DELIVERABLE

Project Acronym: **SDI4Apps**

Grant Agreement number: **621129**

Project Full Title: **Uptake of Open Geographic Information Through Innovative Services Based on Linked Data**

D4.3.1 ADVANCED TOOLS API – RELEASE 1.0

Revision no. 02

Authors: Stein Runar Bergheim (AVINET)
Eirik Csak Knutsen (AVINET)
Tor Gunnar Øverli (AVINET)

Project co-funded by the European Commission within the ICT Policy Support Programme		
Dissemination Level		
P	Public	X
C	Confidential, only for members of the consortium and the Commission Services	

REVISION HISTORY

Revision	Date	Author	Organisation	Description
01	25/03/2016	S.R. Bergheim	AVINET	First draft based on DoW, D4.2
02	28/03/2016	S.R. Bergheim, Eirik Csak Knutsen, Tor Gunnar Øverli	AVINET	Second draft

Statement of originality:

This deliverable contains original unpublished work except where clearly indicated otherwise. Acknowledgement of previously published material and of the work of others has been made through appropriate citation, quotation or both.

Disclaimer:

Views expressed in this document are those of the individuals, partners or the consortium and do not represent the opinion of the Community.

TABLE OF CONTENTS

LISTINGS

List of Figures

List of Code Fragments

ACRONYMS

Table 1: Table of acronyms used in document

Acronym	Explanation
AJAX	Asynchronous JavaScript and XML
API	Application Programming Interface
CORS	Cross-origin resource sharing
DoW	SDI4Apps Description of Work, Annex I to the Grant Agreement
HTTP	Hypertext Transfer Protocol
Http GET	Request supported by the HTTP protocol
Http POST	Request supported by the HTTP protocol
JS	JavaScript
JSON	JavaScript Object Notation
NPM	Node Package Manager
OGC	Open Geospatial Consortium
OSM	OpenStreetMap
SDI	Spatial Data Infrastructure
SFS	Simple Features Specification
SME	Small and medium-sized enterprises
SQL	Structured Query Language
WKT	Well-Known Text

EXECUTIVE SUMMARY

This document describes how the 1.0 release of the s4a.js client-side JavaScript library has been implemented. It is based on D4.2 – the high-level API design document, D3.3.1 – "Open API Design & First Release" and indirectly on the results of D3.1 "Architecture Concept" and D3.2.1 "Enablers Deployment – First Release".

As planned in the design specification, the library has formally been given the name "s4a.js". Instead of implementing large numbers of 'local' utility methods, s4a.js instead depends on jQuery (for cross-browser JavaScript compatibility), D3.js (for data visualization), Crossfilter and DC charts (for dynamic charting), OpenLayers (for map framework) and Cordova (for cross-device compatibility in mobile applications).

The library includes two types of client functionality:

1. stand-alone components that can be used independently;
2. components that depend on a running instance of the SDI4Apps OpenAPI

The partnership is conscious about keeping doors open for exploiting project results in multiple ways, thus the library is implemented in a modular fashion, so that stand-alone components may be split into individual components and contributed to mainstream libraries.

The advanced tools API consists of five modules:

1. A map module provides methods and widgets to add interactive maps, map tools and data registered with an OpenAPI instance to HTML5 applications.
2. An information retrieval module provides methods and widgets to implements keyword, free-text, spatial and faceted search to HTML5 applications.
3. An advanced visualization module provides methods and widgets to implement a variety of chart and map types – as well as coordinated views composed of charts and maps to HTML5 applications.
4. A mobile module that offers offline editing and map browsing capabilities to mixed connectivity mobile applications. The mobile module offers a solution to this via the introduction of cached offline base maps and feature editing layers to HTML5 applications.
5. The final module of the advanced tools API is the analytics and modelling module that permits end-users invoking complex multi-variable analysis processes on the server side from within HTML5 applications.

Starting with release 1.0 (this deliverable), the s4a.js library will be validated by the end-user applications developed as part of the SDI4Apps pilots. It is important to note that the API will not supply application specific or use-case specific functionality to the pilots – it will provide reusable components that have relevance to recurring use cases.

1. PREFACE

This deliverable forms part of the software deliverables of the SDI4Apps project and must be seen in context of the other parallel/connected tasks in work package 3 and 4 respectively.

Particularly, the advanced tools API is dependent on modules and web services exposed by the server-side OpenAPI. To illustrate this dependency, the Information Retrieval module is relies on communicating with web services in order to return meaningful results on the client.

The software deliverables, i.e. D4.3.1 (the software release described in this deliverable) and D4.3.2 feed into the technical test reports D4.5.1 (submitted at the same time as this deliverable) and D4.5.2 at the end of the project. The results from the testing and evaluation activities, in turn feeds back into the improvement cycle of the software development life cycle – and will be guiding for the further development and refinement of the library in the final year of the project.

It is worth noting that the complete functionality of the Open API is not planned to be available until Month 27. For this reason, additional functionality of the advanced tools API will continue to be made available in incremental releases between the milestone releases 1.0 (month 24) and 2.0 (month 36).

Similarly, while evolving the code of the advanced tools API, it may be necessary to make amendments or enhancements to the Open API services in order to achieve the desired target benefits.

This is the first milestone release of the advanced tools API and the work is now entering into a phase where the emphasis will be on implementing functionality exposed by the API in SDI4Apps pilot applications.

Furthermore, the emphasis will from now on also be shifted from pure software development towards identifying sustainability strategies for the resulting software products and 'packaging' the software in a manner suitable for post-project exploitation.

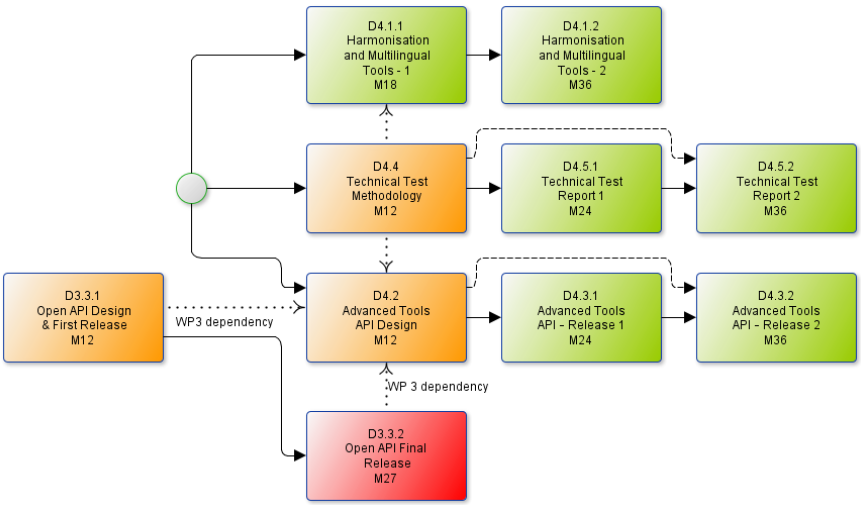

Figure 1: The relationship between D4.3.1 and other SDI4Apps deliverables

2. METHODOLOGY

This chapter describes the methodology that has been followed in translating the user requirements and system requirements specifications into a complete software product.

2.1 Specification

The first release of the client-side JavaScript library 's4a.js' is based on the system requirements specification in deliverable D4.2 Advanced Tools API Design (Bergheim, et al., 2014). Furthermore, the implementation of the API has drawn on the works 'JavaScript Design Patterns' (Osmani, 2015) and the JavaScript code documentation framework 'JSDoc 3.0' (JSDoc 3 contributors, 2016)

2.2 Development

The development of the JavaScript library has adopted good practice from contemporary mainstream JavaScript API library. Since s4a.js is dependent on OpenLayers 3 and jQuery, it has been natural to take the practices adopted by these widely used Open Source frameworks as a starting point for the development.

2.2.1 GitHub

The development has taken place on the 'social' coding platform GitHub, one of the most widely used source control and development management systems on the Internet. The URL of the public repository is https://github.com/SDI4Apps/s4a.js.

GitHub provides a technical framework – but does not dictate the development process itself; therefore, the following workflow process has been established to secure optimal benefits from the use of GitHub.

Two main branches have been created:

- Master - It is not permitted to commit code directly to Master
- Develop - Although permitted, no commits shall normally be made directly to Develop

To contribute code to s4a.js the following steps should be made:

- All work shall be carried out on short-lived feature branches, i.e. the implementation of the map module took place on a branch called feature/map-helper
- Once the work was completed, a pull request was created to merge the branch feature/map-helper into develop.
- Prior to merging pull requests created by a single author, the committed code in the pull request should be reviewed by one peer.
- Upon receiving 'all clear' from the peer reviewer, the code is merged into the 'develop branch' where the most recent version of the complete API sits at all times.
- Prior to merging, all conflicts should be resolved offline by the author by rebasing his or her feature branch towards the most recent version of the develop branch.
- Once a feature branch has been merged, it can and should be deleted

Code is always deployed from the master branch

- The develop branch is used to test the integration of all code committed to the repository
- Many developers work simultaneously and it is necessary to verify that the pieces of the puzzle fits together before the API library is deployed
- Once integration tests have passed, the develop branch can be merged into the master branch and the API may be deployed to a content delivery network – or anywhere of the user's choosing

2.2.2 Development methodology

GitHub as a development framework is particularly well suited for use in agile development techniques such as Scrum (Schwaber, 2004).

The implementation of s4a.js has followed the Scrum methodology with minor adaptions to allow for the fact that development teams are decentralized and not located in the same physical space.

All system requirements have been broken down into manageable component tasks and added to a backlog. Tasks have been distributed into sprints – but instead of having 'one sprint' for all developers, each development team (each partner) has been at liberty to define his/her own sprints to comply with the working mode of the individual teams.

Likewise, the daily Scrum has not been applied across the partnership but individual to each team. Overall coordination has been secured through touch points at project events and through periodic online conferences where the progress of the individual tasks have been verified.

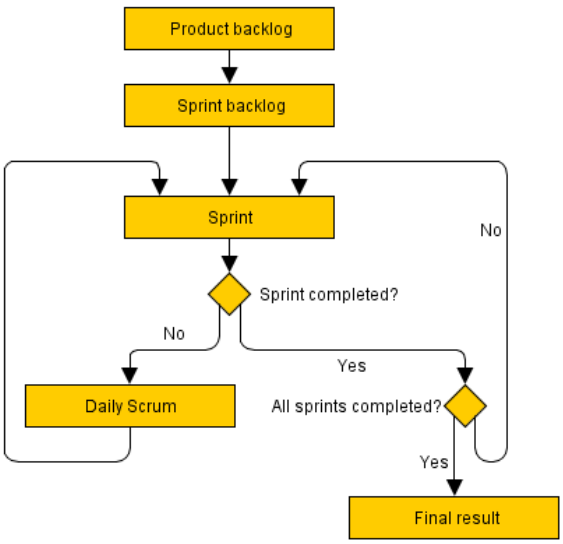

Figure 2: Scrum workflow as it has been applied in the development of s4a.js

2.2.3 Use of open standards

The use of open standards is paramount to the implementation of s4a.js. Providing interfaces to known data sources and structures is an absolute requirement when introducing a new utility library into a field that is flooded by half-baked solutions. The work includes the use of many open standards, but it is necessary to highlight the most important ones:

- GeoJSON (Butler, et al., 2008) – a format that enables the encoding of points, lines and polygons along with their attributes as simple JavaScript Object Notation structures.
- TMS (Maso, Pomakis, & Nuria, 2010) – a scheme for identifying map tiles based on location
- MBTiles (MacWright, White, Kaefer, & Miller, 2013) – an SQLite based data format for bundling offline map tiles into a single file for ease of distribution and management
- SFS-WKT (Open G. I. S. Consortium, 1999) – a notation for describing simple features in well-known-text
- jQuery (The jQuery Foundation, 2016) – a cross-browser infrastructure library that constitute a de-facto standard extension to JavaScript

2.2.4 Common repository layout

The following repository layout has been agreed and used throughout the development of the API library

- css – for any Cascading Style Sheet source files
- src – for any JavaScript source files
    - sub-folder per namespace
- test – for any JavaScript source test files
    - sub-folder per namespace
- (dist) – auto-generated upon build (grunt)
- (doc) – auto-generated upon build (grunt)
- (node_modules) – auto generated and populated by build system (npm)

2.2.5 Common coding style

JavaScript is not a strongly typed language – and its semantics permits users to write a piece of code that does 'the same thing' in a multitude of different ways.

One of the objectives of the library is to make it easily extendible and to encourage the formation of a critical mass of interested parties, i.e. developers, who would like to adopt the system.

For this reason, the s4a.js development has agreed on a number of stylistic conventions that have been followed by all developers.

- Namespaces and nesting
    - Design patterns - simulated Object Orientation using design patterns
    - Modules
    - Classes
    - Singletons
    - Factories
    - MixIns
    - Pub Sub
- Code element naming conventions
    - Camel case for functions, methods
    - All upper case for constants
    - Proper case for classes
    - All lower case for namespaces
- Inline code comments

- Following the JSDoc 3 standard, see sub chapter 2.2.7 below.

2.2.6 Build system

Development has taken place in a large number of smaller files that are stored in a hierarchical file structure as described above. For deployment purposes, it is however highly impractical to include hundreds of individual JavaScript files in the 'head section' of the HTML mark-up.

Thus, it has been necessary to choose a build system that performs a number of tasks in the development workflow:

- Test code in each of the source files for syntax of style errors
- Concatenate individual files into a single file for distribution
- Generate API reference documentation by extracting in-line code comments
- 'Minify' the single file by auto-refactoring private function and method names
- Run unit tests to verify code

While there are contenders like Gulp or Ant in the market, SDI4Apps has chosen to work with the node.js based Grunt task runner (GRUNT development team, 2016) as a build system.

Dependent packages, i.e. libraries that s4a.js depends on for its operation are managed using the Node Package Manager (npm, Inc., 2016)

2.2.7 Documentation

In order for third parties – or for that matter second parties within the project consortium – to use s4a.js it is necessary to provide documentation.

Most JavaScript developers are familiar with API reference documentation that separates between namespaces, modules and classes – and that documents the internal members of each of the afore mentioned concepts as either properties or methods.

By following the simple code commenting conventions of JSDoc 3 it is possible to achieve two objectives:

- Make code human-readable for developers who would like to contribute to or branch off from the s4a.js code
- Enable the auto-generation of API reference documentation by means of a documentation generator that parses comments from source code files and compiles them into HTML.

```
/**
 * This is a description of the namespace s4a.map
 *
 * @namespace s4a.map
 */
```

Code fragment 1: Example of JSDoc comment identifying a namespace

```
/**
 * A helper class to quickly add maps to your HTML5 applications
 *
 * @class
 */
s4a.map.MapHelper = function(nodeSelector) {
   ...
}
```

Code fragment 2: Example of JSDoc comment identifying a JavaScript 'class'

```
/**
  * Calculate the shortest path between two network nodes on a topological network
  *
  * @param {Number} from The ID of the from node
  * @param {Number} to The ID of the to node
  * @return {Promise} description
  * @public
  */
module.getRoute = function(from, to) {
   ...
}
```

Code fragment 3: Example of JSDoc comment identifying a 'class method' or function

2.3 Testing

The code framework is tested using the Jasmine (Pivotal Labs, 2016) testing framework for unit tests as well as through a set of HTML pages that demonstrate the features of each module.

```
describe('Feature synchronization service', function() {
    it('Test 1 km2', function(a) {
        var s = new gh.stats('Test 1 km2');
        for (var i = 0; i < 100; i++) {
            jQuery.post(featureSyncUrl, {
                action: 'CheckOut',
                longitude: 10,
                latitude: 59,
                buffer: 500
            }, function(res) {
                s.log();
            }, 'json').fail(function(res) {
                s.log();
            });
        }
        s.print();
        expect(s.avg).toBeLessThan(1000);
    });
});
```

Code fragment 4: Example of unit test from s4a.js API library

2.4 Deployment

S4a.js is deployed both as part of each SDI4Apps platform instance as well as a stand-alone JavaScript API library that partly may be used independently of the platform.

During the development phase, the software is deployed to the main platform instance at http://platform.sdi4apps.eu as well as to AVINET's development servers at http://s4a.avinet.no. The latest version may always be found on GitHub.

The library is distributed under the Apache-2-0 Open Source license. The creators are at liberty to issue the library under an alternative license at any later stage.

3. WALKTHROUGH OF FIRST RELEASE

This chapter provides a walk-through of the modules that have been completed in the first release and describes each module as well as shows code examples on how to invoke the code and the expected output.

3.1 Map module

The map module is a simple helper module that implements productivity methods that makes it very easy to add a map component with your own custom background and thematic data to an existing HTML5 application.

The map module also implements functions that make it easy to create new tools and controls and add these to the map interface.

Usage example

```
var myMap = new s4a.map.MapHelper('#map');

myMap
    .addData('test.geojson')
    .addTool([
        s4a.map.tools.ZoomIn,
        s4a.map.tools.ZoomOut,
        s4a.map.tools.Pan)
    .draw();
```

Code fragment 5: Example of instantiating the s4a.js MapHelper class to add a map

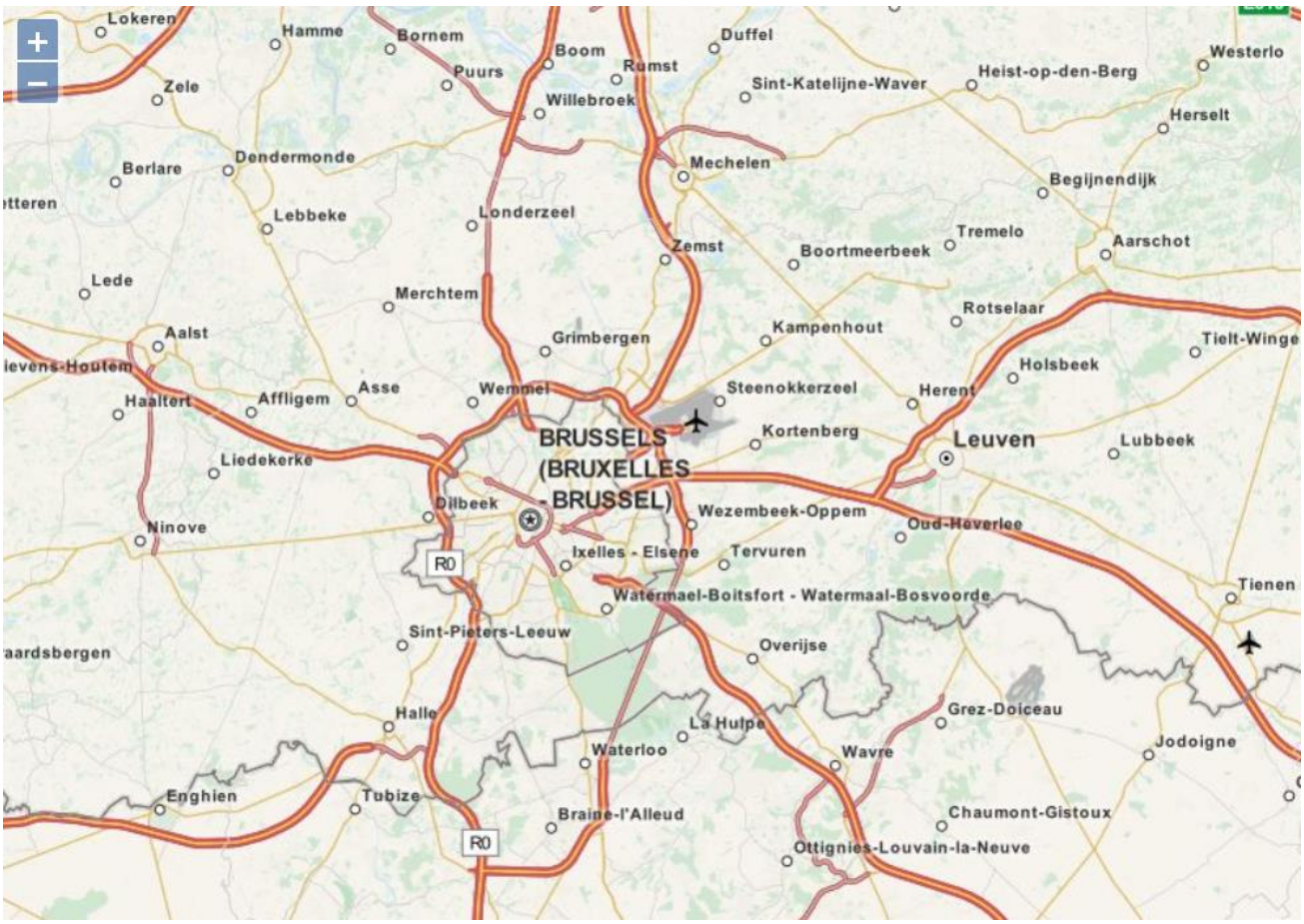

Figure 3: Map created by s4a.map.MapHelper

```
var myMap = ...

var myTool = new s4a.map.Tool(s4a.map.ToolType.TOOL);
    .setIcon('img/icon.png')
    .setLabel('Click me')
    .setGeomType(s4a.map.GeomType.POINT)
    .onClick(function(clickEvent) {
        alert(clickEvent.lonlat);
    });

myMap.addTool(myTool);
```

Code fragment 6: Example of creating a new custom tool

3.2 Information retrieval module

The information retrieval module enables querying of data from a high performance search API based on a Lucene index managed in the SDI4Apps platform, accessible through the Open API.

Data may also be added to the index from ESRI Shapefiles or PostgreSQL/PostGIS databases. These methods must be invoked on the server as exposing them as public methods would lead to liability risks with regards to content quality, licensing – and nature.

Usage example

```
var myQuery = new s4a.ir.QueryHelper();

var myQueryResults = myQuery
    .setQuery('Brussels')
    .setDistance('POINT(4.3517 50.8503)', 10000)
    .query() // Function returns a jQuery 'promise'

myQueryResults.then(function(response) {
    console.log(response.data); // Output the query results
};
```

Code fragment 7: Example of instantiating s4a.js QueryHelper to issue a query

```
{
   "status": "ok",
   "response":    {
      "resultList":       [
                  {
            "address": null,
            "state": "ACTIVE",
            "sourceType": "florence_heritage",
            "path": false,
            "sourceId": "aebAnnunziata_48_1",
            "ente": "null",
            "geometryLine": null,
            "geometryPoint": "MULTIPOINT (11.25755 43.773849)",
            "geometryPolygon": null,
            "geometryType": "POINT",
            "daysBefore": 0,
            "daysAfter": 0,
            "geometry": "MULTIPOINT (11.25755 43.773849)",
            "centroid": "POINT (11.25755 43.773849)",
            "mainImageThumbnailUrl": null,
            "mainImageUrl": null,
            "hash": "ZPI01Q8fCMM97SKCglKkfTfhdHRFWpMuVQckjRFVITc=",
            "facebookId": null,
            "twitterHashtag": null,
            "userGenerated": false,
            "rating": null,
            "recurrency": "",
            "name":             [
                              {
                  "locale": "it",
                  "text": "Via dei Servi"
               },
                              {
                  "locale": "en",
                  "text": "Via dei Servi"
               }
            ],
            "description":             [
                              {
                  "locale": "it",
                  "text": "Via di collegamento tra piazza del Duomo e piazza SS.
Annunziata, deve il suo nome all'Ordine dei Serviti o Servi di Maria, fondato nel XIII
secolo da sette nobili fiorentini che edificarono la Chiesa della SS. Annunziata. La via,
punteggiata di eleganti palazzi cinquecenteschi, offre da entrambi i lati una suggestiva
veduta prospettica"
               },
                              {
                  "locale": "en",
                  "text": "This street joins piazza del Duomo and piazza SS. Annunziata
and takes its name from the Order of the Serviti or Servants of Mary, founded in the 13th
century by seven Florentine noblemen who built the Church of SS. Annunziata. The street is
dotted with some elegant sixteenth century palaces and gives a very suggestive perspective
view from both sides"
               }
            ],
            "uri":
"http://purl.org/sics/digitallocation/113b5b2c71fa4d2b9a25ac8358359a5a",
            "categories": [            {
               "pkCategory": 134,
               "details": [               {
                  "name": "Stazioni di percorsi tematici",
                  "description": null,
                  "locale": "it"
               }],
```

```
            "profile": 0,
            "scopes": [],
            "validForDL": true,
            "validForDR": true
         }],
         "pkDigitalEntity": 10046,
         "deType": "DL",
         "lastUpdateUser":             {
            "email": null,
            "ente": null,
            "pkUser": 10612,
            "cognome": "ETL",
            "nome": "ETL"
         },
         "creationDate": "2013-10-31",
         "lastUpdateDate": "2013-10-31",
         "creationUser":             {
            "email": null,
            "ente": null,
            "pkUser": 10612,
            "cognome": "ETL",
            "nome": "ETL"
         }
      },
      {...}
```


Code fragment 8: Example output from QueryHelper

3.3 Advanced visualization module

The advanced visualization module builds on three existing components: Crossfilter (for performing map reduce operations on JSON data), D3.js for creating data visualizations in SVG and dc.js (for performing dynamic visualization and filtering of data).

The module extends dc.js with several new chart types.

3.3.1 Chart types

The first release of the advanced tools API includes the following chart types:

- Pie chart
- Bar chart
- Row chart (horizontal bar chart)
- Scatter chart

Each chart type object implements a shared interface 'IVizObj' that enables any chart object to be added to a view coordinator that can update it whenever the underlying data that it displays are updated, filtered or otherwise manipulated.

Usage example

```
var data = [
    {country: 'NO', measurement: 10},
    {country: 'CZ', measurement: 12},
    {country: 'DK', measurement: 4},
    {country: 'AT', measurement: 5}
];

var vc = new s4a.viz.ViewCoordinator(data);

var chart1 = new s4a.viz.chart.Pie('#chart1', vc, 'measurement');

var chart2 = new s4a.viz.chart.Bar('#chart2', vc, 'measurement');

vc.draw();
```

Code fragment 9: Pie chart, bar chart example

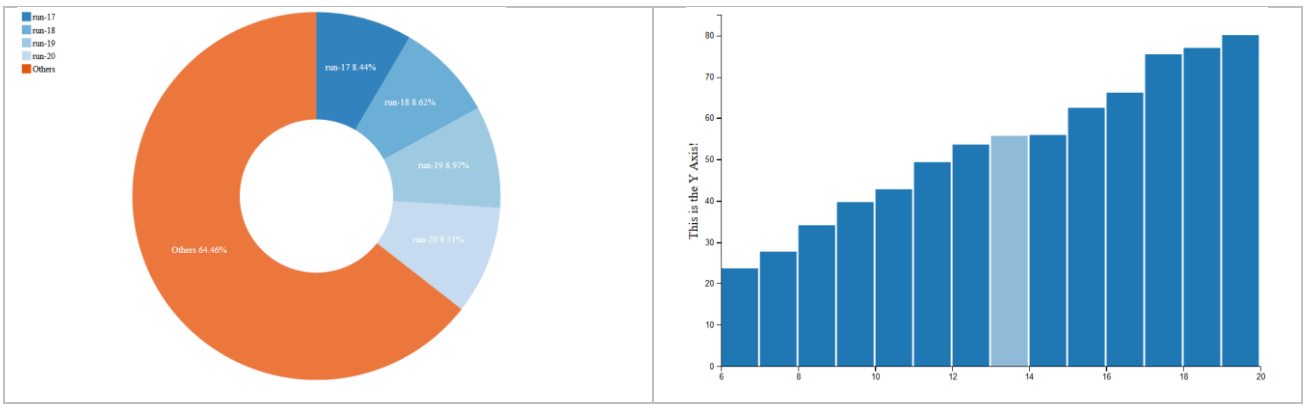

Figure 4: Example visualization objects (generated via s4a.js using dc.js methods)

3.3.2 Map types

The first release of the advanced tools API includes the following map types:

- Choropleth map
- Symbol map
- Heat map
- Live data map
- Bubble pie map

Each map type object implements a shared interface 'IVizObj' that enables any of the map objects to be added to a view coordinator that can update it whenever the underlying data that it displays are updated, filtered or otherwise manipulated.

NB! For the benefit of reviewers, it is worth noting that the Prism map object type has not yet been implemented. This is planned for the second release and has been pending the choice of a suitable 3d library that does not add too much overhead to the overall s4a.js distribution comparing to the rather limited purpose it serves in the infrastructure. Candidate technologies that currently are being assessed include Cesium and Three.js. The decision has been delayed while looking for a solution with a smaller memory and CPU 'foot print'.

Usage example

```
var data = [
    {country: 'NO', measurement: 10},
    {country: 'CZ', measurement: 12},
    {country: 'DK', measurement: 4},
    {country: 'AT', measurement: 5}
];

var vc = new s4a.viz.ViewCoordinator(data);

var map1 = new s4a.viz.chart.Pie('#map1', vc, 'measurement', 'country', 'countries.json');

vc.draw();
```

Code fragment 10: Choropleth map example

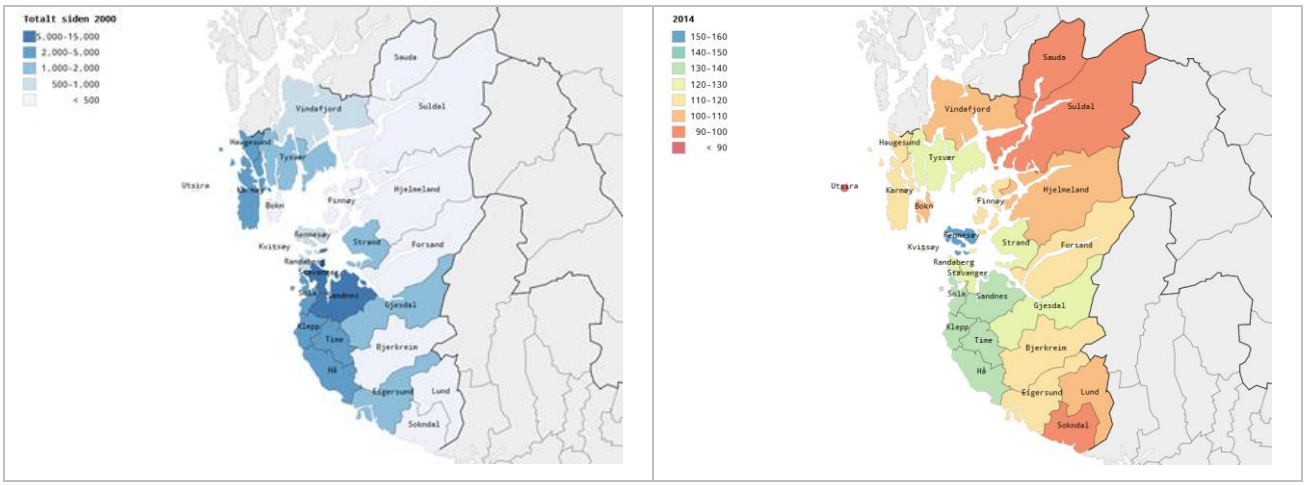

Figure 5: Example of choropleth map (generated via s4a.js using D3.js methods)

3.3.3 Coordinated view

The coordinated view is what puts 'advanced' into the advanced tools API. This is a data object that visualization objects can subscribe to and that will update each subscribed object whenever the underlying data are altered.

The inspiration for this module is the Crossfilter object as it is used in dc.js – but we have extended the object in order to allow a broader range of visualizations compared to the narrow, albeit high-performance, scope addressed by dc.js.

Usage example

```
var data = [
    {country: 'NO', measurement: 10},
    {country: 'CZ, measurement: 12},
    {country: 'DK', measurement: 4},
    {country: 'AT', measurement: 5}
];

var options = {
    geocol = 'country',
    geotype = s4a.viz.GeoType.COUNTRY
};

// Create a view coordinator
var vc = new s4a.viz.ViewCoordinator(data, options);

// Create a choropleth map and add it to the view coordinator
// passing the vc object automaticallyt takes care of this.
var map1 = new s4a.viz.map.Choropleth('#map1', vc, 'measurement', 'country',
'countries.json');

// Add a pie chart too
var chart1 = new s4a.viz.chart.Pie('#chart1', vc, 'measurement');

// Draw map1 and chart1
vc.draw();

// Display only records where the measurement value is 12 and
// redraw all visualisation objects that are connected to the view coordinator
// internally loops thorugh all objects added to the view coordiantor and calls their
// .update() method.
vc.filterExact('12');
```

Code fragment 11: Coordinated view example

3.4 Mobile module

The mobile module implements functions that makes it possible to browse background maps on mobile devices while in offline mode. Furthermore, the module enables checking out a portion of a feature layer as a GeoJSON file, editing it locally (i.e. while offline) and checking it back into the server, resolving any conflicts that may have arisen as a consequence of concurrent editing etc.

3.4.1 Offline feature layer

The first function of the mobile module is the capacity to check out a feature layer. The prerequisite for this function is that the feature layer exists as a PostgreSQL database table on the SDI4Apps platform instance server. Furthermore, the layer must be registered in a metadata table that lists layers that should be made available for offline use.

If these two conditions are met, it is possible to check out a portion of the map by supplying the layer identifier and an extent.

The methods to create the layer communicates with an Open API server instance via AJAX Http POST requests and requires CORS to be implemented on the server side.

Usage example

```
var myCheckOut = s4a.mobile.featureSync.CheckOut('layer-001',
    {
        minX: -180,
        minY: -90,
        maxX: 180,
        maxY: 90});

myCheckOut.then(function(response) {
    // GeoJSON object
    var myGeoJson = response.data;

    // modify GeoJSON

    // Check in the modified data
    var myCheckIn = s4a.mobile.featureSync.CheckIn('layer-001', myGeoJson)
        .then(function(response) {

            // If no conflicts, myConflicts should return an empty array
            var myConflicts = response.data;

        }
}
```

Code fragment 12: Example of checking out, editing and checking in offline feature layer

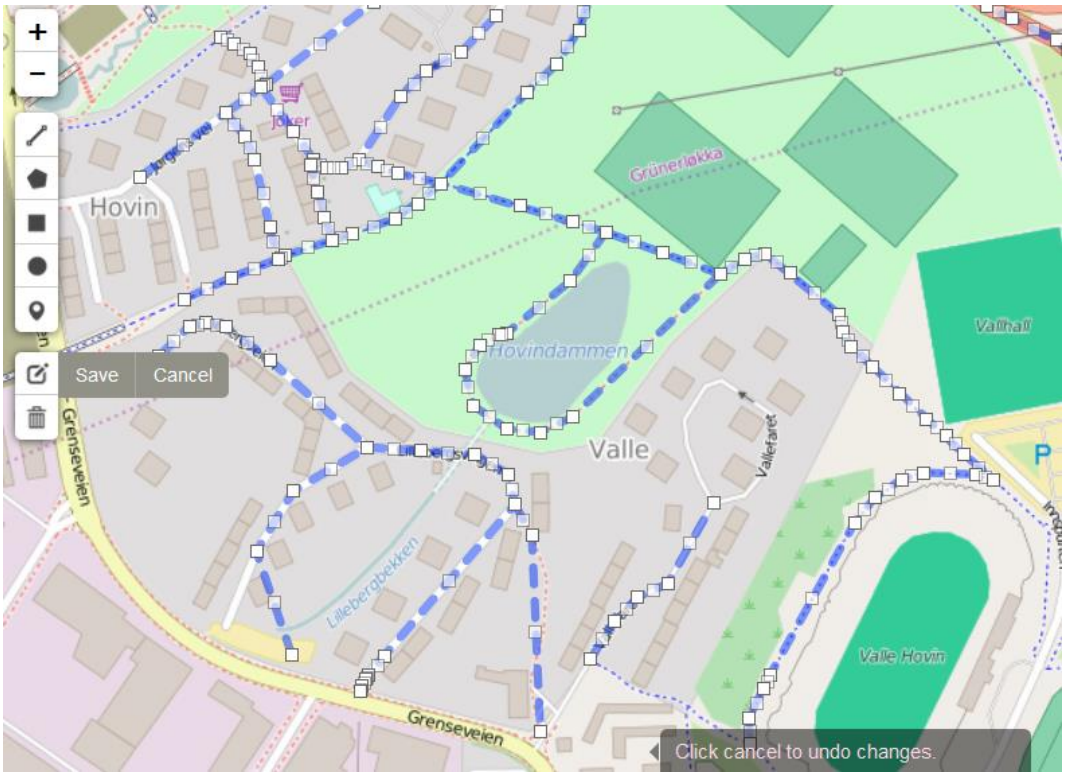

Figure 6: Example showing OfflineFeatureLayer being edited

3.4.2 Offline tile layer

The offline tile layer serves the simple purpose of providing background map tiles while in offline mode. It works by downloading a pre-populated SQLite database onto the device and creating a new OpenLayers layer type.

This layer is ONLY possible to use in mobile applications developed using Cordova and with the SQLite plugin 'cordova-sqlite-ext' installed. It is not possible to use it on desktop computers, as there is no facility to read/write SQLite databases.

Usage example

```
var myMap = new s4a.map.MapHelper();

var myOfflineTileLayer = new s4a.mobile.OfflineTileLayer('osm.mbtiles');

myMap.setBasemap(myOfflineTileLayer);
```

Code fragment 13: Example of creating offline tile layer and adding it to a map

3.5 Analytics and modelling module

The analytics and modelling module is an emphasis for the second release and at present only one service exists within the module, namely the routing service that corresponds to the routing service exposed by the Open API in an SDI4Apps platform instance.

3.5.1 Routing

The routing component is a wrapper on top of PostgreSQL, PostGIS and pgRouting that enables users to execute three common routing scenarios from JavaScript without having to have access to their own server infrastructure or data.

It communicates with the server via AJAX Http POST requests and requires CORS to be implemented on the server side.

- GetNearestNode – routing operations go between nodes in the topological network – function returns the node id closest to a lon/lat location
- GetShortestPath – given the id of a start and an end node, calculates the shortest path between the two
- GetReachableArea – given the id of a start node id and a distance, calculates the reachable area (along the road network)
- GetOptimalRoute – given a set of node ids, calculates the optimal route to visit them all

Usage example

```
s4a.analytics.routing.GetRoute(1, 2)
    .then(function(response) {
        // Route response object, including representation of
        // route as GeoJSON
        var myRoute = response.data;
    });
```

Code fragment 14: Example of routing request on dynamic road network stored in PostGIS

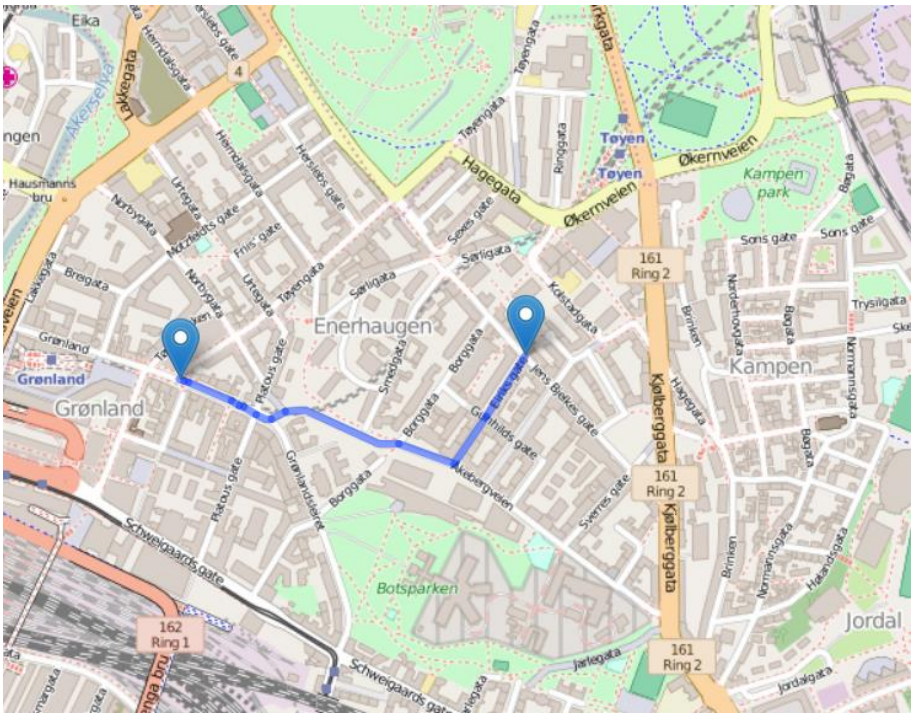

Figure 7: Example of result from invoking routing

4. FEATURES PLANNED FOR SECOND RELEASE

Having introduced the features included in the first release of the s4a.js client-side JavaScript library and knowing that a second release is planned at the very end of the project, the next logical step is to describe what will be the emphasis of the work within work package 4 between month 24 and month 36.

4.1.1 Product packaging in context of exploitation

With one year left of the project it is necessary to put efforts into planning for the sustainability of the project results, including this software deliverable.

The pure 'business aspects' of sustaining and potentially monetizing the products resulting from the project will take place in its own work package. However, there are technical measures that must be taken in order to secure that an 'exploitable' product is available.

Two major hurdles are out of the way:

- The IPR is clarified both in terms of external licensing and internal ownership
- The software architecture is modular, as noted in chapter 6 below

The generic nature of the library is at the same time both a strength and a liability. It can do something for anyone – but 'anyone' is a difficult market group to reach by means of traditional marketing techniques within the geo-informatics sector.

It may be necessary to augment the library to meet the business requirements of a specific market niche – or to take it apart and find 'good homes' for the individual components within other Open Source projects. Both of these 'roads' have technical implications and WP4 will work closely with the sustainability strategy of the project to secure that the two are synchronized.

4.1.2 Robustness, scalability, cloud optimization

While all features in the first release are functioning and have passed their respective tests, it is necessary to be aware that success in a cloud based infrastructure means high concurrent load. Milliseconds become important and it is necessary to optimize all functions. Especially those API methods that communicate with the server platform require optimization. Extensive stress testing for error tolerance is also required in the event of network timeouts.

4.1.3 Analytics and modelling module

At present, only one service is part of the analytics and modelling module, namely the routing service. The last year will seek to identify a relevant multi-variate analysis algorithm to be built into the library. If an algorithm can be identified that corresponds to a legally mandated public sector reporting requirement, this may be a bridgehead to bring SDI4Apps project results to the market.

4.1.4 Additional visualizations, 3D, prism map

The final emphasis for the last year of the project is the addition of more visualization objects. The ones that are present today are in line with the specification. However, the underlying library D3.js has a

multitude of possibilities that have not been exploited and it is necessary to explore the intersection between SDI4Apps pilot user requirements and the capabilities of D3.js.

Furthermore, the remaining map type, the ‘Prism map’ remains to be implemented. This requires the identification of a suitable and lightweight 3D library, which may in turn open doors to additional visualization options.

5. SUSTAINABILITY PRINCIPLES

This chapter re-iterates important sustainability principles that were defined in the specification document and that have been retained and rigorously observed in the implementation of s4a.js

The functionality of the library has been implemented in a modular fashion in order to secure that the components easily may be split into different libraries and exploited independently of each other.

The prime objective of the JavaScript library is to be an interface to advanced server-side functionality exposed by the SD4Apps platform; the modular design does however allow a wider range of sustainability options to ensure that SDI4Apps results will be used after the project ends.

These measures have not imposed limitations on functionally in the library – nor have they added to the complexity of the implementation task. They just serve to demonstrate that following well-proven design principles when writing code is beneficial on many levels.

Modules that do not rely on the SDI4apps platform but merely implements client-side visualization functions may have an after-life both as part of s4a.js and as contributed modules in existing Open Source libraries that already have a critical mass of users. This is a two-tier exploitation strategy.

6. WORKS CITED

The jQuery Foundation. (2016). *jQuery API documentation*. Retrieved from jQuery: https://api.jquery.com/

Bergheim, S. R., Pironi, A., Tarini, D., Berzins, R., Charvat, K., Mildorf, T., & Kuba, M. (2014). *D4.2 Advanced Tools API Design*. SDI4Apps consortium.

Butler, H., Daly, M., Doyle, A., Gillies, S., Schaub, T., & Schmidt, C. (2008, 06 16). *GeoJson Specification*. Retrieved from GeoJson: http://geojson.org/geojson-spec.html

GRUNT development team. (2016). *GRUNT The JavaScript Task Runner*. (GRUNT development team) Retrieved from gruntjs.com: http://gruntjs.com/

JSDoc 3 contributors. (2016). *Use JSDoc: Index*. (JSDoc 3 Project) Retrieved 2016, from Use JSDoc: http://usejsdoc.org/index.html

MacWright, T., White, W., Kaefer, K., & Miller, J. (2013, 08 07). *MBTiles Specification*. Retrieved from mbtiles-spec: https://github.com/mapbox/mbtiles-spec

Maso, J., Pomakis, K., & Nuria, J. (2010). *OpenGIS web map tile service implementation standard*. Open Geospatial Consortium Inc.

npm, Inc. (2016). *npm*. Retrieved from www.npmjs.com: https://www.npmjs.com/

Open G. I. S. Consortium. (1999). *OpenGIS Simple Features Specification for SQL, Revision1. 1*. Open G. I. S. Consortium.

Osmani, A. (2015). *Learning JavaScript Design Patterns* (1.62 ed.). O'Reilly.

Pivotal Labs. (2016). *Jasmine: Behavior-Driven JavaScript*. (Pivotal Labs) Retrieved from Jasmine: http://jasmine.github.io/

Schwaber, K. (2004). *Agile project management with Scrum*. Microsoft press.

7. APPENDICES

Appendix A: s4a.js API reference

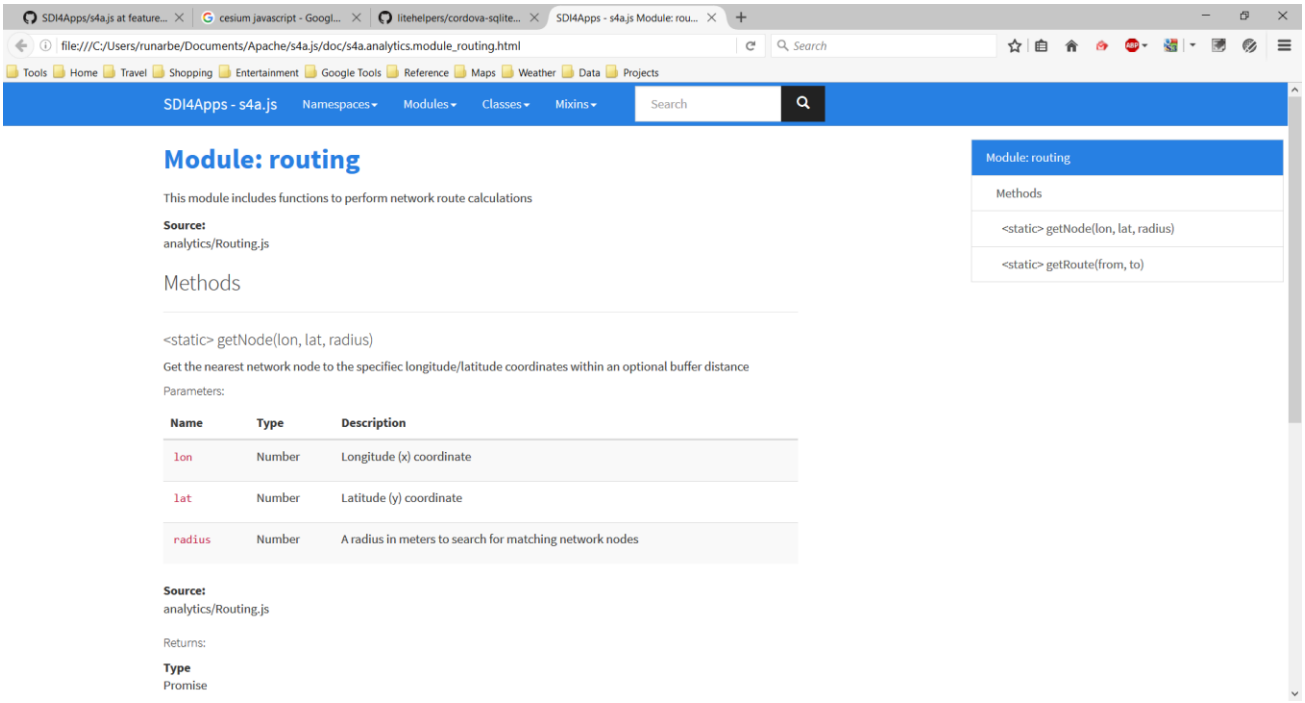

Figure 8: Screenshot from the online s4a.js API reference documentation

The reference documentation may be found at http://s4a.avinet.no/apidocs or may be built by cloning the GitHub repository and running the build commands npm/grunt.