Entwurfsbeschreibung

SWP12-10

Peggy Lucke
Felix
Ha Tran
Paul Röwer
Alexander Richter
Nathanael Philipp

15. April 2012

Inhaltsverzeichnis

1 Allgemeines

LGDEditTool ist ein einfaches Bearbeitungswerkzeug für LinkedGeoData, welches den Nutzer befähigen soll, Mappings der OpenStreetMap-Daten zu editieren und auch zu löschen. Es vereinfacht die komplexe Eingabe in die Datenbank von LinkedGeoData und bietet in der Edit-History die Möglichkeit, die Änderungen einzusehen und wieder zurück zu nehmen. Außerdem ist ein leichtgewichtiger Ontologie-Editor integriert, mit dem z.B. Subklassenbeziehungen erstellt werden können. Für den Administrator besteht die Möglichkeit, in einem nur für ihn sichtbaren Tab die Edit-Historie zu löschen. Das Tool läuft in jedem Browser mit JavaScript-Unterstützung. Es ist also betriebssystemunabhängig.

2 Produktübersicht

Das Tool wird im Webbrowser gestartet und besteht im Wesentlichen aus einer Weboberfläche, welche die Datenbank, auf die sie zugreift, überdeckt und die Befehle in Buttons ausführbar macht. Damit der Nutzer einen Eintrag in der Datenbank gezielt suchen kann, gibt es noch ein Eingabefeld für die Suche nach Keys und Values. Die Darstellung der Daten erfolgt in Tabellen. Aufgerufen wird das Tool ohne Installation über eine Webseite.

3 Grundsätzliche Struktur- und Entwurfsprinzipien für das Gesamtsystem

Unser Programm setzt bei der Programmierung die Kenntnisse über folgende Programmiersprachen voraus: Java Server Pages, Cascading Style Sheets, Java, JavaScript sowie SQL.
Die Entwicklung unseres Programmes verläuft nach dem Top-Down-Schema. Dabei wurde zuerst die Oberfläche konstruiert und danach die Funktionalitäten der einzelnen Tabs implementiert. Das HTML-Dokument, welches im Browser des Nutzers die Oberfläche des Tools darstellt, wird dynamisch von einem Tomcat-Server aus dem JSP-Code-Dokument *index.jsp* generiert. Je nach ausgewähltem Tab wird dabei das entsprechende Java-Template ausgeführt, welches gegebenenfalls die notwendigen Datenbankabfragen vornimmt und anschließend den HTML-Code zur Darstellung der Ergebnisse erzeugt. Die Autovervollständigung wird mit dem JQuery-Framework implementiert.

4 Grundsätzliche Struktur- und Entwurfsprinzipien der einzelnen Pakete

4.1 Verwendete Bibliotheken

Wir verwenden für die Verbindung mit der PostgreSQL-Datenbank den JDBC-Treiber (PostgreSQL 9.1, JDBC 3/4 ↑). Für die die Autovervollständigung der Suchfelder die jquery.autocomplete.js (↑).

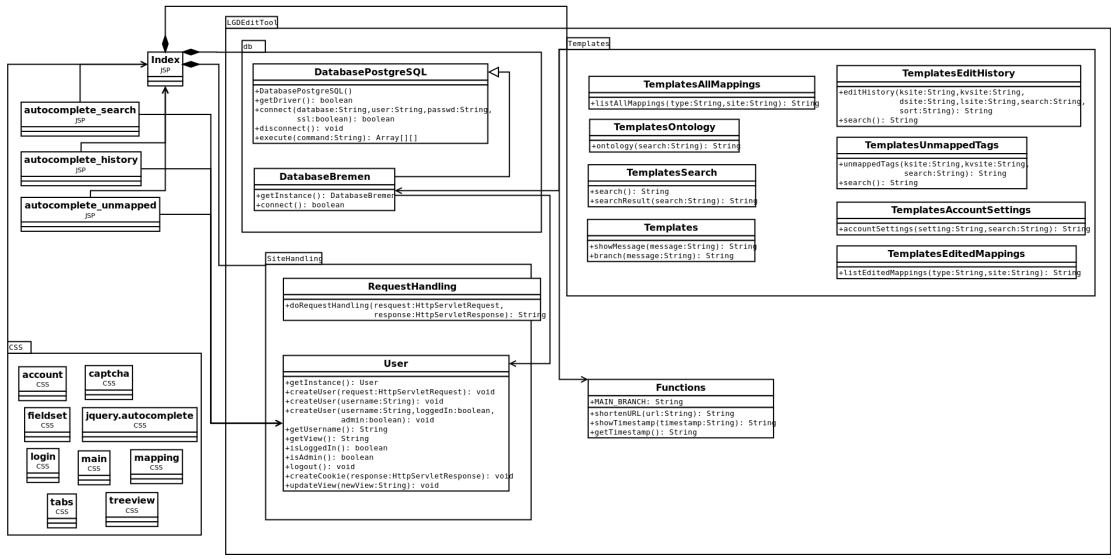

Abbildung 1: statisches Modell

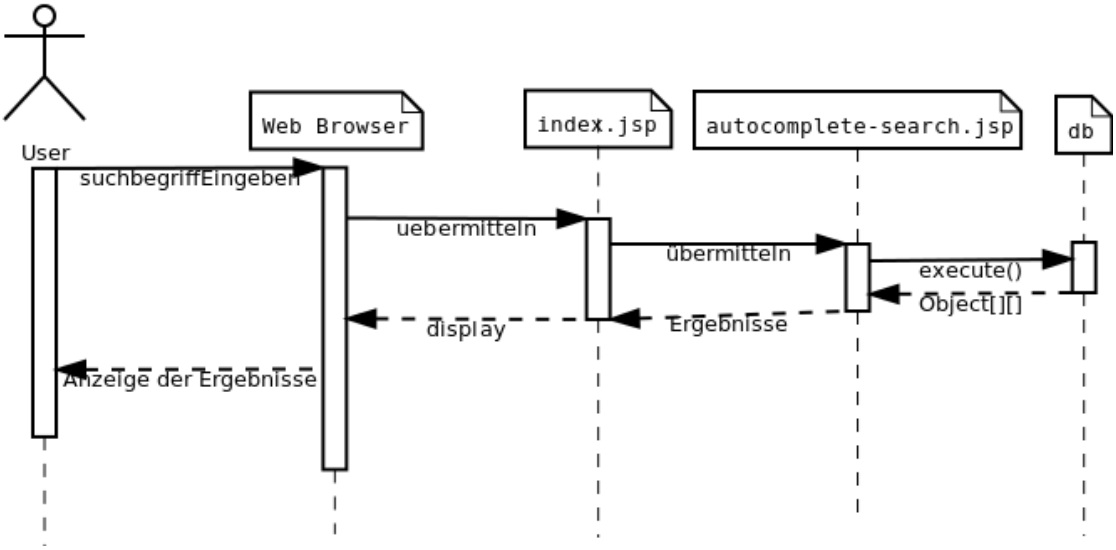

Abbildung 2: Autovervollständigung

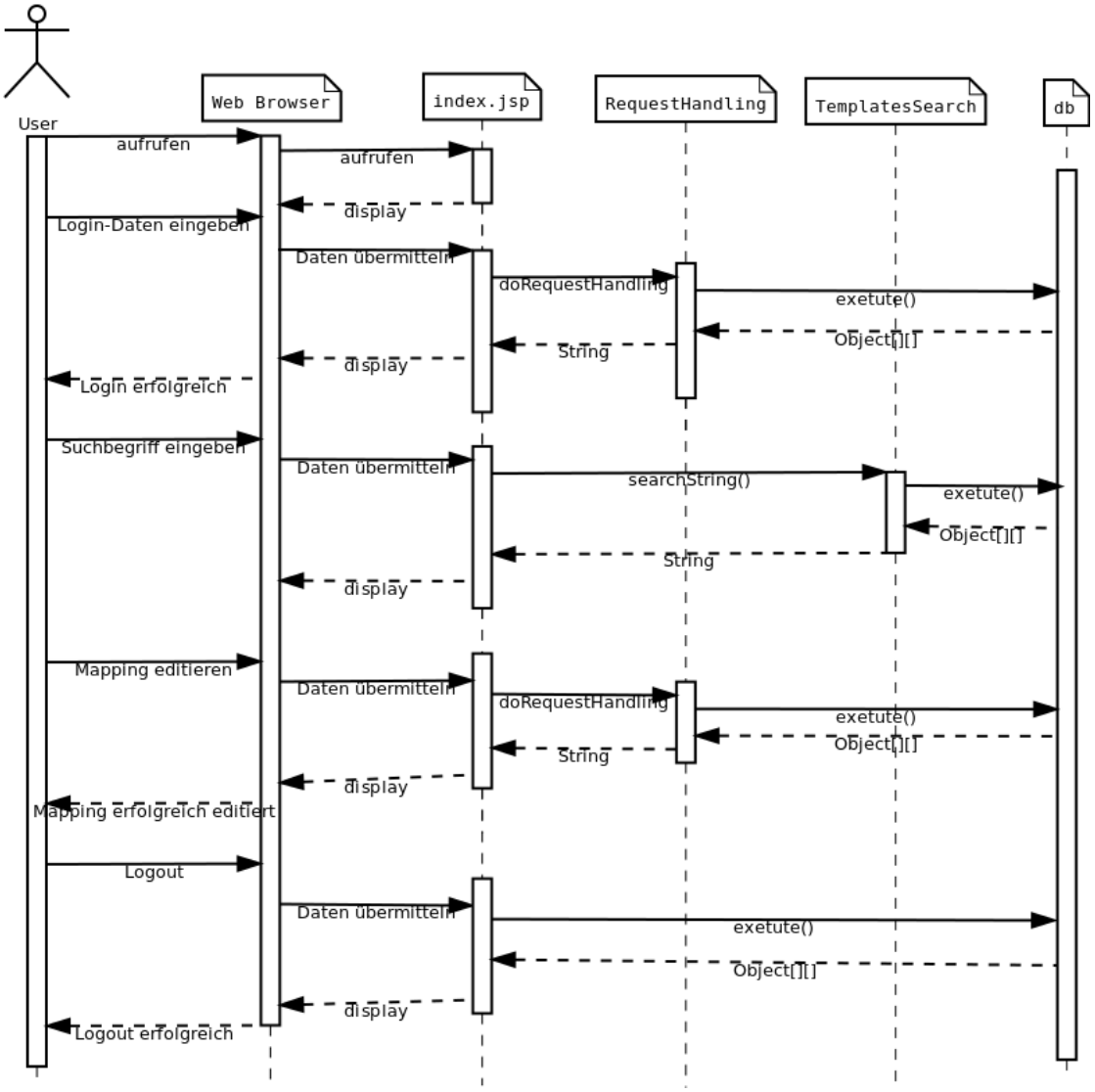

Abbildung 3: Login und Edit

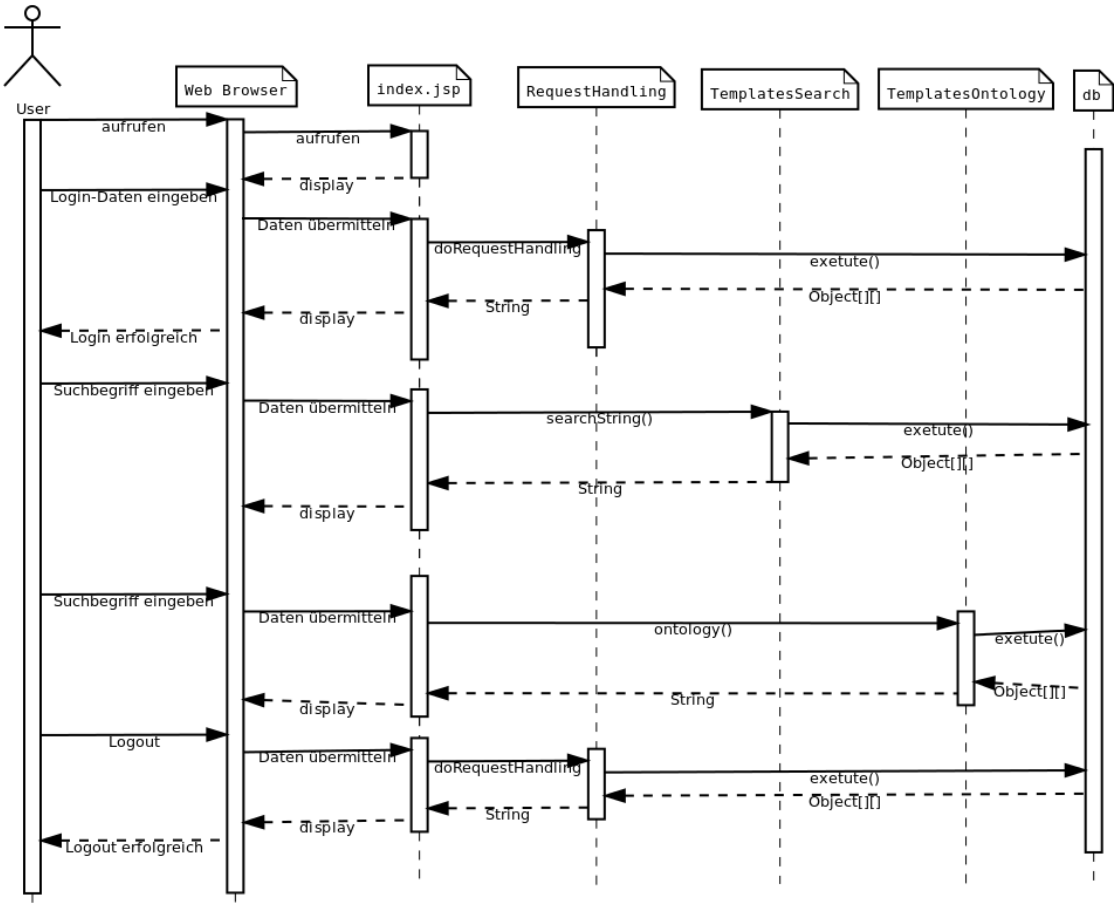

Abbildung 4: Ontologie

4.2 Ordner *css*

In diesem Ordner sind alle Gestaltungsvorgaben, um ein einheitliches Webdesign unseres Tools für alle Webbrowser zu ermöglichen.

4.3 Ordner *js*

Jeder in JavaScript geschriebene Code ist hier vertreten. Die beiden Javascript-Dateien sind nicht selbst geschrieben, sondern wurden übernommen.

4.3.1 *jquery.autocomplete.js & jquery.js*

Enthält den für die Anzeige der Autovervollständigung der Suchfelder erforderlichen Code.

4.4 *autocomplete_search.jsp & autocomplete_history.jsp & autocomplete_unmapped.jsp*

Enthält die Datenbankabfrage für die Autovervollständigung der Suchfelder. Entsprechend der Namen der jsp-Dateien sind diese Dateien für die Autovervollständigung der Suchfelder in den entsprechenden Tabs verantwortlich.

4.5 *index.jsp*

Wird aufgerufen, sobald das Tool im Web angesprochen wird und dient der Verwaltung des kompletten Inhalts der Webseite. Es setzt also die verschiedenen Klassen, jsp-Dateien und JavaScript-Dateien zusammen.

4.6 LGDEditTool-Paket

Ist das Hauptpaket, dem die anderen Pakete, die Javacode enthalten, untergeordnet sind.

4.6.1 Functions

Ist eine Klasse in der die folgenden zwei Felder und drei Methoden enthalten sind.

Attribut	Methode	Funktion
MAIN_BRANCH: String		Speichert den View des Main-Branches für die Autovervollständigung in der Suche.
	shortenURL(url: String): String	Kürzung von URLs zur besseren Darstellung in Tabellen
	showTimestamp (timestamp: String): String	formatiert den timestamp der Historytabellen in vom Nutzer besser lesbares Format
	getTimestamp(): String	generiert aktuellen Timestamp aus der Datenbank im Format YYYY-mm-ddTHH:MM:ss und gibt dieses zurück. Dies wird in allen Mappings der Edit-History benötigt, um genaue Zeitpunkte zu dokumentieren.

4.6.2 SiteHandling-Paket

Enthält die folgenden beiden Klassen

RequestHandling:
Die Auswertung der Formular (K/KV/Datatype/Literal-Mappings mit den Operationen Edit/Delete/Restore/Commit/Clear/Create) werden hier vorgenommen, dazu werden aus den Formulardaten SQL-Querys generiert.

Methode	Funktion
doRequestHandling(resquest: HttpServletRequest, response: HttpServletResponse): String	Formularauswertung

User:
Ist eine Singleton-Klasse, das heißt es gibt nur eine Instanz von dieser Klasse. Sie repräsentiert einen Nutzer.

Methode	Funktion
getInstance(): User	wenn eine Instance besteht, gibt es diese zurück; anderenfalls wird eine neue erstellt
createUser(request: HttpServletRequest): void	erzeugt User
createUser(username: String): void	erzeugt User
createUser(username: String, loggedIn: boolean, admin: boolean): void	erzeugt User
getUsername():String	gibt Nutzernamen des aktuellen eingeloggten Benutzers zurück, sonst leeren String
getView(): String	gibt aktuellen Branch zurück auf dem User arbeitet, userbranch oder mainbranch
isLoggedIn(): boolean	true- wenn Nutzer eingeloggt, false- wenn nicht
isAdmin(): boolean	true- Nutzer ist Admin, false- falls nicht
logout(): void	beendet Nutzersession
createCookie(response: HttpServletResponse): void	erzeugt Cookies, welches Username speichert, und ob der User eingeloggt ist
updateView(newView: String): void	ändert den view auf den übergebenen wert und speichert diesen in der Datenbank

4.6.3 Templates-Paket

Für jeden Tab gibt es ein Template. Diese Templates generieren einen String für jeden Tab, der den HTML-Code für dessen Darstellung beinhaltet. Diese werden aus vorherigen Eingaben mit Hilfe von SQL-Querys generiert.

Templates:

Methode	Funktion
showMessage(message:String): String	dient zur Ausgabe von Nachrichten an den User
branch(message:String):String	liefert den Namen des Branches auf dem zur Zeit gearbeitet wird

TemplatesAllMappings:

Methode	Funktion
listAllMappings(type:String, site:String):String	generiert html-Code, welcher zur Darstellung der Mappings im ListAllMapping-Tab dient

TemplatesOntology:

Methode	Funktion
ontology(search:String):String	generiert html-Code, welcher zur Darstellung der Mappings im Ontology-Tab dient

TemplatesSearch:

Methode	Funktion
search(): String	generiert Html-Code mit Inputbox für die Suche auf der Datenbank
searchResult(search:String): String	generiert Html-Code um das Resultat der Suchanfrage darzustellen

TemplatesEditHistory:

Methode	Funktion
editHistory(ksite: String, kvsite: String, search: String): String	generiert Html-Code, welcher zur Darstellung der Mappings im EditHistory-Tab dient
search(): String	generiert Html-Code mit Inputbox für die Suche auf der Datenbank

TemplatesUnmappedTags:

Methode	Funktion
unmappedTags(ksite: String, kvsite: String, search: String): String	generiert Html-Code, welcher zur Darstellung der Mappings im UnmappedTags-Tab dient
search(): String	generiert Html-Code mit Inputbox für die Suche auf der Datenbank

TemplatesAccountSettings:

Methode	Funktion
accountSettings(setting: String, search: String): String	generiert Html-Code, welcher zur Darstellung der Mappings im AccountSetting-Tab dient

TemplatesEditedMappings:

Methode	Funktion
listEditedMappings(type: String, site: String): String	liefert Html-Code zum Bearbeiten der angezeigten Mappings

4.6.4 db-Paket

Enthält eine Klasse für die Verbindung mit einer PostgreSQL-Datenbank sowie eine davon abgeleitete Singleton-Klasse, die alle Werte beinhaltet um eine Verbindung mit Bremen-Datenbank enthält, diese können später dann durch die Werte für die LGD-Datenbank ersetzt werden.

DatabasePostgreSQL:

Methode	Funktion
DatabasePostgreSQL()	Konstruktor, überprüft ob die Treiber vorhanden sind
getDriver(): boolean	Gibt `true` zurück, wenn Treiber existiert, ansonsten `false`.
connect(database: String, user:String, passwd: String, ssl: boolean): boolean	Stellt eine Verbindung zur Datenbank her, gibt `true` zurück, wenn eine neue Verbindung erfolgreich aufgebaut wurde. Andernfalls wird `false` zurück gegeben. Wenn es zu einem Verbindungsfehler kommt wird eine `SQLException` geworfen.
disconnect(): void	Trennt eine bestehende Verbindung mit der Datenbank.
execute(command: String): Object[ ][ ]	Führt die angegebenen SQL-Query aus und gibt die angeforderten Werte zurück. Sollte es zu einem Fehler kommen wird eine `SQLException` geworfen.

5 Datenbank

Wir haben den Datentyp lgd_datatype um den Wert *deleted* erweitert. Dieser Wert zeigt an, das ein Datatype-Mapping gelöscht ist.

5.1 lgd_map_datatype

Spalten	Typ	Bedingung	Funktion
k	text	not null	
datatype	lgd_datatype	not null	
user_id	text		zur Identifikation des Userspaces
last_history_id	integer		Verknüpfung mit Wert vor der letzten Änderung des Mappings

Wir haben die Spalten *user_id* und *last_history_id* zu dieser Tabelle hinzugefügt. Des weiteren ist *k* nicht länger Primary Key dieser Tabelle.

5.2 lgd_map_datatype_history

Wir haben diese Tabelle komplett neu erstellt. Sie dient dazu die Edit-History für Datatype-Mappings zu speichern.

Spalten	Typ	Bedingung	Funktion
id	integer	not null	Primärschlüssel zur Identifikation der Änderung des Mappings
k	text	not null	Key des Mappings
datatype	lgd_datatype	not null	Datentyp des Mappings
user_id	text		Benutzername, welcher die Änderung durchgeführt hat
comment	text		Kommentar des Nutzers zur Änderung
timestamp	text	not null	Zeitpunkt der Änderung
action	text		Art der Änderung (Editieren, Löschen, etc.)
userspace	text		zur Identifikation des Userspaces
history_id	text		Verknüpfung mit Wert vor der letzten Änderung des Mappings

5.3 lgd_map_label

Spalten	Typ	Bedingung
k	text	not null
v	text	not null
language	char 16	not null
label	text	not null

5.4 lgd_map_literal

Spalten	Typ	Bedingung	Funktion
k	text	not null	
property	text	not null	
language	text	not null	
user_id	text		zur Identifikation des Userspaces
last_history_id	integer		Verknüpfung mit Wert vor der letzten Änderung des Mappings

Wir haben die Spalten *user_id* und *last_history_id* zu dieser Tabelle hinzugefügt. Des weiteren ist *k* nicht länger Primary Key dieser Tabelle.

5.5 lgd_map_literal_history

Wir haben diese Tabelle komplett neu erstellt. Sie dient dazu die Edit-History für Literal-Mappings zu speichern.

Spalten	Typ	Bedingung	Funktion
id	integer	not null	Primärschlüssel zur Identifikation der Änderung des Mappings
k	text	not null	Key des Mappings
property	text	not null	Property des Mappings
language	text	not null	Sprachcode
user_id	text		Benutzername, welcher die Änderung durchgeführt hat
comment	text		Kommentar des Nutzers zur Änderung
timestamp	text	not null	Zeitpunkt der Änderung
action	text		Art der Änderung (Editieren, Löschen, etc.)
userspace	text		zur Identifikation des Userspaces
history_id	text		Verknüpfung mit Wert vor der letzten Änderung des Mappings

5.6 lgd_map_resource_k

Spalten	Typ	Bedingung	Funktion
k	text	not null	
property	text	not null	
object	text	not null	
user_id	text		zur Identifikation des Userspaces
last_history_id	integer		Verknüpfung mit Wert vor der letzten Änderung des Mappings

Wir haben die Spalten *user_id* und *last_history_id* zu dieser Tabelle hinzugefügt.

5.7 lgd_map_resource_k_history

Wir haben diese Tabelle komplett neu erstellt. Sie dient dazu die Edit-History für K-Mappings zu speichern.

Spalten	Typ	Bedingung	Funktion
id	integer	not null	Primärschlüssel zur Identifikation der Änderung des Mappings
k	text	not null	Key des Mappings
property	text	not null	Property des Mappings
object	text	not null	Objekt des Mappings
user_id	text		Benutzername, welcher die Änderung durchgeführt hat
comment	text		Kommentar des Nutzers zur Änderung
timestamp	text	not null	Zeitpunkt der Änderung
action	text		Art der Änderung (Editieren, Löschen, etc.)
userspace	text		zur Identifikation des Userspaces
history_id	text		Verknüpfung mit Wert vor der letzten Änderung des Mappings

5.8 lgd_map_resource_kv

Spalten	Typ	Bedingung	Funktion
k	text	not null	
v	text	not null	
property	text	not null	
object	text	not null	
user_id	text		zur Identifikation des Userspaces
last_history_id	integer		Verknüpfung mit Wert vor der letzten Änderung des Mappings

Wir haben die Spalten *user_id* und *last_history_id* zu dieser Tabelle hinzugefügt.

5.9 lgd_map_resource_kv_history

Wir haben diese Tabelle komplett neu erstellt. Sie dient dazu die Edit-History für KV-Mappings zu speichern.

Spalten	Typ	Bedingung	Funktion
id	integer	not null	Primärschlüssel zur Identifikation der Änderung des Mappings
k	text	not null	Key des Mappings
v	text	not null	Value des Mappings
property	text	not null	Property des Mappings
object	text	not null	Objekt des Mappings
user_id	text		Benutzername, welcher die Änderung durchgeführt hat
comment	text		Kommentar des Nutzers zur Änderung
timestamp	text	not null	Zeitpunkt der Änderung
action	text		Art der Änderung (Editieren, Löschen, etc.)
userspace	text		zur Identifikation des Userspaces
history_id	text		Verknüpfung mit Wert vor der letzten Änderung des Mappings

5.10 lgd_user

Wir haben diese Tabelle komplett neu erstellt. Sie dient dazu die Edit-History für KV-Mappings zu speichern.

Spalten	Typ	Bedingung	Funktion
email	text	not null	E-Mailadresse des Nutzers
username	text		Benutzername des Nutzers
password	text		Passwort des Nutzers
admin	boolean	not null	Flag ob Admin oder nicht
view	text		View der Autovervollständigung