iVIS Documentation

Release 0.1.0

Dmitry Sikorsky

Apr 24, 2017

Contents

.. **iVIS documentation master file, created by** sphinx-quickstart on Thu Aug 27 14:17:59 2015. You can adapt this file completely to your liking, but it should at least contain the root *toctree* directive.

CHAPTER 1

Contents

Overview of iVIS

iVIS is innovative business system for schools. Here you can get more information.

It is a VINNOVA-funded project to create basic conditions for the digitization of the Swedish school, by building a completely open school administrative system.

The project intends to create a school system that makes it possible for the school to develop faster, in a more innovative way and to move the control over the information from suppliers to the school's stakeholders.

Quick Start

Walkthrough: Install iVIS Server

Prerequisites

To install iVIS server you need:

- Git;
- iVIS project;
- Java;
- MySQL database;
- Maven;
- Tomcat server.

For detailed instructions of installing this components visit this page.

Project structure

iVIS maven project from GitHub consists of 4 modules:

1. ivis-core - entities that you can use in API;
2. ivis-services - service interfaces for entities; they are define provided methods;
3. ivis-sdk - sdk for easier work with the API;
4. ivis-server - iVIS server, implementation of the application logic.

Git configuration

Clone project from GitHub repository:

```
cd /../directory #Directory where project folder with content must be placed

git clone https://github.com/imCodePartnerAB/iVIS.git
```

After cloning will created subdirectory iVIS with content.

Database configuration

In the Terminal (Ctrl+ALt+T) input following commands to create the database:

```
mysql -u {username} -p{password} #{username} - database username, {password} -␣
↪database password

CREATE DATABASE db_ivis_test; #or custom {databaseName}
```

Edit file **local.server.properties** from *ivis-server* subfolder in the next way:

- User[1];
- Password[2];
- JdbcUrl[3];
- Hibernate.hbm2ddl[4];
- Hibernate.dialect[5];
- Server.name[6];

Put file import.sql from Github project repository (v1.0.0-alpha1 release) into *ivis-server/src/main/resources*.

Run application

If you have process on port 8080 you must kill it by executing following command in the Terminal:

[1] **username** of database (default *root*)
[2] **password** of database
[3] consist of **jdbc:{provider}://{hostname}:{port}/{databaseName}?{encoding}** for mysql {provider} = *mysql*, {hostname} = *localhost*, {port} = 3306, {encoding} = *utf-8*
[4] set *create*
[5] for mysql is *org.hibernate.dialect.MySQL5InnoDBDialect*
[6] **host** of server (default *http://localhost:8080*)

```
fuser -k 8080/tcp
```

Then go to Tomcat folder (*{tomcat-folder}*).

Important: In Tomcat must be configured user with roles: manager-script, manager-gui, admin, manager-jmx. They are configured in this `instance`.

Note: In this manual uses username=admin and password=admin for that authority.

Run Tomcat by executing following command in the Terminal:

```
cd  /home/../apache-tomcat-{version}/bin

chmod +x startup.sh

./startup.sh
```

Go to m2 local repository (user/.m2).

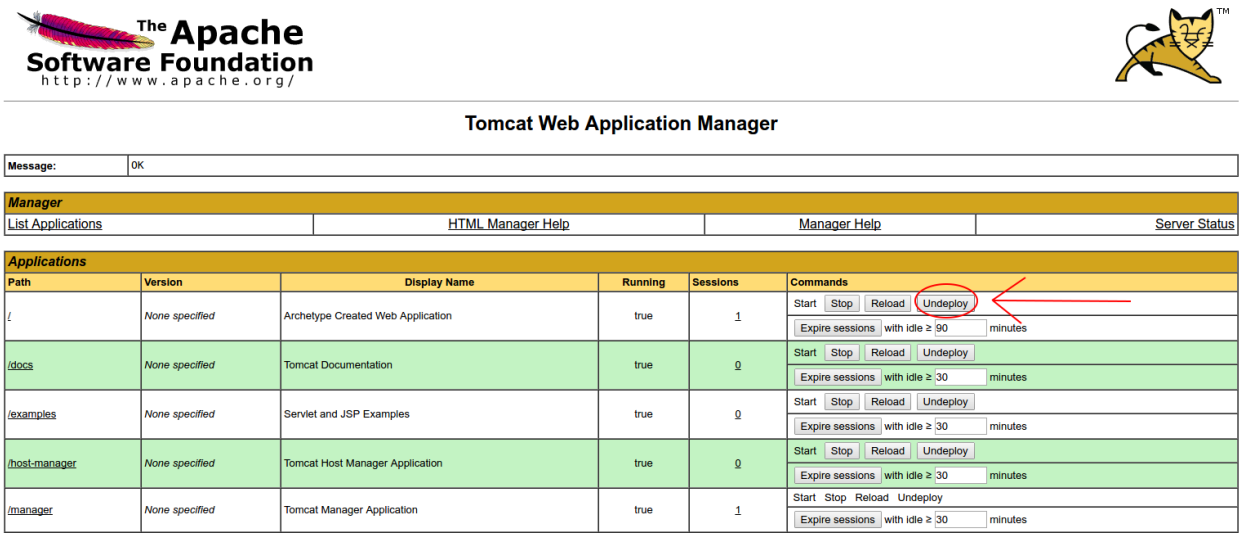
Create/edit file `settings.xml` in next way.

```
        <servers>
                <server>
                        <id>tomcat8</id>
                        <username>admin</username>
                        <password>admin</password>
                </server>
        </servers>
```

In browser input address: http://localhost:8080/manager .

Enter login “admin” and password “admin”.

Click Undeploy where root path (“/”).



In the Terminal (Ctrl+ALt+T) execute following commands:

```
cd /../iVIS #path to iVIS project directory

cd ivis-server

mvn tomcat7:deploy #deploy configured to localhost:8080.
```

Type http://localhost:8080 in browser.

Input Login=admin and Password=pass.

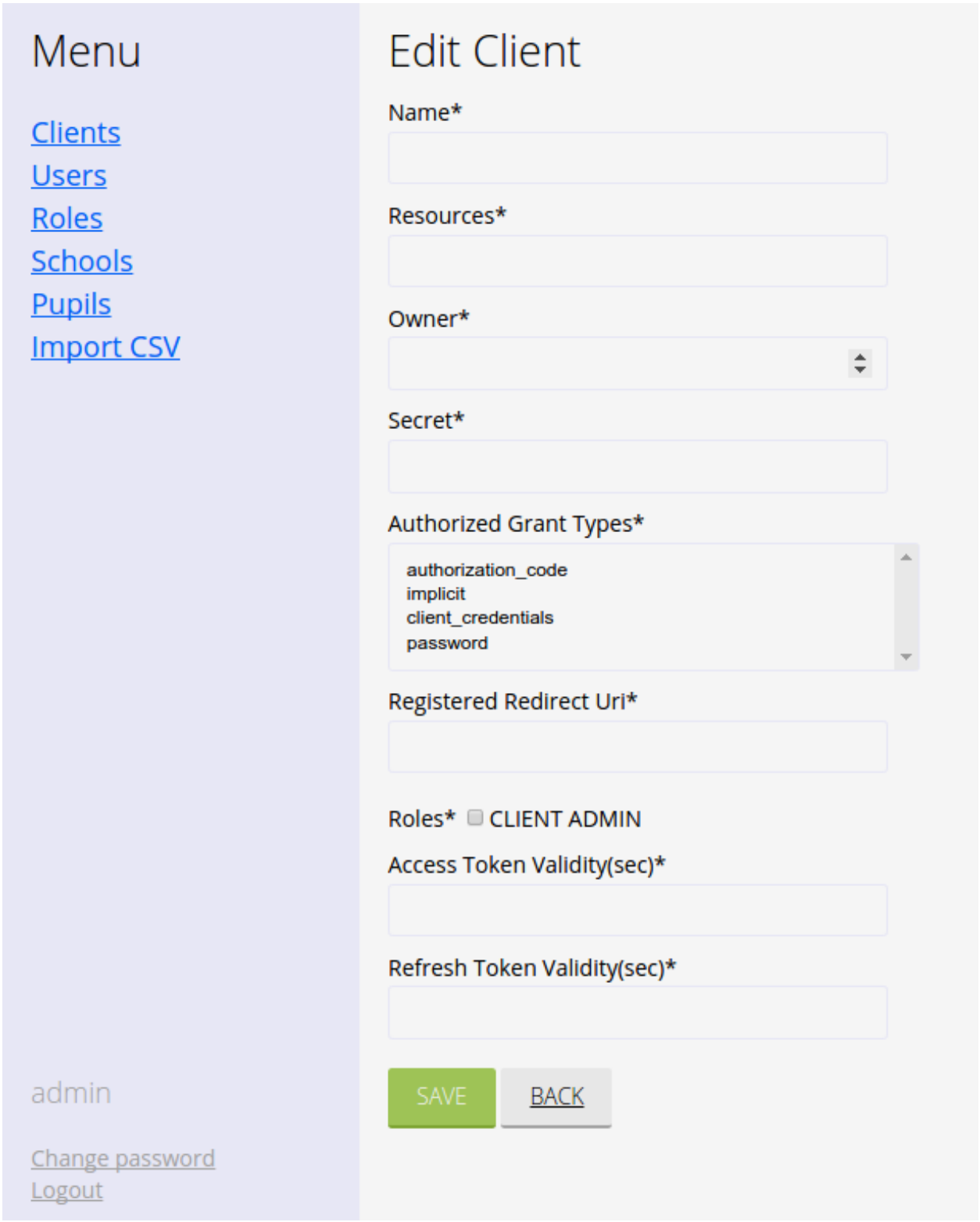
If you see this image, everything is good, congratulations!

Walkthrough: iVIS Server Configuration

- *Create or edit iVIS client*
- *Create or edit iVIS role*

Create/Edit iVIS client

Client is any client application that connects to the iVIS Server and works with data. To create a client go to the clients list and click Create button.



Name

The name of client.

Resources

The name of resource which client obtain.

Owner

The id of User which obtain client.

Secret

The password of client for access to resources.

Authorized Grant Types

This section allows choose ways for obtain access token for client

authorization_code
: About this you can read at Authorization Code Grant

implicit
: About this you can read at Implicit Grant

client_credentials
: About this you can read at Client Credentials Grant

password
: About this you can read at Resource Owner Password Credentials Grant

Registered Redirect Uri

URL for access token or authorization code response.

Roles

Role (roles) that consist from set of permissions for API access.

Described next at *Create or edit iVIS role* .

Access Token Validity(sec)

Number of seconds after which the access token expires, and is no longer valid.

Refresh Token Validity(sec)

Number of seconds after which the refresh token expires, and is no longer valid.

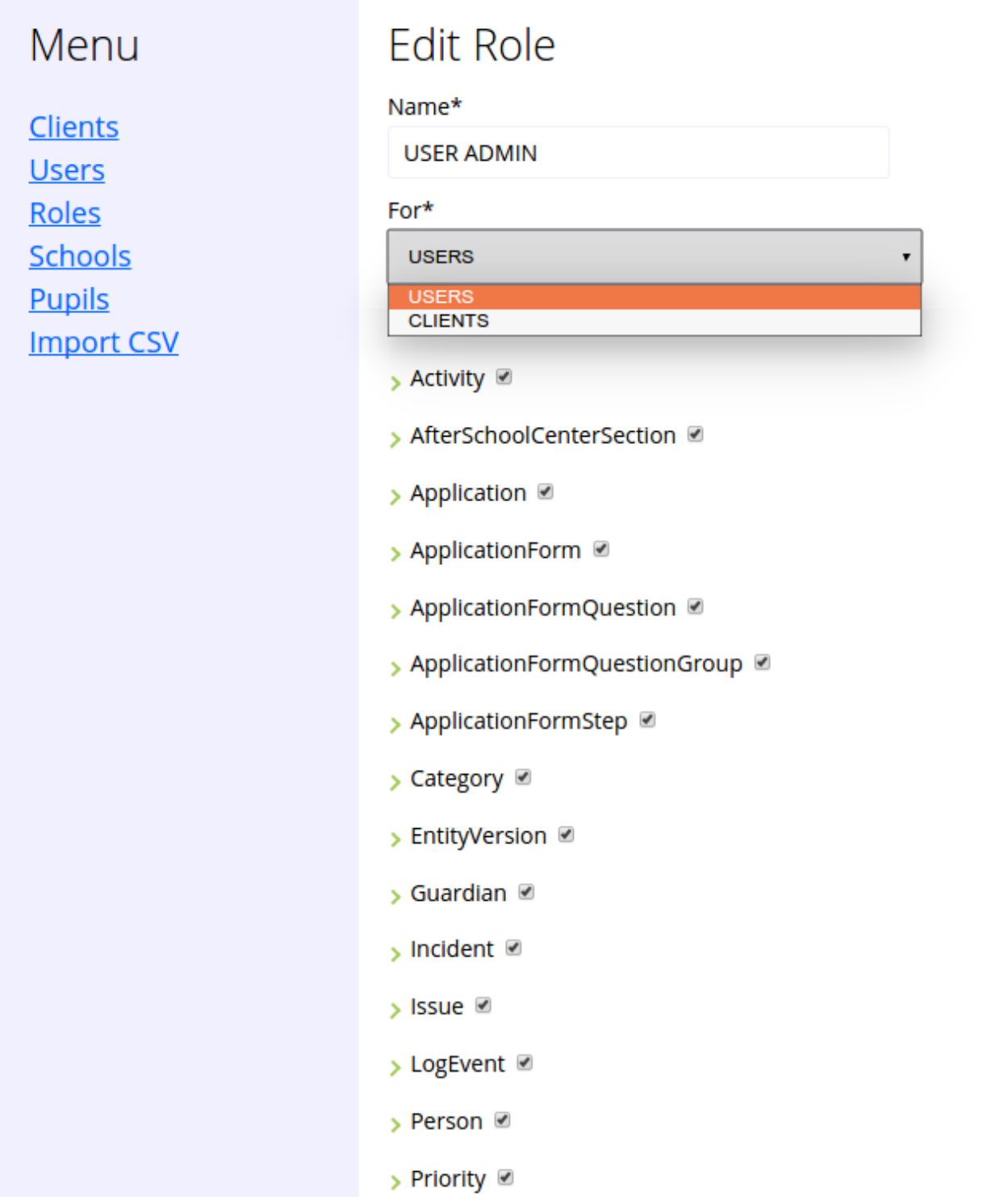
Note: Refresh token validity must be longer than access.

See also:

Read about Access Token and Refresh Token

Create or edit iVIS role

Role in iVIS is a set of permissions to access method API, separated into groups by entity name.



Name

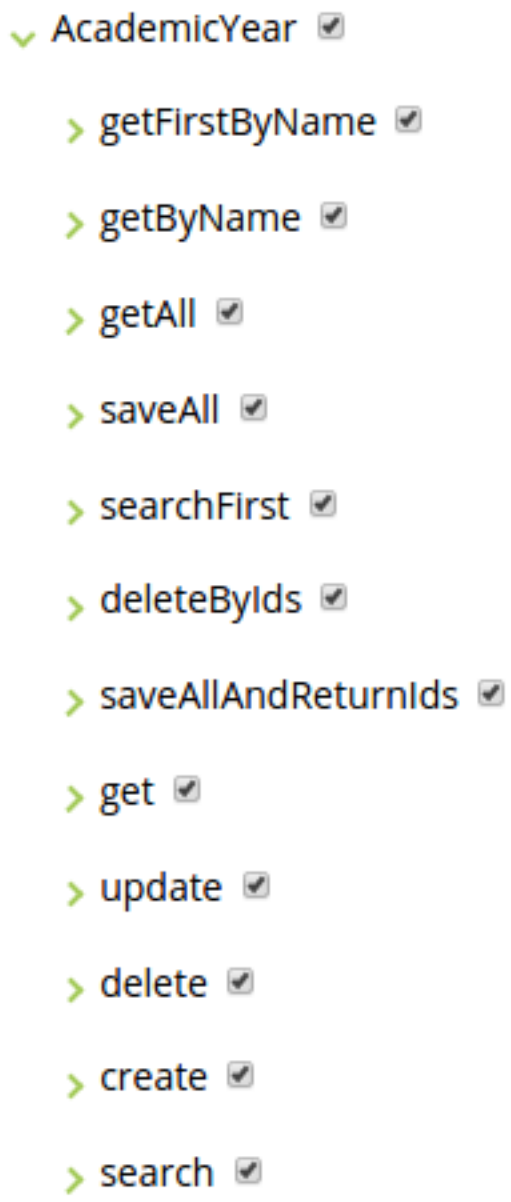
The name of role.

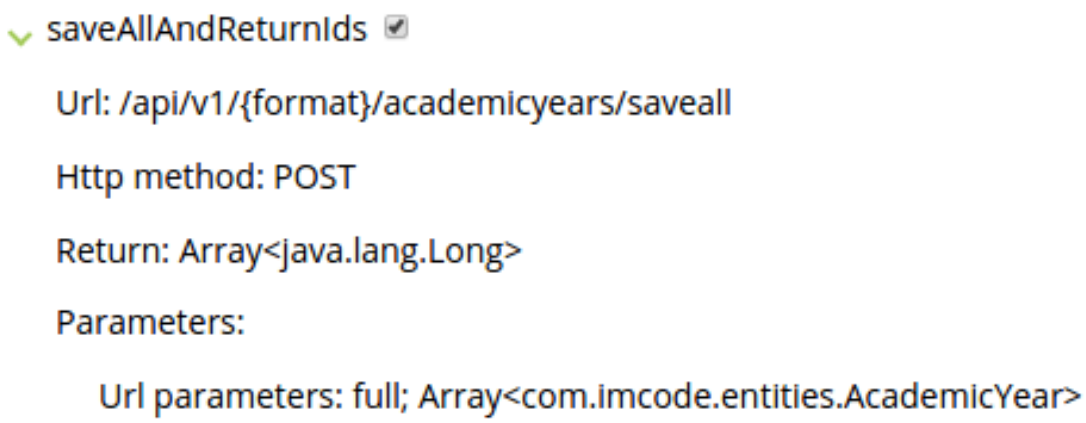
For

Define role purpose.

List of permissions

Permission represents as method API with detail description.





Tip: Checkbox near entity name has three state. It indicates that in group checked no one/all/some. Also it provides possibility to check/uncheck all permissions in any group.

Walkthrough: Install iVIS Server Client Applications

Walkthrough: Install Imcms based client application

Prerequisites

First of all you need install iVIS server. Visit this page for details. When you have working iVIS server we can continue, but to make it possible to communicate with the iVIS Server from the client application you need to register your client application inside the iVIS Server. Here is the walkthrough that describes this process.

This client is based on imCMS.

Git configuration

Clone project from GitHub repository:

```
cd /../directory #Directory where project folder with content must be placed

git clone https://github.com/imCodePartnerAB/iVIS-imCMS-Client-Sample.git
```

After cloning will created subdirectory iVIS with content.

Database configuration

In the Terminal (Ctrl+ALt+T) execute following commands to create the database:

```
mysql -u {username} -p{password} #{username} - database username, {password} -␣
↪database password

CREATE DATABASE db_ivis_info;
```

Download file dump_db_ivis_info.sql from release on Github.

In the Terminal (Ctrl+ALt+T) execute following command to run dump file on created DB:

```
mysql -u {username} -p{password} db_ivis_info < /home/downloads/dump_db_ivis_info.sql
↪#file location after downloading
```

Edit file **local.server.properties** from project directory in the next way:

- User[1];
- Password[2];
- JdbcUrl[3];
- Hibernate.hbm2ddl [#]_;
- Hibernate.dialect [#]_;
- Server.name [#]_;

Run application

Important: iVIS Server must be installed from this guide.

In the Terminal (Ctrl+ALt+T) execute following commands:

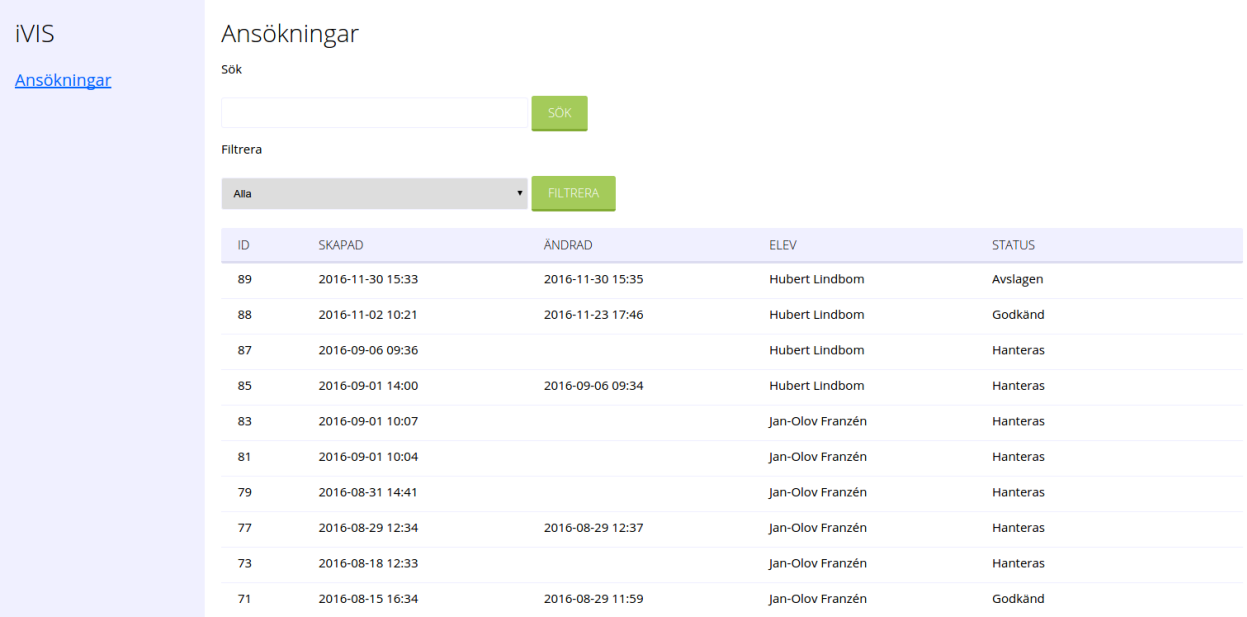
```
cd /../iVIS-imCMS-Client-Sample #path to project from Github

mvn tomcat7:deploy #deploy configured to localhost:8080/imcmscl.
```

Type http://localhost:8080/imcmscl in browser.

Input Login=admin and Password=pass.

If you see this image, everything is good, congratulations!

[1] **username** of database (default *root*)
[2] **password** of database
[3] consist of **jdbc:{provider}://{hostname}:{port}/{databaseName}?{encoding}** for mysql {provider} = *mysql*, {hostname} = *localhost*, {port} = 3306, {encoding} = *utf-8*



Walkthrough: Install OeP based Client application

Prerequisites

First of all you need install iVIS server. Visit this page for details. When you have working iVIS server we can continue, but to make it possible to communicate with the iVIS Server from the client application you need to register your client application inside the iVIS Server. Here is the walkthrough that describes this process.

This client is based on OeP.

Git configuration

Clone project from GitHub repository:

```
cd /../directory #Directory where project folder with content must be placed

git clone https://github.com/imCodePartnerAB/iVIS-OeP-Client-Sample.git
```

After cloning will created subdirectory iVIS with content.

Database configuration

In the Terminal (Ctrl+ALt+T) execute following commands to create the database:

```
mysql -u {username} -p{password} #{username} - database username, {password} -␣
↪database password

CREATE DATABASE db_open_platform;
```

Download file dump_db_open_platform.sql from release on Github.

In the Terminal (Ctrl+ALt+T) execute following command to run dump file on created DB:

```
mysql -u {username} -p{password} db_open_platform < /home/downloads/dump_db_open_
↪platform.sql #file location after downloading
```

Edit file **config.xml** from *demo.oeplatform.org/src/main/webapp/WEB-INF* subfolder in the next way:

- <Username>${username}</Username>[1];
- <Password>${password}</Password>[2];
- <Url>${jdbc-url}</Url>[3];

Run application

Important: iVIS Server must be installed from this guide.

In the Terminal (Ctrl+ALt+T) execute following commands:

```
cd /../iVIS-OeP-Client-Sample #path to project from Github

mvn clean install

cd demo.oeplatform.org

mvn tomcat7:deploy #deploy configured to localhost:8080/oepcl.
```

Type http://localhost:8080/oepcl in browser.

Click “Logga in”.

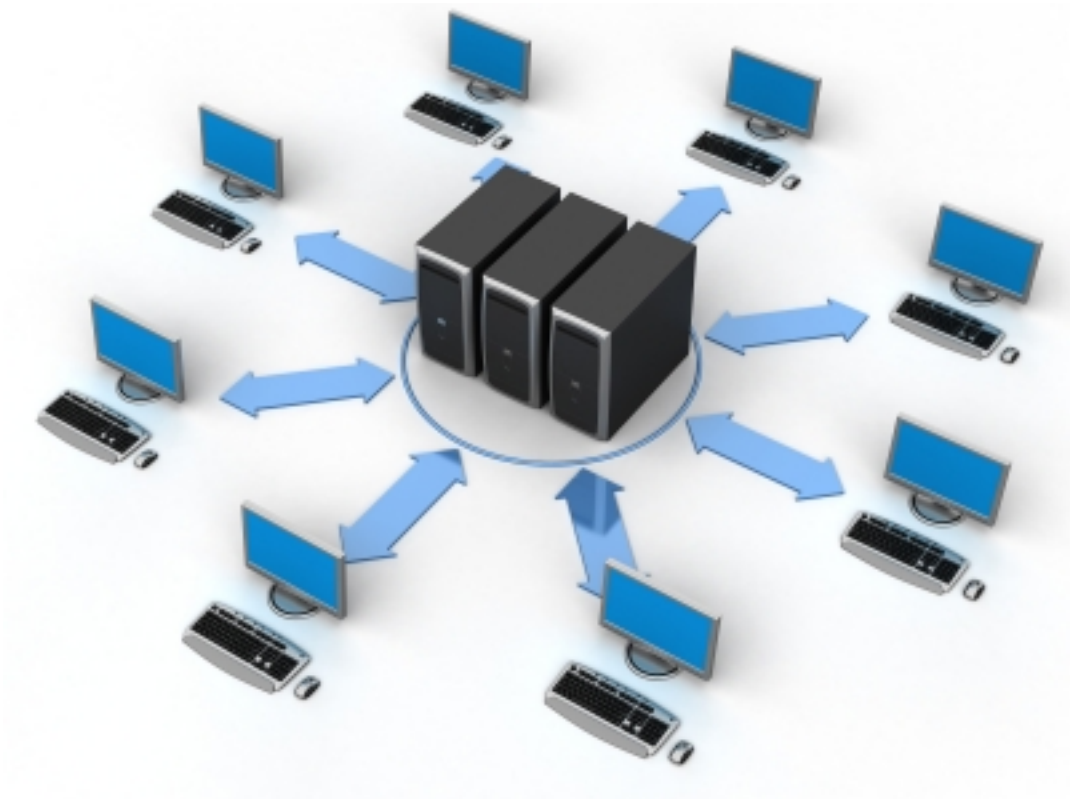
Input Login=admin and Password=pass.

If you see this image, everything is good, congratulations!

[1] **username** of database (default *root*)
[2] **password** of database
[3] consist of **jdbc:{provider}://{hostname}:{port}/{databaseName}?{encoding}** for mysql {provider} = *mysql*, {hostname} = *localhost*, {port} = 3306, {encoding} = *utf-8*

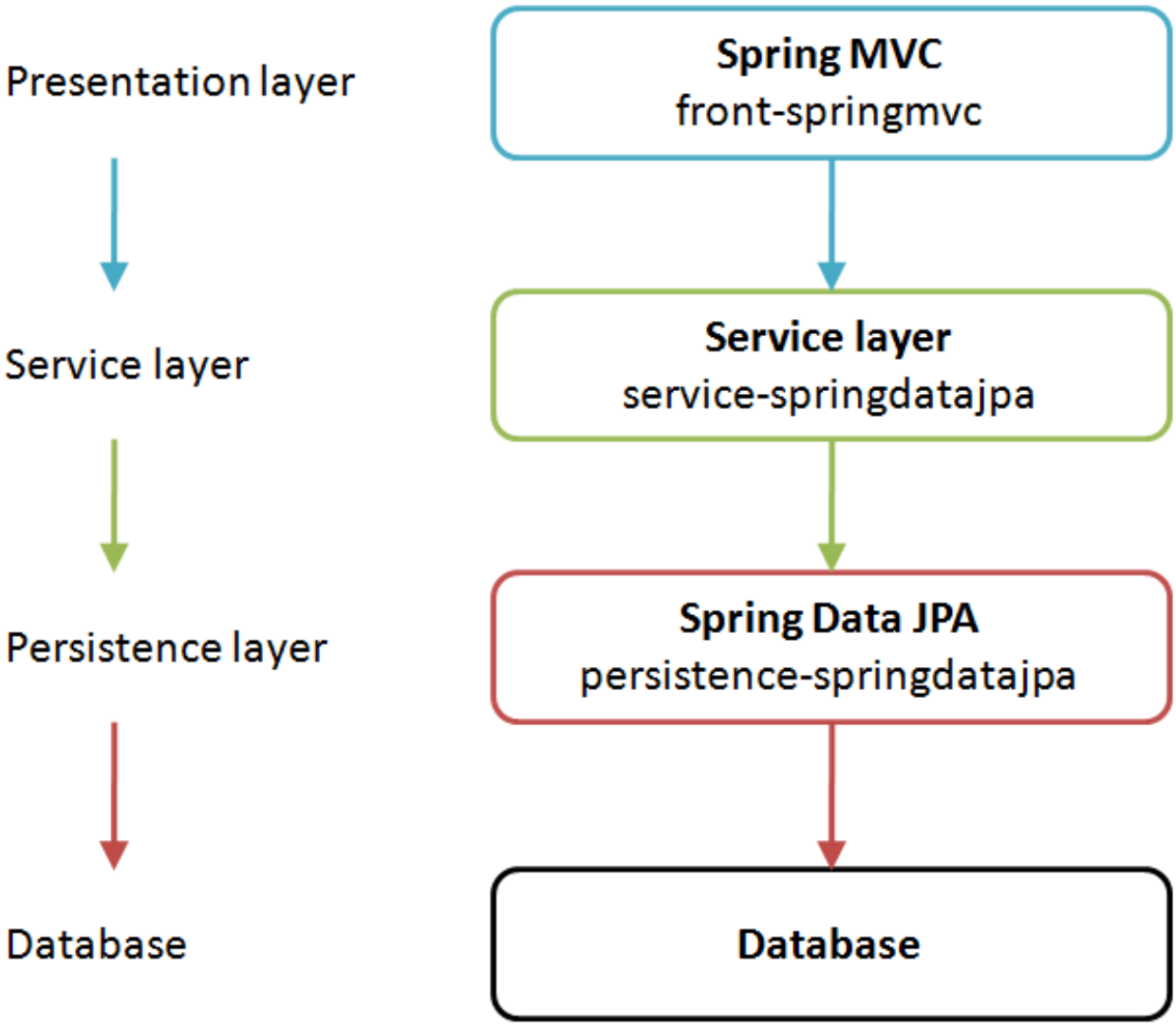
Basic Concepts

Generally iVIS system consists of iVIS server and N clients:



iVIS Server itself is split into 4 main layers:

1. Database Layer.
2. Data Access Layer.
3. Security Layer.
4. API Layer.



Front-end side also reviewed.

Database Layer

- *Database information*
- *Integrity*

Database information

Currently iVIS database represented by MySQL.

You can change it in properties (*JdbcDriver*, *JdbcUrl*, *Hibernate.dialect*).

Note: Using something other database instead MySQL not tested.

Integrity

Reporting about changing of information storing in DB something other way (e.g. manually run sql script) instead Hibernate is missing.

If database has not mapped on entity columns, they are ignored by Hibernate.

Data Access Layer

Data access layer in iVIS is implemented using standard JPA (using Hibernate as ORM provider).

All entities are mapped to the corresponding tables. All tables (except authorization ones) have prefix "**dbo_**" that means "database object". Except of authorization tables (it relate with ideological considerations).

Database diagram `download`.

JPA classes located in *com.imcode.entities* of ivis-core module in iVIS project.

Class diagram `download`.

JPA entities diagram `download`.

Relations in DB also are implement by standard JPA.

Note: Embedded entities don't have id and can't exist without parent entity.

List of all tables and description

dbo_academic_year

Represents AcademicYear entity.

dbo_activity

Represents Activity entity.

dbo_after_school_center_section

Represents AfterSchoolCenterSection entity.

dbo_application

Represents Application entity.

dbo_application_form

Represents ApplicationForm entity.

dbo_application_form_question

Represents ApplicationFormQuestion entity.

dbo_application_form_question_group

Represents ApplicationFormQuestionGroup entity.

dbo_application_form_step

Represents ApplicationFormStep entity.

dbo_category

Represents Category entity.

dbo_entity_version

Represents EntityVersion entity.

dbo_guardian

Represents Guardian entity.

dbo_incident

Represents Incident entity.

dbo_incident_category_cross

Represents many to many relation between Incident and Category entities.

dbo_incident_pupil_cross

Represents many to many relation between Incident and Pupil entities.

dbo_issue

Represents Issue entity.

dbo_issue_category_cross

Represents many to many relation between Issue and Category entities.

dbo_issue_pupil_cross

Represents many to many relation between Issue and Pupil entities.

dbo_issues_authorized_persons_cross

Represents many to many relation between Issue and Person entities.

dbo_log_event

Represents LogEvent entity.

dbo_person

Represents Person entity.

dbo_person_address

Represents Address embeddable in Person.

dbo_person_email

Represents Email embeddable in Person.

dbo_person_phone

Represents Phone embeddable in Person.

dbo_priority

Represents Priority entity.

dbo_pupil

Represents Pupil entity.

dbo_pupil_after_school_center_schema

Represents AfterSchoolCenterSchema embeddable in Pupil.

dbo_pupil_guardians_cross

Represents many to many relation between Pupil and Guardian entities.

dbo_role

Represents Role entity.

dbo_role_permission_cross

Represents many to many relation between Role and Permission entities.

dbo_school

Represents School entity.

dbo_school_class

Represents SchoolClass entity.

dbo_school_class_diaries

Represents Diary embeddable in SchoolClass.

dbo_school_service_cross

Represents ServiceTypeEnum element collection, which equals embeddable in School.

dbo_school_transport

Represents SchoolTransport entity.

dbo_status

Represents Status entity.

dbo_truancy

Represents Truancy entity.

dbo_user

Represents User entity.

dbo_user_roles_cross

Represents many to many relation between User and Role entities.

AUTHORIZATION TABLES

oauth_access_token

Represents AccessToken entity. Managed by Spring Security.

oauth_refresh_token

Represents RefreshToken entity. Managed by Spring Security.

dbo_oauth_client_details

Represents JpaClientDetails entity.

dbo_oauth_client_additional_info

Element collection in JpaClientDetails.

dbo_oauth_client_grant_types

Element collection in JpaClientDetails.

dbo_oauth_client_redirect_uris

Element collection in JpaClientDetails.

dbo_oauth_client_resources

Element collection in JpaClientDetails.

dbo_oauth_client_roles_cross

Represents many to many relation between JpaClientDetails and Role entities.

Security Layer

iVIS uses OAuth 2.0 protocol (implemented by standard Spring Security provider). iVIS is identity provider for any client application that uses it. It means that if user wants to use some iVIS client application he has to login on iVIS (and receive token in the background). It works the same way as popular social networks. In addition, iVIS can use third-party identity providers too. So user after he is redirected to the iVIS login page may choose the option to login using BankId for example.

Users registration

Each user has to be registered in iVIS and obtain some permissions from iVIS administration before he can use the system and any of it's client applications. To become a registered user you need to fill out the form on http://ivis.dev.imcode.com/registration. After that you will have your username and password. All passwords are stored as bcrypt hashes so they can't be read.

While the login username and password from the login page is sent over HTTPS connection using the SSL encryption (if SSL certificate is installed; note: it is not installed on the dev iVIS Server).

Permissions

Allowed users actions set is controlled as intersection of client and user permissions. So each iVIS client application has set of permissions (defined by the iVIS system administrators). User have permissions defined by administrators either. So the resulting user permissions in the given client is intersection of client and user permissions. For example, if the client has permissions to use resource A, B and C and the user has permissions for the resources B, C and D, resulting user permissions using the given client is B and C.

Permission is access to the given API action or some piece of data, it is controlled on the low level of the iVIS API.

Data encryption

Currently data in the iVIS database is stored as plain text, without encryption.

Error handling

iVIS handles the errors on 5 stages:

1. Validation errors.
2. Database level errors.
3. JSON/XML mapping errors.
4. Security errors.
5. Other errors.

Validation errors

Handling and providing corresponding messages about missing required fields, too long text values etc.

Note: Validation works on both sides (client and server). On client powered by **jQuery Validation Plugin**. On server powered by **Spring Validator** interface. Validation use cases in details described here.

Database level errors

Handling and providing corresponding messages about database level errors, like missing values with the given key etc.

JSON/XML mapping errors

Missing or extra fields etc.

Security errors

Expired or invalid tokens etc.

Other errors

All other errors.

Handling user password

Password management

Password handling

Passwords handling divide into saving password and process password in authorization context.

Saving Password

Password is handled by **Spring MVC Controller**. After form submitting from password generate bcrypt hash and hash persisted to database.

Password in authorization context

In authorization process passwords are handled by **Spring Security** standard filters request. Handling means get password from login page, compare password with hash from database.

Password policy

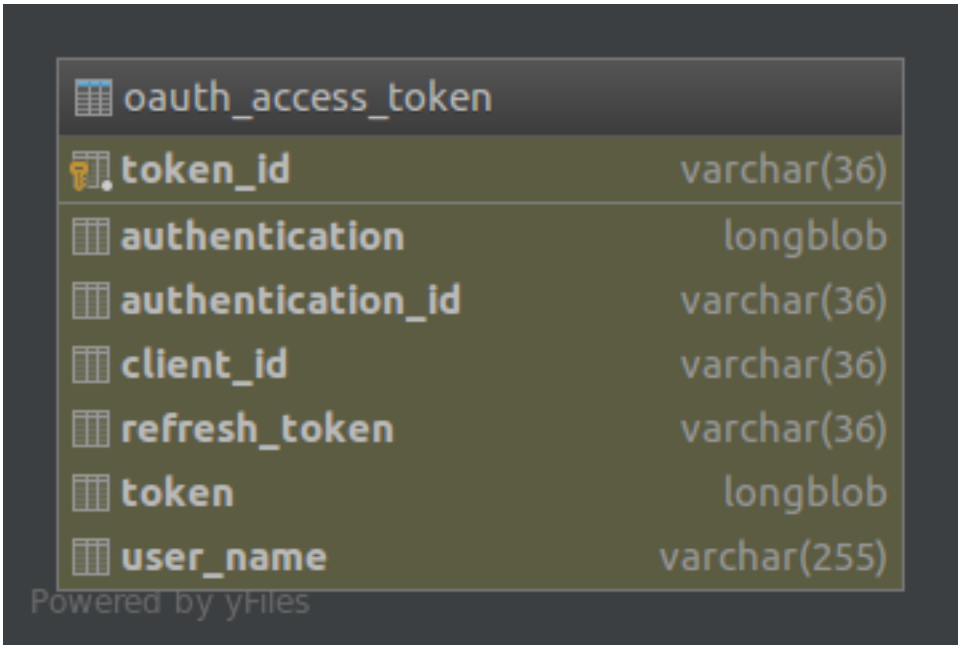
Passwords must have **8 characters**.

Check for password strength is absent.

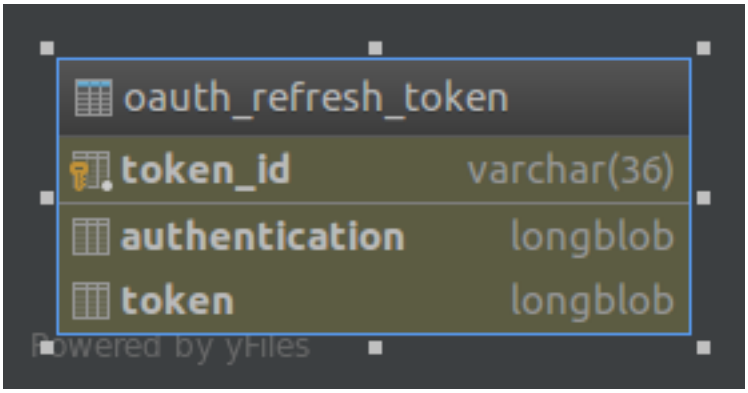
More about OAuth 2.0 implementation

There are two main entities in the iVIS OAuth 2.0 implementation: AccessToken and RefreshToken. Their classes are mapped on the corresponding tables:

oauth_access_token



oauth_refresh_token



The provider role in OAuth 2.0 is actually split between Authorization Service and Resource Service, and these reside in the iVIS with Spring Security OAuth. The requests for the tokens are handled by Spring MVC controller endpoints, and access to protected resources is handled by standard Spring Security request filters. The following endpoints are required in the Spring Security filter chain in order to implement OAuth 2.0 Authorization Server:

- Authorization Endpoint is used to service requests for authorization (URL: /oauth/authorize)
- Token Endpoint is used to service requests for access token (URL: /oauth/token)

You can find details here.

API Layer

API is described in details here.

Front-end side

Endpoint pages

Front-end side based on **HTML+CSS+JS**.

JS in iVIS powered by **jQuery** library.

To expand jQuery functionality was used several plugins:

1. jQuery Validation Plugin;
2. jQuery Tristate Plugin.

Server side pages

Source code of endpoint pages placed in **JSPs**.

JSPs designed with **JSTL** and **Spring** tag libraries.

Apache tiles composed several JSP views with templates to one endpoint page.

Authorization

Few words about OAuth 2.0

In OAuth 2.0 concept for authorization defines 4 different ways, they are called Authorized Grant Types.

They are:

- authorization code
- implicit
- client credentials
- password

iVIS authorization

According to Authorized Grant Type there 4 ways to be authorized in iVIS.

In iVIS administrator define which Authorized Grant Type must use client (it can be 1 or all together).

For authorization client user we recommend use **authorization code grant**.

We implemented authorization of client user in next way:

1. User which want to login click login.
2. Client app send redirect to iVIS server with client credentials (see *Step 1 Get authorization code*).
3. User input username and password and click Login.
4. iVIS redirect back (according to redirect url) with parameter code.
5. Client based on code make request to obtain access token(see *Step 1 Get access token with authorization code*).
6. As response client get access token object.
7. Every API request from client must have the access_token (property from received object).

Authorization in details

Basic there are two steps working with authorization.

Step 1

Get authorization code

In order to be authorized and to obtain the token you have to get authorization code first by sending GET request to

/oauth/authorize?response_type=code&client_id={yourClientId}&redirect_uri={redirectUrl}&display=popup&scope={scope}

where

{yourClientId} is ID of your client (provided by iVIS administrators)

{redirectUrl} is the URL that will receive authorization code after successful authorization

{scope} is the list of the required permissions (currently you can use 'read+write')

Code example **Java** using org.apache.http package

```
    String authorizeUrl = "http://ivis.dev.imcode.com/oauth/authorize";
    String redirectUri = "{yourClientRedirectUrl}";
    String clientId = "{yourClientId}";

    URIBuilder builder = new URIBuilder(authorizeUrl);
    builder.addParameter("response_type", "code");
    builder.addParameter("client_id", clientId);
    builder.addParameter("redirect_uri", redirectUri);
    builder.addParameter("display", "popup");

    String path = builder.build().toString();
    response.sendRedirect(path);
```

Code example **JS** using JQuery

```
    var authorizeURI = "http://ivis.dev.imcode.com/oauth/authorize";
    var redirectUri = "{yourClientRedirectUrl}";
    var clientId = "{yourClientId}";

    var data = {
        'response_type' : 'code',
        'client_id' : clientId,
        'redirect_uri' : redirectUri,
        'display' : 'popup'
```

```
        };

    location.href = authorizeURI + '?' + $.param(data);
```

Get access token with authorization code

When you have the authorization code (it is set as url parameter to {yourClientRedirectUrl}) you can try to get token by sending POST request to

/oauth/token

with parameters

code (= '{code}')

redirect_uri (= '{redirectUrl}')

grant_type = (= 'authorization_code')

Also you need to send client_id and client_secret in request headers. Header parameter looks like following: Authorization (="Basic " + ConvertBase64Encoding(client_id + ":" + client_secret)).

As response to the redirect_uri you will receive json object with next properties:

access_token (token for access to API)

refresh_token (when token is expired, you can exchange refresh_token to new access_token, see step 2)

expires_in (property is a number of seconds after which the access token expires, and is no longer valid)

access_token object has also another properties, but they aren't necessary for accessing to API.

Code example **Java** using org.apache.http package

```
    String tokenURI = "http://ivis.dev.imcode.com/oauth/token";
    String redirectURI = "{redirectUrl}";

    String clientId = "{yourClientId}";
    String clientSecret = "{yourClientSecret}";
    String base64IdAndSecretColonSeparated = new String(
             Base64.getEncoder().encode(
                     (clientId + ":" + clientSecret)
                             .getBytes())
    );

    List<NameValuePair> pairsPost = new LinkedList<NameValuePair>();
    pairsPost.add(new BasicNameValuePair("code", request.getParameter("code")));
    pairsPost.add(new BasicNameValuePair("redirect_uri", redirectURI));
    pairsPost.add(new BasicNameValuePair("grant_type", "authorization_code"));

    HttpPost post = new HttpPost(tokenURI);

    post.setEntity(new UrlEncodedFormEntity(pairsPost));
    post.setHeader("Authorization", "Basic " + base64IdAndSecretColonSeparated);

    HttpClient client = HttpClientBuilder.create().build();
    HttpResponse response = client.execute(post);

    String token = EntityUtils.toString(response.getEntity()); //there is the json␣
↪object response
```

Code example **JS** using JQuery

```
    var tokenURI = "http://ivis.dev.imcode.com/oauth/token";
    var redirectUri = "{redirectUrl}";

    var clientId = "{yourClientId}";
    var clientSecret = "{yourClientSecret}";
    var base64IdAndSecretColonSeparated = btoa(clientId + ':' + clientSecret);//IE 10 and higher
    //For security improvement last line is recommended to generate on server side.

    var code = location.href.split('code=')[1];//get value of parameter code
    // it's only one param, so you can use this way to get code, or write your own

    $.post({
        url : tokenURI,
        data : {
            'code' : code,
            'redirect_uri' : redirectUri,
            'grant_type' : 'authorization_code'
        },
        beforeSend : function (xhr) {
             xhr.setRequestHeader ("Authorization", "Basic " + base64IdAndSecretColonSeparated);
        },
        success : function (token) {
              alert(token['access_token']); //use received token
              alert(token['refresh_token']);
              alert(token['expires_in']);
         }
     });
```

Step 2

When your token is expired you can refresh (update) it without repeating authorization by sending POST request to

/oauth/token

with parameters

refresh_token (='{yourRefreshToken}') - is the refresh token from the step 1

grant_type (= 'refresh_token')

Also you need to send client_id and client_secret in request headers. Header parameter looks like following: Authorization (="Basic " + ConvertBase64Encoding(client_id + ":" + client_secret)).

Code example **Java** using org.apache.http package

```
    String tokenURI = "http://ivis.dev.imcode.com/oauth/token";
    String refreshToken = "{yourRefreshToken}";

    String client_id = "{yourClientId}";
    String client_secret = "{yourClientSecret}";
    String base64IdAndSecretColonSeparated = new String(
             Base64.getEncoder().encode(
                     (clientId + ":" + clientSecret)
                             .getBytes())
    );
```

```

    List<NameValuePair> pairsPost = new LinkedList<NameValuePair>();
    pairsPost.add(new BasicNameValuePair("refresh_token", refreshToken));
    pairsPost.add(new BasicNameValuePair("grant_type", "refresh_token"));

    HttpPost post = new HttpPost(tokenURI);

    post.setEntity(new UrlEncodedFormEntity(pairsPost));
    post.setHeader("Authorization", "Basic " + base64IdAndSecretColonSeparated);

    HttpClient client = HttpClientBuilder.create().build();
    HttpResponse response = client.execute(post);

    String token = EntityUtils.toString(response.getEntity()); //there is a json␣
↪object response
```

Code example **JS** using JQuery

```
    var tokenURI = "http://ivis.dev.imcode.com/oauth/token";
    var refreshToken = "{yourRefreshToken}";

    var client_id = "{yourClientId}";
    var client_secret = "{yourClientSecret}";
    var base64IdAndSecretColonSeparated = btoa(clientId + ':' + clientSecret);//IE 10␣
↪and higher
    //For security improvement last line is recommended to generate on server side.

    $.post({
        url : tokenURI,
        data : {
            'refresh_token' : refreshToken,
            'grant_type' : 'refresh_token'
        },
        beforeSend : function (xhr) {
             xhr.setRequestHeader ("Authorization", "Basic " +␣
↪base64IdAndSecretColonSeparated);
        },
        success : function (token) {
                        alert(token['access_token']); //use received token
                        alert(token['refresh_token']);
                        alert(token['expires_in']);
         }
     });
```

Remark

You need to control expiry of the both tokens:

Make request until access token is good.

Note: To make request you need include token to header parameter:

Authorization (= 'Bearer {access_token}')

Next if token isn't good (expired or you can't make API call) exchange refresh token to new access token.

Tip: Save refresh token as cookie to manage them for a long time.

Finally if refresh token isn't good (expired or you can't exchange it to access) re-login user again.

API

You can't using API unauthorized, so you look first at:

iVIS provides API for data manipulation using JSON or XML format.

In general, the process looks like:

1. Authorization and obtaining token.
2. Sending requests and receiving responses.

Root for the all relative URLs is 'http://ivis.dev.imcode.com/';

Base relative URL for API: /api/v1/{format}

where

{format} is 'json' or 'xml'

Each request requires additional parameter access_token.

If method has url, it concatenates with base URL.

You can have access and operate with following groups of entities:

BASE

AcademicYears

`(implementation of AcademicYear entity)`

Provides following method for API calls:

- *Create*
- *Delete*
- *DeleteByIds*
- *Get*
- *GetAll*
- *GetByName*
- *GetFirstByName*
- *SaveAll*
- *SaveAllAndReturnIds*
- *Search*
- *SearchFirst*
- *Update*

Create

URL:

/api/v1/{format}/academicyears

Method:

POST

Parameters request:

Object< AcademicYear >

Parameters response:

Object

Description:

1. name(STRING)
2. id(NUMBER)

Delete

URL:

/api/v1/{format}/academicyears/{id}

Method:

DELETE

Parameters request:

null

Parameters response:

Object

Description:

1. name(STRING)
2. id(NUMBER)

DeleteByIds

URL:

/api/v1/{format}/academicyears

Method:

DELETE

Parameters request:

Url parameters: ids

Parameters response:

Array

Description:

1. name(STRING)
2. id(NUMBER)

Get

URL:

/api/v1/{format}/academicyears/{id}

Method:

GET

Parameters request:

null

Parameters response:

Object

Description:

1. name(STRING)
2. id(NUMBER)

GetAll

URL:

/api/v1/{format}/academicyears

Method:

GET

Parameters request:

null

Parameters response:

Array

Description:

1. name(STRING)
2. id(NUMBER)

GetByName

URL:

/api/v1/{format}/academicyears

Method:

GET

Parameters request:

Url parameters: name

Parameters response:

Array

Description:

1. name(STRING)
2. id(NUMBER)

GetFirstByName

URL:

/api/v1/{format}/academicyears

Method:

GET

Parameters request:

Url parameters: name, first

Parameters response:

Object

Description:

1. name(STRING)
2. id(NUMBER)

SaveAll

URL:

/api/v1/{format}/academicyears/saveall

Method:

POST

Parameters request:

Array< AcademicYear >

Parameters response:

Array

Description:

1. name(STRING)
2. id(NUMBER)

SaveAllAndReturnIds

URL:

/api/v1/{format}/academicyears/saveall

Method:

POST

Parameters request:

Url parameters: full

Array< AcademicYear >

Parameters response:

Array

Description: ARRAY<NUMBER>

Search

URL:

/api/v1/{format}/academicyears/search

Method:

POST

Parameters request:

Array< SearchCriteries$SearchCriteriaResult >

Parameters response:

Array

Description:

1. name(STRING)
2. id(NUMBER)

SearchFirst

URL:

/api/v1/{format}/academicyears/search/first

Method:

POST

Parameters request:

Array< SearchCriteries$SearchCriteriaResult >

Parameters response:

Object

Description:

1. name(STRING)
2. id(NUMBER)

Update

URL:

/api/v1/{format}/academicyears/{id}

Method:

PUT

Parameters request:

Object< AcademicYear >

Parameters response:

Object

Description:

1. name(STRING)
2. id(NUMBER)

AfterSchoolCenterSections

`(implementation of AfterSchoolCenterSection entity)`

Provides following method for API calls:

- *Create*
- *Delete*
- *DeleteByIds*
- *Get*
- *GetAll*
- *SaveAll*
- *SaveAllAndReturnIds*
- *Search*
- *SearchFirst*
- *Update*

Create

URL:

/api/v1/{format}/afterschoolcentersections

Method:

POST

Parameters request:

Object< AfterSchoolCenterSection >

Parameters response:

Object

Description:

1. school(OBJECT< School >)
2. name(STRING)
3. id(NUMBER)

Delete

URL:

/api/v1/{format}/afterschoolcentersections/{id}

Method:

DELETE

Parameters request:

null

Parameters response:

Object

Description:

1. school(OBJECT< School >)
2. name(STRING)
3. id(NUMBER)

DeleteByIds

URL:

/api/v1/{format}/afterschoolcentersections

Method:

DELETE

Parameters request:

Url parameters: ids

Parameters response:

Array

Description:

1. school(OBJECT< School >)
2. name(STRING)

3. id(NUMBER)

Get

URL:

/api/v1/{format}/afterschoolcentersections/{id}

Method:

GET

Parameters request:

null

Parameters response:

Object

Description:

1. school(OBJECT< School >)
2. name(STRING)
3. id(NUMBER)

GetAll

URL:

/api/v1/{format}/afterschoolcentersections

Method:

GET

Parameters request:

null

Parameters response:

Array

Description:

1. school(OBJECT< School >)
2. name(STRING)
3. id(NUMBER)

SaveAll

URL:

/api/v1/{format}/afterschoolcentersections/saveall

Method:

POST

Parameters request:

Array< AfterSchoolCenterSection >

Parameters response:

Array

Description:

1. school(OBJECT< School >)
2. name(STRING)
3. id(NUMBER)

SaveAllAndReturnIds

URL:

/api/v1/{format}/afterschoolcentersections/saveall

Method:

POST

Parameters request:

Url parameters: full

Array< AfterSchoolCenterSection >

Parameters response:

Array

Description: ARRAY<NUMBER>

Search

URL:

/api/v1/{format}/afterschoolcentersections/search

Method:

POST

Parameters request:

Array< SearchCriteries$SearchCriteriaResult >

Parameters response:

Array

Description:

1. school(OBJECT< School >)
2. name(STRING)
3. id(NUMBER)

SearchFirst

URL:

/api/v1/{format}/afterschoolcentersections/search/first

Method:

POST

Parameters request:

Array< SearchCriteries$SearchCriteriaResult >

Parameters response:

Object

Description:

1. school(OBJECT< School >)
2. name(STRING)
3. id(NUMBER)

Update

URL:

/api/v1/{format}/afterschoolcentersections/{id}

Method:

PUT

Parameters request:

Object< AfterSchoolCenterSection >

Parameters response:

Object

Description:

1. school(OBJECT< School >)
2. name(STRING)
3. id(NUMBER)

Guardians

`(implementation of Guardian entity)`

Provides following method for API calls:

- *Create*
- *Delete*
- *DeleteByIds*

- *Get*
- *GetAll*
- *GetByPersonalId*
- *GetFirstByPersonalId*
- *SaveAll*
- *SaveAllAndReturnIds*
- *Search*
- *SearchFirst*
- *Update*

Create

URL:

/api/v1/{format}/guardians

Method:

POST

Parameters request:

Object< Guardian >

Parameters response:

Object

Description:

1. person(OBJECT< Person >)
2. pupils(ARRAY< Pupil >)
3. id(NUMBER)

Delete

URL:

/api/v1/{format}/guardians/{id}

Method:

DELETE

Parameters request:

null

Parameters response:

Object

Description:

1. person(OBJECT< Person >)
2. pupils(ARRAY< Pupil >)
3. id(NUMBER)

DeleteByIds

URL:

/api/v1/{format}/guardians

Method:

DELETE

Parameters request:

Url parameters: ids

Parameters response:

Array

Description:

1. person(OBJECT< Person >)
2. pupils(ARRAY< Pupil >)
3. id(NUMBER)

Get

URL:

/api/v1/{format}/guardians/{id}

Method:

GET

Parameters request:

null

Parameters response:

Object

Description:

1. person(OBJECT< Person >)
2. pupils(ARRAY< Pupil >)
3. id(NUMBER)

GetAll

URL:

/api/v1/{format}/guardians

Method:

GET

Parameters request:

null

Parameters response:

Array

Description:

1. person(OBJECT< Person >)
2. pupils(ARRAY< Pupil >)
3. id(NUMBER)

GetByPersonalId

URL:

/api/v1/{format}/guardians

Method:

GET

Parameters request:

Url parameters: personalId

Parameters response:

Array

Description:

1. person(OBJECT< Person >)
2. pupils(ARRAY< Pupil >)
3. id(NUMBER)

GetFirstByPersonalId

URL:

/api/v1/{format}/guardians

Method:

GET

Parameters request:

Url parameters: personalId, first

Parameters response:

Object

Description:

1. person(OBJECT< Person >)
2. pupils(ARRAY< Pupil >)

3. id(NUMBER)

SaveAll

URL:

/api/v1/{format}/guardians/saveall

Method:

POST

Parameters request:

Array< Guardian >

Parameters response:

Array

Description:

1. person(OBJECT< Person >)
2. pupils(ARRAY< Pupil >)
3. id(NUMBER)

SaveAllAndReturnIds

URL:

/api/v1/{format}/guardians/saveall

Method:

POST

Parameters request:

Url parameters: full

Array< Guardian >

Parameters response:

Array

Description: ARRAY<NUMBER>

Search

URL:

/api/v1/{format}/guardians/search

Method:

POST

Parameters request:

Array< SearchCriteries$SearchCriteriaResult >

Parameters response:

Array

Description:

1. person(OBJECT< Person >)
2. pupils(ARRAY< Pupil >)
3. id(NUMBER)

SearchFirst

URL:

/api/v1/{format}/guardians/search/first

Method:

POST

Parameters request:

Array< SearchCriteries$SearchCriteriaResult >

Parameters response:

Object

Description:

1. person(OBJECT< Person >)
2. pupils(ARRAY< Pupil >)

3. id(NUMBER)

Update

URL:

/api/v1/{format}/guardians/{id}

Method:

PUT

Parameters request:

Object< Guardian >

Parameters response:

Object

Description:

1. person(OBJECT< Person >)
2. pupils(ARRAY< Pupil >)
3. id(NUMBER)

Persons

`(implementation of Person entity)`

Provides following method for API calls:

- *Create*
- *Delete*
- *DeleteByIds*
- *FindByCriteria*
- *Get*
- *GetAll*
- *GetByPersonalId*
- *GetCurrentPerson*
- *GetFirstByPersonalId*
- *GetPersonRolesByPerson*
- *GetPersonRolesOfCurrentPerson*
- *GetSchoolClassesByPerson*

- *GetSchoolClassesOfCurrentPerson*
- *GetSchoolsByPerson*
- *GetSchoolsOfCurrentPerson*
- *GetWorkRolesByPerson*
- *GetWorkRolesOfCurrentPerson*
- *SaveAll*
- *SaveAllAndReturnIds*
- *Search*
- *SearchFirst*
- *Update*

Create

URL:

/api/v1/{format}/persons

Method:

POST

Parameters request:

Object< Person >

Parameters response:

Object

Description:

1. personal_id(STRING)
2. first_name(STRING)
3. last_name(STRING)
4. addresses(KEY_ENUM_OBJECT_PAIR< AddressTypeEnum , Address >)
5. emails(KEY_ENUM_OBJECT_PAIR< CommunicationTypeEnum , Email >)
6. phones(KEY_ENUM_OBJECT_PAIR< CommunicationTypeEnum , Phone >)
7. id(NUMBER)

Delete

URL:

/api/v1/{format}/persons/{id}

Method:

DELETE

Parameters request:

null

Parameters response:

Object

Description:

1. personal_id(STRING)
2. first_name(STRING)
3. last_name(STRING)
4. addresses(KEY_ENUM_OBJECT_PAIR< AddressTypeEnum , Address >)
5. emails(KEY_ENUM_OBJECT_PAIR< CommunicationTypeEnum , Email >)
6. phones(KEY_ENUM_OBJECT_PAIR< CommunicationTypeEnum , Phone >)
7. id(NUMBER)

DeleteByIds

URL:

/api/v1/{format}/persons

Method:

DELETE

Parameters request:

Url parameters: ids

Parameters response:

Array

Description:

1. personal_id(STRING)
2. first_name(STRING)
3. last_name(STRING)
4. addresses(KEY_ENUM_OBJECT_PAIR< AddressTypeEnum , Address >)
5. emails(KEY_ENUM_OBJECT_PAIR< CommunicationTypeEnum , Email >)
6. phones(KEY_ENUM_OBJECT_PAIR< CommunicationTypeEnum , Phone >)
7. id(NUMBER)

FindByCriteria

URL:

/api/v1/{format}/persons

Method:

GET

Parameters request:

Url parameters: search_text, order_by

Parameters response:

Array

Description:

1. personal_id(STRING)
2. first_name(STRING)
3. last_name(STRING)
4. addresses(KEY_ENUM_OBJECT_PAIR< AddressTypeEnum , Address >)
5. emails(KEY_ENUM_OBJECT_PAIR< CommunicationTypeEnum , Email >)
6. phones(KEY_ENUM_OBJECT_PAIR< CommunicationTypeEnum , Phone >)
7. id(NUMBER)

Get

URL:

/api/v1/{format}/person/{id}

Method:

GET

Parameters request:

null

Parameters response:

Object

Description:

1. personal_id(STRING)
2. first_name(STRING)
3. last_name(STRING)
4. addresses(KEY_ENUM_OBJECT_PAIR< AddressTypeEnum , Address >)
5. emails(KEY_ENUM_OBJECT_PAIR< CommunicationTypeEnum , Email >)
6. phones(KEY_ENUM_OBJECT_PAIR< CommunicationTypeEnum , Phone >)
7. id(NUMBER)

GetAll

URL:

/api/v1/{format}/persons

Method:

GET

Parameters request:

null

Parameters response:

Array

Description:

1. personal_id(STRING)
2. first_name(STRING)
3. last_name(STRING)
4. addresses(KEY_ENUM_OBJECT_PAIR< AddressTypeEnum , Address >)
5. emails(KEY_ENUM_OBJECT_PAIR< CommunicationTypeEnum , Email >)
6. phones(KEY_ENUM_OBJECT_PAIR< CommunicationTypeEnum , Phone >)
7. id(NUMBER)

GetByPersonalId

URL:

/api/v1/{format}/persons

Method:

GET

Parameters request:

Url parameters: personalId

Parameters response:

Array

Description:

1. personal_id(STRING)
2. first_name(STRING)
3. last_name(STRING)
4. addresses(KEY_ENUM_OBJECT_PAIR< AddressTypeEnum , Address >)
5. emails(KEY_ENUM_OBJECT_PAIR< CommunicationTypeEnum , Email >)
6. phones(KEY_ENUM_OBJECT_PAIR< CommunicationTypeEnum , Phone >)
7. id(NUMBER)

GetCurrentPerson

URL:

/api/v1/{format}/person/current

Method:

GET

Parameters request:

null

Parameters response:

Object

Description:

1. personal_id(STRING)
2. first_name(STRING)
3. last_name(STRING)
4. addresses(KEY_ENUM_OBJECT_PAIR< AddressTypeEnum , Address >)
5. emails(KEY_ENUM_OBJECT_PAIR< CommunicationTypeEnum , Email >)
6. phones(KEY_ENUM_OBJECT_PAIR< CommunicationTypeEnum , Phone >)
7. id(NUMBER)

GetFirstByPersonalId

URL:

/api/v1/{format}/persons

Method:

GET

Parameters request:

Url parameters: personalId, first

Parameters response:

Object

Description:

1. personal_id(STRING)
2. first_name(STRING)
3. last_name(STRING)
4. addresses(KEY_ENUM_OBJECT_PAIR< AddressTypeEnum , Address >)
5. emails(KEY_ENUM_OBJECT_PAIR< CommunicationTypeEnum , Email >)
6. phones(KEY_ENUM_OBJECT_PAIR< CommunicationTypeEnum , Phone >)
7. id(NUMBER)

GetPersonRolesByPerson

URL:

/api/v1/{format}/person/{id}/personroles

Method:

GET

Parameters request:

null

Parameters response:

Array

Description:

1. person(OBJECT< Person >)
2. role(OBJECT< WorkRole >)
3. school(OBJECT< School >)
4. school_class(OBJECT< SchoolClass >)
5. date_from(NUMBER(Date representation wrapped))
6. date_to(NUMBER(Date representation wrapped))
7. id(NUMBER)

GetPersonRolesOfCurrentPerson

URL:

/api/v1/{format}/person/current/personroles

Method:

GET

Parameters request:

null

Parameters response:

Array

Description:

1. person(OBJECT< Person >)
2. role(OBJECT< WorkRole >)
3. school(OBJECT< School >)
4. school_class(OBJECT< SchoolClass >)
5. date_from(NUMBER(Date representation wrapped))
6. date_to(NUMBER(Date representation wrapped))
7. id(NUMBER)

GetSchoolClassesByPerson

URL:

/api/v1/{format}/person/{id}/schoolclasses

Method:

GET

Parameters request:

null

Parameters response:

Array

Description:

1. school_day_start(NUMBER(Date representation wrapped))
2. school_day_end(NUMBER(Date representation wrapped))
3. school(OBJECT< School >)
4. pupils(ARRAY< Pupil >)
5. diaries(ARRAY< Diary >)
6. name(STRING)
7. id(NUMBER)

GetSchoolClassesOfCurrentPerson

URL:

/api/v1/{format}/person/current/schoolclasses

Method:

GET

Parameters request:

null

Parameters response:

Array

Description:

1. school_day_start(NUMBER(Date representation wrapped))
2. school_day_end(NUMBER(Date representation wrapped))
3. school(OBJECT< School >)
4. pupils(ARRAY< Pupil >)
5. diaries(ARRAY< Diary >)
6. name(STRING)
7. id(NUMBER)

GetSchoolsByPerson

URL:

/api/v1/{format}/person/{id}/schools

Method:

GET

Parameters request:

null

Parameters response:

Array

Description:

1. school_id(STRING)
2. services(ARRAY< ServiceTypeEnum >)
3. school_classes(ARRAY< SchoolClass >)
4. after_school_center_sections(ARRAY< AfterSchoolCenterSection >)
5. name(STRING)
6. id(NUMBER)

GetSchoolsOfCurrentPerson

URL:

/api/v1/{format}/person/current/schools

Method:

GET

Parameters request:

null

Parameters response:

Array

Description:

1. school_id(STRING)
2. services(ARRAY< ServiceTypeEnum >)
3. school_classes(ARRAY< SchoolClass >)
4. after_school_center_sections(ARRAY< AfterSchoolCenterSection >)
5. name(STRING)
6. id(NUMBER)

GetWorkRolesByPerson

URL:

/api/v1/{format}/person/{id}/workroles

Method:

GET

Parameters request:

null

Parameters response:

Array

Description:

1. name(STRING)
2. id(NUMBER)

GetWorkRolesOfCurrentPerson

URL:

/api/v1/{format}/person/current/workroles

Method:

GET

Parameters request:

null

Parameters response:

Array

Description:

1. name(STRING)
2. id(NUMBER)

SaveAll

URL:

/api/v1/{format}/persons/saveall

Method:

POST

Parameters request:

Array< Person >

Parameters response:

Array

Description:

1. personal_id(STRING)
2. first_name(STRING)
3. last_name(STRING)
4. addresses(KEY_ENUM_OBJECT_PAIR< AddressTypeEnum , Address >)
5. emails(KEY_ENUM_OBJECT_PAIR< CommunicationTypeEnum , Email >)
6. phones(KEY_ENUM_OBJECT_PAIR< CommunicationTypeEnum , Phone >)
7. id(NUMBER)

SaveAllAndReturnIds

URL:

/api/v1/{format}/persons/saveall

Method:

POST

Parameters request:

Url parameters: full

Array< Person >

Parameters response:

Array

Description: ARRAY<NUMBER>

Search

URL:

/api/v1/{format}/persons/search

Method:

POST

Parameters request:

Array< SearchCriteries$SearchCriteriaResult >

Parameters response:

Array

Description:

1. personal_id(STRING)
2. first_name(STRING)
3. last_name(STRING)
4. addresses(KEY_ENUM_OBJECT_PAIR< AddressTypeEnum , Address >)
5. emails(KEY_ENUM_OBJECT_PAIR< CommunicationTypeEnum , Email >)
6. phones(KEY_ENUM_OBJECT_PAIR< CommunicationTypeEnum , Phone >)
7. id(NUMBER)

SearchFirst

URL:

/api/v1/{format}/persons/search/first

Method:

POST

Parameters request:

Array< SearchCriteries$SearchCriteriaResult >

Parameters response:

Object

Description:

1. personal_id(STRING)
2. first_name(STRING)
3. last_name(STRING)
4. addresses(KEY_ENUM_OBJECT_PAIR< AddressTypeEnum , Address >)
5. emails(KEY_ENUM_OBJECT_PAIR< CommunicationTypeEnum , Email >)
6. phones(KEY_ENUM_OBJECT_PAIR< CommunicationTypeEnum , Phone >)
7. id(NUMBER)

Update

URL:

/api/v1/{format}/persons/{id}

Method:

PUT

Parameters request:

null

Parameters response:

Object

Description:

1. personal_id(STRING)
2. first_name(STRING)
3. last_name(STRING)
4. addresses(KEY_ENUM_OBJECT_PAIR< AddressTypeEnum , Address >)
5. emails(KEY_ENUM_OBJECT_PAIR< CommunicationTypeEnum , Email >)
6. phones(KEY_ENUM_OBJECT_PAIR< CommunicationTypeEnum , Phone >)
7. id(NUMBER)

Pupils

`(implementation of Pupil entity)`

Provides following method for API calls:

- *Create*
- *Delete*
- *DeleteByIds*
- *Get*
- *GetAll*
- *GetAllPupils*
- *GetByPersonalId*
- *GetFirstByPersonalId*
- *SaveAll*
- *SaveAllAndReturnIds*
- *Search*
- *SearchFirst*
- *Update*

Create

URL:

/api/v1/{format}/pupils

Method:

POST

Parameters request:

Object< Pupil >

Parameters response:

Object

Description:

1. person(OBJECT< Person >)
2. contact_person(OBJECT< Person >)
3. class_placement_from(NUMBER(Date representation wrapped))
4. class_placement_to(NUMBER(Date representation wrapped))
5. school_class(OBJECT< SchoolClass >)
6. school(OBJECT< School >)
7. academic_year(OBJECT< AcademicYear >)
8. guardians(ARRAY< Guardian >)
9. truancies(ARRAY< Truancy >)
10. after_school_center_section(OBJECT< AfterSchoolCenterSection >)
11. school_center_schema(ARRAY< AfterSchoolCenterSchema >)
12. id(NUMBER)

Delete

URL:

/api/v1/{format}/pupils/{id}

Method:

DELETE

Parameters request:

null

Parameters response:

Object

Description:

1. person(OBJECT< Person >)

2. contact_person(OBJECT< Person >)
3. class_placement_from(NUMBER(Date representation wrapped))
4. class_placement_to(NUMBER(Date representation wrapped))
5. school_class(OBJECT< SchoolClass >)
6. school(OBJECT< School >)
7. academic_year(OBJECT< AcademicYear >)
8. guardians(ARRAY< Guardian >)
9. truancies(ARRAY< Truancy >)
10. after_school_center_section(OBJECT< AfterSchoolCenterSection >)
11. school_center_schema(ARRAY< AfterSchoolCenterSchema >)
12. id(NUMBER)

DeleteByIds

URL:

/api/v1/{format}/pupils

Method:

DELETE

Parameters request:

Url parameters: ids

Parameters response:

Array

Description:

1. person(OBJECT< Person >)
2. contact_person(OBJECT< Person >)
3. class_placement_from(NUMBER(Date representation wrapped))
4. class_placement_to(NUMBER(Date representation wrapped))
5. school_class(OBJECT< SchoolClass >)
6. school(OBJECT< School >)
7. academic_year(OBJECT< AcademicYear >)
8. guardians(ARRAY< Guardian >)
9. truancies(ARRAY< Truancy >)

10. after_school_center_section(OBJECT< AfterSchoolCenterSection >)
11. school_center_schema(ARRAY< AfterSchoolCenterSchema >)
12. id(NUMBER)

Get

URL:

/api/v1/{format}/pupils/{id}

Method:

GET

Parameters request:

null

Parameters response:

Object

Description:

1. person(OBJECT< Person >)
2. contact_person(OBJECT< Person >)
3. class_placement_from(NUMBER(Date representation wrapped))
4. class_placement_to(NUMBER(Date representation wrapped))
5. school_class(OBJECT< SchoolClass >)
6. school(OBJECT< School >)
7. academic_year(OBJECT< AcademicYear >)
8. guardians(ARRAY< Guardian >)
9. truancies(ARRAY< Truancy >)
10. after_school_center_section(OBJECT< AfterSchoolCenterSection >)
11. school_center_schema(ARRAY< AfterSchoolCenterSchema >)
12. id(NUMBER)

GetAll

URL:

/api/v1/{format}/pupils

Method:

GET

Parameters request:

null

Parameters response:

Array

Description:

1. person(OBJECT< Person >)
2. contact_person(OBJECT< Person >)
3. class_placement_from(NUMBER(Date representation wrapped))
4. class_placement_to(NUMBER(Date representation wrapped))
5. school_class(OBJECT< SchoolClass >)
6. school(OBJECT< School >)
7. academic_year(OBJECT< AcademicYear >)
8. guardians(ARRAY< Guardian >)
9. truancies(ARRAY< Truancy >)
10. after_school_center_section(OBJECT< AfterSchoolCenterSection >)
11. school_center_schema(ARRAY< AfterSchoolCenterSchema >)
12. id(NUMBER)

GetAllPupils

URL:

/api/v1/{format}/pupils/all

Method:

GET

Parameters request:

null

Parameters response:

Array

Description:

1. person(OBJECT< Person >)
2. contact_person(OBJECT< Person >)
3. class_placement_from(NUMBER(Date representation wrapped))
4. class_placement_to(NUMBER(Date representation wrapped))
5. school_class(OBJECT< SchoolClass >)
6. school(OBJECT< School >)
7. academic_year(OBJECT< AcademicYear >)
8. guardians(ARRAY< Guardian >)
9. truancies(ARRAY< Truancy >)
10. after_school_center_section(OBJECT< AfterSchoolCenterSection >)
11. school_center_schema(ARRAY< AfterSchoolCenterSchema >)
12. id(NUMBER)

GetByPersonalId

URL:

/api/v1/{format}/pupils

Method:

GET

Parameters request:

Url parameters: personalId

Parameters response:

Array

Description:

1. person(OBJECT< Person >)
2. contact_person(OBJECT< Person >)
3. class_placement_from(NUMBER(Date representation wrapped))
4. class_placement_to(NUMBER(Date representation wrapped))
5. school_class(OBJECT< SchoolClass >)

6. school(OBJECT< School >)
7. academic_year(OBJECT< AcademicYear >)
8. guardians(ARRAY< Guardian >)
9. truancies(ARRAY< Truancy >)
10. after_school_center_section(OBJECT< AfterSchoolCenterSection >)
11. school_center_schema(ARRAY< AfterSchoolCenterSchema >)
12. id(NUMBER)

GetFirstByPersonalId

URL:

/api/v1/{format}/pupils

Method:

GET

Parameters request:

Url parameters: personalId, first

Parameters response:

Object

Description:

1. person(OBJECT< Person >)
2. contact_person(OBJECT< Person >)
3. class_placement_from(NUMBER(Date representation wrapped))
4. class_placement_to(NUMBER(Date representation wrapped))
5. school_class(OBJECT< SchoolClass >)
6. school(OBJECT< School >)
7. academic_year(OBJECT< AcademicYear >)
8. guardians(ARRAY< Guardian >)
9. truancies(ARRAY< Truancy >)
10. after_school_center_section(OBJECT< AfterSchoolCenterSection >)
11. school_center_schema(ARRAY< AfterSchoolCenterSchema >)
12. id(NUMBER)

SaveAll

URL:

/api/v1/{format}/pupils/saveall

Method:

POST

Parameters request:

Array< Pupil >

Parameters response:

Array

Description:

1. person(OBJECT< Person >)
2. contact_person(OBJECT< Person >)
3. class_placement_from(NUMBER(Date representation wrapped))
4. class_placement_to(NUMBER(Date representation wrapped))
5. school_class(OBJECT< SchoolClass >)
6. school(OBJECT< School >)
7. academic_year(OBJECT< AcademicYear >)
8. guardians(ARRAY< Guardian >)
9. truancies(ARRAY< Truancy >)
10. after_school_center_section(OBJECT< AfterSchoolCenterSection >)
11. school_center_schema(ARRAY< AfterSchoolCenterSchema >)
12. id(NUMBER)

SaveAllAndReturnIds

URL:

/api/v1/{format}/pupils/saveall

Method:

POST

Parameters request:

Url parameters: full

Array< Pupil >

Parameters response:

Array

Description: ARRAY<NUMBER>

Search

URL:

/api/v1/{format}/pupils/search

Method:

POST

Parameters request:

Array< SearchCriteries$SearchCriteriaResult >

Parameters response:

Array

Description:

1. person(OBJECT< Person >)
2. contact_person(OBJECT< Person >)
3. class_placement_from(NUMBER(Date representation wrapped))
4. class_placement_to(NUMBER(Date representation wrapped))
5. school_class(OBJECT< SchoolClass >)
6. school(OBJECT< School >)
7. academic_year(OBJECT< AcademicYear >)
8. guardians(ARRAY< Guardian >)
9. truancies(ARRAY< Truancy >)
10. after_school_center_section(OBJECT< AfterSchoolCenterSection >)
11. school_center_schema(ARRAY< AfterSchoolCenterSchema >)
12. id(NUMBER)

SearchFirst

URL:

/api/v1/{format}/pupils/search/first

Method:

POST

Parameters request:

Array< SearchCriteries$SearchCriteriaResult >

Parameters response:

Object

Description:

1. person(OBJECT< Person >)
2. contact_person(OBJECT< Person >)
3. class_placement_from(NUMBER(Date representation wrapped))
4. class_placement_to(NUMBER(Date representation wrapped))
5. school_class(OBJECT< SchoolClass >)
6. school(OBJECT< School >)
7. academic_year(OBJECT< AcademicYear >)
8. guardians(ARRAY< Guardian >)
9. truancies(ARRAY< Truancy >)
10. after_school_center_section(OBJECT< AfterSchoolCenterSection >)
11. school_center_schema(ARRAY< AfterSchoolCenterSchema >)
12. id(NUMBER)

Update

URL:

/api/v1/{format}/pupils/{id}

Method:

PUT

Parameters request:

Object< Pupil >

Parameters response:

Object

Description:

1. person(OBJECT< Person >)
2. contact_person(OBJECT< Person >)
3. class_placement_from(NUMBER(Date representation wrapped))
4. class_placement_to(NUMBER(Date representation wrapped))
5. school_class(OBJECT< SchoolClass >)
6. school(OBJECT< School >)
7. academic_year(OBJECT< AcademicYear >)
8. guardians(ARRAY< Guardian >)
9. truancies(ARRAY< Truancy >)
10. after_school_center_section(OBJECT< AfterSchoolCenterSection >)
11. school_center_schema(ARRAY< AfterSchoolCenterSchema >)
12. id(NUMBER)

SchoolClasses

`(implementation of SchoolClass entity)`

Provides following method for API calls:

- *Create*
- *Delete*
- *DeleteByIds*
- *Get*
- *GetAll*
- *GetByName*
- *GetFirstByName*
- *SaveAll*
- *SaveAllAndReturnIds*
- *Search*
- *SearchFirst*
- *Update*

Create

URL:

/api/v1/{format}/schoolclasses

Method:

POST

Parameters request:

Object< SchoolClass >

Parameters response:

Object

Description:

1. school_day_start(NUMBER(Date representation wrapped))
2. school_day_end(NUMBER(Date representation wrapped))
3. school(OBJECT< School >)
4. pupils(ARRAY< Pupil >)
5. diaries(ARRAY< Diary >)
6. name(STRING)
7. id(NUMBER)

Delete

URL:

/api/v1/{format}/schoolclasses/{id}

Method:

DELETE

Parameters request:

null

Parameters response:

Object

Description:

1. school_day_start(NUMBER(Date representation wrapped))
2. school_day_end(NUMBER(Date representation wrapped))
3. school(OBJECT< School >)
4. pupils(ARRAY< Pupil >)
5. diaries(ARRAY< Diary >)
6. name(STRING)
7. id(NUMBER)

DeleteByIds

URL:

/api/v1/{format}/schoolclasses

Method:

DELETE

Parameters request:

Url parameters: ids

Parameters response:

Array

Description:

1. school_day_start(NUMBER(Date representation wrapped))
2. school_day_end(NUMBER(Date representation wrapped))
3. school(OBJECT< School >)
4. pupils(ARRAY< Pupil >)
5. diaries(ARRAY< Diary >)
6. name(STRING)
7. id(NUMBER)

Get

URL:

/api/v1/{format}/schoolclasses/{id}

Method:

GET

Parameters request:

null

Parameters response:

Object

Description:

1. school_day_start(NUMBER(Date representation wrapped))
2. school_day_end(NUMBER(Date representation wrapped))
3. school(OBJECT< School >)
4. pupils(ARRAY< Pupil >)
5. diaries(ARRAY< Diary >)
6. name(STRING)
7. id(NUMBER)

GetAll

URL:

/api/v1/{format}/schoolclasses

Method:

GET

Parameters request:

null

Parameters response:

Array

Description:

1. school_day_start(NUMBER(Date representation wrapped))
2. school_day_end(NUMBER(Date representation wrapped))
3. school(OBJECT< School >)
4. pupils(ARRAY< Pupil >)
5. diaries(ARRAY< Diary >)
6. name(STRING)
7. id(NUMBER)

GetByName

URL:

/api/v1/{format}/schoolclasses

Method:

GET

Parameters request:

Url parameters: name

Parameters response:

Array

Description:

1. school_day_start(NUMBER(Date representation wrapped))
2. school_day_end(NUMBER(Date representation wrapped))
3. school(OBJECT< School >)
4. pupils(ARRAY< Pupil >)
5. diaries(ARRAY< Diary >)
6. name(STRING)
7. id(NUMBER)

GetFirstByName

URL:

/api/v1/{format}/schoolclasses

Method:

GET

Parameters request:

Url parameters: name, first

Parameters response:

Object

Description:

1. school_day_start(NUMBER(Date representation wrapped))
2. school_day_end(NUMBER(Date representation wrapped))
3. school(OBJECT< School >)
4. pupils(ARRAY< Pupil >)
5. diaries(ARRAY< Diary >)
6. name(STRING)
7. id(NUMBER)

SaveAll

URL:

/api/v1/{format}/schoolclasses/saveall

Method:

POST

Parameters request:

Array< SchoolClass >

Parameters response:

Array

Description:

1. school_day_start(NUMBER(Date representation wrapped))
2. school_day_end(NUMBER(Date representation wrapped))
3. school(OBJECT< School >)
4. pupils(ARRAY< Pupil >)
5. diaries(ARRAY< Diary >)
6. name(STRING)
7. id(NUMBER)

SaveAllAndReturnIds

URL:

/api/v1/{format}/schoolclasses/saveall

Method:

POST

Parameters request:

Url parameters: full

Array< SchoolClass >

Parameters response:

Array

Description: ARRAY<NUMBER>

Search

URL:

/api/v1/{format}/schoolclasses/search

Method:

POST

Parameters request:

Array< SearchCriteries$SearchCriteriaResult >

Parameters response:

Array

Description:

1. school_day_start(NUMBER(Date representation wrapped))
2. school_day_end(NUMBER(Date representation wrapped))
3. school(OBJECT< School >)
4. pupils(ARRAY< Pupil >)
5. diaries(ARRAY< Diary >)
6. name(STRING)
7. id(NUMBER)

SearchFirst

URL:

/api/v1/{format}/schoolclasses/search/first

Method:

POST

Parameters request:

Array< SearchCriteries$SearchCriteriaResult >

Parameters response:

Object

Description:

1. school_day_start(NUMBER(Date representation wrapped))
2. school_day_end(NUMBER(Date representation wrapped))
3. school(OBJECT< School >)
4. pupils(ARRAY< Pupil >)
5. diaries(ARRAY< Diary >)
6. name(STRING)

7. id(NUMBER)

Update

URL:

/api/v1/{format}/schoolclasses/{id}

Method:

PUT

Parameters request:

Object< SchoolClass >

Parameters response:

Object

Description:

1. school_day_start(NUMBER(Date representation wrapped))
2. school_day_end(NUMBER(Date representation wrapped))
3. school(OBJECT< School >)
4. pupils(ARRAY< Pupil >)
5. diaries(ARRAY< Diary >)
6. name(STRING)
7. id(NUMBER)

Schools

`(implementation of School entity)`

Provides following method for API calls:

- *Create*
- *Delete*
- *DeleteByIds*
- *Get*
- *GetAll*
- *GetByName*
- *GetFirstByName*
- *GetPersonsBySchool*

- *SaveAll*
- *SaveAllAndReturnIds*
- *Search*
- *SearchFirst*
- *Update*

Create

URL:

/api/v1/{format}/school

Method:

POST

Parameters request:

Object< School >

Parameters response:

Object

Description:

1. school_id(STRING)
2. services(ARRAY< ServiceTypeEnum >)
3. school_classes(ARRAY< SchoolClass >)
4. after_school_center_sections(ARRAY< AfterSchoolCenterSection >)
5. name(STRING)
6. id(NUMBER)

Delete

URL:

/api/v1/{format}/school/{id}

Method:

DELETE

Parameters request:

null

Parameters response:

Object

Description:

1. school_id(STRING)
2. services(ARRAY< ServiceTypeEnum >)
3. school_classes(ARRAY< SchoolClass >)
4. after_school_center_sections(ARRAY< AfterSchoolCenterSection >)
5. name(STRING)
6. id(NUMBER)

DeleteByIds

URL:

/api/v1/{format}/schools

Method:

DELETE

Parameters request:

Url parameters: ids

Parameters response:

Array

Description:

1. school_id(STRING)
2. services(ARRAY< ServiceTypeEnum >)
3. school_classes(ARRAY< SchoolClass >)
4. after_school_center_sections(ARRAY< AfterSchoolCenterSection >)
5. name(STRING)
6. id(NUMBER)

Get

URL:

/api/v1/{format}/school/{id}

Method:

GET

Parameters request:

null

Parameters response:

Object

Description:

1. school_id(STRING)
2. services(ARRAY< ServiceTypeEnum >)
3. school_classes(ARRAY< SchoolClass >)
4. after_school_center_sections(ARRAY< AfterSchoolCenterSection >)
5. name(STRING)
6. id(NUMBER)

GetAll

URL:

/api/v1/{format}/schools

Method:

GET

Parameters request:

null

Parameters response:

Array

Description:

1. school_id(STRING)
2. services(ARRAY< ServiceTypeEnum >)
3. school_classes(ARRAY< SchoolClass >)
4. after_school_center_sections(ARRAY< AfterSchoolCenterSection >)
5. name(STRING)
6. id(NUMBER)

GetByName

URL:

/api/v1/{format}/schools

Method:

GET

Parameters request:

Url parameters: name

Parameters response:

Array

Description:

1. school_id(STRING)
2. services(ARRAY< ServiceTypeEnum >)
3. school_classes(ARRAY< SchoolClass >)
4. after_school_center_sections(ARRAY< AfterSchoolCenterSection >)
5. name(STRING)
6. id(NUMBER)

GetFirstByName

URL:

/api/v1/{format}/school

Method:

GET

Parameters request:

Url parameters: name, first

Parameters response:

Object

Description:

1. school_id(STRING)
2. services(ARRAY< ServiceTypeEnum >)
3. school_classes(ARRAY< SchoolClass >)
4. after_school_center_sections(ARRAY< AfterSchoolCenterSection >)
5. name(STRING)
6. id(NUMBER)

GetPersonsBySchool

URL:

/api/v1/{format}/school/{id}/persons

Method:

GET

Parameters request:

null

Parameters response:

Array

Description:

1. personal_id(STRING)
2. first_name(STRING)
3. last_name(STRING)
4. addresses(KEY_ENUM_OBJECT_PAIR< AddressTypeEnum , Address >)

5. emails(KEY_ENUM_OBJECT_PAIR< CommunicationTypeEnum , Email >)
6. phones(KEY_ENUM_OBJECT_PAIR< CommunicationTypeEnum , Phone >)
7. id(NUMBER)

SaveAll

URL:

/api/v1/{format}/schools/saveall

Method:

POST

Parameters request:

Array< School >

Parameters response:

Array

Description:

1. school_id(STRING)
2. services(ARRAY< ServiceTypeEnum >)
3. school_classes(ARRAY< SchoolClass >)
4. after_school_center_sections(ARRAY< AfterSchoolCenterSection >)
5. name(STRING)
6. id(NUMBER)

SaveAllAndReturnIds

URL:

/api/v1/{format}/schools/saveall

Method:

POST

Parameters request:

Url parameters: full

Array< School >

Parameters response:

Array

Description: ARRAY<NUMBER>

Search

URL:

/api/v1/{format}/schools/search

Method:

POST

Parameters request:

Array< SearchCriteries$SearchCriteriaResult >

Parameters response:

Array

Description:

1. school_id(STRING)
2. services(ARRAY< ServiceTypeEnum >)
3. school_classes(ARRAY< SchoolClass >)
4. after_school_center_sections(ARRAY< AfterSchoolCenterSection >)
5. name(STRING)
6. id(NUMBER)

SearchFirst

URL:

/api/v1/{format}/school/search

Method:

POST

Parameters request:

Array< SearchCriteries$SearchCriteriaResult >

Parameters response:

Object

Description:

1. school_id(STRING)
2. services(ARRAY< ServiceTypeEnum >)
3. school_classes(ARRAY< SchoolClass >)
4. after_school_center_sections(ARRAY< AfterSchoolCenterSection >)
5. name(STRING)
6. id(NUMBER)

Update

URL:

/api/v1/{format}/school/{id}

Method:

PUT

Parameters request:

null

Parameters response:

Object

Description:

1. school_id(STRING)
2. services(ARRAY< ServiceTypeEnum >)
3. school_classes(ARRAY< SchoolClass >)
4. after_school_center_sections(ARRAY< AfterSchoolCenterSection >)

5. name(STRING)
6. id(NUMBER)

Users

`(implementation of User entity)`

Provides following method for API calls:

- *Create*
- *Delete*
- *DeleteByIds*
- *Get*
- *GetAll*
- *GetByPersonalId*
- *GetCurrentUser*
- *GetFirstByPersonalId*
- *GetLoggedInUser*
- *SaveAll*
- *SaveAllAndReturnIds*
- *Search*
- *SearchFirst*
- *Update*

Create

URL:

/api/v1/{format}/users

Method:

POST

Parameters request:

Object< User >

Parameters response:

Object

Description:

1. person(OBJECT< Person >)
2. roles(ARRAY< Role >)
3. saml2_id(STRING)
4. name(STRING)
5. id(NUMBER)

Delete

URL:

/api/v1/{format}/users/{id}

Method:

DELETE

Parameters request:

null

Parameters response:

Object

Description:

1. person(OBJECT< Person >)
2. roles(ARRAY< Role >)
3. saml2_id(STRING)
4. name(STRING)
5. id(NUMBER)

DeleteByIds

URL:

/api/v1/{format}/users

Method:

DELETE

Parameters request:

Url parameters: ids

Parameters response:

Array

Description:

1. person(OBJECT< Person >)
2. roles(ARRAY< Role >)
3. saml2_id(STRING)
4. name(STRING)
5. id(NUMBER)

Get

URL:

/api/v1/{format}/users/{id}

Method:

GET

Parameters request:

null

Parameters response:

Object

Description:

1. person(OBJECT< Person >)
2. roles(ARRAY< Role >)
3. saml2_id(STRING)
4. name(STRING)
5. id(NUMBER)

GetAll

URL:

/api/v1/{format}/users

Method:

GET

Parameters request:

null

Parameters response:

Array

Description:

1. person(OBJECT< Person >)
2. roles(ARRAY< Role >)
3. saml2_id(STRING)
4. name(STRING)
5. id(NUMBER)

GetByPersonalId

URL:

/api/v1/{format}/users

Method:

GET

Parameters request:

Url parameters: personalId

Parameters response:

Array

Description:

1. person(OBJECT< Person >)
2. roles(ARRAY< Role >)
3. saml2_id(STRING)
4. name(STRING)
5. id(NUMBER)

GetCurrentUser

URL:

/api/v1/{format}/users/current

Method:

GET

Parameters request:

null

Parameters response:

Object

Description:

1. person(OBJECT< Person >)
2. roles(ARRAY< Role >)
3. saml2_id(STRING)
4. name(STRING)
5. id(NUMBER)

GetFirstByPersonalId

URL:

/api/v1/{format}/users

Method:

GET

Parameters request:

Url parameters: personalId, first

Parameters response:

Object

Description:

1. person(OBJECT< Person >)
2. roles(ARRAY< Role >)
3. saml2_id(STRING)
4. name(STRING)
5. id(NUMBER)

GetLoggedInUser

URL:

/api/v1/{format}/users/loggedin

Method:

GET

Parameters request:

null

Parameters response:

Object

Description:

Note: This method return only 3 properties: id, person, roles.

SaveAll

URL:

/api/v1/{format}/users/saveall

Method:

POST

Parameters request:

Array< User >

Parameters response:

Array

Description:

1. person(OBJECT< Person >)
2. roles(ARRAY< Role >)
3. saml2_id(STRING)
4. name(STRING)
5. id(NUMBER)

SaveAllAndReturnIds

URL:

/api/v1/{format}/users/saveall

Method:

POST

Parameters request:

Url parameters: full

Array< User >

Parameters response:

Array

Description: ARRAY<NUMBER>

Search

URL:

/api/v1/{format}/users/search

Method:

POST

Parameters request:

Array< SearchCriteries$SearchCriteriaResult >

Parameters response:

Array

Description:

1. person(OBJECT< Person >)
2. roles(ARRAY< Role >)
3. saml2_id(STRING)
4. name(STRING)
5. id(NUMBER)

SearchFirst

URL:

/api/v1/{format}/users/search/first

Method:

POST

Parameters request:

Array< SearchCriteries$SearchCriteriaResult >

Parameters response:

Object

Description:

1. person(OBJECT< Person >)
2. roles(ARRAY< Role >)
3. saml2_id(STRING)
4. name(STRING)
5. id(NUMBER)

Update

URL:

/api/v1/{format}/users/{id}

Method:

PUT

Parameters request:

Object< User >

Parameters response:

Object

Description:

1. person(OBJECT< Person >)
2. roles(ARRAY< Role >)
3. saml2_id(STRING)
4. name(STRING)
5. id(NUMBER)

APPLICATION

ApplicationFormQuestionGroups

`(implementation of ApplicationFormQuestionGroup entity)`

Provides following method for API calls:

- *Create*

- *Delete*
- *DeleteByIds*
- *Get*
- *GetAll*
- *SaveAll*
- *SaveAllAndReturnIds*
- *Search*
- *SearchFirst*
- *Update*

Create

URL:

/api/v1/{format}/applicationformquestiongroups

Method:

POST

Parameters request:

Object< ApplicationFormQuestionGroup >

Parameters response:

Object

Description:

1. text(STRING)
2. questions(ARRAY< ApplicationFormQuestion >)
3. question_type(STRING)
4. step(OBJECT< ApplicationFormStep >)
5. sort_order(NUMBER)
6. name(STRING)
7. id(NUMBER)

Delete

URL:

/api/v1/{format}/applicationformquestiongroups/{id}

Method:

DELETE

Parameters request:

null

Parameters response:

Object

Description:

1. text(STRING)
2. questions(ARRAY< ApplicationFormQuestion >)
3. question_type(STRING)
4. step(OBJECT< ApplicationFormStep >)
5. sort_order(NUMBER)
6. name(STRING)
7. id(NUMBER)

DeleteByIds

URL:

/api/v1/{format}/applicationformquestiongroups

Method:

DELETE

Parameters request:

Url parameters: ids

Parameters response:

Array

Description:

1. text(STRING)
2. questions(ARRAY< ApplicationFormQuestion >)
3. question_type(STRING)
4. step(OBJECT< ApplicationFormStep >)
5. sort_order(NUMBER)
6. name(STRING)
7. id(NUMBER)

Get

URL:

/api/v1/{format}/applicationformquestiongroups/{id}

Method:

GET

Parameters request:

null

Parameters response:

Object

Description:

1. text(STRING)
2. questions(ARRAY< ApplicationFormQuestion >)
3. question_type(STRING)
4. step(OBJECT< ApplicationFormStep >)
5. sort_order(NUMBER)
6. name(STRING)
7. id(NUMBER)

GetAll

URL:

/api/v1/{format}/applicationformquestiongroups

Method:

GET

Parameters request:

null

Parameters response:

Array

Description:

1. text(STRING)
2. questions(ARRAY< ApplicationFormQuestion >)
3. question_type(STRING)
4. step(OBJECT< ApplicationFormStep >)
5. sort_order(NUMBER)
6. name(STRING)
7. id(NUMBER)

SaveAll

URL:

/api/v1/{format}/applicationformquestiongroups/saveall

Method:

POST

Parameters request:

Array< ApplicationFormQuestionGroup >

Parameters response:

Array

Description:

1. text(STRING)
2. questions(ARRAY< ApplicationFormQuestion >)
3. question_type(STRING)
4. step(OBJECT< ApplicationFormStep >)
5. sort_order(NUMBER)
6. name(STRING)
7. id(NUMBER)

SaveAllAndReturnIds

URL:

/api/v1/{format}/applicationformquestiongroups/saveall

Method:

POST

Parameters request:

Url parameters: full

Array< ApplicationFormQuestionGroup >

Parameters response:

Array

Description: ARRAY<NUMBER>

Search

URL:

/api/v1/{format}/applicationformquestiongroups/search

Method:

POST

Parameters request:

Array< SearchCriteries$SearchCriteriaResult >

Parameters response:

Array

Description:

1. text(STRING)
2. questions(ARRAY< ApplicationFormQuestion >)
3. question_type(STRING)
4. step(OBJECT< ApplicationFormStep >)
5. sort_order(NUMBER)
6. name(STRING)
7. id(NUMBER)

SearchFirst

URL:

/api/v1/{format}/applicationformquestiongroups/search/first

Method:

POST

Parameters request:

Array< SearchCriteries$SearchCriteriaResult >

Parameters response:

Object

Description:

1. text(STRING)
2. questions(ARRAY< ApplicationFormQuestion >)
3. question_type(STRING)
4. step(OBJECT< ApplicationFormStep >)
5. sort_order(NUMBER)
6. name(STRING)

7. id(NUMBER)

Update

URL:

/api/v1/{format}/applicationformquestiongroups/{id}

Method:

PUT

Parameters request:

Object< ApplicationFormQuestionGroup >

Parameters response:

Object

Description:

1. text(STRING)
2. questions(ARRAY< ApplicationFormQuestion >)
3. question_type(STRING)
4. step(OBJECT< ApplicationFormStep >)
5. sort_order(NUMBER)
6. name(STRING)
7. id(NUMBER)

ApplicationFormQuestions

`(implementation of ApplicationFormQuestion entity)`

Provides following method for API calls:

- *Create*
- *Delete*
- *DeleteByIds*
- *Get*
- *GetAll*
- *SaveAll*
- *SaveAllAndReturnIds*
- *Search*

- *SearchFirst*
- *Update*

Create

URL:

/api/v1/{format}/applicationformquestions

Method:

POST

Parameters request:

Object< ApplicationFormQuestion >

Parameters response:

Object

Description:

1. text(STRING)
2. value(STRING)
3. multi_values(BOOLEAN)
4. values(ARRAY<STRING>)
5. multi_variants(BOOLEAN)
6. variants(ARRAY<STRING>)
7. question_type(STRING)
8. question_group(OBJECT< ApplicationFormQuestionGroup >)
9. sort_order(NUMBER)
10. name(STRING)
11. id(NUMBER)

Delete

URL:

/api/v1/{format}/applicationformquestions/{id}

Method:

DELETE

Parameters request:

null

Parameters response:

Object

Description:

1. text(STRING)
2. value(STRING)
3. multi_values(BOOLEAN)
4. values(ARRAY<STRING>)
5. multi_variants(BOOLEAN)
6. variants(ARRAY<STRING>)
7. question_type(STRING)
8. question_group(OBJECT< ApplicationFormQuestionGroup >)
9. sort_order(NUMBER)
10. name(STRING)
11. id(NUMBER)

DeleteByIds

URL:

/api/v1/{format}/applicationformquestions

Method:

DELETE

Parameters request:

Url parameters: ids

Parameters response:

Array

Description:

1. text(STRING)
2. value(STRING)
3. multi_values(BOOLEAN)
4. values(ARRAY<STRING>)
5. multi_variants(BOOLEAN)
6. variants(ARRAY<STRING>)
7. question_type(STRING)
8. question_group(OBJECT< ApplicationFormQuestionGroup >)
9. sort_order(NUMBER)
10. name(STRING)
11. id(NUMBER)

Get

URL:

/api/v1/{format}/applicationformquestions/{id}

Method:

GET

Parameters request:

null

Parameters response:

Object

Description:

1. text(STRING)
2. value(STRING)
3. multi_values(BOOLEAN)
4. values(ARRAY<STRING>)
5. multi_variants(BOOLEAN)
6. variants(ARRAY<STRING>)

7. question_type(STRING)
8. question_group(OBJECT< ApplicationFormQuestionGroup >)
9. sort_order(NUMBER)
10. name(STRING)
11. id(NUMBER)

GetAll

URL:

/api/v1/{format}/applicationformquestions

Method:

GET

Parameters request:

null

Parameters response:

Array

Description:

1. text(STRING)
2. value(STRING)
3. multi_values(BOOLEAN)
4. values(ARRAY<STRING>)
5. multi_variants(BOOLEAN)
6. variants(ARRAY<STRING>)
7. question_type(STRING)
8. question_group(OBJECT< ApplicationFormQuestionGroup >)
9. sort_order(NUMBER)
10. name(STRING)
11. id(NUMBER)

SaveAll

URL:

/api/v1/{format}/applicationformquestions/saveall

Method:

POST

Parameters request:

Array< ApplicationFormQuestion >

Parameters response:

Array

Description:

1. text(STRING)
2. value(STRING)
3. multi_values(BOOLEAN)
4. values(ARRAY<STRING>)
5. multi_variants(BOOLEAN)
6. variants(ARRAY<STRING>)
7. question_type(STRING)
8. question_group(OBJECT< ApplicationFormQuestionGroup >)
9. sort_order(NUMBER)
10. name(STRING)
11. id(NUMBER)

SaveAllAndReturnIds

URL:

/api/v1/{format}/applicationformquestions/saveall

Method:

POST

Parameters request:

Url parameters: full

Array< ApplicationFormQuestion >

Parameters response:

Array

Description: ARRAY<NUMBER>

Search

URL:

/api/v1/{format}/applicationformquestions/search

Method:

POST

Parameters request:

Array< SearchCriteries$SearchCriteriaResult >

Parameters response:

Array

Description:

1. text(STRING)
2. value(STRING)
3. multi_values(BOOLEAN)
4. values(ARRAY<STRING>)
5. multi_variants(BOOLEAN)
6. variants(ARRAY<STRING>)
7. question_type(STRING)
8. question_group(OBJECT< ApplicationFormQuestionGroup >)
9. sort_order(NUMBER)
10. name(STRING)
11. id(NUMBER)

SearchFirst

URL:

/api/v1/{format}/applicationformquestions/search/first

Method:

POST

Parameters request:

Array< SearchCriteries$SearchCriteriaResult >

Parameters response:

Object

Description:

1. text(STRING)
2. value(STRING)
3. multi_values(BOOLEAN)
4. values(ARRAY<STRING>)
5. multi_variants(BOOLEAN)
6. variants(ARRAY<STRING>)
7. question_type(STRING)
8. question_group(OBJECT< ApplicationFormQuestionGroup >)
9. sort_order(NUMBER)
10. name(STRING)
11. id(NUMBER)

Update

URL:

/api/v1/{format}/applicationformquestions/{id}

Method:

PUT

Parameters request:

Object< ApplicationFormQuestion >

Parameters response:

Object

Description:

1. text(STRING)
2. value(STRING)
3. multi_values(BOOLEAN)
4. values(ARRAY<STRING>)
5. multi_variants(BOOLEAN)
6. variants(ARRAY<STRING>)
7. question_type(STRING)
8. question_group(OBJECT< ApplicationFormQuestionGroup >)
9. sort_order(NUMBER)
10. name(STRING)
11. id(NUMBER)

ApplicationForms

`(implementation of ApplicationForm entity)`

Provides following method for API calls:

- *Create*
- *Delete*
- *DeleteByIds*
- *Get*
- *GetAll*
- *SaveAll*
- *SaveAllAndReturnIds*
- *Search*
- *SearchFirst*
- *Update*

Create

URL:

/api/v1/{format}/applicationforms

Method:

POST

Parameters request:

Object< ApplicationForm >

Parameters response:

Object

Description:

1. version(NUMBER)
2. steps(ARRAY< ApplicationFormStep >)
3. applications(ARRAY< Application >)
4. name(STRING)
5. id(NUMBER)

Delete

URL:

/api/v1/{format}/applicationforms/{id}

Method:

DELETE

Parameters request:

null

Parameters response:

Object

Description:

1. version(NUMBER)
2. steps(ARRAY< ApplicationFormStep >)
3. applications(ARRAY< Application >)
4. name(STRING)
5. id(NUMBER)

DeleteByIds

URL:

/api/v1/{format}/applicationforms

Method:

DELETE

Parameters request:

Url parameters: ids

Parameters response:

Array

Description:

1. version(NUMBER)
2. steps(ARRAY< ApplicationFormStep >)
3. applications(ARRAY< Application >)
4. name(STRING)
5. id(NUMBER)

Get

URL:

/api/v1/{format}/applicationforms/{id}

Method:

GET

Parameters request:

null

Parameters response:

Object

Description:

1. version(NUMBER)
2. steps(ARRAY< ApplicationFormStep >)
3. applications(ARRAY< Application >)
4. name(STRING)
5. id(NUMBER)

GetAll

URL:

/api/v1/{format}/applicationforms

Method:

GET

Parameters request:

null

Parameters response:

Array

Description:

1. version(NUMBER)
2. steps(ARRAY< ApplicationFormStep >)
3. applications(ARRAY< Application >)
4. name(STRING)
5. id(NUMBER)

SaveAll

URL:

/api/v1/{format}/applicationforms/saveall

Method:

POST

Parameters request:

Array< ApplicationForm >

Parameters response:

Array

Description:

1. version(NUMBER)
2. steps(ARRAY< ApplicationFormStep >)
3. applications(ARRAY< Application >)
4. name(STRING)
5. id(NUMBER)

SaveAllAndReturnIds

URL:

/api/v1/{format}/applicationforms/saveall

Method:

POST

Parameters request:

Url parameters: full

Array< ApplicationForm >

Parameters response:

Array

Description: ARRAY<NUMBER>

Search

URL:

/api/v1/{format}/applicationforms/search

Method:

POST

Parameters request:

Array< SearchCriteries$SearchCriteriaResult >

Parameters response:

Array

Description:

1. version(NUMBER)
2. steps(ARRAY< ApplicationFormStep >)
3. applications(ARRAY< Application >)
4. name(STRING)
5. id(NUMBER)

SearchFirst

URL:

/api/v1/{format}/applicationforms/search/first

Method:

POST

Parameters request:

Array< SearchCriteries$SearchCriteriaResult >

Parameters response:

Object

Description:

1. version(NUMBER)
2. steps(ARRAY< ApplicationFormStep >)
3. applications(ARRAY< Application >)
4. name(STRING)
5. id(NUMBER)

Update

URL:

/api/v1/{format}/applicationforms/{id}

Method:

PUT

Parameters request:

Object< ApplicationForm >

Parameters response:

Object

Description:

1. version(NUMBER)
2. steps(ARRAY< ApplicationFormStep >)
3. applications(ARRAY< Application >)
4. name(STRING)
5. id(NUMBER)

ApplicationFormSteps

`(implementation of ApplicationFormStep entity)`

Provides following method for API calls:

- *Create*
- *Delete*
- *DeleteByIds*

- *Get*
- *GetAll*
- *SaveAll*
- *SaveAllAndReturnIds*
- *Search*
- *SearchFirst*
- *Update*

Create

URL:

/api/v1/{format}/applicationformsteps

Method:

POST

Parameters request:

Object< ApplicationFormStep >

Parameters response:

Object

Description:

1. text(STRING)
2. question_groups(ARRAY< ApplicationFormQuestionGroup >)
3. application_form(OBJECT< ApplicationForm >)
4. sort_order(NUMBER)
5. name(STRING)
6. id(NUMBER)

Delete

URL:

/api/v1/{format}/applicationformsteps/{id}

Method:

DELETE

Parameters request:

null

Parameters response:

Object

Description:

1. text(STRING)
2. question_groups(ARRAY< ApplicationFormQuestionGroup >)
3. application_form(OBJECT< ApplicationForm >)
4. sort_order(NUMBER)
5. name(STRING)
6. id(NUMBER)

DeleteByIds

URL:

/api/v1/{format}/applicationformsteps

Method:

DELETE

Parameters request:

Url parameters: ids

Parameters response:

Array

Description:

1. text(STRING)
2. question_groups(ARRAY< ApplicationFormQuestionGroup >)
3. application_form(OBJECT< ApplicationForm >)
4. sort_order(NUMBER)

5. name(STRING)
6. id(NUMBER)

Get

URL:

/api/v1/{format}/applicationformsteps/{id}

Method:

GET

Parameters request:

null

Parameters response:

Object

Description:

1. text(STRING)
2. question_groups(ARRAY< ApplicationFormQuestionGroup >)
3. application_form(OBJECT< ApplicationForm >)
4. sort_order(NUMBER)
5. name(STRING)
6. id(NUMBER)

GetAll

URL:

/api/v1/{format}/applicationformsteps

Method:

GET

Parameters request:

null

Parameters response:

Array

Description:

1. text(STRING)
2. question_groups(ARRAY< ApplicationFormQuestionGroup >)
3. application_form(OBJECT< ApplicationForm >)
4. sort_order(NUMBER)
5. name(STRING)
6. id(NUMBER)

SaveAll

URL:

/api/v1/{format}/applicationformsteps/saveall

Method:

POST

Parameters request:

Array< ApplicationFormStep >

Parameters response:

Array

Description:

1. text(STRING)
2. question_groups(ARRAY< ApplicationFormQuestionGroup >)
3. application_form(OBJECT< ApplicationForm >)
4. sort_order(NUMBER)
5. name(STRING)
6. id(NUMBER)

SaveAllAndReturnIds

URL:

/api/v1/{format}/applicationformsteps/saveall

Method:

POST

Parameters request:

Url parameters: full

Array< ApplicationFormStep >

Parameters response:

Array

Description: ARRAY<NUMBER>

Search

URL:

/api/v1/{format}/applicationformsteps/search

Method:

POST

Parameters request:

Array< SearchCriteries$SearchCriteriaResult >

Parameters response:

Array

Description:

1. text(STRING)
2. question_groups(ARRAY< ApplicationFormQuestionGroup >)
3. application_form(OBJECT< ApplicationForm >)
4. sort_order(NUMBER)
5. name(STRING)
6. id(NUMBER)

SearchFirst

URL:

/api/v1/{format}/applicationformsteps/search/first

Method:

POST

Parameters request:

Array< SearchCriteries$SearchCriteriaResult >

Parameters response:

Object

Description:

1. text(STRING)
2. question_groups(ARRAY< ApplicationFormQuestionGroup >)
3. application_form(OBJECT< ApplicationForm >)
4. sort_order(NUMBER)
5. name(STRING)
6. id(NUMBER)

Update

URL:

/api/v1/{format}/applicationformsteps/{id}

Method:

PUT

Parameters request:

Object< ApplicationFormStep >

Parameters response:

Object

Description:

1. text(STRING)
2. question_groups(ARRAY< ApplicationFormQuestionGroup >)
3. application_form(OBJECT< ApplicationForm >)
4. sort_order(NUMBER)
5. name(STRING)
6. id(NUMBER)

Applications

`(implementation of Application entity)`

Provides following method for API calls:

- *Create*
- *Delete*
- *DeleteByIds*
- *Get*
- *GetAll*
- *SaveAll*
- *SaveAllAndReturnIds*
- *Search*
- *SearchFirst*
- *Update*

Create

URL:

/api/v1/{format}/applications

Method:

POST

Parameters request:

Object< Application >

Parameters response:

Object

Description:

1. application_form(OBJECT< ApplicationForm >)
2. submitted_user(OBJECT< User >)
3. regarding_user(OBJECT< User >)
4. registration_number(NUMBER)
5. decision(OBJECT< Decision >)
6. handled_user(OBJECT< Person >)
7. create_date(NUMBER(Date representation wrapped))
8. update_date(NUMBER(Date representation wrapped))
9. id(NUMBER)

Delete

URL:

/api/v1/{format}/applications/{id}

Method:

DELETE

Parameters request:

null

Parameters response:

Object

Description:

1. application_form(OBJECT< ApplicationForm >)
2. submitted_user(OBJECT< User >)
3. regarding_user(OBJECT< User >)
4. registration_number(NUMBER)
5. decision(OBJECT< Decision >)
6. handled_user(OBJECT< Person >)
7. create_date(NUMBER(Date representation wrapped))
8. update_date(NUMBER(Date representation wrapped))

9. id(NUMBER)

DeleteByIds

URL:

/api/v1/{format}/applications

Method:

DELETE

Parameters request:

Url parameters: ids

Parameters response:

Array

Description:

1. application_form(OBJECT< ApplicationForm >)
2. submitted_user(OBJECT< User >)
3. regarding_user(OBJECT< User >)
4. registration_number(NUMBER)
5. decision(OBJECT< Decision >)
6. handled_user(OBJECT< Person >)
7. create_date(NUMBER(Date representation wrapped))
8. update_date(NUMBER(Date representation wrapped))
9. id(NUMBER)

Get

URL:

/api/v1/{format}/applications/{id}

Method:

GET

Parameters request:

null

Parameters response:

Object

Description:

1. application_form(OBJECT< ApplicationForm >)
2. submitted_user(OBJECT< User >)
3. regarding_user(OBJECT< User >)
4. registration_number(NUMBER)
5. decision(OBJECT< Decision >)
6. handled_user(OBJECT< Person >)
7. create_date(NUMBER(Date representation wrapped))
8. update_date(NUMBER(Date representation wrapped))
9. id(NUMBER)

GetAll

URL:

/api/v1/{format}/applications

Method:

GET

Parameters request:

null

Parameters response:

Array

Description:

1. application_form(OBJECT< ApplicationForm >)
2. submitted_user(OBJECT< User >)
3. regarding_user(OBJECT< User >)
4. registration_number(NUMBER)

5. decision(OBJECT< Decision >)
6. handled_user(OBJECT< Person >)
7. create_date(NUMBER(Date representation wrapped))
8. update_date(NUMBER(Date representation wrapped))
9. id(NUMBER)

SaveAll

URL:

/api/v1/{format}/applications/saveall

Method:

POST

Parameters request:

Array< Application >

Parameters response:

Array

Description:

1. application_form(OBJECT< ApplicationForm >)
2. submitted_user(OBJECT< User >)
3. regarding_user(OBJECT< User >)
4. registration_number(NUMBER)
5. decision(OBJECT< Decision >)
6. handled_user(OBJECT< Person >)
7. create_date(NUMBER(Date representation wrapped))
8. update_date(NUMBER(Date representation wrapped))
9. id(NUMBER)

SaveAllAndReturnIds

URL:

/api/v1/{format}/applications/saveall

Method:

POST

Parameters request:

Url parameters: full

Array< Application >

Parameters response:

Array

Description: ARRAY<NUMBER>

Search

URL:

/api/v1/{format}/applications/search

Method:

POST

Parameters request:

Array< SearchCriteries$SearchCriteriaResult >

Parameters response:

Array

Description:

1. application_form(OBJECT< ApplicationForm >)
2. submitted_user(OBJECT< User >)
3. regarding_user(OBJECT< User >)
4. registration_number(NUMBER)
5. decision(OBJECT< Decision >)
6. handled_user(OBJECT< Person >)
7. create_date(NUMBER(Date representation wrapped))
8. update_date(NUMBER(Date representation wrapped))
9. id(NUMBER)

SearchFirst

URL:

/api/v1/{format}/applications/search/first

Method:

POST

Parameters request:

Array< SearchCriteries$SearchCriteriaResult >

Parameters response:

Object

Description:

1. application_form(OBJECT< ApplicationForm >)
2. submitted_user(OBJECT< User >)
3. regarding_user(OBJECT< User >)
4. registration_number(NUMBER)
5. decision(OBJECT< Decision >)
6. handled_user(OBJECT< Person >)
7. create_date(NUMBER(Date representation wrapped))
8. update_date(NUMBER(Date representation wrapped))
9. id(NUMBER)

Update

URL:

/api/v1/{format}/applications/{id}

Method:

PUT

Parameters request:

Object< Application >

Parameters response:

Object

Description:

1. application_form(OBJECT< ApplicationForm >)
2. submitted_user(OBJECT< User >)
3. regarding_user(OBJECT< User >)
4. registration_number(NUMBER)
5. decision(OBJECT< Decision >)
6. handled_user(OBJECT< Person >)
7. create_date(NUMBER(Date representation wrapped))
8. update_date(NUMBER(Date representation wrapped))
9. id(NUMBER)

EntityVersions

`(implementation of EntityVersion entity)`

Provides following method for API calls:

- *Create*
- *Delete*
- *DeleteByIds*
- *Get*
- *GetAll*
- *GetByEntityId*
- *SaveAll*
- *SaveAllAndReturnIds*
- *Search*
- *SearchFirst*
- *Update*

Create

URL:

/api/v1/{format}/entityversions

Method:

POST

Parameters request:

Object< EntityVersion >

Parameters response:

Object

Description:

1. timestamp(NUMBER(Date representation wrapped))
2. entity_class(STRING<ENTITY_CLASS_NAME>)
3. entity_id(NUMBER)
4. entity(NUMBER)
5. id(NUMBER)

Delete

URL:

/api/v1/{format}/entityversions/{id}

Method:

DELETE

Parameters request:

null

Parameters response:

Object

Description:

1. timestamp(NUMBER(Date representation wrapped))
2. entity_class(STRING<ENTITY_CLASS_NAME>)
3. entity_id(NUMBER)
4. entity(NUMBER)
5. id(NUMBER)

DeleteByIds

URL:

/api/v1/{format}/entityversions

Method:

DELETE

Parameters request:

Url parameters: ids

Parameters response:

Array

Description:

1. timestamp(NUMBER(Date representation wrapped))
2. entity_class(STRING<ENTITY_CLASS_NAME>)
3. entity_id(NUMBER)
4. entity(NUMBER)
5. id(NUMBER)

Get

URL:

/api/v1/{format}/entityversions/{id}

Method:

GET

Parameters request:

null

Parameters response:

Object

Description:

1. timestamp(NUMBER(Date representation wrapped))
2. entity_class(STRING<ENTITY_CLASS_NAME>)
3. entity_id(NUMBER)
4. entity(NUMBER)
5. id(NUMBER)

GetAll

URL:

/api/v1/{format}/entityversions

Method:

GET

Parameters request:

null

Parameters response:

Array

Description:

1. timestamp(NUMBER(Date representation wrapped))
2. entity_class(STRING<ENTITY_CLASS_NAME>)
3. entity_id(NUMBER)
4. entity(NUMBER)
5. id(NUMBER)

GetByEntityId

URL:

/api/v1/{format}/entityversions

Method:

GET

Parameters request:

Url parameters: entityId

Parameters response:

Array

Description:

1. timestamp(NUMBER(Date representation wrapped))
2. entity_class(STRING<ENTITY_CLASS_NAME>)
3. entity_id(NUMBER)
4. entity(NUMBER)
5. id(NUMBER)

SaveAll

URL:

/api/v1/{format}/entityversions/saveall

Method:

POST

Parameters request:

Array< EntityVersion >

Parameters response:

Array

Description:

1. timestamp(NUMBER(Date representation wrapped))
2. entity_class(STRING<ENTITY_CLASS_NAME>)
3. entity_id(NUMBER)
4. entity(NUMBER)
5. id(NUMBER)

SaveAllAndReturnIds

URL:

/api/v1/{format}/entityversions/saveall

Method:

POST

Parameters request:

Url parameters: full

Array< EntityVersion >

Parameters response:

Array

Description: ARRAY<NUMBER>

Search

URL:

/api/v1/{format}/entityversions/search

Method:

POST

Parameters request:

Array< SearchCriteries$SearchCriteriaResult >

Parameters response:

Array

Description:

1. timestamp(NUMBER(Date representation wrapped))
2. entity_class(STRING<ENTITY_CLASS_NAME>)
3. entity_id(NUMBER)
4. entity(NUMBER)

5. id(NUMBER)

SearchFirst

URL:

/api/v1/{format}/entityversions/search/first

Method:

POST

Parameters request:

Array< SearchCriteries$SearchCriteriaResult >

Parameters response:

Object

Description:

1. timestamp(NUMBER(Date representation wrapped))
2. entity_class(STRING<ENTITY_CLASS_NAME>)
3. entity_id(NUMBER)
4. entity(NUMBER)
5. id(NUMBER)

Update

URL:

/api/v1/{format}/entityversions/{id}

Method:

PUT

Parameters request:

Object< EntityVersion >

Parameters response:

Object

Description:

1. timestamp(NUMBER(Date representation wrapped))
2. entity_class(STRING<ENTITY_CLASS_NAME>)
3. entity_id(NUMBER)
4. entity(NUMBER)
5. id(NUMBER)

SchoolTransports

`(implementation of SchoolTransport entity)`

Provides following method for API calls:

- *Create*
- *Delete*
- *DeleteByIds*
- *Get*
- *GetAll*
- *GetByName*
- *GetFirstByName*
- *SaveAll*
- *SaveAllAndReturnIds*
- *Search*
- *SearchFirst*
- *Update*

Create

URL:

/api/v1/{format}/schooltransports

Method:

POST

Parameters request:

Object< SchoolTransport >

Parameters response:

Object

Description:

1. name(STRING)
2. id(NUMBER)

Delete

URL:

/api/v1/{format}/schooltransports/{id}

Method:

DELETE

Parameters request:

null

Parameters response:

Object

Description:

1. name(STRING)
2. id(NUMBER)

DeleteByIds

URL:

/api/v1/{format}/schooltransports

Method:

DELETE

Parameters request:

Url parameters: ids

Parameters response:

Array

Description:

1. name(STRING)
2. id(NUMBER)

Get

URL:

/api/v1/{format}/schooltransports/{id}

Method:

GET

Parameters request:

null

Parameters response:

Object

Description:

1. name(STRING)
2. id(NUMBER)

GetAll

URL:

/api/v1/{format}/schooltransports

Method:

GET

Parameters request:

null

Parameters response:

Array

Description:

1. name(STRING)
2. id(NUMBER)

GetByName

URL:

/api/v1/{format}/schooltransports

Method:

GET

Parameters request:

Url parameters: name

Parameters response:

Array

Description:

1. name(STRING)
2. id(NUMBER)

GetFirstByName

URL:

/api/v1/{format}/schooltransports

Method:

GET

Parameters request:

Url parameters: name, first

Parameters response:

Object

Description:

1. name(STRING)
2. id(NUMBER)

SaveAll

URL:

/api/v1/{format}/schooltransports/saveall

Method:

POST

Parameters request:

Array< SchoolTransport >

Parameters response:

Array

Description:

1. name(STRING)
2. id(NUMBER)

SaveAllAndReturnIds

URL:

/api/v1/{format}/schooltransports/saveall

Method:

POST

Parameters request:

Url parameters: full

Array< SchoolTransport >

Parameters response:

Array

Description: ARRAY<NUMBER>

Search

URL:

/api/v1/{format}/schooltransports/search

Method:

POST

Parameters request:

Array< SearchCriteries$SearchCriteriaResult >

Parameters response:

Array

Description:

1. name(STRING)
2. id(NUMBER)

SearchFirst

URL:

/api/v1/{format}/schooltransports/search/first

Method:

POST

Parameters request:

Array< SearchCriteries$SearchCriteriaResult >

Parameters response:

Object

Description:

1. name(STRING)
2. id(NUMBER)

Update

URL:

/api/v1/{format}/schooltransports/{id}

Method:

PUT

Parameters request:

Object< SchoolTransport >

Parameters response:

Object

Description:

1. name(STRING)
2. id(NUMBER)

INCIDENT

Activities

`(implementation of Activity entity)`

Provides following method for API calls:

- *Create*
- *Delete*
- *DeleteByIds*
- *Get*
- *GetAll*
- *GetAttachment*
- *SaveAll*

- *SaveAllAndReturnIds*
- *Search*
- *SearchFirst*
- *SetAttachment*
- *Update*

Create

URL:

/api/v1/{format}/activities

Method:

POST

Parameters request:

Object< Activity >

Parameters response:

Object

Description:

1. description(STRING)
2. file_name(STRING)
3. issue(OBJECT< Issue >)
4. reported_date(NUMBER(Date representation wrapped))
5. reported_by(OBJECT< Person >)
6. id(NUMBER)

Delete

URL:

/api/v1/{format}/activities/{id}

Method:

DELETE

Parameters request:

null

Parameters response:

Object

Description:

1. description(STRING)
2. file_name(STRING)
3. issue(OBJECT< Issue >)
4. reported_date(NUMBER(Date representation wrapped))
5. reported_by(OBJECT< Person >)
6. id(NUMBER)

DeleteByIds

URL:

/api/v1/{format}/activities

Method:

DELETE

Parameters request:

Url parameters: ids

Parameters response:

Array

Description:

1. description(STRING)
2. file_name(STRING)
3. issue(OBJECT< Issue >)
4. reported_date(NUMBER(Date representation wrapped))
5. reported_by(OBJECT< Person >)
6. id(NUMBER)

Get

URL:

/api/v1/{format}/activities/{id}

Method:

GET

Parameters request:

null

Parameters response:

Object

Description:

1. description(STRING)
2. file_name(STRING)
3. issue(OBJECT< Issue >)
4. reported_date(NUMBER(Date representation wrapped))
5. reported_by(OBJECT< Person >)
6. id(NUMBER)

GetAll

URL:

/api/v1/{format}/activities

Method:

GET

Parameters request:

null

Parameters response:

Array

Description:

1. description(STRING)
2. file_name(STRING)
3. issue(OBJECT< Issue >)
4. reported_date(NUMBER(Date representation wrapped))
5. reported_by(OBJECT< Person >)
6. id(NUMBER)

GetAttachment

URL:

/api/v1/{format}/activities/attach/{id}

Method:

GET

Parameters request:

null

Parameters response:

Object

Description: Starting downloading of file

Note: After redirecting user to url, will be started downloading.

SaveAll

URL:

/api/v1/{format}/activities/saveall

Method:

POST

Parameters request:

Array< Activity >

Parameters response:

Array

Description:

1. description(STRING)
2. file_name(STRING)
3. issue(OBJECT< Issue >)
4. reported_date(NUMBER(Date representation wrapped))
5. reported_by(OBJECT< Person >)
6. id(NUMBER)

SaveAllAndReturnIds

URL:

/api/v1/{format}/activities/saveall

Method:

POST

Parameters request:

Url parameters: full

Array< Activity >

Parameters response:

Array

Description: ARRAY<NUMBER>

Search

URL:

/api/v1/{format}/activities/search

Method:

POST

Parameters request:

Array< SearchCriteries$SearchCriteriaResult >

Parameters response:

Array

Description:

1. description(STRING)
2. file_name(STRING)
3. issue(OBJECT< Issue >)
4. reported_date(NUMBER(Date representation wrapped))
5. reported_by(OBJECT< Person >)
6. id(NUMBER)

SearchFirst

URL:

/api/v1/{format}/activities/search/first

Method:

POST

Parameters request:

Array< SearchCriteries$SearchCriteriaResult >

Parameters response:

Object

Description:

1. description(STRING)
2. file_name(STRING)
3. issue(OBJECT< Issue >)
4. reported_date(NUMBER(Date representation wrapped))

5. reported_by(OBJECT< Person >)
6. id(NUMBER)

SetAttachment

URL:

/api/v1/{format}/activities/attach/{id}

Method:

POST

Parameters request:

null

Parameters response:

Object

Description:

Note: You need submit form with action url.

In form must be input file!

Max file size 20Mb.

Update

URL:

/api/v1/{format}/activities/{id}

Method:

PUT

Parameters request:

Object< Activity >

Parameters response:

Object

Description:

1. description(STRING)
2. file_name(STRING)
3. issue(OBJECT< Issue >)
4. reported_date(NUMBER(Date representation wrapped))
5. reported_by(OBJECT< Person >)
6. id(NUMBER)

Categories

`(implementation of Category entity)`

Provides following method for API calls:

- *Create*
- *Delete*
- *DeleteByIds*
- *Get*
- *GetAll*
- *SaveAll*
- *SaveAllAndReturnIds*
- *Search*
- *SearchFirst*
- *Update*

Create

URL:

/api/v1/{format}/categories

Method:

POST

Parameters request:

Object< Category >

Parameters response:

Object

Description:

1. name(STRING)
2. id(NUMBER)

Delete

URL:

/api/v1/{format}/categories/{id}

Method:

DELETE

Parameters request:

null

Parameters response:

Object

Description:

1. name(STRING)
2. id(NUMBER)

DeleteByIds

URL:

/api/v1/{format}/categories

Method:

DELETE

Parameters request:

Url parameters: ids

Parameters response:

Array

Description:

1. name(STRING)
2. id(NUMBER)

Get

URL:

/api/v1/{format}/categories/{id}

Method:

GET

Parameters request:

null

Parameters response:

Object

Description:

1. name(STRING)
2. id(NUMBER)

GetAll

URL:

/api/v1/{format}/categories

Method:

GET

Parameters request:

null

Parameters response:

Array

Description:

1. name(STRING)
2. id(NUMBER)

SaveAll

URL:

/api/v1/{format}/categories/saveall

Method:

POST

Parameters request:

Array< Category >

Parameters response:

Array

Description:

1. name(STRING)
2. id(NUMBER)

SaveAllAndReturnIds

URL:

/api/v1/{format}/categories/saveall

Method:

POST

Parameters request:

Url parameters: full

Array< Category >

Parameters response:

Array

Description: ARRAY<NUMBER>

Search

URL:

/api/v1/{format}/categories/search

Method:

POST

Parameters request:

Array< SearchCriteries$SearchCriteriaResult >

Parameters response:

Array

Description:

1. name(STRING)
2. id(NUMBER)

SearchFirst

URL:

/api/v1/{format}/categories/search/first

Method:

POST

Parameters request:

Array< SearchCriteries$SearchCriteriaResult >

Parameters response:

Object

Description:

1. name(STRING)
2. id(NUMBER)

Update

URL:

/api/v1/{format}/categories/{id}

Method:

PUT

Parameters request:

Object< Category >

Parameters response:

Object

Description:

1. name(STRING)
2. id(NUMBER)

Incidents

`(implementation of Incident entity)`

Provides following method for API calls:

- *Create*
- *Delete*
- *DeleteByIds*
- *FindByCriteria*
- *Get*
- *GetAll*
- *SaveAll*
- *SaveAllAndReturnIds*
- *Search*

- *SearchFirst*
- *Update*

Create

URL:

/api/v1/{format}/incidents

Method:

POST

Parameters request:

Object< Incident >

Parameters response:

Object

Description:

1. title(STRING)
2. description(STRING)
3. reported_date(NUMBER(Date representation wrapped))
4. categories(ARRAY< Category >)
5. pupils(ARRAY< Pupil >)
6. status(OBJECT< Status >)
7. priority(OBJECT< Priority >)
8. issue(OBJECT< Issue >)
9. assigned_date(NUMBER(Date representation wrapped))
10. assigned_by(OBJECT< Person >)
11. archived_date(NUMBER(Date representation wrapped))
12. archived_by(OBJECT< Person >)
13. reported_by(OBJECT< Person >)
14. modified_by(OBJECT< Person >)
15. modified_date(NUMBER(Date representation wrapped))
16. id(NUMBER)

Delete

URL:

/api/v1/{format}/incidents/{id}

Method:

DELETE

Parameters request:

null

Parameters response:

Object

Description:

1. title(STRING)
2. description(STRING)
3. reported_date(NUMBER(Date representation wrapped))
4. categories(ARRAY< Category >)
5. pupils(ARRAY< Pupil >)
6. status(OBJECT< Status >)
7. priority(OBJECT< Priority >)
8. issue(OBJECT< Issue >)
9. assigned_date(NUMBER(Date representation wrapped))
10. assigned_by(OBJECT< Person >)
11. archived_date(NUMBER(Date representation wrapped))
12. archived_by(OBJECT< Person >)
13. reported_by(OBJECT< Person >)
14. modified_by(OBJECT< Person >)
15. modified_date(NUMBER(Date representation wrapped))
16. id(NUMBER)

DeleteByIds

URL:

/api/v1/{format}/incidents

Method:

DELETE

Parameters request:

Url parameters: ids

Parameters response:

Array

Description:

1. title(STRING)
2. description(STRING)
3. reported_date(NUMBER(Date representation wrapped))
4. categories(ARRAY< Category >)
5. pupils(ARRAY< Pupil >)
6. status(OBJECT< Status >)
7. priority(OBJECT< Priority >)
8. issue(OBJECT< Issue >)
9. assigned_date(NUMBER(Date representation wrapped))
10. assigned_by(OBJECT< Person >)
11. archived_date(NUMBER(Date representation wrapped))
12. archived_by(OBJECT< Person >)
13. reported_by(OBJECT< Person >)
14. modified_by(OBJECT< Person >)
15. modified_date(NUMBER(Date representation wrapped))
16. id(NUMBER)

FindByCriteria

URL:

/api/v1/{format}/incidents

Method:

GET

Parameters request:

Url parameters: search_text, order_by

Parameters response:

Array

Description:

1. title(STRING)
2. description(STRING)
3. reported_date(NUMBER(Date representation wrapped))
4. categories(ARRAY< Category >)
5. pupils(ARRAY< Pupil >)
6. status(OBJECT< Status >)
7. priority(OBJECT< Priority >)
8. issue(OBJECT< Issue >)
9. assigned_date(NUMBER(Date representation wrapped))
10. assigned_by(OBJECT< Person >)
11. archived_date(NUMBER(Date representation wrapped))
12. archived_by(OBJECT< Person >)
13. reported_by(OBJECT< Person >)
14. modified_by(OBJECT< Person >)
15. modified_date(NUMBER(Date representation wrapped))
16. id(NUMBER)

Get

URL:

/api/v1/{format}/incidents/{id}

Method:

GET

Parameters request:

null

Parameters response:

Object

Description:

1. title(STRING)
2. description(STRING)
3. reported_date(NUMBER(Date representation wrapped))
4. categories(ARRAY< Category >)
5. pupils(ARRAY< Pupil >)
6. status(OBJECT< Status >)
7. priority(OBJECT< Priority >)
8. issue(OBJECT< Issue >)
9. assigned_date(NUMBER(Date representation wrapped))
10. assigned_by(OBJECT< Person >)
11. archived_date(NUMBER(Date representation wrapped))
12. archived_by(OBJECT< Person >)
13. reported_by(OBJECT< Person >)
14. modified_by(OBJECT< Person >)
15. modified_date(NUMBER(Date representation wrapped))
16. id(NUMBER)

GetAll

URL:

/api/v1/{format}/incidents

Method:

GET

Parameters request:

null

Parameters response:

Array

Description:

1. title(STRING)

2. description(STRING)
3. reported_date(NUMBER(Date representation wrapped))
4. categories(ARRAY< Category >)
5. pupils(ARRAY< Pupil >)
6. status(OBJECT< Status >)
7. priority(OBJECT< Priority >)
8. issue(OBJECT< Issue >)
9. assigned_date(NUMBER(Date representation wrapped))
10. assigned_by(OBJECT< Person >)
11. archived_date(NUMBER(Date representation wrapped))
12. archived_by(OBJECT< Person >)
13. reported_by(OBJECT< Person >)
14. modified_by(OBJECT< Person >)
15. modified_date(NUMBER(Date representation wrapped))
16. id(NUMBER)

SaveAll

URL:

/api/v1/{format}/incidents/saveall

Method:

POST

Parameters request:

Array< Incident >

Parameters response:

Array

Description:

1. title(STRING)
2. description(STRING)
3. reported_date(NUMBER(Date representation wrapped))
4. categories(ARRAY< Category >)
5. pupils(ARRAY< Pupil >)

6. status(OBJECT< Status >)
7. priority(OBJECT< Priority >)
8. issue(OBJECT< Issue >)
9. assigned_date(NUMBER(Date representation wrapped))
10. assigned_by(OBJECT< Person >)
11. archived_date(NUMBER(Date representation wrapped))
12. archived_by(OBJECT< Person >)
13. reported_by(OBJECT< Person >)
14. modified_by(OBJECT< Person >)
15. modified_date(NUMBER(Date representation wrapped))
16. id(NUMBER)

SaveAllAndReturnIds

URL:

/api/v1/{format}/incidents/saveall

Method:

POST

Parameters request:

Url parameters: full

Array< Incident >

Parameters response:

Array

Description: ARRAY<NUMBER>

Search

URL:

/api/v1/{format}/incidents/search

Method:

POST

Parameters request:

Array< SearchCriteries$SearchCriteriaResult >

Parameters response:

Array

Description:

1. title(STRING)
2. description(STRING)
3. reported_date(NUMBER(Date representation wrapped))
4. categories(ARRAY< Category >)
5. pupils(ARRAY< Pupil >)
6. status(OBJECT< Status >)
7. priority(OBJECT< Priority >)
8. issue(OBJECT< Issue >)
9. assigned_date(NUMBER(Date representation wrapped))
10. assigned_by(OBJECT< Person >)
11. archived_date(NUMBER(Date representation wrapped))
12. archived_by(OBJECT< Person >)
13. reported_by(OBJECT< Person >)
14. modified_by(OBJECT< Person >)
15. modified_date(NUMBER(Date representation wrapped))
16. id(NUMBER)

SearchFirst

URL:

/api/v1/{format}/incidents/search/first

Method:

POST

Parameters request:

Array< SearchCriteries$SearchCriteriaResult >

Parameters response:

Object

Description:

1. title(STRING)
2. description(STRING)
3. reported_date(NUMBER(Date representation wrapped))
4. categories(ARRAY< Category >)
5. pupils(ARRAY< Pupil >)
6. status(OBJECT< Status >)
7. priority(OBJECT< Priority >)
8. issue(OBJECT< Issue >)
9. assigned_date(NUMBER(Date representation wrapped))
10. assigned_by(OBJECT< Person >)
11. archived_date(NUMBER(Date representation wrapped))
12. archived_by(OBJECT< Person >)
13. reported_by(OBJECT< Person >)
14. modified_by(OBJECT< Person >)
15. modified_date(NUMBER(Date representation wrapped))
16. id(NUMBER)

Update

URL:

/api/v1/{format}/incidents/{id}

Method:

PUT

Parameters request:

Object< Incident >

Parameters response:

Object

Description:

1. title(STRING)

2. description(STRING)
3. reported_date(NUMBER(Date representation wrapped))
4. categories(ARRAY< Category >)
5. pupils(ARRAY< Pupil >)
6. status(OBJECT< Status >)
7. priority(OBJECT< Priority >)
8. issue(OBJECT< Issue >)
9. assigned_date(NUMBER(Date representation wrapped))
10. assigned_by(OBJECT< Person >)
11. archived_date(NUMBER(Date representation wrapped))
12. archived_by(OBJECT< Person >)
13. reported_by(OBJECT< Person >)
14. modified_by(OBJECT< Person >)
15. modified_date(NUMBER(Date representation wrapped))
16. id(NUMBER)

Issues

`(implementation of Issue entity)`

Provides following method for API calls:

- *Create*
- *Delete*
- *DeleteByIds*
- *FindByCriteria*
- *Get*
- *GetAll*
- *SaveAll*
- *SaveAllAndReturnIds*
- *Search*
- *SearchFirst*
- *Update*

Create

URL:

/api/v1/{format}/issues

Method:

POST

Parameters request:

Object< Issue >

Parameters response:

Object

Description:

1. title(STRING)
2. description(STRING)
3. status(OBJECT< Status >)
4. responsible_person(OBJECT< Person >)
5. authorized_persons(ARRAY< Person >)
6. incidents(ARRAY< Incident >)
7. reported_date(NUMBER(Date representation wrapped))
8. reported_by(OBJECT< Person >)
9. modified_by(OBJECT< Person >)
10. modified_date(NUMBER(Date representation wrapped))
11. activities(ARRAY< Activity >)
12. categories(ARRAY< Category >)
13. priority(OBJECT< Priority >)
14. pupils(ARRAY< Pupil >)
15. id(NUMBER)

Delete

URL:

/api/v1/{format}/issues/{id}

Method:

DELETE

Parameters request:

null

Parameters response:

Object

Description:

1. title(STRING)
2. description(STRING)
3. status(OBJECT< Status >)
4. responsible_person(OBJECT< Person >)
5. authorized_persons(ARRAY< Person >)
6. incidents(ARRAY< Incident >)
7. reported_date(NUMBER(Date representation wrapped))
8. reported_by(OBJECT< Person >)
9. modified_by(OBJECT< Person >)
10. modified_date(NUMBER(Date representation wrapped))
11. activities(ARRAY< Activity >)
12. categories(ARRAY< Category >)
13. priority(OBJECT< Priority >)
14. pupils(ARRAY< Pupil >)
15. id(NUMBER)

DeleteByIds

URL:

/api/v1/{format}/issues

Method:

DELETE

Parameters request:

Url parameters: ids

Parameters response:

Array

Description:

1. title(STRING)
2. description(STRING)
3. status(OBJECT< Status >)
4. responsible_person(OBJECT< Person >)
5. authorized_persons(ARRAY< Person >)
6. incidents(ARRAY< Incident >)
7. reported_date(NUMBER(Date representation wrapped))
8. reported_by(OBJECT< Person >)
9. modified_by(OBJECT< Person >)
10. modified_date(NUMBER(Date representation wrapped))
11. activities(ARRAY< Activity >)
12. categories(ARRAY< Category >)
13. priority(OBJECT< Priority >)
14. pupils(ARRAY< Pupil >)
15. id(NUMBER)

FindByCriteria

URL:

/api/v1/{format}/issues

Method:

GET

Parameters request:

Url parameters: search_text, order_by

Parameters response:

Array

Description:

1. title(STRING)
2. description(STRING)

3. status(OBJECT< Status >)
4. responsible_person(OBJECT< Person >)
5. authorized_persons(ARRAY< Person >)
6. incidents(ARRAY< Incident >)
7. reported_date(NUMBER(Date representation wrapped))
8. reported_by(OBJECT< Person >)
9. modified_by(OBJECT< Person >)
10. modified_date(NUMBER(Date representation wrapped))
11. activities(ARRAY< Activity >)
12. categories(ARRAY< Category >)
13. priority(OBJECT< Priority >)
14. pupils(ARRAY< Pupil >)
15. id(NUMBER)

Get

URL:

/api/v1/{format}/issues/{id}

Method:

GET

Parameters request:

null

Parameters response:

Object

Description:

1. title(STRING)
2. description(STRING)
3. status(OBJECT< Status >)
4. responsible_person(OBJECT< Person >)
5. authorized_persons(ARRAY< Person >)
6. incidents(ARRAY< Incident >)
7. reported_date(NUMBER(Date representation wrapped))

8. reported_by(OBJECT< Person >)
9. modified_by(OBJECT< Person >)
10. modified_date(NUMBER(Date representation wrapped))
11. activities(ARRAY< Activity >)
12. categories(ARRAY< Category >)
13. priority(OBJECT< Priority >)
14. pupils(ARRAY< Pupil >)
15. id(NUMBER)

GetAll

URL:

/api/v1/{format}/issues

Method:

GET

Parameters request:

null

Parameters response:

Array

Description:

1. title(STRING)
2. description(STRING)
3. status(OBJECT< Status >)
4. responsible_person(OBJECT< Person >)
5. authorized_persons(ARRAY< Person >)
6. incidents(ARRAY< Incident >)
7. reported_date(NUMBER(Date representation wrapped))
8. reported_by(OBJECT< Person >)
9. modified_by(OBJECT< Person >)
10. modified_date(NUMBER(Date representation wrapped))
11. activities(ARRAY< Activity >)
12. categories(ARRAY< Category >)

13. priority(OBJECT< Priority >)
14. pupils(ARRAY< Pupil >)
15. id(NUMBER)

SaveAll

URL:

/api/v1/{format}/issues/saveall

Method:

POST

Parameters request:

Array< Issue >

Parameters response:

Array

Description:

1. title(STRING)
2. description(STRING)
3. status(OBJECT< Status >)
4. responsible_person(OBJECT< Person >)
5. authorized_persons(ARRAY< Person >)
6. incidents(ARRAY< Incident >)
7. reported_date(NUMBER(Date representation wrapped))
8. reported_by(OBJECT< Person >)
9. modified_by(OBJECT< Person >)
10. modified_date(NUMBER(Date representation wrapped))
11. activities(ARRAY< Activity >)
12. categories(ARRAY< Category >)
13. priority(OBJECT< Priority >)
14. pupils(ARRAY< Pupil >)
15. id(NUMBER)

SaveAllAndReturnIds

URL:

/api/v1/{format}/issues/saveall

Method:

POST

Parameters request:

Url parameters: full

Array< Issue >

Parameters response:

Array

Description: ARRAY<NUMBER>

Search

URL:

/api/v1/{format}/issues/search

Method:

POST

Parameters request:

Array< SearchCriteries$SearchCriteriaResult >

Parameters response:

Array

Description:

1. title(STRING)
2. description(STRING)
3. status(OBJECT< Status >)
4. responsible_person(OBJECT< Person >)

5. authorized_persons(ARRAY< Person >)
6. incidents(ARRAY< Incident >)
7. reported_date(NUMBER(Date representation wrapped))
8. reported_by(OBJECT< Person >)
9. modified_by(OBJECT< Person >)
10. modified_date(NUMBER(Date representation wrapped))
11. activities(ARRAY< Activity >)
12. categories(ARRAY< Category >)
13. priority(OBJECT< Priority >)
14. pupils(ARRAY< Pupil >)
15. id(NUMBER)

SearchFirst

URL:

/api/v1/{format}/issues/search/first

Method:

POST

Parameters request:

Array< SearchCriteries$SearchCriteriaResult >

Parameters response:

Object

Description:

1. title(STRING)
2. description(STRING)
3. status(OBJECT< Status >)
4. responsible_person(OBJECT< Person >)
5. authorized_persons(ARRAY< Person >)
6. incidents(ARRAY< Incident >)
7. reported_date(NUMBER(Date representation wrapped))
8. reported_by(OBJECT< Person >)
9. modified_by(OBJECT< Person >)

10. modified_date(NUMBER(Date representation wrapped))
11. activities(ARRAY< Activity >)
12. categories(ARRAY< Category >)
13. priority(OBJECT< Priority >)
14. pupils(ARRAY< Pupil >)
15. id(NUMBER)

Update

URL:

/api/v1/{format}/issues/{id}

Method:

PUT

Parameters request:

Object< Issue >

Parameters response:

Object

Description:

1. title(STRING)
2. description(STRING)
3. status(OBJECT< Status >)
4. responsible_person(OBJECT< Person >)
5. authorized_persons(ARRAY< Person >)
6. incidents(ARRAY< Incident >)
7. reported_date(NUMBER(Date representation wrapped))
8. reported_by(OBJECT< Person >)
9. modified_by(OBJECT< Person >)
10. modified_date(NUMBER(Date representation wrapped))
11. activities(ARRAY< Activity >)
12. categories(ARRAY< Category >)
13. priority(OBJECT< Priority >)
14. pupils(ARRAY< Pupil >)

15. id(NUMBER)

LogEvents

`(implementation of LogEvent entity)`

Provides following method for API calls:

- *Create*
- *Delete*
- *DeleteByIds*
- *Get*
- *GetAll*
- *GetByEntityId*
- *SaveAll*
- *SaveAllAndReturnIds*
- *Search*
- *SearchFirst*
- *Update*

Create

URL:

/api/v1/{format}/logevents

Method:

POST

Parameters request:

Object< LogEvent >

Parameters response:

Object

Description:

1. timestamp(NUMBER(Date representation wrapped))
2. entity_class_name(STRING)
3. entity_id(NUMBER)
4. action(OBJECT< LogEvent$Action >)

5. field_name(STRING)
6. previous_value(STRING)
7. new_value(STRING)
8. user(OBJECT< User >)
9. id(NUMBER)

Delete

URL:

/api/v1/{format}/logevents/{id}

Method:

DELETE

Parameters request:

null

Parameters response:

Object

Description:

1. timestamp(NUMBER(Date representation wrapped))
2. entity_class_name(STRING)
3. entity_id(NUMBER)
4. action(OBJECT< LogEvent$Action >)
5. field_name(STRING)
6. previous_value(STRING)
7. new_value(STRING)
8. user(OBJECT< User >)
9. id(NUMBER)

DeleteByIds

URL:

/api/v1/{format}/logevents

Method:

DELETE

Parameters request:

Url parameters: ids

Parameters response:

Array

Description:

1. timestamp(NUMBER(Date representation wrapped))
2. entity_class_name(STRING)
3. entity_id(NUMBER)
4. action(OBJECT< LogEvent$Action >)
5. field_name(STRING)
6. previous_value(STRING)
7. new_value(STRING)
8. user(OBJECT< User >)
9. id(NUMBER)

Get

URL:

/api/v1/{format}/logevents/{id}

Method:

GET

Parameters request:

null

Parameters response:

Object

Description:

1. timestamp(NUMBER(Date representation wrapped))

2. entity_class_name(STRING)
3. entity_id(NUMBER)
4. action(OBJECT< LogEvent$Action >)
5. field_name(STRING)
6. previous_value(STRING)
7. new_value(STRING)
8. user(OBJECT< User >)
9. id(NUMBER)

GetAll

URL:

/api/v1/{format}/logevents

Method:

GET

Parameters request:

null

Parameters response:

Array

Description:

1. timestamp(NUMBER(Date representation wrapped))
2. entity_class_name(STRING)
3. entity_id(NUMBER)
4. action(OBJECT< LogEvent$Action >)
5. field_name(STRING)
6. previous_value(STRING)
7. new_value(STRING)
8. user(OBJECT< User >)
9. id(NUMBER)

GetByEntityId

URL:

/api/v1/{format}/logevents

Method:

GET

Parameters request:

Url parameters: entityId

Parameters response:

Array

Description:

1. timestamp(NUMBER(Date representation wrapped))
2. entity_class_name(STRING)
3. entity_id(NUMBER)
4. action(OBJECT< LogEvent$Action >)
5. field_name(STRING)
6. previous_value(STRING)
7. new_value(STRING)
8. user(OBJECT< User >)
9. id(NUMBER)

SaveAll

URL:

/api/v1/{format}/logevents/saveall

Method:

POST

Parameters request:

Array< LogEvent >

Parameters response:

Array

Description:

1. timestamp(NUMBER(Date representation wrapped))
2. entity_class_name(STRING)
3. entity_id(NUMBER)
4. action(OBJECT< LogEvent$Action >)
5. field_name(STRING)
6. previous_value(STRING)
7. new_value(STRING)
8. user(OBJECT< User >)
9. id(NUMBER)

SaveAllAndReturnIds

URL:

/api/v1/{format}/logevents/saveall

Method:

POST

Parameters request:

Url parameters: full

Array< LogEvent >

Parameters response:

Array

Description: ARRAY<NUMBER>

Search

URL:

/api/v1/{format}/logevents/search

Method:

POST

Parameters request:

Array< SearchCriteries$SearchCriteriaResult >

Parameters response:

Array

Description:

1. timestamp(NUMBER(Date representation wrapped))
2. entity_class_name(STRING)
3. entity_id(NUMBER)
4. action(OBJECT< LogEvent$Action >)
5. field_name(STRING)
6. previous_value(STRING)
7. new_value(STRING)
8. user(OBJECT< User >)
9. id(NUMBER)

SearchFirst

URL:

/api/v1/{format}/logevents/search/first

Method:

POST

Parameters request:

Array< SearchCriteries$SearchCriteriaResult >

Parameters response:

Object

Description:

1. timestamp(NUMBER(Date representation wrapped))

2. entity_class_name(STRING)
3. entity_id(NUMBER)
4. action(OBJECT< LogEvent$Action >)
5. field_name(STRING)
6. previous_value(STRING)
7. new_value(STRING)
8. user(OBJECT< User >)
9. id(NUMBER)

Update

URL:

/api/v1/{format}/logevents/{id}

Method:

PUT

Parameters request:

Object< LogEvent >

Parameters response:

Object

Description:

1. timestamp(NUMBER(Date representation wrapped))
2. entity_class_name(STRING)
3. entity_id(NUMBER)
4. action(OBJECT< LogEvent$Action >)
5. field_name(STRING)
6. previous_value(STRING)
7. new_value(STRING)
8. user(OBJECT< User >)
9. id(NUMBER)

PersonRoles

`(implementation of PersonRole entity)`

Provides following method for API calls:

- *Create*
- *Delete*
- *DeleteByIds*
- *Get*
- *GetAll*
- *SaveAll*
- *SaveAllAndReturnIds*
- *Search*
- *SearchFirst*
- *Update*

Create

URL:

/api/v1/{format}/personroles

Method:

POST

Parameters request:

Object< PersonRole >

Parameters response:

Object

Description:

1. person(OBJECT< Person >)
2. role(OBJECT< WorkRole >)
3. school(OBJECT< School >)
4. school_class(OBJECT< SchoolClass >)
5. date_from(NUMBER(Date representation wrapped))
6. date_to(NUMBER(Date representation wrapped))
7. id(NUMBER)

Delete

URL:

/api/v1/{format}/personroles/{id}

Method:

DELETE

Parameters request:

null

Parameters response:

Object

Description:

1. person(OBJECT< Person >)
2. role(OBJECT< WorkRole >)
3. school(OBJECT< School >)
4. school_class(OBJECT< SchoolClass >)
5. date_from(NUMBER(Date representation wrapped))
6. date_to(NUMBER(Date representation wrapped))
7. id(NUMBER)

DeleteByIds

URL:

/api/v1/{format}/personroles

Method:

DELETE

Parameters request:

Url parameters: ids

Parameters response:

Array

Description:

1. person(OBJECT< Person >)
2. role(OBJECT< WorkRole >)
3. school(OBJECT< School >)
4. school_class(OBJECT< SchoolClass >)
5. date_from(NUMBER(Date representation wrapped))
6. date_to(NUMBER(Date representation wrapped))
7. id(NUMBER)

Get

URL:

/api/v1/{format}/personroles/{id}

Method:

GET

Parameters request:

null

Parameters response:

Object

Description:

1. person(OBJECT< Person >)
2. role(OBJECT< WorkRole >)
3. school(OBJECT< School >)
4. school_class(OBJECT< SchoolClass >)
5. date_from(NUMBER(Date representation wrapped))
6. date_to(NUMBER(Date representation wrapped))
7. id(NUMBER)

GetAll

URL:

/api/v1/{format}/personroles

Method:

GET

Parameters request:

null

Parameters response:

Array

Description:

1. person(OBJECT< Person >)
2. role(OBJECT< WorkRole >)
3. school(OBJECT< School >)
4. school_class(OBJECT< SchoolClass >)
5. date_from(NUMBER(Date representation wrapped))
6. date_to(NUMBER(Date representation wrapped))
7. id(NUMBER)

SaveAll

URL:

/api/v1/{format}/personroles/saveall

Method:

POST

Parameters request:

Array< PersonRole >

Parameters response:

Array

Description:

1. person(OBJECT< Person >)
2. role(OBJECT< WorkRole >)
3. school(OBJECT< School >)
4. school_class(OBJECT< SchoolClass >)
5. date_from(NUMBER(Date representation wrapped))
6. date_to(NUMBER(Date representation wrapped))
7. id(NUMBER)

SaveAllAndReturnIds

URL:

/api/v1/{format}/personroles/saveall

Method:

POST

Parameters request:

Url parameters: full

Array< PersonRole >

Parameters response:

Array

Description: ARRAY<NUMBER>

Search

URL:

/api/v1/{format}/personroles/search

Method:

POST

Parameters request:

Array< SearchCriteries$SearchCriteriaResult >

Parameters response:

Array

Description:

1. person(OBJECT< Person >)
2. role(OBJECT< WorkRole >)
3. school(OBJECT< School >)
4. school_class(OBJECT< SchoolClass >)
5. date_from(NUMBER(Date representation wrapped))
6. date_to(NUMBER(Date representation wrapped))
7. id(NUMBER)

SearchFirst

URL:

/api/v1/{format}/personroles/search/first

Method:

POST

Parameters request:

Array< SearchCriteries$SearchCriteriaResult >

Parameters response:

Object

Description:

1. person(OBJECT< Person >)
2. role(OBJECT< WorkRole >)
3. school(OBJECT< School >)
4. school_class(OBJECT< SchoolClass >)
5. date_from(NUMBER(Date representation wrapped))
6. date_to(NUMBER(Date representation wrapped))

7. id(NUMBER)

Update

URL:

/api/v1/{format}/personroles/{id}

Method:

PUT

Parameters request:

Object< PersonRole >

Parameters response:

Object

Description:

1. person(OBJECT< Person >)
2. role(OBJECT< WorkRole >)
3. school(OBJECT< School >)
4. school_class(OBJECT< SchoolClass >)
5. date_from(NUMBER(Date representation wrapped))
6. date_to(NUMBER(Date representation wrapped))
7. id(NUMBER)

Priorities

```
(implementation of Priority entity)
```

Provides following method for API calls:

- *Create*
- *Delete*
- *DeleteByIds*
- *Get*
- *GetAll*
- *SaveAll*
- *SaveAllAndReturnIds*
- *Search*

- *SearchFirst*
- *Update*

Create

URL:

/api/v1/{format}/priorities

Method:

POST

Parameters request:

Object< Priority >

Parameters response:

Object

Description:

1. name(STRING)
2. id(NUMBER)

Delete

URL:

/api/v1/{format}/priorities/{id}

Method:

DELETE

Parameters request:

null

Parameters response:

Object

Description:

1. name(STRING)
2. id(NUMBER)

DeleteByIds

URL:

/api/v1/{format}/priorities

Method:

DELETE

Parameters request:

Url parameters: ids

Parameters response:

Array

Description:

1. name(STRING)
2. id(NUMBER)

Get

URL:

/api/v1/{format}/priorities/{id}

Method:

GET

Parameters request:

null

Parameters response:

Object

Description:

1. name(STRING)
2. id(NUMBER)

GetAll

URL:

/api/v1/{format}/priorities

Method:

GET

Parameters request:

null

Parameters response:

Array

Description:

1. name(STRING)
2. id(NUMBER)

SaveAll

URL:

/api/v1/{format}/priorities/saveall

Method:

POST

Parameters request:

Array< Priority >

Parameters response:

Array

Description:

1. name(STRING)
2. id(NUMBER)

SaveAllAndReturnIds

URL:

/api/v1/{format}/priorities/saveall

Method:

POST

Parameters request:

Url parameters: full

Array< Priority >

Parameters response:

Array

Description: ARRAY<NUMBER>

Search

URL:

/api/v1/{format}/priorities/search

Method:

POST

Parameters request:

Array< SearchCriteries$SearchCriteriaResult >

Parameters response:

Array

Description:

1. name(STRING)
2. id(NUMBER)

SearchFirst

URL:

/api/v1/{format}/priorities/search/first

Method:

POST

Parameters request:

Array< SearchCriteries$SearchCriteriaResult >

Parameters response:

Object

Description:

1. name(STRING)
2. id(NUMBER)

Update

URL:

/api/v1/{format}/priorities/{id}

Method:

PUT

Parameters request:

Object< Priority >

Parameters response:

Object

Description:

1. name(STRING)
2. id(NUMBER)

Statuses

`(implementation of Status entity)`

Provides following method for API calls:

- *Create*
- *Delete*
- *DeleteByIds*
- *Get*
- *GetAll*
- *SaveAll*
- *SaveAllAndReturnIds*
- *Search*
- *SearchFirst*
- *Update*

Create

URL:

/api/v1/{format}/statuses

Method:

POST

Parameters request:

Object< Status >

Parameters response:

Object

Description:

1. name(OBJECT< Status$State >)

2. id(NUMBER)

Delete

URL:

/api/v1/{format}/statuses/{id}

Method:

DELETE

Parameters request:

null

Parameters response:

Object

Description:

1. name(OBJECT< Status$State >)
2. id(NUMBER)

DeleteByIds

URL:

/api/v1/{format}/statuses

Method:

DELETE

Parameters request:

Url parameters: ids

Parameters response:

Array

Description:

1. name(OBJECT< Status$State >)
2. id(NUMBER)

Get

URL:

/api/v1/{format}/statuses/{id}

Method:

GET

Parameters request:

null

Parameters response:

Object

Description:

1. name(OBJECT< Status$State >)
2. id(NUMBER)

GetAll

URL:

/api/v1/{format}/statuses

Method:

GET

Parameters request:

null

Parameters response:

Array

Description:

1. name(OBJECT< Status$State >)
2. id(NUMBER)

SaveAll

URL:

/api/v1/{format}/statuses/saveall

Method:

POST

Parameters request:

Array< Status >

Parameters response:

Array

Description:

1. name(OBJECT< Status$State >)
2. id(NUMBER)

SaveAllAndReturnIds

URL:

/api/v1/{format}/statuses/saveall

Method:

POST

Parameters request:

Url parameters: full

Array< Status >

Parameters response:

Array

Description: ARRAY<NUMBER>

Search

URL:

/api/v1/{format}/statuses/search

Method:

POST

Parameters request:

Array< SearchCriteries$SearchCriteriaResult >

Parameters response:

Array

Description:

1. name(OBJECT< Status$State >)
2. id(NUMBER)

SearchFirst

URL:

/api/v1/{format}/statuses/search/first

Method:

POST

Parameters request:

Array< SearchCriteries$SearchCriteriaResult >

Parameters response:

Object

Description:

1. name(OBJECT< Status$State >)
2. id(NUMBER)

Update

URL:

/api/v1/{format}/statuses/{id}

Method:

PUT

Parameters request:

Object< Status >

Parameters response:

Object

Description:

1. name(OBJECT< Status$State >)
2. id(NUMBER)

WorkRoles

`(implementation of WorkRole entity)`

Provides following method for API calls:

- *Create*
- *Delete*
- *DeleteByIds*
- *Get*
- *GetAll*
- *SaveAll*
- *SaveAllAndReturnIds*
- *Search*
- *SearchFirst*
- *Update*

Create

URL:

/api/v1/{format}/workroles

Method:

POST

Parameters request:

Object< WorkRole >

Parameters response:

Object

Description:

1. name(STRING)
2. id(NUMBER)

Delete

URL:

/api/v1/{format}/workroles/{id}

Method:

DELETE

Parameters request:

null

Parameters response:

Object

Description:

1. name(STRING)
2. id(NUMBER)

DeleteByIds

URL:

/api/v1/{format}/workroles

Method:

DELETE

Parameters request:

Url parameters: ids

Parameters response:

Array

Description:

1. name(STRING)
2. id(NUMBER)

Get

URL:

/api/v1/{format}/workroles/{id}

Method:

GET

Parameters request:

null

Parameters response:

Object

Description:

1. name(STRING)
2. id(NUMBER)

GetAll

URL:

/api/v1/{format}/workroles

Method:

GET

Parameters request:

null

Parameters response:

Array

Description:

1. name(STRING)
2. id(NUMBER)

SaveAll

URL:

/api/v1/{format}/workroles/saveall

Method:

POST

Parameters request:

Array< WorkRole >

Parameters response:

Array

Description:

1. name(STRING)
2. id(NUMBER)

SaveAllAndReturnIds

URL:

/api/v1/{format}/workroles/saveall

Method:

POST

Parameters request:

Url parameters: full

Array< WorkRole >

Parameters response:

Array

Description: ARRAY<NUMBER>

Search

URL:

/api/v1/{format}/workroles/search

Method:

POST

Parameters request:

Array< SearchCriteries$SearchCriteriaResult >

Parameters response:

Array

Description:

1. name(STRING)
2. id(NUMBER)

SearchFirst

URL:

/api/v1/{format}/workroles/search/first

Method:

POST

Parameters request:

Array< SearchCriteries$SearchCriteriaResult >

Parameters response:

Object

Description:

1. name(STRING)
2. id(NUMBER)

Update

URL:

/api/v1/{format}/workroles/{id}

Method:

PUT

Parameters request:

Object< WorkRole >

Parameters response:

Object

Description:

1. name(STRING)
2. id(NUMBER)

ALL (list)

iVIS SDK

iVIS provides SDK for easier api uses.

You think login, token management, api calls etc very hard to implement and will take many hours of your precious time. You will see that is not true. In this chapter you will look how to do routines jobs by few lines of code and iVIS SDK.

And as conclusion in Use cases (Client example) described how create step by step client application.

Get started

- *Add Maven repository*
- *Add Maven dependencies*

iVIS SDK consists of ivis-core, ivis-services and ivis-sdk dependencies. To use them you need include dependencies from Imcode © maven repository to your Maven configuration file (pom.xml).

Add Maven repository

```
<repositories>
    <repository>
        <id>imcode</id>
        <url>http://repo.imcode.com/maven2</url>
        <snapshots>
            <updatePolicy>always</updatePolicy>
            <enabled>true</enabled>
        </snapshots>
        <releases>
            <updatePolicy>always</updatePolicy>
            <enabled>true</enabled>
        </releases>
    </repository>
</repositories>
```

Add Maven dependencies

```
<properties>
    <ivis.version>1.0.0-alpha2</ivis.version>
</properties>

<dependencies>
    <dependency>
        <groupId>com.imcode.ivis</groupId>
        <artifactId>ivis-core</artifactId>
        <version>${ivis.version}</version>
    </dependency>
    <dependency>
        <groupId>com.imcode.ivis</groupId>
        <artifactId>ivis-sdk</artifactId>
        <version>${ivis.version}</version>
    </dependency>
    <dependency>
        <groupId>com.imcode.ivis</groupId>
        <artifactId>ivis-services</artifactId>
        <version>${ivis.version}</version>
    </dependency>
</dependencies>
```

API description

SDK simplify API invocation to easy steps:

1. Get service for entity
2. Invoke methods declared in the service

Examples how to do that you will find at Routines.

Routines

Routines operation vast your time? Don't worry! You will see how to do basic concepts working with iVIS API faster. And as conclusion we will create simple web application which calls iVIS API. So let's started!

Build configuration

As build tool we would use Maven.

In Get started described iVIS dependency management.

iVIS now uses Jackson 2.4, it isn't compatible with latest version that's why also in pom.xml we must include/override Jackson as dependency.

```
<properties>
    <jackson.version>2.4.0</jackson.version>
</properties>

<dependency>
    <groupId>com.fasterxml.jackson.core</groupId>
    <artifactId>jackson-core</artifactId>
    <version>${jackson.version}</version>
</dependency>
<dependency>
    <groupId>com.fasterxml.jackson.core</groupId>
    <artifactId>jackson-databind</artifactId>
    <version>${jackson.version}</version>
</dependency>
<dependency>
    <groupId>com.fasterxml.jackson.core</groupId>
    <artifactId>jackson-annotations</artifactId>
    <version>${jackson.version}</version>
</dependency>
<dependency>
    <groupId>com.fasterxml.jackson.datatype</groupId>
    <artifactId>jackson-datatype-jsr310</artifactId>
    <version>${jackson.version}</version>
</dependency>
```

Configuration properties

Properties

```
api-server-address=http://localhost:8080
client-address=http://localhost:8085/client
client-id=ff11397c-3e3b-4398-80a9-feba203f1928
client-secret=secret
user-authorization-relative-uri=/oauth/authorize
access-token-relative-uri=/oauth/token
ivis-logout-relative-uri=/logout.do
#one month
refresh-token-validity-seconds=2592000
api-relative-uri=/api/v1/json
```

Properties mappings

```
@Component
@ConfigurationProperties(locations = "classpath:client.properties")
public class ClientProperties {

    private String apiServerAddress;
    private String clientAddress;
    private String clientId;
    private String clientSecret;
    private String userAuthorizationRelativeUri;
    private String accessTokenRelativeUri;
    private String ivisLogoutRelativeUri;
    private Integer refreshTokenValiditySeconds;
    private String apiRelativeUri;

    //getters and setters
}
```

Beans configuration

To use services with provided methods we need next beans:

```
private final ClientProperties clientProperties;

@Autowired
public ClientConfiguration(ClientProperties clientProperties) {
    this.clientProperties = clientProperties;
}

@Bean(name = "clientInformation")
public AuthorizationCodeResourceDetails ivisClient() {
    IvisAuthorizationCodeResourceDetails client = new␣
↪IvisAuthorizationCodeResourceDetails();
    String userAuthorizationUrl = clientProperties.getApiServerAddress() +␣
↪clientProperties.getUserAuthorizationRelativeUri();
    String accessTokenUrl = clientProperties.getApiServerAddress() + clientProperties.
↪getAccessTokenRelativeUri();

    client.setClientId(clientProperties.getClientId());
    client.setClientSecret(clientProperties.getClientSecret());
    client.setUserAuthorizationUri(userAuthorizationUrl);
    client.setAccessTokenUri(accessTokenUrl);
```

```
    return client;
}

@Bean
public OAuth2ClientContext clientContext() {
    return new DefaultOAuth2ClientContext();
}

@Bean
public ProxyIvisServiceFactory ivisServiceFactory() {
    String apiUrl = clientProperties.getApiServerAddress() + clientProperties.
↪getApiRelativeUri();
    return new ProxyIvisServiceFactory(apiUrl, clientContext(), ivisClient());
}
```

Mvc configuration

To use services direct in handler methods we need following:

```
@Bean
public HandlerMethodArgumentResolver ivisServiceArgumentResolver() {
    return new IvisServiceArgumentResolver();
}

@Bean
public IvisIdToDomainClassConverter ivisIdToDomainClassConverter() {
    return new IvisIdToDomainClassConverter(conversionServiceFactoryBean().
↪getObject());
}

@Bean
public ConversionServiceFactoryBean conversionServiceFactoryBean() {
    return new ConversionServiceFactoryBean();
}
```

Login

Prerequisites

- Configuration
- Authorization

To login you need:

1. send redirect in context based on HttpServlet (JSP, Spring MVC Controllers, RestControllers etc) to authorizeURI.
2. send POST request with client credentials with code.
3. receive access token.

Let's see how it looks like.

```
@RequestMapping(value = "/", method = RequestMethod.GET)
public ModelAndView welcome(ModelAndView view, WebRequest webRequest, Model model) {
    view.setViewName(START_VIEW_NAME);
    return view;
}

@RequestMapping(value = LOGIN_RELATIVE_URL, method = RequestMethod.GET)
public View login(ModelAndView modelAndView,  WebRequest webRequest) throws␣
↪URISyntaxException, IOException {
    String oAuth2AuthirizationUrl = IvisOAuth2Utils.getOAuth2AuthirizationUrl(client,␣
↪redirectUrl, false);
    return new RedirectView(oAuth2AuthirizationUrl, false);
}

@RequestMapping(value = REDIRECT_RELATIVE_URL, method = RequestMethod.GET)
public View authorizationClientProcess(HttpServletRequest request,
                                       HttpServletResponse response,
                                       WebRequest webRequest,
                                       @SessionAttribute(value = IvisAuthorizedFilter.
↪REQUEST_URI_ATTRIBUTE_NAME, required = false) String protectedResourcesUrl,
                                       @RequestParam("code") String code) throws␣
↪IOException {
    OAuth2AccessToken accessToken = IvisOAuth2Utils.getAccessToken(client, code,␣
↪redirectUrl);
    if (accessToken == null) {
        return new RedirectView(LOGIN_RELATIVE_URL, true);
    }
    IvisOAuth2Utils.setAccessToken(request, accessToken);
    IvisOAuth2Utils.setRefreshTokenAsCokie(response, accessToken.getRefreshToken(),␣
↪clientProperties.getRefreshTokenValiditySeconds());
    String redirect = StringUtils.isEmpty(protectedResourcesUrl) ? "/" :␣
↪protectedResourcesUrl;
    if (redirect.startsWith(request.getContextPath())) {
        redirect = redirect.replace(request.getContextPath(), "");
    }
    return new RedirectView(redirect, true);
}
```

Tokens flow

Prerequisites

- Login

Need say few words how to use tokens flow.

After login user in way described at Login in session placed access token. And also refresh token value from access token object put in cookie.

Important: Cookie has expiration time. It is defined by value “Refresh token validity” in seconds, contact system administrator to know that.

So tokens flow looks like

1. Client app login user (access token -> session, refresh token -> cookie with expiration time).

2. If token is expired (IvisOAuth2Utils.isTokenGood(httpServletRequest) -> exchange refresh token from cookie (cookie key "refreshToken") to access token.
3. If cookie does not exist -> login user again.

Let's see how it looks like.

For last two points let's define handler that will work with unauthorized users.

```
public static final String PATH = "/unauthorized";

private final AuthorizationCodeResourceDetails client;
private final ClientProperties clientProperties;

private final String ivisLogoutUrl;

@Autowired
public UnauthorizedErrorController(
        @Qualifier("clientInformation")
                AuthorizationCodeResourceDetails client,
        ClientProperties clientProperties) {
    this.client = client;
    this.clientProperties = clientProperties;
    this.ivisLogoutUrl = clientProperties.getApiServerAddress() + clientProperties.
↪getIvisLogoutRelativeUri();
}

@RequestMapping(value = PATH)
public View unauthorizedUsers(ModelAndView view,
                              HttpServletRequest request,
                              HttpServletResponse response,
                              @CookieValue(value = "refreshToken", required = false)␣
↪String refreshTokenCookie) throws UnsupportedEncodingException, URISyntaxException {
    logger.info("User unauthorized!");
    response.setStatus(HttpServletResponse.SC_UNAUTHORIZED);
    OAuth2AccessToken accessToken = IvisOAuth2Utils.getAccessToken(client,␣
↪refreshTokenCookie);
    //logout client
    if (accessToken == null) {
        String loginUrl = clientProperties.getClientAddress() +␣
↪IvisAuthorizationController.LOGIN_RELATIVE_URI;
        String redirectUrl = new URIBuilder(ivisLogoutUrl)
                .addParameter("redirect_url", loginUrl)
                .build()
                .toString();
        logger.debug(redirectUrl);
        return new RedirectView(redirectUrl, false);
    }

    IvisOAuth2Utils.setAccessToken(request, accessToken);
    return new RedirectView("/", true);
}
```

As you can see this method also logout user from iVIS.

Note: In Access to protected resources routine described IvisAuthorizedFilter.

If user not logged in, filter intercept access to protected resources by error thrown:

1. org.springframework.security.oauth2.common.exceptions.UnauthorizedUserException with message "Token isn't good".
2. org.springframework.security.access.AccessDeniedException with message "Token is good, but roles aren't".

Access to protected resources

Prerequisites

- Tokens flow

You can limit access to specific urls, or some code areas on JSP page. iVIS provides SDK in both cases.

Both variants has optional parameter roles (String). "roles" it is comma separated list of roles which gives user access to protected resources.

Filter

Beans config

```
@Bean(name = "ivisAuthorizedFilter")
public Filter ivisAuthorizedFilter() {
    IvisAuthorizedFilter ivisAuthorizedFilter = new IvisAuthorizedFilter();
    return ivisAuthorizedFilter;
}

@Bean
public FilterRegistrationBean ivisAuthorizedFilterRegistration() {
    FilterRegistrationBean registration = new FilterRegistrationBean();
    registration.setFilter(ivisAuthorizedFilter());
    registration.addUrlPatterns("/services/classes/*");
    registration.setName("ivisAuthorizedFilter");
    registration.setOrder(1);
    return registration;
}
```

Tag

To know if user login on JSP you can invoke special tag <ivis:authorized> with optional parameter role.

```
<%@taglib prefix="ivis" uri="ivis.sdk" %>

...

<ivis:authorized>
    Information for authorized users
</ivis:authorized>

...

<ivis:authorized roles="ROLE_ADMIN">
    Information for authorized users in admin role
</ivis:authorized>
```

Important: You can use this two cases if you have permission to use method getCurrent user. After invoking Filter or tag with parameter role in session persisted user object ("loggedInUser" key to parameter).

Use api methods

Important: You can use API method only if admin check permission for that method for client and user (result it is a cross of permissions).

First you need get iVIS service factory from request.

```
HttpServletRequest request = ... ; // get HttpServletRequest
IvisServiceFactory ivisServiceFactory = IvisOAuth2Utils.getServiceFactory(request);
```

Note: If service factory invokes first time it wil create service factory in user session context.

Next uses exist IvisServiceFactory instance need get service for specific entity.

```
//for example you need ApplicationService
ApplicationService applicationService = ivisServiceFactory.
↪getService(ApplicationService.class);
```

And then just invoke API methods like Java methods.

```
List<Application> allApplications = applicationService.findAll();
```

All API services described at ivis-services module.

`Class diagram` of all interfaces with methods.

Note: Registered bean IvisServiceArgumentResolver provide you possibility use SDK in handler method next way. Also IvisIdToDomainClassConverter allow use id definition instead whole object.

```
@RequestMapping(value = "/classes/save", method = RequestMethod.POST)
public ModelAndView saveSchoolClass(@ModelAttribute("schoolClass") SchoolClass␣
↪schoolClass,
                                    SchoolClassService classService) {
    //in this object pupils and school fields are objects with only one non null␣
↪property id
    classService.save(schoolClass);
    view.setViewName("school_classes/created");
    return view;
}
```

Use cases

Developing client

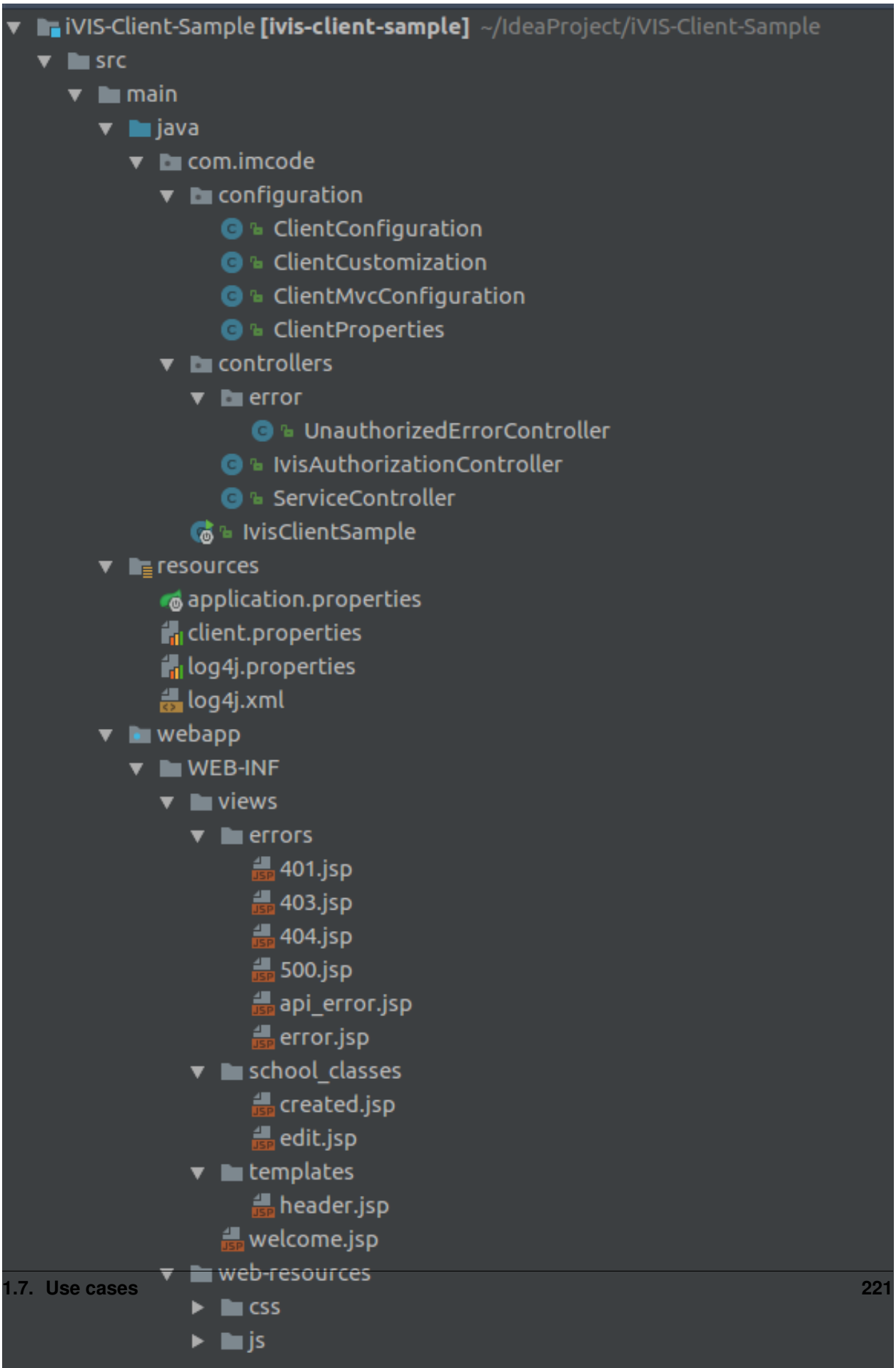
This is the guide how correct and right create (also configure) simplest secure client application for iVIS including all tips from iVIS SDK chapter.

Complete client example placed at Github repository (https://github.com/imCodePartnerAB/iVIS-Client-Sample).

Project structure

Spring is a most popular framework for Java Web apps, so all examples will be based on Java configuration with Spring Boot that configured for Spring MVC.

With this tools project structure looks next way:


iVIS-Client-Sample [ivis-client-sample] ~/IdeaProject/iVIS-Client-Sample
src
main
java
com.imcode
configuration
ClientConfiguration
ClientCustomization
ClientMvcConfiguration
ClientProperties
controllers
error
UnauthorizedErrorController
IvisAuthorizationController
ServiceController
IvisClientSample
resources
application.properties
client.properties
log4j.properties
log4j.xml
webapp
WEB-INF
views
errors
401.jsp
403.jsp
404.jsp
500.jsp
api_error.jsp
error.jsp
school_classes
created.jsp
edit.jsp
templates
header.jsp
welcome.jsp
web-resources
css
js

Maven configuration

A thorough explanation you can find at Get started and Build configuration.

Client properties

Of client properties respond two files

- ClientProperties.java;
- client.properties

Client configuration

Client configuration represented at this file.

Injecting client properties:

```
private final ClientProperties clientProperties;

@Autowired
public ClientConfiguration(ClientProperties clientProperties) {
    this.clientProperties = clientProperties;
}
```

Current iVIS client:

```
@Bean(name = "clientInformation")
public AuthorizationCodeResourceDetails ivisClient() {
    IvisAuthorizationCodeResourceDetails client = new␣
↪IvisAuthorizationCodeResourceDetails();
    String userAuthorizationUrl = clientProperties.getApiServerAddress() +␣
↪clientProperties.getUserAuthorizationRelativeUri();
    String accessTokenUrl = clientProperties.getApiServerAddress() + clientProperties.
↪getAccessTokenRelativeUri();

    client.setClientId(clientProperties.getClientId());
    client.setClientSecret(clientProperties.getClientSecret());
    client.setUserAuthorizationUri(userAuthorizationUrl);
    client.setAccessTokenUri(accessTokenUrl);

    return client;
}
```

Client context for service invocation:

```
@Bean
public OAuth2ClientContext clientContext() {
    return new DefaultOAuth2ClientContext();
}
```

Service factory for creating service classes:

```
@Bean
public ProxyIvisServiceFactory ivisServiceFactory() {
    String apiUrl = clientProperties.getApiServerAddress() + clientProperties.
↪getApiRelativeUri();
```

```
    return new ProxyIvisServiceFactory(apiUrl, clientContext(), ivisClient());
}
```

Client MVC configuration

Client MVC configuration represented at this file.

When invoking ModelAndView method setViewName next configuration concatenates it with viewPrefix and viewSuffix:

```
@Value("${spring.mvc.view.prefix}")
private String viewPrefix;

@Value("${spring.mvc.view.suffix}")
private String viewSuffix;

@Bean
public InternalResourceViewResolver viewResolver() {
    InternalResourceViewResolver resolver = new InternalResourceViewResolver();
    resolver.setPrefix(viewPrefix);
    resolver.setSuffix(viewSuffix);
    return resolver;
}
```

Methods for creating beans through ivisAuthorizedFilter() and ivisAuthorizedFilterRegistration() described here.

Initialize service class in handler methods:

```
@Bean
public HandlerMethodArgumentResolver ivisServiceArgumentResolver() {
    return new IvisServiceArgumentResolver();
}

@Override
public void addArgumentResolvers(List<HandlerMethodArgumentResolver>␣
↪argumentResolvers) {
    argumentResolvers.add(ivisServiceArgumentResolver());
}
```

Converter that used for init Ivis entity object instead id property:

```
@Override
public void addFormatters(FormatterRegistry registry) {
    registry.addConverter(ivisIdToDomainClassConverter());
}

@Bean
public IvisIdToDomainClassConverter ivisIdToDomainClassConverter() {
    return new IvisIdToDomainClassConverter(conversionServiceFactoryBean().
↪getObject());
}

@Bean
public ConversionServiceFactoryBean conversionServiceFactoryBean() {
    return new ConversionServiceFactoryBean();
}
```

Custom error views:

```
@Bean
public ServerProperties errorHandling() {
    return new ClientCustomization();
}
```

Enable default configuration:

```
@Override
public void configureDefaultServletHandling(
        DefaultServletHandlerConfigurer configurer) {
    configurer.enable();
}
```

Static content management:

```
@Override
public void addResourceHandlers(ResourceHandlerRegistry registry) {
    registry.addResourceHandler("/resources/**")
            .addResourceLocations("/WEB-INF/web-resources/");
}
```

Client customization

Client customization represented at this file.

Purpose of this configuration is create custom error views on which servlet container will be redirect when exception will occurred.

The main thing is that need process two types of errors from IvisAuthorizedFilter:

1. org.springframework.security.oauth2.common.exceptions.UnauthorizedUserException.
2. org.springframework.security.access.AccessDeniedException.

Ivis authorization controller

Ivis authorization controller represented at this file.

This class purpose and logic described here.

Results

Run client application

Run iVIS server from tutorial.

See also:

You need configure permissions for client sample application. This guide will be helpful.

Application must be started on port 8080.

Clone project from Github (https://github.com/imCodePartnerAB/iVIS-Client-Sample.git).

Execute Maven package

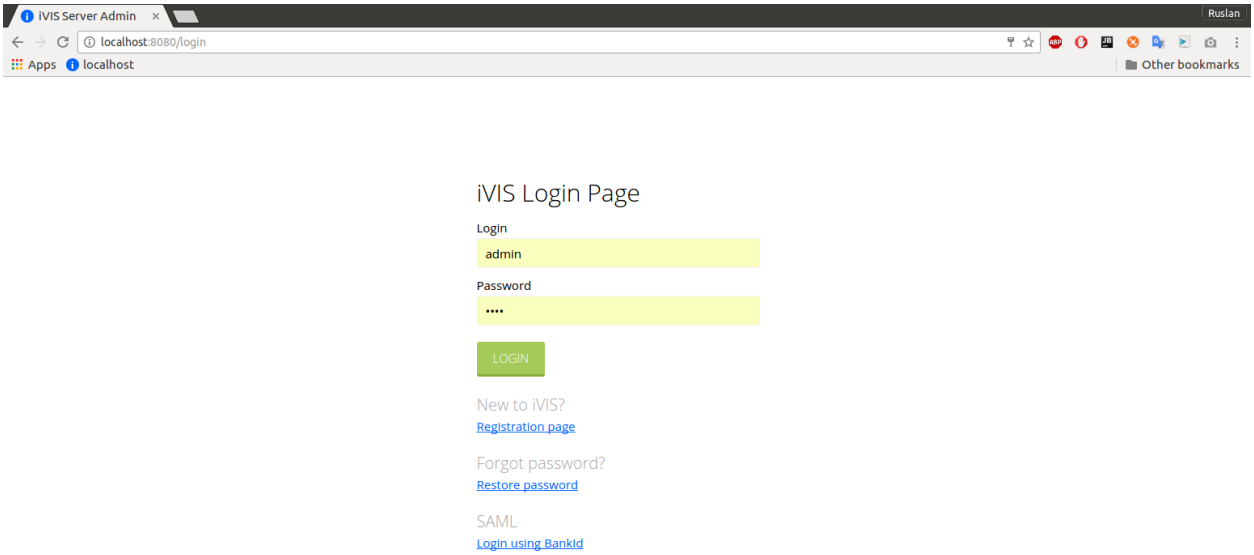
```
mvn package spring-boot:run
```

Application must be started on port 8085 with context path "/client".

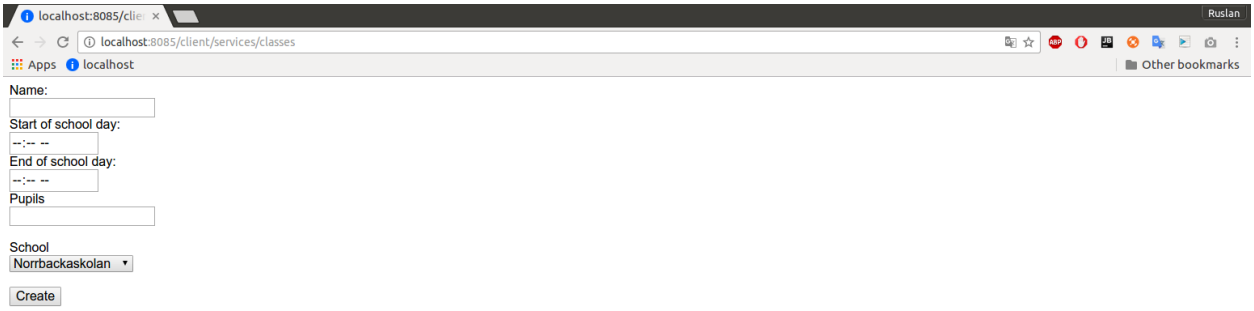
Check all

At url http://localhost:8085/client we see:

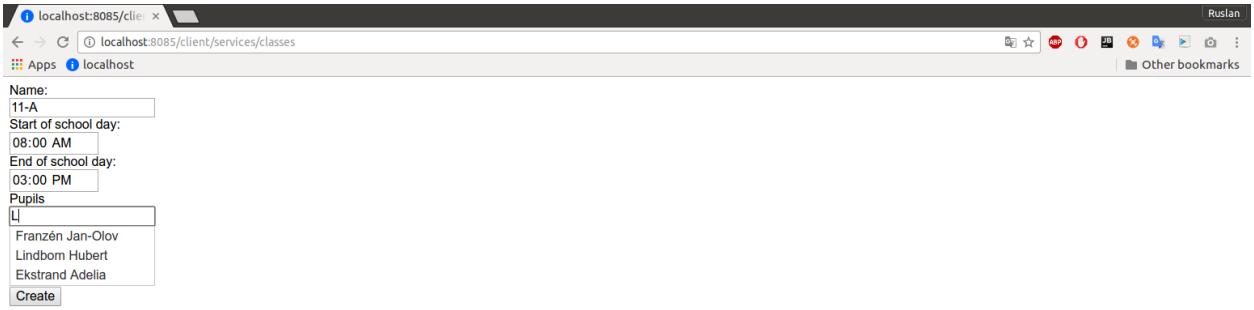
Click Login.

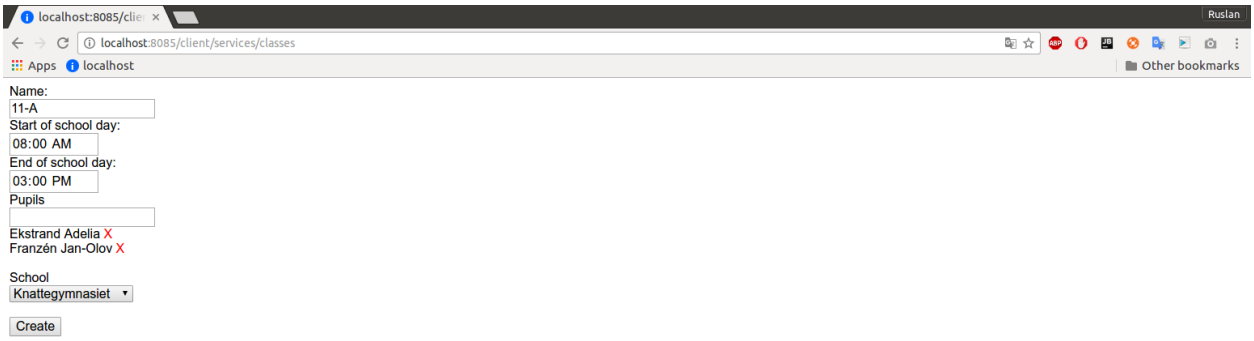


Click Create school class.

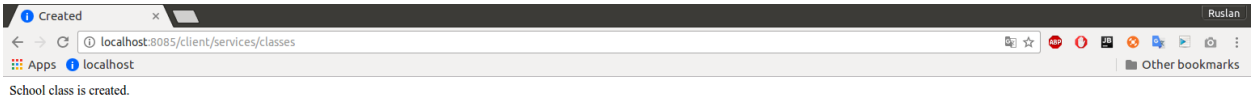


Fill test data.





Click Create.



Security test

Open new incognito window.

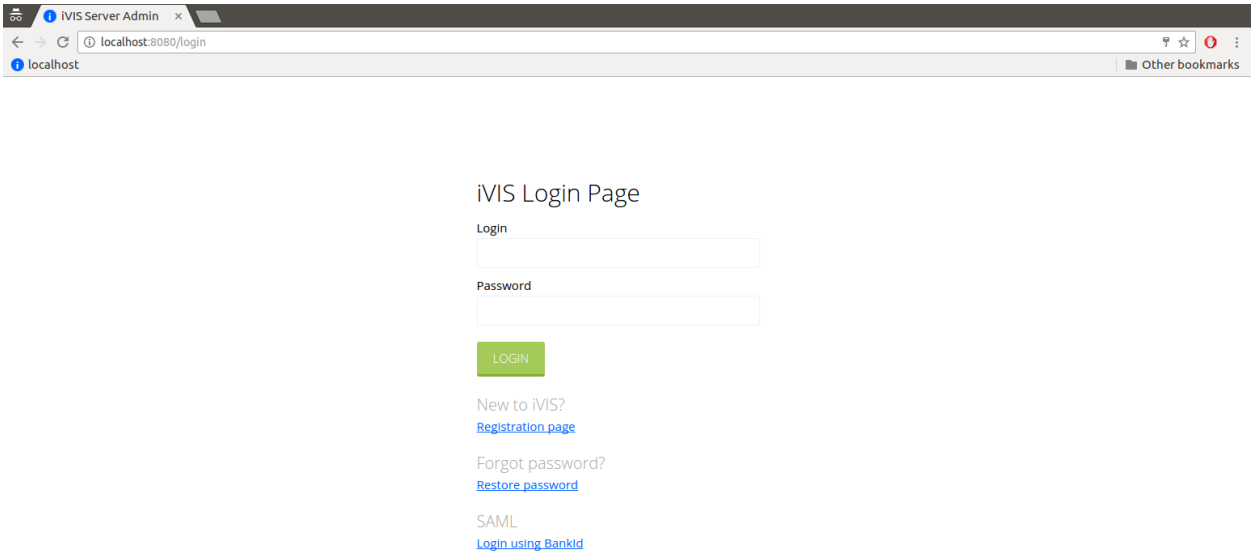
Type http://localhost:8085/client/services/classes.

In logs you get following

```
SEVERE: Servlet.service() for servlet [dispatcherServlet] in context with path [/
↪client] threw exception
error="unauthorized_user", error_description="Token isn't good."
    at imcode.services.filter.IvisAuthorizedFilter.doFilter(IvisAuthorizedFilter.
↪java:42)
    at org.apache.catalina.core.ApplicationFilterChain.
↪internalDoFilter(ApplicationFilterChain.java:192)
    at org.apache.catalina.core.ApplicationFilterChain.
↪doFilter(ApplicationFilterChain.java:165)
    at org.springframework.web.filter.RequestContextFilter.
↪doFilterInternal(RequestContextFilter.java:99)
    at org.springframework.web.filter.OncePerRequestFilter.
↪doFilter(OncePerRequestFilter.java:107)
    at org.apache.catalina.core.ApplicationFilterChain.
↪internalDoFilter(ApplicationFilterChain.java:192)
    at org.apache.catalina.core.ApplicationFilterChain.
↪doFilter(ApplicationFilterChain.java:165)
    at org.springframework.web.filter.HttpPutFormContentFilter.
↪doFilterInternal(HttpPutFormContentFilter.java:89)
    at org.springframework.web.filter.OncePerRequestFilter.
↪doFilter(OncePerRequestFilter.java:107)
    at org.apache.catalina.core.ApplicationFilterChain.
↪internalDoFilter(ApplicationFilterChain.java:192)
    at org.apache.catalina.core.ApplicationFilterChain.
↪doFilter(ApplicationFilterChain.java:165)
```

```
    at org.springframework.web.filter.HiddenHttpMethodFilter.
↪doFilterInternal(HiddenHttpMethodFilter.java:77)
    at org.springframework.web.filter.OncePerRequestFilter.
↪doFilter(OncePerRequestFilter.java:107)
    at org.apache.catalina.core.ApplicationFilterChain.
↪internalDoFilter(ApplicationFilterChain.java:192)
    at org.apache.catalina.core.ApplicationFilterChain.
↪doFilter(ApplicationFilterChain.java:165)
    at org.springframework.web.filter.CharacterEncodingFilter.
↪doFilterInternal(CharacterEncodingFilter.java:197)
    at org.springframework.web.filter.OncePerRequestFilter.
↪doFilter(OncePerRequestFilter.java:107)
    at org.apache.catalina.core.ApplicationFilterChain.
↪internalDoFilter(ApplicationFilterChain.java:192)
    at org.apache.catalina.core.ApplicationFilterChain.
↪doFilter(ApplicationFilterChain.java:165)
    at org.apache.catalina.core.StandardWrapperValve.invoke(StandardWrapperValve.
↪java:198)
    at org.apache.catalina.core.StandardContextValve.invoke(StandardContextValve.
↪java:108)
    at org.apache.catalina.authenticator.AuthenticatorBase.invoke(AuthenticatorBase.
↪java:472)
    at org.apache.catalina.core.StandardHostValve.invoke(StandardHostValve.java:140)
    at org.apache.catalina.valves.ErrorReportValve.invoke(ErrorReportValve.java:79)
    at org.apache.catalina.core.StandardEngineValve.invoke(StandardEngineValve.
↪java:87)
    at org.apache.catalina.connector.CoyoteAdapter.service(CoyoteAdapter.java:349)
    at org.apache.coyote.http11.Http11Processor.service(Http11Processor.java:784)
    at org.apache.coyote.AbstractProcessorLight.process(AbstractProcessorLight.
↪java:66)
    at org.apache.coyote.AbstractProtocol$ConnectionHandler.process(AbstractProtocol.
↪java:802)
    at org.apache.tomcat.util.net.NioEndpoint$SocketProcessor.doRun(NioEndpoint.
↪java:1410)
    at org.apache.tomcat.util.net.SocketProcessorBase.run(SocketProcessorBase.java:49)
    at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1142)
    at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:617)
    at org.apache.tomcat.util.threads.TaskThread$WrappingRunnable.run(TaskThread.
↪java:61)
    at java.lang.Thread.run(Thread.java:745)

INFO : com.imcode.controllers.IvisAuthorizationController - User unauthorized!
DEBUG: com.imcode.controllers.IvisAuthorizationController - http://localhost:8080/
↪logout.do?redirect_url=http%3A%2F%2Flocalhost%3A8085%2Fclient%2Flogin
```

And you will be redirected to iVIS login page.



Okey let's see our welcome page (http://localhost:8085/client/).

You can't see link "Create school class" because it placed under protected resource:

```
<ivis:authorized>
    <a href="${pageContext.servletContext.contextPath}/services/classes">Create␣
↪school class</a>
</ivis:authorized>
```

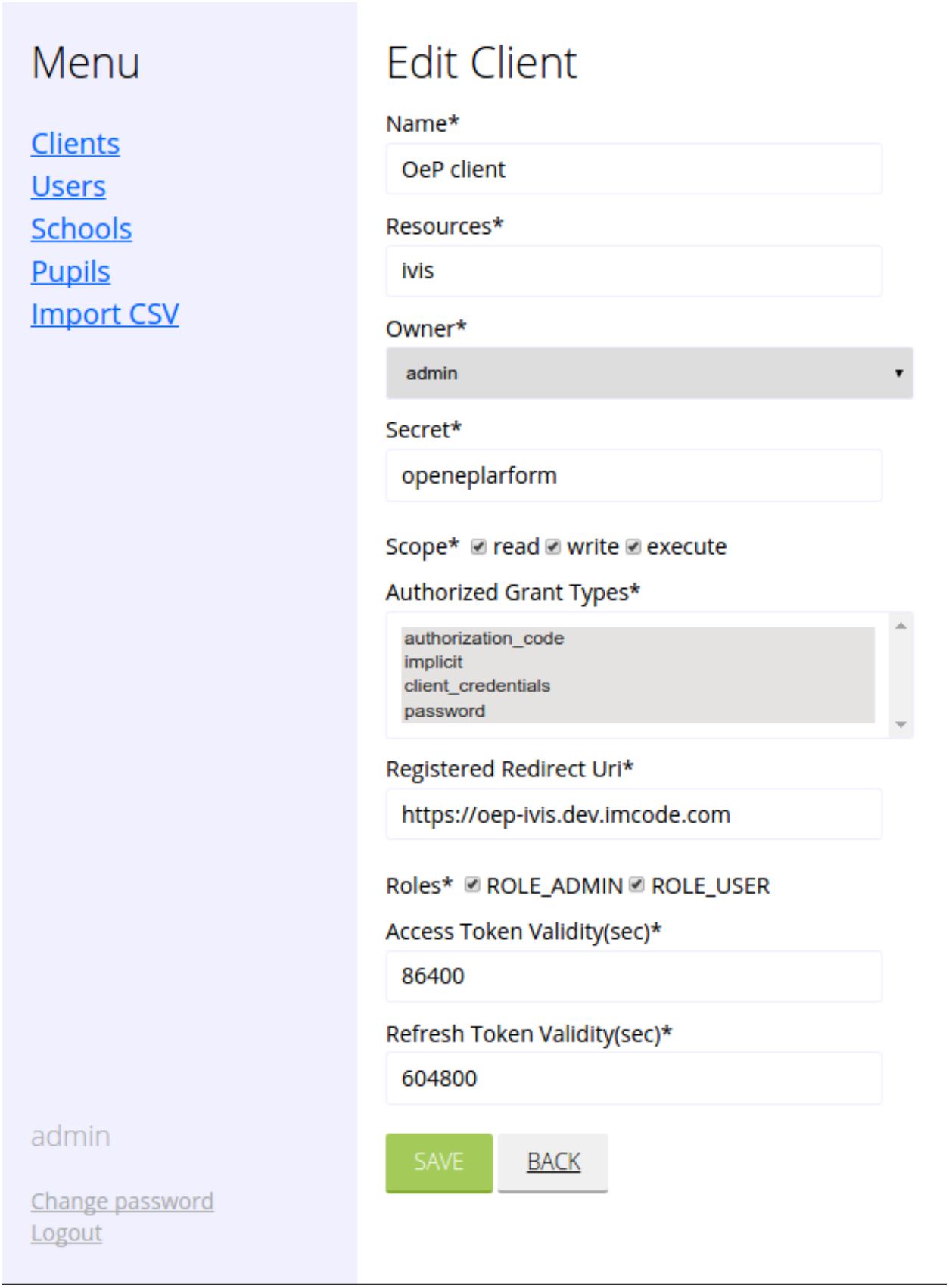
School Transport

This demo consists of 3 parts: iVIS Server, OeP client and imCMS client. OeP client is used to collect applications for school transport from pupil guardians for their pupils. imCMS client allows school administration to review that applications and approve or decline them.

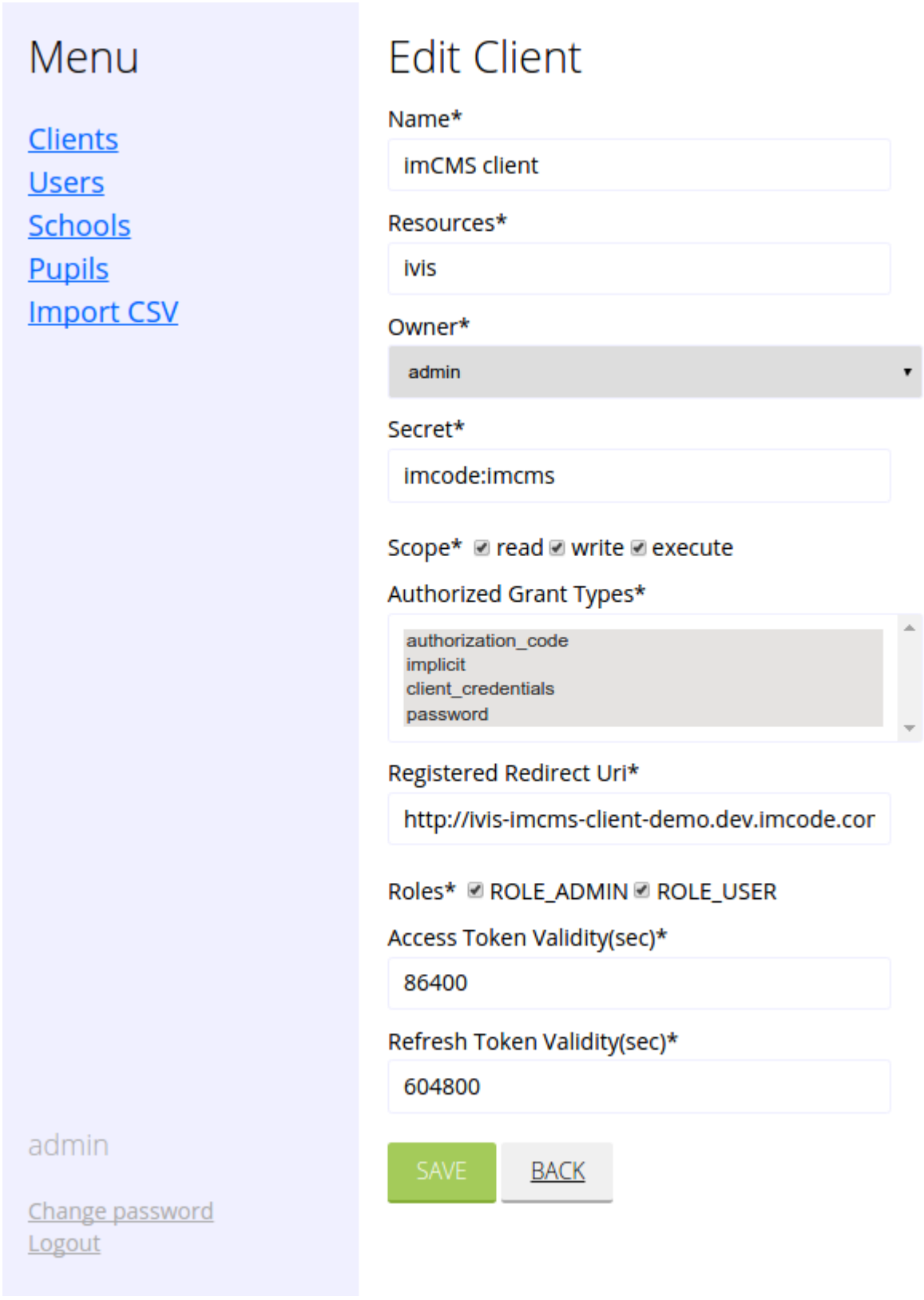
- *iVIS Server*
  - *Configure OeP client*
  - *Configure imCMS client*
- *OeP client*
  - *Create application*
- *imCMS client*
  - *View application*
  - *Edit application*

iVIS Server

Configure OeP client

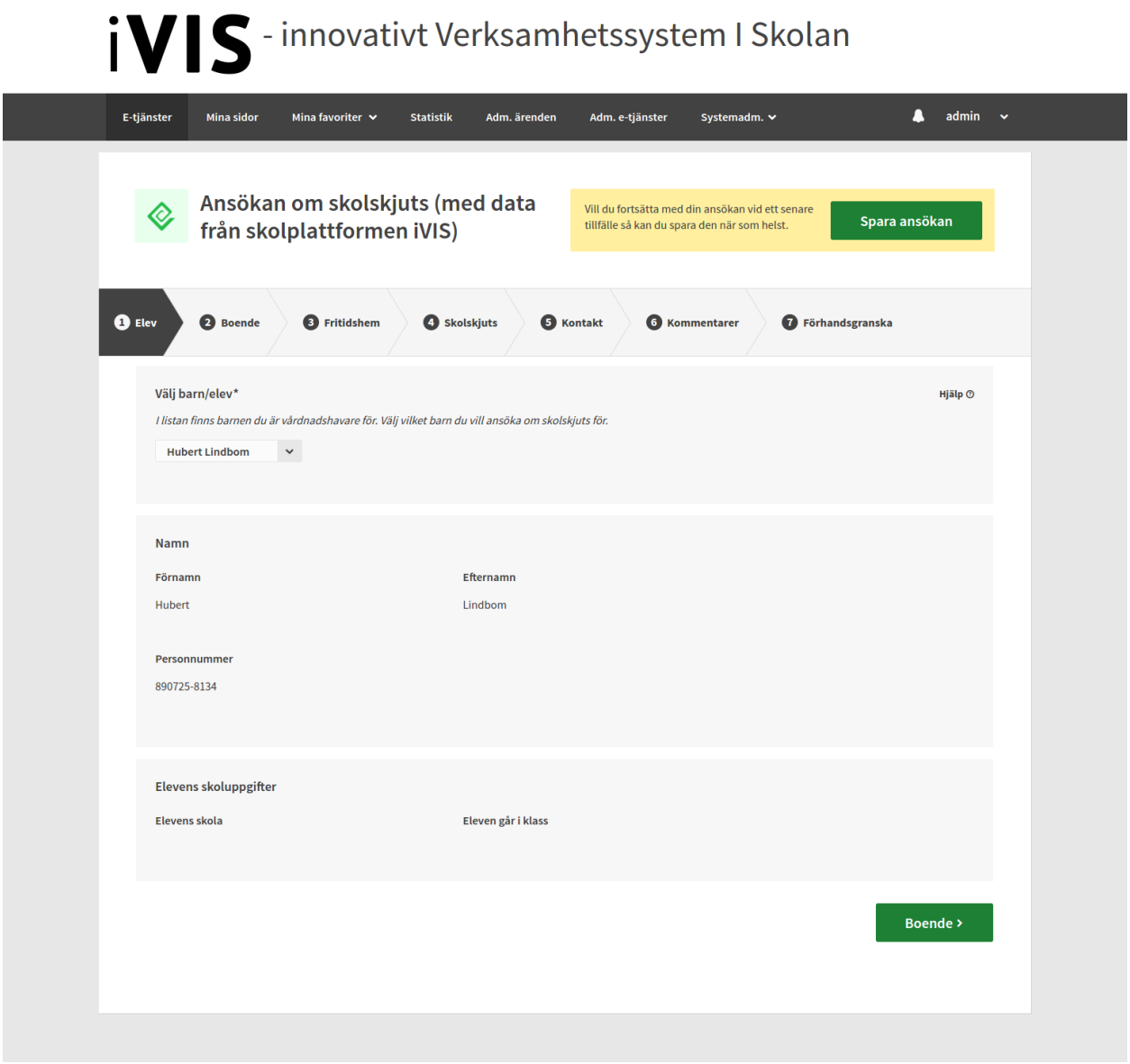


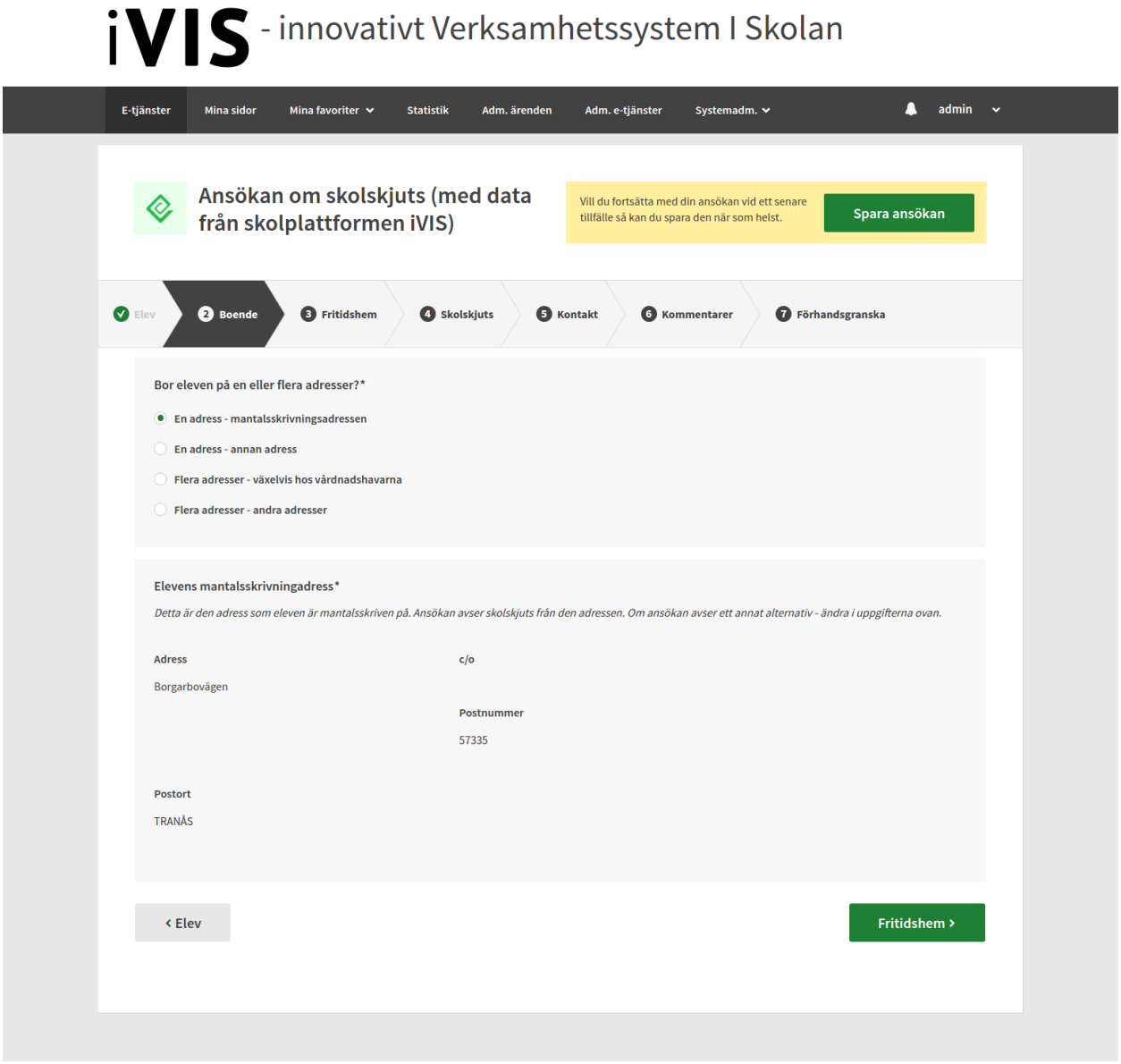
Configure imCMS client

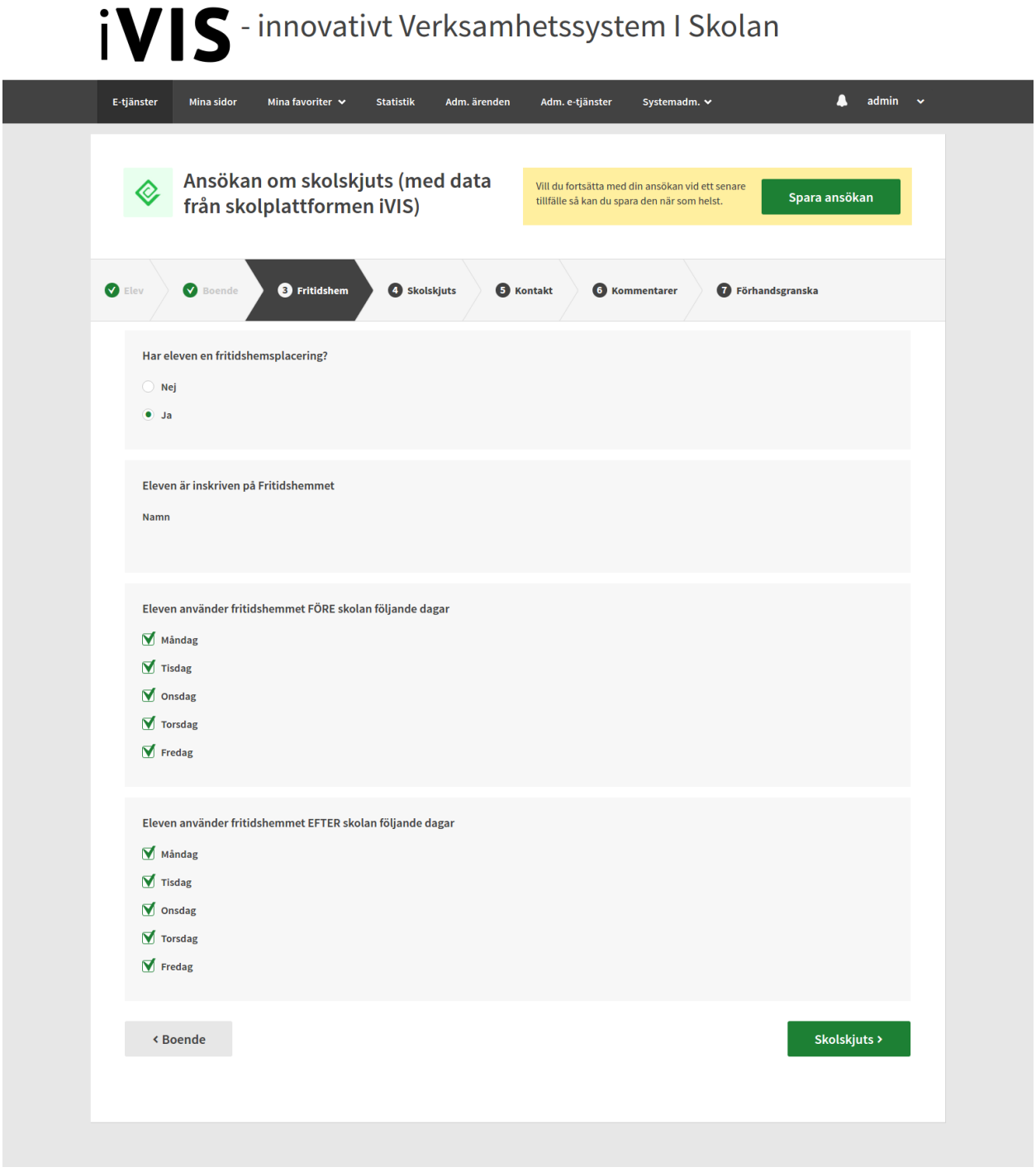


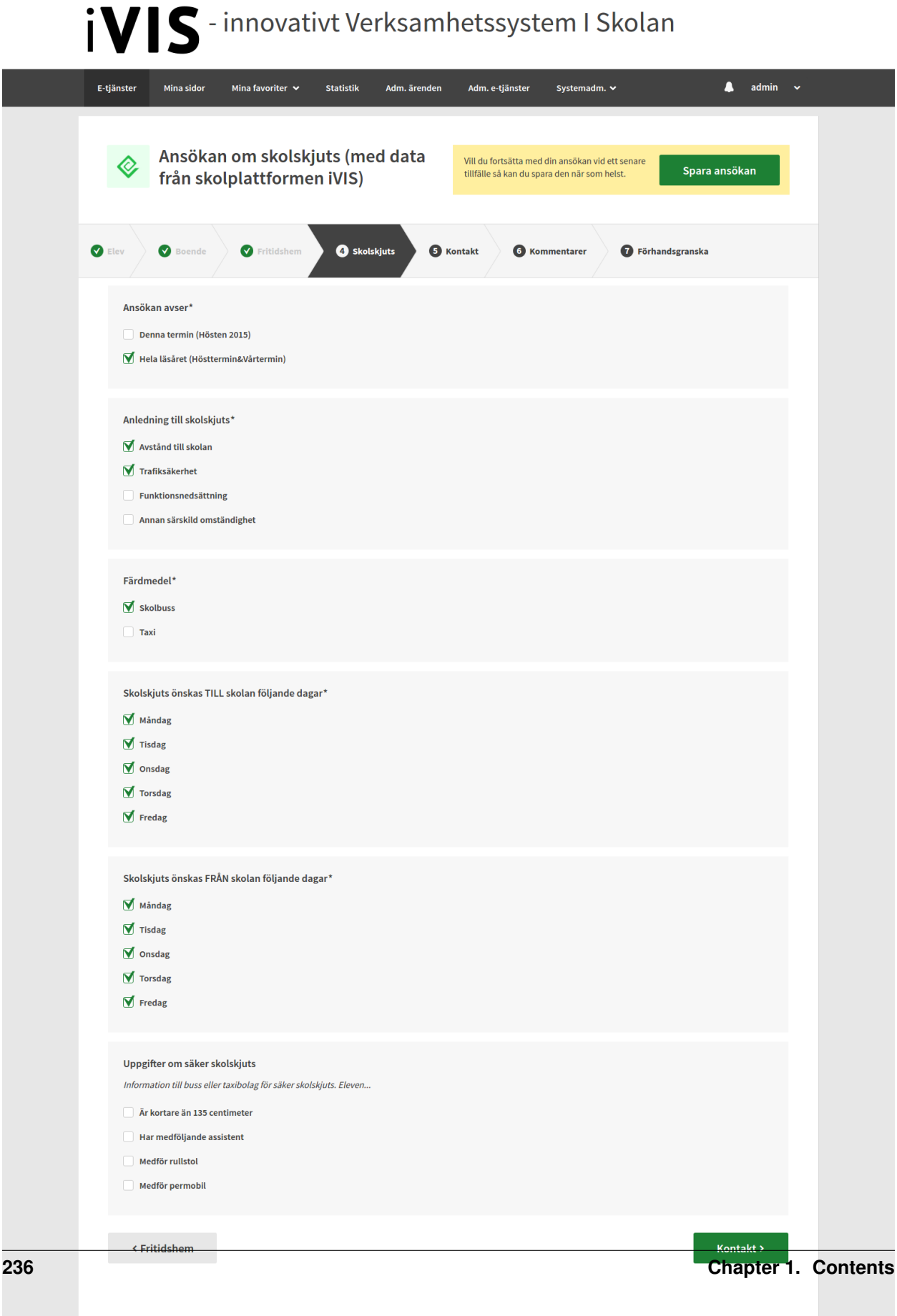
OeP client

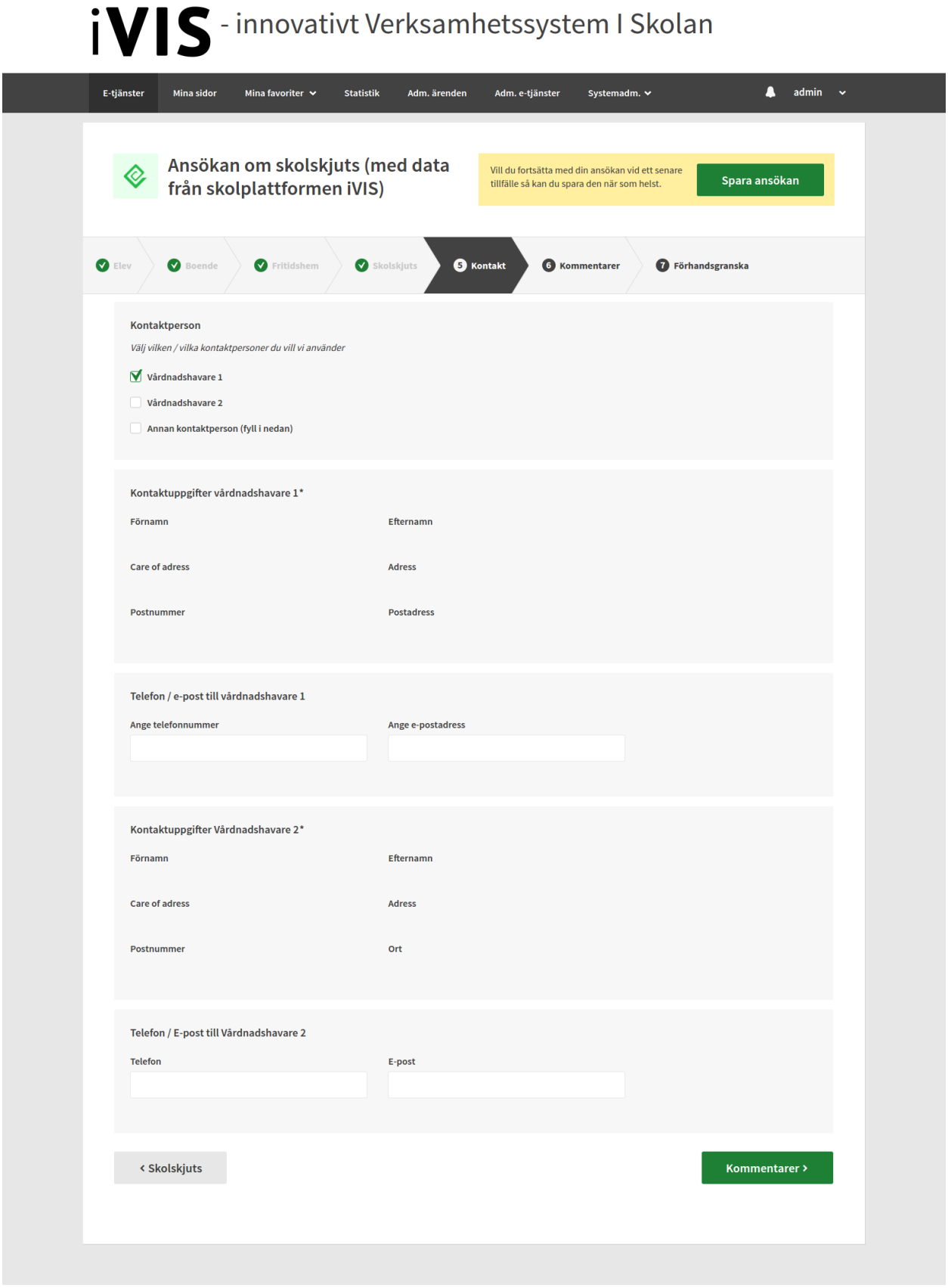
Create application

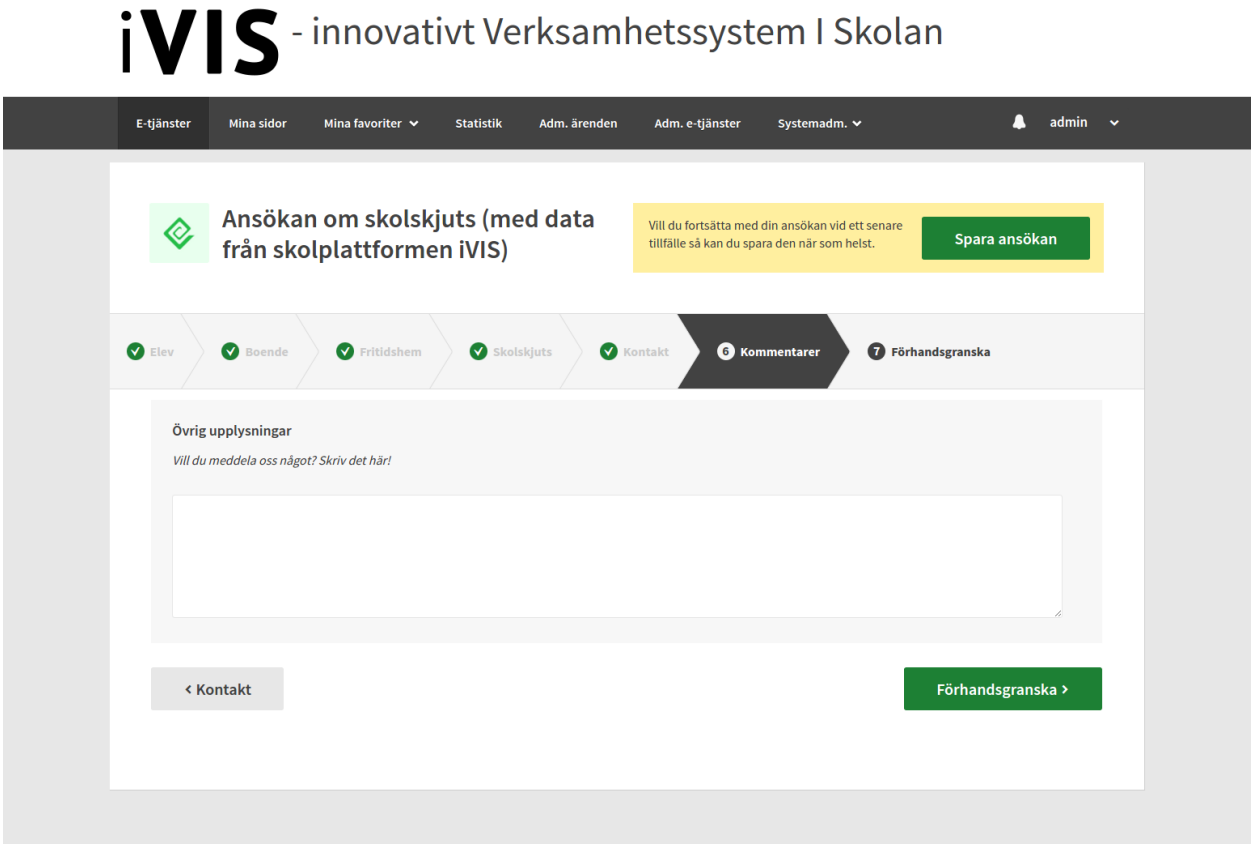


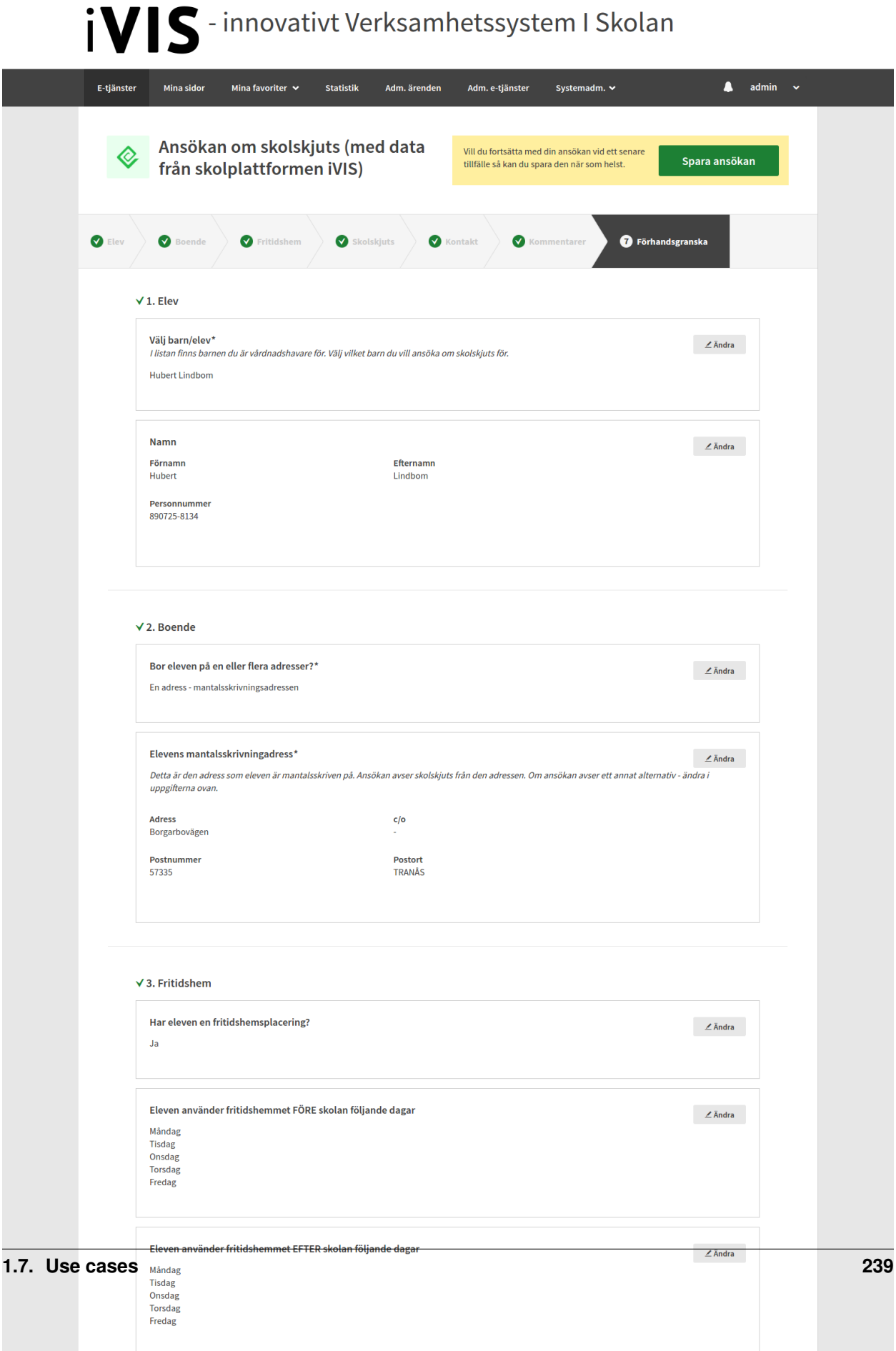

iVIS - innovativt Verksamhetssystem I Skolan
E-tjänster
Mina sidor
Mina favoriter
Statistik
Adm. ärenden
Adm. e-tjänster
Systemadm.
admin
Ansökan om skolskjuts (med data från skolplattformen iVIS)
Vill du fortsätta med din ansökan vid ett senare tillfälle så kan du spara den när som helst.
Spara ansökan
Elev
2 Boende
3 Fritidshem
4 Skolskjuts
5 Kontakt
6 Kommentarer
7 Förhandsgranska
Bor eleven på en eller flera adresser?*
En adress - mantalsskrivningsadressen
En adress - annan adress
Flera adresser - växelvis hos vårdnadshavarna
Flera adresser - andra adresser
Elevens mantalsskrivningadress*
Detta är den adress som eleven är mantalsskriven på. Ansökan avser skolskjuts från den adressen. Om ansökan avser ett annat alternativ - ändra i uppgifterna ovan.
Adress
Borgarbovägen
c/o
Postnummer
57335
Postort
TRANÅS
< Elev
Fritidshem >


iVIS - innovativt Verksamhetssystem I Skolan
E-tjänster
Mina sidor
Mina favoriter
Statistik
Adm. ärenden
Adm. e-tjänster
Systemadm.
admin
Ansökan om skolskjuts (med data från skolplattformen iVIS)
Vill du fortsätta med din ansökan vid ett senare tillfälle så kan du spara den när som helst.
Spara ansökan
Elev
Boende
3 Fritidshem
4 Skolskjuts
5 Kontakt
6 Kommentarer
7 Förhandsgranska
Har eleven en fritidshemsplacering?
Nej
Ja
Eleven är inskriven på Fritidshemmet
Namn
Eleven använder fritidshemmet FÖRE skolan följande dagar
Måndag
Tisdag
Onsdag
Torsdag
Fredag
Eleven använder fritidshemmet EFTER skolan följande dagar
Måndag
Tisdag
Onsdag
Torsdag
Fredag
‹ Boende
Skolskjuts ›


iVIS - innovativt Verksamhetssystem I Skolan
E-tjänster
Mina sidor
Mina favoriter
Statistik
Adm. ärenden
Adm. e-tjänster
Systemadm.
admin
Ansökan om skolskjuts (med data från skolplattformen iVIS)
Vill du fortsätta med din ansökan vid ett senare tillfälle så kan du spara den när som helst.
Spara ansökan
Elev
Boende
Fritidshem
4 Skolskjuts
5 Kontakt
6 Kommentarer
7 Förhandsgranska
Ansökan avser*
Denna termin (Hösten 2015)
Hela läsåret (Hösttermin&Vårtermin)
Anledning till skolskjuts*
Avstånd till skolan
Trafiksäkerhet
Funktionsnedsättning
Annan särskild omständighet
Färdmedel*
Skolbuss
Taxi
Skolskjuts önskas TILL skolan följande dagar*
Måndag
Tisdag
Onsdag
Torsdag
Fredag
Skolskjuts önskas FRÅN skolan följande dagar*
Måndag
Tisdag
Onsdag
Torsdag
Fredag
Uppgifter om säker skolskjuts
Information till buss eller taxibolag för säker skolskjuts. Eleven...
Är kortare än 135 centimeter
Har medföljande assistent
Medför rullstol
Medför permobil
< Fritidshem
Kontakt >

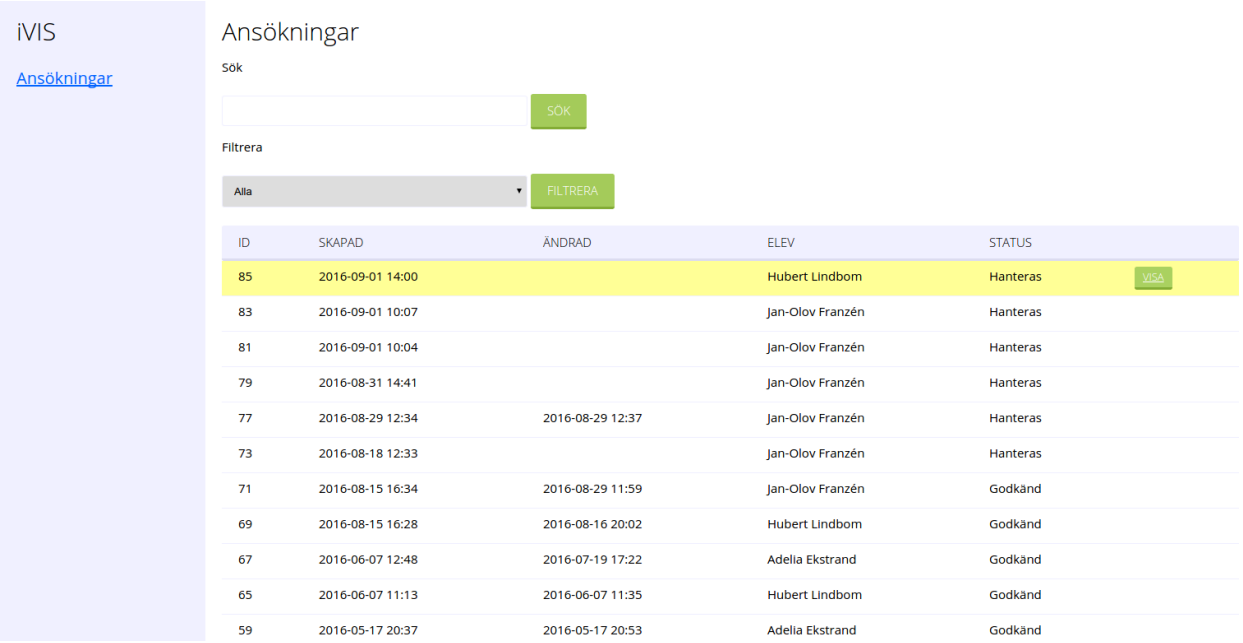

iVIS - innovativt Verksamhetssystem I Skolan
E-tjänster
Mina sidor
Mina favoriter
Statistik
Adm. ärenden
Adm. e-tjänster
Systemadm.
admin
Ansökan om skolskjuts (med data från skolplattformen iVIS)
Vill du fortsätta med din ansökan vid ett senare tillfälle så kan du spara den när som helst.
Spara ansökan
Elev
Boende
Fritidshem
Skolskjuts
5 Kontakt
6 Kommentarer
7 Förhandsgranska
Kontaktperson
Välj vilken / vilka kontaktpersoner du vill vi använder
Vårdnadshavare 1
Vårdnadshavare 2
Annan kontaktperson (fyll i nedan)
Kontaktuppgifter vårdnadshavare 1*
Förnamn
Efternamn
Care of adress
Adress
Postnummer
Postadress
Telefon / e-post till vårdnadshavare 1
Ange telefonnummer
Ange e-postadress
Kontaktuppgifter Vårdnadshavare 2*
Förnamn
Efternamn
Care of adress
Adress
Postnummer
Ort
Telefon / E-post till Vårdnadshavare 2
Telefon
E-post
‹ Skolskjuts
Kommentarer ›

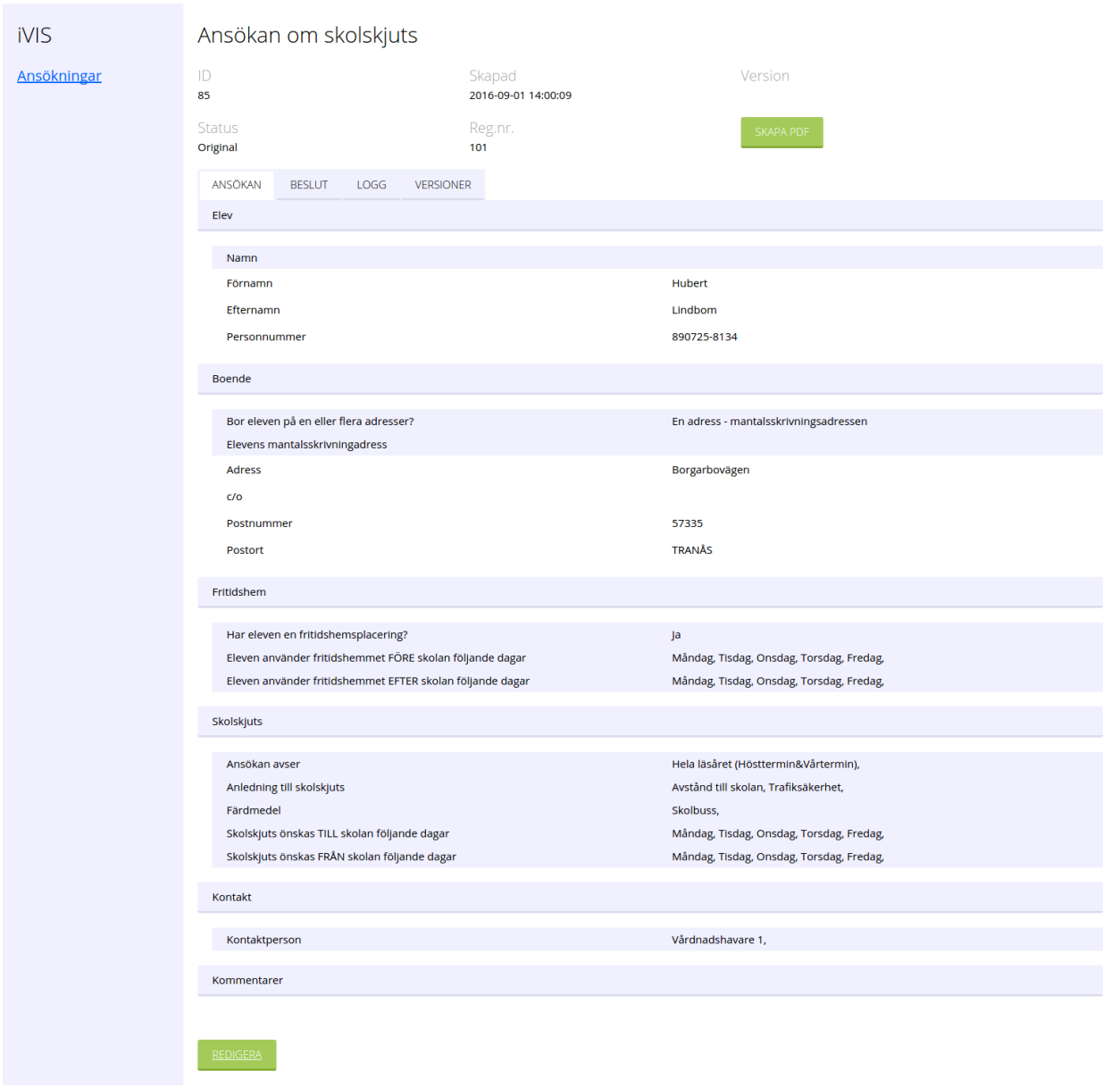

iVIS - innovativt Verksamhetssystem I Skolan
E-tjänster
Mina sidor
Mina favoriter
Statistik
Adm. ärenden
Adm. e-tjänster
Systemadm.
admin
Ansökan om skolskjuts (med data från skolplattformen iVIS)
Vill du fortsätta med din ansökan vid ett senare tillfälle så kan du spara den när som helst.
Spara ansökan
Elev
Boende
Fritidshem
Skolskjuts
Kontakt
6 Kommentarer
7 Förhandsgranska
Övrig upplysningar
Vill du meddela oss något? Skriv det här!
‹ Kontakt
Förhandsgranska ›


iVIS - innovativt Verksamhetssystem I Skolan
E-tjänster
Mina sidor
Mina favoriter
Statistik
Adm. ärenden
Adm. e-tjänster
Systemadm.
admin
Ansökan om skolskjuts (med data från skolplattformen iVIS)
Vill du fortsätta med din ansökan vid ett senare tillfälle så kan du spara den när som helst.
Spara ansökan
Elev
Boende
Fritidshem
Skolskjuts
Kontakt
Kommentarer
7 Förhandsgranska
1. Elev
Välj barn/elev*
I listan finns barnen du är vårdnadshavare för. Välj vilket barn du vill ansöka om skolskjuts för.
Hubert Lindbom
Ändra
Namn
Förnamn
Hubert
Efternamn
Lindbom
Personnummer
890725-8134
2. Boende
Bor eleven på en eller flera adresser?*
En adress - mantalsskrivningsadressen
Elevens mantalsskrivningadress*
Detta är den adress som eleven är mantalsskriven på. Ansökan avser skolskjuts från den adressen. Om ansökan avser ett annat alternativ - ändra i uppgifterna ovan.
Adress
Borgarbovägen
c/o
-
Postnummer
57335
Postort
TRANÅS
3. Fritidshem
Har eleven en fritidshemsplacering?
Ja
Eleven använder fritidshemmet FÖRE skolan följande dagar
Måndag
Tisdag
Onsdag
Torsdag
Fredag
Eleven använder fritidshemmet EFTER skolan följande dagar
Måndag
Tisdag
Onsdag
Torsdag
Fredag

imCMS client

View application




iVIS
Ansökningar
Ansökan om skolskjuts
ID
85
Skapad
2016-09-01 14:00:09
Version
Status
Original
Reg.nr.
101
SKAPA PDF
ANSÖKAN
BESLUT
LOGG
VERSIONER
Elev
Namn
Förnamn
Hubert
Efternamn
Lindbom
Personnummer
890725-8134
Boende
Bor eleven på en eller flera adresser?
En adress - mantalsskrivningsadressen
Elevens mantalsskrivningadress
Adress
Borgarbovägen
c/o
Postnummer
57335
Postort
TRANÅS
Fritidshem
Har eleven en fritidshemsplacering?
Ja
Eleven använder fritidshemmet FÖRE skolan följande dagar
Måndag, Tisdag, Onsdag, Torsdag, Fredag,
Eleven använder fritidshemmet EFTER skolan följande dagar
Måndag, Tisdag, Onsdag, Torsdag, Fredag,
Skolskjuts
Ansökan avser
Hela läsåret (Hösttermin&Vårtermin),
Anledning till skolskjuts
Avstånd till skolan, Trafiksäkerhet,
Färdmedel
Skolbuss,
Skolskjuts önskas TILL skolan följande dagar
Måndag, Tisdag, Onsdag, Torsdag, Fredag,
Skolskjuts önskas FRÅN skolan följande dagar
Måndag, Tisdag, Onsdag, Torsdag, Fredag,
Kontakt
Kontaktperson
Vårdnadshavare 1,
Kommentarer
REDIGERA

Edit application

iVIS

Ansökningar

Ansökan om skolskjuts

ID
85

Skapad
2016-09-01 14:00:09

Senast ändrad

Status
Hanteras

Reg.nr.
101

ANSÖKAN

Elev

Namn

Förnamn: Hubert

Efternamn: Lindbom

Personnummer: 890725-8134

Elevens skoluppgifter

Elevens skola:

Eleven går i klass:

Eleven är inskriven på fritidshem

Boende

Bor eleven på en eller flera adresser?

- (•) En adress - mantalsskrivningsadressen
- ( ) En adress - annan adress
- ( ) Flera adresser - växelvis hos vårdnadshavarna
- ( ) Flera adresser - andra adresser

Elevens mantalsskrivningadress

Adress: Borgarbovägen

c/o:

Postnummer: 57335

Postort: TRANÅS

Annan bostadsadress

Adress:

Care of adress:

Postnummer:

Postort:

Mantalsskrivningsadress för inloggad vårdnadshavare

Förnamn:

Efternamn:

Adress:

Care of adress:

Postnummer:

Postort:

Hur lång period bor eleven hos dig som fyller i denna ansökan ?

När sker skifte till den andra vårdnadshavaren?

Mantalsskrivningsadress till den andra vårdnadshavaren

Förnamn:

Efternamn:

Adress:

Care of adress:

Incidents

This demo consists of 2 parts: iVIS Server and imCMS client. imCMS client is used to collect information about incidents in the school and to process that incidents.

Validation

- *Client side*
- *Server side*

Client side

To make client side validation need define rules for special form (see jQuery Validation Plugin).

Example of how to do it you can find at restore_password_validation.js

```
$('#restorePasswordFormEmail').validate({

    rules: {

        email: {
            required: true,
            email: true,
            checkNotUnique: "/restore_password/emailunique"
        }

    },

    messages: {

        email: {
            required: "Email is required",
            email: "Email not valid",
            checkNotUnique: "User with this email does not exist"
        }

    }

});
```

Server side

Server side validation based on class GeneralValidator. It uses interface Validator.

For API object validation need override method getFieldsConstraints() for example in IncidentRestControllerImpl

```
@Override
protected Map<String, Map<GeneralValidator.Constraint, String>>␣
↪getFieldsConstraints() {
    Map<String, Map<GeneralValidator.Constraint, String>> fieldsConstraints = super.
↪getFieldsConstraints();

    GeneralValidator.buildField(fieldsConstraints, "title",
            new AbstractMap.SimpleEntry<>(GeneralValidator.Constraint.NOT_NULL_OR_
↪EMPTY, null),
```

```
            new AbstractMap.SimpleEntry<>(GeneralValidator.Constraint.MIN, "4")
    );

    GeneralValidator.buildField(fieldsConstraints, "description",
            new AbstractMap.SimpleEntry<>(GeneralValidator.Constraint.NOT_NULL_OR_
↪EMPTY, null),
            new AbstractMap.SimpleEntry<>(GeneralValidator.Constraint.MIN, "4")
    );

    GeneralValidator.buildField(fieldsConstraints, "categories",
            new AbstractMap.SimpleEntry<>(GeneralValidator.Constraint.NOT_NULL_OR_
↪EMPTY, null)
    );

    GeneralValidator.buildField(fieldsConstraints, "pupils",
            new AbstractMap.SimpleEntry<>(GeneralValidator.Constraint.NOT_NULL_OR_
↪EMPTY, null)
    );

    GeneralValidator.buildField(fieldsConstraints, "priority",
            new AbstractMap.SimpleEntry<>(GeneralValidator.Constraint.NOT_NULL_OR_
↪EMPTY, null)
    );

    return fieldsConstraints;
}
```

Example how create validation from AdminController for form parameters.

```
Map<String, Map<GeneralValidator.Constraint, String>> constraints = new HashMap<>();

GeneralValidator.buildField(constraints, "password",
        new SimpleEntry<>(GeneralValidator.Constraint.NOT_NULL_OR_EMPTY, null),
        new SimpleEntry<>(GeneralValidator.Constraint.MIN, "4"),
        new SimpleEntry<>(GeneralValidator.Constraint.MATCH_WITH, "confirmPassword")
);

GeneralValidator.buildField(constraints, "person.firstName",
        new SimpleEntry<>(GeneralValidator.Constraint.NOT_NULL_OR_EMPTY, null),
        new SimpleEntry<>(GeneralValidator.Constraint.MIN, "4")
);

GeneralValidator.buildField(constraints, "person.lastName",
        new SimpleEntry<>(GeneralValidator.Constraint.NOT_NULL_OR_EMPTY, null),
        new SimpleEntry<>(GeneralValidator.Constraint.MIN, "4")
);

GeneralValidator.buildField(constraints, "person.emails",
        new SimpleEntry<>(GeneralValidator.Constraint.NOT_NULL_OR_EMPTY, null),
        new SimpleEntry<>(GeneralValidator.Constraint.REGEX, GeneralValidator.EMAIL_
↪PATTERN)
);

GeneralValidator.buildField(constraints, "person.phones",
        new SimpleEntry<>(GeneralValidator.Constraint.NOT_NULL_OR_EMPTY, null),
        new SimpleEntry<>(GeneralValidator.Constraint.MIN, "8")
);

```

```
264
265 if (userService.findByUsername(user.getUsername()) != null) {
266     bindingResult.reject(null, "username not unique");
267 }
268
269 if (userService.findByEmail(email) != null) {
270     bindingResult.reject(null, "email not unique");
271 }
272
273 new GeneralValidator(constraints).invoke(user, bindingResult);
```

Index

E