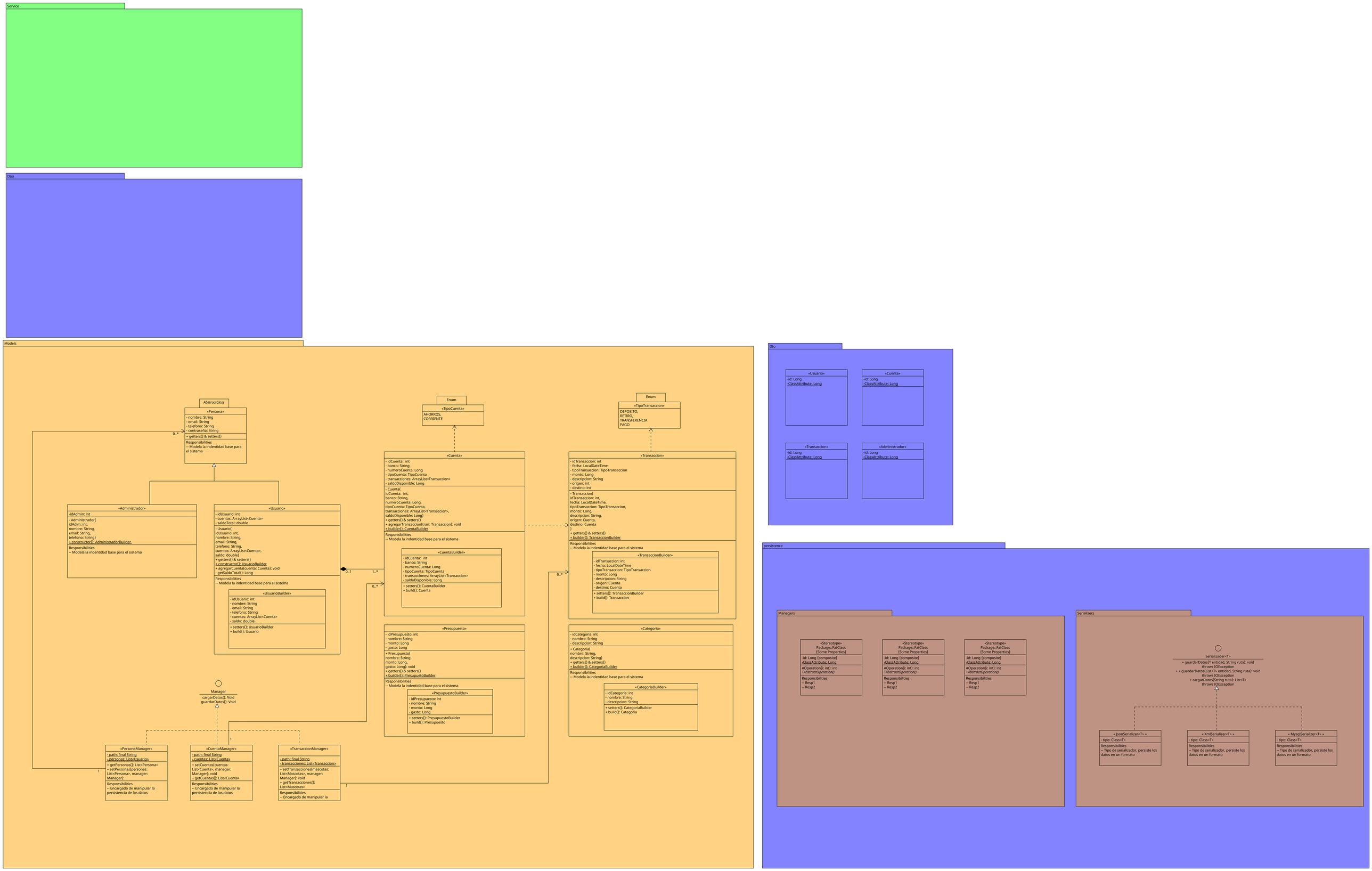
Service
Dao
Models
AbstractClass
«Persona»
- nombre: String
- email: String
- telefono: String
- contraseña: String
+ getters() & setters()
Responsibilities
-- Modela la indentidad base para el sistema
0..*
1
«Administrador»
-idAdmin: int
- Administrador(
idAdim: int,
nombre: String,
email: String,
telefono: String)
+ constructor(): AdministradorBuilder
Responsibilities
-- Modela la indentidad base para el sistema
«Usuario»
- idUsuario: int
- cuentas: ArrayList<Cuenta>
- saldoTotal: double
- Usuario(
idUsuario: int,
nombre: String,
email: String,
telefono: String,
cuentas: ArrayList<Cuenta>,
saldo: double)
+ getters() & setters()
+ constructor(): UsuarioBuilder
+ agregarCuenta(cuenta: Cuenta): void
- getSaldoTotal(): Long
Responsibilities
-- Modela la indentidad base para el sistema
«UsuarioBuilder»
- idUsuario: int
- nombre: String
- email: String
- telefono: String
- cuentas: ArrayList<Cuenta>
- saldo: double
+ setters(): UsuarioBuilder
+ build(): Usuario
0..1
1..*
0..*
Enum
«TipoCuenta»
AHORROS,
CORRIENTE
«Cuenta»
- idCuenta: int
- banco: String
- numeroCuenta: Long
- tipoCuenta: TipoCuenta
- transacciones: ArrayList<Transaccion>
- saldoDisponible: Long
- Cuenta(
idCuenta: int,
banco: String,
numeroCuenta: Long,
tipoCuenta: TipoCuenta,
transacciones: ArrayList<Transaccion>,
saldoDisponible: Long)
+ getters() & setters()
+ agregarTransaccion(tran: Transaccion): void
+ builder(): CuentaBuilder
Responsibilities
-- Modela la indentidad base para el sistema
«CuentaBuilder»
- idCuenta: int
- banco: String
- numeroCuenta: Long
- tipoCuenta: TipoCuenta
- transacciones: ArrayList<Transaccion>
- saldoDisponible: Long
+ setters(): CuentaBuilder
+ build(): Cuenta
«Presupuesto»
- idPresupuesto: int
- nombre: String
- monto: Long
- gasto: Long
+ Presupuesto(
nombre: String
monto: Long,
gasto: Long): void
+ getters() & setters()
+ builder(): PresupuetoBuilder
Responsibilities
-- Modela la indentidad base para el sistema
«PresupuestoBuilder»
- idPresupuesto: int
- nombre: String
- monto: Long
- gasto: Long
+ setters(): PresupuestoBuilder
+ build(): Presupuesto
Enum
«TipoTransaccion»
DEPOSITO,
RETIRO,
TRANSFERENCIA
PAGO
«Transaccion»
- idTransaccion: int
- fecha: LocalDateTime
- tipoTransaccion: TipoTransaccion
- monto: Long
- descripcion: String
- origen: int
- destino: int
- Transaccion(
idTransaccion: int,
fecha: LocalDateTime,
tipoTransaccion: TipoTransaccion,
monto: Long,
descripcion: String,
origen: Cuenta,
destino: Cuenta
)
+ getters() & setters()
+ builder(): TransaccionBuilder
Responsibilities
-- Modela la indentidad base para el sistema
«TransaccionBuilder»
- idTransaccion: int
- fecha: LocalDateTime
- tipoTransaccion: TipoTransaccion
- monto: Long
- descripcion: String
- origen: Cuenta
- destino: Cuenta
+ setters(): TransaccionBuilder
+ build(): Transaccion
«Categoria»
- idCategoria: int
- nombre: String
- descripcion: String
+ Categoria(
nombre: String,
descripcion: String)
+ getters() & setters()
+ builder(): CategoriaBuilder
Responsibilities
-- Modela la indentidad base para el sistema
«CategoriaBuilder»
- idCategoria: int
- nombre: String
- descripcion: String
+ setters(): CategoriaBuilder
+ build(): Categoria
0..*
Manager
cargarDatos(): Void
guardarDatos(): Void
«PersonaManager»
- path: final String
- personas: List<Usuario>
+ getPersonas(): List<Persona>
+ setPersonas(personas: List<Persona>, manager: Manager):
Responsibilities
-- Encargado de manipular la persistencia de los datos
«CuentaManager»
- path: final String
- cuentas: List<Cuenta>
+ setCuentas(cuentas: List<Cuenta>, manager: Manager): void
+ getCuentas(): List<Cuenta>
Responsibilities
-- Encargado de manipular la persistencia de los datos
«TransaccionManager»
- path: final String
- transacciones: List<Transaccion>
+ setTransacciones(mascotas: List<Mascotas>, manager: Manager): void
+ getTransacciones(): List<Mascotas>
Responsibilities
-- Encargado de manipular la
1
1
Dto
«Usuario»
-id: Long
-ClassAttribute: Long
«Cuenta»
-id: Long
-ClassAttribute: Long
«Transaccion»
-id: Long
-ClassAttribute: Long
«Administrador»
-id: Long
-ClassAttribute: Long
persistence
Managers
«Stereotype»
Package::FatClass
(Some Properties)
-id: Long {composite}
-ClassAttribute: Long
#Operation(i: int): int
+AbstractOperation()
Responsibilities
-- Resp1
-- Resp2
«Stereotype»
Package::FatClass
(Some Properties)
-id: Long {composite}
-ClassAttribute: Long
#Operation(i: int): int
+AbstractOperation()
Responsibilities
-- Resp1
-- Resp2
«Stereotype»
Package::FatClass
(Some Properties)
-id: Long {composite}
-ClassAttribute: Long
#Operation(i: int): int
+AbstractOperation()
Responsibilities
-- Resp1
-- Resp2
Serializers
Serializader<T>
+ guardarDatos(T entidad, String ruta): void
throws IOException
+ + guardarDatos(List<T> entidad, String ruta): void
throws IOException
+ cargarDatos(String ruta): List<T>
throws IOException
« JsonSerializer<T> »
- tipo: Class<T>
Responsibilities
-- Tipo de serializador, persiste los datos en un formato
« XmlSerializer<T> »
- tipo: Class<T>
Responsibilities
-- Tipo de serializador, persiste los datos en un formato
« MysqlSerializer<T> »
- tipo: Class<T>
Responsibilities
-- Tipo de serializador, persiste los datos en un formato