

DESARROLLO Y PRUEBAS

Área Desarrollo:		No. FSP	
Autor del Desarrollo:	Melquis Jair Peralta Vega		
Proyecto:			
Proveedor:			
Fecha:	28/05/2024		

Hoja de Estado de Documento

Versión	Fecha	Responsable	Descripción
1.0	28/05/2024	Melquis Peralta	

1. Introducción

Este documento tiene como objetivo describir las especificaciones para la para la creaciond una API RESTful para el registro de usuarios.

2. Aspectos Generales de la Aplicación.

Todos los endpoints deben aceptar y retornar solamente JSON, inclusive los mensajes de error.

Todos los mensajes deben seguir el formato:

{"mensaje": "mensaje de error"}

Registro

- Ese endpoint deberá recibir un usuario con los campos "nombre", "correo", "contraseña", más un listado de objetos "teléfono", respetando el siguiente formato:

```
{  
  "name": "Juan Rodriguez",  
  "email": "juan@rodriguez.org",  
  "password": "hunter2",  
  "phones": [  

```

```
{
  "number": "1234567",
  "citycode": "1",
  "contrycode": "57"
}
]
```

- Responder el código de status HTTP adecuado
- En caso de éxito, retorne el usuario y los siguientes campos:
 - id: id del usuario (puede ser lo que se genera por el banco de datos, pero sería más deseable un UUID)
 - created: fecha de creación del usuario
 - modified: fecha de la última actualización de usuario
 - last_login: del último ingreso (en caso de nuevo usuario, va a coincidir con la fecha de creación).
 - token: token de acceso de la API (puede ser UUID o JWT)
 - isactive: Indica si el usuario sigue habilitado dentro del sistema.
- Si caso el correo conste en la base de datos, deberá retornar un error "El correo ya registrado".
- El correo debe seguir una expresión regular para validar que formato sea el correcto (aaaaaaa@dominio.cl).
- La clave debe seguir una expresión regular para validar que formato sea el correcto. (El valor de la expresión regular debe ser configurable).
- El token deberá ser persistido junto con el usuario.

Especificaciones

- Banco de datos en memoria. H2DATABASE
- Proceso de build Maven.
- Persistencia con JPA. Hibernate
- Framework SpringBoot IntelliJ IDEA 2022
- Java 17
- Entrega en un repositorio publico <https://github.com/mejapeve/Poryecto-Creasion-de-Usuarios-Spring-Boot-.git>
- Readme explicando cómo probarlo.
- Diagrama de la solución.

Requisitos opcionales

- JWT como token
- Pruebas unitarias

- Swagger

3. Resultados de las Pruebas:

La API es accesible a través de Swagger <http://localhost:8070/swagger-ui/index.html/#/>

The screenshot displays the Swagger UI for the 'Restrictive Bank System' API, version 1.0. The interface is in Spanish and shows the 'Controller for user service' endpoint for 'POST /users/create'. The request body is required and in 'application/json' format. The response section shows a 200 status for 'HTTP status create' and a 500 status for 'HTTP internal server error'. The schema section lists several data models: PhoneDto, UserDto, ErrorResponseDto, UserResponseDto, and ContactInfoDto.

Swagger UI Header: Swagger, /v3/api-docs, Explore

API Info: Restrictive Bank System V1 OAS 3.0
User registration API
Melquis Jair Peralta Vega - Website
Send email to Melquis Jair Peralta Vega
Apache 2.0
Project for user registration

Servers: http://localhost:8070 - Generated server url

Controller for user service Endpoint for service CRUD operations

POST /users/create Create User

Create a user in the system

Parameters: No parameters

Request body required application/json

Example Value | Schema

```
{
  "name": "string",
  "email": "string",
  "password": "string",
  "phones": [
    {
      "number": "7890831543",
      "citycode": "string",
      "countrycode": "string"
    }
  ]
}
```

Responses

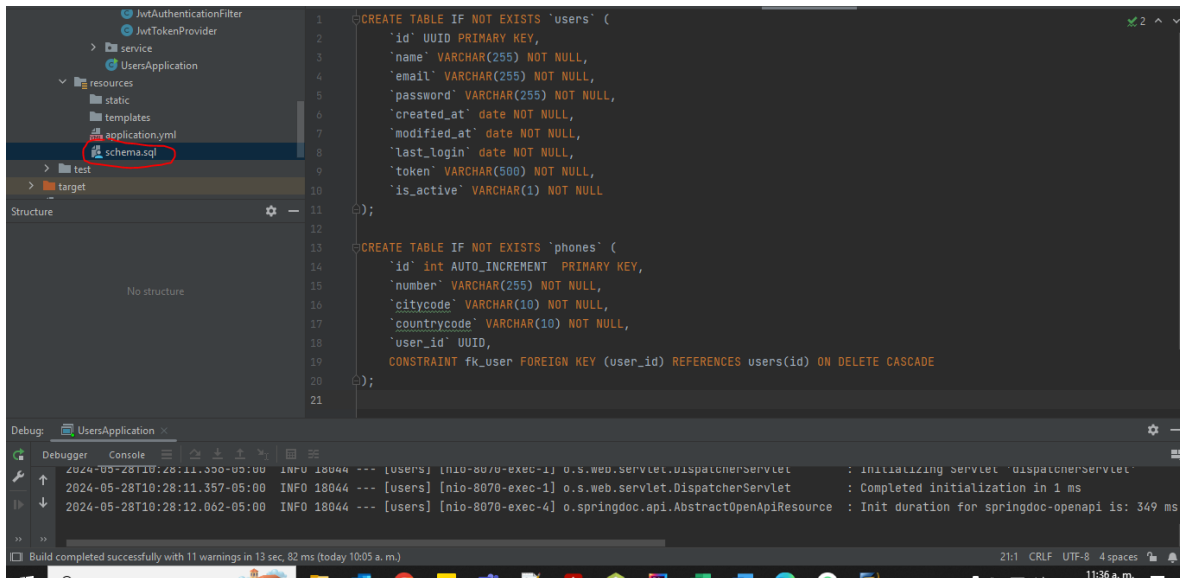
Code	Description	Links
200	HTTP status create Media type: application/json Controls Accept header. Example Value Schema <pre>{ "username": "string", "id": "3f8b164-3737-4562-b3fc-2c963f66af46", "created_at": "2024-05-28T16:01:29.348Z", "modified_at": "2024-05-28T16:01:29.348Z", "last_login": "2024-05-28T16:01:29.348Z", "token": "string", "active": true }</pre>	No links
500	HTTP internal server error Media type: application/json Example Value Schema <pre>{ "apiPath": "string", "errorCode": "100 CONTINUE", "errorMessage": "string", "errorTime": "2024-05-28T16:01:29.352Z" }</pre>	No links

GET /users/contact-info Contact Info User

Schemas

- PhoneDto >
- UserDto >
- ErrorResponseDto >
- UserResponseDto >
- ContactInfoDto >

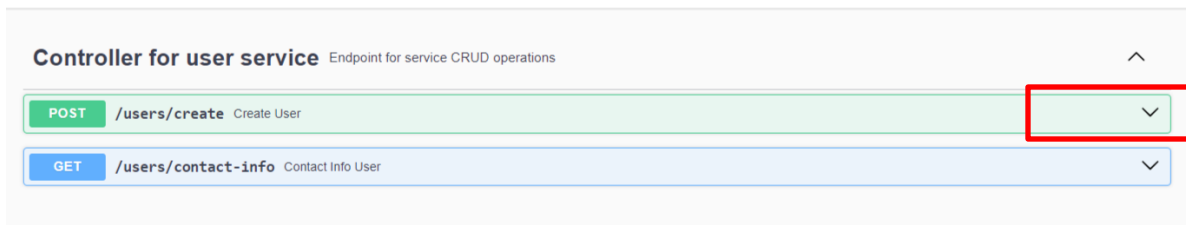
La creacion de las tablas se realiza por persistencia:



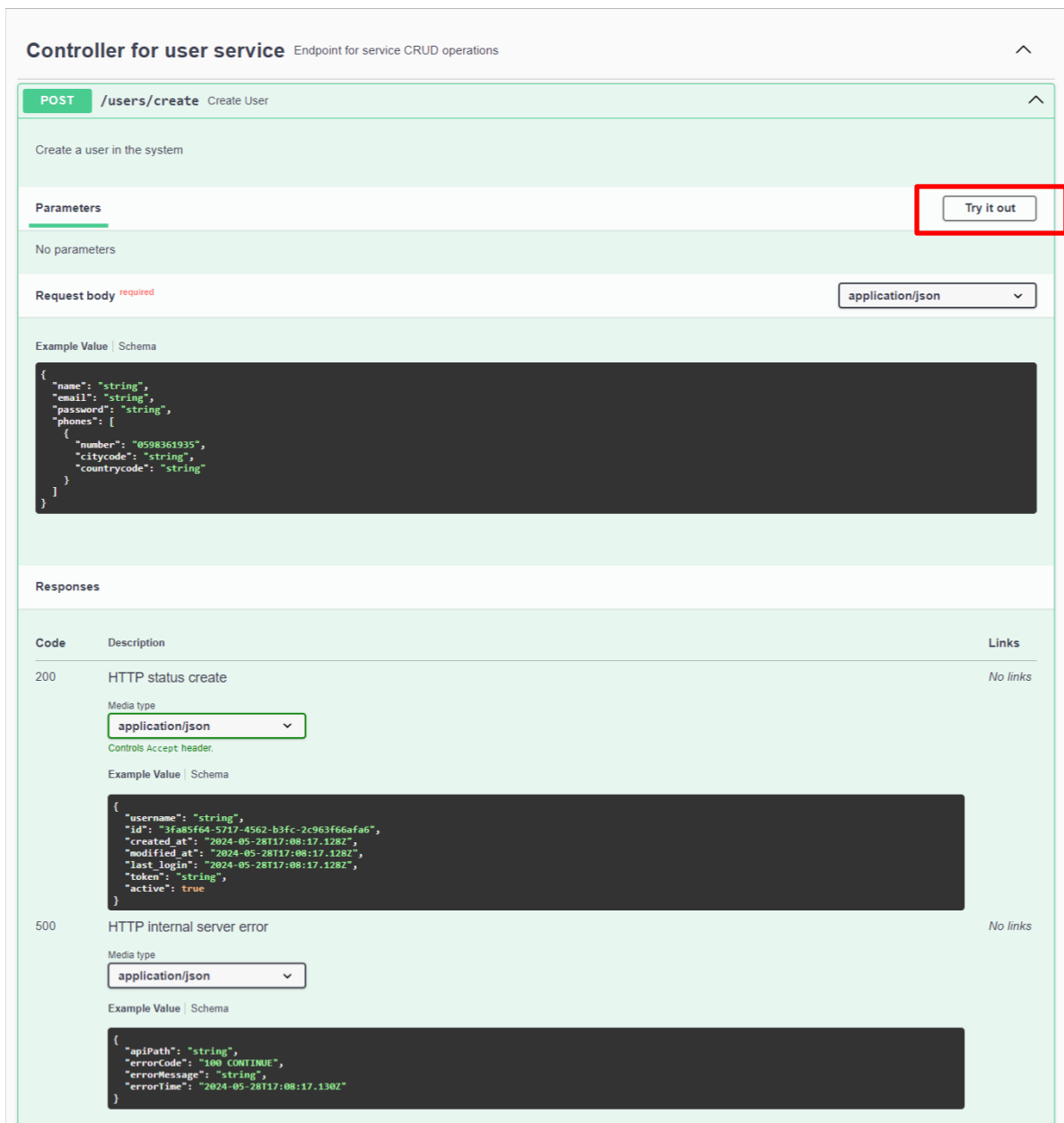
Los scrips para la creacion de dichas tablas son los siguientes:

```
CREATE TABLE IF NOT EXISTS `users` (  
  `id` UUID PRIMARY KEY,  
  `name` VARCHAR(255) NOT NULL,  
  `email` VARCHAR(255) NOT NULL,  
  `password` VARCHAR(255) NOT NULL,  
  `created_at` date NOT NULL,  
  `modified_at` date NOT NULL,  
  `last_login` date NOT NULL,  
  `token` VARCHAR(500) NOT NULL,  
  `is_active` VARCHAR(1) NOT NULL  
);  
  
CREATE TABLE IF NOT EXISTS `phones` (  
  `id` int AUTO_INCREMENT PRIMARY KEY,  
  `number` VARCHAR(255) NOT NULL,  
  `citycode` VARCHAR(10) NOT NULL,  
  `countrycode` VARCHAR(10) NOT NULL,  
  `user_id` UUID,  
  CONSTRAINT fk_user FOREIGN KEY (user_id) REFERENCES users(id) ON  
DELETE CASCADE  
);
```

Ingresamos a Swagger y escogemos el primer servicio



Al ingresar seleccionamos try it out en seccion de parametros



Aquí podemos validar el json de entrada del registro de usuario de acuerdo a la solicitud. Digitamos los datos de entrada y hacemos click en Execute.

Controller for user service Endpoint for service CRUD operations

POST /users/create Create User

Create a user in the system

Parameters Cancel

No parameters

Request body required application/json

```
{
  "name": "string",
  "email": "string",
  "password": "string",
  "phones": [
    {
      "number": "",
      "citycode": "string",
      "countrycode": "string"
    }
  ]
}
```

Execute

Controller for user service Endpoint for service CRUD operations

POST /users/create Create User

Create a user in the system

Parameters Cancel Reset

No parameters

Request body required application/json

```
{
  "name": "Melquis Jair Peralta Vega",
  "email": "mejapeve@gmail.com",
  "password": "Melquis83",
  "phones": [
    {
      "number": "3124698132",
      "citycode": "1",
      "countrycode": "57"
    }
  ]
}
```

Execute

De acuerdo a la solicitud el servicio debe validar el patron del correo y la contraseña y estos patrones deben ser configurables.

```
application.yml
schema.sql
test
target
structure
application.yml D:\bc\users\src\main\resources
YAML document
server
spring
info
build
contact
endpoints
management
63 # regular expression for password validation
64
65 password:
66   regexp: ^(?=[0-9])(?=.*[a-z])(?=.*[A-Z])(?=.*[@#$%^&*+.,])(?=[S+$).{8,}$
67   message: The password must be min 8 length containing atleast 1 uppercase, 1 lowercase, 1 special character and 1 dig
68
69 # regular expression for email validation
70
71 email:
72   regexp: ^\w+([.-_+]?[w+])?@([.-]?[w+])+(\.[w{2,10})+$
73   message: The email must have the following format usuario@dominio.com
74
```

Se realiza una primera prueba con los datos correctos y el resultado es el siguiente:

Server response

Code Details

201
Undocumented

Response body

```
{
  "username": "mejapeve@gmail.com",
  "id": "1b9a3b85-5695-4157-a158-bc82efb6a860",
  "created_at": "2024-05-28T12:22:50.7228191",
  "modified_at": "2024-05-28T12:22:50.7228191",
  "last_login": "2024-05-28T12:22:50.7228191",
  "token": "927a0b36-21e8-4e15-9ce0-2d1bc02e8915",
  "active": true
}
```

Response headers

```
connection: keep-alive
content-type: application/json
date: Tue, 28 May 2024 17:22:50 GMT
keep-alive: timeout=60
transfer-encoding: chunked
```

Database

jdbc:h2:mem:testdb
PHONES
USERS
INFORMATION_SCHEMA
Users
H2 2.2.224 (2023-09-17)

Run Run Selected Auto complete Clear SQL statement:

SELECT * FROM PHONES ;
SELECT * FROM USERS ;

SELECT * FROM PHONES ;

ID	NUMBER	CITYCODE	COUNTRYCODE	USER_ID
1	3124698132	1	57	1b9a3b85-5695-4157-a158-bc82efb6a860

(1 row, 0 ms)

SELECT * FROM USERS ;

ID	NAME	EMAIL	PASSWORD	CREATED_AT	MODIFIED_AT	LAST_LOGIN
1b9a3b85-5695-4157-a158-bc82efb6a860	Melquis Jair Peralta Vega	mejapeve@gmail.com	Melquis*83	2024-05-28 12:22:50.722819	2024-05-28 12:22:50.722819	2024-05-28 12:22:50.722819

(1 row, 0 ms)

	LAST_LOGIN	TOKEN	IS_ACTIVE
19	2024-05-28 12:22:50.722819	927a0b36-21e8-4e15-9ce0-2d1bc02e8915	TRUE

La segunda prueba se realiza con el escenario de un correo sin el formato adecuado

```
{
  "username": "Mejor Juicio Basado en la Vega",
  "email": "mejapeve@gmail",
  "password": "mejqwz-02",
  "phones": [
    {
      "number": "3124698132",
      "citycode": "1",
      "countrycode": "57"
    }
  ]
}
```

El resultado es el siguiente:

Server response

Code	Details
400 <small>Undocumented</small>	<div>Error: response status is 400</div> <div>Response body</div> <pre>{ "apiPath": "uri-/users/create", "errorCode": "BAD_REQUEST", "errorMessage": "The email is incorrect: The email must have the following format usuario@dominio.com", "errorTime": "2024-05-28T12:36:28.3729952" }</pre> <div>Download</div> <div>Response headers</div> <pre>connection: close content-type: application/json date: Tue, 28 May 2024 17:36:28 GMT transfer-encoding: chunked</pre>

Responses

La tercera prueba se realiza ingresando la contraseña sin el formato adecuado (The password must be min 8 length containing atleast 1 uppercase, 1 lowercase, 1 special character and 1 digit).


```
{
  "name": "Melquis Jair Peralta Vega",
  "email": "mejapeve@gmail.com",
  "password": "melquis*83",
  "phones": [
    {
      "number": "3124698132",
      "citycode": "1",
      "countrycode": "57"
    }
  ]
}
```

El resultado es el siguiente:

Server response	
Code	Details
400 <small>Undocumented</small>	<p>Error: response status is 400</p> <p>Response body</p> <pre>{ "apiPath": "uri=/users/create", "errorCode": "BAD_REQUEST", "errorMessage": "The password is incorrect: The password must be min 8 length containing atleast 1 uppercase, 1 lowercase, 1 special character and a digit", "errorTime": "2024-05-28T12:54:42.4940074" }</pre> <p>Response headers</p> <pre>connection: close content-type: application/json date: Tue, 28 May 2024 17:54:42 GMT transfer-encoding: chunked</pre>

La cuarta prueba se realiza tratando de ingresar el mismo correo por segunda vez

```
{
  "name": "Melquis Jair Peralta Vega",
  "email": "mejapeve@gmail.com",
  "password": "Melquis*83",
  "phones": [
    {
      "number": "3124698132",
      "citycode": "1",
      "countrycode": "57"
    }
  ]
}
```

El resultado es el siguiente:

Server response

Code	Details
400 <i>Undocumented</i>	<p>Error: response status is 400</p> <p>Response body</p> <pre>{ "apiPath": "uri=/users/create", "errorCode": "BAD_REQUEST", "errorMessage": "The email mejapeve@gmail.com it already exists", "errorTime": "2024-05-28T12:57:48.6751203" }</pre> <p>Response headers</p> <pre>connection: close content-type: application/json date: Tue, 28 May 2024 17:57:48 GMT transfer-encoding: chunked</pre>

Para la quinta prueba se envia un request sin uno de los datos("name").

```
{
  "email": "mejapeve@gmail.com",
  "password": "Melquis*83",
  "phones": [
    {
      "number": "3124698132",
      "citycode": "1",
      "countrycode": "57"
    }
  ]
}
```

El resultado es el siguiente:

Server response

Code	Details
400 <i>Undocumented</i>	<p>Error: response status is 400</p> <p>Response body</p> <pre>{ "name": "Username is required", "email": "Invalid email format" }</pre> <p>Response headers</p> <pre>connection: close content-type: application/json date: Tue, 28 May 2024 18:02:46 GMT transfer-encoding: chunked</pre>

Para la sexta prueba ingresamos un usuario con un correo diferente.

```
{
  "name": "Melquis Jair Peralta Vega",
  "email": "mejapeve@hotmail.com",
  "password": "Melquis*83",
  "phones": [
    {
      "number": "3124698132",
      "citycode": "1",
      "countrycode": "57"
    }
  ]
}
```

El resultado es el siguiente:

Server response

Code

Details

201

Undocumented

Response body

```
{
  "username": "mejapeve@hotmail.com",
  "id": "2f569a0e-753c-430b-af67-ae05b41c8e56",
  "created_at": "2024-05-28T13:04:49.9174589",
  "modified_at": "2024-05-28T13:04:49.9174589",
  "last_login": "2024-05-28T13:04:49.9174589",
  "token": "091a7c05-f4ea-4fe8-8630-a281b7018d82",
  "active": true
}
```

Download

Response headers

```
connection: keep-alive
content-type: application/json
date: Tue, 28 May 2024 18:04:49 GMT
keep-alive: timeout=60
transfer-encoding: chunked
```

SELECT * FROM PHONES;

ID	NUMBER	CITYCODE	COUNTRYCODE	USER_ID
1	3124698132	1	57	1b9a3b85-5695-4157-a158-bc82efb6a860
2	3124698132	1	57	2f569a0e-753c-430b-af67-ae05b41c8e56

(2 rows, 0 ms)

SELECT * FROM USERS;

ID	NAME	EMAIL	PASSWORD	CREATED_AT	MODIFIED_AT	LAST_LOGIN
1b9a3b85-5695-4157-a158-bc82efb6a860	Melquis Jair Peralta Vega	mejapeve@gmail.com	Melquis*83	2024-05-28 12:22:50.722819	2024-05-28 12:22:50.722819	2024-05-28 12:22:50.7
2f569a0e-753c-430b-af67-ae05b41c8e56	Melquis Jair Peralta Vega	mejapeve@hotmail.com	Melquis*83	2024-05-28 13:04:49.917459	2024-05-28 13:04:49.917459	2024-05-28 13:04:49.9

(2 rows, 1 ms)