UNIVERSIDAD PRIVADA DE TACNA

FACULTAD DE INGENIERÍA

Escuela Profesional de Ingeniería de Sistemas

"App de Gestión Financiera para el Registro y Análisis de Gastos Personales"

Curso: Patrones de Software

Docente: Ing. Patrick Cuadros

Integrantes:

Ayma Choque, Erick Yoel (2021072616)
Poma Machicado, Fabiola Estefani (2021070030)
Tapia Vargas, Dylan Yariet (2021072630)

Tacna – Perú
2025

Digix

“App de Gestión Financiera para el Registro y Análisis de Gastos Personales”
Diccionario de Datos

Versión *{1.0}*

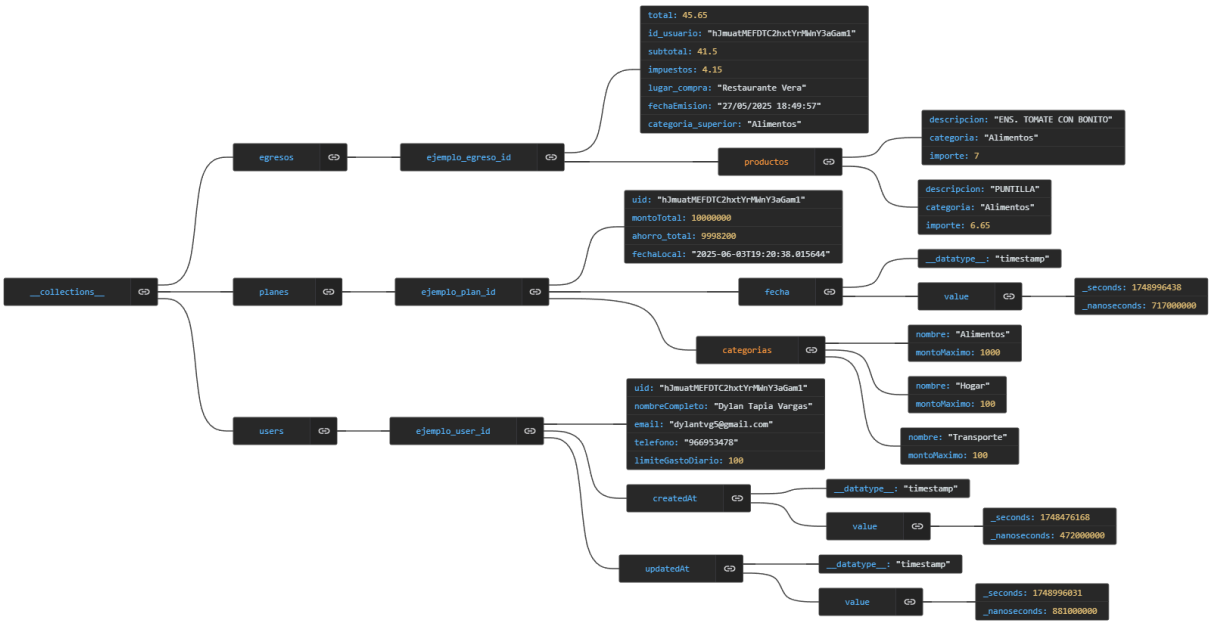
Digix

ÍNDICE GENERAL

Diccionario de Datos

1. Modelo Entidad / no relación

1.1. Diseño



Digix

2. DICCIONARIO DE DATOS - Vanguard Money (Firestore NoSQL)

2.1. Colecciones

COLECCIÓN 1: users

Nombre de la Colección:	users
Descripción de la Colección:	Almacena la información de los usuarios registrados en la aplicación VanguardMoney, incluyendo datos personales, configuraciones de límites de gasto y fechas de auditoría
Objetivo:	Gestionar el registro, autenticación y configuración personal de cada usuario del sistema
Tipo de Datos:	Documento NoSQL (Firestore)
Clave Primaria:	uid (String)

Descripción de los campos

Campo	Tipo	Logitud	Descripcion	Restricciones	Ejemplo
uid	String	28	Identificador único del usuario (Firebase Auth)	PK, NOT NULL	"hJmuatMEFDTC2hxtYrMWnY3aGam1"
nombreCompleto	String	100	Nombre completo del usuario	NOT NULL	"Dylan Tapia Vargas"
email	String	100	Correo electrónico del usuario	NOT NULL, UNIQUE	"dylantvg5@gmail.com"
telefono	String	15	Número telefónico del usuario	NOT NULL	"966953478"
limiteGastoDiario	Number	-	Límite de gasto diario establecido por el usuario	>= 0	100
createdAt	Timestamp	-	Fecha y hora de creación de la cuenta	NOT NULL	_seconds: 1748476168
updatedAt	Timestamp	-	Fecha y hora de última actualización	NOT NULL	_seconds: 1748996031

Digix

COLECCIÓN 2: egresos

Nombre de la Colección:	egresos
Descripción de la Colección:	Registra todas las transacciones de gastos realizadas por los usuarios, incluyendo detalles de compra, productos adquiridos y categorización automática mediante IA
Objetivo:	Controlar y monitorear todos los gastos del usuario para análisis financiero y cumplimiento de presupuestos
Tipo de Datos:	Documento NoSQL con array embebido (productos)
Clave Primaria:	uid (String)
Clave Foránea	id_usuario → users.uid

Descripción de los campos

Campo	Tipo	Logitud	Descripcion	Restricciones	Ejemplo
id	String	20	Identificador único del egreso	PK, NOT NULL	"ejemplo_egreso_id"
id_usuario	String	28	ID del usuario propietario	FK → users.uid	"hJmuatMEFDTC2hxtYrMWnY3aGam1"
total	Number	-	Monto total del egreso (con impuestos)	> 0	45.65
subtotal	Number	-	Subtotal sin impuestos	>= 0	41.5
impuestos	Number	-	Monto de impuestos aplicados	>= 0	4.15
lugar_compra	String	200	Establecimiento donde se realizó la compra	NOT NULL	"Restaurante Vera"
fechaEmision	String	25	Fecha y hora de emisión del comprobante	NOT NULL	"27/05/2025 18:49:57"
categoria_superior	String	50	Categoría principal del gasto	NOT NULL	"Alimentos"
productos	Array	-	Lista de productos/serv	NOT NULL, >=	[...]

			icios comprados	1 elemento	

Subcolección Embebida: productos

Campo	Tipo	Longitud	Descripción	Restricciones	Ejemplo
descripcion	String	200	Descripción detallada del producto	NOT NULL	"ENS. TOMATE CON BONITO"
categoria	String	50	Categoría específica del producto	NOT NULL	"Alimentos"
importe	Number	-	Precio individual del producto	> 0	7.0

COLECCIÓN 3: planes

Nombre de la Colección:	planes
Descripción de la Colección:	Almacena los planes de presupuesto y ahorro de cada usuario, incluyendo límites por categoría de gasto y seguimiento de metas financieras
Objetivo:	Facilitar la planificación financiera personal mediante el establecimiento de límites de gasto por categorías y control de ahorros
Tipo de Datos:	Documento NoSQL con array embebido (productos)
Clave Primaria:	uid (String)
Clave Foránea	uid → users.uid

Descripción de los campos

Campo	Tipo	Logitud	Descripcion	Restricciones	Ejemplo
id	String	20	Identificador único del plan	PK, NOT NULL	"ejemplo_plan_id"
uid	String	28	ID del usuario propietario	FK → users.uid	"hJmuatMEFDTC2hxtYrMWnY3aGam1"
montoTotal	Number	-	Monto total del plan de ahorro/presupuesto	> 0	10000000
ahorro_total	Number	-	Total de ahorro acumulado hasta la fecha	>= 0	9998200

fecha	Timestamp	-	Fecha de creación del plan	NOT NULL	_seconds: 1748996438
fechaLocal	String	30	Fecha local en formato ISO string	NOT NULL	"2025-06-03T19:20:38.015644"
categorias	Array	-	Límites de gasto por categoría	NOT NULL, >= 1 elemento	[...]

Subcolección Embebida: categorias

Campo	Tipo	Longitud	Descripción	Restricciones	Ejemplo
nombre	String	50	Nombre de la categoría de gasto	NOT NULL	"Alimentos"
montoMaximo	Number	-	Límite máximo de gasto para la categoría	> 0	1000

Digix

1.1. Lenguaje de Definición de Datos (DDL)

```
// Estructura de la base de datos NoSQL - Collections Schema
      // Colección: users
      const userSchema = {
        uid: "string", // ID único del usuario
        nombreCompleto: "string",
        email: "string",
        telefono: "string",
        limiteGastoDiario: "number",
        createdAt: "timestamp",
        updatedAt: "timestamp"
      }
      // Colección: planes
      const planSchema = {
        uid: "string", // Referencia al usuario
        montoTotal: "number",
        ahorro_total: "number",
        fecha: "timestamp",
        fechaLocal: "string",
        categorias: [ // Array de objetos
          {
            nombre: "string",
            montoMaximo: "number"
          }
        ]
      }
      // Colección: egresos
      const egresoSchema = {
        total: "number",
        id_usuario: "string", // Referencia al usuario
        subtotal: "number",
        impuestos: "number",
        lugar_compra: "string",
        fechaEmision: "string",
        categoria_superior: "string",
        productos: [ // Array de objetos
          {
            descripcion: "string",
            categoria: "string",
            importe: "number"
          }
        ]
```

Digix

1.2. Lenguaje de Manipulación de Datos (DML)

```
// Operaciones CRUD para Firebase Firestore

import { db } from './firebase-config.js';
import { collection, doc, addDoc, getDoc, getDocs, updateDoc,
deleteDoc, query, where } from 'firebase/firestore';

// === CREAR (CREATE) ===
// Crear usuario
async function crearUsuario(userData) {
  try {
    const docRef = await addDoc(collection(db, "users"),
userData);
    return docRef.id;
  } catch (error) {
    console.error("Error creando usuario:", error);
  }
}

// Crear plan
async function crearPlan(planData) {
  try {
    const docRef = await addDoc(collection(db, "planes"),
planData);
    return docRef.id;
  } catch (error) {
    console.error("Error creando plan:", error);
  }
}

// Crear egreso
async function crearEgreso(egresoData) {
  try {
    const docRef = await addDoc(collection(db, "egresos"),
egresoData);
    return docRef.id;
  } catch (error) {
    console.error("Error creando egreso:", error);
  }
}
```

```
// === LEER (READ) ===
// Obtener usuario por ID
async function obtenerUsuario(uid) {
  try {
    const docRef = doc(db, "users", uid);
    const docSnap = await getDoc(docRef);
    return docSnap.exists() ? docSnap.data() : null;
  } catch (error) {
    console.error("Error obteniendo usuario:", error);
  }
}

// Obtener egresos por usuario
async function obtenerEgresosPorUsuario(uid) {
  try {
    const q = query(collection(db, "egresos"),
where("id_usuario", "==", uid));
    const querySnapshot = await getDocs(q);
    return querySnapshot.docs.map(doc => ({ id: doc.id,
...doc.data() }));
  } catch (error) {
    console.error("Error obteniendo egresos:", error);
  }
}

// === ACTUALIZAR (UPDATE) ===
// Actualizar usuario
async function actualizarUsuario(uid, updateData) {
  try {
    const docRef = doc(db, "users", uid);
    await updateDoc(docRef, updateData);
    return true;
  } catch (error) {
    console.error("Error actualizando usuario:", error);
  }
}

// === ELIMINAR (DELETE) ===
// Eliminar egreso
async function eliminarEgreso(egresoId) {
  try {
```

```
    await deleteDoc(doc(db, "egresos", egresoId));
    return true;
  } catch (error) {
    console.error("Error eliminando egreso:", error);
  }
}
```