



# WEB-BASED TECHNOLOGIES FOR DISPLAY AND ANALYSIS OF SATELLITE BASED INDICATORS IN WATER MANAGEMENT

---

Institute of Surveying, Remote Sensing and Land Information  
University of Natural Resources and Life Sciences, Vienna

## **Master's thesis**

Requirements for the academic degree of

**Dipl.-Ing.**

for the master's programme  
Environment and Bio-Resources Management

by

**Martin Šiklar, BSc**

June, 2015

**Supervisor 1:** Univ.Prof. Dr.rer.nat. Clement Atzberger

**Supervisor 2:** Dr. Francesco Vuolo

## Acknowledgements

I would like to thank my supervisor Dr. Francesco Vuolo for guiding me through the process of writing my thesis and always having the door open for me waiting with good advice and help.

Furthermore I have to specially mention my mother and father who supported me during my entire time of study and made it possible that I could light-heartedly focus on my education. As well as my brother who has always been great inspiration and motivation for me.

Many thanks to my colleagues and friends, especially David who helped me during my degree with his knowledge and friendship and assisted me at various (late night) learning sessions.

I am also very grateful that I had the opportunity to study abroad at Iowa State University (Ames, IA) during the spring term of 2013, and for all the great people I met there, especially Nina, who had her ears open for me and pointed me into the right direction.

## Declaration

I hereby declare that I have written this Master's Thesis completely by myself, and that I haven't used any sources and material without declaration in the text. All thoughts from others or literal quotations are clearly marked. Furthermore I declare that this Master's Thesis was not used in the same or in a similar version to achieve an academic grading or is being published elsewhere.

*Baden, June 2015*

# Abstract

## English

Actuality and accessibility make the World-Wide-Web a very well suited medium for the communication of socially relevant information. For this reason is this thesis aiming to provide an overview on current open-source web-based mapping technologies suitable for the display and exploitation of spatial data such as the one derived from the spectral reflectance of satellite imagery.

The author first describes all visual elements of a typical Web-GIS that can be seen as the interface of such an application and secondly the technical components that are needed for its set-up and their interactions.

In this study, the author focuses on applications in the context of water management and concludes his findings with two case studies, one with a regional scale in the agricultural area of Marchfeld, Austria within the context of water management and water use, and the second with a national scale located in Kenya, Africa within the context of drought monitoring and anomaly analysis service.

For each of these case studies, which are supported by real world data, The author applies the open-source technologies described in his findings from the literature review, by developing a web-based GIS application using both Server- and Client Side programming (PHP, JavaScript and related Frameworks) combined with database systems that are suitable for spatial data (PostGIS) and discusses them in terms of their usability, performance and compatibility on different platforms and devices.

## Deutsch

Aktualität und Erreichbarkeit machen das World-Wide-Web zu einem geeigneten Medium zur Kommunikation von sozial relevanten Informationen. Das Ziel der Arbeit ist es deshalb einen Überblick über frei verfügbare und web-basierte Kartografie Technologien zu geben, welche geeignet sind räumliche Daten abgeleitet von spektralen Reflexionen von Satellitenbildern zu visualisieren und auszuwerten.

Der Autor beschreibt zunächst alle visuellen Elemente einer typischen Web-GIS Anwendung und darauf folgend die technischen Bestandteile und ihre Interaktionen.

Diese Arbeit legt den Schwerpunkt auf Anwendungen im Bereich des Wassermanagements im Rahmen von zwei Fallstudien: Die Erste, auf einer regionalen Ebene in der landwirtschaftlich genutzten Region des Marchfeldes in Österreich, im Kontext von Wasserverbrauch und -management. Die Zweite, auf nationaler Ebene in Kenia, Afrika im Kontext von Dürrebeobachtung und Anomalie-Untersuchung.

Für jede der Fallstudien wurden die frei verfügbaren Technologien aus der vorange-

gangenen Literaturrecherche angewandt, indem eine web-basierte GIS-Anwendung erstellt wurde und das mit Hilfe von server- und clientseitigen Programmiersprachen (PHP, JavaScript) kombiniert mit Datenbanksystemen welche geeignet für den Umgang mit räumlichen Daten (PostGIS). Abschließend werden diese Technologien unter den Blickpunkten ihrer Benutzerfreundlichkeit, Ladezeiten und Kompatibilität auf verschiedenen Plattformen und Geräten diskutiert.

# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Motivation . . . . .	1
1.2	Aim . . . . .	2
<b>2</b>	<b>Literature Review and Method</b>	<b>3</b>
2.1	Types of web-maps . . . . .	3
2.1.1	Static . . . . .	3
2.1.2	Dynamic . . . . .	3
2.1.3	Animated . . . . .	3
2.1.4	Real-time . . . . .	4
2.1.5	Interactive . . . . .	4
2.1.6	Analytic - Web GIS . . . . .	4
2.2	Elements of a Web-GIS . . . . .	5
2.2.1	Basemap . . . . .	5
2.2.2	Thematic Layer Overlay . . . . .	6
2.2.3	Further (optional) components . . . . .	7
2.3	Technical components of a Web-GIS . . . . .	8
2.3.1	Client/Server communication . . . . .	8
2.3.2	Server side . . . . .	10
2.3.2.1	PHP Hypertext Preprocessor . . . . .	10
2.3.2.2	PostGIS Database . . . . .	11
2.3.3	Client side . . . . .	12
2.3.3.1	HTML - Hypertext Markup Language . . . . .	12
2.3.3.2	CSS - Cascading Style Sheets . . . . .	13
2.3.3.3	JS - JavaScript . . . . .	14
2.3.3.4	JavaScript Libraries . . . . .	16
2.3.3.4.1	Web-Mapping . . . . .	17
2.3.3.4.2	Charting . . . . .	18
2.3.3.4.3	jQuery . . . . .	19
2.4	Case studies . . . . .	21
2.4.1	Marchfeld, Austria . . . . .	21
2.4.1.1	Data . . . . .	22
2.4.1.2	Stakeholder . . . . .	24
2.4.2	Kenya, Africa . . . . .	26
2.4.2.1	Data . . . . .	26
2.4.2.2	Stakeholder . . . . .	27

<b>3</b>	<b>Results</b>	<b>28</b>
3.1	Marchfeld, Austria . . . . .	28
3.1.1	Database set-up . . . . .	28
3.1.2	Technical components of the final Web-GIS application . . . .	31
3.1.2.1	Filtering of data - Query database . . . . .	33
3.1.2.2	Data processing . . . . .	35
3.1.2.3	Visualization of the map . . . . .	37
3.1.2.4	User interaction and visualization of charts and tables	39
3.1.3	Use of application . . . . .	42
3.1.3.1	Map - analysis of spatial trends and dependencies . .	42
3.1.3.2	Tables and Charts - aggregation over time and space	44
3.2	Kenya, Africa . . . . .	47
3.2.1	Database set-up . . . . .	47
3.2.2	Technical components of the final Web-GIS application . . . .	51
3.2.3	Use of application . . . . .	52
3.2.3.1	Map . . . . .	52
3.2.3.2	Time Series Chart . . . . .	53
3.2.3.3	Multi-year-seasonality-chart . . . . .	54
<b>4</b>	<b>Discussion and Conclusions</b>	<b>56</b>
4.1	Advantages and challenges of web-mapping . . . . .	56
4.2	Lessons learned and outlook . . . . .	57

## List of Figures

1	Classification of web-maps based on Kraak, 2001 and modified after (Detwiler, 2014) . . . . .	3
2	Example of a real-time interactive web-map (Vaidyanathan, 2014) . .	4
3	(a). Tiles across various scales (Quinn, 2014). (b). Single tile: MapQuest-Basemap (OSM, 2013). . . . .	6
4	(a). Choropleth map, showing US population density for each state. (Agafonkin, 2014). (b). Interactive map with a pop up, showing conflicts in Africa (own illustration). . . . .	7
5	Same web-page with a CSS Stylesheet (left side) and without (right side)	14
6	Hierarchical structure of the Document Object Model (John, 2008) .	15
7	(a). Google Charts Combo Chart showing monthly coffee production by country (Google, 2015). (b). D3 Line Chart showing Baseball statistics on strike outs (NYT, 2012). . . . .	19
8	Overview on the Marchfeldregion showing the "Marchfeldkanal" and the three groundwater recharging stations Stallingerfeld, Russbach/Mühlbach, Speltengarten. Edited after (Karl and Neudorfer, 2002, p.583). . . . .	22
9	Table <i>agg_data</i> from the PostGIS database containing all daily values (Marchfeld). . . . .	29
10	Table <i>regions</i> from the PostGIS database containing region names and geometries (Marchfeld). . . . .	30
11	Table <i>landcover_type</i> from the PostGIS database containing the land-cover types (Marchfeld). . . . .	31
12	Final Web-GIS application for the first case study (Marchfeld). . . . .	32
13	Final map display with tile-layer basemap and the coloured regions overlay (Marchfeld). . . . .	39
14	Table that shows the selected statistic, date and the computed values for the municipality, district and overall region. . . . .	40
15	Google Line Chart displaying Evapotranspiration for the municipality "Obersiebenbrunn". . . . .	42
16	(a). Spatial trend of the crop water requirement (left) and precipitation (right) for June 2013 in the Marchfeldregion. . . . .	43
17	(a). Dependence of evapotranspiration (left) on crop coefficient (right) for June 2013 in the Marchfeldregion. . . . .	43
18	Ranking of municipalities after aggregated evapotranspiration from May to October 2013 in the Marchfeldregion. . . . .	44



19	Crop coefficient for the entire growing season 2013 for the municipality "Zwerndorf", its district and the overall region. . . . .	45
20	Comparison of aggregated evapotranspiration on municipality-, district- and regional level for "Deutsch-Wagram". . . . .	46
21	Table <i>agg_data</i> from the PostGIS database containing all monthly values (Kenya). . . . .	48
22	Table <i>land_zones</i> from the PostGIS database containing the names of the landzones (Kenya). . . . .	49
23	Table <i>adm_units</i> from the PostGIS database containing the names and geometries of the counties (Kenya). . . . .	50
24	Table <i>zones</i> from the PostGIS database showing the link between <i>adm_uni_id</i> and <i>lz_id</i> (Kenya). . . . .	51
25	Final Web-GIS application for the second case study (Kenya). . . . .	52
26	(a) <i>NDVI</i> and (b) precipitation for March 2015 in Kenya, Africa. . . . .	53
27	<i>NDVI</i> for the county Nakuru for all landzones. . . . .	54
28	<i>VCI</i> multi-year-seasonality-chart for agro-pastoralists in "Narok", Kenya. . . . .	55

## List of abbreviations

- A** API - Application Program Interface
- C** CGI - Common Gateway Interface  
CSS - Cascading Style Sheets  
CWR - Crop Water Requirement
- D** D3 - Data Driven Documents  
DOM - Document Object Model
- E** EO - Earth Observation  
ETp - Evapotranspiration
- G** GIS - Geoinformation System
- H** HEX - Hexadecimal  
HTML - Hyper Text Markup Language  
HTTP - Hyper Text Transfer Protocol
- I** IE - Microsoft Internet Explorer  
IP - Internet Protocol
- J** JS - JavaScript
- K** Kc - Crop Coefficient
- L** LAI - Leaf Area Index
- N** NDMA - National Drought Management Authority  
NDVI - Normalized Difference Vegetation Index  
NIR - Near-infrared
- O** OSM - Open Street Map
- P** PHP - PHP: Hypertext Preprocessor
- S** SQL - Structured Query Language
- T** TCP - Transmission Control Protocol

- U**    UI - User Interface  
      URL - Universal Resource Locator
- V**    VCI - Vegetation Condition Index  
      VIS - Visible light/energy
- W**    WDV - Weighted Difference Vegetation Index  
      WWW - World Wide Web

# 1 Introduction

The purpose of this thesis is to provide an overview on current open-source web-based mapping technologies suitable for the display and exploitation of spatial data such as the one derived from the spectral reflectance of satellite imagery. In this study, the author focuses on applications in the context of water management and concludes his findings with two case studies supported by real world data.

This chapter starts with the authors motivation for this topic and then declares the aims and goals of this work and presents its structure.

## 1.1 Motivation

The scientific research process is divided in a chain of sub processes starting with hypothesis building and ending with the generation of new insights and information. In most cases, this newly generated information from underlying data has to be visualized and communicated to stakeholders.

This process of presentation and communication is not only the last part of the research process but also one of the most important components.

Especially when it comes to socially relevant information, for example information on water use, drought monitoring or food security, the communication and visualization is crucial but at the same time challenging. In most cases the data is only useful if its communication meets three requirements:

- **Actuality:** Only if the data and information is up-to-date and communicated fast enough so there is time to react accordingly, the data is useful for its stakeholders.
- **Accessibility:** The data has to be accessible at any time and to everybody in need of it.
- **Comprehensibility:** If both requirements above are met but the communication and visualization of the information is too hard to understand or confusing, it is useless to its recipients.

What technologies of data communication and visualization are able to cover these three basic requirements?

Already in the year 2000, when the internet was still a relatively new phenomenon, and web-mapping itself was only a very small border area in the context of the WWW (Google Maps was not introduced until 2005), Van Elzakker pointed out the two main strengths of web-maps: Accessibility and Actuality:

*"Information, including web-maps, is easily accessible through user-friendly web browsers, 24 hours a day and not hindered by political and geographical boundaries.[...] The WWW, makes it possible to supply the users with really up-to-date geographic and cartographic information"* (van Elzakker, 2000, p.35).

When it comes to the third requirement "Comprehensibility", it is really up to the author to display and communicate the information in a way that the recipients are able to understand it. The Internet in general is a technology that is integrated in the daily life of almost every "modern" person and is therefore reliable and proven to be user-friendly. About 42.3% of the worlds population uses the internet today and the user growth in the last four years was 741 % (InternetWorldStats, 2014).

## 1.2 Aim

The previous insights suggest that the Internet and web-based technologies are very well suited for data communication and visualization. The author's aim is therefore to:

- First describe what types of web-based maps and open source technologies for data visualization currently exist.
- Secondly, to give an overview on all the components needed for creating a web application, describing the visible elements of a web-map as well as the technical components and their interactions underneath the surface.
- The third and last part consists of two case studies, based on real data derived from spectral reflectance of satellite imagery:
  - The first case study has a regional scale in the agricultural area of **Marchfeld, Austria** within the context of water management and water use application.
  - The second case study has a national scale and is located in **Kenya, Africa** within the context of drought monitoring and anomaly analysis service.
- For each of these case studies, the author applies the open source technologies described in his findings from the literature review, by developing a web-based GIS application, and discusses it in terms of their usability, performance and compatibility.

## 2 Literature Review and Method

### 2.1 Types of web-maps

The first classification of web-maps was made by Kraak in the year 2001. Since then the WWW made a huge step forward and evolved enormously. Nevertheless is this first classification still a good starting point and serves well as an introduction to the topic of web-mapping. Kraak divides web-maps into various categories, where the rank of the category stands for the amount of sophistication of such application. Figure 1 shows a slightly modified and reduced version of this classification (Kraak, 2001).

#### 2.1.1 Static

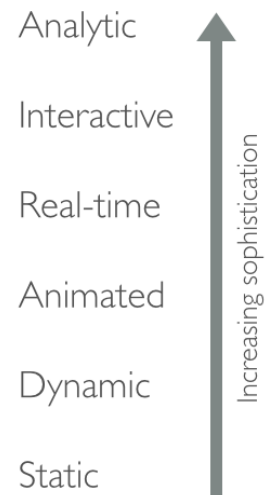
The simplest form of a web-map is a static image linked on a web-page or a scan or photography of a map. Static maps may allow the user to zoom in/out and pan, but in general the possibilities of interaction with these maps are very limited. The general use of static maps is to display spatial information that doesn't change over a longer time period, i.e. country borders.

#### 2.1.2 Dynamic

Dynamic web-maps are being generated from new every time the user reloads the page. This means that this map type is able to display data that changes over time. Usually the data is stored in some file or database and queried on every page visit. The most common type of dynamic web-maps are weather maps that for example display temperature over a certain regions.

#### 2.1.3 Animated

Animated maps do not only display information of one point in time but are able to depict a trend and changes in data. Again, the probably most



*Figure 1: Classification of web-maps based on Kraak, 2001 and modified after (Detwiler, 2014)*

common form of animated maps is weather maps, for example showing the movement of cyclones and anticyclones over the continent. Note however that animated doesn't necessarily have to be dynamic. A static and animated map would for example be a map depicting the presidential election results for the last decades.

#### 2.1.4 Real-time

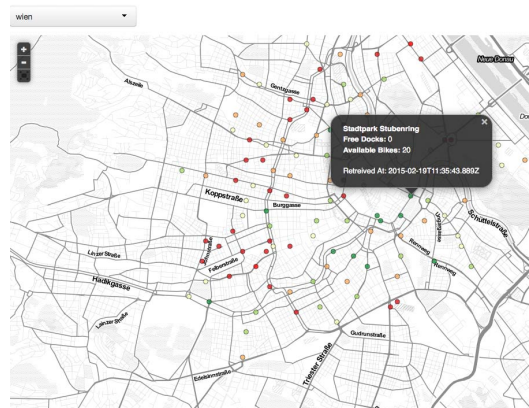
Real-time web-maps are an extension to dynamic web-maps. They are also generated from new every time the user enters the page, but as an extension they are able to handle real-time data. This means that usually there is some kind of automatic process lying beneath the applications surface that collects the data, processes it and stores it in a database or file that is being queried.

#### 2.1.5 Interactive

Interactivity can also be seen as an extension. This feature allows the user the interact with the map, for example to click on areas to get more information, query and filter data, change colors, toggle layers, etc.

In contrast to the past, nowadays almost every web-map is interactive and gives the user the power to filter what he/she wants to see. *"Having a separate category for interactive web-maps seems a bit like having separate categories for college students that own a cell phone and those that don't."* (Detwiler, 2014, s.p.)

An example of an interactive and real-time web-map can be seen in figure 2. This map visualizes about 100 bike sharing systems around the world. The user can select a city from a drop-down menu and then see the distribution of available bikes across the city. By clicking on the bike stations, the user gets more information about the number of available bikes via a pop-up window.



#### 2.1.6 Analytic - Web GIS

The last category with the highest amount of sophistication is an analytic map. It can be seen as a synonym to "Web-GIS" and requires interactivity and allows the user to perform analytic operations. The probably

Figure 2: Example of a real-time interactive web-map (Vaidyanathan, 2014)

most commonly used analytic map is Google Maps, which for example allows the user to analyse the road network to determine the best route between two points. Other typical features of such maps might allow the user to perform GIS operations like buffer, intersect, or graphical analysis like charting and plotting of selected spatial information.

## 2.2 Elements of a Web-GIS

An analytic web-map or *Web-GIS*, as can be seen above, is one of the most sophisticated web-mapping types. The following chapter describes the visible elements on the surface of such Web-GIS applications - simply spoken what the user sees when he/she enters the web-page.

### 2.2.1 Basemap

A commonality of all web-maps is the so-called basemap. It is the foundation of every application and gives the user a geographical orientation and puts the displayed information into context.

Most of the web-basemaps today, are built out of small tiles, usually with a size of 256x256 pixels each. Depending on the zoom-level and extent a certain number of tiles are mosaicked together and displayed.

The big upside to tiled layers is that they are much faster to load and therefore have a much better performance: The tiles are pre-generated and stored on a server and only those parts of the map that are being viewed by the user are downloaded and rendered. Figure 3 (a) depicts the tile-architecture across various scales showing the extent of the mosaicked tiles, forming a "pyramid"-like shape (Quinn, 2014).

When using tiles the user doesn't have control which thematic elements he/she wants to see on the map. If for example roads, buildings, etc. are part of the tile image, they can't be turned on or off, because as described above, tiles are static and pre-generated images. Figure 3 (b) shows an example of a tile from the MapQuest-Basemap including road networks, rivers, natural areas, etc.



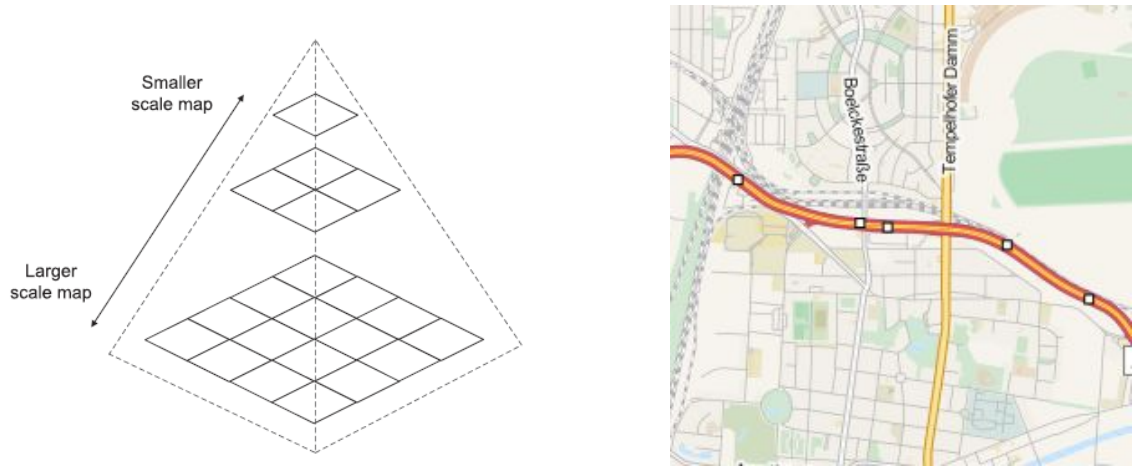


Figure 3: (a). Tiles across various scales (Quinn, 2014). (b). Single tile: MapQuest-Basemap (OSM, 2013).

### 2.2.2 Thematic Layer Overlay

The reason why users visit a map usually isn't the basemap but its thematic layers. If the map's title for example is "Water supply infrastructure in Austria", the thematic layer could be an overlay showing water storage facilities and pipelines. These thematic layers can also be composed out of pre-generated tiles but are mostly shapes like circles, lines, polygons, rectangles and markers. To be able to display real-time data, these shapes are being queried out of a database and rendered on top of the map. If a map has more than one thematic layer, usually the user can toggle between those layers via a control element.

In general these thematic (spatial) layers are attached to other data and information, there are various ways to display and exploit this data, two very common options are described below:

- The shape elements can have a fill color, where the intensity of the color corresponds to the data value. These maps are very common and called "choropleth maps" and are usually combined with a legend explaining the colors. Figure 4 (a) shows an example of such a map, showing US population density for each state.
- By hovering over or clicking on each shape element, further information can be displayed by opening a pop-up with text, generating a table or chart. This

however requires the map to be interactive and gives the user power to filter information. Figure 4 (b) depicts an example of such a map, showing conflicts in Africa.

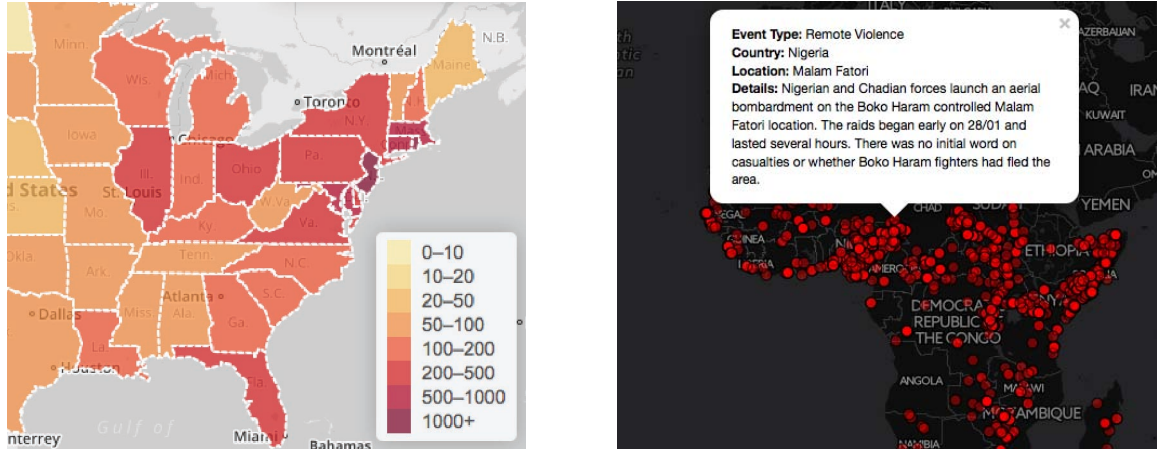


Figure 4: (a). Choropleth map, showing US population density for each state. (Agafonkin, 2014). (b). Interactive map with a pop up, showing conflicts in Africa (own illustration).

### 2.2.3 Further (optional) components

Basemaps and thematic layers are components that can be found in every Web-GIS. Depending on the application however, other elements might be needed, these include:

- **Filtering:** Switches or menus that enable the user to switch between different datasets and allow the filtering and further specification of the data to be displayed on the map.
- **Analytical:** Tools that allow the user to measure distances, calculate areas or define an area of interest via a bounding box.
- **Additional information:** A pop-up with text might sometimes not be enough to display all the information the user needs. Therefore further overlay elements are added on or next to the map that might contain data tables, graphs and charts, etc.
- **Map control:** Switches to turn on and off thematic layers, buttons for zooming and panning, etc.

- **Styling:** Enable the user to change the color scheme of the map, the size, fonts, etc.

## 2.3 Technical components of a Web-GIS

Besides the elements described above, that are visible to the user and can be seen as the interface of a Web-GIS application, there is a variety of important hard- & software components running and interacting underneath the surface. This chapter gives an overview of the system architecture of Web-GIS applications and describes each component and its interactions.

### 2.3.1 Client/Server communication

The basic concept that builds the foundation for the communication via the Internet is the so called Client/Server model:

The WWW is divided into clients (i.e. users) and servers. When a client accesses services (i.e. a website) from a server, a connection is being established between those two hosts via the TCP/IP protocols. TCP and IP are the two main protocols in the web, *"whereas the IP protocol deals only with packets, TCP guarantees delivery of data and that packets will be delivered in the same order in which they were sent"* (Bael, 2014, p. s.p.). A client connects to the Internet via a web browser by typing in the Universal Resource Locator (URL) into its search window. The URL is used to locate files on the web and provides a pointer to any object on a server connected to the Internet. The URL in its standard format can be broken into four parts (Buchanan, 2002, p.818ff):

aaaa://bbb.bbb.bbb/ccc/ccc/ccc?ddd

- **aaaa:**  
Specifies the mechanism to be used by the browser to communicate with the resource. When accessing a Web-GIS application the most common mechanism is the Hyper Text Transfer Protocol (HTTP).
- **//bbb.bbb.bbb**  
Is the Internet node and specifies the node on the internet where the file is located.

- `/ccc/ccc/ccc`

Is the file path (including subdirectories and the filename).

- `?ddd`

Is the argument that depends upon the access method, and the file accessed.

An example URL is:

`http://www.opengeospatial.org/standards/wms#downloads`

where *http* is the file protocol, *www.opengeospatial.org* is the server name, */standards/wms* is the file path and *#downloads* is the specific searched argument on the website.

After the user has typed in the URL into the web browser the communication between the client and server begins, and runs after the following steps (Chantelau and Brothuhn, 2010, p. 133ff):

1. The client establishes a connection to the server via TCP.
2. A HTTP-request is being sent to the server with the specifications from the URL.
3. The server processes the request (i.e. dynamic generation of the website content).
4. The server sends an HTTP-response with the result to the client.
5. The client processes the response (i.e. renders the HTML-Code).
6. The connection is being closed.

In the context of a dynamic, real-time and interactive Web-GIS application these steps have to be extended. For a further analysis, the author breaks down these processes into two parts - the server side and the client side.

### 2.3.2 Server side

The server side involves all processes that are executed on the server and therefore don't run on the clients (users) device. Typically when a web-server receives an HTTP-request from a client, this request doesn't end at the web-server, because not all data and information that have to be processed and later displayed are located on the web-server itself but on other servers and databases. For this purpose the web-server established a connection to other file-servers and databases where the needed data specified by the user is stored. The next chapters give an overview on how the server database communication works in detail.

#### 2.3.2.1 PHP Hypertext Preprocessor

*PHP - Hypertext Preprocessor* is a general-purpose programming language that enables the development of dynamic web-pages. It is mainly focused on server-side scripting and is able to handle operations like collection and management of data, generate dynamic page content, send and receive cookies and much more (PHPnet, 2015). PHP requires three things:

1. **PHP Parser** (i.e. CGI): CGI stands for Common Gateway Interface and enables the storage of executable (dynamic) and not only pre-written (static) files on a web server.
2. **Web server** with a connected PHP installation. The current stable version is PHP 5.6.7.
3. **Web browser** that allows viewing the PHP page through the server.

Usually the data viewed by a user in a Web-GIS application are extracted from a database and dynamically displayed upon the users request. PHP establishes a link between the database and the user by preparing and executing a user defined query and sending the requested and translated information back to the users web browser. The query execution by PHP is done with the help of SQL. This abbreviation stands for Structured Query Language that is a special-purpose programming language for managing data located in relational database management systems.

Web-GIS applications deal with spatial data and therefore use special data types, formats and data query operations (i.e. buffer, union,...) that differ from conventional data queries. To be able to handle spatial data formats, extended database-types are needed. The most widely used open source spatial database today is *PostGIS* database.

### 2.3.2.2 PostGIS Database

PostGIS is an open source software that follows the Open Geospatial Consortium’s specifications. It is a spatial database extender for the PostgreSQL database (an object-relational database management system) and adds extra types like geometries, geographies, and raster as well as functions that apply to these spatial types and are able to query spatial properties and relationships (PostGIS.net, 2015a).

Databases in general are used to store and manage large amounts of data: The data is stored in tables where every single row represents one data entry and every column represents a property of this entry (i.e. date, name, value, descriptions). Entries from different tables can be linked via a unique ID to each other.

An ordinary database has strings, numbers, and dates. PostGIS adds additional types for representing geographic features like polygons, points and linestrings (PostGIS.net, 2015b). The following tables 1 and 2 show how tables in a PostGIS database are set up, depicting one table with city names, population size and coordinates and a second table with region names and coordinates.

*Table 1:* PostGIS table "*cities*" listing city names in Austria, their population and spatial information

id	name	population	geometry
1	Baden	25000	0106002...
2	Linz	190000	0106002...
3	Vienna	1700000	0106002...
...	...	...	...

*Table 2:* PostGIS table "*regions*" containing region names in Austria and their spatial information

id	name	geometry
1	Lower Austria	0106002...
2	Upper Austria	0106002...
3	Vienna	0106002...
...	...	...

Usually users don’t want to see all the data contained in a table at once but only a subset defined by some previous user input. To filter the data SQL-Statements are

used that query the database and return the requested data subset. In dynamic web-pages these SQL-Statements are constructed with PHP on the server. The following example shows how a spatial SQL query of a PostGIS database looks like:

```

1 <?php
2 #... connect to database ...
3
4 #Construct and execute SQL-Statement
5 $result = $db->query("SELECT cities.name, cities.population
6                      FROM regions, cities
7                      WHERE ST_CONTAINS(regions.geometry, cities.geometry)
8                      AND regions.name = '" . $UserSelectedRegion. "'");
9
10 #Fetch the data selected by the query
11 $r = $result->fetchAll();
12 ?>

```

In this case both tables are queried for the cities names and its population that are contained by the region specified by a user (i.e. via a user input field) using the PostGIS ST\_CONTAINS() function.

### 2.3.3 Client side

Once the server processed all the necessary data, an HTTP-response with the result is sent to the client. The client processes the response and renders the HTML-Code so that page can be interpreted and viewed via the web browser.

#### 2.3.3.1 HTML - Hypertext Markup Language

HTML is the Web's core language for creating documents and applications. Its a mark-up language that structures the content on web-pages and was first described in 1991 by Tim Berners-Lee who is called to be the inventor of the Web (W3C, 2015b). The content elements are structured with the help of HTML-tags that have the following form: *< elementstart > content < /elementend >*. The code snippet beneath shows basic HTML code of a simple web-page with text and a picture:

```
1 <html><head><title>Homepage title</title></head>
2 <body>
3 <h1>Heading 1</h1>
4 <div id="text">This is a text content element.</div>
5 
6 </body></html>
```

Plain HTML pages are static, they allow some user interaction like clicking on links and images, filling out forms, etc. but these options are very limited. Web-GIS applications on the other hand contain, as mentioned in the previous chapters, highly interactive and dynamic elements like maps that allow zooming and clicking, as well as graphs and charts with hovering mouse effects, etc. To create and layout such elements two other components that are executed on the client side are needed: CSS and JavaScript. Both CSS and JavaScript code can be embedded inside the HTML page or located in separate files.

### 2.3.3.2 CSS - Cascading Style Sheets

Cascading Style Sheets (CSS) is a simple mechanism for adding style like fonts, colors and alignments to Web documents (W3C, 2015c). By separating the style from the content of a web document, web-pages become independent from its layout. This separation brings several benefits to the web developer as well as the viewer of the website (Meyer, 2004):

1. **Easy maintenance:** Stylesheets can reduce the authors workload significantly by centralizing the commands for certain visual effects.
2. **Compact file size:** CSS requires much less lines of code to style and structure web-pages than HTML. This yields in faster loading times and rendering of web pages.
3. **Rich styling:** CSS allows much more styling options than HTML.

The following example shows how a CSS stylesheet is set up:



```

1 body {
2 background-color:grey;
3 font-size:87.5%;
4 font-family: Arial, 'Lucida Sans Unicode';
5 }
6 #text {
7 background-color:orange;
8 }
9 img {
10 margin-top: 5%;
11 margin-left: 35%;
12 }

```

On the left side of the curly bracket is the so called selector (body, img,...), this defines which element on the web-page will be affected by the declarations inside the brackets. A declaration contains two elements: a property (i.e. background-color) and a value (i.e. grey). The following figure 5 shows the rendered web-page with the HTML code from above, on the left side with the CSS Stylesheet and on the right side without.



Figure 5: Same web-page with a CSS Stylesheet (left side) and without (right side)

### 2.3.3.3 JS - JavaScript

JavaScript (JS) enables executable content to be included in web-pages. This means that web pages need no longer to be static HTML, but can include programs that interact with the user and dynamically create HTML content (Flanagan, 2002). Usu-

ally when referred to JS, client-side JS is meant: *"Client-side JavaScript combines the scripting abilities of a JavaScript interpreter with the document object model (DOM) defined by a web browser"* (Flanagan, 2002, p.3). The Document Object Model is an interface that allows programs and scripts to dynamically access and update the content, structure and style of documents. (W3C, 2015a).

When a HTML page loads, the browser creates an invisible blueprint of all the elements it recognizes as scriptable objects. This blueprint is hierarchical, i.e. starting with the browser window which contains the document, which might contain a form, which might contain several form input elements like buttons, etc. (Goodman, 2002). Figure 6 shows an example of the DOM hierarchy.

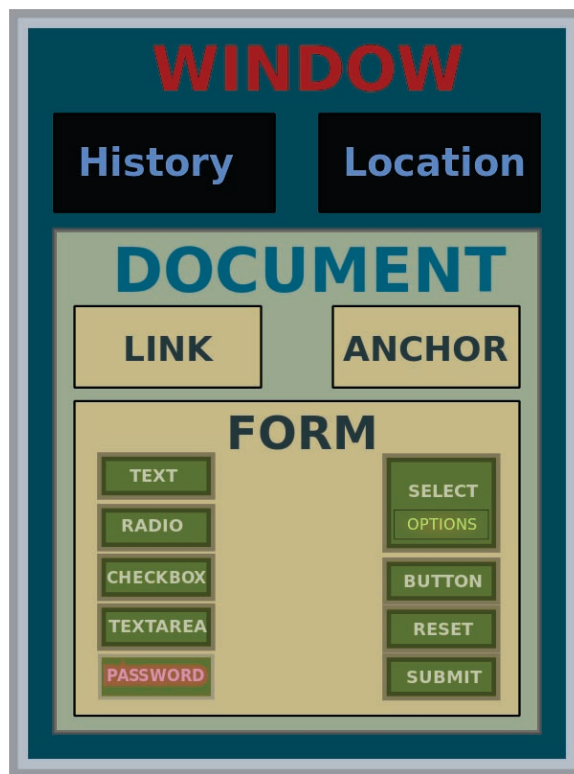


Figure 6: Hierarchical structure of the Document Object Model (John, 2008)

The strength of JS is the so called event handling: Web browsers are programmed to recognize actions like page loading, clicking, typing etc. Each action is seen by the browser as an event which are the key to interactive programming. An event represents the exact moment when something on the web page happens. Four event

types can be distinguished (McFarland, 2011):

1. **Mouse Events:** click, dblclick (double-click), mousemove,...
2. **Document/Window Events:** load (page), resize (window), scroll,...
3. **Form Events:** submit, reset, change (form field),...
4. **Keyboard Events:** keydown (press key), keyup (release key),...

An event can for example trigger a JS code to be executed. This event triggering as well as the result from the executed script is often combined with elements from the DOM. The following code shows for example how a button click event triggers a display of text:

```

1 <button id="button">Display Text</button>
2 <div id="textcontainer"></div>
3
4 <script>
5 document.getElementById("button").addEventListener("click",
6     writeText);
7
8 function writeText() {
9     document.getElementById("textcontainer").innerHTML = "You
10        clicked the button!";
11 }
12 </script>

```

First, two DOM elements are created: a button and a div text container. Secondly, a so-called event listener, that waits for the event to happen, is included (in this case for the click on the button). This click calls a function `writeText()` that changes the content of the previously created DOM element by calling it with its ID and changing its HTML with the `.innerHTML` property.

#### 2.3.3.4 JavaScript Libraries

JavaScript Libraries or Frameworks are a collection of pre-written JS functions that provide solutions to frequent applications of JS. These functions make it easy to complete day-to-day tasks and in many cases replace a lot of lines of code of own JS (McFarland, 2011). Due to the fact that JS is nowadays part of almost every sophisticated web page, its field of application is very broad. This yields in a lot of different

and specialized JS Libraries with different focus. Therefore the author will give an overview on selected libraries suited for web-mapping, charting and general JS day to day tasks in the upcoming chapters.

#### 2.3.3.4.1 Web-Mapping

There is a variety of web-mapping libraries available on the web. With their help the publishing of web-maps became much easier. Authors can use the pre-written functions of these libraries which i.e. allow them to create a basemap, add thematic overlays (markers, points, polygons) with just a couple lines of JS code.

Two very prominent web-mapping libraries are Leaflet and OpenLayers, they are for example used by large companies and organizations like Flickr, Pintrest, Wikimedia, foursquare but also have many applications in the scientific domain, for example an online vegetation map for eastern Africa (VECEA, 2015), or a California Urban Water Use Map (PacificInstitute, 2015) to only name a few.

Both frameworks have a lot of contributors and a well-documented application program interface (API) that allows new users to get acquainted with their functions. Table 3 gives an overview on both of these libraries (Openhub.net, 2015):

*Table 3: Openlayers vs. Leaflet*

	<b>OpenLayers</b>	<b>Leaflet</b>
Version	3.3.0	0.7.3
Size	3.4 MB	385 KB
Lines of Code	230,692	6,689
Developers	185	206
Mobile friendly	Yes	Yes
Cross Browser Compatibility	Yes	Yes

As can be seen from the version numbers in the table, OpenLayers is more mature which makes it more stable than Leaflet. Also its file size is much larger, given that it offers much more functions and options to integrate into web-map applications. Leaflet on the other hand is lightweight, it has a much smaller file size and focuses on the main functions needed to create a web-map, additional features can be included by free available plugins from various contributors. The following Leaflet JS code snippet shows an example on how to create a map with a basic OSM basemap without any thematic overlay centered over London:

```

1 var map = L.map('map').setView([51.505, -0.09], 13);
2
3 L.tileLayer('http://{s}.tile.osm.org/{z}/{x}/{y}.png', {
4     attribution: '&copy; <a href="http://osm.org/copyright">
        OpenStreetMap</a> contributors',
5 }).addTo(map);

```

Comparisons on which library is better are hard to make, since both approaches offer good solutions for most applications.

#### 2.3.3.4.2 Charting

When it comes to charting even more libraries that support web developers with useful functions to create charts and plots, are available. The main differences between these libraries are the number of supported chart types and the degree of customization they offer. Table 4 compares two popular JS charting libraries Data Driven Documents (D3) and Google Charts:

Table 4: Data Driven Documents vs. Google Charts

	Google Charts	D3
Chart types	25	unlimited*
Customization	limited	unlimited*

(\* see description below)

Google Charts is compared to other JS Libraries easy to learn and offers fast and simple solutions to basic charting needs - in total about 25 chart types are supported. Customization of charts is also possible as Google Charts offers a lot of possibilities to adjust the chart to the web-pages layout, like changing colors, labels, axes, etc. (Google, 2015). For more complex visualizations and customization Google Charts however reaches its limits.

D3 on the other hand is not limited with a number of supported chart types and customizations. It *"is not a monolithic framework that seeks to provide every conceivable feature. Instead, D3 solves the crux of the problem: efficient manipulation of documents based on data. This avoids proprietary representation and affords extraordinary flexibility, exposing the full capabilities of web standards [...]"* (D3js, 2013). This makes D3 enormously powerful and flexible but at the same time much harder

to master than for example Google Charts. Figure 7 shows example visualizations made with Google Charts (a) and D3 (b).

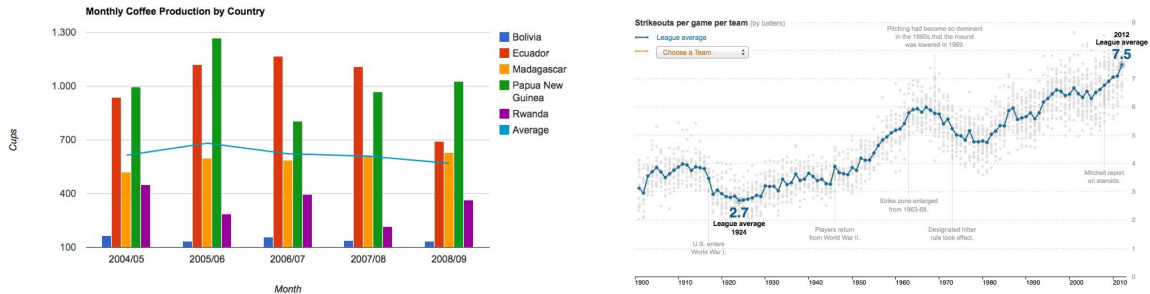


Figure 7: (a). Google Charts Combo Chart showing monthly coffee production by country (Google, 2015). (b). D3 Line Chart showing Baseball statistics on strike outs (NYT, 2012).

### 2.3.3.4.3 jQuery

jQuery / jQueryUI is a JS Library for handling day-to-day JS tasks and offers functions that make event-handling, effects, DOM element manipulation as well as the adding and managing of UI elements, etc. much easier. It *“has taken the frontend development world by storm. Its dead-simple syntax makes once-complicated tasks downright trivial - enjoyable, even.”* (Lindley, 2009).

What makes jQuery special is that it is used and adapted by many large companies like IBM, Microsoft, Dell which guarantees stability. It is optimized for all major web browsers (IE, Safari, Chrome, Firefox,...), lightweight with a minimal footprint and open source which means that anybody can contribute to bug fixes.

The difference between native JS and jQuery can be best shown on an example. The first code snippet below shows how to change the background color of a web-page with JS and the second code snippet shows the same for jQuery (Wilde, 2013):

#### JavaScript

```

1 function changeBgColor(color) {
2     document.body.style.background = color;
3 }
4 onload= changeBgColor('#ccc');
```

### jQuery

```
1 $('body').css('background', '#ccc');
```

This rather simple example is still able demonstrate how code complexity, especially for more sophisticated web applications, can be reduced with the help of jQuery.

## 2.4 Case studies

To be able to demonstrate how a Web-GIS application is set up, two case studies with different background in the field of water management, supported by real data, were selected. The next chapters describe both datasets that are the fundament for the further applied analysis of the previously mentioned technical components.

### 2.4.1 Marchfeld, Austria

The first case study has a regional scale and is set in the  $1000\text{km}^2$  wide Marchfeld region in Lower Austria, located in the north-east of the capital Vienna. The region is characterized by a semi-arid climate and the average annual precipitation ranges from 500 to 550mm with a minimum of 300mm (Vuolo et al., 2013, p. 5). Although this makes it the driest region in Austria, about 65000ha of this area are used for agricultural production that makes it one of the primary producers of agricultural goods in Austria. The main crops are vegetables (11%), sugar beet (10%) and potatoes (7%) (Neugebauer, 2013, p. 10).

The region is embedded into the so called Danube-March-Triangle, two main rivers that surround this region whose history is full of constant variations of both floods and drought periods. To prevent and minimize the impact of floods, the rivers were regulated in 1870 and to fight the risks of droughts, field irrigation with the use of groundwater was introduced in 1950. This however yielded in falling levels of the groundwater table of 3m in just 15 years, with the need for a sustainable solution of the regions irrigation management (Neudorfer et al., 2012, p.5).

This was the trigger for the so-called "Marchfeldkanal"-project which was implemented from 1986 until 1995 and established an approximately 100km long water network across the whole region consisting of newly build rivers and existing streams. These are fed with up to  $15\text{m}^3\text{s}^{-1}$  water from the Danube to support irrigation. Furthermore to renew the groundwater body, three artificial groundwater recharging stations were established with a water flow of approximately  $0.38\text{m}^3\text{s}^{-1}$  (Karl and Neudorfer, 2002). Figure 6 depicts the river network as well as the three groundwater recharging stations.



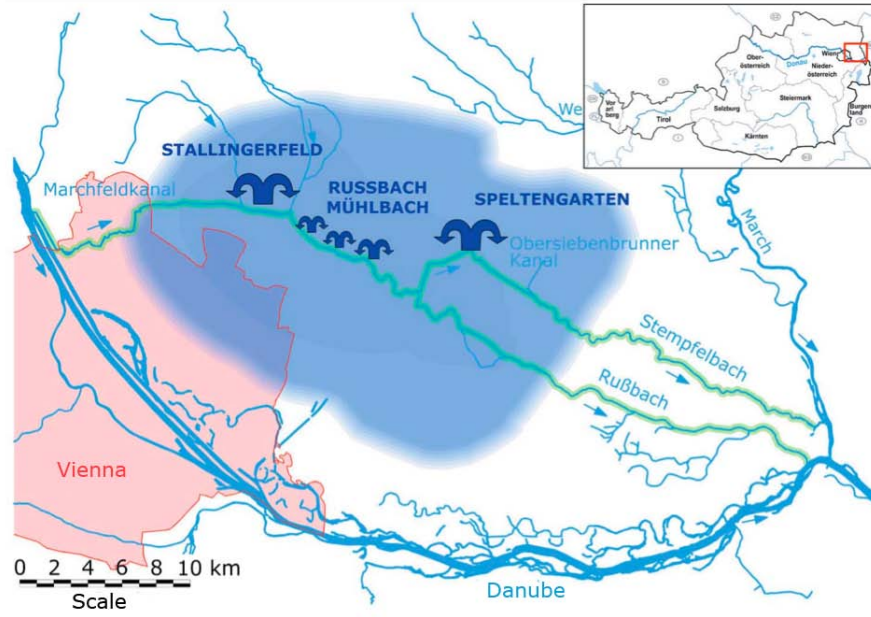


Figure 8: Overview on the Marchfeldregion showing the "Marchfeldkanal" and the three groundwater recharging stations Stallingerfeld, Russbach/Mühlbach, Speltengarten. Edited after (Karl and Neudorfer, 2002, p.583).

#### 2.4.1.1 Data

The data for this case study was provided by the Institute of Surveying, Remote Sensing and Land Information (IVFL) of the University of Natural Resources and Applied Life Sciences, Vienna and was generated during the project "*EO4Water - Application of Earth observation technologies for rural water management*". During this project satellite imagery from the DEIMOS-1 and Landsat 5 satellites was used to estimate the Crop Coefficient  $K_c$  to further calculate the evapotranspiration  $ET_p$  and crop water requirement  $CWR$ .

The data is divided into four indicators:

1. Evapotranspiration  $ET_p$
2. Crop Coefficient  $K_c$
3. Precipitation  $Precip$  (provided by the ZAMG via ground measurement stations)
4. Crop Water Requirement  $CWR$

**Evapotranspiration  $ET_p$** 

According to the FAO-56 paper evapotranspiration is *"the combination of two separate processes whereby water is lost on the one hand from the soil surface by evaporation and on the other hand from the crop by transpiration"* (Allen et al., 1998, p.5). To calculate the evapotranspiration the reference evapotranspiration  $ET_0$  and the crop coefficient  $K_c$  are needed:

$$ET_p = ET_0 * K_c \quad (1)$$

**Reference evapotranspiration  $ET_0$** 

The reference evapotranspiration is the rate of water that evapotranspires from a standard grass reference surface not short on water, and can be calculated using the *"FAO Penman-Monteith method"* that requires radiation, air temperature, air humidity and wind speed data, which can be collected via ground meteorological stations.

**Crop Coefficient  $K_c$** 

The crop coefficient is the ratio between the crop evapotranspiration  $ET_p$  and the reference evapotranspiration  $ET_0$ :

$$K_c = ET_p / ET_0 \quad (2)$$

It represents an integration of the effects of the four main characteristics that distinguish a crop from the reference grass surface: Crop height, Albedo, Canopy resistance and Evaporation from soil (Allen et al., 1998, see p.90).

A crop coefficient of 0.8 for example means that a crop is evapotranspiring 20% less amount of water than the reference grass surface does.

$K_c$  can be derived from pre-calculated tables or using a remote sensing approach according to (D'Urso et al., 2010).

A detailed description of the methodologies for the generation of maps of  $ET_p$  and crop water requirements ( $CWR$ ) was recently reported in a comparative study in different pilot areas (Vuolo et al., 2015).

### Crop Water Requirement $CWR$

The crop water requirement is defined as the amount of water that is needed to compensate the loss of evapotranspiration. It is the amount of water per crop needed to grow optimally:

$$CWR = ET_p - Precip \quad (3)$$

where  $Precip$  is the precipitation.

In the context of this EO project the  $K_c$  values were calculated by using the satellite-based approach proposed by (D’Urso, 2001). Here the Leaf Area Index (LAI) and albedo ( $\alpha$ ) are derived from the spectral reflectance of the acquired satellite imagery. The LAI can be derived through the Weighted Difference Vegetation Index (WDVI) and the albedo is simply the ratio of the reflected radiation of a surface to the incoming radiation. By setting a crop height  $h_c$  the crop coefficient can then be estimated via a polynomial function.

Due to the fact that the focus of this thesis is the analysis of the technical parts of web-applications, the author kept the explanation of the data processing and derivation short since it is not needed for the further comprehension. More details however can be found in the reference literature (Vuolo et al., 2015), (D’Urso et al., 2010).

### Spatial and temporal extent

The provided data have a temporal extent of one growing season, starting from **May 15th until October 1st 2013**. For every indicator the data consist of **mean daily** values per landcover type for every municipality in the Marchfeldregion. In total there are 81 municipalities and 10 landcover types each. This yields in 810 values per indicator a day. The analyzed landcover types are: Maize, Soya, Potatoes, Carrots, Peas, Grass, Winter Crops, Sugar beet, Cucurbitaceous and Other Crops.

#### 2.4.1.2 Stakeholder

Stakeholder who are in need of this data are on one side the operators of the "Marchfeldkanal": Measuring the crop water requirement and evapotranspiration on municipality- or field level provides a good means for monitoring of the agricultural water needs of the region.

On the other side is this information also useful to regional decision makers and authorities in charge of water rights: To be able to estimate how much water is being

allocated to households, agriculture and industry each year per region or district, information on crop water requirement and evapotranspiration is of relevance. For this reason the development of the Web-GIS application for this case study focuses on the usability for those two stakeholders.

### 2.4.2 Kenya, Africa

The second case study has a broader, national scale. Kenya is located at the Horn of Africa with an area of approximately  $584.000km^2$ . The country is both characterized through climatic and ecological extremes: The altitude varies from sea level up to  $5000m$  in the highlands and the mean annual precipitation ranges from  $< 250mm$  to  $> 2000m$ . Agriculture is very important in Kenya and contributes 26% to the Gross Domestic Product. About 75% of the Kenyan population depends on agricultural food and income, however only one third of the total land area is agriculturally productive. The other parts of the land are semi-arid to arid, and characterized by low, unreliable and poorly distributed rainfall (Orodho, 2006). Kenya has been plagued by recurrent droughts in the past decades and climate change studies further indicate a trend towards increasing climate variability in the Horn of Africa including Kenya, most likely resulting in more frequent drought events (Balint et al., 2013).

#### 2.4.2.1 Data

The data for this case study were also provided by the Institute of Surveying, Remote Sensing and Land Information (IVFL) of the University of Natural Resources and Applied Life Sciences, Vienna. The data was collected by analysing time series of MODIS NDVI during a project regarding Drought Monitoring and Anomaly Analysis. It is divided into three indicators:

1. Normalized Vegetation Index *NDVI*
2. Vegetation Condition Index *VCI*
3. Precipitation *Precip* (based on the TAMSAT product from the University Reading, UK)

#### Normalized Difference Vegetation Index *NDVI*

The *NDVI* is probably the most frequently used index in remote sensing and explains the vegetation density per pixel with the following relation:

$$NDVI = \frac{NIR - VIS}{NIR + VIS} \quad (4)$$

where NIR is the near-infrared energy and VIS is the amount of visible energy reflected by a surface.

Chlorophyll is a pigment in plant leaves and strongly absorbs visible light for use in photosynthesis. On the other hand, the cell structure of the leaves strongly reflects near-infrared light. The more leaves a plant has, the higher is the value of  $NDVI$  (NASA, 2015).

### **Vegetation Condition Index $VCI$**

The Vegetation Condition Index is based on the  $NDVI$  and is used in Vegetation Anomaly Analysis. Here the actual  $NDVI$  value is compared to historical maximal and minimal values of the time series. Higher values (50+) of  $VCI$  correspond to favourable vegetation conditions, lower values (0-30) indicate stressed vegetation.

$$VCI = \frac{NDVI - NDVI_{min}}{NDVI_{max} + NDVI_{min}} \quad (5)$$

where  $NDVI$  represents the actual value and  $NDVI_{max}$  and  $NDVI_{min}$  are the multiyear absolute maximum and its minimum (Kogan, 2001).

### **Spatial and temporal extent**

The provided time series starts in March 2001 and is updated once a month. For every indicator several statistics are available: mean, median and the quantiles (P05-P95). The values are mean daily values per landcover type for every county in Kenya. In total there are 47 municipalities and 4 landcover types each. The analysed landcover types are: pastoralists, agro-pastoralists, agrarians, national parks and forests.

#### **2.4.2.2 Stakeholder**

According to (Balint et al., 2013, p. 341) *"numerous researches and the general opinion in the region agree that there is a need for adequate and up-to-date information on the occurrence and severity of drought episodes, probability of their duration, and their possible impacts."* To establish a framework and a monitoring system with the aim of minimizing the risks of droughts, the National Drought Management Authority of Kenya (NDMA) was established with its missions statement *"to provide leadership and coordination of Kenyas effort in the management of drought risks and enhancing adaptation to climate change"* (NDMA, 2012).

A Web-GIS application that is able to provide up-to-date information on vegetation condition and precipitation might therefore be a valuable tool for this organization to make monitoring and visualization of information easier and accessible.

## 3 Results

### 3.1 Marchfeld, Austria

#### 3.1.1 Database set-up

The above described data are set up in a PostGIS database in three main tables. The first contains all aggregated data with daily values for each indicator, landcover type and region, the second with the region names and geometries, the third providing a key with the landcover names. All three can be seen in the following figures: Figure 9 shows the PostGIS table *agg\_data* that contains all data values and is divided into 7 columns:

1. **indicator:** Name of the indicator. (*CWR*, *Kc*, *ETp*, *precip*)
2. **zone\_id:** Id of the region, linked to the table *regions* [1,81].
3. **lc\_id:** Id of the landcover type, linked to the table *land\_cover* [1,10].
4. **date:** Date of the value in the format YYYY-MM-DD,
5. **period:** Indicating that the values are daily,
6. **statistic:** Either Mean or Std. (Standard Deviation)
7. **value:** Value

	indicator text	zone_id integer	lc_id integer	date date	period text	statistic text	value double precision
1	CWR	1	1	2013-05-15	daily	Mean	3.122
2	CWR	1	2	2013-05-15	daily	Mean	2.693
3	CWR	1	3	2013-05-15	daily	Mean	5.315
4	CWR	1	4	2013-05-15	daily	Mean	1.792
5	CWR	1	5	2013-05-15	daily	Mean	1.268
6	CWR	1	6	2013-05-15	daily	Mean	2.245
7	CWR	1	7	2013-05-15	daily	Mean	4.795
8	CWR	1	8	2013-05-15	daily	Mean	3.262
9	CWR	1	9	2013-05-15	daily	Mean	3.286
10	CWR	1	10	2013-05-15	daily	Mean	3.916
11	CWR	2	1	2013-05-15	daily	Mean	3.409
12	CWR	2	2	2013-05-15	daily	Mean	3.183
13	CWR	2	3	2013-05-15	daily	Mean	5.445
14	CWR	2	4	2013-05-15	daily	Mean	1.98
15	CWR	2	5	2013-05-15	daily	Mean	2.06
16	CWR	2	6	2013-05-15	daily	Mean	2.771
17	CWR	2	7	2013-05-15	daily	Mean	5.188
18	CWR	2	8	2013-05-15	daily	Mean	4.161
19	CWR	2	9	2013-05-15	daily	Mean	2.979
20	CWR	2	10	2013-05-15	daily	Mean	4.692

Figure 9: Table *agg\_data* from the PostGIS database containing all daily values (Marchfeld).

Figure 10 shows the table *regions* that is divided into 6 columns:

1. **gid**: Corresponds to the *zone\_id* from the table *agg\_data*. [1,81]
2. **NAME**: Official names of the municipalities.
3. **PG\_NAME**: District names that contain municipalities.
4. **GEM\_I**: Id for the districts.
5. **id**: Data entry id. [0,80]
6. **the\_geom**: Geometry of the municipalities.

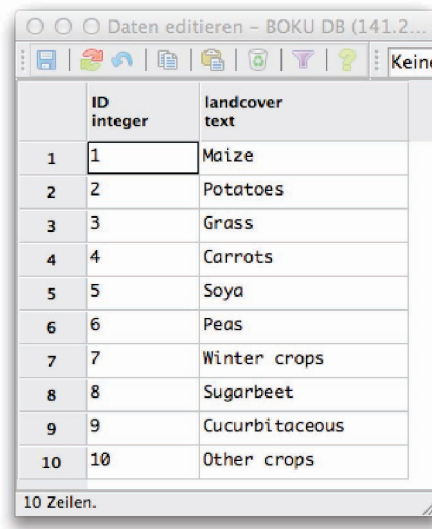


	gid [PK] integer	NAME character varying(33)	PG_NAME character varying(40)	GEM_I integer	id integer	the_geom geometry
1	1	Kleinharras	Matzen-Raggendorf	30838	0	0103000020797F0000010000003F000000
2	2	Stillfried	Angern an der March	30803	1	0103000020797F0000010000006F000000
3	3	Grub an der March	Angern an der March	30803	2	0103000020797F0000010000002C000000
4	4	Großschweinbarth	Groß-Schweinbarth	30824	3	0103000020797F00000100000056000000
5	5	Wolkersdorf	Wolkersdorf im Weinviertel	31655	4	0103000020797F00000100000095000000
6	6	Matzen	Matzen-Raggendorf	30838	5	0103000020797F00000100000054000000
7	7	Raggendorf	Matzen-Raggendorf	30838	6	0103000020797F0000010000002F000000
8	8	Riedenthal	Wolkersdorf im Weinviertel	31655	7	0103000020797F00000100000034000000
9	9	Ollersdorf	Angern an der March	30803	8	0103000020797F00000100000030000000
10	10	Prottes	Prottes	30848	9	0103000020797F00000100000041000000
11	11	Mannersdorf	Angern an der March	30803	10	0103000020797F00000100000038000000
12	12	Pföding	Wolkersdorf im Weinviertel	31655	11	0103000020797F00000100000028000000
13	13	Münichsthal	Wolkersdorf im Weinviertel	31655	12	0103000020797F00000100000028000000
14	14	Manhartsbrunn	Großbebersdorf	31614	13	0103000020797F00000100000024000000
15	15	Angern	Angern an der March	30803	14	0103000020797F00000100000028000000
16	16	Auersthal	Auersthal	30804	15	0103000020797F00000100000042000000
17	17	Reyersdorf	Schönkirchen-Reyersdorf	30852	16	0103000020797F0000010000002E000000
18	18	Obersdorf	Wolkersdorf im Weinviertel	31655	17	0103000020797F0000010000004C000000
19	19	Tallesbrunn	Weikendorf	30860	18	0103000020797F0000010000001C000000
20	20	Pillichsdorf	Pillichsdorf	31642	19	0103000020797F0000010000002D000000

Figure 10: Table *regions* from the PostGIS database containing region names and geometries (Marchfeld).

Figure 11 shows the table *landcover\_type* that is divided into 2 columns:

1. **ID:** Corresponds to the *lc\_id* from the table *agg\_data*. [1,81]
2. **landcover:** Landcover type.



	ID integer	landcover text
1	1	Maize
2	2	Potatoes
3	3	Grass
4	4	Carrots
5	5	Soya
6	6	Peas
7	7	Winter crops
8	8	Sugarbeet
9	9	Cucurbitaceous
10	10	Other crops

10 Zeilen.

Figure 11: Table *landcover\_type* from the PostGIS database containing the landcover types (Marchfeld).

### 3.1.2 Technical components of the final Web-GIS application

Using the technologies covered in the previous chapters a Web-GIS application was developed aiming at the mentioned stakeholders from (2.4.1.2). The final application was uploaded on a web-server and can be accessed via the following link:

[http://ivfl-geomap.boku.ac.at/demo\\_WG/eo4water/](http://ivfl-geomap.boku.ac.at/demo_WG/eo4water/)

In the upcoming chapter the author shows how these technologies work in "action" by giving code examples for each step of the data transformation from the raw database to the layouted web-map. Figure 12 depicts the final application as it can be seen in a web-browser by a user.

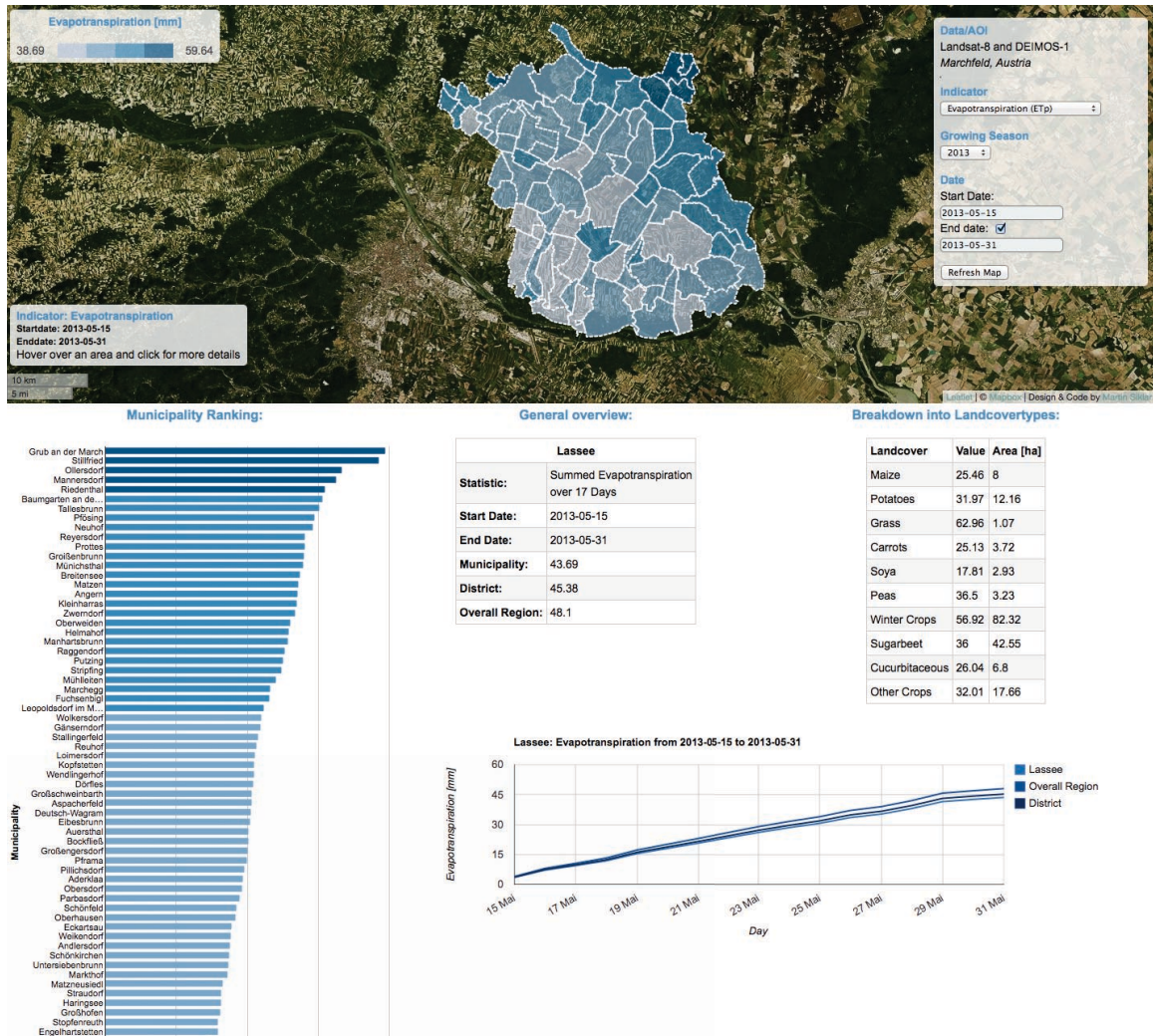


Figure 12: Final Web-GIS application for the first case study (Marchfeld).

The web-page can be divided into two parts: The upper part containing the map with an overlay of the municipalities borders, a menu for querying the data and some further information elements. The lower part is the analytical part, where data for a selected municipality can be displayed and analysed with the help of tables and graphs.

The technology used for the final map is the *JS Library Leaflet*, which offers all functionalities that were needed for the set-up of the application and has a much lighter footprint than the other previously introduced technology *OpenLayers*. The

following features were used to create the final map and are provided by Leaflet:

- Basemap
- Menu Overlay: For filtering and querying data.
- Thematic Overlay: Chloropleth map on municipality level including a legend.
- Map control: Zooming and panning.
- Interactivity and additional information elements: Customizeable pop-ups and mouse-events (onClick, hover).

### 3.1.2.1 Filtering of data - Query database

When a user enters the web-application he has the power to specify whatever indicator and period of time he wants to analyse by using the form in the menu. This is based on a combination of HTML for the basic input fields and jQueryUI for the date input. The jQuery UI datepicker plugin guarantees that the date input field is properly displayed in all web browsers.

When submitted, the filled out information is being sent to the server via the HTTP POST method and automatically stored in PHP variables with the following structure: `$_POST['inputname']`.

A PHP script then constructs two SQL queries:

- One *static* query that requests the region boundaries that will later be displayed on the map. The query uses two PostGIS functions: *ST\_AsGeoJson* which transforms the queried data into a *geoJSON* format that can later be read by Leaflet and the function *ST\_Transform* that transforms the polygons into *EPSG:4326*, which is the projection of the basemap.
- A second *dynamic* query matching the user criteria to request the actual data values. The example below shows the code for the menu as well as the constructed SQL queries.

## Menu

```

1 <form id="form" class="form" action="" method="POST">
2 <select name="indicator">
3     <option value="CWR">Crop Water Requirement (CWR)<br>
4     <option value="Kc">Crop Coefficient (Kc)<br>
5     <option value="ETp">Evapotranspiration (ETp)<br>
6     <option value="precip">Precipitation (precip)<br>
7 </select>
8 <h4>Date</h4>
9 Start Date:<br>
10 <input type="text" name="startdate" id="startdate" readonly="
    readonly" style="cursor:text"><br>
11 End date: <br>
12 <input id="enddate" name="enddate" id="enddate" type="text"
    readonly="readonly" style="cursor:text">
13
14 <script>
15 //jQuery DatePicker
16 $(function() {
17     $( "#startdate" ).datepicker({
18         dateFormat: 'yy-mm-dd',
19         minDate: MinDate,
20         maxDate: MaxDate,
21         defaultDate: MinDate,
22     });
23 });
24 $(function() {
25     $( "#enddate" ).datepicker({
26         dateFormat: 'yy-mm-dd',
27         minDate: MinDate,
28         maxDate: MaxDate,
29         defaultDate: MinDate,
30     });
31 });
32 </script>
33 </p>
34 <input type="submit" value="Refresh Map" name="submit">
35 </form>

```

### SQL Queries

```

1 //Dynamic query for data values
2 $data = $db->query("SELECT zone_id, SUM(value)
3                     FROM agg_data
4                     WHERE indicator = '" .$_POST['indicator'] . "'
5                     and statistic = 'Mean'
6                     and date >= '" .$_POST['startdate'] . "'
7                     and date <= '" .$_POST['enddate'] . "'")
8                     GROUP BY zone_id, lc_id
9                     ORDER by zone_id, lc_id");
10
11 //Static query for region boundaries
12 $regions = $db->query("SELECT * , public.ST_AsGeoJSON(public.
    ST_Transform((the_geom),4326),6) AS geojson FROM regions");

```

#### 3.1.2.2 Data processing

The result of the data query is a long multidimensional data array with a data entry for every land cover type and municipality summed over the specified range of dates. The resulting map however shows the aggregated sum for all landcover types within every municipality. Therefore the queried data has to be processed and transformed first:

Mean values for each municipality (not for every individual landcover type) and district have to be calculated and transformed to an array structure and format that is suitable for the upcoming visualization steps, where the region *id* will be used as an index to access the values in the data array.

This process is done on the server side with PHP, which is very powerful and fast when it comes to data manipulation. Once the data is processed, it is being sent back to the client side and handed over to JS. The following code shows the printed out data array structure before and after the data processing step as well as the geometry array in the *geoJSON*-format:

**Data array before processing:**

```

1 Array ( [0] => Array ( [zone_id] => 1 [0] => 1 [sum] => 6.377
    [1] => 6.377 )
2 [1] => Array ( [zone_id] => 1 [0] => 1 [sum] => 5.537 [1] =>
    5.537 )
3 [2] => Array ( [zone_id] => 1 [0] => 1 [sum] => 10.836 [1] =>
    10.836 )
4 [3] => Array ( [zone_id] => 1 [0] => 1 [sum] => 3.715 [1] =>
    3.715 )
5 [4] => Array ( [zone_id] => 1 [0] => 1 [sum] => 2.537 [1] =>
    2.537 )
6 [5] => Array ( [zone_id] => 1 [0] => 1 [sum] => 4.595 [1] =>
    4.595 )
7 [6] => Array ( [zone_id] => 1 [0] => 1 [sum] => 9.791 [1] =>
    9.791 )
8 [7] => Array ( [zone_id] => 1 [0] => 1 [sum] => 6.675 [1] =>
    6.675 )
9 [8] => Array ( [zone_id] => 1 [0] => 1 [sum] => 6.71 [1] =>
    6.71 )
10 [9] => Array ( [zone_id] => 1 [0] => 1 [sum] => 7.987 [1] =>
    7.987 )
11 [10] => Array ( [zone_id] => 2 [0] => 2 [sum] => 6.941 [1] =>
    6.941 )
12 [11] => Array ( [zone_id] => 2 [0] => 2 [sum] => 6.552 [1] =>
    6.552 )
13 ...)

```

Data array after processing:

```

1 //order corresponds to the id of the regions
2 Array ( [0] => 9.969 //sum for region 0
3 [1] => 10.060 //sum for region 1
4 [2] => 8.016
5 [3] => 8.245
6 [4] => 8.813
7 ...)
```

Geometry array as *geoJSON*:

```

1 var geometries = {"type":"FeatureCollection",
2 "features":[{"
3 "type":"Feature",
4 "geometry":{"
5 "type":"Polygon",
6 "coordinates"
7 :[[[16.668855,48.422831],[16.661963,48.426338],...]],
8 "properties":{"
9 "gid":1,
10 "NAME":"Kleinharras",
11 "PG_NAME":"Matzen-Raggendorf",
12 "GEM_I":30838,
13 "id":0}},
14 {"type":"Feature",
15 "geometry":{"
16 "type":"Polygon",
17 "coordinates"
18 :[[[16.855656,48.430434],[16.857849,48.427088],...]],
19 "properties":{"
20 "gid":2,
21 "NAME":"Stillfried",
22 "PG_NAME":"Angern an der March",
23 "GEM_I":30803,
24 "id":1}},...}]}
```

### 3.1.2.3 Visualization of the map

Leaflet was used for the visualization of the map: The basemap tile-layer is provided by Mapbox, a platform that offers several thematic basemap layers that can



be included into a Leaflet map after a registration. The code below shows how the basemap and the layer overlay are set up using Leaflet:

```

1 //define map extent and zoomlevel
2 var map = L.map('map', { zoomControl:false }).setView([48.27,
3     16.66], 10);
4 //add mapbox TileLayer
5 L.tileLayer('http://{s}.tiles.mapbox.com/v3/name.code/{z}/{x}
6    }/{y}.png').addTo(map);
7 //add geometry overlay
8 geojson = L.geoJson geometries, {
9     style: style
10 }).addTo(map);

```

The fill color for each polygon/municipality corresponds to the data value linked to it. The color is determined via a style function that accesses the data with the id of each municipality and returns a HEX color code upon the scheme below:

```

1 function style(feature)
2 {
3   return {
4     //call function that accesses values in dataArray using the
5     //region id
6     fillColor: getColor(dataArray[feature.properties.id]),
7     weight: 2,
8     opacity: 1,
9     color: 'white',
10    dashArray: '3',
11    fillOpacity: 0.7 };
12 //color scheme (fixed intervals)
13 function getColor(value) {
14   return
15   value >= 4 ? '#045a8d' :
16   value >= 3 ? '#2b8cbe' :
17   value >= 2 ? '#74a9cf' :
18   value >= 1 ? '#bdc9e1' :
19   '#f1eef6';
20 }

```

The resulting map with the tile-layer basemap and the coloured regions overlay is depicted in figure 13 below:

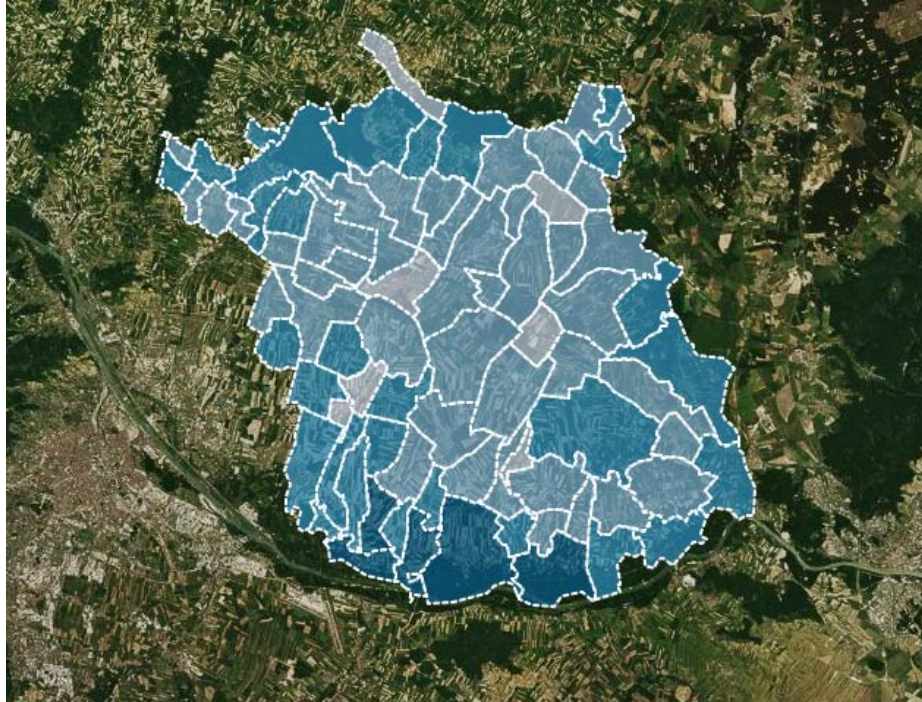


Figure 13: Final map display with tile-layer basemap and the coloured regions overlay (Marchfeld).

#### 3.1.2.4 User interaction and visualization of charts and tables

To enable interaction with the map and the user, an event listener has to be set up that defines the type of event and the functions that is triggered upon the event. This is done by adding the property *onEachFeature* to the layer overlay object:

```

1 //add onEachFeature function event to overlay object
2 geojson = L.geoJson(marchfeld, {
3     style: style,
4     onEachFeature: onEachFeature
5 }).addTo(map);
6
7 //define event and triggered functions
8 function onEachFeature(feature, layer) {
9     layer.on({click: feautureClick});
10 }

```

```

11
12 function featureClick(e) {
13   var layer = e.target;
14   updateTable(layer.feature.properties);
15   updateChart(layer.feature.properties);
16 }

```

Line 9 in the code block above, shows how a click event is being defined, which states that always when a user clicks on a polygon, a function is triggered that updates the table and the chart with the corresponding values for each municipality.

### Tables

The tables in the Web-GIS application are created with plain HTML and JS. Figure 14 shows an example table for the municipality "Untersiebenbrunn", displaying the selected statistic, the date, and the corresponding value for the selected municipality, district as well as the entire region.

Untersiebenbrunn	
<b>Statistic:</b>	Mean Crop Water Req.
<b>Date:</b>	2013-10-01
<b>Municipality:</b>	1.12
<b>District:</b>	1.04
<b>Overall Region:</b>	1.21

Figure 14: Table that shows the selected statistic, date and the computed values for the municipality, district and overall region.

The code below shows how the cells of the table are accessed and updated with the custom function *updateTable()* by making use of the DOM.

```

1 tableUpdate = function (id) {
2   document.getElementById("Name").innerHTML = '<b><center>' + id
   .NAME + '</center></b>';
3   document.getElementById("Municipality").innerHTML = dataArray[
   id];
4   document.getElementById("District").innerHTML = districtsArray
   [id];}

```

The cells are accessed using their DOM element ID, and updated via the *.innerHTML* property with the corresponding values for each municipality.

## Charts

Since only basic chart types (line chart and bar chart) were needed to visualize the data, and chart customizations weren't really necessary since the web-applications layout is neutral (i.e. with a standard white background), the Google Charts Library was the right way to go. The code snippet below shows how the basic syntax of a Google line chart looks like:

```

1 //input data
2 var data = google.visualization.arrayToDataTable(dataLineChart
3   );
4 //customize chart
5 var options = {
6   vAxis: {title: Indicator + ' ' + Units},
7   title: regionNames[props.id] + ': ' + Indicator + ' from '
8     + startDate + ' to ' + endDate,
9   hAxis: {title: "Day" ,
10     slantedText:true,
11     format: 'dd MMM'
12   },
13   colors: ['#2171b5', '#08519c', '#08306b'],
14   chartArea:{width:"70%",height:"50%"}
15 };
16 var chart = new google.visualization.LineChart(document.
17   getElementById('lineChart'));
18 //draw final chart
19 chart.draw(data, options);
20 }

```

In a first step the data is being converted to a Google-DataTable, if the input data structure doesn't meet certain specification (depending on the chart type) the rendered chart output will be blank, since the data couldn't be converted properly.

The second step is to set custom options, like labels for the axis, title, size, colors for the various lines.

The third step is to call the specific Google Chart function that creates the chart object. For creating a line chart this function is called *google.visualization.LineChart()*.

After this step, the chart can be drawn using the function `chart.draw()`. Figure 15 shows the final line chart layout as it can be seen in the Web-GIS application for the municipality "Obersiebenbrunn".

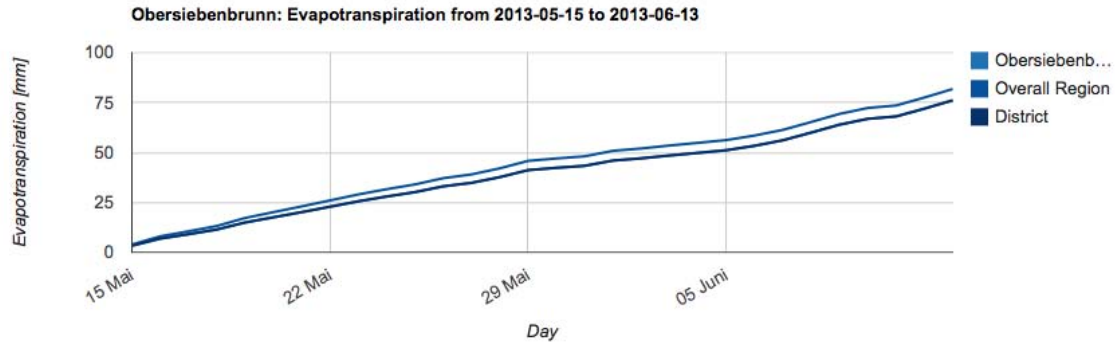


Figure 15: Google Line Chart displaying Evapotranspiration for the municipality "Obersiebenbrunn".

### 3.1.3 Use of application

After discussing the technical aspects of the Web-GIS application, this chapter focuses on what information can be generated from viewing the application.

Therefore selected indicators and parameters will be used to explain the applications output (map, tables, charts) and the meaning of displayed information

#### 3.1.3.1 Map - analysis of spatial trends and dependencies

The map is a good starting point to get a first impression if there is any spatial trend in the data. Due to possible local weather variability, precipitation is not equally distributed over the entire region, therefore it should be possible to see a spatial trend in the data.

An indicator that depends on precipitation is the crop water requirement, therefore it should depict a similar trend. Figure 16 shows both the *CWR* (left) and precipitation (right) for the June 2013.



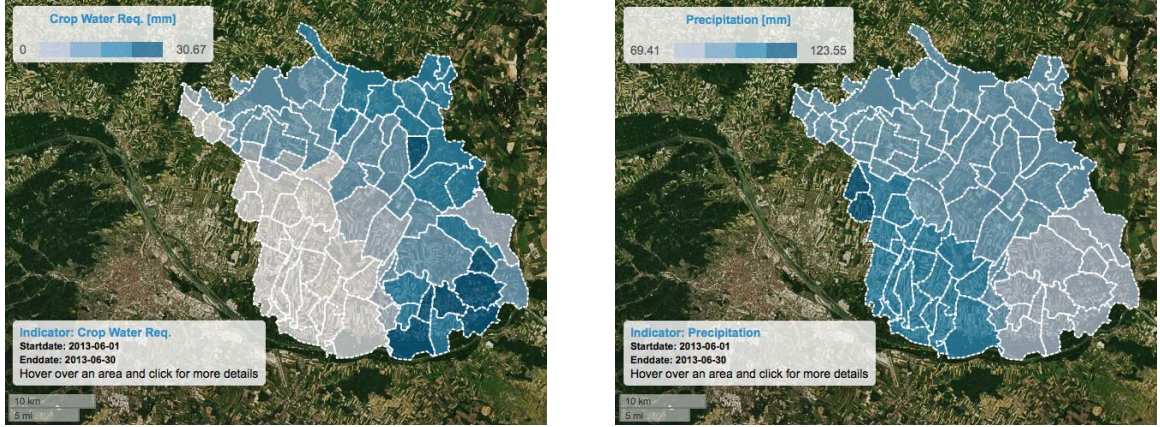


Figure 16: (a). Spatial trend of the crop water requirement (left) and precipitation (right) for June 2013 in the Marchfeldregion.

As can be seen in the figure, municipalities with more precipitation have a lower crop water requirement than municipalities with less precipitation. The map also clearly shows the spatial trend of the rainfall distribution for June 2013 (right). The dependence of evapotranspiration on the crop coefficient can also be seen by comparing them on a map - figure 17 depicts both indicators,  $ET_p$  (left) and  $K_c$  (right) for June 2013.

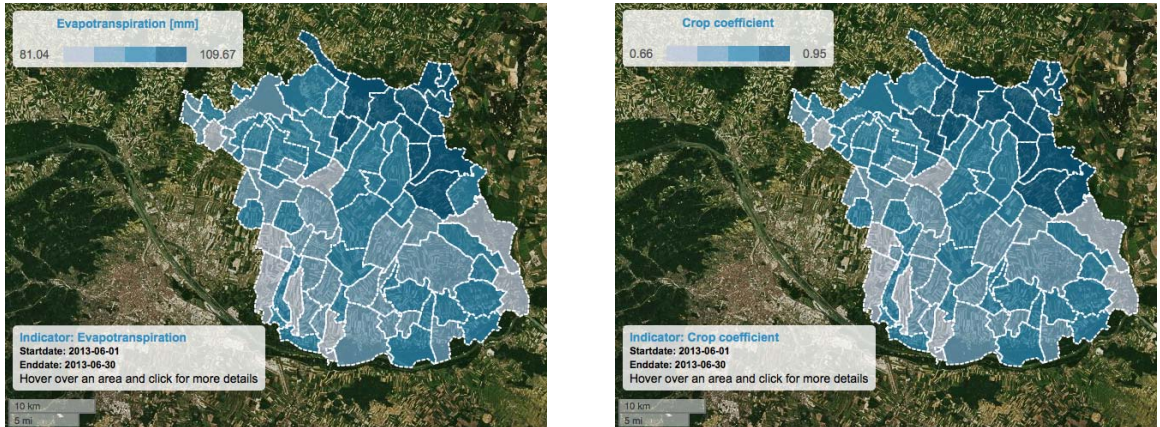


Figure 17: (a). Dependence of evapotranspiration (left) on crop coefficient (right) for June 2013 in the Marchfeldregion.

As can be seen in the figure above, the colouring of both maps is identical (note however the difference in the values), since the  $ETp$  is linearly dependent on the  $Kc$  (see equation 1).

### 3.1.3.2 Tables and Charts - aggregation over time and space

An aggregation of evapotranspiration and crop water requirement over time can yield in interesting insights. By comparing the aggregated values for each municipality a ranking can be established that depicts which municipality uses the most water i.e. within a given growing season. Figure 18 shows this ranking for aggregated evapotranspiration from May to October 2013.

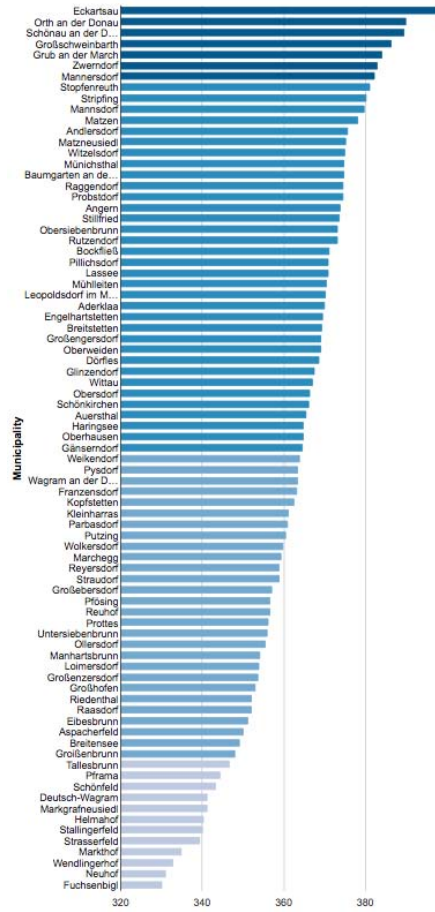


Figure 18: Ranking of municipalities after aggregated evapotranspiration from May to October 2013 in the Marchfeldregion.

Aggregated information on water use is very valuable for organizations, like i.e. the "Marchfeldkanal", that monitor the amount of water that is being used by each municipality to be able to ensure that enough water is available in periods of low rainfall and secure sustainable water distribution over time by not stressing the ground water table of the region.

Another important aspect of water supply is not only the total amount of water that is needed within a given period but also the timing of peak periods with the highest water requirements and evapotranspiration. Both indicators strongly depend on the crop development that is represented by the crop coefficient. By plotting the  $K_c$  over the entire growing season in a graph, information on the timing of peak crop water requirement and evapotranspiration can be generated, see figure 19.

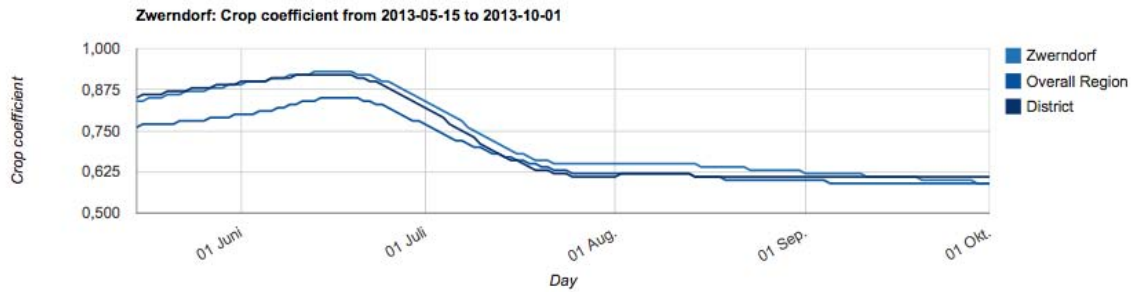


Figure 19: Crop coefficient for the entire growing season 2013 for the municipality "Zwerndorf", its district and the overall region.

As can be seen in the figure,  $K_c$  peaks in the selected municipality as well as in the entire region at the end of June and then constantly decreases. This means that in this period evapotranspiration also reaches its maximum and water supply has to be planned accordingly.

An aggregation over space, for example over a number of municipalities, districts or even over the entire region can also yield valuable information for authorities in charge of water rights. Figure 20 depicts a table that compares the evapotranspiration for the entire growing season of a selected municipality, to its district and the region.



Deutsch-Wagram	
<b>Statistic:</b>	Summed Evapotranspiration over 140 Days
<b>Start Date:</b>	2013-05-15
<b>End Date:</b>	2013-10-01
<b>Municipality:</b>	341.39
<b>District:</b>	340.62
<b>Overall Region:</b>	363.1

*Figure 20:* Comparison of aggregated evapotranspiration on municipality-, district- and regional level for "Deutsch-Wagram".

By collecting this information over longer time periods (i.e. multiple years) a benchmark on water use on different spatial levels can be established and can be used as a decision support for future water rights distribution on various levels as well as a reference for water use monitoring.

## 3.2 Kenya, Africa

### 3.2.1 Database set-up

The data for this case study is also set up in a PostGIS database. Four main tables are used: the first contains all aggregated data with monthly values for each indicator, landzone and county. The second table provides the landzone names and ids, the third contains the county names, county ids and geometries, and a fourth table provides a key that links county- and landzone ids to each other. All four can be seen in the following figures:

Figure 21 shows the PostGIS table *agg\_data* that contains all data values and is divided into 7 columns:

1. **sensor:** The satellite that was used to acquire the image.
2. **indicator:** Name of the indicator (*NDVI*, *VCI*, *RAIN*)
3. **zone\_id:** Id that indicates county and landzone of the value. The key to this Id is explained in the table *zones*.
4. **date:** Date of the value in the format YYYY-MM-DD
5. **period:** Indicating that the values are monthly.
6. **statistic:** Mean, Median or Quantiles.
7. **value:** Value

	sensor [PK] text	indicator [PK] text	zone_id [PK] integer	date [PK] date	period [PK] text	statistic [PK] text	value double precis
1	MODIS	NDVI_ABS	10	2001-03-01	month	mean	0.42
2	MODIS	NDVI_ABS	10	2001-03-01	month	P05	0.25
3	MODIS	NDVI_ABS	10	2001-03-01	month	P10	0.28
4	MODIS	NDVI_ABS	10	2001-03-01	month	P30	0.34
5	MODIS	NDVI_ABS	10	2001-03-01	month	P50	0.4
6	MODIS	NDVI_ABS	10	2001-03-01	month	P70	0.45
7	MODIS	NDVI_ABS	10	2001-03-01	month	P90	0.59
8	MODIS	NDVI_ABS	10	2001-03-01	month	P95	0.66
9	MODIS	NDVI_ABS	10	2001-04-01	month	mean	0.5
10	MODIS	NDVI_ABS	10	2001-04-01	month	P05	0.29
11	MODIS	NDVI_ABS	10	2001-04-01	month	P10	0.34
12	MODIS	NDVI_ABS	10	2001-04-01	month	P30	0.43
13	MODIS	NDVI_ABS	10	2001-04-01	month	P50	0.5
14	MODIS	NDVI_ABS	10	2001-04-01	month	P70	0.56
15	MODIS	NDVI_ABS	10	2001-04-01	month	P90	0.66
16	MODIS	NDVI_ABS	10	2001-04-01	month	P95	0.73
17	MODIS	NDVI_ABS	10	2001-05-01	month	mean	0.57

Figure 21: Table *agg\_data* from the PostGIS database containing all monthly values (Kenya).

Figure 22 shows the PostGIS table *zone\_names* that contains the name of each landzone and is divided into two columns:

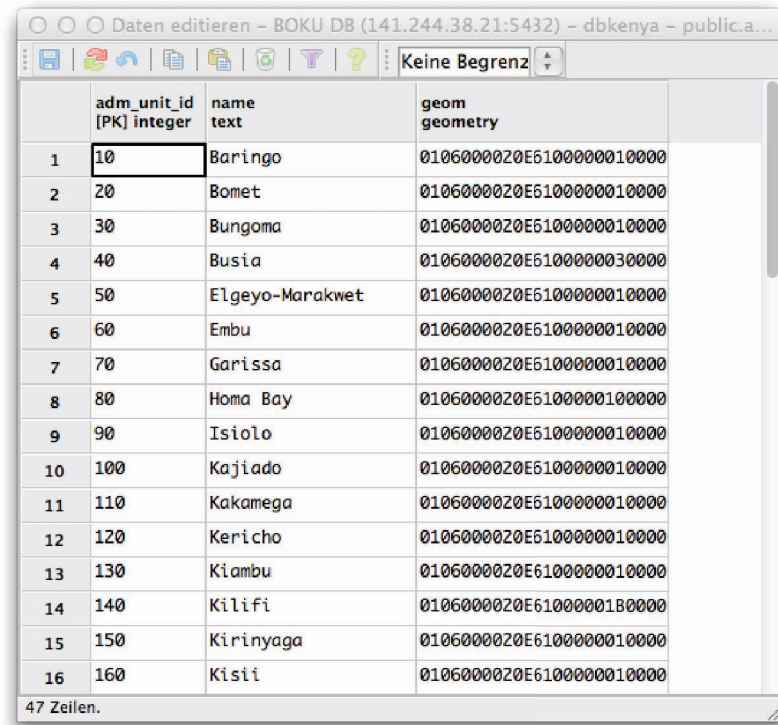
1. **lz\_id:** Landzone ID.
2. **description:** Corresponding name of the landzone ID.

	lz_id [PK] integer	description text
1	0	all
2	1	pastoralists
3	2	agro-pastoralists
4	3	agrarians
5	4	nat park & forest
*		

Figure 22: Table *land\_zones* from the PostGIS database containing the names of the landzones (Kenya).

Figure 23 shows the PostGIS table *adm\_units* that contains the geometries of the counties and is divided into three columns:

1. **adm\_unit\_id:** County ID.
2. **name:** Corresponding county name.
3. **geom:** Geometry of each county.



	adm_unit_id [PK] integer	name text	geom geometry
1	10	Baringo	0106000020E6100000010000
2	20	Bomet	0106000020E6100000010000
3	30	Bungoma	0106000020E6100000010000
4	40	Busia	0106000020E6100000030000
5	50	Elgeyo-Marakwet	0106000020E6100000010000
6	60	Embu	0106000020E6100000010000
7	70	Garissa	0106000020E6100000010000
8	80	Homa Bay	0106000020E6100000010000
9	90	Isiolo	0106000020E6100000010000
10	100	Kajiado	0106000020E6100000010000
11	110	Kakamega	0106000020E6100000010000
12	120	Kericho	0106000020E6100000010000
13	130	Kiambu	0106000020E6100000010000
14	140	Kilifi	0106000020E61000001B0000
15	150	Kirinyaga	0106000020E6100000010000
16	160	Kisii	0106000020E6100000010000

47 Zeilen.

Figure 23: Table *adm\_units* from the PostGIS database containing the names and geometries of the counties (Kenya).

Figure 24 shows the PostGIS table *zones*. This table shows how the *zone\_id*, that is used to identify to what county and lanzone each value belongs, is constructed using the *lz\_id* and the *adm\_unit\_id*.

	zone_id [PK] integer	adm_unit_id integer	lz_id integer
41	410	410	0
42	420	420	0
43	430	430	0
44	440	440	0
45	450	450	0
46	460	460	0
47	470	470	0
48	1010	10	1
49	1020	20	1
50	1030	30	1
51	1040	40	1
52	1050	50	1
53	1060	60	1
54	1070	70	1
55	1080	80	1
56	1090	90	1
57	1100	100	1
58	1110	110	1

235 Zeilen.

Figure 24: Table *zones* from the PostGIS database showing the link between *adm\_unit\_id* and *lz\_id* (Kenya).

### 3.2.2 Technical components of the final Web-GIS application

As can be seen in the previous chapter, is the database-set of both case studies very similar. Hence the technologies used to create the Web-GIS application for the Kenya case study also don't differ from the ones used to build the Marchfeld case study: The JS mapping library Leaflet was used to visualize the map as well as the overlay elements and Google Charts was used to create data visualizations in the form of graphs and charts.

The set-up process of both case studies is similar so that the author refers the reader to chapter 3.1.2 of the previous case study where the technical components used for the set-up were already presented. Figure 25 shows the final Web-GIS application

for Kenya that will be discussed in detail in the next chapter. It is divided into a map window and an analytical part that displays two types of charts: a "multi-year-seasonality-chart" and a "time series" chart. The final application was uploaded on a web-server and can be accessed via the following link:  
[http://ivfl-geomap.boku.ac.at/demo\\_WG/kenya/](http://ivfl-geomap.boku.ac.at/demo_WG/kenya/).

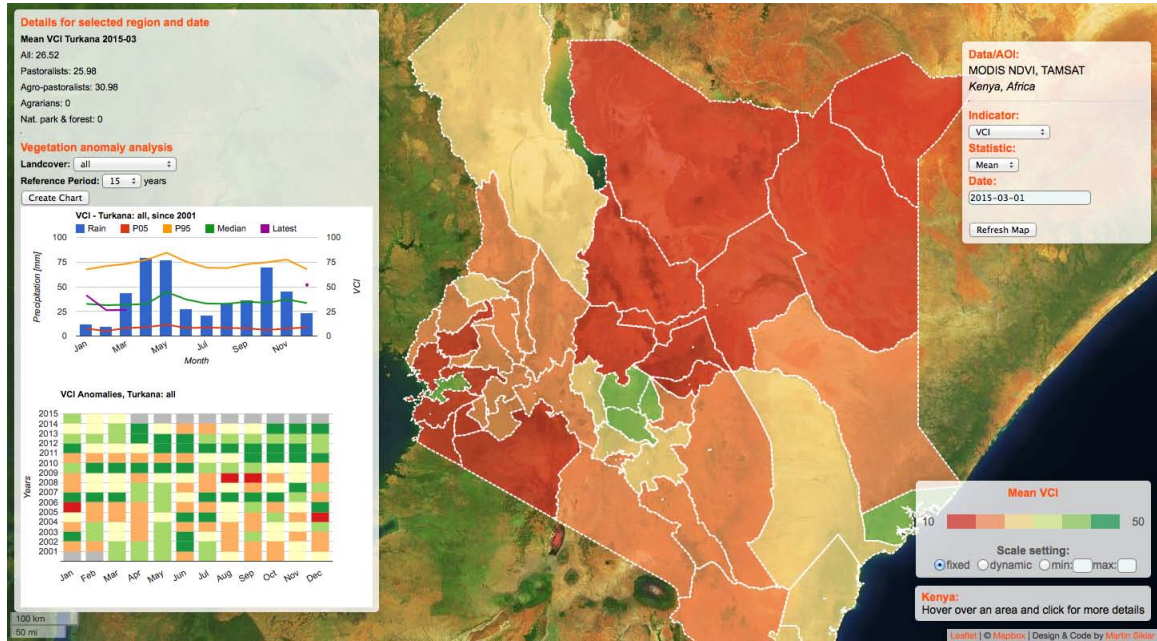


Figure 25: Final Web-GIS application for the second case study (Kenya).

### 3.2.3 Use of application

The main purpose of this application is the visualization and communication of detected anomalies of vegetation and precipitation to minimize impacts of droughts in Kenya.

The application provides three ways on how anomalies can be analysed: A spatial visualization with a map, a "time series" chart with the latest values and historical minimum and maximum values as a benchmark and a "multi-year-seasonality-chart" visualization.

#### 3.2.3.1 Map

The map can for example be used to visualize the distribution of vegetation density



(*NDVI*) and precipitation for a specific point in time in Kenya. Figure 26 shows two maps that depict the vegetation density (left) and the precipitation (right) for March 2015.

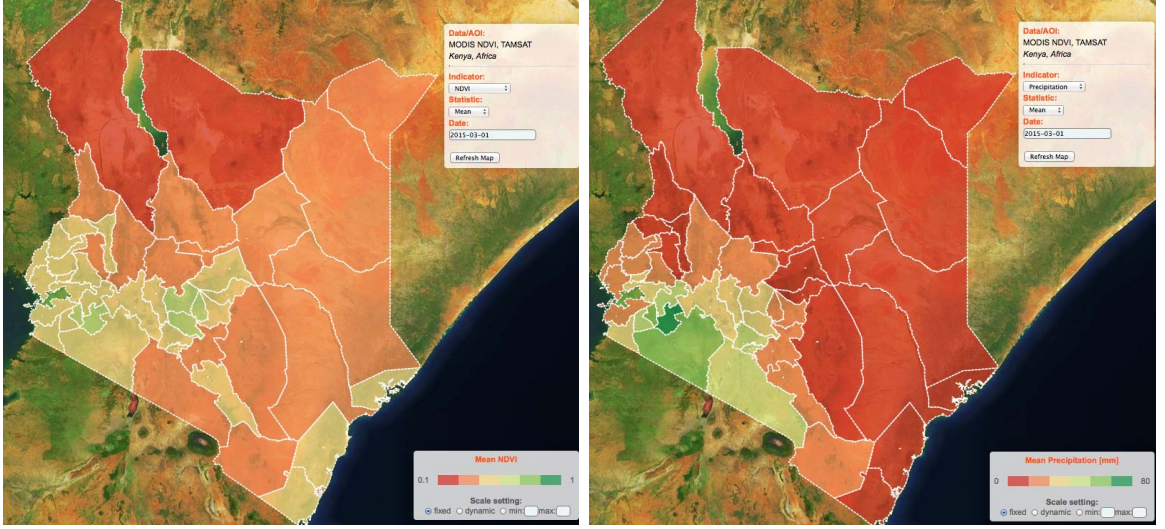


Figure 26: (a) *NDVI* and (b) precipitation for March 2015 in Kenya, Africa.

The maps give a first good overview on vegetation and precipitation in Kenya for a given month and provide information on the countries climatic conditions. Red colors indicate low precipitation and *NDVI*, green colors represent counties with high *NDVI* and precipitation.

It is however impossible to tell from a single point in time whether the observed value of *NDVI* or precipitation constitutes an anomaly, the *VCI* is instead used to display anomalies (see eq. 5).

### 3.2.3.2 Time Series Chart

The time series chart allows the user to select any county, landzone and indicator of his/her choice and puts the corresponding data into the context of historical values by a graphical comparison of the actual value to the historical p05, p50 and p95 quantiles over the whole year. Figure 27 shows an example graph depicting *NDVI* for the county "Nakuru" for all land cover types.



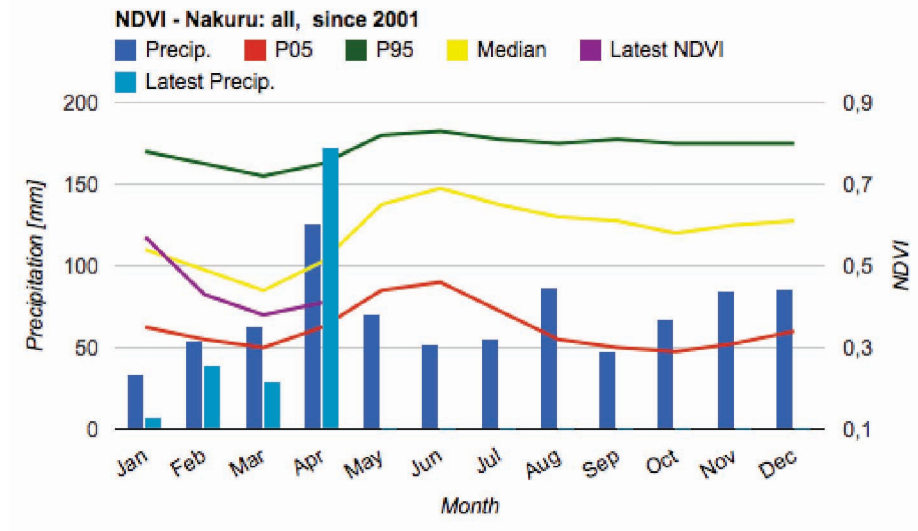


Figure 27: NDVI for the county Nakuru for all landzones.

The graph is updated each month with real-time data and supports the user (i.e. the NDMA) with valuable information: Whenever the purple line (representing the latest values) is getting closer or even crosses the red line (p05 quantile) a negative anomaly can be detected. If this anomaly could for example result in a drought or shortage of harvest, the NDMA would then be able to undertake further action: i.e. to contact the authorities of the affected county in time and put mechanisms in place that cope and minimize the effects of a potential drought.

### 3.2.3.3 Multi-year-seasonality-chart

So far, only maps and charts showing precipitation and *NDVI* were presented. The heatmap chart however is a chart type that only supplies useful information in combination with indicators like the *VCI*. Figure 28 depicts such a chart showing *VCI* for the county "Narok" and the landzone type agro-pastoralists.

It has the years of all observations on the y-axis and the months on the x-axis and can be seen as a type of "calendar chart" that shows the value of *VCI* in each cell with a color fill upon the following scheme:



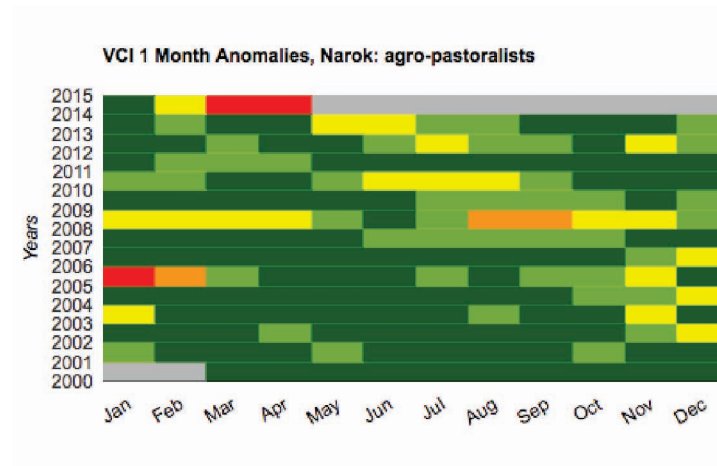


Figure 28: VCI multi-year-seasonality-chart for agro-pastoralists in "Narok", Kenya.

This chart is a powerful tool that doesn't only supply the viewer with information on current anomalies but also shows historical anomalies at one glance in a user friendly and intuitive way.

## 4 Discussion and Conclusions

### 4.1 Advantages and challenges of web-mapping

The results show that the World Wide Web offers a variety of options on how to publish spatial data, respectively satellite based indicators on the internet.

As stated by (van Elzakker, 2000) maintain web-based mapping applications the advantage of being accessible, actual and comprehensible to its users at any time. In addition to that they allow an interaction with the user, which gives him/her the power to filter and query data to only display useful information the user really needs. This feature becomes even more important in the context of a increasingly digitalized world and *big data*.

Everything comes with a price, which means there are also some challenges and obstacles that have to be kept in mind while using free and open source web-mapping technologies:

- **Compatibility:** The IT world, especially the world wide web are fast changing environments: Hardware technologies, operation systems evolve and allow more and more features. This however also changes web browsers, web standards and how code this being interpreted and rendered by these web browsers. What has to be kept in mind is that not everybody uses the latest technologies and updates available: Some people prefer the IE to Firefox or Chrome, some user have old browser version installed, which may yield in rendering errors of HTML or JS code and some users might even have JS turned off entirely while they browse the web.

This problem has intensified with the advent of hand-held applications like smartphones and tablets. Entirely new operations systems like android and iOS, as well as much smaller screen sizes of these devices alter the layout of the viewed web application massively if they are not optimized by the developer.

These are issues web developers have to face while developing web-map applications that requires a lot of time-consuming testing and optimization across various platforms and devices.

- **Stability:** Not only hard- and software evolves but also the code itself. This fact becomes more important if external libraries like Google Charts are included into the web application. These libraries change over time, some functions may contain bugs that are being fixed, features may be added and removed, etc. This sudden change in code may result in parts of the web application not being displayed correctly as intended by the developer or errors and warnings because used functions were removed or altered entirely.

- **Performance:** Another challenge in web development is the performance and loading times of web applications. Although Internet connection speed increased in the last years in general, not everybody uses the fastest connection available to browse the web. Studies show that the tolerable waiting time for a page to load is app. 2 seconds (Nah, 2004). When designing a Web-GIS application this fact should be kept in mind.

Loading times can for example be minimized with a database set-up, that already contains the final information the user wants to query so data processing steps which slow down the rendering process are kept to a minimum. Especially when working with spatial data, i.e. polygons, a simplification and reduction of vertices might also result in faster loading times.

## 4.2 Lessons learned and outlook

Open source web-based mapping technologies are powerful and enable the sharing of spatial data, basically without any visualization restraint. The set-up process of an efficient and modern application however requires knowledge on web technologies, various scripting languages like PHP, JS and database structures like PostGIS. The process of getting acquainted with these technologies can be very time consuming and challenging especially for conventional cartographers and GIS specialists without any background knowledge. Various free and open-source mapping and charting libraries like Leaflet and Openlayers however have made this process easier for beginners by supplying developers with well-documented APIs and tutorials for creating web-maps. However, at the same time new web-mapping technologies like CartoDB and MapBox are stepping onto the stage, that are not open source and only limited free (i.e. 50,000 map views/month) but supply developers with an simple and intuitive user interface for the set-up of complex web-mapping applications.

Concluding, the domain of web-mapping is growing and many new companies like foursquare, pinterest, etc. run their businesses nearly exclusively through web-maps. It can be said that web-mapping is a dynamic field with a lot of potential for the future and still open for new technologies and developments to come. From this standing point however it is hard to foretell in which direction the next leap will go.

## References

- Agafonkin, V. (2014). *LeafletJS - Choropleth Map*. <http://leafletjs.com/examples/choropleth.html>. Accessed: 2015-02-23.
- Allen, R. G., Pereira, L. S., Raes, D., and Smith, M. (1998). FAO irrigation and drainage paper No. 56. *Rome: Food and Agriculture Organization of the United Nations*, pages 26–40.
- Bael, V. (2014). *TCP - Transfer Control Protocol*. <http://www.webopedia.com/TERM/T/TCP.html>. Accessed: 2015-02-23.
- Balint, Z., Mutua, F., Muchiri, P., and Omuto, C. T. (2013). *Monitoring Drought with the Combined Drought Index in Kenya*. University of Nairobi.
- Buchanan, W. (2002). *The Complete Handbook of the Internet*, volume 1. Kluwer Academic Publishers, Boston.
- Chantelau, K. and Brothuhn, R. (2010). *Multimediale Client-Server-Systeme*. Springer-Verlag Berlin Heidelberg.
- D3js (2013). *Data Driven Documents - Introduction*. <http://d3js.org/>. Accessed: 2015-03-26.
- Detwiler, J. (2014). *GIS Mashups for Geospatial Professionals. Types of Web Maps*. [https://www.e-education.psu.edu/geog863/resources/l3\\_p4.html](https://www.e-education.psu.edu/geog863/resources/l3_p4.html). The Pennsylvania State University, Accessed: 2015-02-23.
- D’Urso, G. (2001). *Simulation and management of on-demand irrigation systems: a combined agrohydrological and remote sensing approach*. Wageningen Universiteit.
- D’Urso, G., Richter, K., Calera, A., Osann, M., Escadafal, R., Garatuza-Pajan, J., Hanich, L., Perdigo, A., Tapia, J., and Vuolo, F. (2010). Earth Observation products for operational irrigation management in the context of the {PLEIADeS} project. *Agricultural Water Management*, 98(2):271 – 282.
- Flanagan, D. (2002). *JavaScript: the definitive guide*. ” O’Reilly Media, Inc.”.
- Goodman, D. (2002). *Dynamic HTML: The Definitive Reference: A Comprehensive Resource for HTML, CSS, DOM & JavaScript*. ” O’Reilly Media, Inc.”.
- Google (2015). *Googel Developers - Using Google Charts*. <https://developers.google.com/chart/interactive/docs/index>. Accessed: 2015-03-26.

- InternetWorldStats (2014). *World Internet usage and population statistics. June 30, 2014 - Mid Year Update*. <http://www.internetworldstats.com/stats.htm>. Accessed: 2015-02-23.
- John, M. (2008). *Hierarchy of objects in an example HTML DOM Document Object Model*. [http://en.wikipedia.org/wiki/Document\\_Object\\_Model#/media/File:JKDOM.SVG](http://en.wikipedia.org/wiki/Document_Object_Model#/media/File:JKDOM.SVG). Accessed: 2015-03-25.
- Karl, S. and Neudorfer, W. (2002). Das Marchfeldkanalsystem in Österreich. *Vermessung, Photogrammetrie, Kulturtechnik*, pages 582–585.
- Kogan, F. N. (2001). Operational space technology for global vegetation assessment. *Bulletin of the American Meteorological Society*, 82(9):1949–1964.
- Kraak, M.-J. (2001). Settings and needs for web cartography. *Web cartography: Developments and prospects*, pages 1–7.
- Lindley, C. (2009). *jQuery Cookbook: Solutions & Examples for jQuery Developers*. " O'Reilly Media, Inc."
- McFarland, D. S. (2011). *Javascript & jQuery: the missing manual*. " O'Reilly Media, Inc."
- Meyer, E. A. (2004). *Cascading style Sheets: The definitive guide*. " O'Reilly Media, Inc."
- Nah, F. F.-H. (2004). A study on tolerable waiting time: how long are Web users willing to wait? *Behaviour & Information Technology*, 23(3):153–163.
- NASA (2015). *Measuring Vegetation NDVI EVI*. [http://earthobservatory.nasa.gov/Features/MeasuringVegetation/measuring\\_vegetation\\_2.php](http://earthobservatory.nasa.gov/Features/MeasuringVegetation/measuring_vegetation_2.php). Accessed: 2015-03-16.
- NDMA (2012). *National Drought Management Authority, Vision*. [http://www.ndma.go.ke/index.php?option=com\\_content&view=article&id=25:vision&catid=5:who-we-are&Itemid=20](http://www.ndma.go.ke/index.php?option=com_content&view=article&id=25:vision&catid=5:who-we-are&Itemid=20). Accessed: 2015-03-16.
- Neudorfer, W., Weyermayer, H., Karl, S., and Mötz, M. (2012). *20 Jahre Marchfeldkanal*. Betriebsgesellschaft Marchfeldkanal. G.G, Buchbinderrei GeshmbH: 2020 Hollabrunn.

- Neugebauer, N. (2013). Estimation of crop water requirements based on remote sensing data - A case study of the Marchfeld region in 2010. Master's thesis, University of Natural Resources and Applied Life Sciences, Vienna.
- NYT (2012). *Baseball - Strikeouts on the Rise*. <http://www.nytimes.com/interactive/2013/03/29/sports/baseball/Strikeouts-Are-Still-Soaring.html?ref=baseball&r=0>. Accessed: 2015-03-26.
- Openhub.net (2015). *Compare Projects: Leaflet - OpenLayers*. [https://www.openhub.net/p/compare?project\\_0=OpenLayers&project\\_1=Leaflet](https://www.openhub.net/p/compare?project_0=OpenLayers&project_1=Leaflet). Accessed: 2015-03-26.
- Orodho, A. B. (2006). Kenya country pasture/forage resource profiles. *FAO*.
- OSM (2013). *Tiles*. <http://wiki.openstreetmap.org/wiki/Tiles>. Accessed: 2015-02-23.
- PacificInstitute (2015). *Californian Urban Water Suppliers Water Use Map*. <http://www2.pacinst.org/gpcd/map.html>. Accessed: 2015-05-29.
- PHPnet (2015). *What can PHP do?* <http://php.net/manual/en/intro-whatcando.php>. Accessed: 2015-03-23.
- PostGIS.net (2015a). *PostGIS Features*. <http://postgis.net/features>. Accessed: 2015-03-23.
- PostGIS.net (2015b). *PostGIS Introduction - Spatial Data Types*. <http://workshops.boundlessgeo.com/postgis-intro/introduction.html>. Accessed: 2015-03-24.
- Quinn, S. (2014). *Open Web Mapping. Why tiled maps?* <https://www.e-education.psu.edu/geog585/node/706>. The Pennsylvania State University, Accessed: 2015-02-23.
- Vaidyanathan, R. (2014). *Visualizing Bike Sharing Networks*. <http://ramnathv.github.io/bikeshare/>. Accessed: 2015-02-23.
- van Elzakker, C. P. (2000). Use and users of maps on the Web. *Cartographic Perspectives*, pages 34–50.
- VECEA (2015). *Vegetation map for eastern Africa*. [http://maps.vegetationmap4africa.org/ea\\_pnv.html](http://maps.vegetationmap4africa.org/ea_pnv.html). Accessed: 2015-05-29.

- Vuolo, F., D'Urso, G., Michele, C. D., Bianchi, B., and Cutting, M. (2015). Satellite-based irrigation advisory services: A common tool for different experiences from Europe to Australia. *Agricultural Water Management*, 147(0):82 – 95. Agricultural Water Management: Priorities and Challenges.
- Vuolo, F., Neugebauer, N., Bolognesi, S. F., Atzberger, C., and D'Urso, G. (2013). Estimation of leaf area index using DEIMOS-1 data: Application and transferability of a semi-empirical relationship between two agricultural areas. *Remote Sensing*, 5(3):1274–1291.
- W3C (2015a). *Document Object Model (DOM)*. <http://www.w3.org/DOM>. Accessed: 2015-03-25.
- W3C (2015b). *HTML- Background*. <https://docs.webplatform.org/wiki/html>. Accessed: 2015-03-24.
- W3C (2015c). *What is CSS?* <http://www.w3.org/Style/CSS/Overview.en.html>. Accessed: 2015-03-24.
- Wilde, B. (2013). *jQuery vs. JavaScript: Whats the Difference Anyway?* <https://blog.udemy.com/jquery-vs-javascript/>. Accessed: 2015-03-28.