

# Presentación Final



IIC-2113

Integrantes:

Sebastián Bitar

Patricio de Solminihac

Vicente Lira

Sebastián Morales

Felipe Trejo

# **1-. Aspectos relevantes del trabajo en relación a la materia del curso**

# 1.0 Separación de responsabilidades

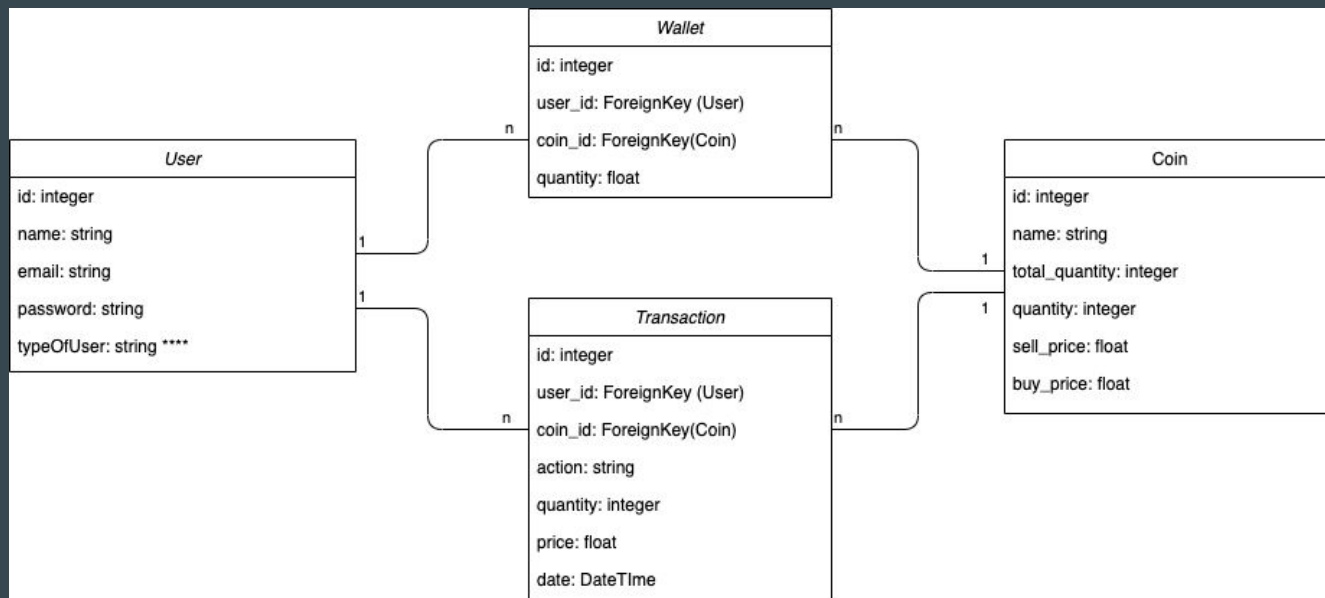
Debido a las cualidades que tenía el grupo como programadores que decidimos separar el equipo en dos.

Team BackEnd: Patricio de Solminihac, Sebastián Morales, Vicente Lira

Team FrontEnd: Sebastián Bitar, Felipe Trejo

# 1.1 Diagrama de Clases

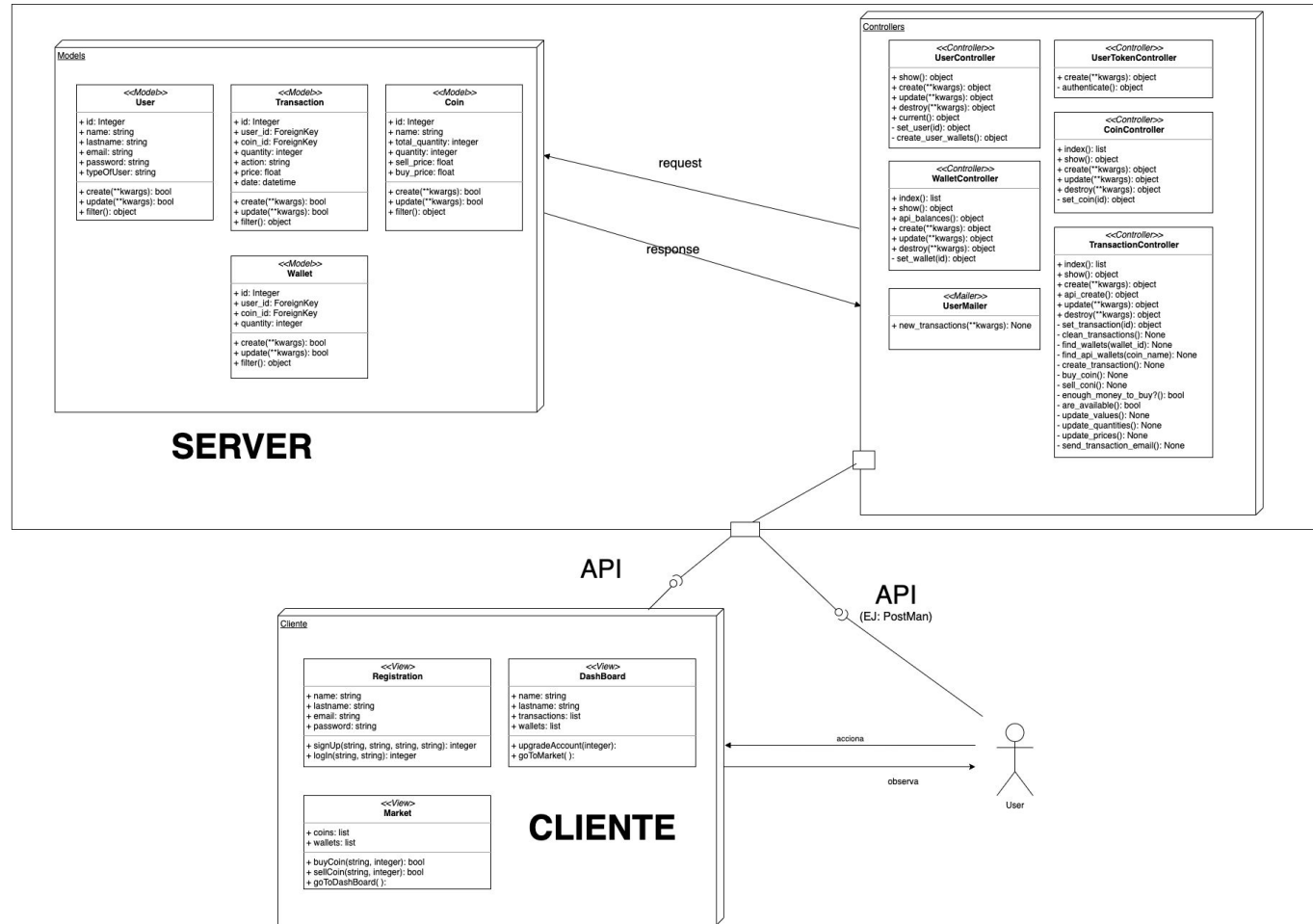
- Comprensión general del problema
- Cardinalidad de las instancias



## 1.2 Arquitectura del software

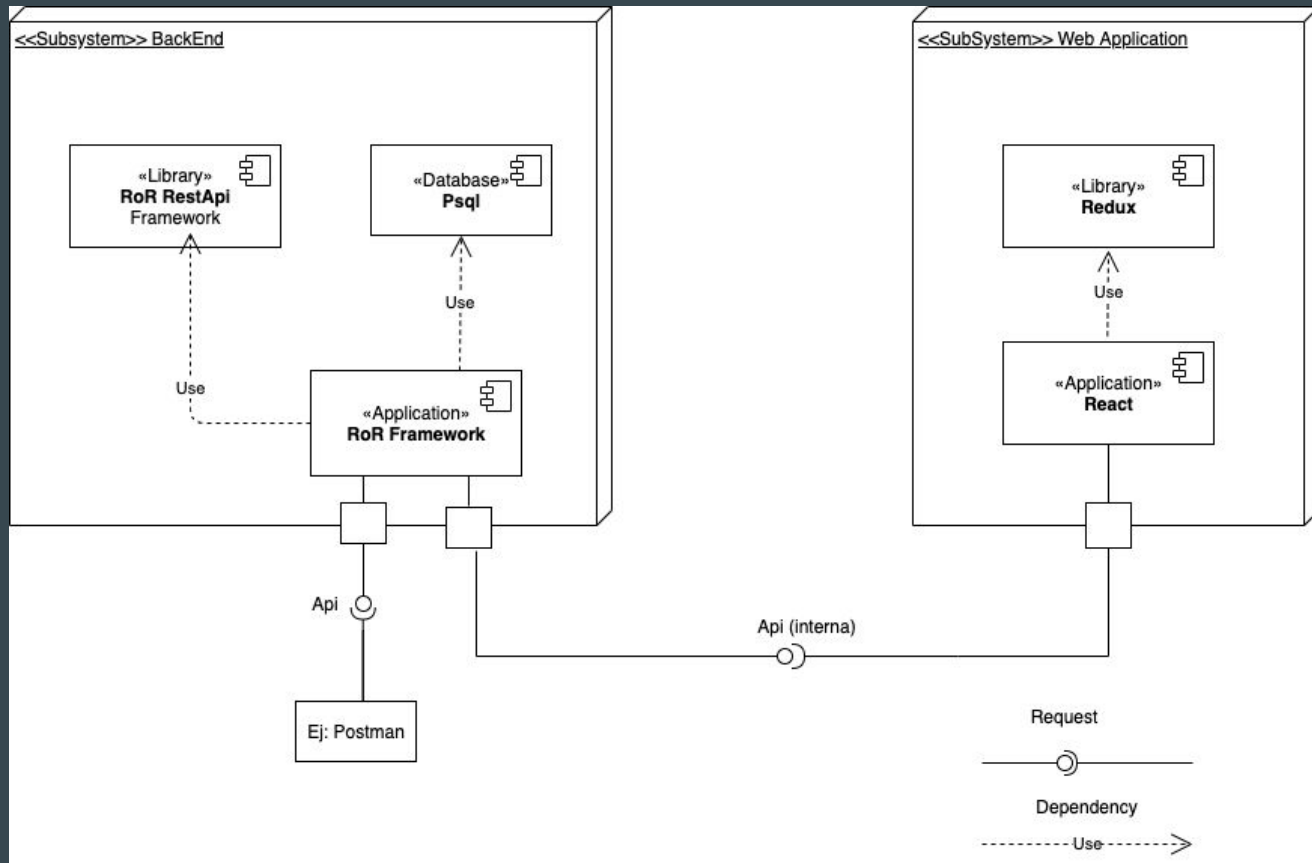
- Modelo-Vista-Controlador
- Arquitectura cliente y servidor

# Model - View - Controller



# 1.3 Diagrama de Componentes

- Separación Back-end del Front-end
  - Experiencia
  - Manejo de lenguajes
  - 2 repositorios
- Diferenciación del lenguaje
  - React
  - Ruby (RoR)










# 1.4 Nueva metodología de Commits

Hubo un gran avance en cuanto a cómo crear commits descriptivos que ayuden a ordenar el flujo de código entre los distintos participantes del grupo.

Ejemplo:

 log	Initial commit	4 months ago
 public	initial api commit	2 months ago
 storage	Initial commit	4 months ago



 docs	Merge branch 'development' into feat/wallets	last month
 lib/tasks	<a href="#">created single function to handle price variations and created rake t...</a>	3 days ago

# 1.5 Nueva metodología de Pull Request

Para cada Pull Request se necesitaba una aprobación de dos personas del grupo, las cuales tenían que revisar el código para poder “mergear” la rama. Junto a esto y para facilitar la corrección, es que se ocupaba una plantilla que describe lo que estaba.

Ejemplo:

```
## Type of change
```

```
Please delete options that are not relevant.
```

- ☐ Bug fix (non-breaking change which fixes an issue)
- ☐ 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:
```

- ☐ My code follows the style guidelines of this project
- ☐ I have performed a self-review of my own code
- ☐ I have commented my code, particularly in hard-to-understand areas
- ☐ I have made corresponding changes to the documentation
- ☐ 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
- ☐ Any dependent changes have been merged and published in downstream modules

## Ejemplo práctico 1



pdesolminihac on 15 Oct



Me parece que está al revés. Acá va el screenshot del enunciado:

- Si la transacción es venta, el precio del BTF de compra disminuye en 0.03 sobre el porcentaje de su valor.
- Si la transacción es compra, el precio del BTF de venta aumenta en 0.03 sobre el porcentaje de su valor.



stmorales on 15 Oct • edited Author



Cacha que la dudé cuándo leí jajaja pero es verdad.

Suggested change ⓘ

```
180 - if @action=='buy'
181 -   new_buy_price *= 1.03
182 - else
183 -   new_sell_price *= 0.97
184 - end
180 + if @action=='buy'
181 +   new_sell_price *= 1.03
182 + else
183 +   new_buy_price *= 0.97
184 + end
```



pdesolminihac on 16 Oct • edited



Léelo bien. Cuando se **vende**, el **precio de compra** cambia. Y cuando se **compra**, el **precio de venta** cambia.



stmorales on 16 Oct Author



Ahí sí. Toda la razón. Mala mía

## Ejemplo práctico 2

Feat/user creation #7



Merged



vjlira merged 18 commits into development from feat/user-creation on 18 Oct



app/controllers/users\_controller.rb Outdated Hide resolved

```
28 + @wallet_clp = @user.wallets.build(quantity: clp_random, user_id: @user.id, coin_id:1)
29 + if @wallet_clp.save
30 +   @wallet_eth = @user.wallets.build(quantity: 0, user_id: @user.id, coin_id:2)
31 +   if @wallet_eth.save
32 +     @wallet_bit = @user.wallets.build(quantity: 0, user_id: @user.id, coin_id:3)
33 +     if @wallet_bit.save
34 +       render json: @user, status: :created, location: @user
35 +       return
36 +     end
37 +   end
```

Comment on lines 28 to 37

 **stmorales** on 17 Oct  ...  
Toda esta parte la dejaría en un método aparte. Cada método tiene que tener una única función. Pondría un private method que se llame ``create\_clp\_wallet`` y que sea llamado el método en el create.

 **vjlira** on 18 Oct Author  ...  
perfecto, lo hago

 **stmorales** on 18 Oct  ...  
👍

```
64 +
65 + def create_user_wallets
66 +   clp_random = rand 10000..100000
67 +   @wallet_clp = @user.wallets.build(quantity: clp_random, user_id: @user.id, coin_id:1)
68 +   if @wallet_clp.save
69 +     @wallet_eth = @user.wallets.build(quantity: 0, user_id: @user.id, coin_id:2)
70 +     if @wallet_eth.save
71 +       @wallet_bit = @user.wallets.build(quantity: 0, user_id: @user.id, coin_id:3)
72 +       if @wallet_bit.save
73 +         render json: @user, status: :created, location: @user
74 +         return
75 +       end
76 +     end
77 +   end
78 +   render json: "Error with the accounts", status: :unprocessable_entity
79 + end
```

# 1.6 Flujo dentro de la aplicación

Flujo: SubmitBuy -> validateQuantity (“middleware”) -> sendRequest

```
const validateQuantity = (quantity) => {
  // Check if the quantity is an integer
  var msg
  if (!/^[0-9]+$/.test(quantity) && !Number.isInteger(quantity)) {
    msg = 'Quantity to buy should be an integer'
    return msg
  }
  // Check if the quantity is greater than 0.
  if (quantity <= 0) {
    msg = 'Quantity to buy should be more than 0'
    return msg
  }
  return
}

const submitBuy = () => {
  // First it is set the transfer action
  setBuyErrorMessage('')
  // Check if the quantity is valid
  var msg = validateQuantity(buyQuantity)
  // If the message doesn't exist it is ok
  if (!msg) {
    sendRequest(buyQuantity)
  } else {
    setBuyErrorMessage(msg)
  }
}
```

```
const sendRequest = async (quantity) => {
  try {
    // Send information to API
    const response = await axios.post(`${process.env.REACT_APP_API_URL}/transactions`, {
      type: type,
      quantity: parseInt(quantity),
      wallet_id: selectedWallet
    },
    {
      headers: {
        'Authorization': `Bearer ${localStorage.getItem('JWTToken')}`
      }
    })
    // Created status
    if (response.status === 201) {
      // If valid, we update the wallets with the response params.
      transactionMessage("Your transfer has been made successfully")
      reloadCoins()
    }
  } catch (err) {
    // Http Error
    let e = { ...err }
    console.log(e)
    if (e.response.data.transaction.message) {
      transactionMessage(e.response.data.transaction.message)
    } else {
      transactionMessage("In this moments we can't make your transaction")
    }
  }
}
```

## **2-. Dominio de lo aprendido en el curso**

## 2.1 Principios fundamentales

- Seiri: Convención de nombres.
- Seiso: Sin código que no aporte.
- Seiton: Orden dentro del código.
- Shutsuke: Disciplina al respetar lo anterior
- Seiketsu: Consenso previo (organización de carpetas)

# 2.1 Principios Fundamentales

## src

- > actions
- > components
  - > Dashboard
  - > Layout
  - > Login
  - > SignUp
  - > Transactions
- > reducers
- > store
- > utils
- > views
  - > Dashboard
  - > LoginPage
  - > Transaction

## app

- > channels
- > controllers
- > jobs
- > mailers
- > models
- > serializers
- > views

```
const getToken = async (email, password) => {  
  // Here we get the token once the account is created. Equivalent to Login  
  try {  
    const tokenResponse = await axios.post(`${process.env.REACT_APP_API_URL}/user_token`, {  
      auth: {  
        email,  
        password  
      }  
    })  
    if (tokenResponse.status === 201) {  
      // Sets the token for all the following requests  
      setToken(tokenResponse.data.jwt)  
    }  
  } catch (err) {  
    setErrorMessage("The username or password doesn't match")  
  }  
}  
  
const setToken = (token) => {  
  // Sets the authorization header with the token for every request.  
  const JWTToken = `Bearer ${token}`  
  localStorage.setItem('JWTToken', JWTToken)  
  axios.defaults.headers.common['Authorization'] = JWTToken;  
  // Now we get the user using the token.  
  setUserInfo(token)  
}
```



## 2.2 SOLID

- Ejemplo de Single responsibility principle:

```
const Page = forwardRef(({
  title,
  children,
}, ref) => {

  return (
    <div
      ref={ref}
      className="page-container"
    >
      <Helmet>
        <title>{title}</title>
      </Helmet>
      <NavBar />
      <div className="page-sub-container">
        {children}
      </div>
    </div>
  );
});

export default Page;
```

## 2.3 Code Smells

- Bloaters: Large Class
- Bloaters: Long Parameter List

```
@coin = Coin.new(coin_params)
```

```
<div className='login-container'>  
  <div className='login-sub-container'>  
    <SignUp />  
    <Login />  
  </div>  
</div>
```

✓ Login

# Login.css

JS Login.js

✓ SignUp

# SignUp.css

JS SignUp.js

## 2.3 Otros aprendizajes

- Diagramación UML
- Patrones de Diseño

### **3. Principales problemas enfrentados**

## 3.1 Mala documentación de la API

Problema que nos enfrentamos tras la entrega número tres, es que se nos entregó un proyecto el cual tenía varios problemas de documentación. Esto nos llevó a tener que “adivinar” ciertos aspectos.

Ejemplo:

<code>/api/v1/sign_up</code>	<code>POST</code>	<code>email, password,</code> <code>password_confirmation</code>	<code>registrations#create</code>
------------------------------	-------------------	---	-----------------------------------

En este ejemplo, los parámetros que están en la tercera columna no eran suficientes para poder hacer el “request”.

## 3.2 Funcionalidades no terminadas

Otro problema que nos enfrentamos especialmente en esta última etapa, fue el hecho que habían funcionalidades no terminadas.

Ejemplo:

POST	Content-type, X-User-Token, X-User-Email	savings_account[investment_type]	savings_accounts#create	Crea una nueva instancia de savings_account. Por el momento no asocia este nueva cuenta al usuario, pero pronto debería hacerlo.		
------	--	----------------------------------	-------------------------	--	--	--

## 3.3 Falta de tiempo

Otro problema que nos enfrentamos fue a la falta de tiempo para aplicar otros conocimientos del curso como refactoring y tests.

# Presentación Final



IIC-2113

Integrantes:

Sebastián Bitar

Patricio de Solminihac

Vicente Lira

Sebastián Morales

Felipe Trejo