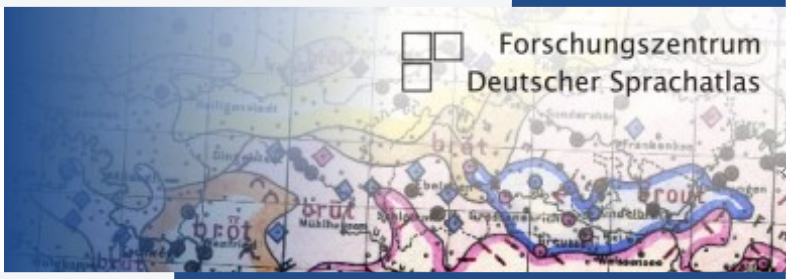
Akademie der Wissenschaften und der Literatur - Mainz

Philipps Universität Marburg

Robert Engsterhold, Jeffrey Pheiff, Tillmann Pistor

07.07.17

Das REDE SprachGIS – Technische Umsetzung und Herausforderungen



Übersicht

- Das REDE Projekt
- Der Aufbau des REDE SprachGIS
  - Frontend
  - Backend
- Funktionsweise: Datenvisualisierung
- Funktionsweise: Kartenexport
- Integration von Sprachkarten

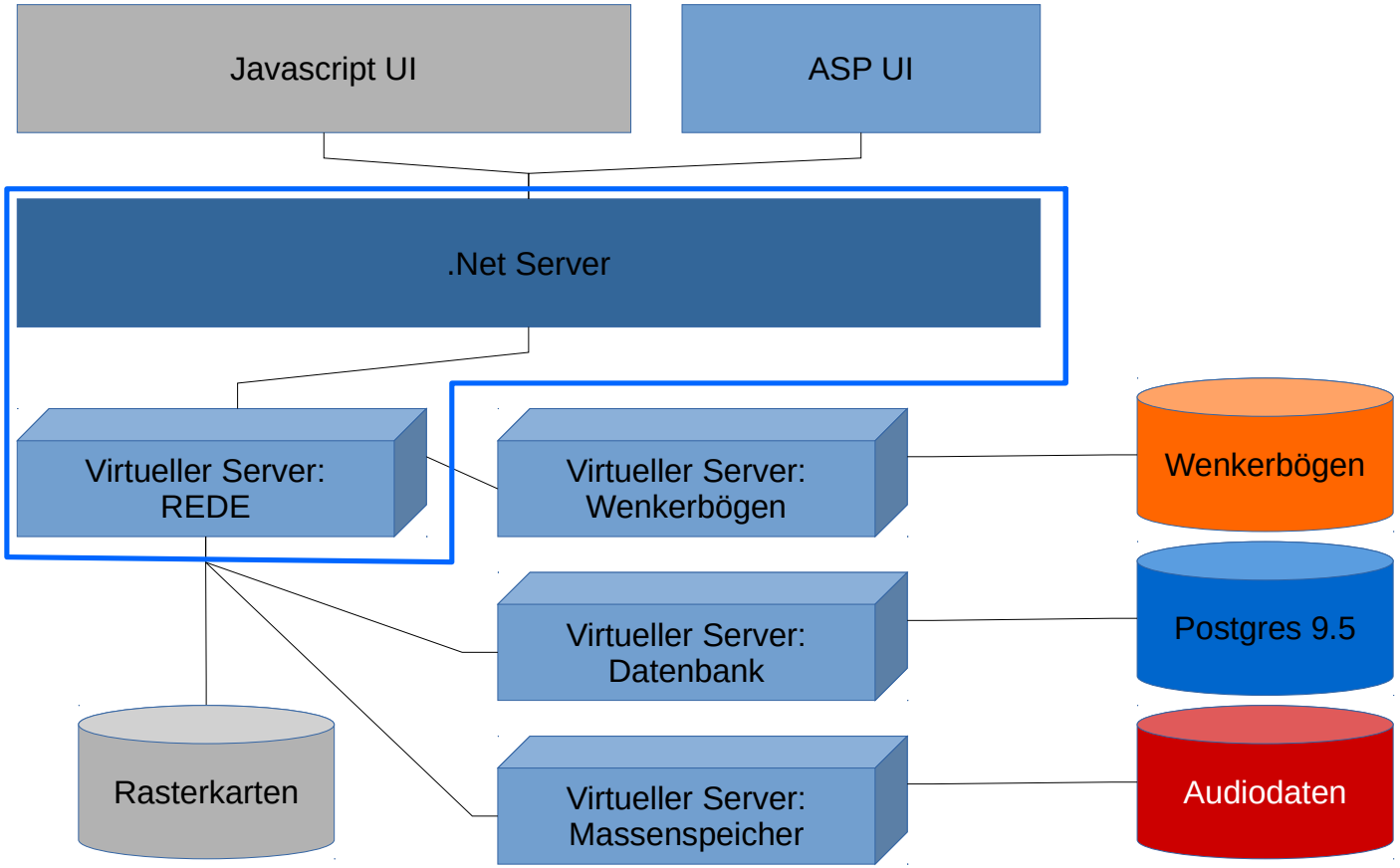
Das Projekt „Regionalsprache.de (REDE)“

- Gefördert durch:
  - Akademie der Wissenschaften und der Literatur | Mainz
- Projektlaufzeit:
  - 2008-2026
- Projektleitung:
  - Jürgen Erich Schmidt, Joachim Herrgen, Roland Kehrein

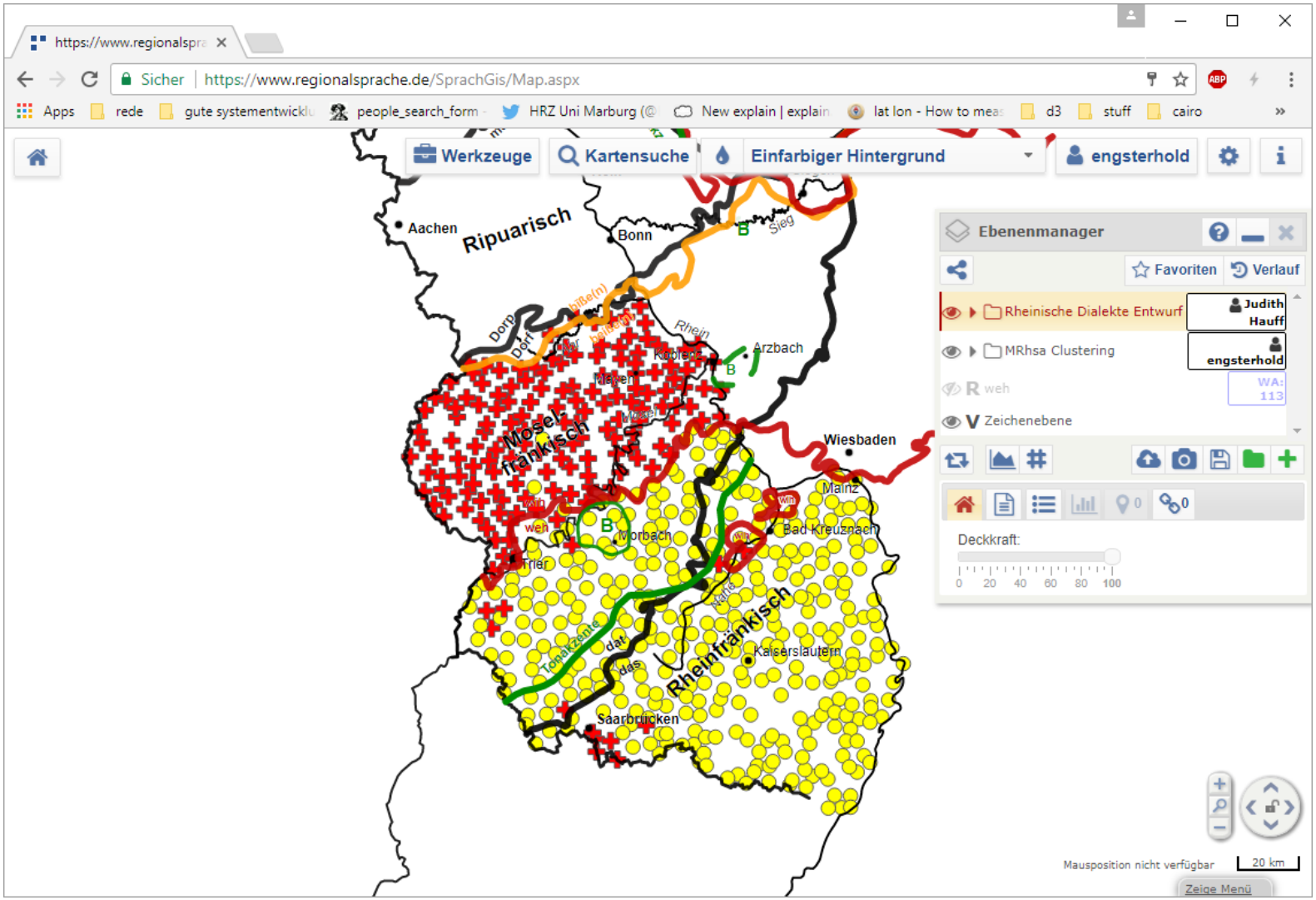
Das Projekt „Regionalsprache.de (REDE)“

- Zwei Teilziele:
  1. Aufbau eines forschungszentrierten Informationssystems zu den modernen Regionalsprachen (Darum geht es heute)
  2. Ersterhebung und Analyse der linguistischen Struktur der modernen Regionalsprachen des Deutschen

Die REDE Infrastruktur



REDE Oberfläche



Der REDE Server

- REDE basiert auf der Webframework mojoPortal (mojoportal.com). Einem umfassenden CMS für die .NET Plattform.
- Das SprachGIS ist eine SinglePage Javascript Applikation eingebettet in die Plattform und basiert hauptsächlich auf OpenLayers2 (openlayers.org) und jQuery (jquery.com)
- Die Angebundenen Kataloge basieren auf den .Net Templates. Es wurde aber angefangen die mit Javascript etwas aufzupeppen.

Der REDE Server

- Kommunikation mit dem Backend geschieht für das SprachGIS entweder über Handler (sprich „normalen“ Webadressen, die mittels GET, POST angesprochen werden und mit Parametern oder Body Payload versehen werden können) oder über Services (.NET spezifische Wrapper, die eine mehr programmatische Einbindung in den JS Quellcode erlauben)

Das REDE Backend

- Basiert auf .NET 4.5
- Fungiert hauptsächlich als Proxy für die Postgres Datenbank oder die anderen Server, wie dem Wenkerserver oder dem Mapserver (mapserver.org)
- Kommunikation mit der Datenbank erfolgt mittels Npgsql
- Stellt die Services / Schnittstellen zur Verfügung.
- Konvertiert die Daten aus der Datenbank in ein für Javascript lesbares Format, sprich JSON.
- Kümmert sich um die Nutzer, Session und Cookie Verwaltung.
- Die Kataloge werden über .NET Databinding erzeugt.

Die REDE Datenbank

- Npgsql erlaubt einen prozedualen Zugriff auf die Datenbank.

```
/// <summary>
///Returns a list of legend elements for a specific map
/// </summary>
/// <param name="MapId">The map id</param>
/// <param name="ShowDeleted">Include deleted legend items</param>
/// <returns>A lsit of legend items</returns>
[DataObjectMethod(DataObjectMethodType.Select, true)]
public List<BaseLegendItem> GetLegendItemsForMap(Int32 MapId, Boolean ShowDeleted)
{
    List<BaseLegendItem> result = new List<BaseLegendItem>();
    using (Connection)
    {
        Connection.Open();
        NpgsqlTransaction t = Connection.BeginTransaction();
        NpgsqlCommand command = new NpgsqlCommand("issg.sp_legend_items_select", Connection);
        command.CommandType = CommandType.StoredProcedure;
        command.Parameters.Add(new NpgsqlParameter(PARAM_MAP_ID, NpgsqlTypes.NpgsqlDbType.Integer)).Value = MapId;
        command.Parameters.Add(new NpgsqlParameter("param_show_deleted", NpgsqlTypes.NpgsqlDbType.Boolean)).Value = ShowDeleted;
        using (NpgsqlDataReader reader = command.ExecuteReader())
        {
            FillWithLegendItems(MapId, result, reader);
        }
        t.Commit();
    }
    return result;
}
```

Die REDE Datenbank

```
-- DROP FUNCTION issg.sp_legend_items_select(integer, boolean);

CREATE OR REPLACE FUNCTION issg.sp_legend_items_select(
    param_map_id integer,
    param_show_deleted boolean)
  RETURNS SETOF issg.legend_item AS
$BODY$

        SELECT
                legend.id,
                legend.sortorder,
                CASE WHEN linked_li.id IS NULL OR legend.overwrite_symbol = true THEN legend.symbol_entry ELSE linked_li.symbol_entry END,
                legend.type,
                legend.deleted,
                legend.edit_user_id,
                users."name",
                legend.change_date,
                legend.comment,
                legend.verified,
                CASE WHEN linked_li.id IS NULL OR legend.overwrite_symbol = true THEN legend.color ELSE linked_li.color END,
                CASE WHEN linked_li.id IS NULL OR legend.overwrite_symbol = true THEN legend.bold ELSE linked_li.bold END,
                CASE WHEN linked_li.id IS NULL OR legend.overwrite_symbol = true THEN legend.italic ELSE linked_li.italic END,
                CASE WHEN linked_li.id IS NULL OR legend.overwrite_description = true THEN legend.color_legend ELSE linked_li.color_legend END,
                CASE WHEN linked_li.id IS NULL OR legend.overwrite_symbol = true THEN legend.label_size_factor ELSE linked_li.label_size_factor END,
                CASE WHEN linked_li.id IS NULL OR legend.overwrite_symbol = true THEN legend.font_family_symbol ELSE linked_li.font_family_symbol END,
                CASE WHEN linked_li.id IS NULL OR legend.overwrite_description = true THEN legend.font_family_legend ELSE linked_li.font_family_legend END,
                CASE WHEN linked_li.id IS NULL THEN legend.transcription ELSE linked_li.transcription END,
                CASE WHEN linked_li.id IS NULL THEN legend.transcription_ipa ELSE linked_li.transcription_ipa END,
                legend.image_url,
                legend.hidden,
                legend.position,
                legend.linked_legend_item_id,
                CASE WHEN linked_li.id IS NULL OR legend.overwrite_description = true THEN array_agg(li21.legend_entry) ELSE array_agg(linked_li21.legend_entry) END,
                CASE WHEN linked_li.id IS NULL OR legend.overwrite_description = true THEN array_agg(li21.occurrence) ELSE array_agg(linked_li21.occurrence) END,
                legend.concat_string,
                legend.overwrite_symbol,
                legend.overwrite_description,
                legend.background_symbol,
                legend.hide_explanation_on_grouping,
                CASE WHEN linked_li.id IS NULL OR legend.overwrite_description = true THEN array_agg(li21.language_id) ELSE array_agg(linked_li21.language_id) END

        FROM issg.tbl_legend_items AS legend
        JOIN mp_users AS users ON legend.edit_user_id=users.userid
        LEFT OUTER JOIN issg.tbl_legend_items2languages AS li21 ON legend.id=li21.legend_item_id

        LEFT OUTER JOIN issg.tbl_legend_items AS linked_li ON legend.linked_legend_item_id=linked_li.id
        LEFT OUTER JOIN issg.tbl_legend_items2languages AS linked_li21 ON linked_li.id=linked_li21.legend_item_id

        WHERE legend.map_id=param_map_id AND (param_show_deleted=true OR legend.deleted=false)
        GROUP BY linked_li.id, legend.id, legend.sortorder, legend.symbol_entry, legend.type, legend.deleted, legend.edit_user_id, users."name", legend.change_date,
                legend.comment, legend.verified, legend.color, legend.bold, legend.italic, legend.color_legend, legend.label_size_factor, legend.font_family_symbol,
                legend.font_family_legend, legend.transcription, legend.transcription_ipa, legend.image_url, legend.hidden, legend.position, legend.background_symbol
        ORDER BY legend.sortorder, legend.position;

  $BODY$
  LANGUAGE sql STABLE
  COST 100
  ROWS 100;
ALTER FUNCTION issg.sp_legend_items_select(integer, boolean)
  OWNER TO mojouser;
GRANT EXECUTE ON FUNCTION issg.sp_legend_items_select(integer, boolean) TO mojouser;
GRANT EXECUTE ON FUNCTION issg.sp_legend_items_select(integer, boolean) TO public;
```

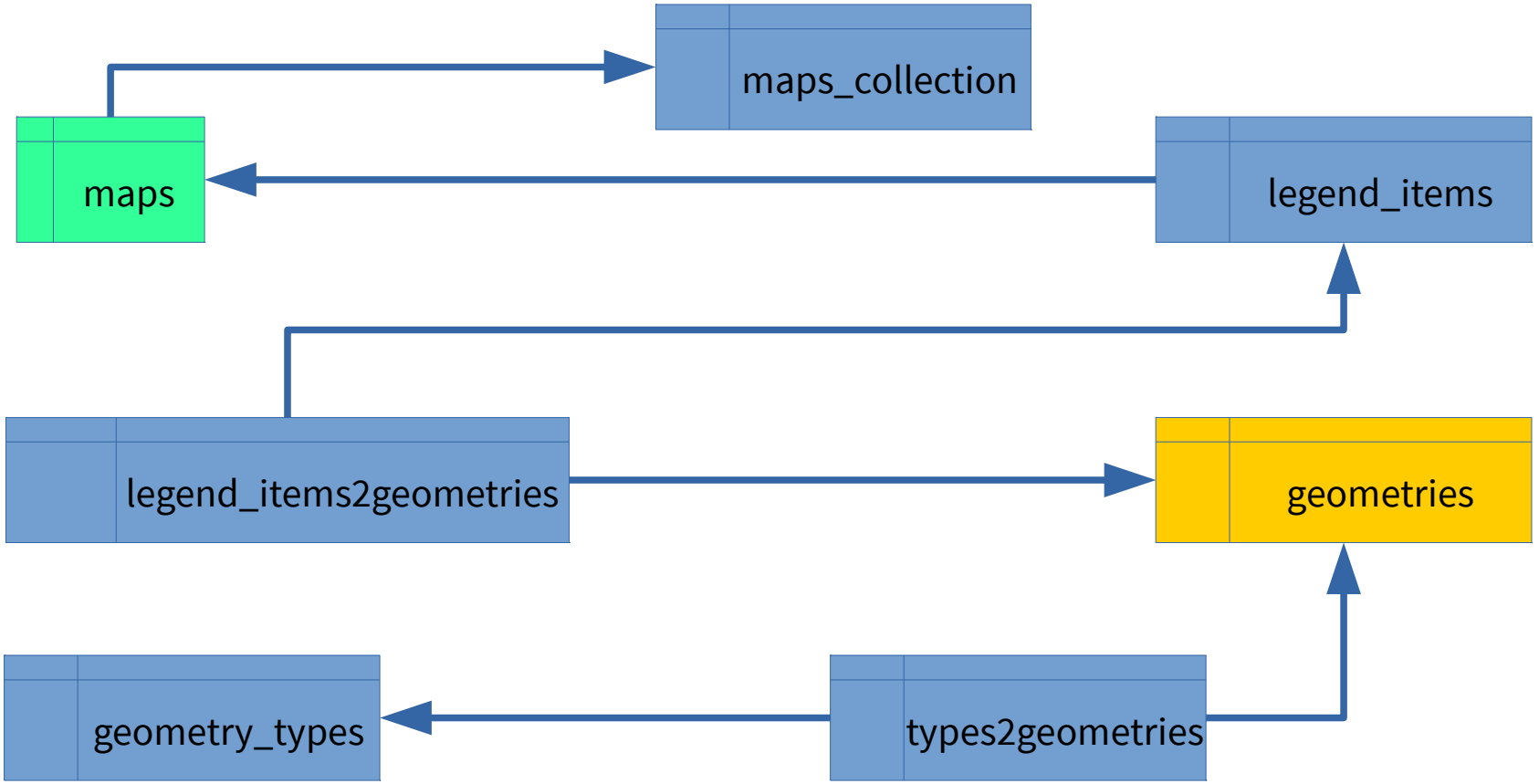
Die REDE Datenbank

- Zur Zeit Postgres9.5 (Wegen der schönen JSON Features)
- Natürlich Postgis2.x für spatiale Operationen und Geodatenspeicherung.
- Plr um Zugriff auf R Funktionen zu bekommen. Zum Beispiel zur Voronoi Tessellation
- Plv8 um Zugriff auf Javascript Funktionen zu bekommen. Sehr Hilfreich für verschachtelte Objekte.

Die REDE Datenbank

- Unterteilt in thematische Schemata (geodata, maps, audio, issg, public)
- Zentrale Tabelle ist geodata.tbl_geometries für die Systemgeometrien
- Maps.tbl_maps speichert alle Arten von Karten
  - Benutzerkarten werden mit Informationen aus issg.tbl_geometries2attributes gejoint. Die Daten / Attribute sind als jsonb Objekte gespeichert
  - Vektorkarten werden mit issg.tbl_legend_items gejoint. Da drinne sind die Daten für die Kartensymbole erfasst.
  - Rasterkarten werden mit maps.tbl_maps_raster gejoint, wo die Verweise auf die .map-Datei für den Mapserver gespeichert sind.

Die REDE Datenbank



Das SprachGIS (nochmal)

- Das SprachGIS stellt verschiedene Werkzeuge zur Verfügung.
- Diese Werkzeuge sind Unterklassen von OpenLayers.Control.
- Diese Werkzeuge liegen über der Karte, dem #map-Container.
- Die Karte besteht aus verschiedenen Ebenen. Hauptsächlich den geladenen Karten (Raster oder Vektor) aber auch „unsichtbaren“ Kontrollebenen, die zum Beispiel klicken auf Elemente verhindern.

Das SprachGIS (immer noch)

- Die Hauptanwendungsgebiete sind Zugriff auf Karten (Raster oder Vektor), Audioaufnahmen, räumliche Literaturrecherche und Darstellung und Einbindung eigener Daten und Erzeugung von eigenen Karten.
- Suche funktioniert ungefähr so:
  - **POST**: https://www.regionalsprache.de/Handler/AsyncFeatureInformationService.ashx
  - **PAYLOAD**: filterOptions={"Epsg":"900913","Buffer":1,"PlaceName":"Weh","WktGeometry":null,"Id":0,"SortBy":"ALPHABETIC","SortOrder":"ASC","GeometryCategoryIds":[],"Gids":[]}&emptyGeometries=true&bibliography=true&audio=true&wenkerboegen=true&mapCollection=true&woerterbuchnetz=true&dgdInformation=true
- Das Aufarbeiten der Karten kann sehr aufwendig sein.
- Rasterkarten werden aus einem Informationsobjekt über den Mapserver geladen
- Vektorkarten werden aus der Datenbank generiert

Das SprachGIS - Datenvisualisierung

- Datenvisualisierung erfolgt über D3 (Data Driven Documents, d3js.org), was die populärste Visualisierungsbibliothek für Javascript ist.
- Beim erstmaligen Laden des Visualisierungswerkzeug wird auf Basis der Daten eine Visualisierungsvorschrift als JS Objekt erstellt und die Daten in eine Form gebracht, dass sie dem D3 Prinzip entsprechen.
- Da JS Objekte Referenzen sind ist gewährleistet, dass alle Komponenten des Visualisierungswerkzeug auf dem selben Objekt arbeiten.
- Es herrscht eine strikte Trennung zwischen den Daten und der Visualisierungsvorschrift, welche in der Karte abgespeichert werden kann.
- Laden einer Karte mit dieser Visualisierungsvorschrift initialisiert das Visualisierungswerkzeug mit dieser Vorschrift.

Das SprachGIS - Export

- Das Erstellen der Karte geschieht auf dem Server. Was gerendert werden soll, kommt allerdings vom Client. Für Vektorebenen der svg String. Für Rasterkarten die Links zu den einzelnen Tiles.
- Damit der richtige Ausschnitt in der richtigen Größe gerendert wird, wird ein auf dem Client eine „virtuelle“ #map erzeugt. Dadurch werden von OpenLayer alle Werte passend berechnet.
- Diese Daten werden dann an den Server gesendet.

Kartenexport

- Der Kartenexport ist ein python Modul.
- Er basiert auf dem Pillow Modul und sehr stark auf dem cairo-Renderer. Das ist der Renderer für das GTK einer grafischen Bibliothek für Linux. Das GTK kommt zum Beispiel in Gimp oder Inkscape zum Einsatz und ist der Unterbau von Gnome.
- SVG wird in ein Bild mittels rsvg umgewandelt (technisch gesehen, konvertiert rsvg die svg Anweisungen in cairo Anweisungen)
- Wenn man nur svg-Daten exportieren will, kann man neuerdings als Exportformat pdf auswählen und bekommt eine PDF die die svg-Daten auf eine PDF-Surface gerendert hat. D.h. keine Verpixelung und eine beliebige Skalierung.

Sprachatlanten

- REDE beherbergt inzwischen viele Sprachatlanten.
- Ausgangspunkt (auch für das Design der Datenbank dazu) war der MrhSA
- Zur Zeit in Bearbeitung: die Bairischen Atlanten
- Neu: der KNSA – Als neu aufgelegt digitale Publikation in REDE

Atlanten in REDE

Atlas	Kartentypen	Bearbeitungsstand
Dialektatlas Westmünsterland (DWALN)	Laut, Form, Wortschatz	verfügbar
Fränkischer Sprachatlas (FSA)	Form, Wortschatz	verfügbar
Großer Historischer Weltatlas (GHW)	außersprachliche Interpretamente	verfügbar
Mittelrheinischer Sprachatlas (MRhSA)	Laut, Form, Wortschatz	verfügbar
Sprachatlas von Bayerisch-Schwaben (SBS)	Laut, Form, Syntax, Wortschatz	verfügbar
Schlesischer Sprachatlas (Schles.SA)	Laut, Form, Wortschatz	verfügbar
Sudetendeutscher Sprachatlas (SDWA)	Wortschatz	verfügbar
Sprachatlas von Mittelfranken (SMF)	Laut, Form, Syntax, Wortschatz	verfügbar
Sprachatlas von Niederbayern (SNiB)	Laut, Form, Syntax, Wortschatz	verfügbar
Sprachatlas von Nordostbayern (SNOB)	Laut	verfügbar
Sprachatlas von Oberbayern (SOB)	Laut, Form, Syntax, Wortschatz	verfügbar
Sprachregion München (SRM)	Laut, Form, Wortschatz	spätestens 2022 verfügbar
Sprachregion Nürnberg (SRN)	Wortschatz	spätestens 2022 verfügbar
Südwestdeutscher Sprachatlas (SSA)	Laut, Form, Wortschatz	verfügbar
Sprachatlas von Unterfranken (SUF)	Laut, Form, Syntax, Wortschatz	verfügbar
Thüringischer Dialektatlas (ThDA)	Wortschatz	verfügbar
Vorarlberger Sprachatlas (VALTS)	Laut, Form, Wortschatz	verfügbar
Wortatlas der städtischen Umgangssprache (WASU)	Wortschatz	spätestens 2022 verfügbar
Wortatlas der deutschen Umgangssprachen (WDU)[38]	Wortschatz	spätestens 2022 verfügbar
Wortgeographie der städtischen Alltagssprache in Hessen (WSAH)	Wortschatz	spätestens 2022 verfügbar

Sprachatlanten

- Zwei Arten der Digitalisierung:
  - Scan mit anschließender Georeferenzierung für eine Rasterkarte.
  - Vektorisieren der Daten. Sprich: Transformieren der Kartendaten in Datenbankeinträge (legend_items).
- Vektorisieren der Daten ist sehr aufwendig:
  - Im schlechtesten Fall muss unser umfangreiches Kartenteam diese Daten über eine spezielle Maske in REDE per Hand eingeben
  - Im besten Fall kann ein semiautomatischer Import einen Teil der Daten automatisch eintragen
  - In beiden Fällen müssen die Daten noch überprüft und gegebenenfalls Angepasst werden.

Sprachatlanten – Semiautomatischer Import

- Daten lagen entweder als dbf-Datei vor oder als position-indizierte Textdatei.
- Symbole wurden als HPGL Anweisungen beschrieben.
- Damals wurden die Atlanten mit einem selbst entwickelten Programm namens TeuthoTex erzeugt.
- Ein dazugehöriges interaktives Programm wurde in Pascal entwickelt.

Sprachatlanten – Semiautomatischer Import

- Alle Symbole (die in einer extra dbf Datei vorlagen) wurden mit einem selbstgeschriebenen Programm von HPGL in SVG übersetzt und dann in mittels Fontforge in eine Schrift importiert. Problem: Rasterungen gingen dabei verloren.
- Dbf wurde in csv umgewandelt und dann mittels eines eines selbstgeschriebenen python Skripts zusammen mit den txt-Files auf ein einheitliches Format gebracht und als csv gespeichert.
- Diese csv wurde in die Datenbank importiert und mittels Mapping-Tabellen erweitert.
- Diese Daten wurden dann in legend_items eingefügt und mit den entsprechenden Orten verknüpft.

Sprachatlanten – Semiautomatischer Import

- Nachbearbeitung erforderlich Insbesondere da die Legendeneinträge nicht mit importiert wurden.
- Die Daten unterscheiden sich SEHR von den gedruckten Atlanten. Auch was verwendete Symbole angeht.
- Überprüfung und Nachbearbeitung immer noch sehr zeitaufwendig.
- Insbesondere mussten noch viele Zeichen (Symbole und Glyphen) gebastelt werden

Theutonista goes Unicode

- Seit 2014 sind ein paar Theutonista Diakritika in Unicode 7 aufgenommen. (1AB0-1ABF)
- Die Unterstützung dieser Zeichen in Schriften und Programmen ist mangelhaft
- In den Atlanten tauchen auch Zeichen(-kombinationen) auf die nicht in Unicode aufgenommen wurden.
- Bisher: Man hat das (komplette) Zeichen in einer Schrift gebastelt und mit einem Codepoint aus der PUA (oder auch anderen Bereichen) versehen.
- Neu: Wir bauen die Diakritika und verwenden das Ankerpunkt Paradigma in Opentype, um diese Zeichen mit anderen zu kombinieren. (work in progrss)

Der Kleine Niederländische Sprachatlas (KNSA) Eckdaten

- *Kleiner Niederländischer Sprachatlas unter Einschluss des Westfriesischen* (KNSA)
  - Autoren: Werner H. Veith (†) und Lutz Hummel
  - Johannes-Gutenberg Mainz (1996–2006)
  - westliche Fortsetzung des *Kleinen Deutschen Sprachatlas*
  - Umfang: 450 Ortspunkte, 374 Karten + Manuskript
- Timeline:
  - 2009: Projekt Stillstand
  - 2014: Anfang der Zusammenarbeit zwischen Lutz Hummel und dem Forschungszentrum Deutscher Sprachatlas in Marburg
  - 2017: Fertigstellung und **erstmalige** Publikation des KNSA im REDE SprachGIS

Der KNSA

- Der KNSA wurde als Onlinepublikation in REDE neu aufgelegt.
- Dafür wurde eine Dereferenzierungfunktion für URLs eingeführt:
  https://www.regionalsprache.de/SprachGis/vectormap/knsa/v44
  bringt einen direkt zur gewünschten Karte.
- Diese Dereferenzierung funktioniert für alle freigeschalteten Atlanten nach dem Format:

  /SprachGIS/[KartenTyp]/[Atlas]/[Band?]/[Kartennummer]

Der KNSA

- Bei der Darstellung Trennung zwischen verschiedenen Ebenen. Dazu wurden Ebenengruppen eingeführt. Das heißt eine „Karte“ kann nun entweder eine Karte oder eine Gruppe von Karten sein.
- Das selbe für Legenden
- Das erforderte Änderungen an der Datenbank um hierarchische Strukturen abbilden zu können

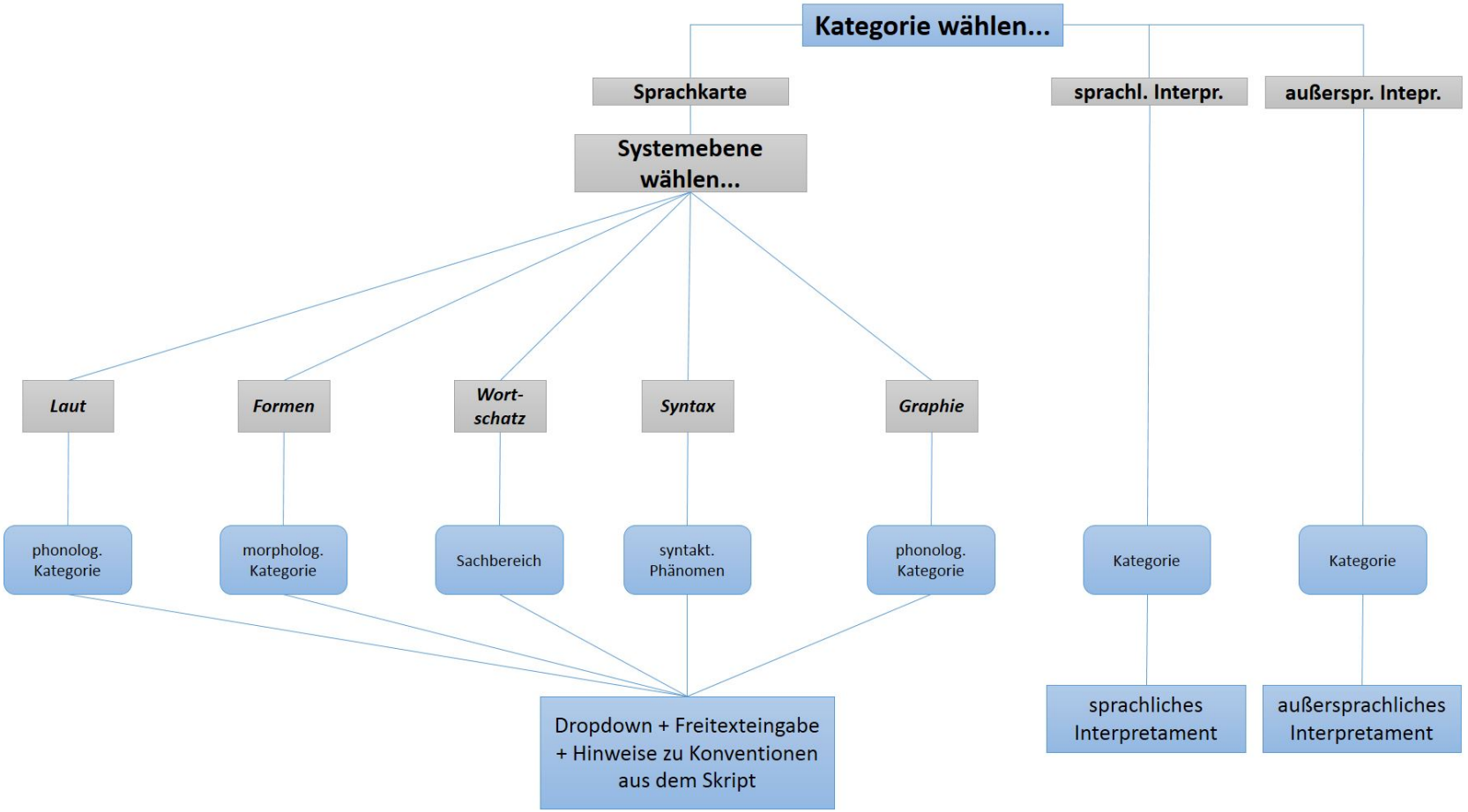
Baustellen

- Wir wollen unser Unicode Problem beheben. Arbeiten an einem Mapping der selbstgebastelten Zeichen auf offizielle Unicodepoints
- Topologische Normalisierung unserer Geometrien für Europa (und darüber hinaus)
- Überarbeitung der Metadaten zu den Karten, um eine bessere linguistische Annotation …

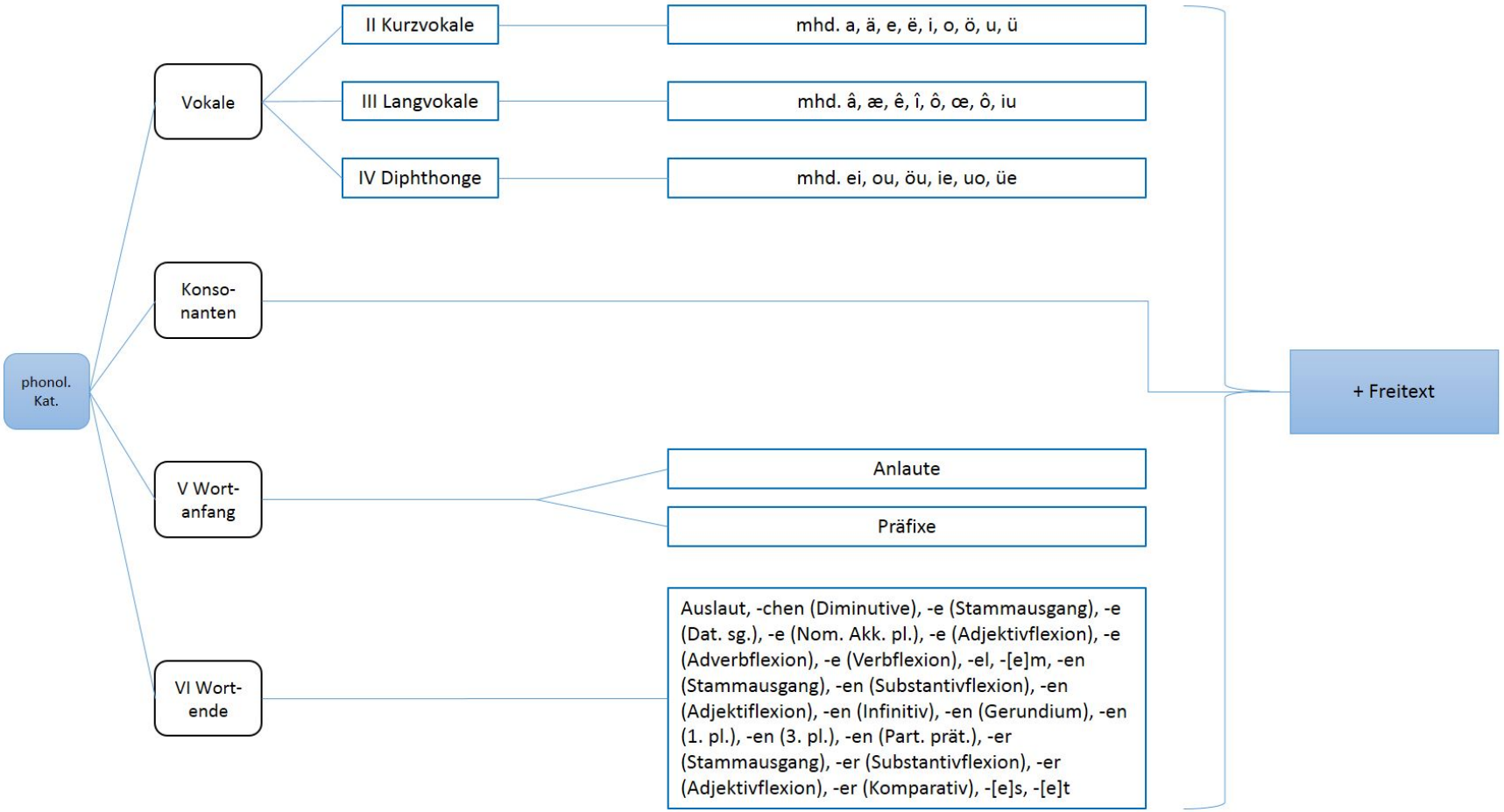
Baustellen - Sprachkarten

- Bessere semantische Annotation der Karten, um gezieltere Suche und Filterung vor nehmen zu können.
- Grundprinzipen basieren auf bewährten Methoden und werden bereits (teilweise) miterfasst.
- Erweiterung dieser Prinzipien und Nutzbarmachung durch eine deutlich vereinfachte Suchmaske.

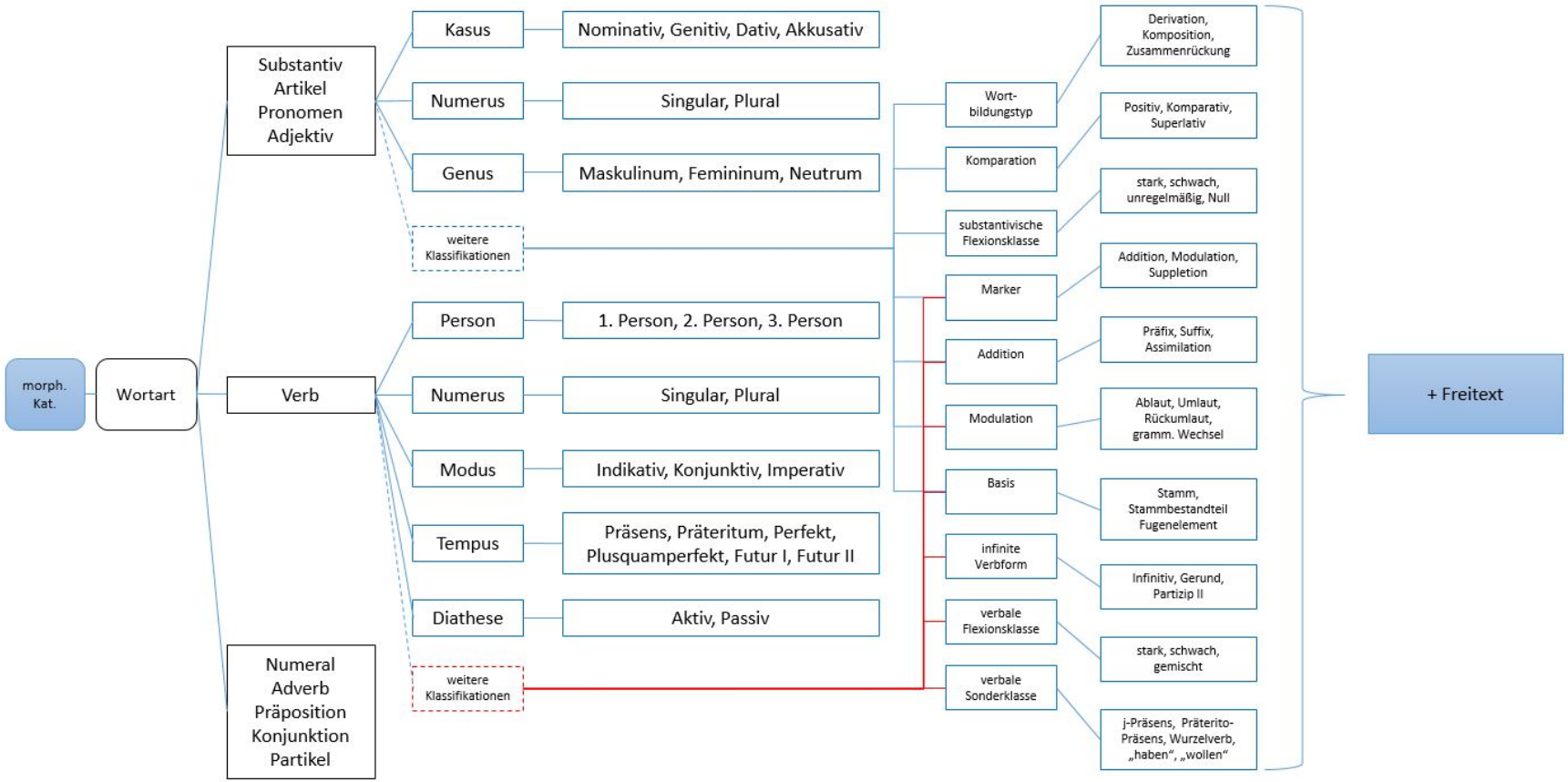
Kartentypen



Lautkarten


phonol. Kat.
Vokale
II Kurzvokale
mhd. a, ä, e, ë, i, o, ö, u, ü
III Langvokale
mhd. â, æ, ê, î, ô, œ, ô, iu
IV Diphthonge
mhd. ei, ou, öu, ie, uo, üe
Konso-nanten
V Wort-anfang
Anlaute
Präfixe
VI Wort-ende
Auslaut, -chen (Diminutive), -e (Stammausgang), -e (Dat. sg.), -e (Nom. Akk. pl.), -e (Adjektivflexion), -e (Adverbflexion), -e (Verbflexion), -el, -[e]m, -en (Stammausgang), -en (Substantivflexion), -en (Adjektiflexion), -en (Infinitiv), -en (Gerundium), -en (1. pl.), -en (3. pl.), -en (Part. prät.), -er (Stammausgang), -er (Substantivflexion), -er (Adjektivflexion), -er (Komparativ), -[e]s, -[e]t
+ Freitext

Formenkarten



Probleme und Diskussion

- Das Internet entwickelt sich sehr schnell. Es ist schwierig da am Ball zu bleiben.
- Welche „Technologien“ sind längerfristig, was ist ein Fad.
- Wieviele Freiheiten lässt man Leuten bei der Kartenerstellung (in einem gerichteten Websystem).
- Datenbanken ändern sich über die Zeit. Auch die Anforderungen an eine Datenbank ändern sich.
- Der Unterschied zwischen was der Mensch sieht und was der Computer „sieht“
- Heterogene Daten vereinen ohne Informationen zu verlieren.
- Unterschiede zwischen den Daten und einem Endprodukt.
- Nachbearbeitung und Überprüfung kann sehr aufwendig sein.
- (computer) semantische Korrektheit der Karten / Daten.
- Schriften sind ein Thema für sich.