# USER\_SERVICE

## Mapping - "/users/v1"

## ADD\_USER

### API

**POST**

127.0.0.1:8011/user-service/users/v1

### Request-response

|  |  |  |
| --- | --- | --- |
| body | response | error |
| {  "mainEmail": "smilyk1982@gmail.com",  "notDecryptedPassword":"12345678",  "firstName": "natalya"  } | {  "content": {  "uuidUser": "41cc2c52-9034-4cdd-8a25-6da524d42bcf",  "mainEmail": "smilyk1982@gmail.com",  "firstName": "natalya",  "secondName": **null**,  "altEmail": **null**,  "tz": **null**  },  "code": 201,  "timestamp": "2020-11-24"  } | User exists in DB  {  "timestamp": "2020-11-24T15:36:50.362+00:00",  "message": "user with provided e-mail exists smilyk1982@gmail.com"  } |
| Validation error  {  "timestamp": "2020-11-24T15:37:40.332+00:00",  "message": "Validation failed for argument [0] in public com.users.dto.Response com.users.controllers.UserController.createUser(com.users.dto.AddUserDto): [Field error in object 'addUserDto' on field 'mainEmail': rejected value [null]; codes [NotNull.addUserDto.mainEmail,NotNull.mainEmail,NotNull.java.lang.String,NotNull]; arguments [org.springframework.context.support.DefaultMessageSourceResolvable: codes [addUserDto.mainEmail,mainEmail]; arguments []; default message [mainEmail]]; default message [e-mail can not be null]] "  } |

DB

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| id | alt\_email | confirm\_email | confirm\_email\_token | deleted | first\_name |
| 9 | null | 0 | some\_token | 0 | natalya |

|  |  |  |  |
| --- | --- | --- | --- |
| password | second\_name | tz | uuid\_user |
| Password\_in\_code\_Base64 | null | null | 41cc2c52-9034-4cdd-8a25-6da524d42bcf |

После регистрации пользователь должен подтвердить свой e-mail

## ­CONFIRM\_EMAIL

### API

По сути – этот api не используется на сайте. По нему переходит пользователь, когда нажимает ссылку на подтверждения email в своем письме.

Изображение выглядит как текст

Автоматически созданное описание

**GET**

127.0.0.1:8011/user-service/users/v1/email-verification/token=some\_token\_whisch\_user\_get\_in\_his\_email

После того, как он его получит – изменятся поля:

confirm\_email = 1

confirm\_email\_token = null

теперь пользователь может войти в приложение

### Request-response

|  |  |
| --- | --- |
| response | error |
| {  "content": {  "operationResult": "SUCCESS",  "operationName": "VERIFY\_EMAIL"  },  "code": 200,  "timestamp": "2020-11-24"  } | Wrong link – got error  403- Forbidden  {  "timestamp": "2020-11-24T16:53:23.469+00:00",  "status": 403,  "error": "Forbidden",  "message": "",  "path": "/users/v1/email-vedsdsdsdrification"  } |
| Wrong token or user confirmed his email before – got 200OK +  {  "content": {  "operationResult": "ERROR",  "operationName": "VERIFY\_EMAIL"  },  "code": 200,  "timestamp": "2020-11-24"  } |

## RESET\_PASSWORD

Reset password contains 2 steps. In first step user send request to application with his email, new password and first name. After that user should get email.

In this email user should get link to change his password. Kink contains token for changing password (token is relevant during 1 hour)

In second step user should click on link in his email. It will send request to application.

Application should check token and if everything OK – change password for user.

### Step1

### API

**POST**

127.0.0.1:8011/user-service/users/v1/password-reset-request

### Request-response

|  |  |  |
| --- | --- | --- |
| request | response | error |
| {  "email": "smilyk1982@gmail.com",  "notDecryptedPassword":"12345678",  "firstName": "natalya"  } | {  "operationResult": "SUCCESS",  "operationName": "PASSWORD\_RESET"  } | Wrong credentials  {  "operationResult": "ERROR",  "operationName": "PASSWORD\_RESET"  } |

When user send request for changing his password in DB in table password\_reset\_tokens (DB – a\_b\_user\_service), appears record, which contains token, userId and id of records.

Изображение выглядит как текст

Автоматически созданное описание

### Step1

### API

**POST**

127.0.0.1:8011/user-service/users/v1/password-reset

### Request-response

|  |  |  |
| --- | --- | --- |
| request | response | error |
| {  "token":"token\_from\_db",  "password":"12345678"  } | {  "operationResult": "SUCCESS",  "operationName": "PASSWORD\_RESET"  } | If user didn`t done step 1  {  "operationResult": "ERROR",  "operationName": "PASSWORD\_RESET"  } |

When application get request with password\_token, in table password\_reset\_tokens (DB – a\_b\_user\_service) deleting record for this user and in table users (same DB) changing password for this user.

## LOGIN\_USER

API we need for getting token for a specific user

### API

**POST**

127.0.0.1:8011/user-service/users/login

### Request-response

|  |  |
| --- | --- |
| request | error |
| {  "mainEmail": "smilyk1982@gmail.com",  "password":"12345678"  } | Wrong credentials  Getting 403 - Forbidden |

In response should get token.

Изображение выглядит как стол

Автоматически созданное описание

After this user can use this token for authentication

## USING\_TOKEN\_FOR\_AUTHENTIFICATION

When user want use methods with authentication – he should write his token in headers

Изображение выглядит как стол

Автоматически созданное описание

## NEED\_AUTHENTICATION

"/users/v1"

## GET\_ALL\_USERS

### API

**GET**

localhost:8011/user-service/users/v1

### Request-response

|  |  |
| --- | --- |
| response | error |
| {  "content": [  {  "uuidUser": "41cc2c52-9034-4cdd-8a25-6da524d42bcf",  "mainEmail": "smilyk1982@gmail.com",  "firstName": "natalya",  "secondName": **null**,  "altEmail": **null**,  "tz": **null**  }  ],  "code": 302,  "timestamp": "2020-11-25"  } | User didn`t provide his token  Error 403 |

## GET\_USER\_BY\_USERUUID

### API

**GET**

127.0.0.1:8011/user-service/users/v1/41cc2c52-9034-4cdd-8a25-6da524d42bcf

### Request-response

|  |  |
| --- | --- |
| response | error |
| {  "content": {  "uuidUser": "41cc2c52-9034-4cdd-8a25-6da524d42bcf",  "mainEmail": "smilyk1982@gmail.com",  "firstName": "natalya",  "secondName": **null**,  "altEmail": **null**,  "tz": **null**  },  "code": 302,  "timestamp": "2020-11-25"  } | User didn`t provide his token  Error 403 |
| Wrong UUID  {  "timestamp": "2020-11-25T08:18:57.569+00:00",  "message": "USER\_WITH\_UUID41cc2c52-9034-4cdd-8a25-6da524d42bcftNOT\_FOUND"  } |

## UPDATE\_USER

### API

**PUT**

127.0.0.1:8011/user-service/users/v1/888c2721-1f6c-4a7a-8e49-61bf3a75dcfc

### Request-response

|  |  |  |
| --- | --- | --- |
| request | response | error |
| {  "firstName": "Vasya",  "secondName":"VASILEV",  "altEmail": "alt@al.v",  "tz": "333333333"  } | {  "content": {  "uuidUser": "888c2721-1f6c-4a7a-8e49-61bf3a75dcfc",  "firstName": "Vasya",  "secondName": "VASILEV",  "altEmail": "alt@al.v",  "tz": "333333333"  },  "code": 200,  "timestamp": "2020-11-25"  } | User didn`t provide his token  Error 403 |
| Wrong UUID  {  "timestamp": "2020-11-25T08:18:57.569+00:00",  "message": "USER\_WITH\_UUID41cc2c52-9034-4cdd-8a25-6da524d42bcftNOT\_FOUND"  } |

## DELETE\_USER

### API

**DELETE**

127.0.0.1:8011/user-service/users/v1/888c2721-1f6c-4a7a-8e49-61bf3a75dcfc

### Request-response

|  |  |
| --- | --- |
| response | error |
| {  "content": {  "operationResult": "SUCCESS",  "operationName": "DELETE"  },  "code": 200,  "timestamp": "2020-11-25"  } | User didn`t provide his token  Error 403 |
| Wrong UUID  {  "timestamp": "2020-11-25T08:18:57.569+00:00",  "message": "USER\_WITH\_UUID41cc2c52-9034-4cdd-8a25-6da524d42bcftNOT\_FOUND"  } |

## Mapping - "/resp\_pers/v1"

## ADD\_RESPONSE\_PERSON

### API

**POST**

127.0.0.1:8011/user-service/resp\_pers/v1

### Request-response

|  |  |  |
| --- | --- | --- |
| body | response | error |
| {  "emailRespPerson": "liza@ukr.net",  "firstName": "liza",  "secondName": "lizova",  "tzRespPers": "222222222"  } | {  "content": {  "uuidRespPerson": "a081d358-0dc6-4ba7-a469-61812eec75c8",  "firstName": "liza",  "secondName": "lizova"  },  "code": 201,  "timestamp": "2020-11-25 } | User exists in DB  {  "timestamp": "2020-11-24T15:36:50.362+00:00",  "message": "user with provided e-mail exists smilyk1982@gmail.com"  } |
| Validation error  {  "timestamp": "2020-11-25T08:32:24.685+00:00",  "message": "Validation failed for argument [0] in public com.users.dto.Response com.users.controllers.RespPersonController.createRespPerson(com.users.dto.AddRespPersonDto): [Field error in object 'addRespPersonDto' on field 'emailRespPerson': rejected value [null]; codes [NotNull.addRespPersonDto.emailRespPerson,NotNull.emailRespPerson,NotNull.java.lang.String,NotNull]; arguments [org.springframework.context.support.DefaultMessageSourceResolvable: codes [addRespPersonDto.emailRespPerson,emailRespPerson]; arguments []; default message [emailRespPerson]]; default message …} |

## GET\_ALL\_RESPONSE\_PERSON

### API

**POST**

127.0.0.1:8011/user-service/resp\_pers/v1/?limit=10

### Request-response

|  |  |
| --- | --- |
| response | error |
| {  "content": [  {  "uuidRespPerson": "a081d358-0dc6-4ba7-a469-61812eec75c8",  "firstName": "liza",  "secondName": "lizova"  }  ],  "code": 302,  "timestamp": "2020-11-25" | User didn`t provide his token  Error 403 |
|  |

## UPDATE\_ RESPONSE\_PERSON \_BY\_USERUUID

### API

**PUT**

127.0.0.1:8011/user-service/resp\_pers/v1/

### Request-response

|  |  |  |
| --- | --- | --- |
| request | response | error |
| {  "uuidRespPerson": "4e0ee485-8bfe-4902-9374-8a9e57750f44",  "emailRespPerson": " va@e.c",  "firstName":"kyzma",  "secondName":"kyzmov",  "tzRespPers":12345676  } | {  "content": {  "uuidRespPerson": "4e0ee485-8bfe-4902-9374-8a9e57750f44",  "emailRespPerson": " va@e.c",  "firstName": "kyzma",  "secondName": "kyzmov",  "tzRespPers": **null**  },  "code": 302,  "timestamp": "2020-11-25"  } | User didn`t provide his token  Error 403 |
| Wrong UUID  {  "timestamp": "2020-11-25T08:18:57.569+00:00",  "message": "USER\_WITH\_UUID41cc2c52-9034-4cdd-8a25-6da524d42bcftNOT\_FOUND"  } |

## DELETE\_ALL\_RESPONSE\_PERSON

### API

**DELETE**

127.0.0.1:8011/user-service/resp\_pers/

### Request-response

|  |  |
| --- | --- |
| response | error |
| {  "content": {  "operationResult": "SUCCESS",  "operationName": "DELETE"  },  "code": 200,  "timestamp": "2020-11-25"  } | User didn`t provide his token  Error 403 |
| Wrong UUID  {  "timestamp": "2020-11-25T08:18:57.569+00:00",  "message": "USER\_WITH\_UUID41cc2c52-9034-4cdd-8a25-6da524d42bcftNOT\_FOUND"  } |

## Mapping – "users/v1/ping "

## PING\_SERVICE

### API

**GET**

127.0.0.1:8011/user-service/users/v1/ping

### Response

Users-Service working 2020-11-25

## Mapping - "/users/v1/admin"-creating administrator

## CREATE\_SUPER\_ADMINISTRATOR

Creating super administrator. Need for getting token in some places

### API

localhost:8090/users/v1/admin