Google Cloud. Por la naturaleza de este ejercicio, se mantendrá activa la instancia de VM solo en los momentos en que se requiera tener acceso a la base de datos, y se accederá a esta por medio de protocolos TCP/IP y SSH, el primero se usara para el acceso por parte de la API y el segundo nos permitirá la manipulación de la instancia de forma remota.
* **API**: contemplando las limitaciones expuestas y la necesidad de un sistema distribuido, se construirá una API con Spring Boot, Framework para Java el cual permite la creación y manipulación de APIs de forma rápida. La API será probada con Postman con el fin de probar el correcto funcionamiento de los servicios que esta poseerá.
* **Aplicación móvil**: como cliente se tendrá una aplicación móvil desarrollada por medio de Android Studio, la cual hace manejo de lenguaje Java y XML para la manipulación de la interfaz. Para el consumo de la API Rest se hará uso de la librería Retrofit.
* **Usuario**: contemplando las necesidades del usuario, se especifican las acciones básicas del usuario para implementarlas en la aplicación móvil y definir los servicios en la API Rest.

**FASE 3: PROCEDIMIENTO O DESARROLLO INGENIERÍL**

**MySQL Cloud**

El despliegue de una base de datos cuando no se tiene una infraestructura física que la soporte y teniendo como requisito que esta no se encuentre en el mismo servidor con otros componentes del sistema puede parecer una tarea complicada, pero en la actualidad contamos con las tecnologías cloud las cuales buscan prestar como servicios infraestructura, software y plataformas en las cuales se puedan soportar componentes de software. En Google cloud se nos presentan dos alternativas en las cuales se podría desplegar una base de datos MySQL:

1. Instancia de VM: es una maquina virtual a la cual se tiene acceso de forma remota y en la cual se pueden instalar cualquier componente de software requerido. En una solución mucho más general, pero cuenta con la complejidad de su configuración.
2. Instancia de Cloud SQL: es una maquina virtual pero dedicada únicamente para un sistema de gestión de bases de datos, saltándose toda la configuración de la instancia y enfocándose únicamente en el almacenamiento de los datos.

Para este proyecto se desplegará una instancia de Cloud SQL por medio de los siguientes pasos:

1. Tener una cuenta de Google por medio de la cual pueda acceder a los servicios de [Google Cloud](https://cloud.google.com/), además de contar con créditos que le permitan usar los servicios. (puede usar la prueba gratuita o hacer un pago con su tarjeta de crédito/debito)
2. Ya con la suscripción a los servicios de Cloud, diríjase al siguiente enlace <https://console.cloud.google.com/sql/instances?project=elite-bird-311413> en el cual podrá iniciar el proceso de creación de la instancia:

Interfaz de usuario gráfica

Descripción generada automáticamente

1. De clic en el botón Crear Instancia:

Interfaz de usuario gráfica

Descripción generada automáticamente

1. En la siguiente página se mostrarán las bases SQL disponibles en Google Cloud, en este caso, elegiremos a MySQL:

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico, Teams

Descripción generada automáticamente

1. A continuación, se realizará la configuración de la instancia:
   * ID de instancia: para este ejercicio la instancia se llamará **banco-db-sd8**
   * Contraseña: esta contraseña será por medio de la cual accederemos al usuario root, se aconseja que tenga un nivel de dificultad alto.
   * Revise por favor que la casilla “Sin contraseña” este desmarcada
   * Versión de la base de datos: para este ejercicio se empleará a MySQL en su versión **5.7**
   * Región: Google en la actualidad presenta la región en la cual se puede desplegar las instancias en sistemas de bajo consumo, por lo que se aconseja emplear una región cercana y que cuente con esta característica, en nuestro caso empleamos la región **southamerica-east1 (São Paulo)**
   * Disponibilidad zonal: debido a que este es un ejercicio práctico, se emplea la **Zona única**, la cual es de bajo costo y no es para entornos de producción.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

* + Dentro de la misma página, al final se encuentra la opción de personalizar la instancia, debido a la naturaleza del ejercicio y lo simple de este, se va a configurar la instancia, para esto dar clic en **Mostrar opciones de configuración**

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

* + Tipo de máquina: para un bajo consumo, se empleará una máquina de tipo **Estándar**, con **1 CPU virtuales, 3.75 GB.** La configuración de tipo de maquina va acorde a las necesidades del proyecto.
  + Tipo de almacenamiento: para tener un buen rendimiento de la instancia, se usará un almacenamiento **SSD.**
  + Capacidad de almacenamiento: para este ejercicio solo se empleará **10GB** de almacenamiento.
  + Desactivar la opción de **Habilitar los aumentos automáticos de almacenamiento,** esto con el fin de evitar que Google aumente el espacio sin nuestra autorización.

Tabla

Descripción generada automáticamente

Las demás opciones se dejarán tal cual, ya que estas controlan el mantenimiento, copias de seguridad y actualizaciones requeridas por la instancia, todo esto administrado por Google. Al finalizar dar clic en **Crear** **instancia**.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

1. Al finalizar el proceso de configuración y creación de la instancia, nos encontraremos con el dashboard que nos permitirá tener control de la instancia. El proceso de creación de la instancia puede demorar algunos minutos:

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

1. Para poder conectarnos a la instancia, debemos conocer nuestra IP publica, por lo que nos dirigimos a la siguiente pagina <https://www.ionos.es/tools/direccion-ip> en la cual podemos ver la IP de cada equipo que se conectara a la instancia

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

1. Con la IP de los equipos que se conectaran a la instancia, nos dirigimos al apartado de Conexiones de la instancia y configuraremos la conexión para las maquinas

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

NOTA: se puede definir la conexión desde cualquier equipo mediante la configuración de la red 0.0.0.0/0

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

1. Para verificar la conexión desde el equipo de la IP ingresada, emplearemos a Workbench, en donde crearemos una conexión, en donde ingresaremos la IP publica de la instancia y la contraseña de root que ingresamos previamente

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Para comprobar la conexión, damos clic en **Test connection** y si recibiremos el siguiente mensaje que nos notificara que la instancia está funcionando correctamente y que podremos acceder a esta desde nuestro equipo.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Con todos estos pasos, ya podremos crear la base de datos, la cual tiene el siguiente modelo relacional realizado con Workbench, en el cual se contempla las necesidades expuestas previamente, definiendo las tablas de usuarios, cuentas de la aplicación y bancarias, un registro de actividades y de las transacciones realizadas por parte de los usuarios.

Diagrama

Descripción generada automáticamente

Con este modelo, se puede crear el script DDL para la base de datos, por medio de las opciones Database / Forward Engineer:

-- -----------------------------------------------------

-- Schema bancapp\_db

-- -----------------------------------------------------

CREATE SCHEMA IF NOT EXISTS `bancapp\_db` DEFAULT CHARACTER SET utf8 ;

USE `bancapp\_db` ;

-- -----------------------------------------------------

-- Table `bancapp\_db`.`user`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `bancapp\_db`.`user` (

`num\_doc` VARCHAR(100) NOT NULL,

`nombre` VARCHAR(45) NOT NULL,

`apellido` VARCHAR(45) NOT NULL,

`fecha\_nam` DATE NOT NULL,

`correo\_electronico` VARCHAR(100) NOT NULL,

`telefono` VARCHAR(100) NOT NULL,

PRIMARY KEY (`num\_doc`))

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `bancapp\_db`.`rol`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `bancapp\_db`.`rol` (

`nombre\_rom` VARCHAR(45) NOT NULL,

`descrip` VARCHAR(45) NOT NULL,

PRIMARY KEY (`nombre\_rom`))

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `bancapp\_db`.`estado\_cuenta`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `bancapp\_db`.`estado\_cuenta` (

`nombre` VARCHAR(45) NOT NULL,

`descrip` VARCHAR(45) NOT NULL,

PRIMARY KEY (`nombre`))

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `bancapp\_db`.`tipo\_cuenta`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `bancapp\_db`.`tipo\_cuenta` (

`nombre` VARCHAR(45) NOT NULL,

`descrip` VARCHAR(45) NOT NULL,

PRIMARY KEY (`nombre`))

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `bancapp\_db`.`cuenta\_bancaria`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `bancapp\_db`.`cuenta\_bancaria` (

`num` VARCHAR(100) NOT NULL,

`clave` VARCHAR(100) NOT NULL,

`fecha\_inicio` VARCHAR(45) NOT NULL,

`fecha\_fin` VARCHAR(45) NULL,

`saldo` DOUBLE NOT NULL,

`estado\_cuenta` VARCHAR(45) NOT NULL,

`tipo\_cuenta` VARCHAR(45) NOT NULL,

`usuario` VARCHAR(100) NOT NULL,

PRIMARY KEY (`num`),

INDEX `fk\_cuenta\_bancaria\_estado\_cuenta1\_idx` (`estado\_cuenta` ASC) ,

INDEX `fk\_cuenta\_bancaria\_tipo\_cuenta1\_idx` (`tipo\_cuenta` ASC) ,

INDEX `fk\_cuenta\_bancaria\_user1\_idx` (`usuario` ASC) ,

UNIQUE INDEX `usuario\_UNIQUE` (`usuario` ASC) ,

CONSTRAINT `fk\_cuenta\_bancaria\_estado\_cuenta1`

FOREIGN KEY (`estado\_cuenta`)

REFERENCES `bancapp\_db`.`estado\_cuenta` (`nombre`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk\_cuenta\_bancaria\_tipo\_cuenta1`

FOREIGN KEY (`tipo\_cuenta`)

REFERENCES `bancapp\_db`.`tipo\_cuenta` (`nombre`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk\_cuenta\_bancaria\_user1`

FOREIGN KEY (`usuario`)

REFERENCES `bancapp\_db`.`user` (`num\_doc`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `bancapp\_db`.`cuenta\_app`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `bancapp\_db`.`cuenta\_app` (

`user\_num\_doc` VARCHAR(100) NOT NULL,

`contrasena` VARCHAR(100) NOT NULL,

`ultimo\_ingreso` DATETIME NOT NULL,

`estado\_cuenta\_nombre` VARCHAR(45) NOT NULL,

`rol\_nombre\_rom` VARCHAR(45) NOT NULL,

INDEX `fk\_cuenta\_app\_estado\_cuenta1\_idx` (`estado\_cuenta\_nombre` ASC) ,

INDEX `fk\_cuenta\_app\_rol1\_idx` (`rol\_nombre\_rom` ASC) ,

INDEX `fk\_cuenta\_app\_user1\_idx` (`user\_num\_doc` ASC) ,

PRIMARY KEY (`user\_num\_doc`),

CONSTRAINT `fk\_cuenta\_app\_estado\_cuenta1`

FOREIGN KEY (`estado\_cuenta\_nombre`)

REFERENCES `bancapp\_db`.`estado\_cuenta` (`nombre`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk\_cuenta\_app\_rol1`

FOREIGN KEY (`rol\_nombre\_rom`)

REFERENCES `bancapp\_db`.`rol` (`nombre\_rom`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk\_cuenta\_app\_user1`

FOREIGN KEY (`user\_num\_doc`)

REFERENCES `bancapp\_db`.`user` (`num\_doc`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `bancapp\_db`.`transferencia`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `bancapp\_db`.`transferencia` (

`id\_trans` VARCHAR(100) NOT NULL,

`valor` DOUBLE NOT NULL,

`cuenta\_origen` VARCHAR(100) NOT NULL,

`cuenta\_destino` VARCHAR(100) NOT NULL,

`fecha` DATETIME NOT NULL,

PRIMARY KEY (`id\_trans`),

INDEX `fk\_transferencia\_cuenta\_bancaria\_idx` (`cuenta\_origen` ASC) ,

INDEX `fk\_transferencia\_cuenta\_bancaria1\_idx` (`cuenta\_destino` ASC) ,

CONSTRAINT `fk\_transferencia\_cuenta\_bancaria`

FOREIGN KEY (`cuenta\_origen`)

REFERENCES `bancapp\_db`.`cuenta\_bancaria` (`num`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk\_transferencia\_cuenta\_bancaria1`

FOREIGN KEY (`cuenta\_destino`)

REFERENCES `bancapp\_db`.`cuenta\_bancaria` (`num`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `bancapp\_db`.`accion`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `bancapp\_db`.`accion` (

`nombre` INT NOT NULL,

`descrip` VARCHAR(45) NOT NULL,

PRIMARY KEY (`nombre`))

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `bancapp\_db`.`log`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `bancapp\_db`.`log` (

`fecha` INT NOT NULL,

`descrip` VARCHAR(45) NOT NULL,

`cuenta` VARCHAR(100) NOT NULL,

`accion` INT NOT NULL,

PRIMARY KEY (`fecha`),

INDEX `fk\_log\_cuenta\_app1\_idx` (`cuenta` ASC) ,

INDEX `fk\_log\_accion1\_idx` (`accion` ASC) ,

CONSTRAINT `fk\_log\_cuenta\_app1`

FOREIGN KEY (`cuenta`)

REFERENCES `bancapp\_db`.`cuenta\_app` (`contrasena`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk\_log\_accion1`

FOREIGN KEY (`accion`)

REFERENCES `bancapp\_db`.`accion` (`nombre`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `bancapp\_db`.`rol\_accion`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `bancapp\_db`.`rol\_accion` (

`accion` INT NOT NULL,

`rol` VARCHAR(45) NOT NULL,

INDEX `fk\_rol\_accion\_accion1\_idx` (`accion` ASC) ,

INDEX `fk\_rol\_accion\_rol1\_idx` (`rol` ASC) ,

CONSTRAINT `fk\_rol\_accion\_accion1`

FOREIGN KEY (`accion`)

REFERENCES `bancapp\_db`.`accion` (`nombre`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk\_rol\_accion\_rol1`

FOREIGN KEY (`rol`)

REFERENCES `bancapp\_db`.`rol` (`nombre\_rom`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

Con la base de datos montada, se crearán algunos registros básicos de las tablas con dependencia 0:

INSERT INTO `bancapp\_db`.`rol` (`nombre\_rom`,`descrip`) VALUES ("ADMINISTRADOR","");

INSERT INTO `bancapp\_db`.`rol` (`nombre\_rom`,`descrip`) VALUES ("USUARIO","");

INSERT INTO `bancapp\_db`.`estado\_cuenta` (`nombre`,`descrip`) VALUES ("ACTIVA","");

INSERT INTO `bancapp\_db`.`estado\_cuenta` (`nombre`,`descrip`) VALUES ("BLOQUEADA","");

INSERT INTO `bancapp\_db`.`estado\_cuenta` (`nombre`,`descrip`) VALUES ("SUSPENDIDA","");

INSERT INTO `bancapp\_db`.`accion`(`nombre`,`descrip`) VALUES("Login exitoso","");

INSERT INTO `bancapp\_db`.`accion`(`nombre`,`descrip`) VALUES("Login fallido","");

INSERT INTO `bancapp\_db`.`accion`(`nombre`,`descrip`) VALUES("Logout exitoso","");

INSERT INTO `bancapp\_db`.`accion`(`nombre`,`descrip`) VALUES("Logout fallido","");

INSERT INTO `bancapp\_db`.`accion`(`nombre`,`descrip`) VALUES("Ver estado Cuenta exitoso","");

INSERT INTO `bancapp\_db`.`accion`(`nombre`,`descrip`) VALUES("Ver estado Cuenta fallido","");

INSERT INTO `bancapp\_db`.`accion`(`nombre`,`descrip`) VALUES("Transferencia exitosa","");

INSERT INTO `bancapp\_db`.`accion`(`nombre`,`descrip`) VALUES("Transferencia fallida","");

INSERT INTO `bancapp\_db`.`accion`(`nombre`,`descrip`) VALUES("Ver historial transacciones exitoso","");

INSERT INTO `bancapp\_db`.`accion`(`nombre`,`descrip`) VALUES("Ver historial transacciones fallido","");

**Nota:** se aconseja tener la base de datos desplegada en su equipo local, esto con el fin de no tener siempre la instancia de Cloud SQL activa y poder realizar los demás componentes basados en la base de datos sin depender directamente en el desarrollo de la instancia.

Para apagar la instancia, en el dashboard principal dar clic en el botón “” en la parte superior:

Interfaz de usuario gráfica

Descripción generada automáticamente con confianza baja

**API Rest Spring Boot**

**Aplicación móvil**

**FASE 4: VERIFICACION**

**FASE 5: MANTENIMIENTO**

**BIBLIOGRAFÍA:**

Ferreira, H. (2015). *Requerimientos del Negocio-Modelos*. http://cs.uns.edu.ar/materias/rs/downloads/PRACTICA/Clase Practica3 Requerimientos del Negocio Modelos RdeS 2015.pdf

IBM. (2021, January 25). *Servicios web*. Documentación de IBM. https://www.ibm.com/docs/es/was/9.0.5?topic=services-web

Microsoft. (2021, January 12). *What Is Cloud Computing? A Beginner’s Guide | Microsoft Azure*. Microsoft Azure. https://azure.microsoft.com/en-us/overview/what-is-cloud-computing/

Oracle Mexico. (2021, January 16). *¿Qué es una base de datos? | Oracle México*. ¿Qué Es Una Base de Datos? https://www.oracle.com/mx/database/what-is-database/

Royce, W. W. (1970). *MANAGING THE DEVELOPMENT OF LARGE SOFTWARE SYSTEMS*. http://www-scf.usc.edu/~csci201/lectures/Lecture11/royce1970.pdf

**AUTOR: Luis Felipe Velasco Tao – Juan David González Dimaté**