The management and distribution of multiform communication in mobile applications: The case of the Mikael Agricola application

Lingsoft has been outlining a publishing platform for content production, the purpose of which is to generate and manage the distribution of multiform content for different purposes through an API. The project is built on the One Hour Words concept, currently published in book form, which is to be digitalised. One important goal has been to combine the following key ideas: the structuring, enrichment, updatability and multi-channel distribution of content combined with Lingsoft's language services, such as translations, term management and services related to availability and accessibility. Furthermore, the data content must be dynamic in relation to time and place, and it must enable updating without application updates or measures taken by the user.

Lingsoft wants to share this concept with the public, with the aim of finding parties in need of content production services or interested in content production, and who could join in to collaborate on the further development of the concept. Lingsoft can offer the solution as it is for the foundation of implementation, implement the content management service either itself or through collaboration, or participate as a language technology provider in the enrichment of the content in the content management service.

Objectives

Lingsoft’s goal was not so much to implement the application or determine its features in detail, but rather to test the actual concept through the use of the application. The objective was to outline and produce a content management mechanism and an application and communication framework, which would also be generic enough to be applicable to a diverse range of use cases. The implementation and operation of these features will ensure that there is a feasible technical distribution and utilisation model for content produced with a content production service.

The implementation of the concept posed the following concrete goals:

- **The content will be multiform** (for example, multilingual)
  - maps
  - positioning and destination data
  - topical information, such as news, offers, events, advertisements
  - functionalities that combine the positioning data and topical information
  - images
  - descriptions of a topic or theme

- dictionaries related to a specific subject matter
- question and answer pairs

- **The contents must lend themselves to a structural description**, so that the content can be managed and interoperability between systems can be ensured. However, the structure should not be too confining, so that users can be offered the information they are looking for in the format they need. Furthermore, the structure must be flexible for new content requirements.

- The cross-system **communication structure and procedures** shall be defined to distribute contents,

- The system **must allow for content enrichment**, i.e. the content management system must support the management of metadata and semantic data, as well as different standards,

- **It must be possible to manage content** with the applicable processes, i.e.
  - production management
  - the storage, organisation and utilisation of content
  - content distribution
  - updating content in data terminal equipment and target systems

Concept testing

To concretise the outline and test the plan, a mobile-based version of the Mikael Agricola application was produced in 2017, combining the positioning function with the provision of content and topical information on the basis of the location data. The application content is produced through the Lingsoft content management service (a concept test version that is not viable for production).

To describe the content, a JSON data structure was defined, which in itself already allows for a highly versatile range of content description and distribution. Furthermore, JSON is a clear and light-weight format that can also be used as a communication structure. It is also well-known and widely used and primarily flexible in terms of changing needs, such as adding semantic and linked data on the basis of use cases and needs that will only be specified later.

The content is produced on the Lingsoft server, and it is then forwarded to the application through a simulated API interface over an https connection. The content is read and updated in the customer application which interprets the markings and time stamps added to the content in version management and is resilient in terms of communication and contents. The application logic contains simple and generally applicable basic functions for interpreting the communication structure and content.

The aim was to design the application logic so that it makes a distinction between functionalities and information content and communication, enabling reutilisation in various sectors.

The content and communication structure

The following terms are referred to in the implementation process (in this case, the objects of the data structure).

- project – A general concept that ties together all the information related to the application. The project identifier can be used to track all version data, the owner of the data, the producer of the data, contracts, a description of the data content, terms and conditions, and so forth, which, in turn, are described in the content production system. Each project has an individual identifier.
- location – The destination on the map as indicated by the positioning data (longitude, latitude)
- resource – Any online resource, including files, schemes, interfaces, websites. Each resource has a specific URL.
- event – Any news-like content that includes a headline, time, body text etc., for example an event, offer, advertisement, or notification
- dictionary – A multiform information structure in keeping with the OHW concept, which can be used to depict a glossary's language versions, name, description, and term + definition pairings. A definition, on the other hand, can contain links, references to resources, places, and some style choices.
- quiz – An information structure containing combinations of questions, alternatives and answers, which is used to implement a quiz.
- language version (appLang) – The terms and language of the application interface.

In this context, event, dictionary and quiz refer to objects needed in the application in question. They serve as examples. The data structure in itself does not restrict or dictate what objects can be presented, and the data structure can be expanded to cover any desired items. The rest of the objects are more universal and can be applied to different use cases as such.

The data content is described with the following type of object structure in JSON format.

Project
    Id
    Locations
        location1
        location2
        …
    Maps
        resource1
        resource2
        ...

Events
event1
event2
...
Document
Sections
Dictionary
dictionary1
dictionary2
…
Quiz
Question-alternatives-answer set1
Question-alternatives-answer set2
…
Application language versios
appLang1
appLang2
…..

Please note that the description above and below are the original outline of the data structure. Below, you will find links to the files available for downloading. These files contain the data structures in their final form, which may deviate slightly from those presented here, as the structure was specified and modified for the final implementation.

Object data structure

The data content of the project

```
project: {
        id: 1, // application id
        name: ”Agricolan ajan Turku”,
        language: ”fi”
        dataset: 1,
        lastMod: 2017-08-14
}
```

The dataset (= version number) is only linked to the project. If there is a larger number available on the server than in the application, all data from the server is retrieved and added to the application. If there is a lot of data, the dataset can be linked to the object level, such as the dictionary, for individual applications.

Data content on location

The application shows several locations, so this is a multivalued feature. Each location means a destination that can be searched for in the application and that is shown on the map. When a user gets close to a destination, it opens up in the Destinations menu images. When location data is added to notifications, these destinations are also shown on the map.

```
locations : [
{
        project: 1, // application id
        language: “fi”,
        name: "Mikael Agricolan patsas",
        description": ”Tämä patsas on pystytetty….”,
        location: { "latitude": xxxxx, "longitude": yyyyyy }
 } ,
{…}
]
```

The sata content of the online resource (afile, scheme, interface, website...)

The application can contain references to different online resources, so the online resource object is multivalued. An individual resource can be referred to in the form <resourse[id]>, where the tag is replaced, for example, in the body text of a notification with a similar URL address that complies with the resource type (datatype), such as .

```
resources : [
{
        project: 1,
        id: 1, // resource id, used to refer to the resource within the application
        language: “fi”,
        url: “http://….”,
        datatype: “file”,
} ,
{…}
]
```

The content of topical information (event, offer, advertisement, notification...)

The aim was to include one data structure that can be used to describe different types of news; event, offer, advertisement, notification etc. Data that describes the news and is applicable to the situation is selected and utilised.

events: [
{
project: 1,
language: “fi”,
dtPubl: “2017-08-01”, // publication date, required
dtStart: “2017-08-15”, // event start date
dtEnd: “2017-08-15”, // event end date
dtArchive: “2017-10-30”, // archive date. If not given, always in the event list
dtHide: “2018-01-01”, // if given, hidden after this date (e.g. offer)
title: “Agricolan muistolaatan paljastus”; // title
summary: “Keskiviikkona 9.8.2017 paljastetaan Agricolan muistolaatta Turun....”, // event list summary text
img_small: {resource}, // e.g. small image with the event list summary
location: “Turku”, // location name
url: [“http://www….”, http://www….], // list of URLs, e.g. at bottom of the event body
duration: “08:00”, //hh:min, event duration
rDate: NULL, // recurrence date
rRule: NULL, // recurrence rule
category: “Tiedote”, // e.g. News, Event, Offer, Advertisement
description: “Turussa järjestetään….” , // formatted content
geo: { "latitude": xxxxx, "longitude": yyyyyy } // if location is referred
} ,
{…}
]

Document

Document refers to a data structure entity, which is in itself structural. For example, a document can describe a book, such as One Hour Words in a printed version. A document contains sections, which, in turn, can contain different types of subsections. In this example, we have only used a dictionary inside a document, but the type value can also be used to present content such as front page, introduction, a table of contents, body text, a quiz, etc.

"document":
{
"name": "Agricolan ajan Turku", // document name
"type": "dictionary", // document type
"project": "1",
"languages": ["fi"], // supported languages
"LanguageTextPair": [// dokumentin names per language
{
"language": "fi",
"text": "Agricolan ajan Turku"

```
                }
            ],
        "id": "1", // document id, order number
        "sections": [  // document sections
            {
                        "type": "dictionarysection",  // section type
                        ”name": "dict", // section name
                "bool": "1", //section id, order number
                        "index": "true",  // is in index or not
                        " LanguageTextPair ": [  // section language versions
                        {
                                "language": "fi",
                                "text": "Agricolan ajan Turku -sanasto"
                        }
                ],
                ”dictionaryEntry”: […] // contents of the section, can also be e.g.
cover, index, body,..
            }
}
```

Dictionary

The dictionary is an example of structural content. It is a structured presentation of term definition pairs for each language version.

```
"dictionary": [
{
        "id": "001", // id or order number
        "versions": [ // language versions
        {
        "language": "fi",
        "term": "9 kpl",  // name of the term in the dictionary
        "definition": "Agricolan teosten lukumäärä. Saavutus on kunnioitettava, sillä työ on tehty
kymmenessä vuodessa ja sisältää hyvin laajoja teoksia. Käännöstyön Agricola teki muun työn
ohessa ja pitkälti omilla varoillaan."
        }
        ]
},
{
        "id": "002",
        "versions": [
        {
        "language": "fi",
        "term": "60%",
```

```
        "definition": "Prosenttiluku, joka kertoo, kuinka suuri osa Agricolan käyttämästä
sanastosta on edelleen käytössä. Näin ollen hänen kanssaan pystyisi helposti keskustelemaan,
jos hän sattuisi kadulla vastaan. Agricolan Rukouskirjan esipuheen sanoin: ”Kyllä se kuulee
suomen kielen, joka ymmärtää kaikkein mielen."
        }
        ]
},…
]
```

Quiz

Another example of structural content is the quiz, which here is a structured presentation of a triad consisting of a question, alternatives, and the correct answer.

```
"quiz": [
{
        "id": "001", // id or order number
        "versions": [ //language versions
        {
        "language": "fi",
        "question": "Missä Mikael Agricola syntyi?",  // the question
        "alternatives": ["Turussa", "Perniössä", "Viipurissa"], // answer alternatives
        ”correct”: "Pernajassa", // correct alternative
        }
        ]
},…
]
```

According to the examples presented above, the document data structure can contain a highly versatile range of structural content, of which the dictionary and quiz are just individual examples.

Application terminology according to language versions

Managing the language versions of the application is also an essential feature of content management. The application may be available in many different languages and the selection can be expanded later, in which case the application terminology will be presented in a separate language version data structure.

```
"appLangs": [
{
        "project": 1,
        "version": 1,  // language version id
        "language": "fi",
```

```
        "terms":
        [
{
"tag": "open_map",  // internal reference name
"term": "Avaa kartta"  // term used for the languagei
},
{
"tag": "locations",
"term": "Kohteet"
},…
        ]
},…
]
```

Interface calls in the content service

The content management system outlined by Lingsoft includes an interface, to which calls can be made in the future to retrieve, store, and modify data. Below, you will find the calls needed in the “Agricolan ajan Turku” (“Turku in Agricola’s Era”) application to read data.

In the execution, a GET call is sent to the content management server’s interface, which returns the project information. The latest dataset value (a whole number) can be read from this data. For example,

https://www.lingsoft.fi/content-api/proj/1/projdata
which consequently returns the data in instance number 1 in the object “project”, e.g.

```
project: {
        id: 1, // identification for the application
        name:”Agricolan ajan Turku”,
        language: “FI”,
        dataset: 2,
        lastmod: 2017-08-14
}
```

If there is a later dataset number on the server than that in the application, the application will read all the project data with the GET call, e.g.

https://www.lingsoft.fi/content-api/proj/1/dataset/2

which returns the information in dataset 2 of project 1.

Because the data package of the entire project is small in this specific application, the whole package is returned as a single entity. If it is later desired to use restricted calls in order to reduce unnecessary network traffic, the calls could, for example, be formulated as follows:

https://content.lingsoft.fi/api/proj/1/events (all news-like items)
https://content.lingsoft.fi/api/proj/1/dictionaries
https://content.lingsoft.fi/api/proj/1/locations
etc...

Source code

From the following links, you can download the files that implement the data structures described above in their final form.

DataPackage.cs	object specification
ContentManager.cs	reading the data file

Example of a data file that is loaded from the simulated API service.

Content management service

The next stage in the implementation is to outline the architecture and required functionalities of the content production environment. At least the following sectors have been defined as needing more detailed specifications:

- lifecycle management of the delivered contents, including e.g. customer, order, and delivery management
- tools for the acquisition, production, and modification of content
- a content storage architecture and search methods
- the enrichment of contents, such as language versions, dictionaries, semantic data, linked data
- methods for content collection and design, such as a presentation format that conforms to different standards in accordance with the target and purpose of use
- content distribution methods in accordance with the requirements of different distribution channels
- monitoring, reporting, and invoicing methods related to the use of content

The design and implementation of the content management service is included in Lingsoft's development plan.

Additional information: Simo Vihjanen, Jari Kesti, firstname.lastname@lingsoft.fi