Unicorn Vysoká škola ❯ BIOT 2025S FT Student Team 004 ❯ Production Set

📄 ▶️ BIOT FT Application Model - Team 004

Commands

Method	Command	Description	Application profiles
POST	/user/create	creates a new user	public
PUT	/household/add-user	adds user ID to	household owner
DELETE	/household/remove-	removes user ID	household owner
GET	/user/list-	lists all users	user
PUT	/household/add-	adds device ID to	household owner
DELETE	/household/remove-	removes device ID	household owner
POST	/household/create	creates a new	user
DELETE	/household/delete	deletes the	household owner
PUT	/household/update-	updates the name of	household owner
GET	/user/get	gets a specific	household owner,
PUT	/device/set-	sets alarm_triggered	household owner,
PUT	/device/set-	sets alarm_triggered	
PUT	/device/set-state-	sets the active state	household owner,
PUT	/device/set-state-	sets the state of	

POST/user/create

Basic Info

Name	Creating a new user
Description	This method creates a new user in our
HTTP Method	POST
Url	/user/create
Application profiles	public

This algorithm describes [fill in what end-point does] and all possible errors and warnings that can happen during this process.

Input

```
1  const dtoIn = {
2      firstName: "Jane",
3      lastName: "Smith",
4      email: "jane.smith@example.com",
5      password: "Save123",
6      role: "user", //deafult
7      refreshToken: "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
8  };
```

Input validation

```
1   const dtoInValidationSchema = {
2    firstName: Joi.string()
3        .min(2)
4        .max(50)
5        .required()
6        .pattern(/^[a-zA-Z]+$/)
7        .messages({
8            'string.pattern.base': 'First name should contain only letters',
9            'string.empty': 'First name is required',
10       }),
11   lastName: Joi.string()
12        .min(2)
13        .max(50)
14        .required()
15        .pattern(/^[a-zA-Z]+$/)
16        .messages({
17            'string.pattern.base': 'Last name should contain only letters',
18       }),
19   email: Joi.string()
20        .email()
21        .required()
22        .messages({
23            'string.email': 'Please provide a valid email address',
```

```
24            }),
25        password: Joi.string()
26            .min(6)
27            .max(30)
28            .required()
29            .pattern(/^(?=.*[a-z])(?=.*[A-Z])(?=.*\d).*$/)
30            .messages({
31                'string.pattern.base': 'Password must contain at least one uppercas
32            }),
33        role: Joi.string()
34            .default('user'),
35        refreshToken: Joi.string()
36            .pattern(/^eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9\..*/)
37            .messages({
38                'string.pattern.base': 'Invalid refresh token format',
39            }),
40    };
```

Output

```
1    const dtoOut = {
2        "firstName": "Jane",
3        "lastName": "Smith",
4        "email": "jane.smith@example.com",
5        "password": "Save123",
6        "role": "user",
7        "refreshToken": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9..."
8    };
```

Output validation

```
1    const dtoOutValidationSchema = {
2    };
```

Error prefix: {host}/{command}

1. ≡ Sequence   **Validation of Input Data**

Validation of dtoIn.

1.1. ☐ Step   **Validate dtoIn using userSchema:**

- Ensures required fields are present and valid.
- Ensures optional fields meet type and format requirements.

1.2. ⊙ Error   **Invalid Input Data**

Indicates invalid or missing fields in the request body.

`invalid_${invalidPart}`
`"Invalid data provided."`

2. ≡ Sequence   **Save to Database**

2.1. ☐ Step   **Creates a new user in the database**

- Use the validated input (`dtoIn`) to create a new user.
- Save the user to the database.

2.2. ✓ Success   **User created**

Indicates that the user was successfully created and saved to the database.

`200 OK`

"User created successfully."

3. ⊙ **Error**   **Internal Server Error**

Error used for indicating an unexpected error occurred during processing.

internal_server_error

4. ⇥ **Return**

Returns properly filled dtoOut.

Error list

Type	Code	Message	Parameters
Error	invalid_${invalidPart}	"Invalid data provided."	
Error	internal_server_error		

PUT/household/add-user

Name	Adding a user to a household
Description	adds a user to a household
HTTP Method	PUT
Url	household/add-user
Application profiles	household owner

This algorithm describes [fill in what end-point does] and all possible errors and warnings that can happen during this process.

Input

```
1   const dtoIn = {
2     userId: "67decca2f8843afabaf38976"
3   };
```

Input validation

```
1   const dtoInValidationSchema = {
2     userId: Joi.string()
3       .pattern(/^[0-9a-fA-F]{24}$/)
4       .required()
5       .messages({
6         'string.pattern.base': 'Invalid user ID format',
7         'any.required': 'User ID is required'
8       })
9   };
```

Output

```
1   const dtoOut = {
2     success: true,
3     "message": "User added successfully to Household",
4     "data": {
5       "household": {
6         "id": "67decca2f8843afabaf38976",
7         "name": "Tonikova domactnost",
8         "owner": "507f1f77bcf86cd799439011",
9         "logs": [
10        { id: "507f191e810c19729de860ec", log_type: "507f191e810c19729de860ee", t
11        { id: "507f191e810c19729de860ed", log_type: "507f191e810c19729de860ef", t
12      ],
13        "members": ["507f191e810c19729de860eb", "607f1f77bcf86cd799439012"],
14        "devices": ["607f1f77bcf86cd799439011"]
15      }
16    }
17  };
```

Output validation

```
1   const dtoOutValidationSchema = {
2   };
```

Error prefix: `{host}/{command}`

1. ≡ Sequence **Authentication**

Ensures the user is authenticated by verifying their JWT token.

    1.1. ⊙ Error **Unauthenticated**

    Error used for indicating the user is unauthenticated.

    `unauthenticated`

2. ≡ Sequence **Validation of Input Data**

Validation of dtoIn.

    2.1. | Step | **Validate dtoIn:**

- Ensures required fields are present and valid.
- Ensures optional fields meet type and format requirements.

    2.2. ⊙ Error **Invalid Input Data**

Indicates invalid fields in the request body.

    `invalid_${invalidPart}`

    `'Invalid ${invalidPart} format'`

    2.3. ⊙ Error **Missing Input Data**

Indicates missing fields in the request body.

    `missing_${missingPart}`

    `'${missingPart} is required'`

3. ≡ Sequence **Save to Database**

    3.1. ✓ Success **Household added to a user**

Indicates that the user was successfully added to a household.

    `200 OK`

    `"Household successfully added to a user"`

4. ⊙ Error **Internal Server Error**

Error used for indicating an unexpected error occurred during processing.

```
internal_server_error
```

5. ⇥ **Return**

Returns properly filled dtoOut.

Error list

Type	Code	Message	Parameters
Error	unauthenticated		
Error	invalid_${invalidPart}	'Invalid ${invalidPart} format'	
Error	missing_${missingPart}	'${missingPart} is required'	
Error	internal_server_error		

DELETE/household/remove-user

Basic Info

Name	Deleting user from a household
Description	deletes userfrom a household
HTTP Method	DELETE
Url	household/remove-user
Application profiles	household owner

This algorithm describes [fill in what end-point does] and all possible errors and warnings that can happen during this process.

Input

```
1   const dtoIn = {
2     userId: "5f9d88b3e1c2a4d5e6f7a8b9"
3   };
```

Input validation

```
1   const dtoInValidationSchema = {
2   userId: Joi.string()
3       .pattern(/^[0-9a-fA-F]{24}$/)
4       .required()
5       .messages({
6         'string.pattern.base': 'Neplatný formát ID uživatele',
7         'any.required': 'ID uživatele je povinné'
8       })
9   };
```

Output

```
1   const dtoOut = {
2     "success": true,
3     "message": "User added successfully to Household",
4     "data": {
5       "household": {
6         "id": "67decca2f8843afabaf38976",
7         "name": "Tonikova domactnost",
8         "owner": "507f1f77bcf86cd799439011",
9         "logs": [
10        { id: "507f191e810c19729de860ec", log_type: "507f191e810c19729de860ee", t
11        { id: "507f191e810c19729de860ed", log_type: "507f191e810c19729de860ef", t
12        ],
13        "members": [],
14        "devices": ["607f1f77bcf86cd799439011"]
15      }
16    }
17  };
```

Output validation

```
1   const dtoOutValidationSchema = {
2
3   };
```

Error prefix: `{host}/{command}`

1. ≡ Sequence **Authentication**

Ensures the user is authenticated by verifying their JWT token.

  1.1. ⊙ Error **Unauthenticated**

Error used for indicating the user is unauthenticated.

`unauthenticated`

2. ≡ Sequence **Validation of Input Data**

Validation of dtoIn.

  2.1. [Step] **Validate dtoIn:**

- Ensures required fields are present and valid.
- Ensures optional fields meet type and format requirements.

  2.2. ⊙ Error **Invalid Input Data**

Indicates invalid or missing fields in the request body.

`invalid_${invalidPart}`

`'Neplatný formát ${invalidPart}'`

3. ≡ Sequence **Household removed from a user**

  3.1. ✓ Success

`200 OK`

`"Household deleted successfully"`

4. ⊙ Error **Internal Server Error**

Error used for indicating an unexpected error occurred during processing.

`internal_server_error`

5. ⇥ Return

Returns properly filled dtoOut.

Error list

Type	Code	Message	Parameters
Error	unauthenticated		
Error	invalid_${invalidPart}	'Neplatný formát ${invalidPart}'	
Error	internal_server_error		

GET/user/list-households

Basic Info

Name	Listing all user's households
Description	lists all user's households
HTTP Method	GET
Url	user/list-households
Application profiles	user

This algorithm describes [fill in what end-point does] and all possible errors and warnings that can happen during this process.

Input

```
1   const dtoIn = {
2     userId: "67decca2f8843afabaf38976"
3   };
```

Input validation

```
1   const dtoInValidationSchema = {
2       userId: Joi.string()
3     .pattern(/^[0-9a-fA-F]{24}$/)
4     .required()
5     .messages({
6       'string.pattern.base': 'Neplatný formát ID uživatele',
7       'any.required': 'ID uživatele je povinné'
8     })
9   };
```

Output

```
1   const dtoOut = {
```

```
 2        "success": true,
 3        "message": "Seznam domácností úspěšně načten",
 4        "data": {
 5          "households": [
 6            {
 7            "id": "12e3f4567890abc123def456",
 8            "name": "Tonikova domactnost",
 9            "owner": "507f1f77bcf86cd799439011",
10            "logs": [
11            { id: "507f191e810c19729de860ec", log_type: "507f191e810c19729de860ee", t
12            { id: "507f191e810c19729de860ed", log_type: "507f191e810c19729de860ef", t
13          ],
14            "members": [],
15            "devices": ["607f1f77bcf86cd799439011"]
16          },
17            {
18            "id": "789abc012def3456789abcde",
19            "name": "Tonikova chaticka",
20            "owner": "507f1f77bcf86cd799439011",
21            "logs": [
22            { id: "507f191e810c19729de860ed", log_type: "507f191e810c19729de860ef", t
23          ],
24            "members": [],
25            "devices": []
26          }
27          ]
28        }
29      };
```

Output validation

```
1    const dtoOutValidationSchema = {
2
3    };
```

Error prefix: {host}/{command}

1.   ≡ Sequence   **Authentication**

Ensures the user is authenticated by verifying their JWT token.

   1.1.   ⊙ Error   **Unauthenticated**

Error used for indicating the user is unauthenticated.

   `unauthenticated`

   1.2.   ≡ Sequence   **Validation of Input Data**

   Validation of dtoIn.

      1.2.1.   [Step]   **Validate dtoIn:**

   - Ensures required fields are present and valid.
   - Ensures optional fields meet type and format requirements.

      1.2.2.   ⊙ Error   **Invalid Input Data**

Indicates invalid or missing fields in the request body.

   `invalid_${invalidPart}`

   `'Neplatný formát ID uživatele'`

2.   ≡ Sequence   **Successfully listed all user's households**

    2.1.   ✓ **Success**

    `200 OK`

`"Seznam domácností úspěšně načten"`

3.   ⓘ **Error**   **Internal Server Error**

Error used for indicating an unexpected error occurred during processing.

`internal_server_error`

4.   ⇥ **Return**

Returns properly filled dtoOut.

Error list

Type	Code	Message	Parameters
Error	unauthenticated		
Error	invalid_${invalidPart}	'Neplatný formát ID uživatele'	
Error	internal_server_error		

PUT/household/add-device

Basic Info

Name	Adding a device to a the household
Description	adds a device to the household
HTTP Method	PUT
Url	household/add-device
Application profiles	houshold owner

This algorithm describes [fill in what end-point does] and all possible errors and warnings that can happen during this process.

Input

```
1    const dtoIn = {
2      "deviceId": "665f1f77bcf86cd799439022",
3    };
```

Input validation

```
1    const dtoInValidationSchema = {
2      deviceId: Joi.string()
3        .pattern(/^[0-9a-fA-F]{24}$/)
4        .required()
5        .messages({
```

```
6          'string.pattern.base': 'Neplatný formát ID zařízení',
7          'any.required': 'ID zařízení je povinné'
8        }),
9    };
```

Output

```
1    const dtoOut = {
2      "success": true,
3      "message": "User added successfully to Household",
4      "data": {
5        "household": {
6          "id": "67decca2f8843afabaf38976",
7          "name": "Tonikova domactnost",
8          "owner": "507f1f77bcf86cd799439011",
9          "logs": [
10         { id: "507f191e810c19729de860ec", log_type: "507f191e810c19729de860ee", t
11         { id: "507f191e810c19729de860ed", log_type: "507f191e810c19729de860ef", t
12        ],
13          "members": [],
14          "devices": ["665f1f77bcf86cd799439022"]
15        }
16      }
17    };
```

Output validation

```
1    const dtoOutValidationSchema = {
2    };
```

Error prefix: `{host}/{command}`

1. ☰ Sequence   **Authentication**

Ensures the user is authenticated by verifying their JWT token.

   1.1. ⊙ Error   **Unauthenticated**

   ## Error used for indicating the user is unauthenticated.
   `unauthenticated`

   1.2. ☰ Sequence   **Validation of Input Data**

   Validation of dtoIn.

      1.2.1. Step   **Validate dtoIn:**

      - Ensures required fields are present and valid.
      - Ensures optional fields meet type and format requirements.

      1.2.2. ⊙ Error   **Invalid Input Data**

      ## Indicates invalid or missing fields in the request body.
      `invalid_${invalidPart}`
      `'Neplatný formát ${invalidPart}'`

2. ☰ Sequence

   2.1. ✓ Success   **Device successfully added to a household**

   `200 OK`
   `"Zařízení bylo úspěšně přidáno k householdu"`

3. ⊙ Error   **Internal Server Error**

Error used for indicating an unexpected error occurred during processing.

internal_server_error

4. ⇥ **Return**

Returns properly filled dtoOut.

Error list

Type	Code	Message	Parameters
Error	unauthenticated		
Error	invalid_${invalidPart}	'Neplatný formát ${invalidPart}'	
Error	internal_server_error		

DELETE /household/remove-device

Basic Info

Name	Removing a device from the household
Description	removes a deviceid from a household
HTTP Method	DELETE
Url	household/remove-device
Application profiles	household owner

This algorithm describes [fill in what end-point does] and all possible errors and warnings that can happen during this process.

Input

```
1  const dtoIn = {
2    "deviceId": "665f1f77bcf86cd799439022",
3  };
```

Input validation

```
1  const dtoInValidationSchema = {
2    deviceId: Joi.string()
3      .pattern(/^[0-9a-fA-F]{24}$/)
4      .required()
5      .messages({
6        'string.pattern.base': 'Neplatný formát ID zařízení',
7        'any.required': 'ID zařízení je povinné'
8      })
9  };
```

Output

```
1    const dtoOut = {
2      "success": true,
3      "message": "User added successfully to Household",
4      "data": {
5        "household": {
6          "id": "67decca2f8843afabaf38976",
7          "name": "Tonikova domactnost",
8          "owner": "507f1f77bcf86cd799439011",
9          "logs": [
10         { id: "507f191e810c19729de860ec", log_type: "507f191e810c19729de860ee", t
11         { id: "507f191e810c19729de860ed", log_type: "507f191e810c19729de860ef", t
12       ],
13         "members": [],
14         "devices": []
15       }
16     }
17   };
```

Output validation

```
1    const dtoOutValidationSchema = {
2    };
```

Error prefix: `{host}/{command}`

1. ≡ Sequence   **Authentication**

Ensures the user is authenticated by verifying their JWT token.

1.1.   ⊙ Error   **Unauthenticated**

Error used for indicating the user is unauthenticated.

`unauthenticated`

1.2.   ≡ Sequence   **Validation of Input Data**

Validation of dtoIn.

1.2.1.   Step   **Validate dtoIn:**

- Ensures required fields are present and valid.
- Ensures optional fields meet type and format requirements.

1.2.2.   ⊙ Error   **Invalid Input Data**

Indicates invalid or missing fields in the request body.

`invalid_${invalidPart}`
`"Invalid data provided."`

2. ≡ Sequence   **Device successfully removed from a household**

2.1.   ✓ Success

`200 OK`
`"Zařízení bylo úspěšně odebráno z householdu"`

3. ⊙ Error   **Internal Server Error**

Error used for indicating an unexpected error occurred during processing.

`internal_server_error`

4. ⇥ Return

Returns properly filled dtoOut.

Error list

Type	Code	Message	Parameters
Error	unauthenticated		
Error	invalid_${invalidPart}	"Invalid data provided."	
Error	internal_server_error		

POST/household/create

Basic Info

Name	Creating household
Description	creates a new household
HTTP Method	POST
Url	household/create
Application profiles	user

This algorithm describes [fill in what end-point does] and all possible errors and warnings that can happen during this process.

Input

```
1   const dtoIn = {
2      "name": "Hrobarovi",
3   };
```

Input validation

```
1    const dtoInSchema = Joi.object({
2       name: Joi.string()
3       .min(1)
4       .max(100)
5       .required()
6       .messages({
7         'string.empty': 'Název domácnosti je povinný',
8         'string.min': 'Název musí mít alespoň 1 znak',
9         'string.max': 'Název může mít maximálně 100 znaků'
10      })
11   });
```

Output

```
1   const dtoOut = {
2     "success": true,
3     "message": "Domácnost 'Hrobarovi' byla úspěšně vytvořena",
4     "data": {
5       "household": {
6         "id": "5f8d0d55b54764421b7156c3",
7         "name": "Hrobarovi",
8         "owner": "67decca2f8843afabaf38976",
9         "logs": [],
10        "members": [],
11        "devices: [],
12        "createdAt": "2023-10-15T10:00:00.000Z"
13      }
14    }
15  };
```

Output validation

```
1   const dtoOutValidationSchema = {
2   };
```

Error prefix: `{host}/{command}`

1. ≡ Sequence   **Validation of Input Data**

Validation of dtoIn.

   1.1. ⬚ Step   **Validate dtoIn:**

- Ensures required fields are present and valid.
- Ensures optional fields meet type and format requirements.
- Ensures password and name follow length requirements.

   1.2. ⓘ Error   **Invalid Input Data**

Indicates invalid or missing fields in the request body.

`invalid_${invalidPart}`
`"Invalid data provided."`

   1.3. ⓘ Error   **Invalid ownerId**

- Ensures owner Id is valid.

`invalid_ownerId`
`'Neplatné ID vlastníka'`

2. ≡ Sequence   **Save to Database**

   2.1. ⬚ Step

Creates a new household in the database

- Use the validated input (`dtoIn`) to create a new household.
- Save the household to the database.

   2.2. ✓ Success   **New household successfully created**

`200 OK`
`"Domácnost 'Hrobarovi' byla úspěšně vytvořena"`

3. ⓘ Error   **Internal Server Error**

Error used for indicating an unexpected error occurred during processing.

`internal_server_error`

4.   ⇥ **Return**

Returns properly filled dtoOut.

Error list

Type	Code	Message	Parameters
Error	invalid_${invalidPart}	"Invalid data provided."	
Error	invalid_ownerId	'Neplatné ID vlastníka'	
Error	internal_server_error		

DELETE /household/delete

Basic Info

Name	Deleting household
Description	deletes a household
HTTP Method	DELETE
Url	household/delete
Application profiles	household owner

This algorithm describes [fill in what end-point does] and all possible errors and warnings that can happen during this process.

Input

```
1   const dtoIn = {
2       "householdId": "5f8d0d55b54764421b7156c3"
3   };
```

Input validation

```
1   const dtoInValidationSchema = {
2   householdId: Joi.string()
3       .pattern(/^[0-9a-fA-F]{24}$/)
4       .required()
5       .messages({
6           'string.pattern.base': 'Neplatné ID domácnosti',
7           'any.required': 'ID domácnosti je povinné'
8       })
9   };
```

Output

```
1   const dtoOut = {
```

```
 2        "success": true,
 3        "message": "Domácnost 'Hrobarovi' byla úspěšně smazána",
 4        "data": {
 5          "deletedHousehold": {
 6            "id": "5f8d0d55b54764421b7156c3",
 7            "name": "Hrobarovi"
 8          }
 9        }
10      };
```

Output validation

```
1    const dtoOutValidationSchema = {
2    };
```

Error prefix: `{host}/{command}`

1. ≡ Sequence   **Authentication**

Ensures the user is authenticated by verifying their JWT token.

1.1. ⊙ Error   **Unauthenticated**

Error used for indicating the user is unauthenticated.

`unauthenticated`

1.2. ≡ Sequence   **Validation of Input Data**

Validation of dtoIn.

1.2.1. ⎡Step⎤ **Validate dtoIn:**

- Ensures required fields are present and valid.
- Ensures optional fields meet type and format requirements.

1.2.2. ⊙ Error   **Invalid Input Data**

Indicates invalid or missing fields in the request body.

`invalid_${invalidPart}`

`'Neplatné ${invalidPart}'`

2. ≡ Sequence

2.1. ✓ Success   **Household successfully deleted**

`200 OK`

`"Domácnost 'Hrobarovi' byla úspěšně smazána"`

3. ⊙ Error   **Internal Server Error**

Error used for indicating an unexpected error occurred during processing.

`internal_server_error`

4. ⇥ **Return**

Returns properly filled dtoOut.

Error list

Type	Code	Message	Parameters
Error	unauthenticated		
Error	invalid_${invalidPart}	'Neplatné ${invalidPart}'	
Error	internal_server_error		

PUT/household/update-name

Basic Info

Name	Updating household name
Description	updates the name of a household
HTTP Method	PUT
Url	household/update-name
Application profiles	household owner

This algorithm describes [fill in what end-point does] and all possible errors and warnings that can happen during this process.

Input

```
1   const dtoIn = {
2     "newName": "Hrobarovi123"
3   };
```

Input validation

```
1   const dtoInValidationSchema = {
2     newName: Joi.string()
3       .min(1)
4       .max(100)
5       .required()
6       .messages({
7         'string.empty': 'Nový název je povinný',
8         'string.min': 'Název musí mít alespoň 1 znak',
9         'string.max': 'Název může mít maximálně 100 znaků'
10      })
11  };
```

Output

```
1   const dtoOut = {
2     "success": true,
3     "message": "Název domácnosti byl úspěšně změněn na 'Hrobarovi123'",
4     "data": {
5       "updatedHousehold": {
```

```
 6          "id": "5f8d0d55b54764421b7156c3",
 7          "previousName": "Hrobarovi",
 8          "newName": "Hrobarovi123",
 9          "updatedAt": "2023-10-20T14:30:00.000Z"
10        }
11     }
12   };
```

Output validation

```
1    const dtoOutValidationSchema = {
2
3    };
```

Error prefix: `{host}/{command}`

1. ≡ Sequence   **Authentication**

Ensures the user is authenticated by verifying their JWT token.

   1.1.   ⓘ Error   **Unauthenticated**

Error used for indicating the user is unauthenticated.

   `unauthenticated`

   1.2.   ≡ Sequence   **Validation of Input Data**

   Validation of dtoIn.

      1.2.1.   [Step]   **Validate dtoIn:**

- Ensures required fields are present and valid.
- Ensures optional fields meet type and format requirements.
- Ensures required name meet length requirements.

      1.2.2.   ⓘ Error   **Invalid Input Data**

Indicates invalid or missing fields in the request body.

      `invalid_${invalidPart}`

      `'Neplatné ${invalidPart}'`

2. ≡ Sequence   **Household name successfully changed**

   2.1.   ✓ Success

   `200 OK`

   `"Název domácnosti byl úspěšně změněn na ${newName}"`

3. ⓘ Error   **Internal Server Error**

Error used for indicating an unexpected error occurred during processing.

   `internal_server_error`

4. ⇥ Return

Returns properly filled dtoOut.

Error list

Type	Code	Message	Parameters
Error	unauthenticated		
Error	invalid_${invalidPart}	'Neplatné ${invalidPart}'	
Error	internal_server_error		

GET/user/get-specific-household

Basic Info

Name	Retriving specific household
Description	retrives a specific household
HTTP Method	GET
Url	
Application profiles	household owner, household member

This algorithm describes [fill in what end-point does] and all possible errors and warnings that can happen during this process.

Input

```
1  const dtoIn = {
2      householdId:"5f8d0d55b54764421b7156c3",
3  };
```

Input validation

```
1  const dtoInValidationSchema = {
2    householdId: Joi.string()
3     .pattern(/^[0-9a-fA-F]{24}$/)
4     .required()
5     .messages({
6       'string.pattern.base': 'Neplatné ID domácnosti',
7       'any.required': 'ID domácnosti je povinné'
8     })
9  };
```

Output

```
1  const dtoOut = {
2    "success": true,
3    "message": "Informace o domácnosti úspěšně získána",
4    "data": {
5       "id": "5f8d0d55b54764421b7156c3",
6       "name": "Hrobarovi",
7       "owner": "67decca2f8843afabaf38976",
```

```
 8          "logs": [],
 9          "members": [],
10          "devices: [],
11          "createdAt": "2023-10-15T10:00:00.000Z"
12      }
13    };
```

Output validation

```
1    const dtoOutValidationSchema = {
2
3    };
```

Error prefix: `{host}/{command}`

1. ≡ Sequence **Authentication**

Ensures the user is authenticated by verifying their JWT token.

    1.1. ⓘ Error **Unauthenticated**

Error used for indicating the user is unauthenticated.

    `unauthenticated`

    1.2. ≡ Sequence **Validation of Input Data**

    Validation of dtoIn.

       1.2.1. ⬚Step **Validate dtoIn:**

- Ensures required fields are present and valid.
- Ensures optional fields meet type and format requirements.

       1.2.2. ⓘ Error **Invalid Input Data**

Indicates invalid or missing fields in the request body.

       `invalid_${invalidPart}`

       `"Invalid data provided."`

2. ≡ Sequence

    2.1. ✓ Success **Specific household info successfully showed**

    `200 OK`

    `"Informace o domácnosti úspěšně získána"`

3. ⓘ Error **Internal Server Error**

Error used for indicating an unexpected error occurred during processing.

`internal_server_error`

4. ⇥ Return

Returns properly filled dtoOut.

Error list

Type	Code	Message	Parameters
Error	unauthenticated		
Error	invalid_${invalidPart}	"Invalid data provided."	
Error	internal_server_error		

PUT/device/set-alarm_triggered

Basic Info

Name	Setting state to 0
Description	sets alarm_triggered state to 0 (stops device
HTTP Method	PUT
Url	
Application profiles	household owner, household member

This algorithm describes [fill in what end-point does] and all possible errors and warnings that can happen during this process.

Input

```
1  const dtoIn = {
2    "deviceId": "665f1f77bcf86cd799439022",
3    "householdId": "5f8d0d55b54764421b7156c3",
4    "alarm_triggered": 1
5  };
```

Input validation

```
1   const dtoInValidationSchema = {
2     deviceId: Joi.string()
3       .pattern(/^[0-9a-fA-F]{24}$/)
4       .required()
5       .messages({
6         'string.pattern.base': 'Neplatné ID zařízení',
7         'any.required': 'ID zařízení je povinné'
8       }),
9     householdId: Joi.string()
10      .pattern(/^[0-9a-fA-F]{24}$/)
11      .required()
12      .messages({
13        'string.pattern.base': 'Neplatné ID domácnosti',
14        'any.required': 'ID domácnosti je povinné'
15      }),
16    alarm_triggered: 1,
17  };
```

Output

```
 1    const dtoOut = {
 2      "success": true,
 3      "message": "Stav alarmu zařízení 'alarm_obyvak_hrobarovi' byl aktualizován.",
 4      "data": {
 5        "device": {
 6          "id": "665f1f77bcf86cd799439022",
 7          "name": "alarm_obyvak_hrobarovi",
 8          "device_type": "alarm",
 9          "alarm_triggered": 1,
10          "updatedAt": "2023-10-20T15:30:00.000Z"
11        }
12      }
13    };
```

Output validation

```
 1    const dtoOutValidationSchema = {
 2    };
```

Error prefix: `{host}/{command}`

1. ≡ Sequence **Authentication**

Ensures the user is authenticated by verifying their JWT token.

   1.1. ⊙ Error **Unauthenticated**

Error used for indicating the user is unauthenticated.

   `unauthenticated`

   1.2. ≡ Sequence **Validation of Input Data**

   Validation of dtoIn.

      1.2.1. Step **Validate dtoIn:**

- Ensures required fields are present and valid.
- Ensures optional fields meet type and format requirements.

      1.2.2. ⊙ Error **Invalid Input Data**

Indicates invalid or missing fields in the request body.

      `invalid_${invalidPart}`

      `'Neplatné ${invalidPart}'`

2. ≡ Sequence **Alarm state successfully activated**

   2.1. ✓ Success

   `200 OK`

   `"Stav alarmu zařízení 'alarm_obyvak_hrobarovi' byl aktualizován."`

3. ⊙ Error **Internal Server Error**

Error used for indicating an unexpected error occurred during processing.

`internal_server_error`

4. ⇥ **Return**

Returns properly filled dtoOut.

Error list

Type	Code	Message	Parameters
Error	unauthenticated		
Error	invalid_${invalidPart}	'Neplatné ${invalidPart}'	
Error	internal_server_error		

PUT/device/set-active_state

Basic Info

Name	Active state of the alarm
Description	toggles active state of the alarm (to true)
HTTP Method	PUT
Url	
Application profiles	household owner, household member

This algorithm describes [fill in what end-point does] and all possible errors and warnings that can happen during this process.

Input

```
1  const dtoIn = {
2    "deviceId": "665f1f77bcf86cd799439022",
3    "householdId": "5f8d0d55b54764421b7156c3",
4    "active": true
5  };
```

Input validation

```
1   const dtoInValidationSchema = {
2    deviceId: Joi.string()
3      .pattern(/^[0-9a-fA-F]{24}$/)
4      .required()
5      .messages({
6        'string.pattern.base': 'Neplatné ID zařízení',
7        'any.required': 'ID zařízení je povinné'
8      }),
9    householdId: Joi.string()
10     .pattern(/^[0-9a-fA-F]{24}$/)
11     .required()
12     .messages({
13       'string.pattern.base': 'Neplatné ID domácnosti',
14       'any.required': 'ID domácnosti je povinné'
15     }),
16   active: true
```

```
17  };
```

Output

```
1   const dtoOut = {
2     "success": true,
3     "message": "Stav alarmu zařízení 'alarm_obyvak_hrobarovi' byl aktualizován.",
4     "data": {
5       "device": {
6         "id": "665f1f77bcf86cd799439022",
7         "name": "alarm_obyvak_hrobarovi",
8         "device_type": "alarm",
9         "active": true,
10        "updatedAt": "2023-10-20T15:30:00.000Z"
11      }
12    }
13  };
```

Output validation

```
1   const dtoOutValidationSchema = {
2   };
```

Error prefix: `{host}/{command}`

1. ≡ Sequence   **Authentication**

Ensures the user is authenticated by verifying their JWT token.

   1.1. ⊙ Error   **Unauthenticated**

   Error used for indicating the user is unauthenticated.
   `unauthenticated`

   1.2. ≡ Sequence   **Validation of Input Data**
   Validation of dtoIn.

      1.2.1. Step   **Validate dtoIn:**

- Ensures required fields are present and valid.
- Ensures optional fields meet type and format requirements.

      1.2.2. ⊙ Error   **Invalid Input Data**

   Indicates invalid or missing fields in the request body.
   `invalid_${invalidPart}`
   `'Neplatné ${invalidPart}'`

2. ≡ Sequence   **Alarm state successfully activated**

   2.1. ✓ Success

   `200 OK`
   `"Stav alarmu zařízení 'alarm_obyvak_hrobarovi' byl aktualizován."`

3. ⊙ Error   **Internal Server Error**

Error used for indicating an unexpected error occurred during processing.
`internal_server_error`

4. ⇥ Return

Returns properly filled dtoOut.

Error list

Type	Code	Message	Parameters
Error	unauthenticated		
Error	invalid_${invalidPart}	'Neplatné ${invalidPart}'	
Error	internal_server_error		