**Lista de entidades para Merkadit**

**Tablas de addresses**

**Countries**

* countryID: TINYINT AUTO\_INCREMENT PK
* name: VARCHAR(60)

**States**

* stateID: INT AUTO\_INCREMENT PK
* name: VARCHAR(60)
* countryID: TINYINT FK → Countries

**Cities**

* cityID: INT AUTO\_INCREMENT PK
* name: VARCHAR(60)
* stateID: INT FK → States

**Addresses**

Addresses for users or buildings registered.

* addressID: INT AUTO\_INCREMENT PK
* line1: VARCHAR(200)
* line2: VARCHAR(200)
* zipCode: VARCHAR(10)
* location: POINT
* cityID: INT FK → Cities

**AddressXUsers**

* addressXUserID: INT AUTO\_INCREMENT PK
* addressID: INT PK, FK → Addresses
* userID: INT PK, FK → Users
* postTime: DATETIME DEFAULT NOW()
* enabled: BIT DEFAULT 1
* isDefault: BIT DEFAULT 0

**Currencies**

Different types of currency that Merkadit will manage.

* currencyID: SMALLINT AUTO\_INCREMENT PK
* name: VARCHAR(60)
* isoCode: VARCHAR(3)
* currencySymbol: VARCHAR(5)
* countryID: TINYINT FK → Countries

**ExchangeRates**

* exchangeRateID: SMALLINT AUTO\_INCREMENT PK
* startDate: Datetime
* endDate: Datetime NULL
* exchangeRate: Decimal(10,4)
* currencySourceID: SMALLINT FK → Currencies
* currencyDestinyID: SMALLINT FK → Currencies
* current: BIT DEFAULT 1

**Tablas de usuario**

**Users**

* userID: INT AUTO\_INCREMENT PK
* firstName: VARCHAR(50)
* lastName: VARCHAR(50)
* passwordHash: VARBINARY(250)
* createdDate: DATETIME DEFAULT NOW()
* lastLogin: DATETIME
* isActive: BIT DEFAULT 1
* checksum: VARBINARY(250)

**UserContacts**

* contactID: INT AUTO\_INCREMENT PK
* userID: INT FK → Users
* ContactTypeID: TINYINT FK →ContactTypes
* value: VARCHAR(80)
* postTime: Datetime
* enabled: BIT DEFAULT 1
* deleted: BIT DEFAULT 0

**ContactTypes**

* contactTypeID: TINYINT AUTO\_INCREMENT PK
* name: VARCHAR(60)

**UserXRoles**

* userID: INT PK, FK → Users
* roleID: INT PK, FK → Roles
* postTime: DATETIME DEFAULT NOW()
* enabled: BIT DEFAULT 1
* checksum: VARBINARY(250)

**Roles**

The roles that the app will include are: Administrator, Tenant

* roleID: INT AUTO\_INCREMENT PK
* name: VARCHAR(30)
* description: VARCHAR(200)

**PermissionXRoles**

* permissionID: INT PK, FK → Permissions
* roleID: INT PK, FK → Roles
* postTime: DATETIME DEFAULT NOW()
* enabled: BIT DEFAULT 1
* checksum: VARBINARY(250)

**Permissions**

Every role has specific permissions. An administrator can invest in a location, receive full financial reports, and create a contract. A tenant can register a commerce, insert products or manage sales.

* permissionID: INT AUTO\_INCREMENT PK
* name: VARCHAR(50)
* description: VARCHAR(200)
* code: VARCHAR(20) UNIQUE
* module: VARCHAR(50)

**Tablas de comercios**

**Commerces**

This table manages general data about a specific commerce. Here are the clarifications about some fields: taxID is a unique identification that is assigned by a government authority. Each commerce is related to one user, who has the tenant role.

* commerceID: INT AUTO\_INCREMENT PK
* name: VARCHAR(100)
* legalName: VARCHAR(200)
* taxID: VARCHAR(30) UNIQUE
* commerceTypeID: TINYINT FK → CommerceTypes
* ownerUserID: INT FK → Users
* createdDate: DATETIME DEFAULT NOW()
* isActive: BIT DEFAULT 1

**CommerceTypes**

Categories for the commerces: Food, retail, beauty, gardening…

* commerceTypeID: TINYINT AUTO\_INCREMENT PK
* name: VARCHAR(60)

**CommerceContacts**

* commerceXContactID: INT AUTO\_INCREMENT PK
* commerceID: INT FK → Commerces
* contactTypeID: TINYINT FK → ContactTypes
* value: VARCHAR(80)
* postTime: DATETIME DEFAULT NOW()
* enabled: BIT DEFAULT 1
* deleted: BIT DEFAULT 0

**Buildings**

Building registered in the app. These are the physical locations where a space can be in. Administrators rent spaces, which belong to a major building. An example of a building

* buildingID: INT AUTO\_INCREMENT PK
* name: VARCHAR(100)
* totalArea: DECIMAL(10,2)
* floors: TINYINT
* openingTime: TIME
* closingTime: TIME
* adminUserID: INT FK → Users
* addressID: INT FK → Addresses
* initialInvestment: DECIMAL(15,2)
* createdDate: DATETIME DEFAULT NOW()

**Floors**

In order to keep track of all space locations, there is an intermediate table to indicate floors whenever it is necessary.

* buildingID
* name
* floorNumber

**SpaceTypes** ENUM (‘kiosk’, ‘restaurant, ‘retail store’)

* spaceTypeID: TINYINT AUTO\_INCREMENT PK
* name: VARCHAR(50)
* description: VARCHAR(200)

**Spaces**

The concrete space that you can rent. It includes a code to better identify it. The area is always given in square meters.

* spaceID: INT AUTO\_INCREMENT PK
* floorID: INT FK → floors
* spaceCode: VARCHAR(20)
* name: VARCHAR(100)
* area: DECIMAL(10,2)
* spaceTypeID: TINYINT FK → SpaceTypes
* spaceStatusID FK
* baseRent: DECIMAL(16,2)

**SpaceStatus** ENUM (‘available’,‘occupied’, ‘under renovation’)

* statusID : TINYINT AUTO\_INCREMENT PK
* name : VARCHAR(60)

**Tablas de contratos**

**Contracts**

The contract refers to the agreement an administrator and a tenant make to rent a space. It includes the monetary information necessary to calculate payments from the tenant, on the dates given by the schedule. The user that creates the agreement is the administrator of the space being rented.

* contractID: INT AUTO\_INCREMENT PK
* contractNumber: VARCHAR(50) UNIQUE
* commerceID: INT FK → Commerces
* spaceID: INT FK → Spaces
* startDate: DATE
* endDate: DATE
* documentUrl: varchar(500)
* baseRent: DECIMAL(16,2)
* currencyID: SMALLINT FK → Currencies
* commissionPercentage: DECIMAL(5,2) DEFAULT 0
* scheduleID: INT FK → Schedules
* contractStatusID: TINYINT FK → ContractStatus
* createdDate: DATETIME DEFAULT NOW()
* createdBy: userID FK → Users

**Schedules**

* scheduleID: INT AUTO\_INCREMENT PK
* scheduleRecurrencyID: TINYINT FK → ScheduleRecurrencies
* startDate: DATETIME
* endDate: DATETIME
* lastExecute: DATETIME NULL
* nextExecute: DATETIME
* enabled: BIT DEFAULT 1
* deleted: BIT DEFAULT 0

**ScheduleRecurrencies**

Says how frequent the schedule is: Monthly, daily, anual...

* scheduleRecurrencyID: TINYINT AUTO\_INCREMENT PK
* name: VARCHAR(50)
* intervalDays: INT

**ContractStatus**

* contractStatusID: TINYINT AUTO\_INCREMENT PK
* name: VARCHAR(30)

**Settlements**

Settlements are every instance of payment the tenant must do. Each settlement is related to a specific contract. With this, you can see the history of payments made.

* settlementID: INT AUTO\_INCREMENT PK
* contractID: INT FK → Contracts
* scheduleID: INT FK → Schedules
* baseRentAmount: DECIMAL(16,2)
* totalSales: DECIMAL(16,2)
* commissionAmount: DECIMAL(16,2)
* totalAmount: DECIMAL(16,2)
* currencyID: SMALLINT FK → Currencies
* settlementDate: DATETIME DEFAULT NOW()
* settlementStatusID: TINYINT FK → SettlementStatus
* paidDate: DATETIME
* createdBy: INT FK → Users

**SettlementStatus** ENUM('pending', 'paid', 'overdue', 'cancelled')

* settlementStatusID: TINYINT AUTO\_INCREMENT PK
* name: VARCHAR(30)

**Tablas de inventario**

**ProductCategories** ENUM( ‘food’, ’clothing’ ,’jewelry’)

* categoryID: INT AUTO\_INCREMENT PK
* name: VARCHAR(50)
* description: VARCHAR(200)

**Products**

It is the stock of a specific product that a seller keeps in their inventory. Sku is a unique code that identifies the product. The fields minStock and maxStock keep track of the minimum amount of product that can be stored before warning the seller, and maximum that should be kept most of the time.

* productID: INT AUTO\_INCREMENT PK
* commerceID: INT FK → Commerces
* sku: VARCHAR(50)
* barcode: VARCHAR(50)
* name: VARCHAR(100)
* description: VARCHAR(200)
* categoryID: INT FK → Categories
* currencyID: INT FK → Currencies
* stockQuantity: INT DEFAULT 0
* minStock: INT DEFAULT 0
* maxStock: INT
* isActive: BIT DEFAULT 1
* createdDate: DATETIME DEFAULT NOW()
* deleted: BIT DEFAULT 0

**PriceHistory**

PriceHistory saves the prices of every product at a given period. If it gets changed, the table will register it. It is a value used for sales checks.

* priceHistoryID: INT AUTO\_INCREMENT PK
* productID: FK → Products
* price: DECIMAL(15,2)
* cost: DECIMAL (15,2)
* currencyID: SMALLINT FK → Currencies
* isCurrent: BIT
* createdDate: DATETIME DEFAULT NOW()

**InventoryMovements**

InventoryMovements register modifications made to a product’s stock. The reference fields indicate the reason of a movement, whether it is a sell, restock, or other. Since the quantity of a product is a delicate subject, a checksum has been included in case of suspicious behavior.

* movementID: INT AUTO\_INCREMENT PK
* productID: INT FK → Products
* movementTypeID: TINYINT FK → MovementTypes
* stockQuantity: INT
* referenceDescription: VARCHAR(100)
* referenceID: INT
* movementDate: DATETIME DEFAULT NOW()
* createdBy: VARCHAR(50)
* checksum: VARBINARY(250)

**MovementTypes** ENUM('IN', 'OUT', 'ADJUSTMENT', 'RETURN')

* movementTypeID: TINYINT AUTO\_INCREMENT PK
* name: VARCHAR(60)

**Tablas de ventas**

**Customers**

Customers are people who interact with the commerces, but are not users of Merkadit. This table helps markets keep track of who visits their stores and create a sale check accordingly.

* customerID: INT AUTO\_INCREMENT PK
* customerType: TINYINT FK → CustomerTypes
* name: VARCHAR(60)
* taxID: VARCHAR(30)
* birthdate: Datetime
* addressID: INT FK → Addresses
* createdDate: DATETIME DEFAULT NOW()
* enabled: BIT DEFAULT 1
* deleted: BIT DEFAULT 0

**CustomerTypes** ENUM(‘company’, ‘person’)

* customerTypeID: TINYINT AUTO\_INCREMENT PK
* name: VARCHAR(60)

**Payment Methods**

Payment methods available in Merkadit.

* paymentMethodID: TINYINT AUTO\_INCREMENT PK
* name: VARCHAR(50)
* code: VARCHAR(20) UNIQUE
* requiresReference: BIT DEFAULT 0
* processingFee: DECIMAL(5,2) DEFAULT 0

**Payment**

These are payments made inside the application. Contracts and investments make use of them to make scheduled deposits to the designated user.

* paymentID: INT AUTO\_INCREMENT PK
* paymentMethodID : TINYINT FK → PaymentMethods
* paymentTypeID: TINYINT FK → PaymentTypes
* transactionAmount: Decimal(16,2)
* currencyID: SMALLINT FK → Currencies
* description: VARCHAR(100)
* paymentDate: Datetime
* paymentReference: VARCHAR(100)
* paymentConfirmation: VARCHAR(100)
* paymentDate: DATE
* paymentStatusID: TINYINT FK → PaymentStatus
* checksum: VARBINARY(250)

**PaymentStatus** ENUM('pending', 'completed', 'failed', 'cancelled')

* paymentStatusID: TINYINT AUTO\_INCREMENT PK
* name: VARCHAR(60)

**PaymentTypes** ENUM(‘settlement’, ‘investment’, ‘expense’)

* paymentTypeID: TINYINT AUTO\_INCREMENT PK
* name: VARCHAR(50)
* description: VARCHAR(200)

**Sales**

Similar to a bill. It includes all the necessary information to register a sale.

* saleID: INT AUTO\_INCREMENT PK
* commerceID: INT FK → Commerces
* invoiceNumber: VARCHAR(50)
* saleDate: DATETIME DEFAULT NOW()
* customerID: INT FK → Customers
* subtotal: DECIMAL(16,2)
* discountAmount: DECIMAL(16,2) DEFAULT 0
* taxAmount: DECIMAL(16,2) DEFAULT 0
* totalAmount: DECIMAL(16,2)
* currencyID: SMALLINT FK → Currencies
* computer: VARCHAR(50)
* userID: INT FK → Users
* paymentMethodID: TINYINT FK → PaymentMethods
* paymentReference: VARCHAR(100)
* paymentConfirmation: VARCHAR(100)
* saleStatusID: TINYINT FK → SaleStatus
* checksum: VARBINARY(250)

**SaleStatus** ENUM('completed', 'cancelled', 'pending', 'refunded')

* saleStatusID : TINYINT AUTO\_INCREMENT PK
* name: VARCHAR(30)

**SaleDetails**

It specifies the information for every product being purchased.

* saleDetailID: INT AUTO\_INCREMENT PK
* saleID: INT FK → Sales
* productID: INT FK → Products
* unitPrice: DECIMAL(16,2)
* quantity: INT
* discountAmount: DECIMAL(16,2) DEFAULT 0
* subtotal: DECIMAL(16,2)

**Tablas Financieras**

**Investments**

These are the initial investments made by an administrator to own a space in a building.

* investmentID: INT AUTO\_INCREMENT PK
* spaceID: INT FK → spaces
* contractID: INT FK → contratos
* description: VARCHAR(500)
* amount: DECIMAL(15,2)
* currencyID: SMALLINT FK → Currencies
* investmentDate: DATE
* investmentCategoryID: TINYINT FK → InvestmentCategories
* userID: INT FK → Users
* createdDate: DATETIME DEFAULT NOW()
* checksum: VARBINARY

**InvestmentCategories**

* investmentCategoryID: TINYINT AUTO\_INCREMENT PK
* name : VARCHAR(50)
* description : VARCHAR(200)
* isCapital : BIT
* depreciationMonths : SMALLINT

**Expenses**

Any monetary expense made goes here. We have different IDs depending on where the expense comes from (either the general building or the specific space).

* expenseID: INT AUTO\_INCREMENT PK
* buildingID: INT FK → Buildings NULL
* spaceID: INT FK → Spaces NULL
* description: VARCHAR(500)
* amount: DECIMAL(16,2)
* expenseDate: DATE
* expenseTypeID: TINYINT FK → ExpenseTypes
* invoiceNumber: VARCHAR(50)
* supplierName: VARCHAR(100)
* createdBy: INT FK → Users
* createdDate: DATETIME DEFAULT NOW()

**ExpenseTypes** ENUM('utilities', 'security', 'cleaning', 'marketing', 'maintenance', 'other')

* expenseTypeID: TINYINT
* name: VARCHAR(60)

**FinancialReports**

Reports for administrators. It is generated by combining the gains of every commerce using this administrator’s spaces.

* reportID: INT AUTO\_INCREMENT PK
* userID: INT FK → Users
* startDate: DATE
* endDate: DATE
* totalRevenue: DECIMAL(16,2)
* totalExpenses: DECIMAL(16,2)
* netIncome: DECIMAL(16,2)
* grossMargin: DECIMAL(5,2)
* occupancyRate: DECIMAL(5,2)
* currencyID: SMALLINT FK → Currencies
* reportTypeID: TINYINT FK → ReportTypes
* documentURL: VARCHAR(250) NULL
* postTime: DATETIME DEFAULT NOW()

**FinancialReportDetails**

This table connects financial movements with a report. The rows are generated at the same time the report is created, using prior monetary registers. It becomes helpful for reviewing all the movements that compose a single report.

* financialReportDetailID -> INT AUTO\_INCREMENT PK
* financialReportID : INT FK -> FinancialReports
* expensesID: INT FK NULL → Expenses NULL
* investmentID: INT FK NULL → Investments NULL
* commerceReportID: INT FK NULL-> commerceReports NULL

**ReportTypes**

* reportTypeID: TINYINT AUTO\_INCREMENT PK
* name: VARCHAR(50)
* description: VARCHAR(200)

**CommerceReports**

Reports for commerces. It registers gains through sales.

* commerceReportID: INT AUTO\_INCREMENT PK
* commerceID: INT FK → Commerces
* startDate: DATE
* endDate: DATE
* totalSales: DECIMAL(16,2)
* totalCosts: DECIMAL(16,2)
* grossProfit: DECIMAL(16,2)
* totalRent: DECIMAL(16,2)
* commissionPaid: DECIMAL(16,2)
* netProfit: DECIMAL(16,2)
* currencyID: SMALLINT FK → Currencies
* postTime: DATETIME DEFAULT NOW()
* userID: INT FK → Users

**CommerceReportsDetails**

This table provides specific information about the revenue and expenses from a commerce’s financial reports.

* commerceReportDetailID: INT AUTO\_INCREMENT PK
* commerceReportID: INT FK → CommerceReports
* contractID: INT FK → Contracts -- Esto da acceso a spaceID
* salesAmount: DECIMAL(16,2)
* rentAmount: DECIMAL(16,2)
* commissionAmount: DECIMAL(16,2)

**Tablas de Logs**

**Logs**

* logID: BIGINT AUTO\_INCREMENT PK
* posttime: DATETIME DEFAULT NOW()
* description: VARCHAR(500)
* computer: VARCHAR(100)
* username: VARCHAR(50)
* ref1ID: BIGINT NULL
* ref2ID: BIGINT NULL
* value1: VARCHAR(200) NULL
* value2: VARCHAR(200) NULL
* logTypeID: TINYINT FK → LogTypes
* logLevelID: TINYINT FK → LogLevels
* logSourceID: TINYINT FK → LogSources
* checksum: VARBINARY(250)

**LogTypes**

* logTypeID: TINYINT AUTO\_INCREMENT PK
* name: VARCHAR(50)

**LogLevels**

* logLevelID: TINYINT AUTO\_INCREMENT PK
* name: VARCHAR(30)

**LogSources**

* logSourceID: TINYINT AUTO\_INCREMENT PK
* name: VARCHAR(50)