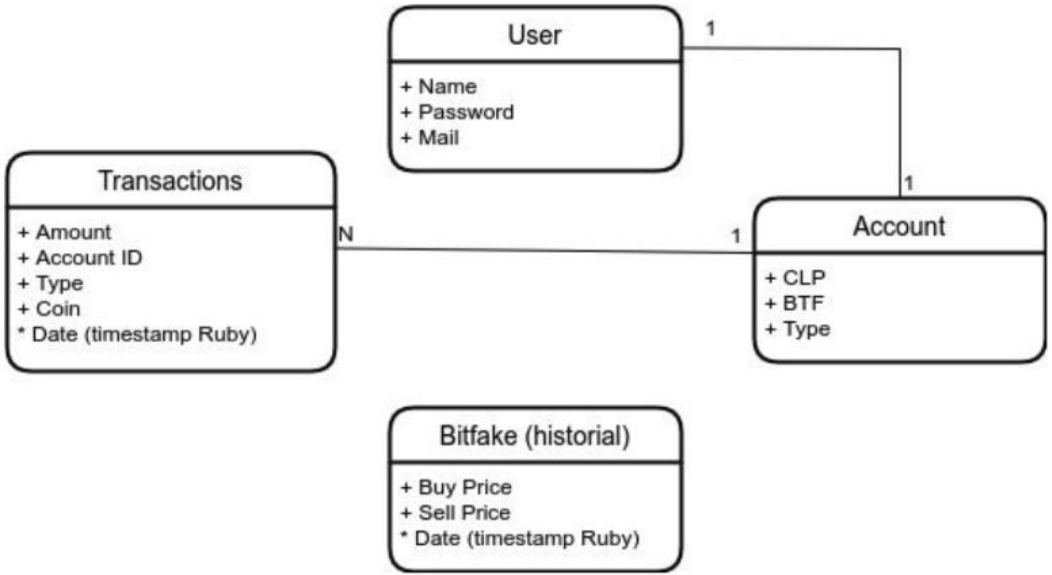
Diseño Detallado de Software

Grupo 12 Proyecto Semestral

Alejandra Uribe
Javiera Ochoa
Marcelo Riveros
Santiago Matus
Sebastián Montoya

Diagrama


User
+ Name
+ Password
+ Mail
1
1
Transactions
+ Amount
+ Account ID
+ Type
+ Coin
* Date (timestamp Ruby)
N
1
Account
+ CLP
+ BTF
+ Type
Bitfake (historial)
+ Buy Price
+ Sell Price
* Date (timestamp Ruby)

Inspección de Código

- Delegación de Tareas

```
def create
  @transaction = Transaction.new(transaction_params)
  respond_to do |format|
    if @transaction.save
      @user = current_user
      TransactionMailer.with(transaction: @transaction, user: @user).new_transaction_email.deliver_now
      format.html { redirect_to root_path, notice: 'Transacción realizada exitosamente' }
    else
      format.html { redirect_to root_path, alert: @transaction.errors.full_messages }
    end
  end
end
```

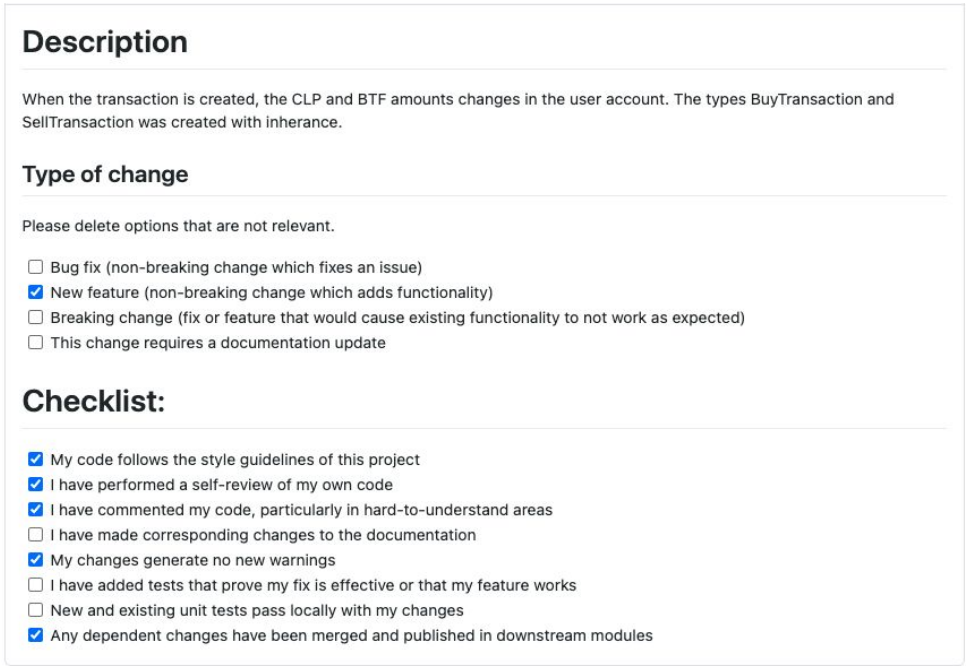
Inspección de Código

```ruby
class BuyTransaction < Transaction      javieraochoa, a month ago • ✨ buy and sell feature
  after_create :edit_account_amounts, :update_historic_price
    TYPE = 'BuyTransaction'.freeze

    private
    def edit_account_amounts
        #Here I set the new amounts. It asumes that the transaction was created correctly
        account = Account.find(self.account_id)
        price = Bitfake.last.buy_price
        clp = account.CLP - price*self.amount
        btf = account.BTF + self.amount
        account.update(CLP: clp, BTF: btf)
    end

    def update_historic_price
        #Here I update the price of BTF after every succesful transaction, in this case the price increases
        @buy_price = Bitfake.last.buy_price
        @sell_price = Bitfake.last.sell_price
        @sell_price = (1.03*@sell_price).round
        bitfake = Bitfake.create!(buy_price: @buy_price, sell_price: @sell_price)
        ActionCable.server.broadcast 'bitfake_channel', bitfake: bitfake
    end

end
```

- Uso de métodos privados
- Uso de Herencia
- Nombre de variables y funciones descriptivas

Buenas Prácticas

1. Uso de Template para Pull Requests



Description

When the transaction is created, the CLP and BTF amounts changes in the user account. The types BuyTransaction and SellTransaction was created with inherance.

Type of change

Please delete options that are not relevant.

- [ ] Bug fix (non-breaking change which fixes an issue)
- [x] New feature (non-breaking change which adds functionality)
- [ ] Breaking change (fix or feature that would cause existing functionality to not work as expected)
- [ ] This change requires a documentation update

Checklist:

- [x] My code follows the style guidelines of this project
- [x] I have performed a self-review of my own code
- [x] I have commented my code, particularly in hard-to-understand areas
- [ ] I have made corresponding changes to the documentation
- [x] My changes generate no new warnings
- [ ] I have added tests that prove my fix is effective or that my feature works
- [ ] New and existing unit tests pass locally with my changes
- [x] Any dependent changes have been merged and published in downstream modules

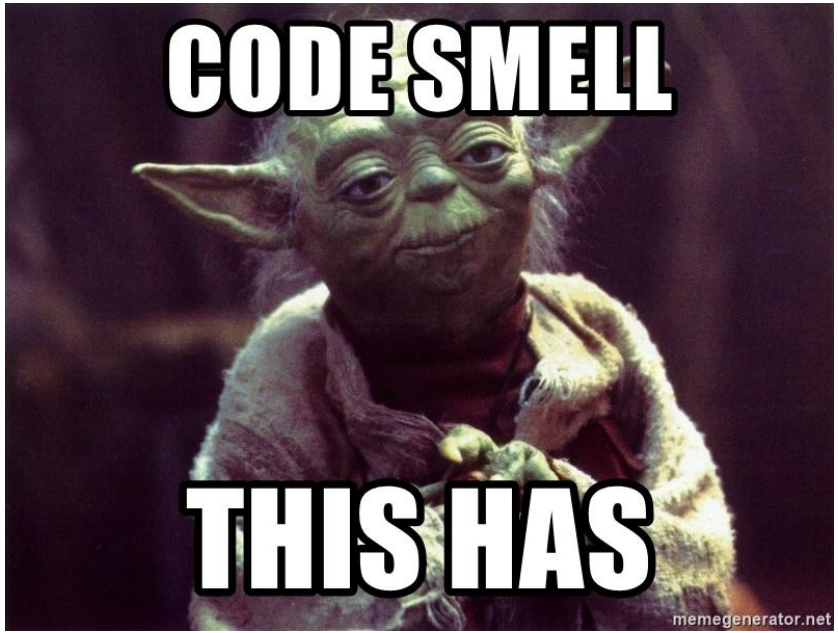
Buenas Prácticas

2. Pair Programming

Problemas

- Entender código legado
- Diferencias en cuanto a cómo habríamos modelado nosotros el problema.
- Replicación de Variables de Entorno

Problemas



- Evitar generar code smells
- Desarrollo más lento para validar con equipo.

Soluciones

- Consultar documentación de código legado.
- Sesiones de revisión de código para mejor comprensión.
- Idear soluciones a problemas en conjunto.
- Revisar posibles code smells en materia vista en el curso.
- Utilizar principios SOLID.
- Generar un canal de comunicación con el equipo responsable del código recibido.

Aprendizajes

- Importancia de una buena comunicación de equipo.
- Revisión de código para tener una mejora grupal en cuanto a buenas prácticas.
- Uso correcto de diagramas UML.
- Diseñar antes de desarrollar.
- Seguir principios SOLID y DRY.
- Uso de MockApi