**1. Register User**

1.1 API Description

This API implements the user registration function. If the loginName field already exists, it is considered a duplicate registration. The userId will only be returned if the registration is successful.

1.2 API Design

|  |  |
| --- | --- |
| Endpoint | |
| HttpMethod | POST |
| URL | /api/users |

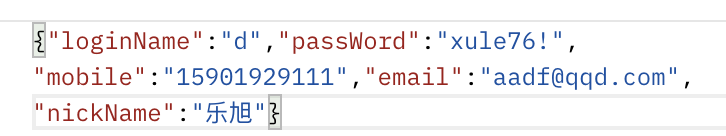
* Request Parameters

This API has no request parameters.

* Request Body:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | | | | |
| Parameter | Nullable | Max Size | Type | Description |
| loginName | N | 250 | string | Unique in the system |
| passWord | N | 255 | string |  |
| mobile | N | 20 | string |  |
| email | N | 100 | string |  |
| nickName | N | 50 | string |  |

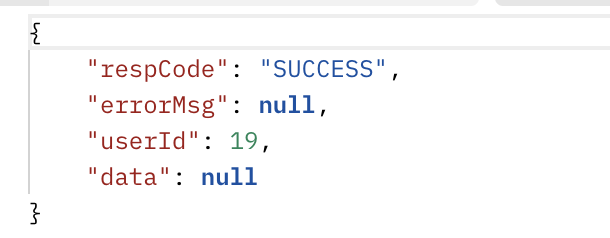
* Request Example

****

* Response Parameters

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | | | | |
| Parameter | Nullable | Max Size | Type | Description |
| respCode | N | 20 | string | Response code indicating the status of the request. Possible values: SUCCESS, BAD\_REQUEST, USER\_ALEADY\_EXIST, UNKNOWN\_ERROR. |
| errorMsg | Y | 20 | string | Contains specific error information when respCode is not SUCCESS. |
| userId | Y | 32 | int | The userId is returned only when the respCode is SUCCESS, indicating a successful registration. |
| data | Y |  | JSON | Reserved field, currently not in use. |

* Respone Example



**2. Read User**

2.1 API Description

This API retrieves user information based on userId. If the userId is found, the corresponding user details will be returned.

2.2 API Design

|  |  |
| --- | --- |
| Endpoint | |
| HttpMethod | GET |
| URL | /api/users/{userId} |

* Request Parameters

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | | | | |
| Parameter | Nullable | Max Size | Type | Description |
| userId | N | 32 | string | The unique identifier of the user. |

* Request Body

This API has no request body.

* Response Parameters

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | | | | |
| Parameter | Nullable | Max Size | Type | Description |
| respCode | N | 20 | string | Response code indicating the status of the request. Possible values: SUCCESS, USER\_NOT\_FOUND, UNKNOWN\_ERROR. |
| errorMsg | Y | 20 | string | Contains specific error information when respCode is not SUCCESS. |
| userId | Y | 32 | int | Returned only if the user is found. The unique identifier of the user. |
| Data | N |  | JSON | A JSON object containing user details, returned only when the user is found. |

Description of data field:

The data field contains a JSON object with the following structure, representing the details of the user:

|  |  |  |  |
| --- | --- | --- | --- |
| Parameter | Nullable | Type | Description |
| loginName | N | string |  |
| passWord | N | string | . |
| mobile | N | string |  |
| email | N | stirng |  |
| nickName | N | stirng |  |

* Respone Example



**3. Edit User**

3.1 API Description

This API updates the user information based on the provided userId. The user data to be updated is passed in the request body. If the update is successful, the response will indicate success and return the updated user information.

3.2 API Design

|  |  |
| --- | --- |
| Endpoint | |
| HttpMethod | Patch |
| URL | /api/users/{userId} |

* Request Parameters:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | | | | |
| Parameter | Nullable | Max Size | Type | Description |
| userId | N | 32 | string | The unique identifier of the user. |

* Request Body:

The request body should be a JSON object containing the fields to be updated. Any fields not included will remain unchanged.

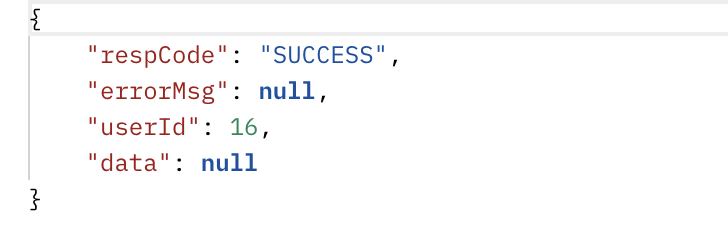
Request Body Example:



* Response Parameters:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | | | | |
| Parameter | Nullable | Max Size | Type | Description |
| respCode | N | 20 | string | Response code indicating the status of the request. Possible values: SUCCESS, USER\_NOT\_FOUND, UNKNOWN\_ERROR. |
| errorMsg | Y | 20 | string | Contains specific error information when respCode is not SUCCESS. |
| userId | Y | 32 | int | Returned only if the user is found. The unique identifier of the user. |
| Data | N |  | JSON | Reserved field, currently not in use. |

* Response Example:



**4. Delete User**

4.1 API Description

This API performs a soft delete on a user by setting the user's status to inactive or deleted, without permanently removing the user from the database. If the operation is successful, the response will indicate success.

2.2 API Design

|  |  |
| --- | --- |
| Endpoint | |
| HttpMethod | Put |
| URL | /api/users/{userId} |

* Request Parameters:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | | | | |
| Parameter | Nullable | Max Size | Type | Description |
| userId | N | 32 | string | The unique identifier of the user. |

* Request Body:

This API has no request body.

* Response Parameters

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | | | | |
| Parameter | Nullable | Max Size | Type | Description |
| respCode | N | 20 | string | Response code indicating the status of the request. Possible values: SUCCESS, USER\_NOT\_FOUND, UNKNOWN\_ERROR. |
| errorMsg | Y | 20 | string | Contains specific error information when respCode is not SUCCESS. |
| userId | Y | 32 | int | The unique identifier of the user that was soft deleted. Returned only if the operation is successful. |
| Data | N |  | JSON | Reserved field, currently not in use. |

* Response Example:

