Diagramas de Componentes, Clases e Implementación de Interfaz

Docente: Alejandro Rodas Vásquez
Universidad Tecnológica de Pereira

23 de mayo de 2023

Índice

1. Clientes - Cuentas

A continuación, se muestra la Arquitectura Basada en Componentes (Figura 1) para una aplicación de Banco (que por el momento solo cuenta con la relación *Cliente / Cuenta*).

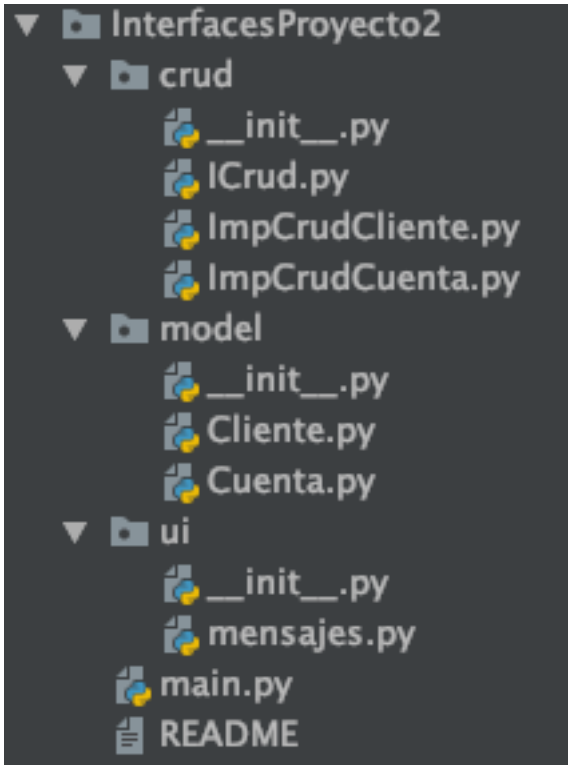


Figura 1: Estructura de la Arquitectura

Esta aplicación está construida bajo los siguientes parámetros arquitectónicos:

1. Los componentes separados por responsabilidades (Modelo, CRUD, Vista (UI) y Programa Principal).
2. Se utiliza el concepto de *Módulos y NameSpace*.

1.1. Arquitectura de Software

Modelo

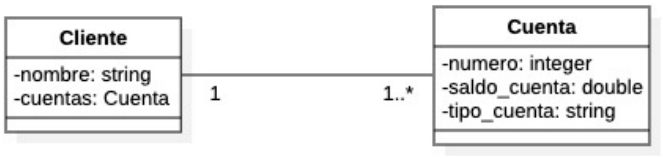


Diagrama de Componentes

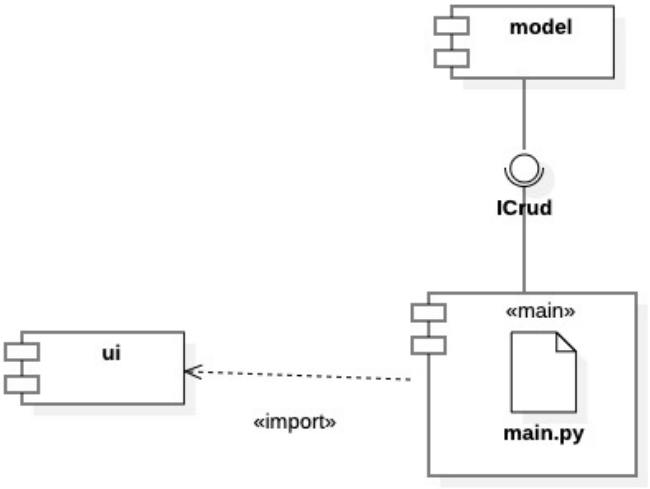
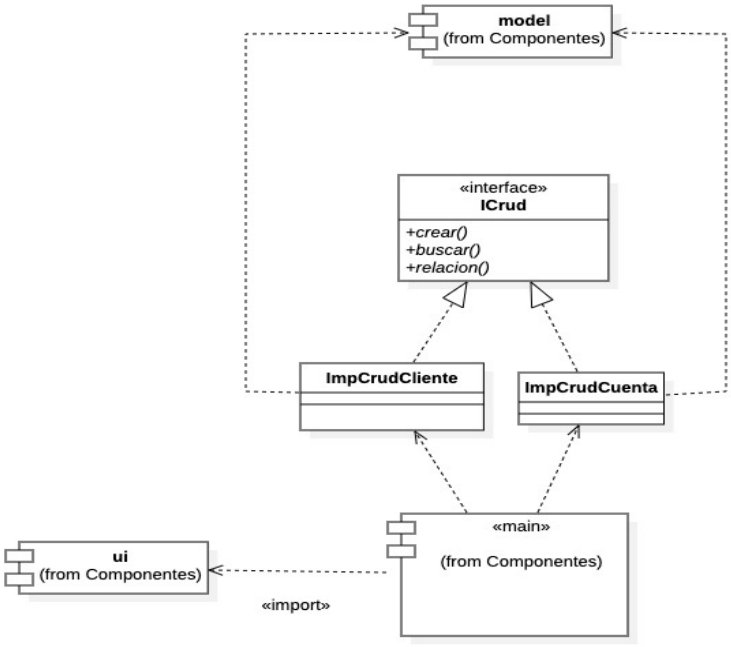


Diagrama de Clases (Interfaz Implementada)



1.2. Código Fuente

1.2.1. Modelo

Cliente.py

```
class Cliente:

  def __init__(self, nombre):
    self.__nombre = nombre
    self.__cuentas = []


  @property
  def nombre(self):
    return self.__nombre

  @nombre.setter
  def nombre(self, nombre):
    self.__nombre = nombre

  @property
  def cuentas(self):
    return self.__cuentas

  @cuentas.setter
  def cuentas(self, cuenta):
    self.__cuentas.append(cuenta)
```

Cuenta.py

```
class Cuenta:
  def __init__(self, numero, saldo_cuenta, tipo_cuenta):
    self.__numero = numero
    self.__saldo_cuenta = saldo_cuenta
    self.__tipo_cuenta = tipo_cuenta

  @property
  def numero(self):
    return self.numero

  @numero.setter
  def numero(self, numero):
    self.__numero = numero

  @property
  def saldo_cuenta(self):
    return self.__saldo_cuenta

  @saldo_cuenta.setter
  def saldo_cuenta(self, saldo_cuenta):
    self.__saldo_cuenta = saldo_cuenta

```

```
23    @property
24    def tipo_cuenta(self):
25      return self.__tipo_cuenta
26
27    @tipo_cuenta.setter
28    def tipo_cuenta(self, tipo_cuenta):
29      self.__tipo_cuenta = tipo_cuenta
```

1.2.2. Interfaces

ICrud.py

```
1  from abc import ABC, abstractmethod
2
3  class ICrud(ABC):
4
5    @abstractmethod
6    def crear(self, **kwargs):
7      raise NotImplementedError
8
9    @abstractmethod
10   def mostrar(self, **kwargs):
11     raise NotImplementedError
12
13   @abstractmethod
14   def relacion(self, **kwargs):
15     raise NotImplementedError
```

1.2.3. Interfaces Implementadas

ImpCrudCliente.py

```
1  from InterfacesProyecto2.model import Cliente as cliente
2  from InterfacesProyecto2.crud import ICrud
3
4  class ImpCrudCliente(ICrud.ICrud):
5
6    def crear(self, **kwargs):
7      return cliente.Cliente(kwargs['nombre'])
8
9    def relacion(self, **kwargs):
10     cliente.cuentas = kwargs['muchos']
11     return cliente
```

ImpCrudCuenta.py

```
from InterfacesProyecto2.model import Cuenta as cuenta
from InterfacesProyecto2.crud import ICrud


class ImpCrudCuenta(ICrud.ICrud):

  def crear(self, **kwargs):
    return cuenta.Cuenta(kwargs['numero'], kwargs['saldo_cuenta'],
   kwargs['tipo_cuenta'])
```

1.2.4. Interfacez de Usuario

mensajes.py

```
def cabecera_de_pantalla():
  str = """
  ------------------------------------------------------
  Bienvenido al Sistema de Gestion de Clientes del Banco
  ------------------------------------------------------
  """
  print(str)

def menu_principal():
  str = """
  1. crear cliente
  """
  print(str)
  opcion = input()
  return opcion
```

1.2.5. Programa Principal

main.py

```
from InterfacesProyecto2.ui import mensajes
from InterfacesProyecto2.crud import ImpCrudCliente
from InterfacesProyecto2.crud import ImpCrudCuenta

baseClientes = []

mensajes.cabecera_de_pantalla()
opcion = mensajes.menu_principal()
crud_cuenta = ImpCrudCuenta.ImpCrudCuenta()
crud_cliente = ImpCrudCliente.ImpCrudCliente()

if opcion == '1':
  cliente = crud_cliente.crear(nombre = "William Wallace")
  cuenta = crud_cuenta.crear(numero = 123, saldo_cuenta = 300,
   tipo_cuenta = 'ahorro')
  crud_cliente.relacion(muchos = cuenta)

  baseClientes.append(cliente)
```

2. Hospital - Doctor

2.1. Arquitectura de Software

Usted ha sido contratado para implementar un *Sistema de Información Hospitalaria* que se encuentra en su fase inicial. De modo que hasta el momento solo se utilizarán las tablas, *Hospital* y *Doctor*. El esquema de la base de datos es el siguiente:

Figura 2: Esquema Entidad/Relación Sistema de Información Hospitalaria.

También, desde una perspectiva de *Programación Orientada a Objetos* el esquema de la base de datos (Figura 2) se puede representar de la siguiente manera:

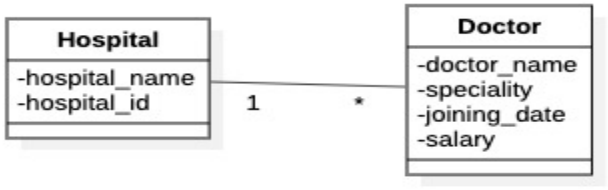


Figura 3: Modelo Sistema de Información Hospitalaria.

Dentro de la Arquitectura de Software empleada se encuentra el Componente *Persistence*, que es el encargado de realizar la conexión con el motor de base de datos *Postgres*. Como se observa en la Figura 4 el componente *Main* y *Persistence* se comunican a través de la interfaz *ICrud*. De esta forma se logra el principio de *desacople* y por lo tanto la *modularidad* en la Arquitectura.

En la Figura 5 se observar el *Diagrama de Clases* con la *Interfaz Implementada del Crud* que permite hacer uso de este por el componente *Main*. Se nota la implementación de la función *obtener_por_id()* por las clases *ImpCrudHospital* y *ImpCrudDoctor* aplicando el concepto de *Polimorfismo*.

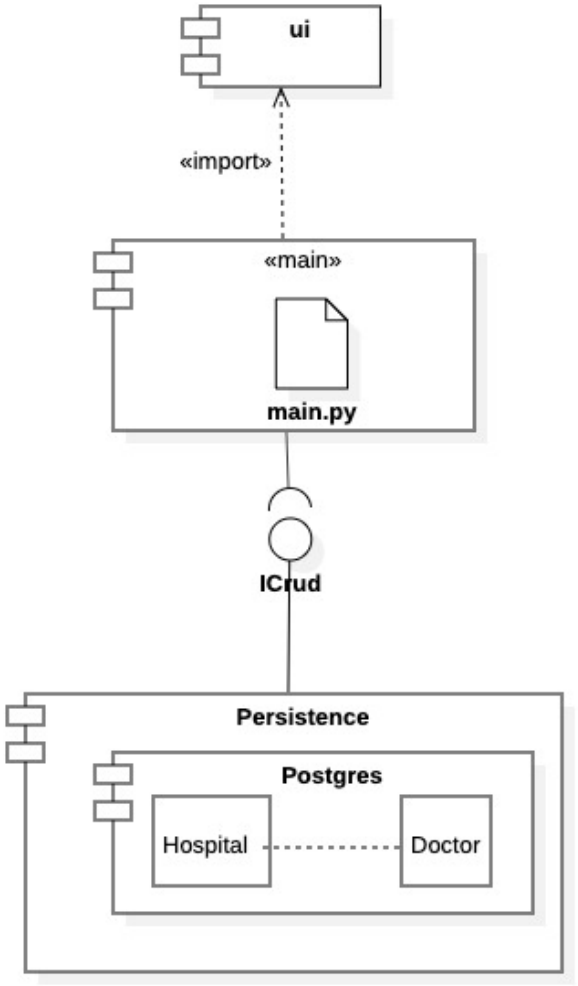


Figura 4: Arquitectura del Sistema de Información Hospitalaria.

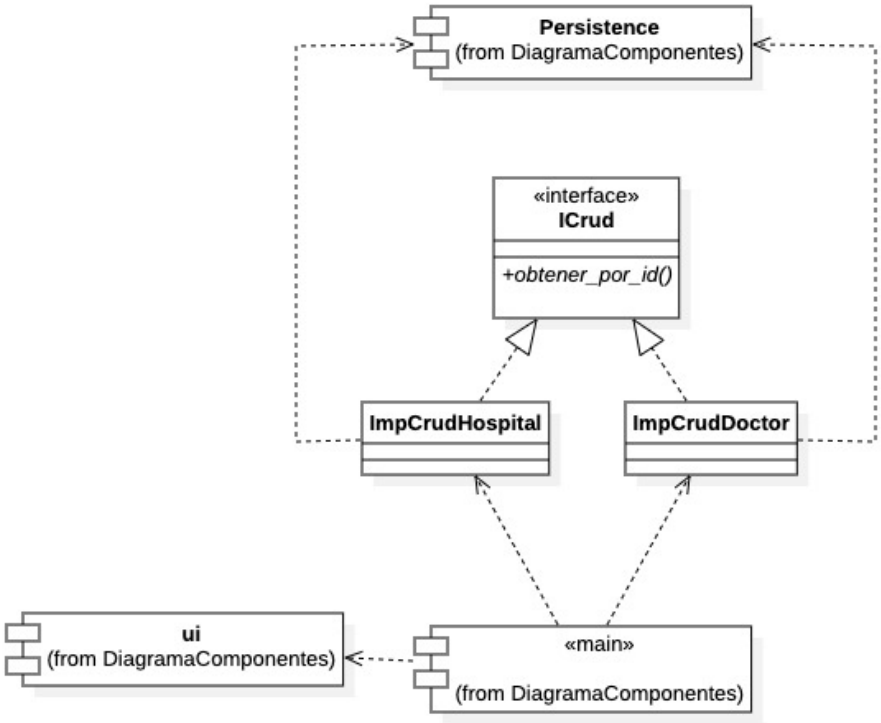


Figura 5: Arquitectura del Sistema de Información Hospitalaria (Diagrama de Clases - Interfaz Implementada).

2.2. Código Fuente

2.2.1. Modelo: Esquema de la Base de Datos

```sql
CREATE database hospital_db;

CREATE TABLE Hospital (
  Hospital_Id serial NOT NULL PRIMARY KEY,
  Hospital_Name VARCHAR (100) NOT NULL,
  Bed_Count serial
);

INSERT INTO Hospital (Hospital_Id, Hospital_Name, Bed_Count)
VALUES
('1', 'Mayo Clinic', 200),
('2', 'Cleveland Clinic', 400),
('3', 'Johns Hopkins', 1000),
('4', 'UCLA Medical Center', 1500);

CREATE TABLE Doctor (
  Doctor_Id serial NOT NULL PRIMARY KEY,
  Doctor_Name VARCHAR (100) NOT NULL,
  Hospital_Id serial NOT NULL,
  Joining_Date DATE NOT NULL,
  Speciality VARCHAR (100) NOT NULL,
  Salary INTEGER NOT NULL,
  Experience SMALLINT
);

INSERT INTO Doctor (Doctor_Id, Doctor_Name, Hospital_Id, Joining_Date,
    Speciality, Salary, Experience)
VALUES
('101', 'David', '1', '2005-2-10', 'Pediatric', '40000', NULL),
('102', 'Michael', '1', '2018-07-23', 'Oncologist', '20000', NULL),
('103', 'Susan', '2', '2016-05-19', 'Garnacologist', '25000', NULL),
('104', 'Robert', '2', '2017-12-28', 'Pediatric ', '28000', NULL),
('105', 'Linda', '3', '2004-06-04', 'Garnacologist', '42000', NULL),
('106', 'William', '3', '2012-09-11', 'Dermatologist', '30000', NULL),
('107', 'Richard', '4', '2014-08-21', 'Garnacologist', '32000', NULL),
('108', 'Karen', '4', '2011-10-17', 'Radiologist', '30000', NULL);
```

2.2.2. Interfaces

ICrud.py

```python
from abc import ABC, abstractmethod

class ICrud(ABC):

  @abstractmethod
  def obtener_por_id(self, **kwargs):
    raise NotImplementedError
```

2.2.3. Interfaces Implementadas

ImpCrudDoctor.py

```
import psycopg2
from ProyecBD_Interfaces_Hospital.persistence import ICrud

class ImpCrudDoctor(ICrud.ICrud):

  def get_connection(self):
      connection = psycopg2.connect(user="alejo",
                                    password="alejo",
                                    host="127.0.0.1",
                                    port="5432",
                                    database="hospital_db")
      print("Conexi n de Postgres est  abierta")
      return connection

  def close_connection(self, connection):
    if connection:
      connection.close()
      print("La conexi n de Postgres est  cerrada")

  def obtener_por_id(self, **kwargs):
    try:
      connection = self.get_connection()
      cursor = connection.cursor()
      select_query = 'SELECT * FROM "Doctor" WHERE Doctor_Id = %s'
      cursor.execute(select_query, (kwargs['doctor_id'],))
      records = cursor.fetchall()
      self.close_connection(connection)
    except (Exception, psycopg2.Error) as error:
      print("Error mientras se establece conexi n", error)

    return records
```

ImpCrudHospital.py

```
import psycopg2
from ProyecBD_Interfaces_Hospital.persistence import ICrud

class ImpCrudHospital(ICrud.ICrud):

  def get_connection(self):
      connection = psycopg2.connect(user="alejo",
                                    password="alejo",
                                    host="127.0.0.1",
                                    port="5432",
                                    database="hospital_db")
      print("Conexi n de Postgres est  abierta")
      return connection

  def close_connection(self, connection):
    if connection:
      connection.close()
      print("La conexi n de Postgres est  cerrada")

  def obtener_por_id(self, **kwargs):
    try:
      connection = self.get_connection()
      cursor = connection.cursor()
      select_query = 'SELECT * FROM "Hospital" WHERE hospital_id = %s'
      cursor.execute(select_query, (kwargs['hospital_id'],))
      records = cursor.fetchall()
      self.close_connection(connection)
    except (Exception, psycopg2.Error) as error:
      print("Error mientras se establece conexi n", error)

    return records
```

2.2.4. Interfaz de Usuario

mensajes.py

```
def cabecera_de_pantalla():
  str = """
  ------------------------------------------------------
  Sistema de Gestion de Hospital
  ------------------------------------------------------
  """
  print(str)

def menu_principal():
  str = """
  1. Obtener informacion del Hospital
  2. Obtener informacion de la Doctor
  """
  print(str)
  opcion = input()
  return opcion

def ingresar_hospital_id():
  str = """
    Ingresar Hospital Id:
    """
  print(str)
  return  input()

def ingresar_doctor_id():
  str = """
    Ingresar Doctor Id:
    """
  print(str)
  return  input()

def mostrar_informacion_hospital(registros):
  print("-----------------------")
  print("Informaci n del Hospital")
  print("-----------------------")
  for fila in registros:
    print("Hospital Id: ", fila[0] )
    print("Nombre: ", fila[1])

def mostrar_informacion_doctor(registros):
  print("-----------------------")
  print("Informaci n del Doctor")
  print("-----------------------")
  for fila in registros:
    print("N mero Doctor: ", fila[1])
    print("Nombre: ", fila[2])
    print("Salario: ", fila[6])
```

2.2.5. Programa Principal

main.py

```
from ProyecBD_Interfaces_Hospital.ui import mensajes
from ProyecBD_Interfaces_Hospital.persistence import ImpCrudHospital
from ProyecBD_Interfaces_Hospital.persistence import ImpCrudDoctor


crud_hospital = ImpCrudHospital.ImpCrudHospital()
crud_doctor = ImpCrudDoctor.ImpCrudDoctor()

mensajes.cabecera_de_pantalla()
opcion = mensajes.menu_principal()

if opcion == '1': #Informacion Hospital
  identificacion_hospital = mensajes.ingresar_hospital_id()
  registros = crud_hospital.obtener_por_id(hospital_id =
   identificacion_hospital)
  mensajes.mostrar_informacion_hospital(registros)

elif opcion == '2': #Informaci n Doctor
  identificacion_doctor = mensajes.ingresar_doctor_id()
  registros = crud_doctor.obtener_por_id(doctor_id =
   identificacion_doctor)
  mensajes.mostrar_informacion_doctor(registros)
```