

DIGITAL HUB - BACKEND ASSIGNMENT - CP

- **DATOS CORPORATIVOS (MANUAL DE MANEJO)**

En el siguiente documento , se describirán los pasos para poder lograr la correcta ejecución del proyectos, se mencionaran las instalaciones, al igual que los métodos de ejecución

- **DESCRIPCIÓN:**

Este sistema tiene la finalidad, las transacciones seguras de datos, media un api, realizada en nodejs

- **INSTALACIONES:**

NODEJS: <https://nodejs.org/es/> (Window)

XAMPP: <https://www.apachefriends.org/es/index.html>

VisualStudio Code: <https://code.visualstudio.com/>

- **INSTALACIONES INTERNAS SISTEMA:**

Instalación de express: `npm install --save` Manejador de cambios median

nodemon: `npm install --save-dev nodemon` webpack para poder usar js dom

evento etc y enlazar archivo

`npm install -D babel-loader @babel/core @babel/preset-env webpack`

`npm install --save-dev -D babel-loader @babel/core @babel/preset-env webpack`

webpack-cli Ejecutar varios comando al mismo tiempo `npm start : npm install --save concurrently`

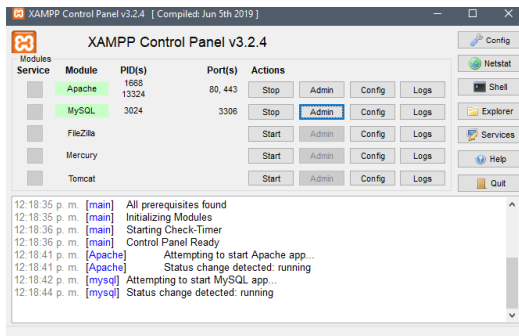
- **EJECUTAR EL PROGRAMA**

`npm run start`

abrir en el navegador deseado en el localhost:3000 (puerto 3000). Con xampp ejecutandose

- **EJECUCIÓN DE LA BASE DE DATOS:**

Si se usa xampp, tendrá que activar el servidor ,para que se puedan hacer las conexiones establecidas en el programa, tan solo al entrar al panel de xampp



- IMPORTAR BASE DE DATOS

Crear una una base nueva de datos con el nombre banco con el parámetro utf8_general_ci como se ve a continuación

Bases de datos

Crear base de datos

Ya creada vamos a importar la base de datos que encontramos en el enlace a git hub, para esto entramos a la base que creamos y damos click en e el botón superior importar. Y seleccionamos la base que como mencione esta en el enlace github llamada banco.sql

Importando en la base de datos "uptasks"

Archivo a importar:

El archivo puede ser comprimido (gzip, bzip2, zip) o descomprimido.
A compressed file's name must end in `.[format].[compression]`. Example: `.sql.zip`

Buscar en su ordenador: uptasks.sql (Máximo: 40MB)

También puede arrastrar un archivo en cualquier página.

Conjunto de caracteres del archivo:

Ya importada solo damos en continuar para importar correctamente la base de datos y listos, podremos usar el sistema correctamente

PARAMETROS ENVIADOS DESDE POSTMAN

En la url solo cambiar el parámetro final según el caso y se enviarán los parámetros con el nombre que a continuación se muestra al igual que el método de envío

Obtener Saldo=saldo METODO: GET

<input checked="" type="checkbox"/>	fromAccount	84927494
-------------------------------------	-------------	----------

Obtener historial = historial METODO: GET

<input checked="" type="checkbox"/>	fromAccount	84927494
-------------------------------------	-------------	----------

Crear Cuenta= crear METODO: GET

<input checked="" type="checkbox"/>	usuario	Martin
<input checked="" type="checkbox"/>	cuenta	2949475
<input checked="" type="checkbox"/>	saldo	\$8000

Enviar Transacciones= enviar METODO: GET

<input checked="" type="checkbox"/>	fromAccount	84927494
<input checked="" type="checkbox"/>	toAccount	1749234
<input checked="" type="checkbox"/>	amount	1000

EJEMPLO REALIZADO DE MUESTRA

GET ▼ http://localhost:3000/enviar Send ▼

Params Authorization Headers (10) **Body ●** Pre-request Script Tests Settings

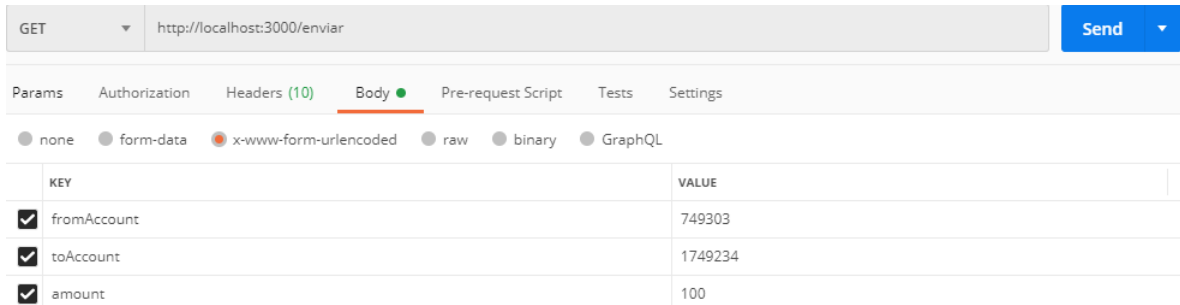
☐ none ☐ form-data ☒ x-www-form-urlencoded ☐ raw ☐ binary ☐ GraphQL

	KEY	VALUE	•
<input checked="" type="checkbox"/>	fromAccount	84927494	
<input checked="" type="checkbox"/>	toAccount	1749234	
<input checked="" type="checkbox"/>	amount	1000	
<input type="checkbox"/>	usuario	Martin	
<input type="checkbox"/>	cuenta	2949475	
<input type="checkbox"/>	saldo	\$8000	

Como se puede ver se realiza una petición get y posteriormente insertamos la url, y presionamos send, para esto tenemos que irnos al apartado de body y así mismo x-www-form, y colocar correctamente los nombres a los campos, esto servirá de muestra como si tuviéramos inputs de página web, y así con cada petición

ENVIO DE SALDO:

Usando POSTMAN hacemos el redireccionamiento hacia la ruta “enviar”



Postman interface showing a GET request to `http://localhost:3000/enviar`. The request is configured with the following form data:

KEY	VALUE
<input checked="" type="checkbox"/> fromAccount	749303
<input checked="" type="checkbox"/> toAccount	1749234
<input checked="" type="checkbox"/> amount	100

A continuación se aran las validaciones que muestre que tanto la cuenta propia como la del destinatario existen:

```
1 {  
2   "message": "fromAccount or toAccount no valid "  
3 }
```

Si la cuenta y el saldo son disponible enviara un mensaje de confirmación

```
1 {  
2   "message": "OK",  
3   "fromAccount": "749303",  
4   "toAccount": "1749234",  
5   "amount": "100"  
6 }
```

Historial de transacciones:

Igualmente marcará un mensaje de error si la cuenta a la que queremos acceder no existe

```
1 {  
2   "message": "fromAccount no valid "  
3 }
```

O un mensaje de éxito y los datos del historial

```
{  
  "transactions": [  
    {  
      "id": 1,  
      "fromAccount": "749303",  
      "toAccount": "1749234",  
      "amount": "100"  
    },  
    {  
      "id": 2,  
      "fromAccount": "749303",  
      "toAccount": "1749234",  
      "amount": "100"  
    }  
  ]  
}
```

Crear cuenta con saldo:

Aquí si el usuario no tiene cuenta podrá registrar una con un saldo específico

GET

http://localhost:3000/crear

Params

Authorization

Headers (10)

Body

Pre-request Script

Tests

Settings

none

form-data

x-www-form-urlencoded

raw

binary

GraphQL

KEY	VALUE
<input type="checkbox"/> fromAccount	749303
<input type="checkbox"/> toAccount	1749234
<input type="checkbox"/> amount	100
<input checked="" type="checkbox"/> usuario	Paco
<input checked="" type="checkbox"/> cuenta	7930284
<input checked="" type="checkbox"/> saldo	\$55373

{

"message": "count create"

}

Saldo de la cuenta:

Ultimo paso de la api donde el usuario podrá obtener información inicial del estado de su cuenta

```
1 {  
2   "balance": {  
3     "account": "749303",  
4     "balanc": "$3042",  
5     "owner": "Ferman",  
6     "id": 1  
7   }  
8 }
```

PRUEBAS REALIZADAS:

Primeramente mostraremos los usuarios registrados en la Base de Datos

	id	usuario	cuenta	saldo
ir  Borrar	1	Ferman	749303	\$3042
ir  Borrar	2	Pablo	1749234	\$2938
ir  Borrar	3	Paco	7930284	\$55373

Después de esto pasaremos a crear un usuario con su saldo mediante la api

GET

http://localhost:3000/crear

Params

Authorization

Headers (10)

Body

Pre-request Script

Tests

Settings

☐ none

☐ form-data

☒ x-www-form-urlencoded

☐ raw

☐ binary

☐ GraphQL

KEY	VALUE
<input type="checkbox"/> fromAccount	749303
<input type="checkbox"/> toAccount	1749234
<input type="checkbox"/> amount	100
<input checked="" type="checkbox"/> usuario	Pedro Mendoza
<input checked="" type="checkbox"/> cuenta	84927494
<input checked="" type="checkbox"/> saldo	\$8000
Key	Value

Body

Cookies

Headers (6)

Test Results

Status: 200 OK


Pretty

Raw

Preview

Visualize





JSON



```
1 {  
2   "message": "count create"
```

Ya teniendo el nuevo usuario podrá realizar las actividades que marca la api.

Nuevo usuario registrado

		id	usuario	cuenta	saldo
	 Borrar	1	Ferman	749303	\$3042
	 Borrar	2	Pablo	1749234	\$2938
	 Borrar	3	Paco	7930284	\$55373
	 Borrar	4	Pedro Mendoza	84927494	\$8000

Como podemos apreciar el usuario pedro Mendoza hizo una transacción al usuario pablo exitosamente

GET

http://localhost:3000/enviar

KEY	VALUE
<input checked="" type="checkbox"/> fromAccount	84927494
<input checked="" type="checkbox"/> toAccount	1749234
<input checked="" type="checkbox"/> amount	1000
<input type="checkbox"/> usuario	Pedro Mendoza
<input type="checkbox"/> cuenta	84927494
<input type="checkbox"/> saldo	\$8000
Key	Value

BodyCookiesHeaders (6)Test ResultsStatus: 200 C

PrettyRawPreviewVisualizeJSON

```
1 {
2   "message": "OK",
3   "fromAccount": "84927494",
4   "toAccount": "1749234",
5   "amount": "1000"
6 }
```

Después de cada transacción el usuario podrá ver las transacciones que ha realizado

GET http://localhost:3000/historial

KEY	VALUE
<input checked="" type="checkbox"/> fromAccount	84927494
<input type="checkbox"/> toAccount	1749234
<input type="checkbox"/> amount	1000
<input type="checkbox"/> usuario	Pedro Mendoza
<input type="checkbox"/> cuenta	84927494
<input type="checkbox"/> saldo	\$8000
Key	Value

Body Cookies Headers (6) Test Results Status: 200 OK Time: 26ms

Pretty Raw Preview Visualize JSON

```
1 {
2   "transactions": [
3     {
4       "id": 3,
5       "fromAccount": "84927494",
6       "toAccount": "1749234",
7       "amount": "1000"
8     }
9   ]
10 }
```

Por ultimo se podrá ver el saldo restante o el que tiene en su cuenta

GET http://localhost:3000/saldo

Body Cookies Headers (6) Test Results Status: 200 OK Time: 19ms

Pretty Raw Preview Visualize JSON

```
1 {
2   "balance": {
3     "account": "84927494",
4     "balanc": "$7000",
5     "owner": "Pedro Mendoza",
6     "id": 4
7   }
8 }
```

NOTA:

A este programa se le añadió xampp un gestor de base de datos mediante mysql, lo cual me permitió guardar los datos que fui añadiendo conforme a las pruebas