droidcon academy

Implementing Entity, DAO, and Setup Database

Belal Khan

@probelalkhan

in/probelalkhan

Overview

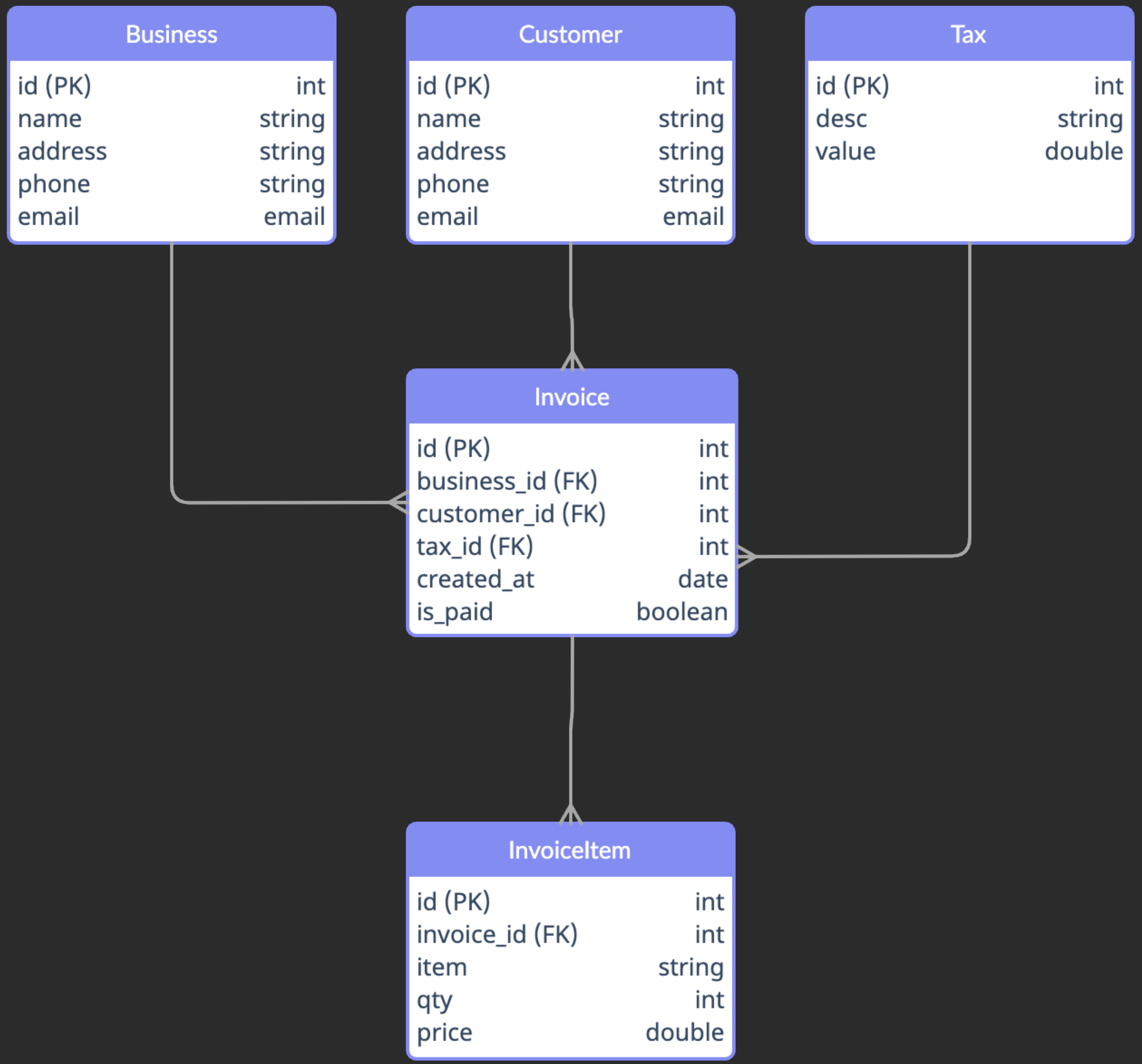
- Learn to design a database or how to reach an appropriate database solution for a given problem statement
- Learn to create database entity and dao for various CRUD operations
- Learn to setup Room database in your project

Easy Invoice Database

- Entities
  - Business
  - Customer
  - Tax
  - Invoice

Invoice Table

- Table contains multiple items.
  - Invoice
  - InvoiceItem


Business
id (PK) int
name string
address string
phone string
email email
Customer
id (PK) int
name string
address string
phone string
email email
Tax
id (PK) int
desc string
value double
Invoice
id (PK) int
business_id (FK) int
customer_id (FK) int
tax_id (FK) int
created_at date
is_paid boolean
InvoiceItem
id (PK) int
invoice_id (FK) int
item string
qty int
price double

Defining our Entities

- Business
- Customer
- Tax
- Invoice
- InvoiceItem

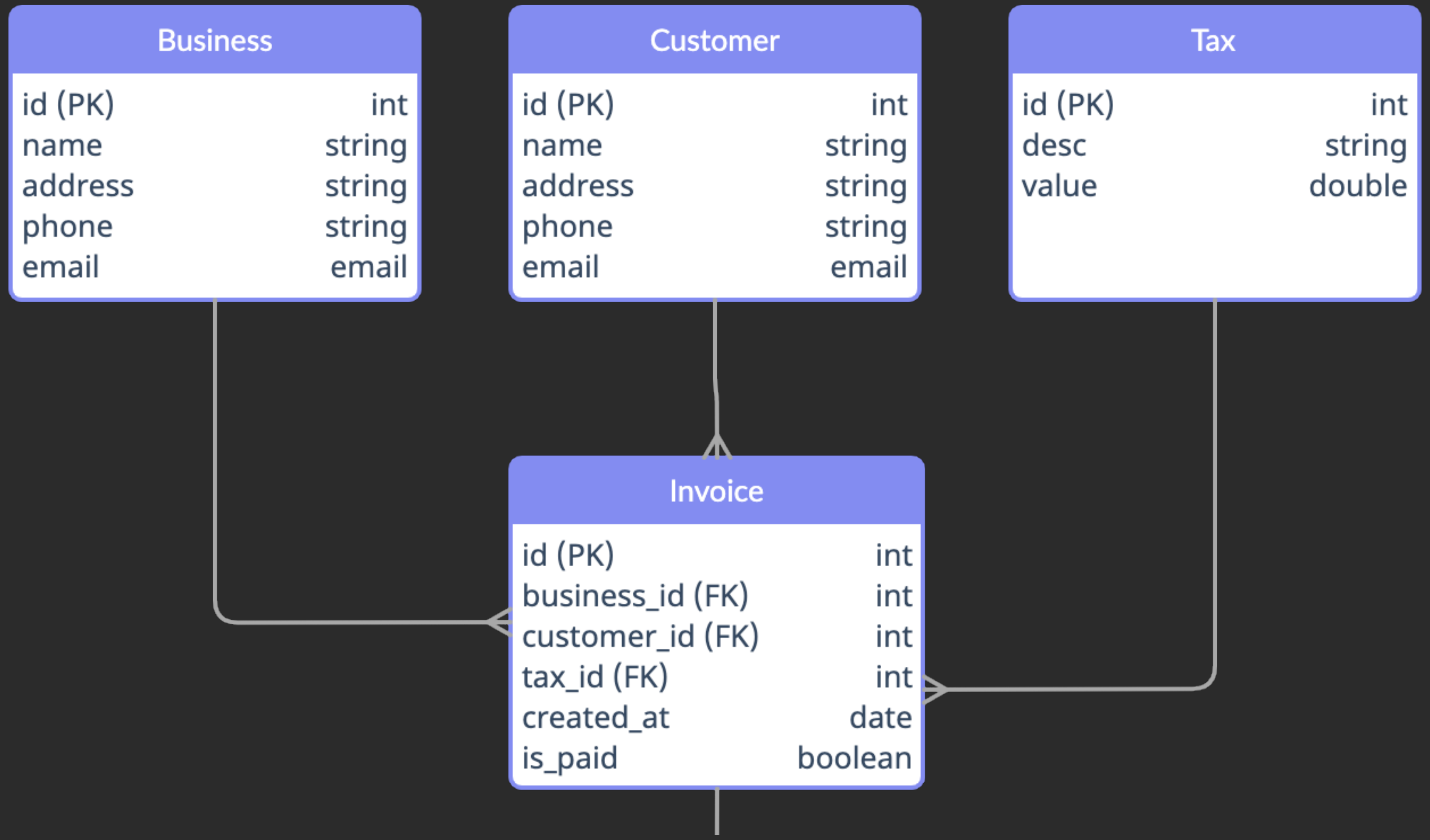
```
@Entity(tableName = "businesses")
data class Business(
    val name: String,
    val address: String,
    val phone: String,
    val email: String
) {
    @PrimaryKey(autoGenerate = true)
    var id: Int? = null

    val completeAddress: String
        get() = "$address\nphone:
                 $phone\nemail: $email"
}
```

Code Challenge

- Define Remaining Entities
  - Customer
  - Tax
  - Invoice
  - InvoiceItem

Using Foreign Key



```
@Entity(
    tableName = "invoices",
    foreignKeys = [
        ForeignKey(
            entity = Business::class,
            parentColumns = ["id"],
            childColumns = ["business_id"],
            onDelete = ForeignKey.CASCADE
        ),
        ForeignKey(
            entity = Customer::class,
            parentColumns = ["id"],
            childColumns = ["customer_id"],
            onDelete = ForeignKey.CASCADE
        )
    ]
)
```

Code Challenge

- Finish code challenge to continue…

Defining Daos

- Business
- Customer
- Tax
- Invoice

```
@Dao
interface BusinessDao {

    @Insert(onConflict = OnConflictStrategy.REPLACE)
    suspend fun addUpdateBusiness(business: Business)

    @Query("SELECT * FROM businesses")
    fun getBusinesses(): Flow<List<Business>>

    @Delete
    suspend fun deleteBusiness(business: Business)
}
```

Code Challenge

- Finish all the remaining daos.
  - Customer
  - Tax
  - Invoice
- Finish this challenge to continue…

The Database Class

- Now let’s define the main entry point of our database.

Summary

- Understood our application’s data requirement
- Designed Easy Invoice Database Schema
- Defined Required Entities
- Understood using Foreign Keys
- Understood creating DAOs for our Database.
- Database Class Setup

Creating Repository Layer

Up Next