Presentación Grupo 21

IIC2113 - Diseño Detallado de Software

Profesora: María Fernanda Sepúlveda

Entrega 1

- **POCO MODULARIZADO**
- **MUCHOS ATRIBUTOS POR CLASE**
- **ACOPLAMIENTO**
- **MALAS PRÁCTICAS**

No estaba considerada una cuenta para el usuario, todo estaba pensado como atributos, los cuales probablemente habrían sido confusos de identificar en el futuro

Entrega 2

- **MODULARIZAR**
- **SEPARAR EN MÁS CLASES**
- **MENOS DEPENDENCIA ENTRE CLASES**
- **REFACTORING CONSTANTE**

Creamos clases para las distintas cuentas, además de separar objetos para transacciones y para inversiones en fondos. De esta forma se nos hizo mucho más fácil trabajar y no confundir los distintos componentes.

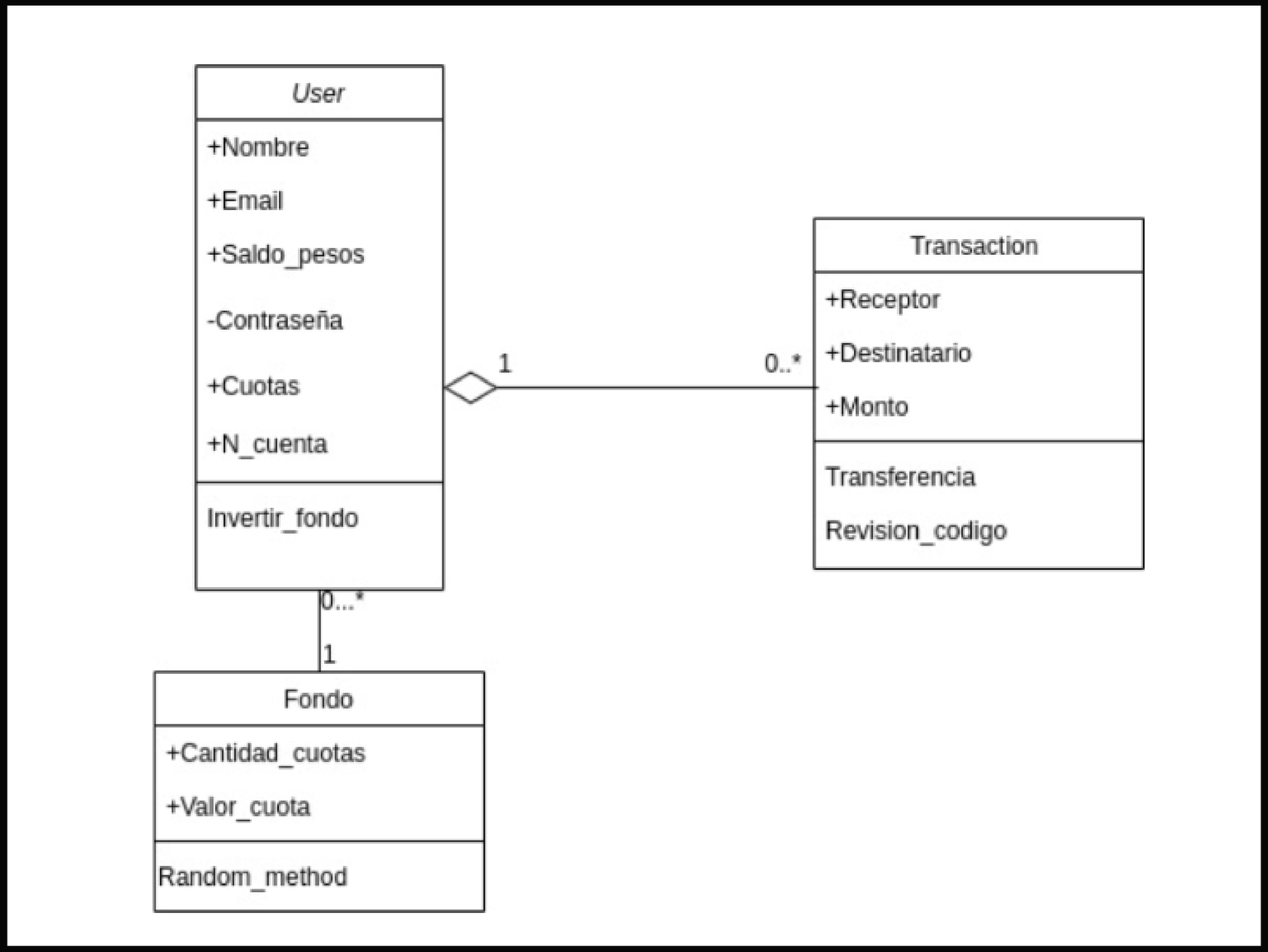
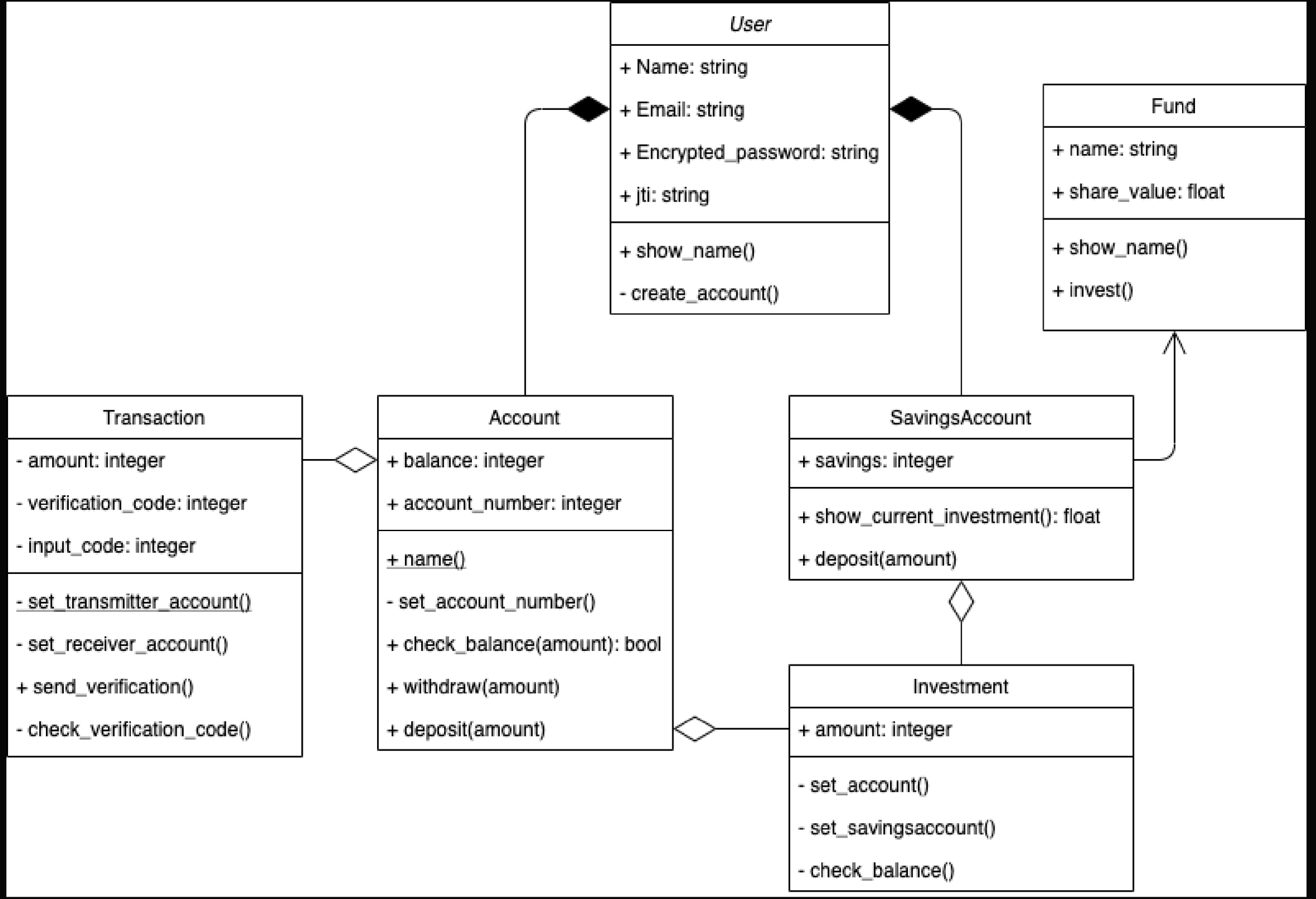


Diagrama Entrega 1


User
+ Name: string
+ Email: string
+ Encrypted_password: string
+ jti: string
+ show_name()
- create_account()
Fund
+ name: string
+ share_value: float
+ show_name()
+ invest()
Transaction
- amount: integer
- verification_code: integer
- input_code: integer
- set_transmitter_account()
- set_receiver_account()
+ send_verification()
- check_verification_code()
Account
+ balance: integer
+ account_number: integer
+ name()
- set_account_number()
+ check_balance(amount): bool
+ withdraw(amount)
+ deposit(amount)
SavingsAccount
+ savings: integer
+ show_current_investment(): float
+ deposit(amount)
Investment
+ amount: integer
- set_account()
- set_savingsaccount()
- check_balance()

SOLID - Interface Segregation Principle

Separamos la información en clases más pequeñas

```ruby
class Account < ApplicationRecord
    belongs_to :user, class_name: 'User'
    has_many :withdrawals, class_name: 'Transaction', foreign_key: 'emisor',
    has_many :deposits, class_name: 'Transaction', foreign_key: 'receptor',
    has_many :investments
```

```ruby
class SavingsAccount < ApplicationRecord
    belongs_to :fund
    belongs_to :user
    has_many :investments, class_name: 'Investment', foreign_key: 'emisor',
```

```ruby
class Investment < ApplicationRecord
    belongs_to :receiver, class_name: 'SavingsAccount'
    belongs_to :account
```

5S-Convención de nombres

Para evitar confusiones con el vocabulario relacionado a temas financieros (como fue mencionado en la charla de Buda)

```
  def withdrawal(amount)
    self.balance -= amount
    save
  end

  def deposit(amount)
    self.balance += amount
    save
  end
end
```

```
def set_transmitter_account
  @transmitter_account = Account.find
end

def set_receiver_account
  if Account.all.exists?(account_numb
```

Code Smells - Long Method

18 app/controllers/transactions_controller.rb

```
@@ -1,5 +1,21 @@
 class TransactionsController < InheritedResources::Base
-
+  def create
+    @transaction = Transaction.new(transaction_params)
+    if Account.exists?(@transaction.emisor_id)
+      @cuenta_emisora = Account.find(@transaction.emisor_id)
+      if @cuenta_emisora.balance >=  @transaction.amount
+        @transaction.save
+        @cuenta_receptora = Account.find(@transaction.receptor_id)
+        @cuenta_emisora.balance -= @transaction.amount
+        @cuenta_receptora.balance += @transaction.amount
+        @cuenta_emisora.save
+        @cuenta_receptora.save
+        redirect_to transactions_path
+      else
+        redirect_to home_index_path
+      end
+    end
+  end
   private

     def transaction_params
```

Code Smells - Long Method

```ruby
def create
  @transmitter_transaction = @transmitter_account.withdrawals.build(transaction_params)
  respond_to do |format|
    if @transmitter_transaction.save
      # subtracts money from account
      @transmitter_account.withdrawal(@transmitter_transaction.amount.to_i)
      # adds money to account
      @receiver_account.deposit(@transmitter_transaction.amount.to_i)
      format.html do
        redirect_to home_index_path,
                    notice: "Transacción realizada exitosamente."
      end
    else
      format.html do
        redirect_to home_index_path,
                    notice: "No se pudo realizar la transacción."
      end
    end
  end
```

```ruby
def set_transmitter_account
  @transmitter_account = Account.find(params[:transaction][:emisor_id])
end

def set_receiver_account
  if Account.all.exists?(account_number: params[:transaction][:receptor_id])
    @receiver_account = Account.find_by(account_number: params[:transaction][:receptor_id])
    params[:transaction][:receptor_id] = @receiver_account.id
  else
    flash[:notice] = "No existe una cuenta con ese número"
    redirect_to home_index_path
  end
end

def check_balance
  return if @transmitter_account.check_balance(params[:transaction][:amount].to_i)

  flash[:notice] = "No tienes suficiente dinero para realizar esa tranferencia."
  redirect_to home_index_path
end

def check_verification_code
  return if params[:transaction][:input_code] == params[:transaction][:verification_code]
  flash[:notice] = "Código de verificación incorrecto."
  redirect_to home_index_path
end
```

*Métodos ejecutados "before create"

Presentación Grupo 21

IIC2113 - Diseño Detallado de Software

Profesora: María Fernanda Sepúlveda