Object Model

Introduction

Class diagram represent API architecture, objects (classes) structure, services (methods) and their relations.
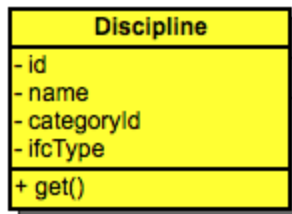
This diagram reflects:

- resulting JSON objects structure, which will be sent to client upon AJAX request with defined services
- class methods (services), which can be used by the client with HTTP requests (see Bimplus Services for detailed description).

Class methods here are not the real class methods, but list of HTTP methods, applicable to particular object. Thus from the scheme you can see which objects are read only and which ones can be modified / updated by the user.

Object name means that you can use it lower cased as a path (route) to resource / collection. Fields define which properties will be transferred to client as resulting JSON string. If you provide an element id in collection, particular object in the collection will be addressed and you will get its detailed information.

Example:

Object:

is read only, has method get(), implemented as web service. This means that by a client it can be used as follows:

```
https://api-stage.bimplus.net/v2/bimplus/disciplines/
```

Default HTTP method is get, so we mustn't specify it explicitly in the request. Server responses with a string, which automatically will be converted into JSON object:

```
{
    "0f106af0-a919-44c5-b211-15bd5ef620b6": {
        id: "0f106af0-a919-44c5-b211-15bd5ef620b6",
        name: "<LOCALIZATION_STRING_ID>",
        category: "/1/",
        ifctype: null
    },
    {...},
    ...
}
```
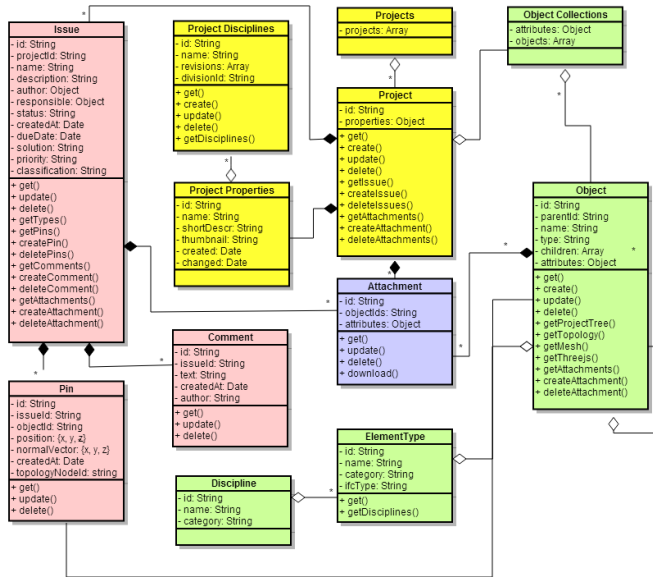
Class diagrams (v2, draft)

General

Issue
- id: String
- projectId: String
- name: String
- description: String
- author: Object
- responsible: Object
- status: String
- createdAt: Date
- dueDate: Date
- solution: String
- priority: String
- classification: String
+ get()
+ update()
+ delete()
+ getTypes()
+ getPins()
+ createPin()
+ deletePins()
+ getComments()
+ createComment()
+ deleteComment()
+ getAttachments()
+ createAttachment()
+ deleteAttachment()

Project Disciplines
- id: String
- name: String
- revisions: Array
- divisionId: String
+ get()
+ create()
+ update()
+ delete()
+ getDisciplines()

Project Properties
- id: String
- name: String
- shortDescr: String
- thumbnail: String
- created: Date
- changed: Date

Projects
- projects: Array

Project
- id: String
- properties: Object
+ get()
+ create()
+ update()
+ delete()
+ getIssue()
+ createIssue()
+ deleteIssues()
+ getAttachments()
+ createAttachment()
+ deleteAttachments()

Object Collections
- attributes: Object
- objects: Array

Object
- id: String
- parentId: String
- name: String
- type: String
- children: Array
- attributes: Object
+ get()
+ create()
+ update()
+ delete()
+ getProjectTree()
+ getTopology()
+ getMesh()
+ getThreejs()
+ getAttachments()
+ createAttachment()
+ deleteAttachment()

Attachment
- id: String
- objectIds: String
- attributes: Object
+ get()
+ update()
+ delete()
+ download()

Comment
- id: String
- issueId: String
- text: String
- createdAt: Date
- author: String
+ get()
+ update()
+ delete()

Pin
- id: String
- issueId: String
- objectId: String
- position: {x, y, z}
- normalVector: {x, y, z}
- createdAt: Date
- topologyNodeId: string
+ get()
+ update()
+ delete()

ElementType
- id: String
- name: String
- category: String
- ifcType: String
+ get()
+ getDisciplines()

Discipline
- id: String
- name: String
- category: String

Fig. 1. Web API Object Model and Services diagram

Objects and its Attributes (v2, draft)

Object Collections
- attributes: Object
- objects: Array

Threejs Atributes
- elementCount: int
- viewBox: Object
- colors: Array
+ get()

Viewbox
- x: Double
- y: Double
- z: Double
- width: Double
- height: Double
- depth: Double

Object
- id: String
- parentId: String
- name: String
- type: String
- children: Array
- attributes: Object
+ get()
+ create()
+ update()
+ delete()
+ getProjectTree()
+ getTopology()
+ getMesh()
+ getThreejs()
+ getAttachments()
+ createAttachment()
+ deleteAttachment()

Geometry
- faces: Array
- vertices: Array
- normals: Array
- uvs: Array
- materials: Array
- colors: Array
- metadata: Object

Color
- value (rgba)

Normal
<coordinate>[3]: Double

UV
- <coordinate>[3]: Double

Vertex
<coordinate>[3]

Material
- id
- name
- opacity
...

Face
<type bitmask>
<vertex index>[3..4]

Metadata
formatVersion: 3

Attributes Group
<name>: {<attrs>}

Attribute
- <geometry>
- <key>: <value>

Geometry.meshblob
<blob>

BLOB, f.e.
ZIP archive

Geometry.mesh
- colors: Array
- vertices: Array
- faces: Array

colors - plain array of decimal RGBA values
vertices - plain array of coordinates in x, y, z sequence
faces - plain array with faces list in sequence: N, v0, v1, v2, ...
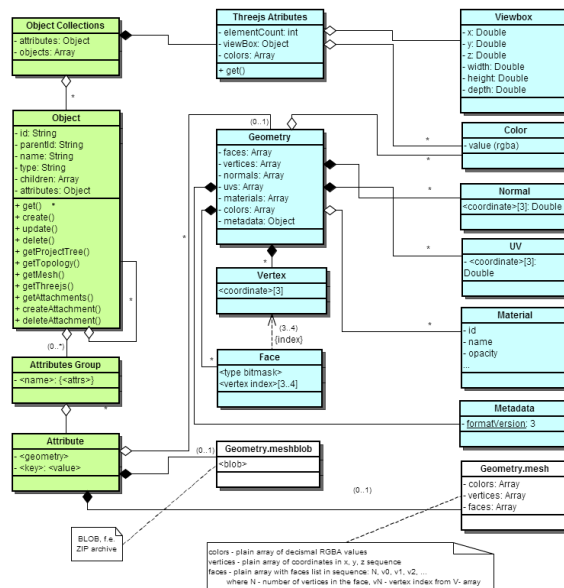 where N - number of vertices in the face, vN - vertex index from V- array

Fig. 1. Web API Object Model and Services diagram