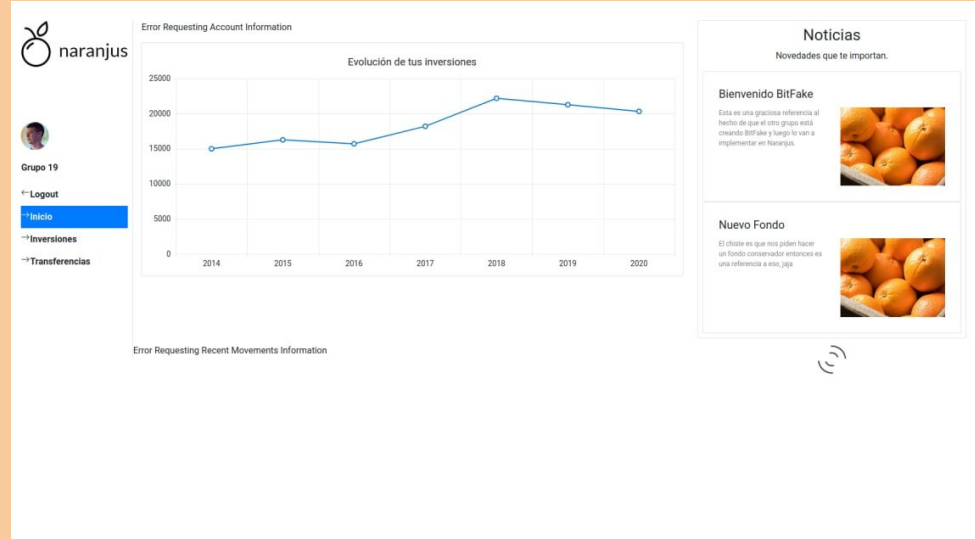


Grupo 19 - Diseño detallado de Software



naranjus

Grupo 19 - Entrega 2



Grupo 19 - Entrega 3

BITFAKE

[Profile](#) [tatanpoker09](#) [Log Out](#)

Balance: CLP\$ 343482

		Buy _(CLP)	Sell _(CLP)		
BTF	-0.72% 24h +6.32% 7d	300	200	<input type="button" value="Buy"/>	<input type="button" value="Sell"/>
BTH	+0.27% 24h +2.59% 7d	9000	8500	<input type="button" value="Buy"/>	<input type="button" value="Sell"/>

Estimated Total
CLP \$ 0

0% CLP

CLP\$ 0

0% BTF

BTF\$ 0

0% BTH

BTH\$ 0

Diagramas usados

Diagrama de Arquitectura

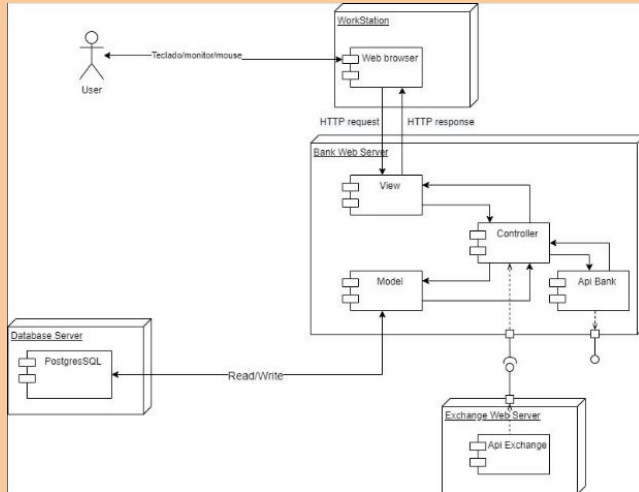
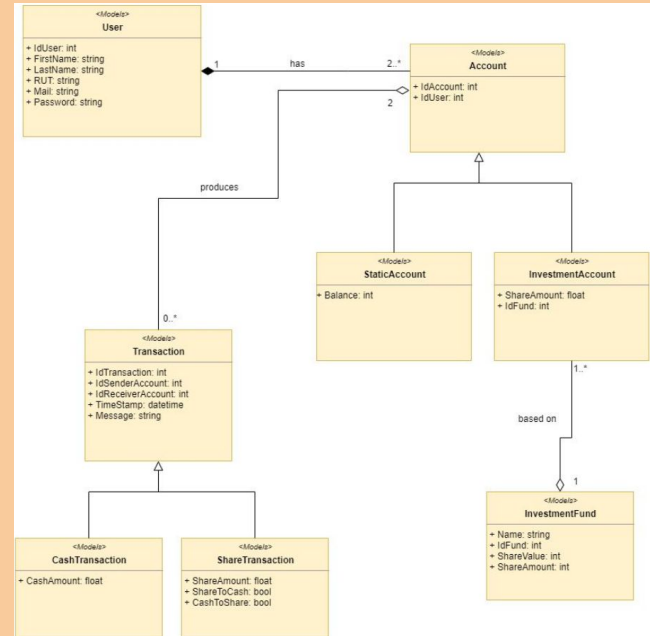
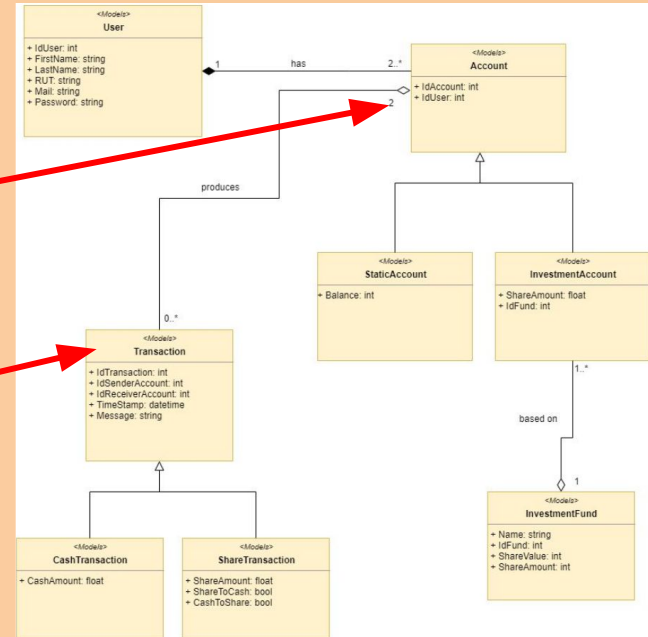
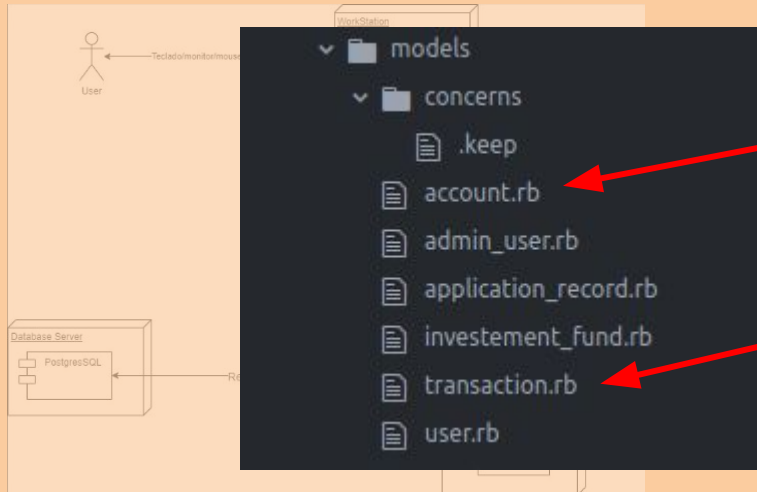


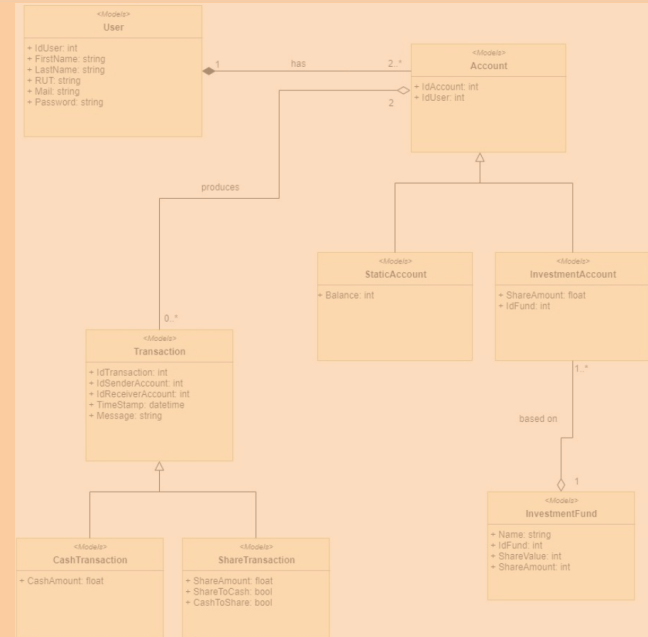
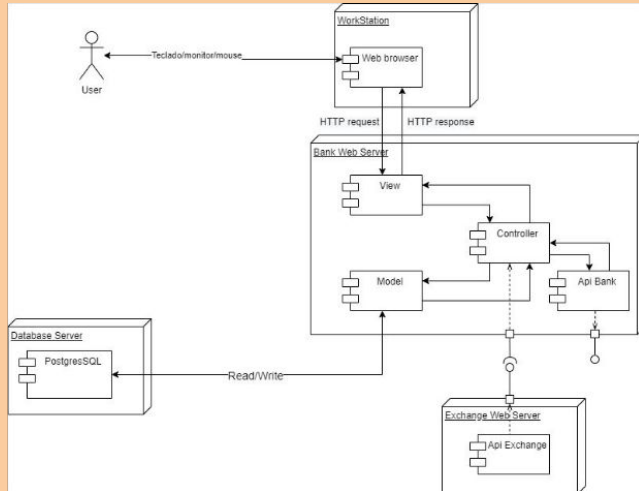
Diagrama de Clases



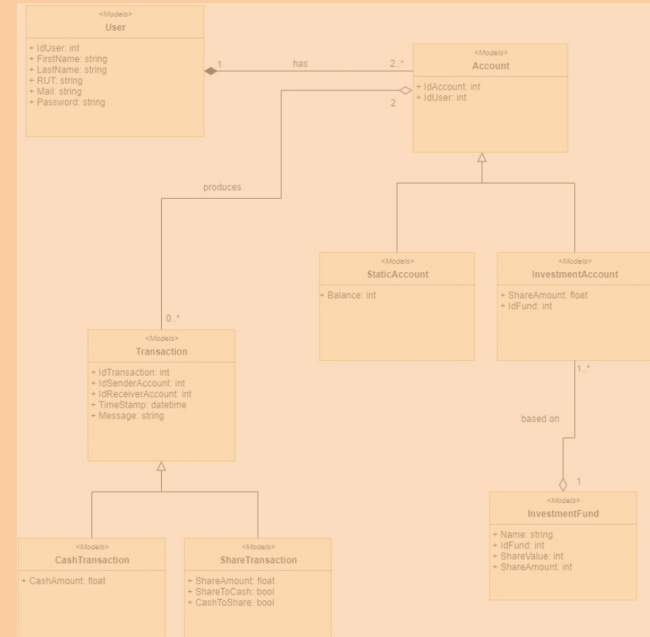
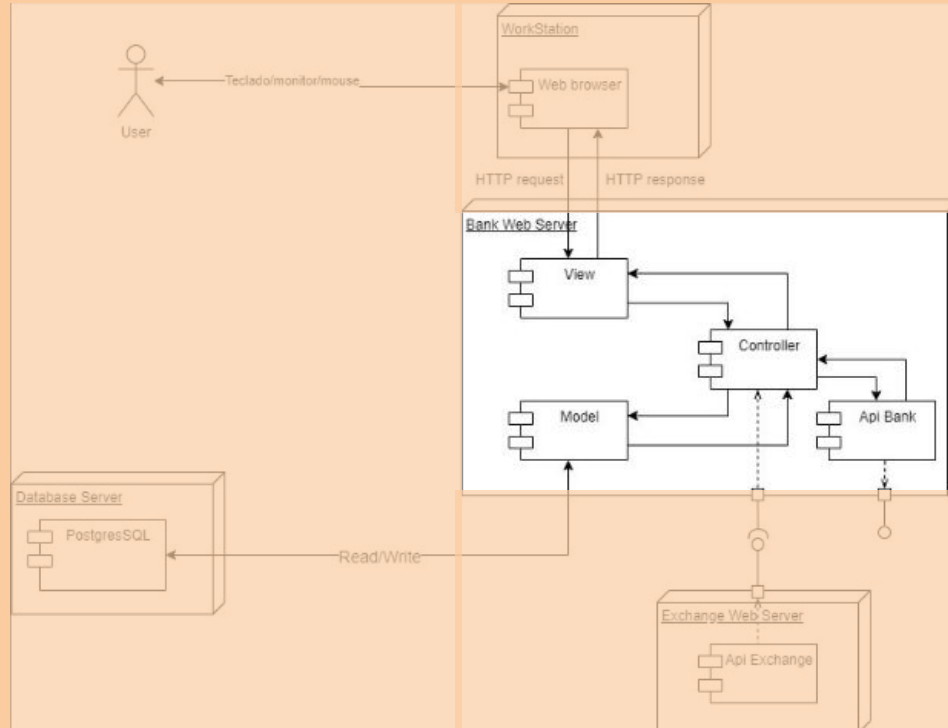
Diagramas usados



Diagramas usados



Diagramas usados



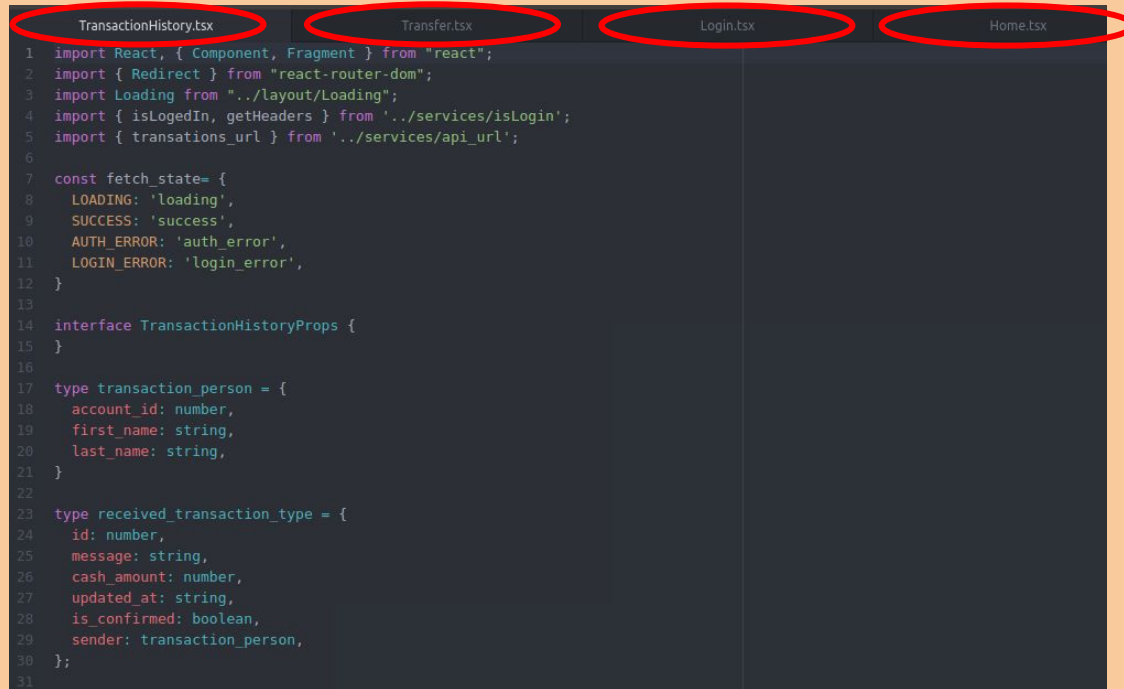
Ejemplos de buenas prácticas

Single-responsibility principle

```
TransactionHistory.tsx      Transfer.tsx      Login.tsx      Home.tsx
1  import React, { Component, Fragment } from "react";
2  import { Redirect } from "react-router-dom";
3  import Loading from "../layout/Loading";
4  import { isLoggedIn, getHeaders } from '../services/isLogin';
5  import { transactions_url } from '../services/api_url';
6
7  const fetch_state= {
8    LOADING: 'loading',
9    SUCCESS: 'success',
10   AUTH_ERROR: 'auth_error',
11   LOGIN_ERROR: 'login_error',
12 }
13
14 interface TransactionHistoryProps {
15 }
16
17 type transaction_person = {
18   account_id: number,
19   first_name: string,
20   last_name: string,
21 }
22
23 type received_transaction_type = {
24   id: number,
25   message: string,
26   cash_amount: number,
27   updated_at: string,
28   is_confirmed: boolean,
29   sender: transaction_person,
30 };
31
```


Ejemplos de buenas prácticas

Single-responsibility principle



```
1 import React, { Component, Fragment } from "react";
2 import { Redirect } from "react-router-dom";
3 import Loading from "../layout/Loading";
4 import { isLoggedIn, getHeaders } from '../services/isLogin';
5 import { transactions_url } from '../services/api_url';
6
7 const fetch_state= {
8   LOADING: 'loading',
9   SUCCESS: 'success',
10  AUTH_ERROR: 'auth_error',
11  LOGIN_ERROR: 'login_error',
12 }
13
14 interface TransactionHistoryProps {
15 }
16
17 type transaction_person = {
18   account_id: number,
19   first_name: string,
20   last_name: string,
21 }
22
23 type received_transaction_type = {
24   id: number,
25   message: string,
26   cash_amount: number,
27   updated_at: string,
28   is_confirmed: boolean,
29   sender: transaction_person,
30 };
31
```

Ejemplos de buenas prácticas

Api encapsulada

```
trxService.js
1  const authHeaders = {
2    "X-User-Email": localStorage.getItem("email"),
3    "X-User-Token": localStorage.getItem("token"),
4  };
5
6  export async function getSharePrice() {
7    const funds = await fetch(`${process.env.REACT_APP_SERVER_URL}/funds`, {
8      headers: authHeaders,
9    });
10   const funds_json = await funds.json();
11   let price = 0;
12   if (funds_json && funds_json.length) {
13     price = funds_json[0].share_value;
14   }
15   return price;
16 }
17
18 export async function getStaticAccount() {
19   let accounts = await fetch(`${process.env.REACT_APP_SERVER_URL}/accounts`, {
20     headers: authHeaders,
21   });
22   accounts = await accounts.json();
23   let account = accounts[0];
24   accounts.forEach((acc) => {
25     if (acc.is_static) account = acc;
26   });
27   account = await fetch(`${process.env.REACT_APP_SERVER_URL}/accounts/${account.id}`, {
28     headers: authHeaders,
29   });
30   account = await account.json();
31   return account;
32 }
33
34 export async function getInvestmentAccount() {
35   let accounts = await fetch(`${process.env.REACT_APP_SERVER_URL}/accounts`, {
36     headers: authHeaders,
37   });
38   accounts = await accounts.json();
39   let account = accounts[0];
40   accounts.forEach((acc) => {
41     if (!acc.is_static) account = acc;
42   });
43   account = await fetch(`${process.env.REACT_APP_SERVER_URL}/accounts/${account.id}`, {
44     headers: authHeaders,
45   });
46   account = await account.json();
47   return account;
48 }
```

Ejemplos de buenas prácticas

Api encapsulada

```
trxService.js
1  const authHeaders = {
2    "X-User-Email": localStorage.getItem("email"),
3    "X-User-Token": localStorage.getItem("token"),
4  };
5
6  export async function getSharePrice() {
7    const funds = await fetch(`${process.env.REACT_APP_SERVER_URL}/funds`, {
8      headers: authHeaders,
9    });
10   const funds_json = await funds.json();
11   let price = 0;
12   if (funds_json && funds_json.length) {
13     price = funds_json[0].share_value;
14   }
15   return price;
16 }
17
18 export async function getStaticAccount() {
19   let accounts = await fetch(`${process.env.REACT_APP_SERVER_URL}/accounts`, {
20     headers: authHeaders,
21   });
22   accounts = await accounts.json();
23   let account = accounts[0];
24   accounts.forEach((acc) => {
25     if (acc.is_static) account = acc;
26   });
27   account = await fetch(`${process.env.REACT_APP_SERVER_URL}/accounts/${account.id}`, {
28     headers: authHeaders,
29   });
30   account = await account.json();
31   return account;
32 }
33
34 export async function getInvestmentAccount() {
35   let accounts = await fetch(`${process.env.REACT_APP_SERVER_URL}/accounts`, {
36     headers: authHeaders,
37   });
38   accounts = await accounts.json();
39   let account = accounts[0];
40   accounts.forEach((acc) => {
41     if (!acc.is_static) account = acc;
42   });
43   account = await fetch(`${process.env.REACT_APP_SERVER_URL}/accounts/${account.id}`, {
44     headers: authHeaders,
45   });
46   account = await account.json();
47   return account;
48 }
```

Nos sentimos orgullosos de:

```
change-mine / exchange-mine / lib / tasks / update rake
Pablo, a day ago | 1 author (Pablo)
MAX_VARIATION = 0.3      Pablo, 2 days ago • Finished with update prices

# Taken from here: https://stackoverflow.com/questions/1711681/how-to-best-create-a-random-float-in-a-range-between-two-float
def range(min, max)
  rand * (max-min) + min
end

def get_specific_variation(currency, price)
  selected_price = currency[price]
  new_price = range(
    selected_price*(1-MAX_VARIATION),
    selected_price*(1+MAX_VARIATION)
  )
  currency[price] = new_price
end

def zero_correction(currency, price)
  if currency[price] < 0
    currency[price] = 0
  end
end

def get_variation(currency)
  get_specific_variation(currency, "buy_price")
  get_specific_variation(currency, "sell_price")

  zero_correction(currency, "buy_price")
  zero_correction(currency, "sell_price")

  if currency.buy_price < currency.sell_price
    average = (currency.buy_price + currency.sell_price) / 2
    currency.buy_price = average
    currency.sell_price = average
  end
  return currency
end

namespace :update do
  desc "-----Update the price randomly-----"
  task :price => :environment do

    to_update_btf = Btf.first
    to_update_bth = Bth.first

    if !to_update_btf.nil? && !to_update_bth.nil?
      get_variation(to_update_btf)
      to_update_btf.save

      get_variation(to_update_bth)
      to_update_bth.save
    end

    message = "---- Update Prices ----\nBTF: buy:%f sell:%f\nBTH: buy:%f sell:%f\n" % [
      to_update_btf.buy_price,
      to_update_btf.sell_price,
      to_update_bth.buy_price,
      to_update_bth.sell_price
    ]
    Rails.logger.info message
  else
    Rails.logger.info "---- Update Prices ----\nThere was an error in the currencies"
  end
end
end

end

namespace :update do
  desc "-----Update the price randomly-----"
  task :price => :environment do

    to_update_btf = Btf.first
    to_update_bth = Bth.first

    if !to_update_btf.nil? && !to_update_bth.nil?
      get_variation(to_update_btf)
      to_update_btf.save
    end
  end
end
```

```
36 end
37
38 namespace :update do
39   desc "-----Update the price randomly-----"
40   task :price => :environment do
41
42     to_update_btf = Btf.first
43     to_update_bth = Bth.first
44
45     if !to_update_btf.nil? && !to_update_bth.nil?
46       get_variation(to_update_btf)
47       to_update_btf.save
48
49       get_variation(to_update_bth)
50       to_update_bth.save
51
52       message = "---- Update Prices ----\nBTF: buy:%f sell:%f\nBTH: buy:%f sell:%f\n" % [
53         to_update_btf.buy_price,
54         to_update_btf.sell_price,
55         to_update_bth.buy_price,
56         to_update_bth.sell_price
57       ]
58       Rails.logger.info message
59     else
60       Rails.logger.info "---- Update Prices ----\nThere was an error in the currencies"
61     end
62   end
63 end
64
65 end
66
```

Nos sentimos orgullosos de:

```
change-mine > exchange-mine > lib > tasks > update rake
Pablo, a day ago | 1 author (Pablo)
MAX_VARIATION = 0.3      Pablo, 2 days ago • Finished with update prices

# Taken from here: https://stackoverflow.com/questions/1711681/how-to-best-create-a-random-float-in-a-range-between-two-floats
def range(min, max)
  rand * (max-min) + min
end

def get_specific_variation(currency, price)
  selected_price = currency[price]
  new_price = range(
    selected_price*(1-MAX_VARIATION),
    selected_price*(1+MAX_VARIATION)
  )
  currency[price] = new_price
end

def zero_correction(currency, price)
  if currency[price] < 0
    currency[price] = 0
  end
end

def get_variation(currency)
  get_specific_variation(currency, "buy_price")
  get_specific_variation(currency, "sell_price")

  zero_correction(currency, "buy_price")
  zero_correction(currency, "sell_price")

  if currency.buy_price < currency.sell_price
    average = (currency.buy_price + currency.sell_price) / 2
    currency.buy_price = average
    currency.sell_price = average
  end
  return currency
end

namespace :update do
  desc "-----Update the price randomly-----"
  task :price => :environment do

    to_update_btf = Btf.first
    to_update_bth = Bth.first

    if !to_update_btf.nil? && !to_update_bth.nil?
      get_variation(to_update_btf)
      to_update_btf.save
    end
  end
end
```

Separamos las funciones

```
36 end
37
38 namespace :update do
39   desc "-----Update the price randomly-----"
40
41   if !to_update_btf.nil? && !to_update_bth.nil?
42     get_variation(to_update_btf)
43     to_update_btf.save
44
45     get_variation(to_update_bth)
46     to_update_bth.save
47
48     message = "----- Update Prices ----- \nBTF: buy:%f sell:%f \nBTH: buy:%f sell:%f \n" % [
49       to_update_btf.buy_price,
50       to_update_btf.sell_price,
51       to_update_bth.buy_price,
52       to_update_bth.sell_price
53     ]
54     Rails.logger.info message
55   else
56     Rails.logger.info "----- Update Prices ----- \nThere was an error in the currencies"
57   end
58 end
59
60 end
```

Nos sentimos orgullosos de:

```
change-rails > exchange-rails > lib > tasks > update rake
Pablo, a day ago | 1 author (Pablo)
MAX_VARIATION = 0.3      Pablo, 2 days ago • Finished with update prices

# Taken from here: https://stackoverflow.com/questions/1711681/how-to-best-create-a-random-float-in-a-range-between-two-floats
def range(min, max)
  rand * (max-min) + min
end

def get_specific_variation(currency, price)
  selected_price = currency[price]
  new_price = range(
    selected_price*(1-MAX_VARIATION),
    selected_price*(1+MAX_VARIATION)
  )
  currency[price] = new_price
end

def zero_correction(currency, price)
  if currency[price] < 0
    currency[price] = 0
  end
end

def get_variation(currency)
  get_specific_variation(currency, "buy_price")
  get_specific_variation(currency, "sell_price")

  zero_correction(currency, "buy_price")
  zero_correction(currency, "sell_price")

  if currency.buy_price < currency.sell_price
    average = (currency.buy_price + currency.sell_price) / 2
    currency.buy_price = average
    currency.sell_price = average
  end
  return currency
end

namespace :update do
  desc "-----Update the price randomly-----"
  task :price => :environment do

    to_update_btf = Btf.first
    to_update_bth = Bth.first

    if !to_update_btf.nil? && !to_update_bth.nil?
      get_variation(to_update_btf)
      to_update_btf.save
    end
  end
end
```

Separamos las funciones

Programación defensiva

```
end

namespace :update do
  desc "-----Update the price randomly-----"

  if !to_update_btf.nil? && !to_update_bth.nil?
    get_variation(to_update_btf)
  end

  message = "----- Update Prices -----\\nBTF: buy:%f sell:%f\\nBTH: buy:%f sell:%f\\n" % [
    to_update_btf.buy_price,
    to_update_btf.sell_price,
    to_update_bth.buy_price,
    to_update_bth.sell_price
  ]
  Rails.logger.info message
else
  Rails.logger.info "----- Update Prices -----\\nThere was an error in the currencies"
end
end

end
```

Nos sentimos orgullosos de:

```
change-mine > exchange-mine > lib > tasks > update rake
Pablo, a day ago | 1 author (Pablo)
MAX_VARIATION = 0.3      Pablo, 2 days ago • Finished with update prices

# Taken from here: https://stackoverflow.com/questions/1711681/how-to-best-create-a-random-float-in-a-range-between-two-floats
def range(min, max)
  rand * (max-min) + min
end

def get_specific_variation(currency, price)
  selected_price = currency[price]
  new_price = range(
    selected_price*(1-MAX_VARIATION),
    selected_price*(1+MAX_VARIATION)
  )
  currency[price] = new_price
end

def zero_correction(currency, price)
  if currency[price] < 0
    currency[price] = 0
  end
end

def get_variation(currency)
  get_specific_variation(currency, "buy_price")
  get_specific_variation(currency, "sell_price")

  zero_correction(currency, "buy_price")
  zero_correction(currency, "sell_price")

  if currency.buy_price < currency.sell_price
    average = (currency.buy_price + currency.sell_price) / 2
    currency.buy_price = average
    currency.sell_price = average
  end
  return currency
end

namespace :update do
  desc "-----Update the price randomly-----"
  task :price => :environment do

    to_update_btf = Btf.first
    to_update_bth = Bth.first

    if !to_update_btf.nil? && !to_update_bth.nil?
      get_variation(to_update_btf)
      to_update_btf.save
    end
  end
end
```

Separamos las funciones

Programación defensiva

Fácil de entender

```
36   end
37
38   namespace :update do
39     desc "-----Update the price randomly-----"
40
41     if !to_update_btf.nil? && !to_update_bth.nil?
42       get_variation(to_update_btf)
43       to_update_btf.save
44     end
45
46     message = "----- Update Prices -----\\nBTF: buy:%f sell:%f\\nBTH: buy:%f sell:%f\\n" % [
47       to_update_btf.buy_price,
48       to_update_btf.sell_price,
49       to_update_bth.buy_price,
50       to_update_bth.sell_price
51     ]
52
53     Rails.logger.info "----- Update Prices -----\\nThere was an error in the currencies"
54   end
55 end
56
```

Nos sentimos orgullosos de:

```
change-mine / exchange-mine / lib / tasks / update rake
Pablo, a day ago | 1 author (Pablo)
MAX_VARIATION = 0.3      Pablo, 2 days ago • Finished with update prices

# Taken from here: https://stackoverflow.com/questions/1711681/how-to-best-create-a-random-float-in-a-range-between-two-floats
def range(min, max)
  rand * (max-min) + min
end

def get_specific_variation(currency, price)
  selected_price = currency[price]
  new_price = range(
    selected_price*(1-MAX_VARIATION),
    selected_price*(1+MAX_VARIATION)
  )
  currency[price] = new_price
end

def zero_correction(currency, price)
  if currency[price] < 0
    currency[price] = 0
  end
end

def get_variation(currency)
  get_specific_variation(currency, "buy_price")
  get_specific_variation(currency, "sell_price")

  zero_correction(currency, "buy_price")
  zero_correction(currency, "sell_price")

  if currency.buy_price < currency.sell_price
    average = (currency.buy_price + currency.sell_price) / 2
    currency.buy_price = average
    currency.sell_price = average
  end
  return currency
end

namespace :update do
  desc "-----Update the price randomly-----"
  task :price => :environment do

    to_update_btf = Btf.first
    to_update_bth = Bth.first

    if !to_update_btf.nil? && !to_update_bth.nil?
      get_variation(to_update_btf)
      to_update_btf.save
    end
  end
end
```

Separamos las funciones

Programación defensiva

Fácil de entender

Nombres descriptivos

```
36 end
37
38 namespace :update do
39   desc "-----Update the price randomly-----"
40
41   if !to_update_btf.nil? && !to_update_bth.nil?
42     get_variation(to_update_btf)
43
44     message = "----- Update Prices -----\\nBTF: buy:%f sell:%f\\nBTH: buy:%f sell:%f\\n" % [
45       to_update_btf.buy_price,
46       to_update_btf.sell_price,
47       to_update_bth.buy_price,
48       to_update_bth.sell_price
49     ]
50
51     Rails.logger.info "----- Update Prices -----\\nThere was an error in the currencies"
52   end
53 end
```


Lecciones Aprendidas

Setear proyecto en conjunto



Lecciones Aprendidas

Setear proyecto en conjunto



Lenguajes de programación
que todo el equipo domine



Lecciones Aprendidas

No setear bibliotecas innecesarias





naranjus

Muchas gracias