PLAN4BUSINESS

A service platform for aggregation, processing and analysis of urban and regional planning data

SEVENTH FRAMEWORK PROGRAMME

Project number:	296282
Project acronym:	*plan4business*
Project title:	A service platform for aggregation, processing and analysis of urban and regional planning data
Instrument: **Call identifier:** **Activity code:**	STREP FP7-ICT-2011-SME-DCL

Start date of Project:	2012-04-01
Duration:	24 month

Deliverable reference number and title (as in Annex 1):	**D5.1 Interim Report on Integration, Analysis and Storage Engines**
Due date of deliverable (as in Annex 1):	**M12**
Actual submission date: **Revision:**	*see "History" Table below*

Organisation name of lead contractor for this deliverable:
University of West Bohemia (UWB)

Project co-funded by the European Commission within the Seventh Framework Programme (2007-2013)		
Dissemination Level		
PU	Public	X
PP	Restricted to other programme participants (including the Commission Services)	
RE	Restricted to a group specified by the consortium	
CO	Confidential, only for members of the consortium (including the Commission Services)	

European Commission Information Society and Media

Title:
Interim Report on Integration, Analysis and Storage Engines
Author(s)/Organisation(s):
Jan Jezek, Michal Kepka, Tomas Mildorf / UWB Eva Klien, Simon Templer, Robert Gregor / Fraunhofer IGD Michal Sredl, Jachym Cepicky / HSRS
Working Group:
WP5
References:
Grant Agreement No. 296282, Annex I Description of Work

Short Description:
This deliverable summarises the work performed and progress achieved within WP5. This is an interim report demonstrating the progress of WP5 development towards the goals given by the Description of Work and stakeholder requirements. This report describes server side engines for the *plan4business* platform including integration, analysis and storage engines, and API and access control system.
Keywords:
integration engine, storage engine, analysis engine, API, *plan4business*, access control

History:

Version	Author(s)	Status	Comment	Date
001	Tomas Mildorf	new	outline of the report	15.12.2012
002	Tomas Mildorf	draft	Introduction and methodology	05.03.2013
003	Robert Gregor	draft	Input for sections 5 and 7	13.02.2012
004	Tomas Mildorf	draft	Integration Engine and terminology	10.03.2013
005	Jan Jezek	draft	Analysis Engine API	17.03.2013
006	Michal Sredl	draft	Chapter 9	20.3.2013
007	Michal Sredl	draft	Chapter 9 – Next Steps	21.3.2013
008	Jan Jezek	draft	Chapters 7 and 9 - revision	22.3.2013
009	Eva Klien, Simon Templer	draft	Chapters 6 and 8 - revision	22.3.2013
010	Tomas Mildorf	draft	List of acronyms, minor revisions	22.3.2013

011	Tomas Mildorf	final draft	Supported formats for the pool data API, Git repository, minor revisions	25.3.2013
012	Tomas Mildorf	final	Final revisions reflecting the reviewer's comments and additions	29.3.2013

Review:			
Version	Reviewer	Comment	Date
011	Stepan Kafka	Final review of the document, a separate quality review form provided	29.3.2013

Table of contents

List of acronyms

ACID	Atomicity, Consistency, Isolation and Durability
API	Application Programming Interface
AVINET	Asplan Viak Internet
CAS	Central Authentication Service
CDDA	Nationally Designated Areas
CLC	Corine Land Cover
CSV	Comma-Separated Values
FP	Framework Programme
GIS	Geographic Information System
GMES	Global Monitoring for Environment and Security
GML	Geography Markup Language
HALE	HUMBOLDT Alignment Editor
HILICS	Hierarchical INSPIRE Land Use Classification System
HSRS	Help Service Remote Sensing
HTTP	Hypertext Transfer Protocol
INSPIRE	Infrastructure for Spatial Information in the European Community
ISO	International Organisation for Standardization
ISOCARP	International Society of City and Regional Planners
JSON	Javascript Object Notation
KML	Keyhole Markup Language
LDAP	Lightweight Directory Access Protocol
OGC	Open Geospatial Consortium
OSM	OpenStreetMap
OWL	Web Ontology Language
OWS	OGC Web Services
PDF	Portable Document Format
RDBMS	Relational Database Management System
RDF	Resource Description Framework
SDI	Spatial Data Infrastructure
SQL	Structured Query Language
SSH	Secure Shell
UML	Unified Modelling Language
URL	Uniform Resource Locator
UWB	University of West Bohemia in Pilsen
WKT	Well Known Text
WMS	Web Map Service
WP	Work Package
XML	Extensible Markup Language

List of Figures

1 Introduction

1.1 *plan4business*

plan4business is a European project running from April 2012 until March 2014 and co-financed by the 7th Framework Programme of the European Commission. The full title is *plan4business – a Service Platform for Aggregation, Processing and Analysing of Urban and Regional Planning Data*.

plan4business develops a service platform for aggregation, processing and analyses of urban and regional planning data in Europe. Harmonised data will be integrated into seamless, homogenous, constantly growing and updated trans-border dataset. The platform will enable spatial analyses across European datasets. The platform should serve not only as a catalogue of planning data but also as their integrator enabling user to search, view, analyse and download spatial planning data on European and regional levels. The main project objectives are the automation of harmonisation processes and possibilities of complex analyses.

The *plan4business* consortium comprises six organisations securing the project execution:

- Fraunhofer IGD - Fraunhofer Institute for Computer Graphics Research, Germany
- UWB - University of West Bohemia in Pilsen, Czech Republic
- HSRS - Help Service - Remote Sensing, s. r. o., Czech Republic
- ISOCARP - International Society of City and Regional Planners, The Netherlands
- GEOSYS - GEOSYSTEMS Polska, Poland
- AVINET - Asplan Viak Internet as, Norway

1.2 Aim of the Report

The *plan4business* project should significantly contribute to decision making processes on various governmental levels and in cross-border activities. Next to issues of data availability and business modelling, the *plan4business* consortium is facing challenging problems of data integration, storage and analysis. These are the main issues that are being developed within WP5 and which are described in this document. This document also focuses on the implementation of API for accessing data including authentication, authorisation and digital rights management mechanisms.

This deliverable summarises the work performed and progress achieved in WP5 Storage, Integration & Analysis Engines of the *plan4business* project.

The main objective of WP5 is the development of the server part of the *plan4business* platform including the Storage, Integration and Analysis Engines, API (Application Programming Interface) and access control system. These are the main components of the *plan4business* server side solution that enable to store spatial planning data, to integrate data into a common data model and to perform data analysis. The main features of the engines are attuned according to the user requirements and marketing needs. The focus is on integration and interoperability of the proposed solution.

The *plan4business* platform is composed of three layers as depicted in Figure 1:

1. **Client layer (portal)** - user interface for performing data integration, management, analyses and visualisation.
2. **Integration and analysis layer** - engines for data harmonisation and data analyses. It includes API for better exploitation of the *plan4business* platform features in other applications.
3. **Storage layer** - optimised storage for data and metadata.

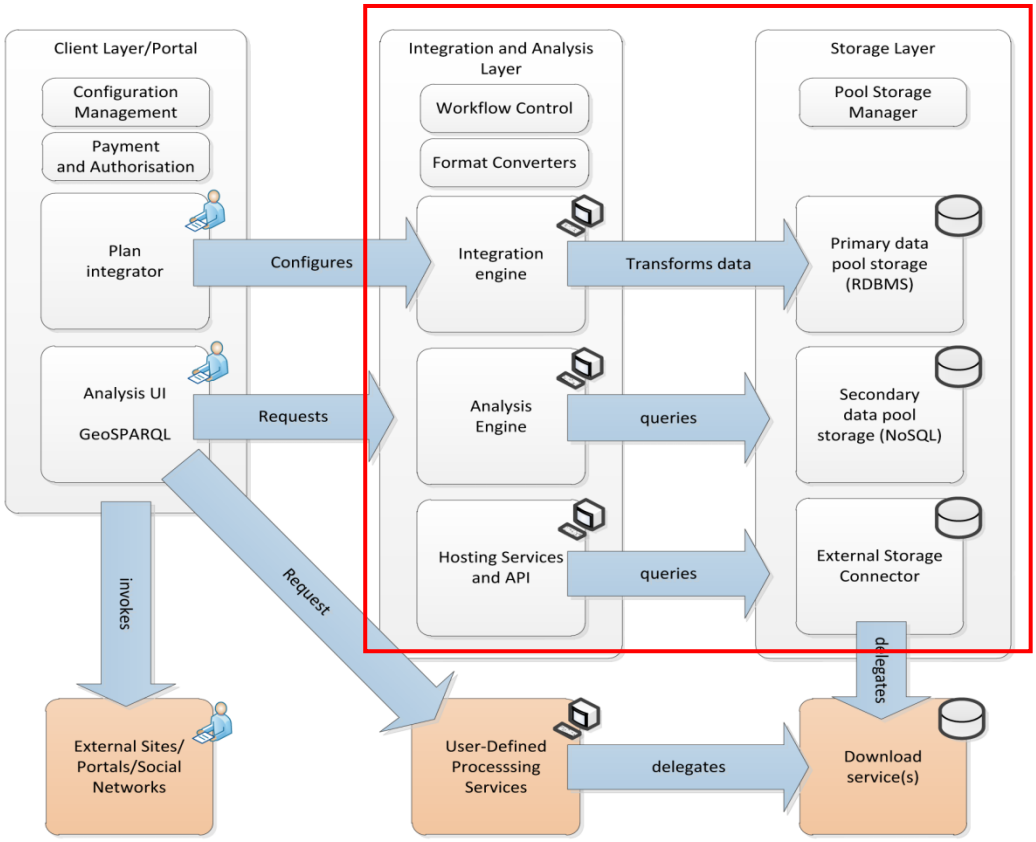


Figure 1 Three tier system of the plan4business platform (Fraunhofer 2012)

The integration and analysis layer and the storage layer are in the scope of this report. The client layer is developed in the frame of WP4 Plan Integration & Analysis Clients.

The *plan4business* platform aggregates the following types of spatial planning data:

- urban and regional planning data from different countries,
- land use data including GMES Urban Atlas data and Corine Land Cover,
- Open Street Map data as representative of road network and as a base map,
- Natura 2000 data as information about potential restrictions coming from environment protection,
- market information (number of properties, sale transactions, price levels),
- social and economic data (e.g. Eurostat data),

- individual property data (legal status, current use) and cadastral parcels.

The diversity and heterogeneity of input data is the main challenge for WP5. The solution for data integration is based on the experience of the project partners from previous projects including HUMBOLDT[1] and PLAN4ALL[2].

This is an interim report demonstrating the progress of the WP5 development towards the goals given by the Description of Work (2012) and reflecting the user requirements given by WP3 Requirements Management and Service Pricing. This report describes the server side engines for the *plan4business* platform including the Integration, Analysis and Storage Engines, API and access control system.

The *plan4business* project should significantly contribute to decision making processes on various governmental levels and in cross-border activities. Next to issues of data availability and business modelling, the *plan4business* consortium is facing challenging problems of data integration, storage and analysis. These are the main issues that are being developed within WP5 and which are described in this document. This document also focuses on the implementation of API for accessing data including authentication, authorisation and digital rights management mechanisms.

1.3 Structure of the Report

The document is structured in 10 chapters. The content of the chapters is as follows:

Chapter 1 contains a brief summary of the project, the main objectives of WP5 and the structure of the document.

Chapter 2 contains definition of terms that are essential for understanding of the content of this document.

Chapter 3 contains the management tools for the organisation of the development, source code control and issues tracking system.

Chapter 4 describes the agile methodology for the design and software development.

Chapter 5 describes the general architecture of all the components and how these components are interlinked.

Chapter 6 describes the status and progress of the Integration Engine.

Chapter 7 describes the status and progress of the Analysis Engine.

Chapter 8 describes the status and progress of the Storage Engine.

Chapter 9 describes the solution for API (Application Programming Interface) and the access control system.

Chapter 10 summarises the next steps in the development and concludes the document.

All the literature used for this deliverable is duly referenced. The list of citations is inserted at the end of the document.

[1] http://www.esdi-humboldt.eu

[2] http://www.plan4all.eu

2 Terminology

Application Programming Interface (API) – “An application programming interface (API) is a protocol intended to be used as an interface by software components to communicate with each other. An API is a library that may include specification for routines, data structures, object classes, and variables.” (Wikipedia contributors 2013a)

Atomicity, Consistency, Isolation and Durability (ACID) - a common behavioural model for databases and distributed systems that support concurrent transactions.

Geography Markup Language (GML) - “OGC`s XML-based language for describing and encoding geospatial information. An application of XML, a specification developed by members of the Open GIS Consortium. http://www.opengis.org/techno/specs/00-029/GML.html ". GML is an XML encoding for spatial data. In a sense, it is a schema-writing language for spatial information.” (OGC 2012)

Geoportal - "A Web site that provides a view into a universe of spatial content and activity through a variety of links to other sites, communication and collaboration tools, and special features geared toward the community served by the portal." (OGC 2012)

HUMBOLDT Alignment Editor (HALE) – “a tool for defining and evaluating conceptual schema mappings. The goal of HALE is to allow domain experts to ensure logically and semantically consistent mappings and consequently transformed geodata. Furthermore, a major focus is put on documentation of the schema transformation process and its impacts, e.g. in the form of lineage information attached to the resultant transformed data.” (Data Harmonisation Panel 2013)

Infrastructure for Spatial Information in the European Community (INSPIRE) – a European directive ensuring that the spatial data infrastructures of the Member States are compatible and usable in a Community and trans-boundary context.

NetworkLink - A network link contains a <Link> element with an <href> (a hypertext reference) that loads a file. The <href> can be a local file specification or an absolute URL. Despite the name, a <NetworkLink> does not necessarily load files from the network.

The <href> in a link specifies the location of any of the following:

- An image file used by icons in icon styles, ground overlays, and screen overlays
- A model file used in the <Model> element
- A KML or KMZ file loaded by a Network Link

The specified file can be either a local file or a file on a remote server. In their simplest form, network links are a useful way to split one large KML file into smaller, more manageable files on the same computer.

NetworkLinks gives the power to serve content from a remote location and are commonly used to distribute data to large numbers of users. In this way, if the data needs to be amended, it has to be changed only at the source location, and all users receive the updated data automatically. (Google 2013)

NoSQL database A “not only SQL” database that can interact with other ways than a SQL dialect. Thus NoSQL databases aren’t strictly required to work in an ACID-compliant way. Some NoSQL databases don’t even support SQL at all. Both of which can significantly improve performance for certain kinds of workloads.

Shapefile – vector data format for spatial data developed by ESRI.

Structured Query Language (SQL) – the most prominent language to formulate queries against databases.

Unified Modelling Language (UML) – a standardized general-purpose modelling language in the field of object-oriented software engineering. The Unified Modelling Language includes a set of graphic notation techniques to create visual models of object-oriented software-intensive systems. (Wikipedia contributors 2013b)

3 WP Management

WP5 is coordinated by UWB. WP5 is divided into four tasks with the following responsibilities:

- Task 5.1 Integration Engine Development – Fraunhofer IGD,
- Task 5.2 Analysis Engine Development – UWB,
- Task 5.3 Storage Engine Development - Fraunhofer IGD,
- Task 5.4 Pool Data API Development & Access Control System – HSRS.

In order to secure a smooth design and development of all the platform components, a Redmine[3] management system was set-up by the Project Office. Redmine is a flexible and open source project management web application.

Redmine serves for the following purposes:

- WIKI based documentation,
- issue tracking – bugs, new features, support issues and system requirements can be managed through issues assigned to a particular person and with specified deadline, priority, status, etc.

For the management of the source code, source documentation and configuration files, several Git repositories are provided. The repositories can be accessed through a Gerrit[4] installation and they are integrated in the Redmine system. An example of the repository for the Analysis Engine is shown in Figure 2. Gerrit is a web based code review system, facilitating online code reviews for projects using the Git version control system.

Gerrit helps avoiding errors getting into the code base, as code is reviewed by developers and could also be verified automatically by a continuous integration system. Basic access to the Gerrit system is restricted to project members, while access to individual underlying git repositories can be further constrained to subsets of project members. Access to Gerrit is available over SSH2 with public key authorization. Accounts for Gerrit and the repositories are managed by the project office and are given on a per-person basis. Any account can principally be either a committer or a reader account, i.e. not all accounts need to be allowed to commit.

[3] http://www.redmine.org/

[4] http://code.google.com/p/gerrit/

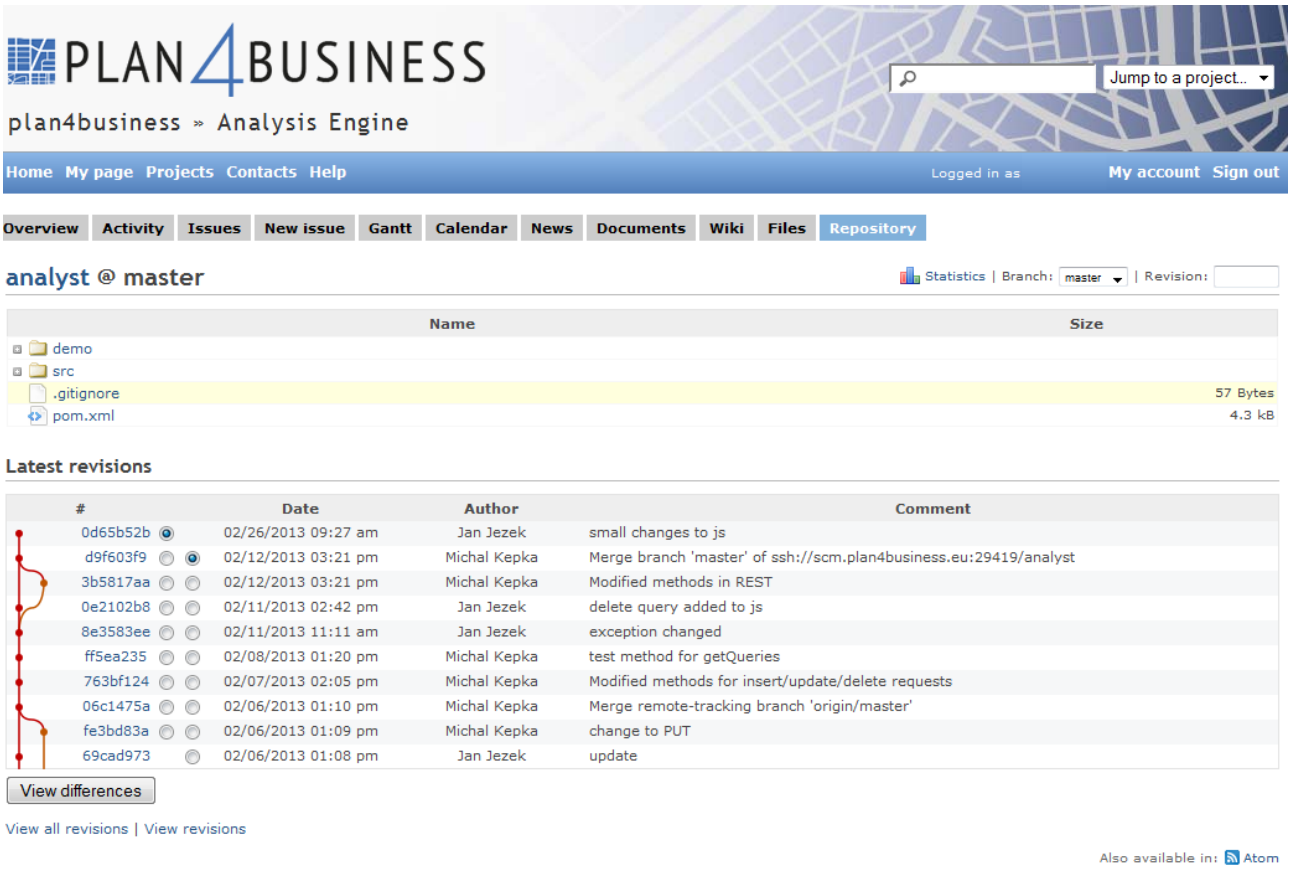


Figure 2 Source code repository for the Analysis Engine

4 Overall Methodology

4.1 Agile Methodology

The main research and software development work in the *plan4business* project is conducted in WP5. WP5 should result in the design and development of the core server side components of the *plan4business* service platform including the Integration, Analysis and Storage Engines, API (Application Programming Interface) and access control system.

Experiences from completed collaborative research projects show that it can be ineffective to create an abstract system design that accommodates all the different, and sometimes conflicting, user requirements. To ensure no such analysis paralysis happens and to ensure that any resulting software design is not unnecessarily complex, the agile approach to the system design is used. Furthermore, the agile approach is also taken to software development, and it is a basic requirement for WP3 (Requirements Management and Service Pricing) that results are delivered early and often.

In the design and development of each platform component, we aimed for a close loop of work between the components design and development team (WP5) and the requirements collection team (WP3). The design and development in WP5 started with a “code camp” workshop held in Pilsen in July 2012. The code camp was highly effective in communicating common coding policies and in actually solving technical issues. The work started with the design activities and infrastructure set-up as well as the creation of the initial data model for data integration and proposal of several use-cases.

The platform components developed based on the user requirements collected within the first six months of the project duration serve as demonstrators to refine the components design and to collect additional requirements. For this purpose, a series of workshops aimed to different groups of stakeholders are being organised and a feedback on the development is tracked using a questionnaire for workshops’ participants.

A complete evaluation including formal testing is conducted within WP6 System Integration and Operation. The testing is scheduled for the project months 12 and 24. The results of the complete evaluation are then fed back to the design and development team in WP5, who work in the second implementation phase to deliver the final application products.

4.2 Service Levels

Based on the user requirements coming from WP3, business model developed in WP2 and the agile methodology used for the system design and implementation (research and development in WP4, WP5 and WP6), four Service Levels related Milestones 3 – 6 were defined. These Service Levels represent high level measures for a successful implementation of the user needs and the business model.

A specific focus of these Service Levels is on a staged rollout of services to be offered by the *plan4business* platform. By using this staged approach, the platform starts to attract customers with concrete and useable services from the early stage of the development. These early results are valuable in providing feedback and in testing the infrastructure.

The four Service Levels are:

Service Level 1 (Milestone 3, month 9): This level includes examples of various components of the future platform which are not necessarily integrated but they show the basic functions that can be further elaborated and extended. This level includes

- a data storage for disharmonised spatial and non-spatial data,
- a common data model for harmonised data based on the INSPIRE Directive,
- mechanisms for data integration into the common data model,
- features (platform prototype) for data display and simple navigation,
- utilisation of pan-European datasets related to spatial planning from scattered resources.

The developed components are used for showcases during workshops, presentations and other meetings in order to provide potential customers an idea of the future platform and its functions and get feedback from end users.

Service Level 2 (Milestone 4, month 12): The main goal for this level is to make the platform prototype publicly available and extend it by the following features:

- analysis of harmonised spatial data based on user requirements (this should include not only predefined queries but also a possibility for user defined queries),
- user customised data mining queries,
- retrieval of the data mining and analysis results for display and download,
- prototype management tools for data upload, download and publication using OGC Web Services,
- catalogue of spatial planning data,
- creation of user defined map compositions.

Service Level 3 (Milestone 5, month 15): This service level includes improvement of the features from previous service levels and in addition the following features will be utilised:

- mapping functions for maps' customisation based on identified use-cases,
- tools for utilising feedback from users of spatial planning data,
- integration of the harmonisation tools into the platform,
- integrated metadata for analyses, map compositions and integration schemas,
- extended catalogue of planning data using RDF principles,
- support of standard analysis methods and geometric functions such as intersection, buffers and unions,
- extended data management tools enabling maintenance of different versions of datasets.

Service Level 4 (Milestone 6, month 18): This service level includes improvement of the features from previous service levels and in addition the following features will be utilised:

- support of more complex queries by using the primary data storage as well as the secondary data storage,
- advanced portrayal of the analysis result in a form of a table, chart or a report.

- support of most of the data formats defined by the users,
- payment module,
- generation of a report from a selected area including information such as data availability, data quality, data source and non-spatial data that are integrated with spatial data.

5 General Architecture

In order to secure the platform components can be integrated and work as a single system, the overall architecture was designed (Figure 3).

The central part of the platform is a metadata catalogue. In this case, the Micka catalogue developed by HSRS will be used. The catalogue enables to store not only metadata about existing datasets, but also about analyses, map compositions and integration services that can be performed.

The Integration Engine accessing and harmonising data in the Storage Engine is supported by the HUMBOLD Alignment Editor (HALE). HALE should be one of the components of the *plan4business* portal (the left part of Figure 3) securing user interaction with data that are being integrated.

The Analysis Engine processes the requests given by users through the *plan4business* portal. The query is processed and the Analysis Engine accesses the data storage and retrieves query results that are then provided to users in standardised form.

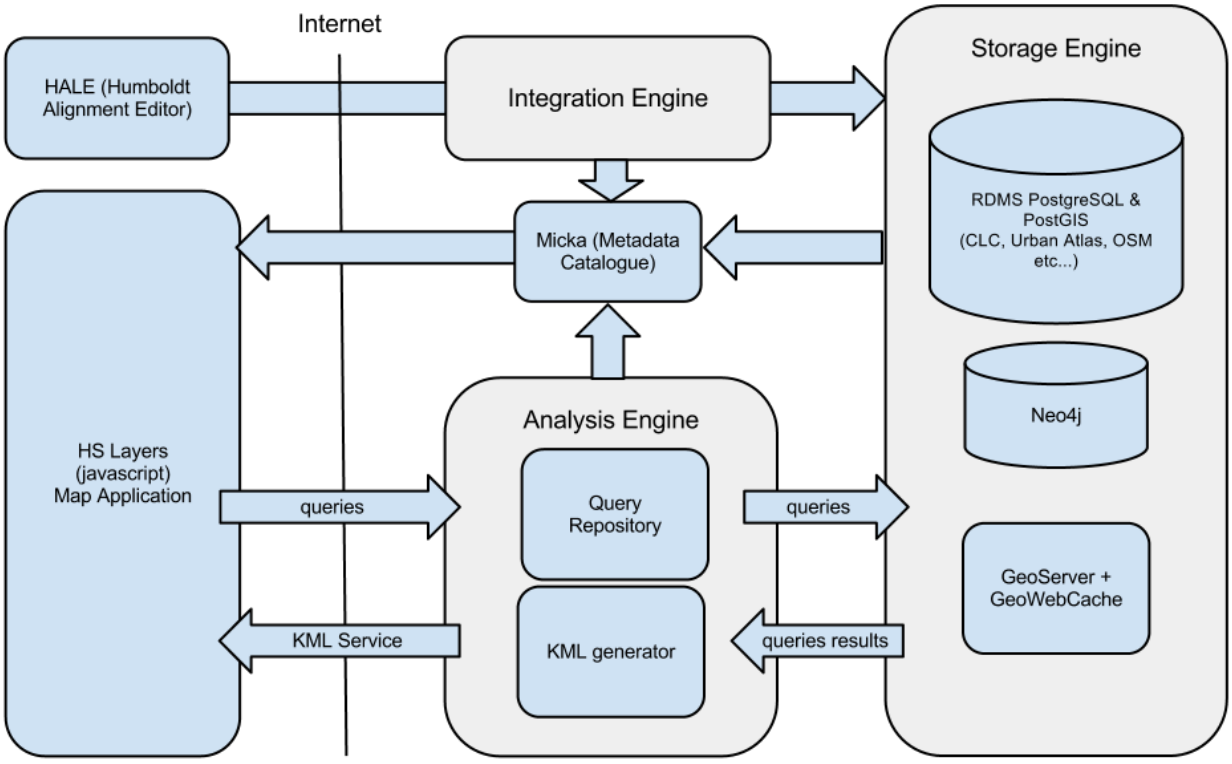


Figure 3 General architecture

6 Integration Engine

6.1 Objectives

The Integration Engine is one of the core components of the *plan4business* system. It is intended to perform all data transformations necessary to seamlessly integrate heterogeneously structured, externally provided spatial vector data into the *plan4business* pool of homogeneously structured vector planning data.

Based on the input data supplied through the Plan Integrator user interface (WP4) and a user provided schema alignment, the Integration Engine should be deployed on the server side of the *plan4business* system and transforms input data to the schema used by the Storage Engine (WP5, see section 7). In this process various conversions and transformations steps such as mapping between different data formats, geometric representations and conceptual schemas will have to be performed by the integration engine. This functionality is to be implemented on top of the functionality of the Humboldt Alignment Editor[5] (HALE), a desktop software that was initiated within the EU funded HUMBOLDT project (FP6) and is continuously developed as open source software under the lead of the Data Harmonisation Panel[6]. In the scope of *plan4business*, however, the actual data transformation is to be performed on the server. It will be possible to perform several data transformation processes that were initiated by different users at the same time. Furthermore the transformed data is not to be written to an output file within the file system but directly to a database[7] so that it can potentially be concurrently retrieved, analysed or extended in a resource-efficient and consistent way. Therefore a database schema has to be developed in close coordination with the work on the Storage Engine (Task 5.3).

6.2 Work Done and Progress Achieved

6.2.1 Overview

In order to integrate various external planning data models into a data model common to the *plan4business* platform, evidently as a first step the design of the Integration Engine has to take into account the structure and relation of different data models in the *plan4business* system. As a first step, this was addressed with the conceptual work done on data models and schema design, which is described in the following subsection.

6.2.2 Data Model Design

Figure 4 gives an overview of the data models for spatial planning data within the *plan4business* system that will be addressed in the following subsections.

[5] http://dhpanel.eu/humboldt-framework/hale.html

[6] http://dhpanel.eu

[7] As one of the central components of the Storage Engine, see section 8.

The *INSPIRE UML model* is chosen as a functional basis for all *plan4business* data models. The INSPIRE UML model is developed in the scope of the INSPIRE initiative that is targeted to develop legal and technical frameworks for a common European spatial data infrastructure (SDI) and a common European pool of geospatial data that can be easily integrated with various geospatial information services. As such this encompasses, from a technical viewpoint, lots of goals that are also relevant for the *plan4business* platform. As the results of the INSPIRE initiative will prospectively have legal character for public authorities, the *plan4business* data model is designed with compatibility to that future standard in mind.

The INSPIRE data specifications encompass 34 themes[8] for each of which a detailed data model is described by the aid of extensive UML models. Amongst other applications, these UML models are used to derive GML based application schemes that can be used to exchange data between different systems or software components in a standardized way. As basis for the work in *plan4business*, version 3.0 rc2 of the INSPIRE data specifications were used.

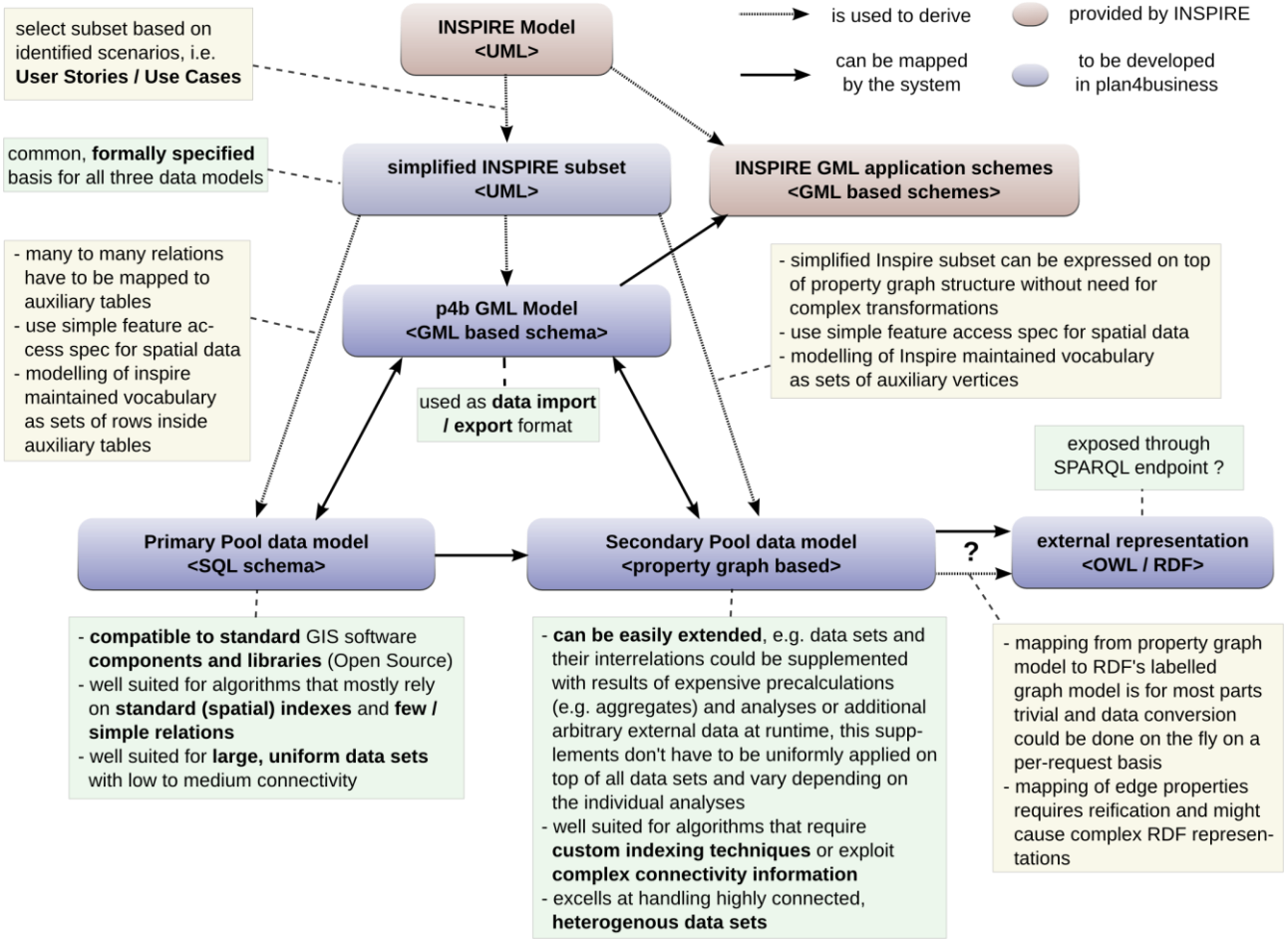


Figure 4 Relations between plan4business and INSPIRE data models for vector and metadata

[8] An overview of the themes is given under: http://inspire.jrc.ec.europa.eu/index.cfm/pageid/2/list/7

6.2.2.1 Simplified INSPIRE Subset

The scope of *plan4business* is focused on spatial planning data and its lifecycle is far more fast moving than the INSPIRE initiative as a whole. It is neither feasible nor functionally required to address and implement each of the 34 themes of the INSPIRE data specifications. In order to reduce the complexity of the resulting data model for *plan4business*, themes that are not relevant for the identified use cases of the *plan4business* system or where no suitable source data sets can be obtained will not be handled. In addition, individual details of those themes that were selected for *plan4business* will be omitted for at least the initial implementation. However during the project lifecycle the initially selected subset of relevant INPIRE themes and the set of adapted features of the themes can be extended as needed.

As shown in Figure 4, the Simplified INSPIRE Subset is used as a logical basis for all other, technology specific planning data models of *plan4business*. A change to the selected INSPIRE subset also implies the adoption of the derived data models.

As a starting point we selected the INSPIRE Land Use Theme[9] which covers the most fundamental aspect of the data relevant for *plan4business* (see Figure 5). This theme can be used to capture spatial data sets that describe existing or planned land use or spatial plans. Each dataset aggregates a (non-empty) set of *LandUseObjects* that are either *ExistingLandUseObjects* (for a *LandUseDataSet*) or *ZoningElements* (for a *SpatialPlan*). All LandUseObjects provide information about their spatial extent, the time period for which they are valid and are classified according to the *Hierarchical INSPIRE Land Use Classification System (HILUCS).*

[9] see Annex III of the INSPIRE Directive

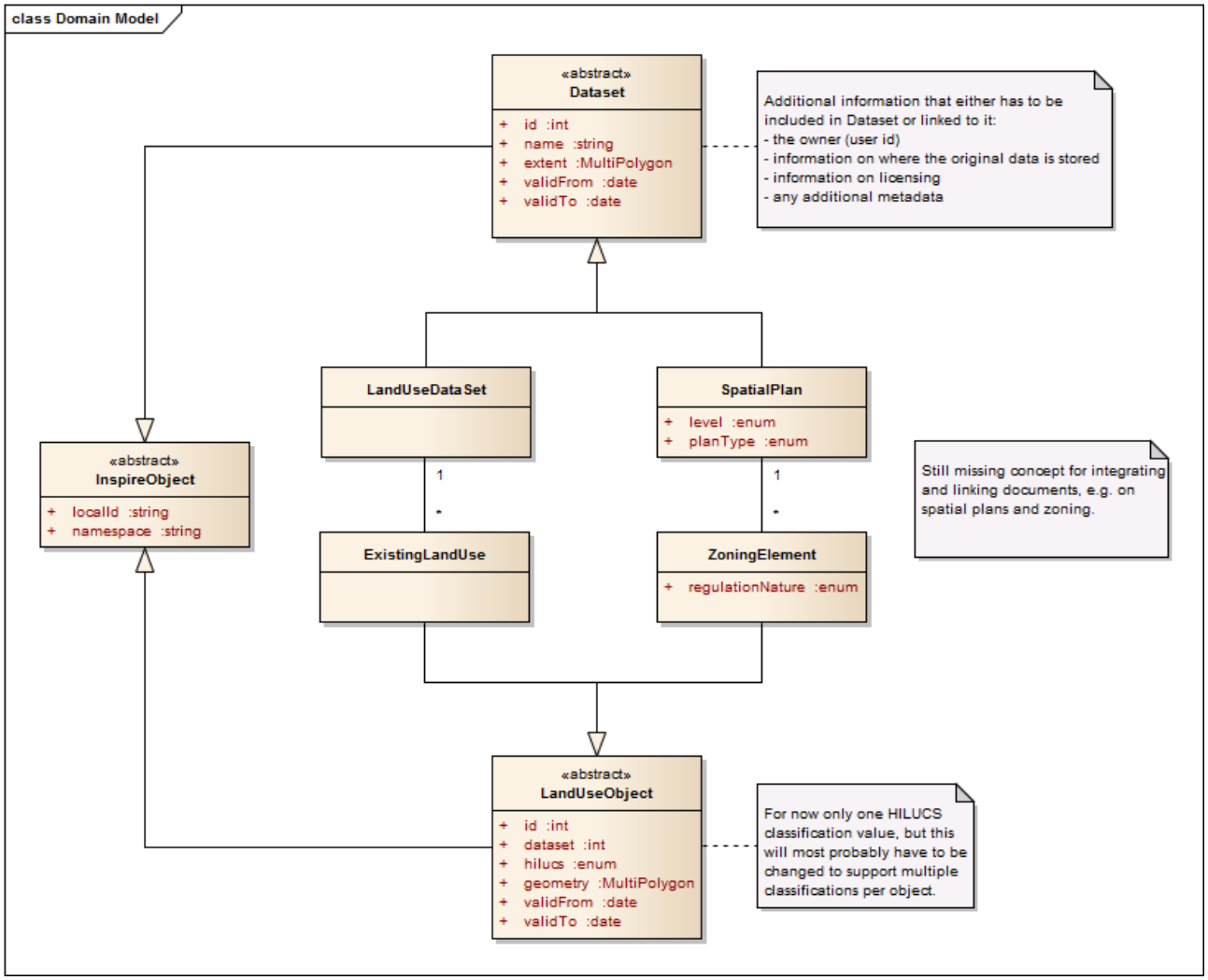


Figure 5 UML Overview of the initial simplified INSPIRE Subset that served as basis for the first plan4business Intermediate Model Draft

6.2.2.2 The *plan4business* Intermediate Model

In a similar way as the GML based INSPIRE application schemes are derived from the INSPIRE UML Models, the *plan4business* intermediate model is derived from the selected simplified INSPIRE UML Subset described in the previous section. The Intermediate Model is a XML based schema. Due to ease of implementation and as close as possible matching the Simplified INSPIRE UML Subset, only the notion of spatial extents is strictly GML compliant, whereas *Datasets* and *LandUseObjects* are not modeled as GML feature types. Figure 6, Figure 7, Figure 8 and Figure 9 showcase the structure of the XML schema of the intermediate model for ExistingLandUse Datasets and Spatial Plans and their dependent subtypes of LandUseObjects.

The *plan4business* intermediate model is used as target schema for the integration of Shapefiles, XML, GML or CSV[10] based input data. The intermediate model can be used as *plan4business* native data format for import and export from, to and among components of the *plan4business* system. For increased INSPIRE compatibility, it is intended to include a transformation from the *plan4business* intermediate Model to (the structurally more complex) relevant INSPIRE GML application schemes.

Once a data set is structured according to the *plan4business* intermediate model it can be transformed to the database specific models of the primary and secondary data pool. During the transformation the data is not actually encoded as XML in the intermediate model, but transformed directly in-memory and stored in the data pool.

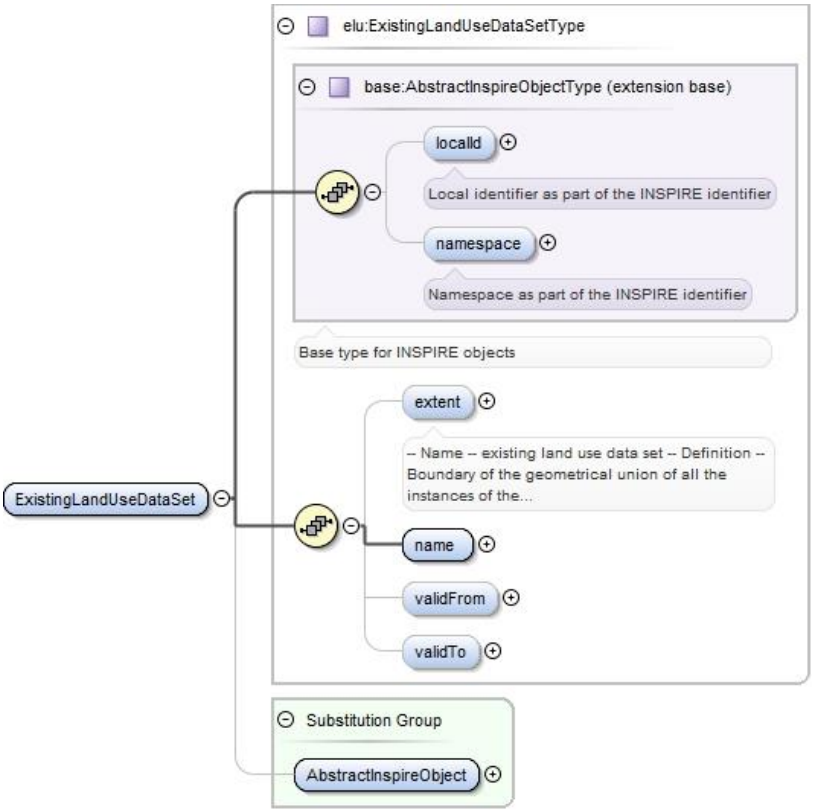


Figure 6 Representation of Existing Land Use Data Set in the plan4business intermediate model

[10] Comma separated values (.csv), e.g. originating from spreadsheets or database export with support for OGC WKT expressions

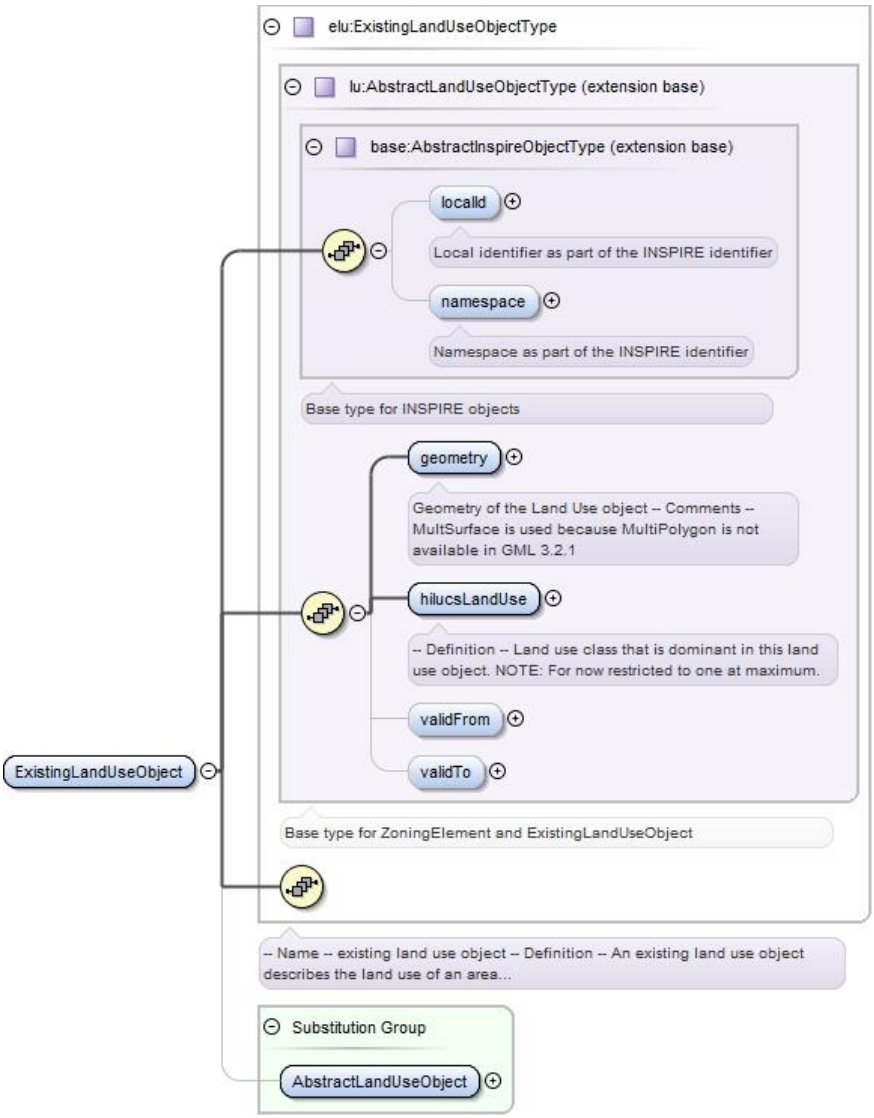


Figure 7 Representation of ExistingLandUseObjects (i.e. features) in the plan4business intermediate model

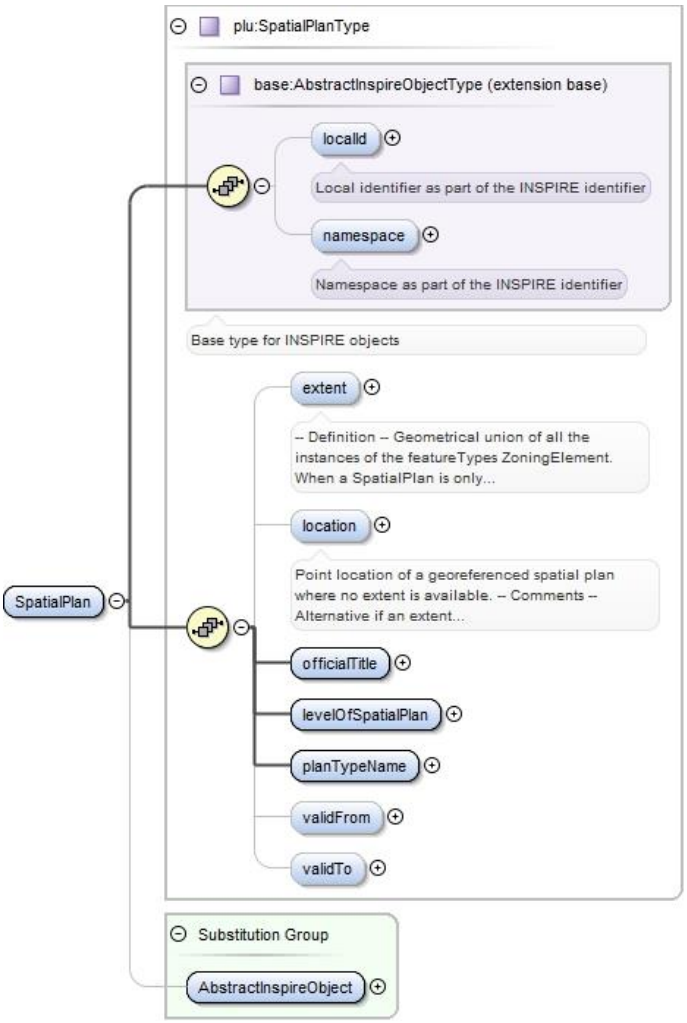


Figure 8 Representation of a Planned Land Use Data Set in the plan4business intermediate model

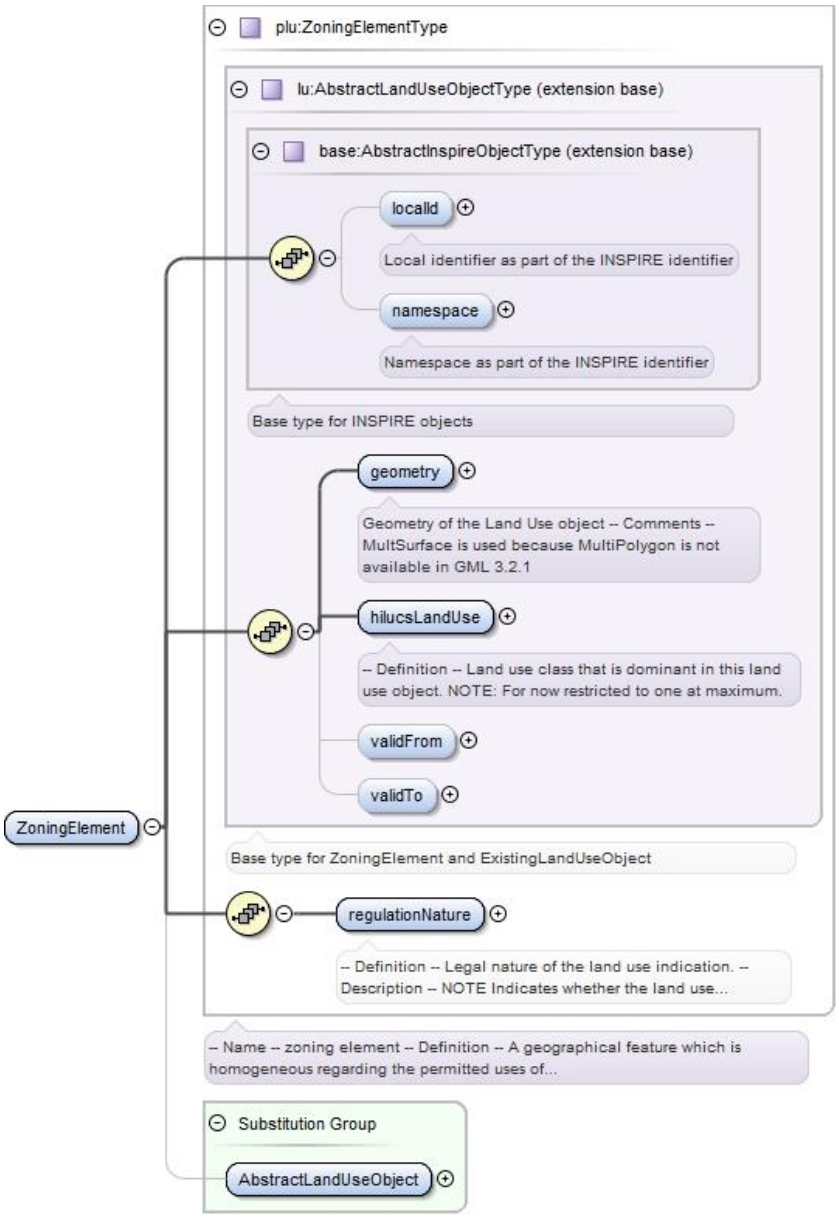


Figure 9 Representation of a ZoningElement (i.e. a feature) of a Planned Land Use Data Set

6.2.2.3 The Primary Pool Data Model

As the Intermediate Model, the Primary Data Pool Model is derived from the simplified INPIRE UML Subset and geared towards unified, concurrently accessible storage, that can be used not only by newly developed components but also by existing third party Open Source components with no or minimal adoption required. The Primary Pool Data Model is a SQL based schema for the PostGIS extension to the PostgreSQL Open Source DBMS. The schema consists of tables that each reflect either the entities (e.g. *ExistingLandUse DataSets*, *SpatialPlans*, *ZoningElements*), code lists (e.g. HILUCS values), additional meta-data (more details in next paragraph) or relations between different table entries. It conforms to the OGC Simple Feature

Access Specification[11,12] to express spatial data and provides support for spatial indexes that greatly increase spatial query execution performance. For more details on third party component integration and DBMS specific aspects or the Primary Data Pool, see Chapter 8. Once dataset have been entirely transformed into the Primary Pool Data Model (and are thus stored in the Primary Data pools Database), they are not subject to frequent modifications.

Besides the core planning data, in comparison to the simplified INSPIRE subset, the Primary Pool Data Model (Figure 10) is extended by metadata tables that provide additional information about the integrated and source data sets. Each integrated vector data set references a metadata entry that also captures the corresponding original source data set. In turn, each metadata entry provides information about the licensing type, the contributing user, a simplified spatial extent, an optional URL entry for WMS raster data and a list of files. Each of those list entries reflects the path of a file that is stored directly in the file system. The *plan4business* metadata catalogue (see Section 6.2.2.4) which is INSPIRE compliant should be synchronised with the system metadata.

The Primary Pool Data Model does not impose strict limitations on what file types can be attached to a meta-data entry. Thus it is possible to store the originating, untransformed source data sets, more detailed licensing and legal information as well as documents of arbitrary file formats[13]. Thus, it is also possible to store data that cannot be integrated to the planning data model by the integration engine. In particular that encompasses planning data that is not available in a sufficiently structured data format, such as PDF documents or plain raster data. For these purposes, the metadata catalogue will be used. For more information, see Section 6.2.2.4.

A second addition for to the Primary Pool Data Model on top of those entities derived from the simplified INSPIRE UML Subset will be the inclusion of auxiliary data sets. The number of auxiliary data sets will be very limited and each of them will have a separate representation in the schema. Potential candidates for inclusion as auxiliary data sets are machine readable spatial data sets that are already publicly available for large parts of Europe and that capture aspects that are thematically orthogonal to the integrated planning data. Currently this includes Open Street Map (OSM), Urban Atlas, CORINE Land Cover (CLC), Natura 2000 and Nationally designated areas (CDDA).

[11] OGC Simple Feature Access, Part 1: Common Architecture http://www.opengeospatial.org/standards/sfa

[12] OGC Simple Feature Access, Part 2: SQL http://www.opengeospatial.org/standards/sfs

[13] This is in fact a crucial feature as in many cases the legally binding part of spatial plans is often provided as not machine-readable text while the machine readable spatial vector data is just a supplementary and not strictly legally binding representation of the planning document.

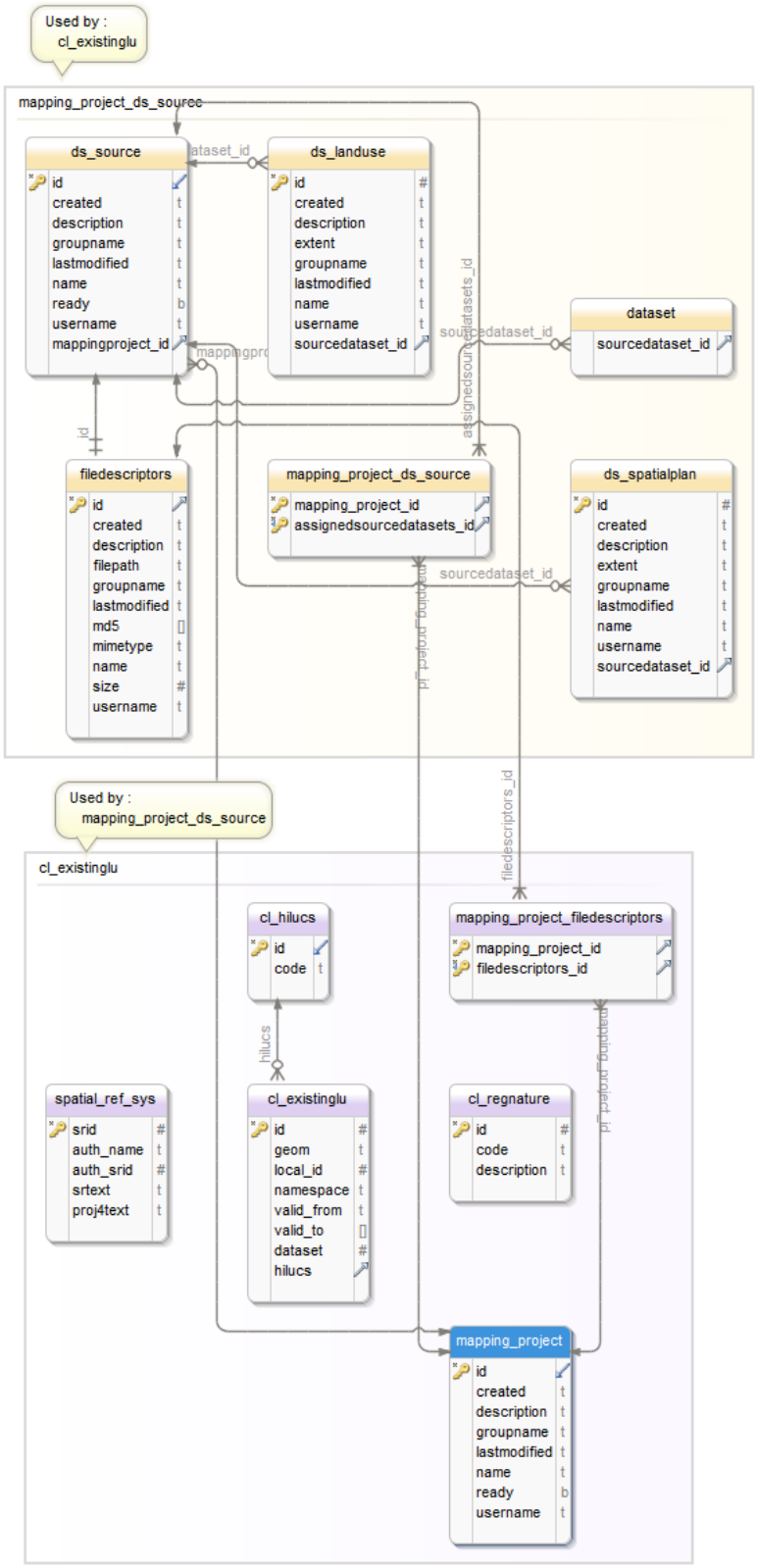


Figure 10 Primary data pool relational data model

6.2.2.4 Metadata Catalogue

The *plan4business* metadata catalogue will be used for documentation, search, browse and invoke the following resources in the *plan4business* system:

- Non-harmonised datasets (e.g. rasterised spatial plans),
- Harmonised datasets (from data pool),
- Third party datasets and services used in the system,
- Web services built on these datasets,
- Analysis tools built on data,
- Map compositions consisting from web services (*plan4business* and external WMS) and analysis tools provided for particular area.

Metadata catalogue should be INSPIRE compliant (ISO 19115/19119/19139, OGC CSW 2.0.2 ISO AP 1.0 standards supported), additional queryables and sorting should be supported for the project purposes. The HSRS metadata catalogue MICKA[14] will be used for these purposes.

6.2.2.5 The Secondary Pool Data Model

The Secondary Pool Data Model will be tailored to a property graph based representation that can be stored inside the Neo4j Graph DBMS. Similar to the Primary Data Pool Model, spatial data is expressed according to the OGC Simple Feature Access Specification and can also be indexed spatially, both of which is supported by the Neo4j-spatial extension. In general using a property graph as basis for the data model significantly reduces the complexity of expressing and querying highly connected data in comparison to the relational model that is used as basis for the Primary Pool Data Model. While the design of the Secondary Pool Data Model is also derived from the simplified INSPIRE UML Subset, it takes into account that advanced analysis tasks will likely require to extend or modify the initial data model[15] especially for performance reasons, either by storing expensive intermediate results, pre-calculations[16] (e.g. aggregates) or entire analysis results. This also includes the possibility to add completely customized index structures that are not supported out of the box by commonly used DBMS[17]. However, it is expected that these custom extensions to the data model will not necessarily be made to all available integrated data sets and / or that each type of analysis can introduce its own modifications to the initial logical data model, so further specification of the data model will be dependent on the steps taken to implement the analysis features of the *plan4business* platform, that are themselves derived from additional Use Cases.

---

[14] http://www.slideshare.net/SDIEDU/micka-manual

[15] In order to reduce the data volume, it might also be useful to reduce the data model by omitting details that are not relevant for a specific analysis

[16] This is applicable in the case that pre-calculations or intermediate results will be reused or are too large to fit into the systems main memory

[17] For more detailed information on this, please refer to section 8.2.2

6.2.2.6 External RDF / OWL Representation

As a way to export the integrated planning data and analysis results out of the Secondary Data Pool could be to rely on a RDF based representation. Since RDF is itself a labelled graph-based data model, its logical structure is very similar to the property-graph based data model of the secondary data pool model.

While being a W3C standard, RDF is seeing more and more usage recently, especially in the context of linked open data. Support for expressing spatial data by the aid of WKT literals and spatial reasoning has recently been added to several RDF storage systems[18,19].

For Neo4j, which forms the basis of the Secondary Data Pool, there are extensions available that can wrap a Neo4j graph database so that it can effectively be used as a backing store for RDF Triple Stores that are compatible to the Sesame Sail API (e.g. such as Strabon which is itself directly based on Sesame). However the actual implementation effort, as well as the maturity of the components involved can hardly be estimated and further evaluation is required.

The data model that is used for the external RDF based representation could be expressed through a Ontology (using the OWL language). To reduce implementation effort, the data model for the external RDF representation should be logically closely aligned with data secondary data pool storage and the technical that are introduced through the Neo4j Sail API implementation.

6.2.3 Work on Implementation

The Open Source tool HALE forms the basis for the Integration Engine. Core transformation functionalities in HALE that are needed to accommodate the *plan4business* use case on integrating land use data have been implemented. Furthermore, a new software component was developed that can utilize core functionality of HALE without the need for a graphical User Interface. Existing schema transformation components of HALE have been adapted so that they can be used on a server, multiple transformation operations there can safely run in parallel.

Based on the design of the simplified INSPIRE data model for Land Use, the intermediate model in GML and the database model for PostGIS have been implemented.

[18] E.g. the stRDF/stSPARQL support developed on top of Strabon: http://www.strabon.di.uoa.gr/

[19] E.g. the OGC GeoSPARQL standard and upcoming implementations: http://www.opengeospatial.org/standards/geosparql

7 Analyses Engine

7.1 Objectives

The Analysis Engine is one of the *plan4business* platform components ensuring management of spatial planning data stored in a relational database. It enables accessing and processing spatial planning data and their retrieval for visualisation in the map client. The Analysis Engine provides access to all analytical functions of the spatial database.

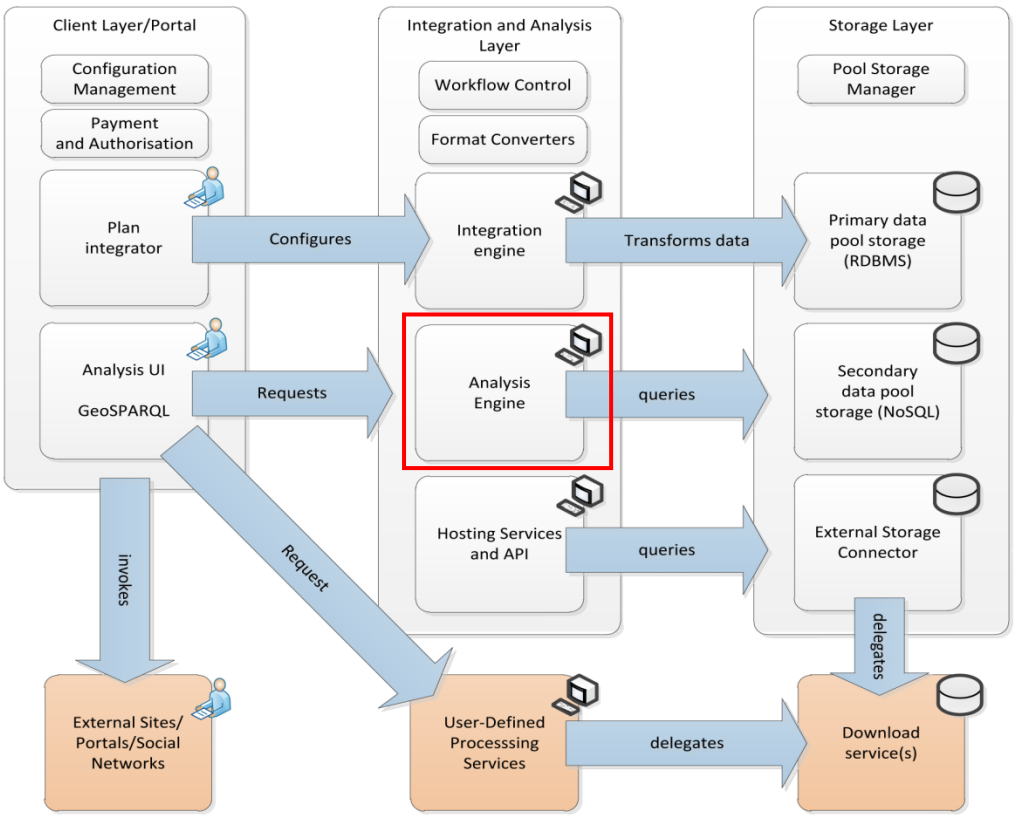


Figure 11 The components of the plan4business platform with the highlighted Analysis Engine (Fraunhofer 2012)

7.2 Work Done and Progress Achieved

7.2.1 Query Repository

The *plan4business* platform uses the open source PostgreSQL database with the PostGIS support programme for spatial data. While the analysed data are stored in various database schemas, the results of analyses are stored in a single schema. The results in that schema are organised through the stored_query table depicted in Figure 12. The table contains the following information:

- query_id – query identifier is used for data management,
- sql_query – original user defined SQL query for the analysis,
- result_table_name - name of the result table,
- time_stamp –continuously changing time stamp indicating either the time of the query initiation, editing or finalisation,
- processing_state – the status of the current process: in process, finalised, deleted,
- geometry_column – geometry type that can be used for visualisation of queries with particular geometry,
- user_id – identifier of the user who set the query and has the right to edit it,
- used_time – an indication whether time is used in the query as another dimension.

Figure 12 Metadata table containing the information about the analysis result (Ježek et al. 2013)

Each record contains a unique identifier for securing the connection between the query and the results. All database operations are managed by the database functions (RDBMS). RDBMS also ensures data consistency by utilising fully transactional behaviour.

7.2.2 Query Processing

The main part of the Analysis Engine runs on a server. This part ensures communication with users and forwarding requests to the database. Two main modules and one supportive module can be distinguished.

The first module ensures the reception of user queries and execution of the requests in the database. A query is received by a servlet through the HTTP protocol, the quality of the mandatory parameters is checked and method for further processing is selected based on the combination of the parameters. User query is stored together with other parameters into an object that can be then stored in Query Repository (the stored_query table). At the same time, the query is processed into an SQL command. This results in a

new table. The object is then stored into the stored_query table together with the status of the query execution. The status is changed to as soon as the execution of the query is successfully finished.

The second module provides publication of user queries' results. A servlet receives user requirements from the portal and then retrieves data from the database. The user requirement includes identification of the original query and bounding box of the map window where the results will be displayed. In the next version of this module, additional parameters for data visualisation, such as coordinate reference system, are envisaged. As soon as user requirements are received and parameters are checked, a utility for data retrieval from the database is called based on the combination of the parameters. Retrieved data are then converted from objects into the hash-map structure, which corresponds with the structure of the resulting KML file. Only the geometry is retrieved from the database directly in the KML format. Templates for data publication are applied to the resulting files before they are sent to the portal.

The supportive module contains methods for query management including checking the query status, accessing past queries and their deletion. All operations can be triggered through secured REST API.

The query processing is part of the Analysis Engine API which is described in the section 9.2.2.2 Analysis Engine API.

7.2.3 Visualisation

The results of the analysis are in the KML format that can be visualised in any map client supporting KML such as the *plan4business* platform or Google Earth. The prototype of the *plan4business* platform is available at http://www.urbanplan-business.eu/. Figure 13 shows the platform with predefined queries that can be performed. These queries serve mainly for testing purposes.

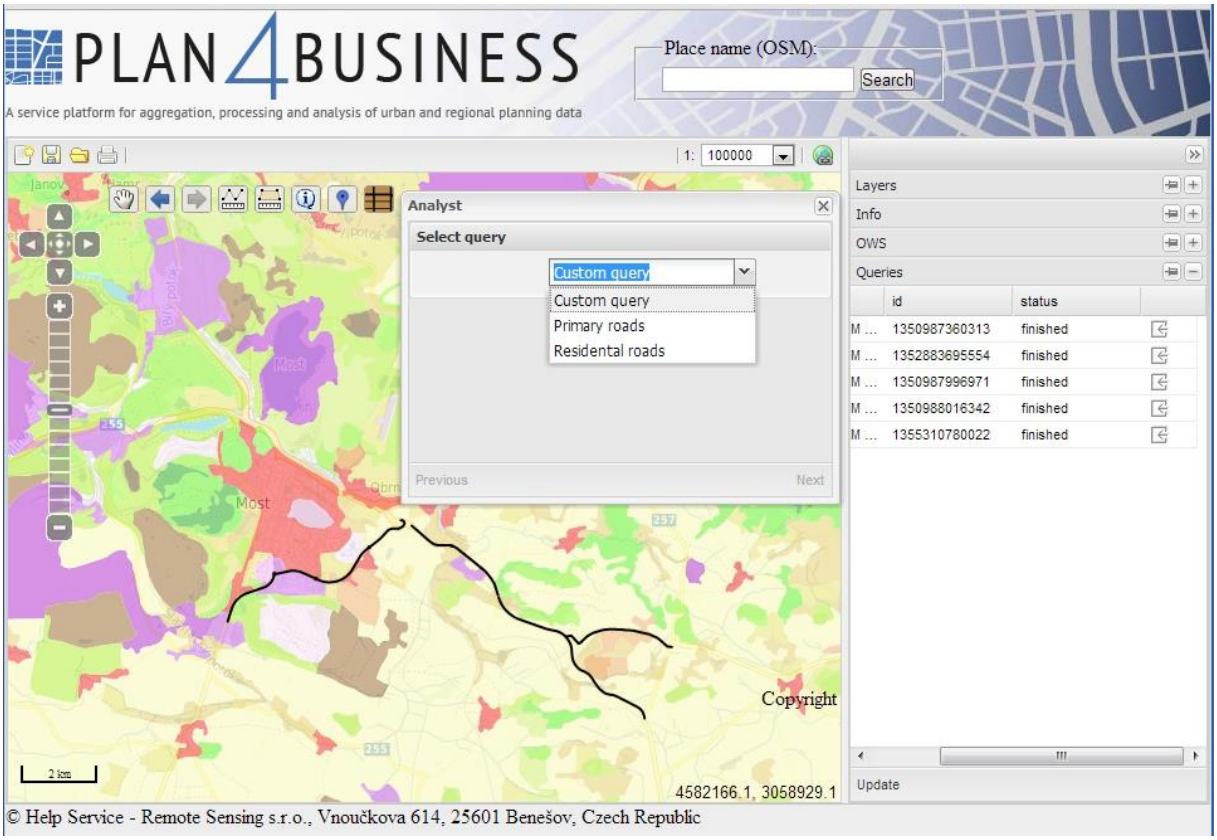


Figure 13 A prototype of the plan4business platform (Ježek et al. 2013)

For debugging and testing purposes a web application for performing predefined and user defined queries and displaying query results without any major configuration was implemented (Figure 14).

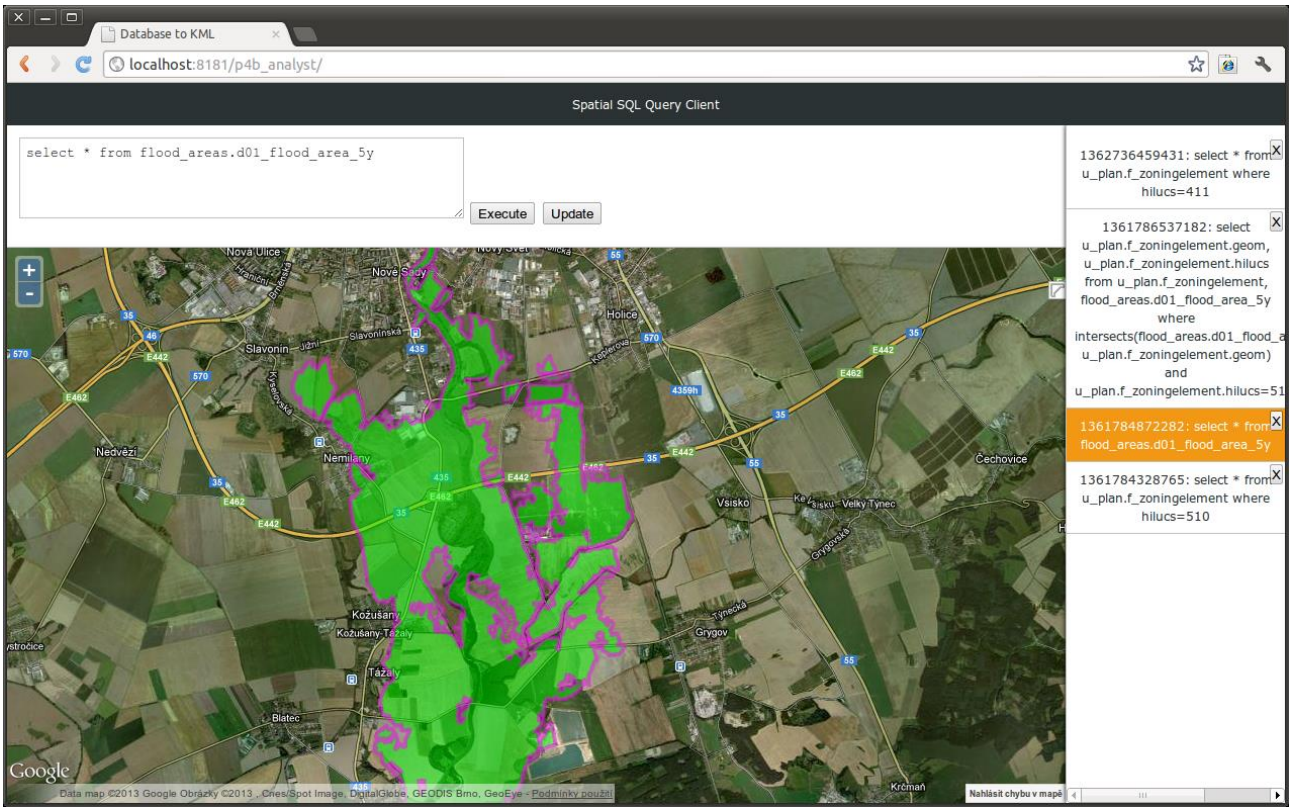


Figure 14 A web application for querying the plan4business database and displaying the results.

8 Storage Engine

8.1 Objectives

The Storage Engine of the *plan4business* system is planned to consist of three components. The first component is the primary data pool storage which should be a hybrid combination of a relational database and a document-oriented storage. The primary data pool storage should provide access to the original (disintegrated) spatial data sets and to the integrated spatial vector data sets.

The second component is the secondary data pool storage which is intended to augment the capabilities that are offered by the primary data pool storage. While the relational database management system used for the primary storage can be expected to scale well with a growing amount of data and to provide fast data retrieval (i.e. especially for read only and / or simple queries), the choice of the database management system for the secondary data pool should be aligned with the support for complex graph-based queries or the possibility for more convenient[20] or performant analysis functionality which can most likely be achieved by using a NoSQL based database that breaks with common design decisions of RDBMS. Given the widespread adoption of RDBMS in industry applications, there are lots of available GIS components[21] that can interact with RDBMS without the need for extensive customization. This can to some extent compensate the expected lack of readily usable third party components for NoSQL-DBMS that are otherwise of great interest for the *plan4business* Use Cases. In conclusion, the secondary data pool storage is intended to implement analysis functionality that would otherwise be too complex, error-prone or resource intensive if it were to be implemented on top of the primary data pool storage. However to determine which particular DBMS technology to use for the secondary data pool storage a more detailed survey should be conducted as part of the work on the storage engine.

Finally, the third component of the Storage Engine is the Storage Manager implementing management functionality to control the data that is available in the primary and secondary data pool storage. E.g. the Storage Manager could thus be used to remove data from the primary data pool or to copy data that is needed for a certain analysis from the primary data pool to the secondary data pool.

8.2 Work Done and Progress Achieved

8.2.1 Methodology

As starting point for the work on the Storage engine, we initially evaluated suitable DBMS that could be used as a basis for the primary and secondary data pool that form the foundation of the Storage Engine. In a first step, we performed an evaluation on suitable components for the primary data pool and started the development on the database schema as well as the actual implementation, which is explained in more detail in the following section. In a second step we performed an evaluation of promising DBMS to use as a basis for the Secondary Data Pool Storage and developed several vertical prototypes concerning technical

[20] that is of potentially less implementation or maintenance effort for equivalent analysis functionality

[21] This also includes the majority of open source implementations of GIS components and libraries, especially in the case of the PostgreSQL / PostGIS RDBMS

feasibility as well as ease of use and integration for the Neo4j Graph Database we chose. Work on both Data Pool Storages is closely linked to our work on data models in section 6.2.2 (see also Figure 4).

8.2.2 Primary Data Pool Design and Development

The central part of the Primary Data Pool is an RDBMS. There are several stable and mature RDBMS available that provide support for spatial data and operations.

Oracle Spatial is an extension to the proprietary Oracle Database. While it can be considered as a mature and widely used RDBMS, it is not free to use and only few components that can be used on top of the oracle Database are available without license fees or even as Open Source. Furthermore the Data Model of Oracle Spatial is not conformant to the OGC Simple Feature Access specification because it uses a completely different approach to express and work with spatial data, hence it's compatibility with common Open Source GIS software components is sparse and often not very mature. There is thorough documentation available for working with Oracle Spatial from within Java. The Oracle Database system can be sharded for vertical scalability.

Microsoft SQL Server Spatial Database is a proprietary extension to Microsoft's SQL Server RDBMS. It can be considered to have a mature codebase and is widely used in the Windows .NET ecosystem. Similar to Oracle Spatial, it is not free to use and support for Microsoft SQL Server Spatial Database in common Open Source GIS components is comparable to the situation with Oracle Spatial. In contrast to Oracle Spatial, the data model of the SQL Server Spatial Database is compatible to the OGC Simple Feature Access specification. However it must be run on top of a Windows based operating system, which adds the licensing costs of Windows itself. In recent versions, SQL Server can be sharded for vertical scalability. At the time of writing, there is no mature JDBC driver available so that the Spatial Extension for SQL Server could be conveniently used from Standard Java.

MySQL Spatial is part of the Open Source MySQL RDBMS with commercial support available. MySQL itself can be considered among the most widely used and efficient RDBMS, in particular for the development of Web Applications. The Spatial support is compatible to the OGC Simple Feature Access specification and a mature JDBC driver is available for Java development. However, full spatial support has just recently been added to MySQL. In versions prior to 5.5 spatial extensions cannot be used at all in conjunction with database transactions. This leads to limited support and testing in conjunction with other Open Source GIS components. MySQL can be sharded for vertical scalability.

MonetDB is an Open Source, column-store RDBMS with commercial support available. MonetDB features support for spatial data types and queries that are compatible to the OGC Simple Feature Access specification. There is a JDBC driver for Java Integration available. As a columnar-store database MonetDB can provide performance benefits for workloads that involve aggregation and yield drawbacks for operation that insert or retrieve entire rows of a table. Underneath the SQL interface, MonetDBs architecture and implementation focuses on exploiting the characteristics of modern operating system and CPU architecture instead of relying on complex optimizations to a B-Tree based index as most other RDBMS. In addition MonetDB offers support for processing multidimensional arrays through an extended SQL dialect (SciQL), which could also be used to process spatial raster data directly in the database. However MonetDBs use is not as widespread as the other mentioned RDBMS and thus support by third party components can be described as very sparse. Also the level of maturity of the implementation and available documentation must

be considered lower than average. Fraunhofer's experience from other projects[22] is that in conjunction with very large spatial data sets (beyond 100GB of RDF triples) and when used as a backend for the Sesame Triple Store, MonetDB can significantly outperform PostGIS[23]. In addition, MonetDB setups can be sharded which offers additional benefits in terms of scalability.

PostGIS is a spatial extension to the OpenSource PostgreSQL RDBMS with commercial support available. Along with MySQL it is one of the most widely used RDBMS. In particular for Open Source GIS software and components, PostGIS seems to be the most widely supported database. PostGIS provides OGC Simple Feature Access conformant spatial data structures and queries that can be considered as a very mature implementation. Since its 2.0 release, the PostGIS extension also supports direct operation on spatial raster data. To Fraunhofers general experience[24], PostgreSQL RDBMS performance for simple queries is close to MySQL and generally significantly faster than Oracle Database for small to medium data set size (below 20GB). As well as the other mentioned RDBMS, PostGIS can be sharded as well for vertical scalability.

Finally we chose PostGIS over MonetDB and MySQL as the backing RDBMS for the Primary Storage as we consider it the best trade-off between maturity, third party support, available documentation, licensing costs, performance and otherwise potentially useful features.

For more details on the data model for the primary data pool storage see section 6.2.2.3.

8.2.3 Survey and Prototype Development for Secondary Storage

The core of the Secondary Data Pool should be a DBMS that complements the characteristics of the RDBMS used for the primary data pool. Regarding the common characteristics of RDBMS, this could be achieved by using a NoSQL DBMS. Currently there are few NoSQL DBMS available that have support for spatial data structures including handling of different spatial reference systems:

MongoDB is an Open Source document oriented data store with commercial support available. Essentially, a document oriented data store is similar to a Key Value store. In addition its values (i.e. the documents) can consist of multiple primitive values themselves. There is no built in support for managing relations between several values. Also there is no built in management for concurrent write access (i.e. MongoDB is not ACID compliant)[25]. Both aspects greatly reduce the complexity of the database implementation and lead to very good performance for simple lookup queries when compared to conventional RDBMS systems. In general, document oriented data stores can be very easily distributed across multiple nodes in a network, which also gives them very good vertical and horizontal scaling capabilities. Instead of using SQL, queries are expressed using a custom BSON[26] protocol. These queries are then executed based on MongoDB provides

[22] In the context of FP7 project TELEIOS both PostGIS and MonetDB were used as a backend for the spatial reasoning enabled RDF triple store Strabon, which is based on the Open Source Sesame Triple Store.

[23] As reported, PostGIS however significantly outperforms MonetDB by several orders of magnitude for smaller RDF data sets in this context.

[24] That is in conjunction with Fraunhofer IGDs CityServer 3D

[25] Though there are some third party libraries for providing safe concurrent write access to MongoDB, as e.g. the Open Source MongoMVCC implementation by Fraunhofer IGD: https://github.com/igd-geo/mongomvcc/wiki

[26] BSON is a binary format similar to the popular JSON (Javascript Object Notation)

spatial support for 2D spatial data and also support different spatial reference systems through its spatial index “geohashing”. However in its current implementation, lines and polygons cannot be directly indexed. However point-in-polygon queries are supported[27]. Overall, it can be said that MongoDBs spatial support is clearly inferior to PostGIS. Since the secondary data pool is intended to be used for advanced analysis functionality and MongoDB does not offer any improvements for handling complex relations and performing spatial operations, it is not considered as a suitable choice. (Baas 2012)

CouchDB is an Open Source Document oriented datastore with commercial support available. CouchDB uses a HTTP/REST based protocol for queries. Its main design goal is ease of use and data consistency. Support for spatial data structures is provided by the GeoCouch extension which provides R-Tree based indexing on top of points, lines and polygons. Queries on top of these indexes are limited to bounding box, polygon and radius searches and are implemented on top of the MapReduce paradigm. While it is easy to use, in practice its performance is expected to be low for large spatial data sets. Taken into consideration that very similar to MongoDB, CouchDB offers no improvements for handling complex relations among data entities when compared to PostGIS, it is not considered as a suitable alternative for the secondary data pool.

Neo4j is an Open Source Graph Database with commercial support available. Data is managed as a property graph which consists of vertices that are connected through edges. Both vertices and edges can have properties which themselves can have either primitive values or arrays of primitive values. While it also supports conventional indexes as other DBMS[28], Neo4j physically stores connectivity information locally along with the vertices and edges and also tries to keep this connectivity information in memory when possible. This leads to extremely fast query execution on that information[29]. In comparison, a conventional RDBMS would have to perform at least one join which results in multiple reads at different physical disk locations (i.e. for the individual rows and / or column indexes). However Neo4js inserts often require more complex writes. Also there exists no generic algorithm to distribute a graph across several that works well for arbitrary graphs. Thus, while Neo4j offers support for replication (horizontal scalability), sharding (vertical scalability) is not supported. Neo4j is implemented in Java and provides a Java based API[30] as well as a HTTP REST interface. Both of them offer full support for transactions and several query languages can be used. Cypher is a graph-matching query language (similar as SPARQL for RDF Triple stores) that can be used for data selection, insertion and modification. Gremlin is a graph-traversal query language based on the Groovy Scripting Language that can as well be used for data selection, insertion and modification. Neo4j also offers seamless integration with the Open Source Tinkerpop Framework[31]. For example Tinkerpop Frames can be used to implement persistence for Java Beans through annotating their classes in a similar fashion as JPA ORM frameworks (e.g. Hibernate or EclipseLink) allow for RDBMS. Via Tinkerpop, Neo4j could also be used as a storage backend for the Sesame RDF triple store. Spatial support is provided via the

[27] E.g. to search for points that are located inside a polygon given as part of a query

[28] That includes lexicographical indexes and search functionality based on the Apache Lucene project: http://lucene.apache.org/

[29] time complexity for access to local connectivity information is always O(1), i.e. constant time independent of data size

[30] When using Java Neo4j can also be used in embedded mode so that the database runs within the same process (i.e. the same JVM) in conjunction with the use of Neo4js Java API, the overhead of inter process communication can be completely avoided

[31] http://www.tinkerpop.com/

Neo4j-spatial extension. It provides R-tree based spatial indexing, which however is expected to be not as optimized as the PostGIS implementation. Thus, analysis tasks that solely rely on global spatial indexing are likely to be slower than when implemented on top of PostGIS. However custom indexes (which are trees that in fact are themselves a subset of graphs) could be modelled directly inside the storage data model to speedup certain analysis tasks. The support for expressing spatial data is compatible to the OGC Simple Feature Access specification.

To conclude, Neo4j-spatial vastly exceeds the spatial support of MongoDB's geohashes and CouchDB's GeoCouch while it also offers more convenient handling of highly connected data that has the potential to be actually faster than PostGIS for workloads that exploit local connectivity information. Based on that information Neo4j is the obvious choice. At the time of writing there is no other Open Source Graph Database with spatial support available.

9 API and Access Control System

9.1 Objectives

Pool Data API and Access Control System are tightly bounded together. Access to the Pool data needs to be controlled (both writing and reading) and therefore an Access Control System surrounding the API is needed. This chapter describes the state of art of the *plan4business* Access Control System and the Pool Data API. The next steps in the development are then described.

9.2 Work Done and Progress Achieved

9.2.1 Access Control System

The development of the *plan4business* Access Control System has been divided in two steps: the development and the integration. In the development step, the Access Control System is designed, implemented and tested with the first *plan4business* module. In the integration step, the Access Control System is integrated with the remaining *plan4business* modules to secure them. The first *plan4business* module to be secured is the Layer Manager.

9.2.1.1 Access Control in Layer Manager

The Layer Manager has been developed in Task 4.3 as a tool to support uploading, publishing, configuring, styling and downloading geospatial data, all with respect to the *plan4business* security needs. For more detail description of the Layer Manager please refer to D4.1 Operational System V1 (*plan4business* Consortium 2013). The functionality of the Access Control System is further described.

The structure of the Layer Manager and its relation to the Access Control System is shown in Figure 15.

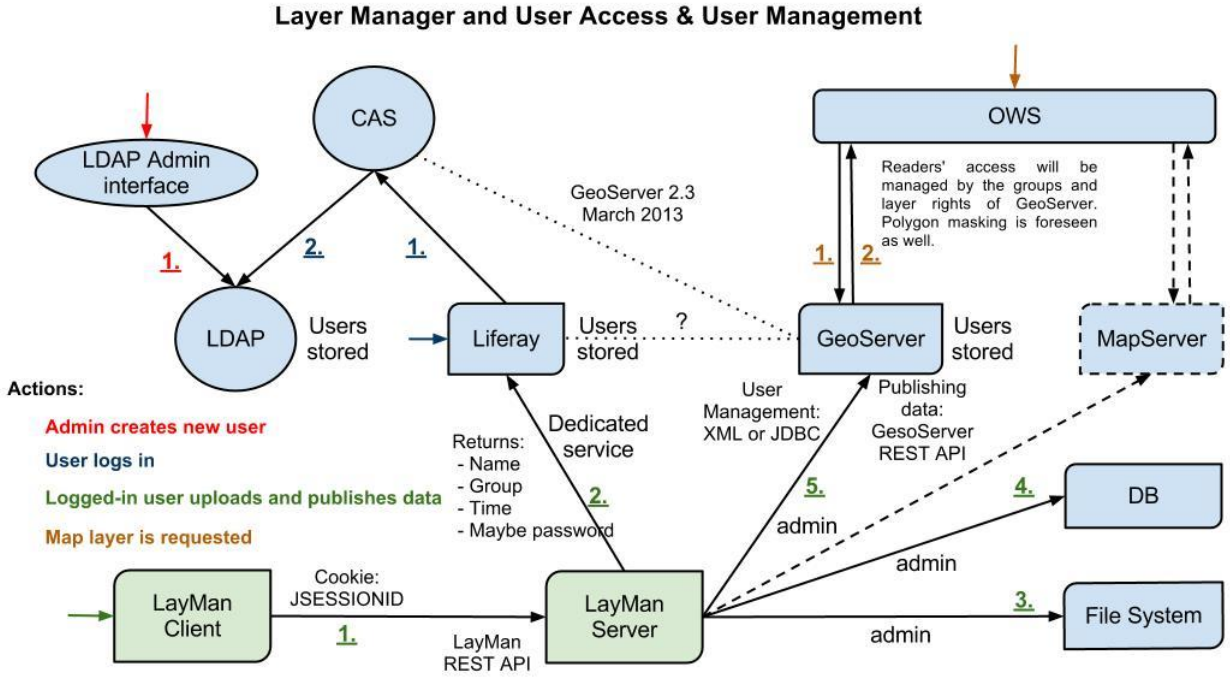


Figure 15 Layer Manager and User Access & User Management

The schema depicted in Figure 15 can be described as follows:

Admin creates new user

The primary user storage is in the LDAP (Lightweight Directory Access Protocol) database, which is accessed through the LDAP admin interface (Figure 15, red 1.).

User logs in

The entire *plan4business* portal is backed by the Liferay[32] system. The user logs in Liferay, which contacts the LDAP database through the CAS (Central Authentication Service) single sign-on system. The users from LDAP are replicated into Liferay as well. (Figure 15, blue 1-2.)

Logged-in user uploads and publishes data
(Writer's access)

Once the user is logged-in, he/she can upload and publish geospatial data (please see the D4.1 Operational System V1 (*plan4business* Consortium 2013) deliverable for detail Layer Manager Client description, in section dedicated to Task 4.3 - Plan Hosting and Feedback Component.). When he/she selects the desired operation, the Layer Manager Client sends the request through the Layer Manager REST API (see below) together with the JSESSIONID provided by Liferay.

[32] http://www.liferay.com/

The Layer Manager Server sends the JSESSIONID to Liferay through a dedicated service. Liferay verifies the JSESSIONID and provided it is valid, it sends back the user info – mainly the name and the group(s) of the user.

From now on, the Layer Manager Server acts as admin to access the file system, the database and the GeoServer. The user's group is used to identify the appropriate file system directory, database schema and the GeoServer workspace. If user belongs to several groups, the desired one comes as a parameter in the REST request. (Figure 15, green 1-5.)

Map layer is requested
(Reader's access)

While the core functionality of the writers' access has been already implemented, the readers' access is work in progress. There have been several experiments showing several possible ways:

The user rights to access different layers will be configured by the means of GeoServer groups and layer security. To achieve that, GeoServer either needs to store the users and groups, or it needs to contact external authority. In contrary to GeoServer documentation, the experiments to retrieve the user groups from LDAP has not been successful yet. The backup solution is to replicate the users in GeoServer through the Layer Manager Server. When user requests a secured layer (through the Plan4Business portal map application), Liferay will log the user into the GeoServer, creating a session and it will provide the session id to the map client, who will then use it to obtain the secured resource.

(OWS access to GeoServer in Figure 15, brown 1-2.)

9.2.2 Pool Data API

9.2.2.1 Layer Manager REST API

The first Pool Data API that has been secured by the Access Control System is the REST API of the Layer Manager. The securing mechanism is described above. For more details about the Layer Manager please refer to D4.1 Operational System V1 (*plan4business* Consortium 2013).

File Manager methods allow management of files in the secured file system. Every user has his/her private upload directory (Table 1).

/fileman

Method	Action	Return Code	Formats
GET	List the user directory	200	JSON
POST	Upload the file. If the file already exists, DO NOT overwrite	201 if created, 409 if the file already exists	
PUT	X	405	
DELETE	Delete all the files.	200	

/fileman/<file>

Method	Action	Return Code	Formats
GET	Get the file	200	JSON
POST	X	405	
PUT	Update the file. If the file does not exist, create it.	200	
DELETE	Delete the file	200	

/fileman/detail/<file>

Method	Action	Return Code	Formats
GET	Get the file details	200	JSON
POST	X	405*	
PUT	X	405*	
DELETE	X	405*	

* we may allow editing the details in future - e.g. set the file projection or add some attributes

Table 1 File Manager

The **Layer Editor** methods allow publishing and managing layers in PostGIS and GeoServer. Every group has its own publishing directory in the file system, database schema and GeoServer workspace (see Table 2).

/layed

Method	Action	Return Code	Formats	Parameters
GET	Get the list of layers	200	JSON	group
POST	Import corresponding file to database and publish in GS	201		group
PUT	X	405		
DELETE	X	405		

/layed/<layer>

Method	Action	Return Code	Formats	Parameters
GET	In future, this should return KML or WMS capabilities	200		
POST	X	405		
PUT	Update / overwrite the layer	200		group
DELETE	Delete layer	200		group

/layed/config/<layer>

Method	Action	Return Code	Formats	Parameters
GET	Get layer parameters	200	JSON	group
POST	X	405		
PUT	Set layer parameters	200	JSON	group
DELETE	X	405		

Table 2 Layer Editor

Workspaces of GeoServer can be managed too (see Table 3).

/layed/workspaces this corresponds to GeoServer REST API. LifeRay authorisation is checked before that.

Method	Action	Return Code	Formats
GET	List all workspaces	200	JSON
POST	Create a new workspace	201	JSON
PUT	X	405	
DELETE	X	405	

/layed/workspaces/<ws>

Method	Action	Return Code	Formats
GET	Returns workspace ws	200	JSON
POST	X	405	
PUT	Modify workspace ws	200	JSON
DELETE	Delete workspace ws	200	JSON

Table 3 Workspaces of GeoServer

9.2.2.2 Analysis Engine API

Analysis Engine - REST services – query management

Function	Method	URL	Parameters	Response
Insert query	GET	/rest/query/insert?	query user_id	JSON
Update query	PUT	/rest/query/update?	query_id query user_id	JSON
Delete query	DELETE	/rest/query/delete?	query_id user_id	JSON
Get queries	GET	/rest/query?	user_id	JSON

Table 4 Analysis Engine - REST services – query management

Analysis Engine – web services – data access

Function	Method	URL	Parameters	Response
Get KML with NL	GET	/KMLServlet?	queryId	KML file with NL
Get KML with data	GET	/KMLServlet?	queryId BBOX	KML file
Get KML with data with timestamp	GET	/KMLServlet?	queryId BBOX when	KML file
Get KML with data during period	GET	/KMLServlet?	queryId BBOX begin end	KML file

Table 5 Analysis Engine – web services – data access

9.2.2.2.1 Using HTML client

1. Main page with the HTTP form:

An SQL query can be inserted and sent for execution (Figure 16).

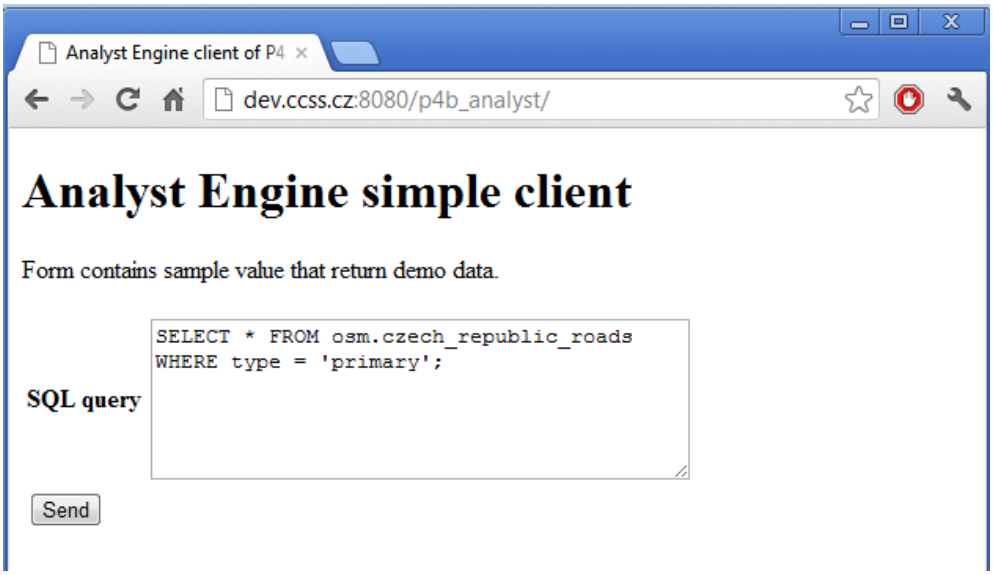
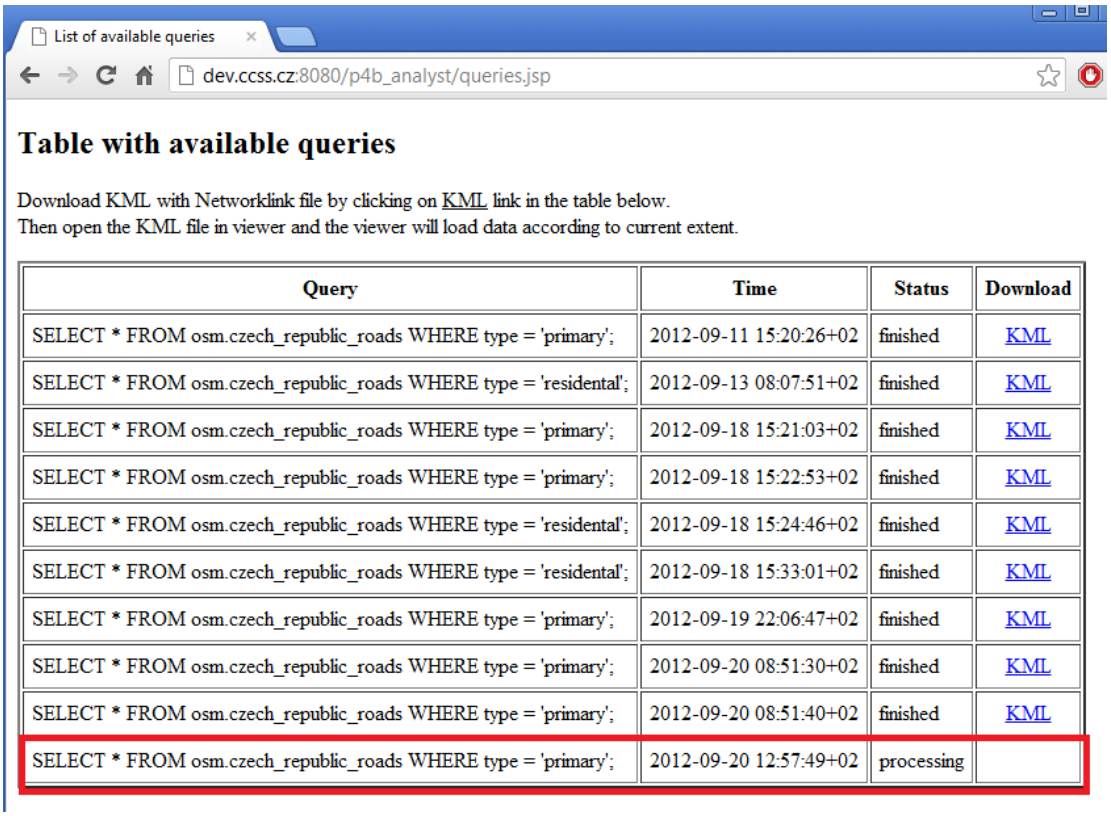


Figure 16 Simple HTML form page

2. Table with available queries – 1st phase

If the Analysis Engine accepts incoming SQL query, it will send it to DBMS and redirect client to page with list of all stored queries. Page *queries.jsp* contains HTML table with all stored queries. If inserted query is still running, its row in table contains status "processing", see Figure 17, and only KML files of finished queries can be downloaded.



Query	Time	Status	Download
SELECT * FROM osm.czech_republic_roads WHERE type = 'primary';	2012-09-11 15:20:26+02	finished	KML
SELECT * FROM osm.czech_republic_roads WHERE type = 'residental';	2012-09-13 08:07:51+02	finished	KML
SELECT * FROM osm.czech_republic_roads WHERE type = 'primary';	2012-09-18 15:21:03+02	finished	KML
SELECT * FROM osm.czech_republic_roads WHERE type = 'primary';	2012-09-18 15:22:53+02	finished	KML
SELECT * FROM osm.czech_republic_roads WHERE type = 'residental';	2012-09-18 15:24:46+02	finished	KML
SELECT * FROM osm.czech_republic_roads WHERE type = 'residental';	2012-09-18 15:33:01+02	finished	KML
SELECT * FROM osm.czech_republic_roads WHERE type = 'primary';	2012-09-19 22:06:47+02	finished	KML
SELECT * FROM osm.czech_republic_roads WHERE type = 'primary';	2012-09-20 08:51:30+02	finished	KML
SELECT * FROM osm.czech_republic_roads WHERE type = 'primary';	2012-09-20 08:51:40+02	finished	KML
SELECT * FROM osm.czech_republic_roads WHERE type = 'primary';	2012-09-20 12:57:49+02	processing	

Figure 17 HTML table with queries, highlighted running query

3. Table with available queries – 2nd phase

Users can repeatedly reload *queries.jsp* page to find out if his query is finished. After query is finished, active link to download KML with NetworkLink appears in table row, see Figure 18.

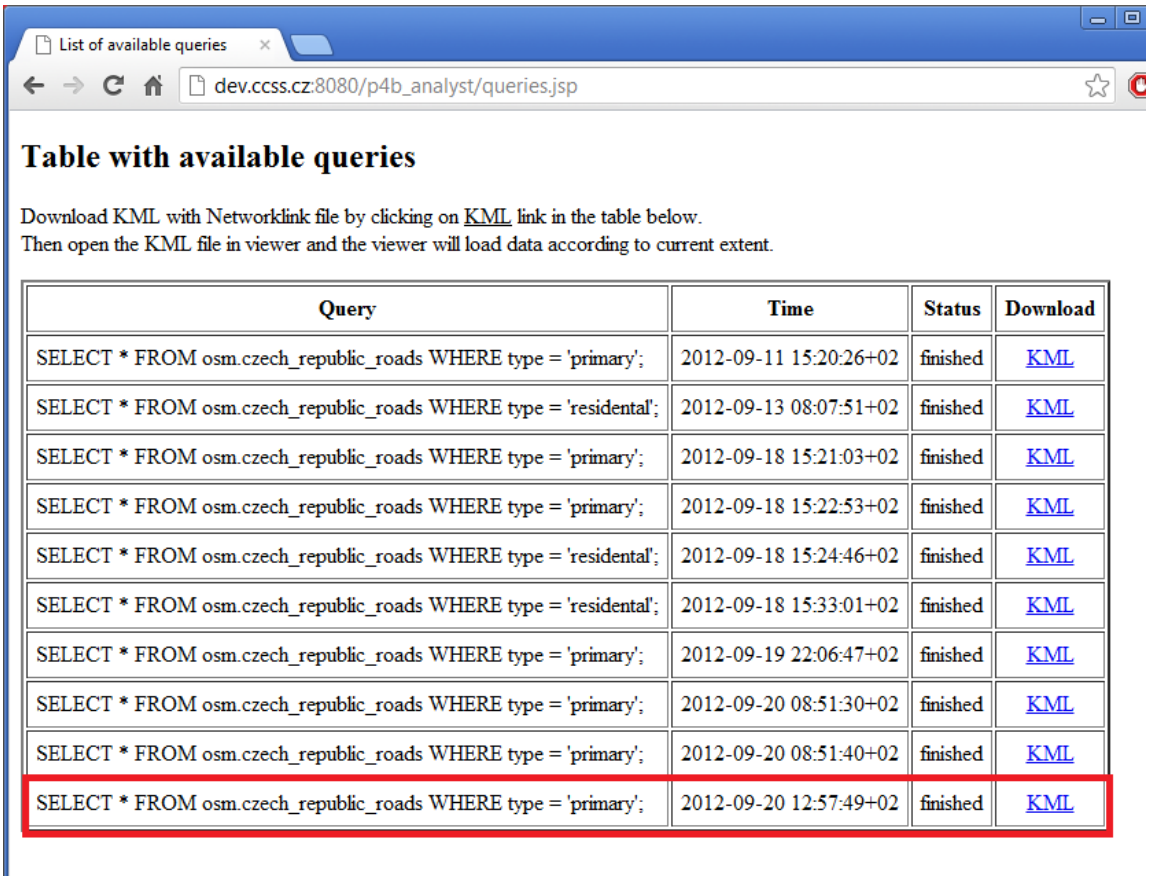


Figure 18 Table with finished query

4. Download KML with NetworkLink

Active link to download result of query in form of dynamically loaded KML appears in table after finishing query in DBMS and reloading page. KML will be downloaded after clicking on this link. KML contain NetworkLink mechanism to dynamically load data.

5. Dynamically data loading

KML with NetworkLink should be open in some client application, e.g. Google Earth. Data will be dynamically loaded according to the current map window extent.

9.2.2.3 Support of Different Formats

The pool data pool API supports a basic set of formats. Data conversion is done on the side of the GeoServer installation. The data formats are depicted in Figure 19.

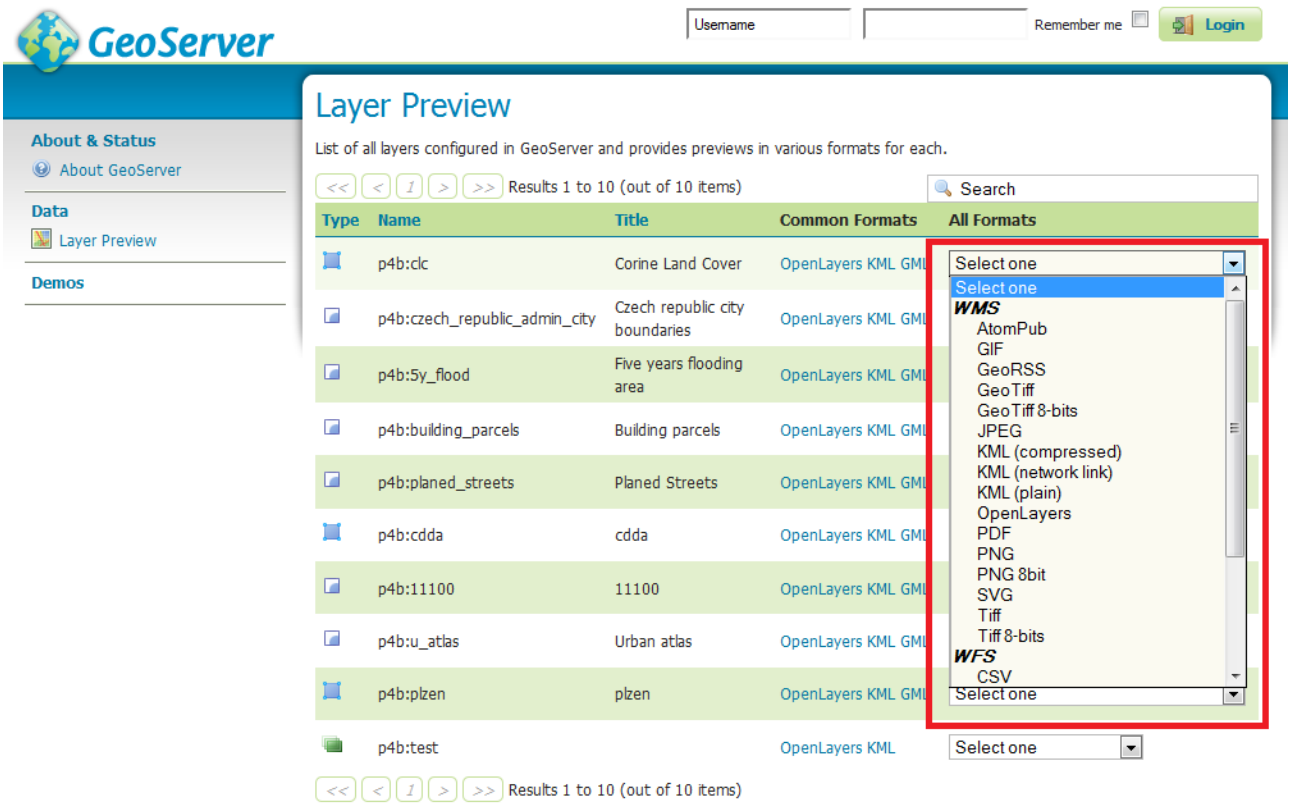


Figure 19 Currently supported formats for the pool data API.

10 Next Steps

10.1 General

The development of the server side components of the *plan4business* platform are in line with the Service Levels and related Milestones described in Section 4.2. Slight delay due to unexpected shortage of staff at Fraunhofer IGD occurred in the development of the Integration and Storage Engines. However, the delay will be eliminated within next month.

The next steps are aiming to reach Service Level 3 in month 15 and Service Level 4 in month 18. The Integration Engine and the Analysis Engine are developed in a way that they can act separately. A common link between the Engines provides the Storage Engine and metadata catalogue. WP5 works closely in cooperation with WP4 (Plan Integration & Analysis Clients) which should provide the integration of the Engines through the client side of the *plan4business* portal.

10.2 Integration Engine

Provide functionality needed by the Plan Integrator:

- Automatic publishing of metadata of integrated data sets in the metadata catalogue, to make them available for analysis and data access. The metadata should be INSPIRE compliant.
- Implementation of a schema repository that allows schema detection on uploaded data sets.
- Format detection and conversion of vector formats not supported directly for the schema transformation.
- Integration of raster data not directly publishable in the metadata catalogue.

10.3 Analysis Engine

The next development of the Analysis Engine will be focused on performance improvement and more convenient approach to particular analyses. This development will be mainly influenced by refinement of the use case development and data availability. Focus will be also given to schema improvement to achieve better data integrity and on analyses performed on the secondary data storage. The secondary data storage is also considered to be as analyses result data storage.

The Analysis Engine will be also extended to enable better visualisation of the results. Currently, it is focused mainly on spatial data with certain attributes. The Analysis Engine will exploit the integration of spatial and non-spatial data and visualise them in the form of a generated report, charts etc.

Next steps include also the API improvement. The *plan4business* API for the Analysis Engine is described in Section 9.2.2.2.

10.4 Storage Engine

- In the next step will investigate possible case studies for making benefit of the graph-based database concept.
- Refinement and/or extension of data models according to analysis requirements.

10.5 API and Access Control System

The development of the API and access control components is in line with the development of other *plan4business* platform engines. The next steps in the development include:

- **Access Control System -** Next steps for Access Control System cover securing the OWS services, so every user (data reader) can access only those parts that he/she has access to. Then, access to the Analysis Engine API should be controlled. Finally, the HALE Integration Engine should be integrated with the Access Control System as well.
- **Data Pool API**

  LayMan REST API - is still under development and its functionality can be extended or modified as the need arise. The calls to store feedback on a given data set need to be added.

  Analysis Engine API - API will be improved for better performance in retrieving analysis results. Also new functions for query complexity information will be added.

  Additional data formats converters over the GeoServer set of supported formats (Figure 19) based on user requirements.

References

Baas, B., 2012. *NoSQL spatial - Neo4j versus PostGIS*. Geographical Information Management and Applicaitons (GIMA). Available at: http://igitur-archive.library.uu.nl/student-theses/2012-0822-200532/MSc_report_final_v101.pdf.

Data Harmonisation Panel, 2013. HUMBOLDT Alignment Editor. Available at: http://www.esdi-community.eu/projects/hale.

Fraunhofer-Gesellschaft zur Foerderung der Angewandten Forschung e. V., 2012. Seventh Framework Programme, Grant Agreement No 296282 plan4business - A service platform for aggregation, processing and analysis of urban and regional planning data, Annex I - Description of Work.

Google, 2013. KML Tutorial. Available at: https://developers.google.com/kml/documentation/kml_tut.

Ježek, J., Kepka, M. & Mildorf, T., 2013. plan4business – SERVISNÍ PLATFORMA PRO AGREGACI, ZPRACOVÁNÍ A ANALÝZU ÚZEMNĚ PLÁNOVACÍCH DAT MĚST A REGIONŮ. In *Sborník sympozia GIS Ostrava 2013*. Sympozium GIS Ostrava 2013. Ostrava: Vysoká škola báňská - Technická univerzita.

Open Geospatial Consortium, 2012. Open Geospatial Consortium. Available at: http://www.opengeospatial.org [Accessed February 18, 2012].

plan4business Consortium, 2013. *D4.1 Operational System V1 (draft)*.

Wikipedia contributors, 2013a. Application programming interface. *Wikipedia, the free encyclopedia*. Available at: http://en.wikipedia.org/w/index.php?title=Application_programming_interface&oldid=542275587 [Accessed March 10, 2013].

Wikipedia contributors, 2013b. Unified Modeling Language. *Wikipedia, the free encyclopedia*. Available at: http://en.wikipedia.org/w/index.php?title=Unified_Modeling_Language&oldid=543637865 [Accessed March 17, 2013].