**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:**

* 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 cada uno de los componentes diseñados.
* Implementar el sistema distribuido con cada uno de los componentes.
* Realizar pruebas para la mejora del sistema.

**TEMÁTICAS:**

1. **Cloud Computing**: DIMATE
2. **Bases de datos**: DIMATE
3. **Servicios web**: DIMATE
4. **Dispositivos móviles**: DIMATE
5. **Requerimientos del negocio**: DIMATE
6. **Pruebas de seguridad**: DIMATE

**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/) | DIMATE |
|  | [MySQL 5.7](https://dev.mysql.com/doc/refman/5.7/en/) | DIMATE |
|  | [MySQL Workbench 8.0](https://dev.mysql.com/doc/workbench/en/) | DIMATE |
| PuTTY 0.76 para Windows - Descargar | [PuTTY / PuTTYgen](https://the.earth.li/~sgtatham/putty/latest/w64/putty-64bit-0.76-installer.msi) | DIMATE |
|  | [Java 1.8](https://docs.oracle.com/javase/8/docs/api/) | DIMATE |
|  | [Spring Boot – Spring Tools 4.2](https://docs.spring.io/spring-boot/docs/current/reference/htmlsingle/) | DIMATE |
| 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) | DIMATE |
|  | [Android Studio 4.2](https://developer.android.com/studio) | DIMATE |
|  | [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/) | DIMATE |
| PC Icon - Free Icons | Equipos personales | DIMATE |
|  | [Github – Github Desktop](https://github.com/lfvelascot/SISTEMAS_DISTRIBUIDOS) | DIMATE |

Ilustración 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.

**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

**Arquitectura del sistema**: este sistema distribuido contará con 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 seria una aplicación 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.

**METODOLOGÍA:**

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. Las fases definidas en esta metodología son las siguientes:

***Metodología en cascada***

DEFINICION DIMATE

1. **Definición de requerimientos**: DIMATE

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

1. **Diseño de componentes**: DIMATE

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**: DIMATE

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**: DIMATE

En esta fase se obtendrán los siguientes productos:

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

1. **Mantenimiento**: DIMATE

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 estructura de la metodología en cascada

**PROCEDIMIENTO O DESARROLLO INGENIERÍL**

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.

|  |  |  |
| --- | --- | --- |
| **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 ultimo como medida de seguridad. |

Ilustración 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 tabla de requerimientos no funcionales

***Diseño de componentes***

***Implementación***

***Verificación***

***Mantenimiento***

**PRODUCTOS Y RESULTADOS ESPERADOS:**

**BIBLIOGRAFÍA LIBROS DIGITALES DE LA UNIVERSIDAD:**

**BIBLIOWEB:**

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