DELIVERABLE

Project Acronym: SDI4Apps
Grant Agreement number: 621129
Project Full Title: Uptake of Open Geographic Information Through Innovative Services Based on Linked Data

D4.2 ADVANCED TOOLS API DESIGN

Revision no. 03

Authors: Stein Runar Bergheim (AVINET)
Alessandro Pironi (Hyperborea)
Daniele Tarini (Hyperborea)
Raitis Berzins (CCSS)
Karel Charvat (CCSS)
Tomas Mildorf (UWB)
Martin Kuba (Masaryk University)

Project co-funded by the European Commission within the ICT Policy Support Programme

<table>
<thead>
<tr>
<th>Dissemination Level</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>P</td>
<td>Public</td>
</tr>
<tr>
<td>C</td>
<td>Confidential, only for members of the consortium and the Commission Services</td>
</tr>
</tbody>
</table>
REVISION HISTORY

<table>
<thead>
<tr>
<th>Revision</th>
<th>Date</th>
<th>Author</th>
<th>Organisation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>01</td>
<td>01/02/2015</td>
<td>Stein Runar Bergheim</td>
<td>AVINET</td>
<td>First draft based on DoW</td>
</tr>
<tr>
<td>02</td>
<td>30/03/2015</td>
<td>Stein Runar Bergheim</td>
<td>AVINET</td>
<td>Updates based on partners' input</td>
</tr>
</tbody>
</table>

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This deliverable contains original unpublished work except where clearly indicated otherwise.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

Revision History .......................................................................................................................... 3
Table of Contents .......................................................................................................................... 4
List of Code Fragments .................................................................................................................. 5
List of Figures ............................................................................................................................... 6
List of Acronyms ........................................................................................................................... 7
Executive Summary ....................................................................................................................... 8
1. Preface ....................................................................................................................................... 9
2. Methodology ............................................................................................................................. 11
   2.1 Specification ....................................................................................................................... 11
   2.2 Development ..................................................................................................................... 11
   2.3 Testing ............................................................................................................................... 13
   2.4 Deployment ....................................................................................................................... 14
3. The building blocks of s4a.js ................................................................................................. 15
4. 5 Advanced Tools API modules ............................................................................................... 17
   4.1 Map module ...................................................................................................................... 17
   4.2 Information retrieval module ............................................................................................. 18
   4.3 Advanced visualization module ......................................................................................... 20
   4.4 Mobile module .................................................................................................................. 28
   4.5 Analytics and modelling module ....................................................................................... 30
5. Sustainability considerations .................................................................................................... 33
6. Appendices ............................................................................................................................ 34
   Appendix 1: Advanced API UML element descriptions ........................................................ 34
LIST OF CODE FRAGMENTS

Code fragment 1: Top-level namespace definition ................................................................. 12
Code fragment 2: JavaScript OOP module design pattern ....................................................... 12
Code fragment 3: JSDoc code comment standard ................................................................. 13
Code fragment 4: Including the s4a.js library in an HTML5 application .............................. 14
LIST OF FIGURES

Figure 1: The relationship between D4.2 and other SDI4Apps deliverables ........................................... 10
Figure 2: Dependencies between s4a.js and reusable function libraries ......................................................... 15
Figure 3: The modules of the s4a.js client-side JavaScript library ................................................................. 17
Figure 4: Overview of simple helper class for quickly adding maps and data to user interfaces .................. 18
Figure 5: Overview of helper class for issuing queries to the server-side OpenAPI .................................. 19
Figure 6: Overview of chart and map based visualization objects ................................................................. 20
Figure 7: Example of a simple pie chart ......................................................................................................... 21
Figure 8: Example of a simple bar chart ....................................................................................................... 21
Figure 9: Example of two-dimensional scatter chart ..................................................................................... 22
Figure 10: Example of line chart with a time dimensional axis ................................................................. 22
Figure 11: Example of graduated symbol map ............................................................................................... 23
Figure 12: Example of map combining a bubble and pie charts ................................................................. 24
Figure 13: Example of heat map .................................................................................................................. 25
Figure 14: Example of prism map ................................................................................................................ 25
Figure 15: Example of choropleth map ........................................................................................................ 26
Figure 16: Example of live data map ........................................................................................................... 27
Figure 17: Example of coordinate view incorporating line charts, bar charts and maps .......................... 28
Figure 18: Solution to the offline editing problem ....................................................................................... 29
Figure 19: The abstract specification of the OpenAPI analytics and modelling service ......................... 32
# LIST OF ACRONYMS

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Full Form</th>
</tr>
</thead>
<tbody>
<tr>
<td>API</td>
<td>Application Programming Interface</td>
</tr>
<tr>
<td>DoW</td>
<td>SDI4Apps Description of Work, Annex I to the Grant Agreement</td>
</tr>
<tr>
<td>EC</td>
<td>European Commission</td>
</tr>
<tr>
<td>ERFC</td>
<td>European Regional Framework for Co-operation Association</td>
</tr>
<tr>
<td>EU</td>
<td>European Union</td>
</tr>
<tr>
<td>GI</td>
<td>Geographic Information</td>
</tr>
<tr>
<td>ICT</td>
<td>Information and communications technology</td>
</tr>
<tr>
<td>IGO</td>
<td>Intergovernmental organization</td>
</tr>
<tr>
<td>INGO</td>
<td>International non-governmental organization</td>
</tr>
<tr>
<td>INSPIRE</td>
<td>Infrastructure for Spatial Information in the European Community</td>
</tr>
<tr>
<td>MIF</td>
<td>INSPIRE Maintenance and Implementation Framework</td>
</tr>
<tr>
<td>NGO</td>
<td>Non-Governmental Organization</td>
</tr>
<tr>
<td>OSM</td>
<td>OpenStreetMap</td>
</tr>
<tr>
<td>SDI</td>
<td>Spatial Data Infrastructure</td>
</tr>
<tr>
<td>SME</td>
<td>Small and medium-sized enterprises</td>
</tr>
</tbody>
</table>
EXECUTIVE SUMMARY

This document describes the high-level design of the client side JavaScript library “companion” to the SDI4Apps OpenAPI (described in D3.3.1 “Open API Design & First Release”). The deliverable builds indirectly on the results of D3.1 “Architecture Concept” and D3.2.1 “Enablers Deployment - First Release”.

The library has been given the working title “s4a.js”. S4a.js builds on state-of-the-art infrastructure libraries including jQuery, D3.js and Crossfilter. It provides a combination of advanced client side visualization widgets that can be used independently and a set of widgets that depends on corresponding methods in an instance of the SDI4Apps OpenAPI.

The purpose of s4a.js is to help application developers quickly build spatial enabled user interface applications using the SDI4Apps platform. It can be cumbersome to write clients to evolving web service APIs, for this reason s4a.js is a toolbox of easy-to-implement user interface widgets that can be added to container elements in the HTML mark-up of end-user applications. If the OpenAPI must be changed, s4a.js serves as an abstraction layer that permits the uninterrupted operation of the custom end-user application despite server-side changes. This allows for great flexibility in terms of incrementally improving the OpenAPI and s4a.js in parallel.

Using s4a.js methods, the user can add interactive maps, diagrams and coordinated views to her or his custom user interface applications with a minimal amount of code, in some cases only a single line.

The second purpose of the s4a.js library is to leverage access to the OpenAPI platform from mobile devices. With the emphasis on responsive design in creating the advanced tools API widgets, SDI4Apps ensure that the library will work equally well on mobile as on desktop platforms.

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.

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

An API is a high-level and abstract term for something that exposes functionality that can be used to build applications of one sort or another. In the context of SDI4Apps, an API can refer to at least four different things:

- A user-interface API in the form of a JavaScript library for HTML5 rendering engines such as Google Maps JavaScript API v3 and similar.
- A web-service API in the form of a HTTP interface for server-side operations such as typical WMS or WMTS services.
- A platform-as-a-service cloud management API for remotely managing the platform on which the application is being hosted.
- An infrastructure-as-a-service cloud management API such as the Google Compute Engine API, something that is needed for scalability and high availability management on clouds from different providers.

This deliverable describes the high level design of the first type of API listed above, a client-side JavaScript library, hereafter referred to as **s4a.js**, that makes it easy to build applications based on the SDI4Apps platform.

s4a.js depends absolutely on the SDI4Apps server-side Web Service API, hereafter referred to as the **SDI4Apps OpenAPI** that provides access to remote data sources and processing services. An instance of the OpenAPI can exist autonomously without an accompanying client-side library, the purpose of s4a.js is to leverage access to the server side SDI4Apps OpenAPI for HTML5 and JavaScript developers; making it easy to create spatial-enabled end-user applications that exploit the broad range of capabilities exposed by our Cloud platform.

The OpenAPI is an example of the second type of API as listed above.
Figure 1: The relationship between D4.2 and other SDI4Apps deliverables

There is a strong dependency between deliverable D4.2 that covers the client-side library including mobile and advanced visualisation modules - and deliverable D3.3.1, the Open API, that specifies the server side of things and the web services API.

The SDI4Apps OpenAPI will consist of Web Services. From the perspective of a Web Service, there is no distinction between mobile and desktop clients - a Web Service accepts requests, returns responses and does not interfere with what happens beyond receiving requests and issuing responses. In the context of SDI4Apps, the distinction between mobile and desktop therefore only becomes relevant for the specification of s4a.js.

Some of the s4a.js functions will be suitable for use in mobile applications, others only on personal computers. The main factors that determine what will be available for each platform are:

- Whether the information can be presented in a meaningful way on the screen size available, i.e. on a small mobile device screen
- Whether the system resources available to the browser is capable of handling the amount of data and rendering elements that are used, i.e. by the advanced visualization API
2. METHODOLOGY

The JavaScript client library will be developed according to an agile software development methodology under the governance of the SDI4Apps project. The methodology ranges from specification through development and testing to deployment.

2.1 Specification

This document serves as a high-level design specification for the Advanced Tools API. At the time of practical implementation, a detailed object specification will be prepared, providing details on classes, properties and methods.

These detailed designs will be integrated into an updated version of this document and submitted as an annex to deliverables D4.3.1 and D4.3.2.

The preparation of this design document has been based on four main inputs:

1. The output from the user requirements analysis
2. State-of-the-art as identified through the DoW and updated through document studies and web research
3. The concurrent design activities of the server side companion OpenAPI, D3.3.1
4. Complimentary assessments based on technical partners skills and experience

The document has been edited and structured by AVINET with contributions from SDI4Apps technical partners.

A focus workshop targeting the Advanced Tools API design was held during March 2015, securing the alignment of interfaces between the Cloud platform, the server-side OpenAPI and the client-side Advanced Tools API.

The workshop also established common understanding of the scope of the API and definitions of key terms and concepts.

This design document uses a combination of literal descriptions, illustrations and UML Use Case diagrams\(^1\) to describe the object model of s4a.js and the relationship between the client-side library and the server-side OpenAPI.

2.2 Development

The implementation of the API will be the shared responsibility of several technical partners. The methodology must therefore be suitable for decentralized development teams to operate independently under joint coordination and governance.

The steps taken to secure this end include:

- Create a base namespace, object architecture, coding style and commenting/documentation guideline

\(^1\) "Use Case Diagram - Wikipedia, the free encyclopedia." 2006. 21 Mar. 2015
Establishing a high level division of modules among development teams
Create a shared, Git\textsuperscript{2} based repository for source code and documentation
Design a coordination structure based on regular interval online conferences combined with fixed sessions at plenary meetings.

Within each development team, work will be organized in accordance with the agile Scrum\textsuperscript{3} methodology. This guarantees both momentum and quality in the work carried out.

\subsection{2.2.1 Common coding style}
The s4a.js library will be implemented using design patterns\textsuperscript{4} that emulate object orientation.

\begin{center}
\textbf{Code fragment 1: Top-level namespace definition}
\end{center}

\begin{verbatim}
var s4a = {
    version: 1.0
};
\end{verbatim}

All the code of the API will be implemented within the top level object 's4a'. In object orientation terms, this object shall be considered as the top-level namespace. No other variables or objects than 's4a' will be injected into the global namespace.

All modules shall be implemented using the JavaScript module design pattern. All private variables and methods shall be encapsulated by the module. Only public methods and variables will be exported to the s4a object.

\begin{center}
\textbf{Code fragment 2: JavaScript OOP module design pattern}
\end{center}

\begin{verbatim}
s4a.maps = (function() {
    var privateVar = 1;
    var publicVar = 2;
    var privateFunction = function() {
        return privateVar + 1;
    };
    var publicFunction = function() {
        alert('public');
    };
    return {
        publicVar: publicVar,
        publicFunction: publicFunction
    }
})(
\end{verbatim}

\textsuperscript{2} "Git (software) - Wikipedia, the free encyclopedia." 2005. 21 Mar. 2015  
\texttt{<http://en.wikipedia.org/wiki/Git_(software)>}

\textsuperscript{3} "Scrum (software development) - Wikipedia, the free ..." 2013. 21 Mar. 2015  
\texttt{<http://en.wikipedia.org/wiki/Scrum_(software_development)>}

\textsuperscript{4} "Design pattern - Wikipedia, the free encyclopedia." 2004. 21 Mar. 2015  
\texttt{<http://en.wikipedia.org/wiki/Design_pattern>}

\vfill
All classes and public methods shall be commented according to the JSDoc\(^5\) “standard”, permitting for automatic generation of API reference documentation.

### Code fragment 3: JSDoc code comment standard

```javascript
/**
 * Description of method
 * @param {data type} paramName Description of parameter
 * @returns {data type} Description of return value
 */
```

#### 2.2.2 Common repository layout

The Git repository will have the following directory and file-layout:

- **dep** (external dependencies, JavaScript libraries)
  - JQuery
  - crossfilter
  - OpenLayers
  - [others as needed]
- **dist** (compiled versions of the development, generated by Grunt\(^6\))
  - s4a.js (a single-file human readable version of the JavaScript library, for development/debugging purposes)
  - s4a.min.js (minified version of above, for production)
  - s4a.css (a single-file human readable version of the stylesheet, for development/debugging purposes)
  - s4a.min.css (minified version of above, for production)
- **src** (all the development source code goes here)
  - css (Cascading Style Sheets)
    - s4a.src.css (any CSS styling required by the s4a.js)
  - js (JavaScript)
    - [module name] (one dir per module)
    - s4a.js (top level object declaration)
- **tests**
  - [one dir per module]
  - s4a.html (tests for the top-level object)
  - index.html (HTML index to the test files)
- **gruntfile.js** (a task configuration file for the JavaScript Task Runner software Grunt)
- **readme.md** (a high-level description of the library including licenses, relationships to other libraries and other constraints in Markdown\(^7\) format)

#### 2.3 Testing

The s4a.js library will have its own tests that will be developed alongside the source code. These tests will cover the functional testing requirements, confirming that all functions are present, working and performing well. Performance shall be measured as the execution time of the individual operation, excluding the HTTP request response time with its network latency overheads and other third-party constraints beyond our control.

---


The integrated tests will be completed by testing through the pilot applications. These tests will emphasize the ease of use from the perspective of (third party) application developers as well as the usability as experienced by application end-users.

It is envisaged that the testing will reveal commonly recurring chains of API requests that application developers would like to chain into single requests in order to reduce performance degrade as a consequence of network latency.

The output from the testing process will be fed back into the development process and added to the product backlog, to be divided into suitable chunks for individual sprints. Cross-module functional considerations will be identified and agreed upon during periodic coordination meetings and during informal day-to-day electronic interaction between the development teams of SDI4Apps partners.

2.4 Deployment

The s4a.js library can be downloaded and hosted within the source trees of individual web applications. A versioning attribute will be maintained on the top-level object to aid application developers in using the system. No methods may be removed from one major release to the next - but methods may be marked as deprecated, allowing third party application developers to identify invalid/expired method calls in their code through their preferred IDEs.

All incremental version of the Advanced Tools API JavaScript library will be hosted in each instance of the SDI4Apps platform. Thus, application developers working against a specific SDI4Apps instance can include the library in their projects by including the <script/> and <link/> tags shown in the following HTML snippet:

<table>
<thead>
<tr>
<th>Code fragment 4: Including the s4a.js library in an HTML5 application</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;html&gt;</td>
</tr>
<tr>
<td>&lt;head&gt;</td>
</tr>
<tr>
<td>&lt;script type=&quot;text/javascript&quot;</td>
</tr>
<tr>
<td>src=&quot;<a href="http://platform.sdi4apps.eu/s4a.min.js?v=1.0%22%3E">http://platform.sdi4apps.eu/s4a.min.js?v=1.0&quot;&gt;</a>&lt;/script&gt;</td>
</tr>
<tr>
<td>&lt;link type=&quot;text/css&quot; href=&quot;http://platform.sdi4apps.eu/s4a.min.css?v=1.0&quot; rel=&quot;stylesheet&quot;/&gt;</td>
</tr>
<tr>
<td>&lt;/head&gt;</td>
</tr>
<tr>
<td>&lt;body&gt;</td>
</tr>
<tr>
<td>...</td>
</tr>
<tr>
<td>&lt;/body&gt;</td>
</tr>
<tr>
<td>&lt;/html&gt;</td>
</tr>
</tbody>
</table>

Please note the version attribute, v=1.0, in the example above. If omitted, the latest version of the s4a.js library will be returned. If included, it will return the specified version. This permits for backwards compatibility.
3. THE BUILDING BLOCKS OF S4A.JS

The objective of the s4a.js is to make it easy to build feature-rich and powerful applications based on the SDI4Apps platform. To do so, the library build on existing, well proven infrastructure libraries that provide ready-to-use solutions to a number of the challenges in the platform.

The diagram below shows the libraries, or packages, that s4a.js relies on. Each library serves a unique purpose in s4a.js and there is no duplication between them.

The core library that s4a.js will build on is HSLayers NG\(^8\), this in turn is an extension of the widely used OpenLayers 3\(^9\) slippy-map library that enable JavaScript developers to create Google Maps similar applications using a variety of public and private GIS data sources. Spatial data are key to SDI4Apps and these two libraries therefore constitute the most important dependency of s4a.js.

In order to visualize data, s4a.js uses D3.js\(^{10}\), a sophisticated library for visualization of data-driven documents that renders a wide range of data graphics as SVG.

---


JavaScript, sadly, is not always the same JavaScript. Methods and implementations, especially for new or experimental functionality, differs to some degree across Internet browsers and in order to provide predictable, harmonized behaviour in a cross-browser environment it is necessary to rely on jQuery\(^{11}\), a cross-browser compatible development framework.

Finally, s4a.js must also work on and exploit the capabilities of mobile devices. Once again, HTML5 and JavaScript offers methods to access sensors such as GPS and camera - but the support across browsers is very limited.

When implementing mobile applications we can choose between three different types of apps:

1. *Pure* HTML5 and JavaScript applications
2. Hybrid applications consisting partly of Web Views and partly of native code
3. *Pure* native code applications targeting

Philosophically, the best approach would seem to be the first one - but the pragma of present day reality dictates that s4a.js must employ the second type of apps; hybrids. The challenges that require us to go *hybrid* are persistent offline storage and unified access to mobile device sensors such as camera and GPS.

There are many frameworks for developing hybrid applications; typically, these permit the developer to write user-interface code and program logic using JavaScript, extending the latter with additional functions that provides access to native OS features.

For the purpose of the SDI4Apps project, we have chosen to work with the Apache Cordova\(^{12}\) framework, an open source project based on donated code from Adobe’s popular PhoneGap application.

4. 5 ADVANCED TOOLS API MODULES

Figure 3: The modules of the s4a.js client-side JavaScript library

4.1 Map module

With SDI4Apps, as with most other spatial related projects, more than 50% of the pilot user requirements can be fulfilled by simply showing a map on the screen, populating it with a few datasets.

Consequently, a very simple - but very useful - component in the s4a.js map module will be a helper class that makes it easy to create a map interface and add data and tools to it.
This helper class will enable simple functions such as creating a map interface and adding it to an existing DOM element in the HTML mark-up. It will be possible to add data to the map by providing the persistent URI\(^\text{13}\) of datasets that are available in the connected instance of the SDI4Apps platform.

In order to provide interactivity in the map, the developer can add tools that are invoked either when the user clicks a button (button) or after a user provides spatial input in the map interface by drawing a point, line or polygon (tool).

These are basic, almost naive, functions - yet they are extremely useful and will leverage the time and effort required to implement a new prototype significantly. The resulting map objects will inherit all properties of an OpenLayers3/HSLayers NG map object.

### 4.2 Information retrieval module

In this module we provide a class helper (QueryHelper) to perform search based on Search Service described in D3.3.1 deliverable section 5.1 Search Service

The methods exposed by this class allow the user to search objects in a dataset by three criteria:

1. full text search
2. geographical criteria
3. facet criteria

Please refer to D3.3.1 deliverable for full explanation of the faceted search mechanism.

---

\(^{13}\) "Persistent uniform resource locator - Wikipedia, the free ..." 2006. 14 Feb. 2015

<http://en.wikipedia.org/wiki/Persistent_uniform_resource_locator>
**Figure 5: Overview of helper class for issuing queries to the server-side OpenAPI**

The helper class exposes the following methods:

- `QueryHelper.createQuery(dataset)`: create an instance of a QueryHelper based on the specified dataset and returns it. In the rest of the calls `<instance>` represent an instance returned by this method.

- `<instance>.setFullTextSearch(string)`: set the string as full text search criteria. Returns the QueryHelper instance for chainability.

- `<instance>.setDistance(geom, distance)`: add the geographical criteria to the query. The distance of the results from the geometry specified in first parameter must be less than the one from the second parameter.

- `<instance>.addFacet(facet)`: add a facet field to the query criteria. The field specified, with the relative values, will be returned as a facet by the service when the query is issued. Returns the QueryHelper instance for chainability.

- `<instance>.removeFacet(facet)`: remove a facet field from query criteria. Returns the QueryHelper instance for chainability.

- `<instance>.filter(facet, value)`: add a criteria based on the value of a facet. Returns the QueryHelper instance for chainability.

- `<instance>.query()`: performs the query using the criteria created by previous method calls. The service is called and his JSON response is parsed into a Javascript object and returned.

### 4.2.1 Implementing facetted browsing

While the service exposed by the Search Service endpoint is stateless, with the query helper class the application can perform a stateful sequence of queries.

In a single page application context, the QueryHelper instance can be created when the page is loaded and used for subsequent queries.
In a typical scenario after a query is issued the results are collected and shown to the user along with facets. When the user wants to add or to remove a facet filters, the application can invoke the `addFacet` or `removeFacet` method on the existing instance, and then resubmit the query with the `query` method.

The class instance is responsible to hold the state of the faceted browsing.

### 4.3 Advanced visualization module

The more innovative part of s4a.js is the advanced visualization module that defines the visualization objects, as shown in the diagram below.

![Diagram of chart and map based visualization objects](image)

**Figure 6: Overview of chart and map based visualization objects**

The objectives that the visualization methods are to meet include:

- The ability to handle big web scale datasets
- The ability to visualize patterns in data in attractive and effective ways
- The ability to create coordinated views that show several dimensions of the same dataset

There are two distinctive type of visualization objects. On one hand, there is basic **charts** that can visualize one or more series of data in a non-spatial context. On the other hand, there are **map-based** visualizations that present one or more series of data on a map surface.

#### 4.3.1 Chart types

The charts will be drawn using D3.js. This library does not per se provide ready-to-use methods to put a data series on a chart - but it provides very detailed control over how the visualization will be generated.

**Pie Chart**

Pie charts are merely a circular chart showing $n$ dimensions as (pie) slices whose size are determined by the percentage/fractional value of the full circle (100%).
Bar Chart

Despite being, perhaps, the simplest and most common form of diagrams, bar charts have undergone a revival in the age of big data when they are used to display huge quantities of data, typically trying to reveal some pattern as a histogram.

Scatter Chart

Another basic form of visualization where $n$ phenomena are plotted on a map along two dimensions $x$ and $y$. 
Line Chart

Basic visualization type where a line graph is showing the development of \( n \) (= number of lines) phenomena along a dimensional axis.
4.3.2 Map types

The maps will be drawing using D3.js and HSLayers NG via the WebGLayer extension to the latter. WebGLayer enables drawing of advanced graphics in geographical space using WebGL\textsuperscript{14} and the capabilities of the GPU\textsuperscript{15}, this provides for far greater performance than the much slower CPU drawing operations. Hence, the capacity of WebGLayer by far exceeds that of a regular vector layer.

**Symbol Map**

The symbol map is a simple cartographic map that can include a number of thematic layers presented on top of a tile-based base map.

![Energy Consumption by European Countries](image)

*Figure 11: Example of graduated symbol map*

**Bubble Pie Map**

The bubble pie map is a combination of pie charts and a map. The pie can show one or more dimensions of the data and can additionally express relative volumes by varying the size of the pie according to another dimension.


Heat Map

A heat map is a way of showing concentration of measured observations as hot areas on a map. The effect is typically using a colour scale from blue (cold) or transparent to red (hot).

The aggregation of hot areas can be by performing frequency count of occurrences of data within a grid cells, hex bins or point locations. Depending on the number of occurrences of e.g. point features that falls within a virtual grid cell, the cell is assigned a weight that is later used to visualize the grid showing heat where there is a large number of observations in a single cell.

Figure 12: Example of map combining a bubble and pie charts
D4.2 Advanced Tools API Design

Figure 13: Example of heat map

**Prism Map**

A prism map is a combination of bar chart and map where, instead of rectangular bars, map features are extruded to a height that is relative to the chosen dimension. For an example, a map of municipalities may be extruded according to the number of inhabitants.

Figure 14: Example of prism map
Choropleth Map

This is the simplest type of statistical map where a single dimension of data (that can be related to a polygon dataset) is shown as varying colours.

Figure 15: Example of choropleth map

Live Data Map

This map type will poll a remote data source at regular intervals, reload data and redraw the content of the map in order to reflect changes. Useful, for an example, for real-time sensor observations or fleet control.
Coordinated Views

Based on the concepts underpinning the Crossfilter\textsuperscript{16} library, the s4a.js advanced visualization module will include the possibility to create coordinated data views that employs several interlinked visualization objects to visualize several dimensions of the same data simultaneously.

When a “filter” is set on one of the visualizations, the other visualizations will be updated to reflect the limitation on the underlying data source that is common to all visualizations. Thus, it is easy to see the significance of each variable in a complex model.

Figure 17: Example of coordinate view incorporating line charts, bar charts and maps

4.4 Mobile module

S4a.js is not going to be the final (pilot) applications, the mobile module is a toolkit that makes it easy to add SDI4Apps platform functionality to any mobile application - and it will be used to implement the pilot applications.

As described above regarding the map helper class in the chapter describing the advanced visualization module, many of the user requirements can be satisfied by a simple interactive slippy map with one or more layers of business data draped on top of it. This functionality is shared across platforms and is thus not part of the mobile-specific module.

The unique challenge of the Mobile module is related to two aspects that are specific to mobile devices:

1. The screen real-estate is limited and navigation paradigm is very different; user interfaces must be adapted to meet these constraints
2. Connectivity cannot be assumed persistent and the API methods must be capable of being used in offline user scenarios.

For this reason, the solution that is outlined in the diagram below is centred on this single purpose; namely how to display and edit map data while in offline mode.

---

Figure 18: Solution to the offline editing problem

The core parts of the mobile module are two classes that represent:

1. A vector dataset, downloaded to the mobile device as GeoJSON\(^{18}\), that shall be possible to edit or add features and properties to while offline
2. A base map, that is either available via a web or local device URL pointing to map tiles.

Maps will be shown in the same way on mobile devices as on any other devices. The difference will be that user controls and interfaces must be adapted to the mobile platform, perhaps through some form of custom skins.

The Cordova framework will render the user interface of the application as full-screen Web Views\(^{19}\)\(^{20}\) that are running within a native wrapper capable of accessing native device features on Android, iOS, Blackberry and Tizen platforms.

Therefore, OpenLayers 3 and HSLayers NG will be responsible for the rendering, also on mobile devices. The challenge lies in how to make the data interoperable in an offline environment. Here we deal with base map tiles and editable vectors separately.

With respect to base map tiles, by default these can be loaded from any of the mainstream web map providers such as Google, Bing, Yahoo! or OSM. However, if you want to have the data available offline, the most efficient way to achieve this is to pre-generate a tile map service (TMS) cache for the area you plan to visit while offline, and preload that cache onto your mobile device.

A suitable mechanism for making TMS caches portable is the MBTiles format\(^{21}\), a simple SQLite based data model that stores each tile as a blob. A service in the server-side API must permit the generation and download of MBTiles data for user specified extents or buffers around lines or points.

Having solved the issue of base maps, the next problem is offline editing of vector data. Downloading and caching the original data is a trivial task that can be achieved as simple as by adding the data to the cache manifest file that is referenced as an attribute of the `<html/>` element in your HTML5 markup.

The tricky bit is that each of the features in the original file can be modified or deleted - and that new features can be added. The dataset that the features belong to must however be possible to display on a map. It must be possible to show both the original version - and the modified current version.

This requires persistent storage of feature edits while the application is in offline mode - and it requires access to a query mechanism that makes it possible to correlate original features with edited ones and determine which to include and exclude. This is provided by the WebSQL\(^{22}\) mechanism of HTML5 that is widely supported on mobile platforms and that is exposed in a unified way through Cordova API plugins.

Once online again, the offline edits must be synchronized with the dataset hosted in the SDI4Apps platform through pushing the edits to a web service. The web service interacts with the original data source and creates a work list for the dataset owner who must approve or reject the edits in an end-user application related to the data management in the platform instance.

### 4.5 Analytics and modelling module

This module enables the running of predefined simulation and/or analysis models on datasets that are (1) available in the SDI4Apps data repository and (2) conform to well-known data models.

---

D4.2 Advanced Tools API Design

The analytical and modelling module is inspired by the achievements of the Plan4business project. That project demonstrated that it was feasible to allow users to perform complex analysis operations on the server-side using components from the same technology stack that is used in the SDI4Apps platform.

By complex analysis, we understand analysis that combines multiple spatial and attribute queries in algorithmic chains where the output from each step feeds into the next. This is distinctively different from common analysis methods like proximity, free-text or single attribute filter expressions.

Analytics and modelling services are “border cases” that are targeting specialized use cases. As such, this module is at odds with the API design constraint that only highly reusable functionality will be integrated centrally. The motivation for bringing this module forward into SDI4Apps is to overcome the limitations and bottlenecks that were identified by Plan4business when evaluating the server-side analysis modules.

Whereas a typical API request accepts a simple query and returns a simple response synchronously, advanced analysis and models require so long time to run that they must be (1) processed asynchronously. Additionally, the analysis models consume (2) more RAM memory and (3) more CPU time than typical requests - and since the process and output is based on user supplied parameters, (4) little or no data can be pre-cached to improve performance.

This is ok when the process is being run on a single user workstation. With great likelihood, the user is aware of the time and resources required and is able to wait for the workstation to complete its processes before it becomes responsive/usable again. This is different in a client-server environment. Here, what one client does affects the performance experienced by all other users.

By implementing the analysis module on top of a cloud platform that can be scaled up to improve both overall and per-request capacity, SDI4Apps may remove several of these obstacles by allowing dedicated nodes to handle comprehensive analytics processes without affecting the response of concurrent API requests. This is essential as otherwise large-volume data requests such as map tiles would experience a severe performance degrade while background analytics processes were running.

In order to ensure that the analytics and modelling module has the potential for reuse, the module will target data conforming to one or more INSPIRE Annexes I-III data models - or other well-known and widely adopted data models.

The availability of INSPIRE compliant data is presently limited - but may safely be envisaged to improve in the near future. Analysis and simulations that are targeting these data models will therefore have a significant re-use potential throughout Europe.

Where Plan4business had a focussed thematic domain centred on land-use and real estate, SDI4Apps is content agnostic and aims to establish a flexible and reusable platform that can be used with any data. For this reason, the project has yet to identify the algorithms and relevant data sources that will used as input for the analytics and modelling module. This will be done prior to the first release of the Advanced Tools API and the specification will be updated into this document. The updated version of this document will then annexed to deliverables D4.3.1 and D4.3.2 at the time of their submission.
4.5.1 Example of an analytics and modelling operation

For the benefit of the readers, we include an example of an analytics and modelling module method step-by-step. It is worth noting that this is merely for illustrational purposes and that the actual analysis methods implemented by the API may be entirely different.

The purpose of this example analysis is to identify the area characteristics and indicative sales value of a piece of real estate, identified by a polygon. The analysis relies on user-supplied parameters in addition to well-known data models for (1) transportation, (2) land-use and (3) cadastral data and (4) community services.

1. The user invokes the module through clicking a button in an application user interface
2. The user provides input (in this case a property polygon) through an input wizard.
3. The user executes the analysis by clicking a button in the input wizard.
4. The Advanced Tools API relays the request to the OpenAPI and polls the API for status updates at a predefined time interval.
5. First, the polygon is used to select neighbouring properties within a buffer of 1 km.
6. Transaction prices for properties are extracted, weighted and averaged for all properties.
7. Second, the distance from the polygon the nearest schools, kindergartens and other community facilities are calculated as distance along road.
8. Finally, the results are fused into a common response object that is passed back to the Advanced Tools API that has continued to poll the OpenAPI for a response.
9. The response is formatted into HTML elements and rendered in the client application.
5. SUSTAINABILITY CONSIDERATIONS

Sustainability planning is beyond the scope of this work package and deliverable, but even at the time of design and implementation it is necessary to pay attention to the possible sustainability strategies of the project outcomes - beyond the time-period when funding is provided by the project - and the resources are needed for the maintenance and further development are readily available.

There are three different scenarios for the sustainability of s4a.js:

1. The JavaScript library will be strongly tied to the SDI4Apps platform and will be further developed and maintained through resources derived from operational income from the overall business model of the platform.
2. Portions of the library must be made independent of the SDI4Apps platform and become widely used as a supporting utility library alongside OpenLayers, Crossfilter, D3.js and other libraries that are ubiquitous among HTML5 application developers.
3. The library may be sliced into pieces that are suitable for contribution to and integration into other Open Source projects, particularly the JavaScript library that s4a.js depends on such as OpenLayers, HSLayers NG etc.

Unless these three strategies are kept in mind from the outset, it is easy that the coding practices closes the door to anything but strategy #1. While it is an overall objective of the SDI4Apps project to establish a sustainable business model for the overall platform, betting everything on this strategy means a binary win/lose scenario. It is wiser, especially as it has no cost-implication, to observe the alternative sustainability strategies at the time of implementation and then have a wider range of options for commercialization and further usage.
6. APPENDICES

Appendix A: Advanced API element descriptions