

## BB:Grundbuchdigitalisierung

Das Programm "Grundbuchdigitalisierung", "Digitales Grundbuch" oder kurz "dgb" ist ein Programm, um die Grundbuch-PDF-Dateien massenweise in maschinenlesbare Form zu digitalisieren. Die Daten sind dann im JSON-Format verfügbar und können z.B. in eine Datenbank eingepflegt werden. Das Programm selbst beschäftigt sich nur mit dem Digitalisieren und Prüfen der Grundbuch-Daten, zum Hochladen in LEFIS kann das Tool "[LEFIS-Upload](#)" verwendet werden.

### Inhaltsverzeichnis

1 Installation (für Administratoren) .....	1
2 Starten des Programms .....	2
3 Übersicht der Funktionen .....	2
4 Automatisches Klassifizieren von Seiten .....	3
5 Spalten und Zeilen setzen .....	3
6 Manuelle Bearbeitung der Einträge .....	4
7 Informationen aus Rechtstext auslesen .....	4
8 Funktionen zum Analysieren von Informationen anpassen .....	5
8.1 Text säubern .....	5
8.2 Abkürzungen anpassen .....	6
8.3 Abteilung 2 und 3: Belastete Flurstücke auslesen .....	6
8.4 Abteilung 2: Klassifizierung RechteArt .....	8
8.5 Abteilung 2: Rechtsinhaber auslesen .....	10
8.6 Abteilung 2: Rangvermerk auslesen .....	11
8.7 Abteilung 2: Text kürzen .....	12
8.8 Abteilung 3: Betrag auslesen .....	12
8.9 Abteilung 3: Klassifizierung SchuldenArt .....	13
8.10 Abteilung 3: Rechtsinhaber auslesen .....	14
8.11 Abteilung 3: Text kürzen .....	15
9 Überprüfen der digitalisierten Informationen .....	16
9.1 Überprüfen der Nebenbeteiligten .....	16
9.2 Zuordnen von Nebenbeteiligten-IDs .....	17
9.3 Überprüfen von Rötungen und Rechten .....	18
9.4 Überprüfen von Teilbelastungen .....	18
9.5 Überprüfen von Kurztext und Rangvermerken .....	18
9.6 Ausgeben von Fehlerberichten .....	18
9.7 Überprüfen von Abteilung 1 / Legitimation .....	18
10 Export und Weiterverarbeitung .....	18
11 Datenschema .....	18
11.1 .GBX-Datei (JSON) .....	18
11.2 .LEFIS-Datei (JSON) .....	22
12 Rechtliche Informationen .....	24



### Installation (für Administratoren)

Das Programm (siehe Abbildung A1) läuft derzeit ausschließlich auf Linux und wird als Debian-Paket (.deb) bereitgestellt. Zum Installieren entweder die Datei doppelklicken oder als Administrator:

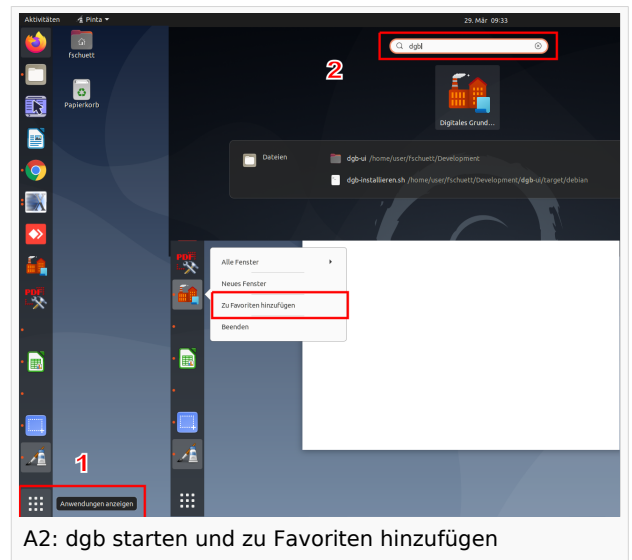
```
sudo dpkg -i ./dgb_1.0.0._amd64.deb
```

eingeben. Alternativ können Administratoren auch `sudo ./dgb-installieren.sh` benutzen, was den gleichen Befehl ausführt.

Das Programm entpackt sich unter `/usr/bin/dgb`, die Konfigurationsdatei ist danach unter `~/config/Konfiguration.json` zu finden. Zum deinstallieren: `sudo apt remove dgb`

Name	Größe	Geändert
 <b>dgb_1.0.0_amd64.deb</b>	2,4 MB	Gestern
 <b>dgb-installieren.sh</b>	201 Bytes	13. Jan

A1: dgb Installationsdateien



## Starten des Programms

Das Programm kann nach der Installation regulär über die GNOME-Suche gestartet werden, ideal ist es, das Programm zu den Favoriten / zur Seitenleiste hinzuzufügen.

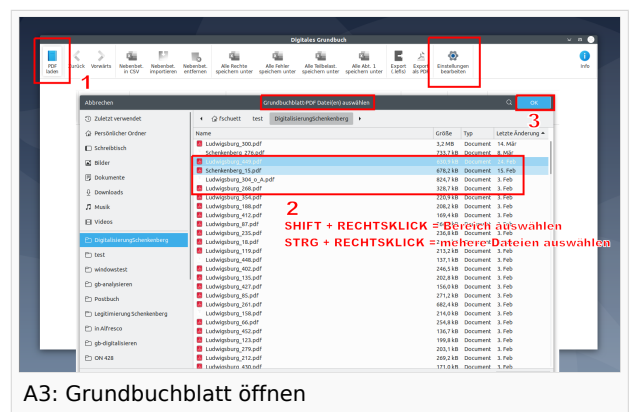
## Übersicht der Funktionen

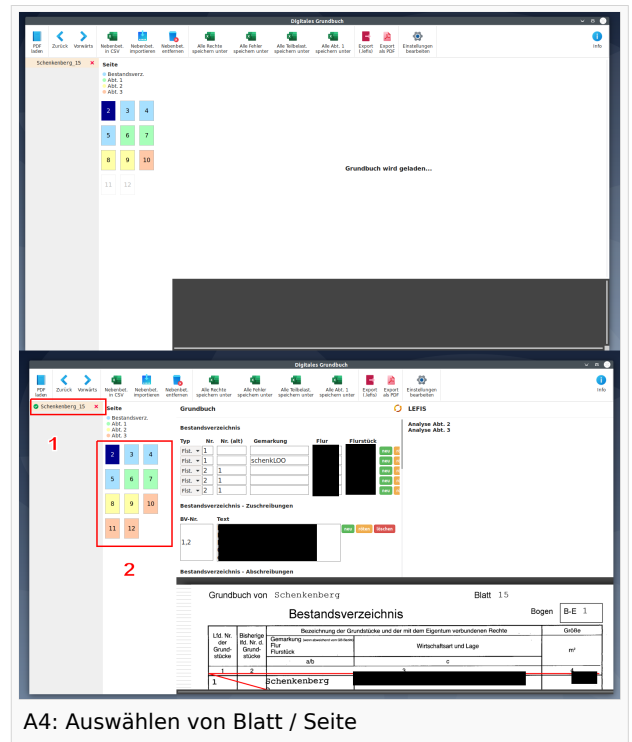
Nach dem Starten sind die meisten Funktionen sichtbar, aber ausgegraut, da zunächst mindestens eine PDF-Datei geöffnet werden muss. In der oberen Leiste können mit Klick auf "PDF laden" eine oder mehrere Grundbuch-PDF-Dateien geladen werden.

Die Funktionen gliedern sich in vier Bereiche:

1. Digitalisierung der Daten (entweder automatisiert oder halbautomatisch)
2. Bearbeitung der digitalisierten Daten
3. Überprüfung mit verschiedenen Ansichten
4. Ausgabe in Dateien (PDF / JSON)

Für jede .pdf-Datei entsteht bei der Bearbeitung eine .gbx-Datei, welche den abdigitalisierten Inhalt des Grundbuchs enthält. Wenn diese Datei nicht vorhanden ist, fängt das Programm nach dem Öffnen der PDF-Datei (siehe Abb. A3) damit an, das Grundbuch zunächst vollautomatisch auszuwerten. Da die vollautomatische Analyse aber nicht perfekt ist, müssen die Daten danach noch einmal manuell überprüft werden.



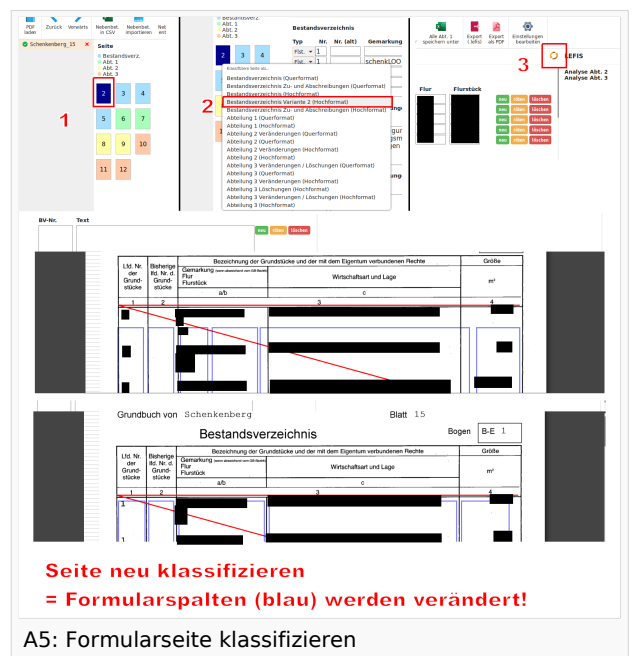


A4: Auswählen von Blatt / Seite

## Automatisches Klassifizieren von Seiten

Beim ersten Laden analysiert das Programm die Seite und versucht, das Formular (Spalten / Zeilen) anhand der Überschrift zu erkennen. In den meisten Fällen läuft das richtig. Anhand der Farbe der Seite in der Seitenauswahl (siehe Abb. A4) erkennt man den Typ der Seite: Bestandsverzeichnis (blau), Abteilung 1 (grün), Abteilung 2 (gelb) oder Abteilung 3 (orange).

Jeder Seitentyp entspricht einem Formular, damit das Programm weiß, "wo" es im PDF nach z.B. der Flurstücksgröße suchen soll. Über "Einstellungen bearbeiten" -> "Spalten ausblenden" kann man die Formelspalten, die das Programm pro Seite annimmt, ein- oder ausblenden. In der unteren Hälfte der Oberfläche ist die Seite zu sehen (siehe Abb. A5).



A5: Formulareseite klassifizieren

## Spalten und Zeilen setzen

Damit nicht jede Zeile / jeder Text manuell eingegeben werden muss, kann das Programm Texte aus der Seite "ausschneiden", per OCR-Schrifterkennung den Text abschreiben / kopieren und in die richtigen Zellen setzen.

Allerdings muss das Programm dafür die Spalten / Zeilen auf der Seite kennen. Per Rechtsklick auf die Seite kann der Typ angepasst werden (vgl. Abb. A6), was zur Folge hat, dass sich die Spaltenanzahl und voreingestellte Breite / Position ändert. Jeder "Seitentyp" entspricht einem voreingestellten Formular. Jede Spalte (blau) kann an den Ecken angefasst und in Größe und Position verändert werden.

Neben der Seite ist ein gestrichelter Rand zu sehen. Wenn der Mauszeiger in diesen Bereich kommt, erscheint eine Vorschau, wo das Programm einen Umbruch zwischen zwei Zeilen einfügen soll. Mit einem Klick kann der Umbruch eingefügt, mit einem Rechtsklick wieder entfernt werden. Sind auf einer Seite keine Umbrüche zu finden, versucht das Programm, die Zeilenumbrüche zu erraten, was allerdings oft misslingt.

Mit einem Klick auf "Grundbuch neu analysieren" (vgl. Abb. "Formular: Zeilen setzen, Spalten anpassen, hier: oben rechts (3)") kann das gesamte Grundbuch neu geladen werden:

**Achtung: Es gehen hierbei alle manuell eingegebenen Daten verloren, da das gesamte Grundbuch neu analysiert wird, d.h. die Zeilen / Spalten neu eingegeben werden.**

## Manuelle Bearbeitung der Einträge

Nachdem die Zellen automatisch gefüllt wurden, muss das nun digitale Grundbuch auf Fehler überprüft werden (vgl. Abb. A7). Gegebenenfalls müssen neue Rechte angelegt (grün), gerötet (gelb) oder gelöscht (rot) werden. Auf der rechten Seite sieht man die Analyse, wie sie in LEFIS übernommen werden wird.

Tastenkürzel für das Bearbeiten der Datensätze sind hierfür:

- STRG + ENTER = neuen Eintrag anlegen (grün)
- STRG + LEERTASTE = Eintrag rötten (gelb)
- STRG + LÖSCHTASTE = Eintrag löschen (rot)

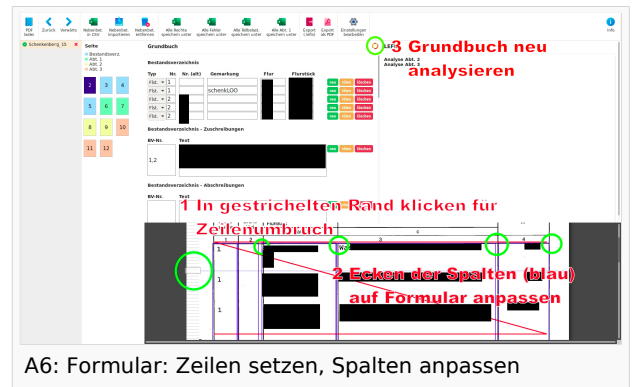
Ebenso kann mit TAB (nächste Zelle) und SHIFT + TAB (vorherige Zelle) in den Zeilen des Grundbuchs navigiert werden.

Die Eingabemaske für das Grundbuch ist relativ selbsterklärend, da sie 1:1 den bestehenden Formularen entspricht - die einzige Ausnahme hier ist das Bestandsverzeichnis: "Flst." steht für ein reguläres Flurstück und "Recht" für einen Herrschvermerk.

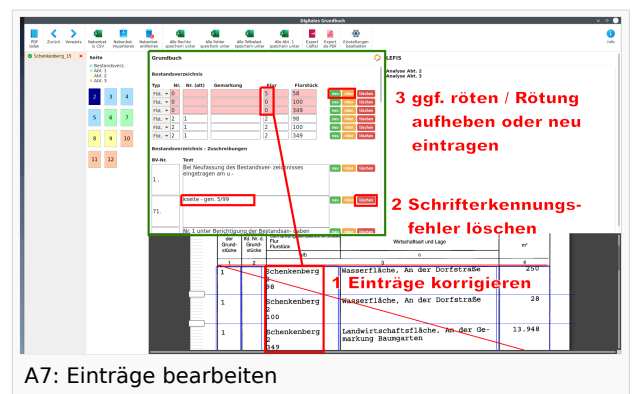
Ein Bearbeiter sollte - Seite für Seite - noch einmal überprüfen, ob die Digitalisierung korrekt war. **Notiz: Bei Seiten, die bereits digital vorhanden waren (d.h. als kopierbarer Text im PDF, nicht als Hintergrundbild), ist das zu fast 99% der Fall, d.h. man kann einfach durch die Seite scrollen und nur die ersten und letzten Einträge überprüfen.**

Um zu vermeiden, dass lange Rechte abgetippt werden, hat das Programm eine interaktive Schrifterkennung: Mit Klicken und Ziehen kann über dem Bild der PDF-Seite ein grünes Rechteck aufgezogen werden - beim Loslassen wird der Text unter dem Rechteck in die Zwischenablage kopiert (siehe Abb. A8).

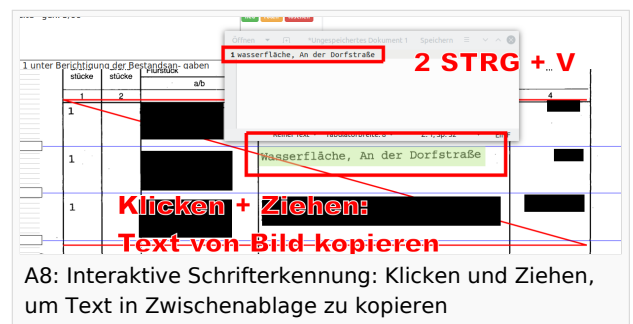
## Informationen aus Rechtstext auslesen



A6: Formular: Zeilen setzen, Spalten anpassen

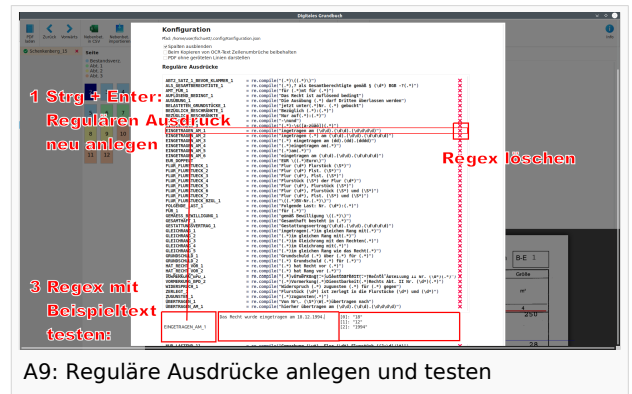


A7: Einträge bearbeiten



A8: Interaktive Schrifterkennung: Klicken und Ziehen, um Text in Zwischenablage zu kopieren

Für die Übernahme in andere Systeme *muss* das Programm wissen, welche Flurstücke mit welchem Recht verbunden werden. Dafür müssen zunächst die Bestandsverzeichnisnummern den Einträgen zugeordnet werden sowie Teilbelastungen mit vielen verschiedenen Formulierungen ausgewertet werden. Um die relevanten Informationen aus dem Text herauszulesen, kann man in der Konfiguration ("Einstellungen bearbeiten") neue reguläre Ausdrücke (regular expressions / RegEx) anlegen (siehe Abb. A9).



A9: Reguläre Ausdrücke anlegen und testen

Die Syntax eines regulären Ausdrucks folgt dem PCRE-Standard; zur Fehlerbehebung (warum bestimmte Informationen nicht erkannt werden) kann sehr gut mit externen Hilfsmitteln wie z.B. dem Debugger von <https://regex101.com/> untersucht werden.

RegEx dient zum Auslesen relevanter Informationen (z.B. Rechteeigentümer, Belastung, Rangvermerk, etc.) - allerdings nicht allein. Jedes Recht (in Abteilung 2 und 3) unterläuft verschiedene Schritte, welche mit Python-Code angepasst werden können. Im Idealfall muss ein Nutzer nichts anpassen, da viele Formulierungen bereits bedacht wurden. In der Konfiguration finden sich verschiedene Masken zum Entwickeln und Testen von Python-Code. Ähnlich wie an einem Fließband durchläuft das Recht verschiedene Funktionen (siehe nächster Abschnitt). Der Sinn dieser Konfiguration ist, dass sie sich im Lauf der Zeit immer weiterentwickelt - und das Programm in Bezug auf das Auswerten von Formulierungen "schlau" wird.

## Funktionen zum Analysieren von Informationen anpassen

### Text säubern

Diese Python-Funktion nimmt den digitalisierten Originaltext ( `recht` ) und normalisiert Zeilenumbrüche. Manchmal kann es beim Kopieren aus dem PDF auch zu komischen Leerzeichen in Worten kommen - hier werden diese Worte normalisiert.

#### Text säubern

```
def text_säubern(recht: String) -> String:
    recht = recht.replace("G r u n d s c h u l d", "Grundschuld")
    recht = recht.replace("o h n e", "ohne")
    recht = recht.replace("B r i e f", "Brief")
    recht = recht.replace("ü b e r", "über")
    recht = recht.replace("f ü r", "für")
    recht = recht.replace("d i e", "die")

    # Worte mit "-" vor diesen Worten sollten nicht zusammengeführt werden
    stop = set(["und", "bzw.", "sowie", "u."])
    recht = " ".join(recht.splitlines())
```

#### Parameter der Funktion:

- `recht` : Der digitalisierte Rohtext des Rechts (von Abteilung 2 / 3)

#### Ausgabe der Funktion:

- der gesäuberte Text

### Beispiele: "Text säubern"

Eingabe	Ausgabe
Mit dem Flur- stück 413 übertragen nach Blatt 300.	Mit dem Flurstück 413 übertragen nach Blatt 300.

Eingabe	Ausgabe
G r u n d s c h u l d ü b e r 400.000 E u r o für die E.ON EDis AG.	Grundschuld über 400.000 Euro für die E.ON EDis AG.

## Abkürzungen anpassen

Nachdem der Text gesäubert wurde, wird der Text in Sätze gebrochen. Beim Schreiben des Kurztexts ist es bei vielen Rechten nur notwendig, den ersten Satz zu beachten. Allerdings kommt das Programm mit einem einfachen `.split` (".") nicht aus, da es sonst Punkte in Abkürzungen wie "Dr." als Satzenden erkennen würde.

### Abkürzungen

```
def abkuerzungen() -> [String]:
    return [
        "Dr",
        "Prof",
        "Co",
        "V",
        "U",
        "E", # "E. ON edis"
        "URNr",
        "Abt",
        "gem",
        "tlw",
    ]
```

### Ausgabe der Funktion:

- eine Liste an Abkürzungen, welche signalisieren, dass der Satz hier noch nicht zu Ende ist

## Abteilung 2 und 3: Belastete Flurstücke auslesen

Diese Funktion ist dafür verantwortlich, die belasteten Flurstücke und Teilbelastungen aus dem Text auszulesen und somit das Recht den Flurstücken zuzuweisen.

### Flurstücke aus Spalte 1 auslesen

```
def flurstuecke_auslesen(spalte_1: String, text: String, re: Mapping[String, Regex]) -> [Spalte1Eintrag]:
    eintraege = []

    spalte_1 = spalte_1.replace("v.", "von")
    spalte_1 = spalte_1.replace("tlw.", "teilweise")
    spalte_1 = spalte_1.replace(",", " ")

    text = re["NUR_LASTEND_AUF_DEM_ANTEIL_1"].replace_all(text, "$1für$3")
    text = re["NUR_LASTEND_AUF_DEM_ANTEIL_2"].replace_all(text, "")
    text = re["NUR_LASTEND_MITEIGENTUMSANTEIL"].replace_all(text, "")
```

Parameter

BV-Nr. (Spalte 1) eingeben

Test Eingabe...

Test Ausgabe der Funktion

### der Funktion:

- `spalte_1`: die Spalte mit den Bestandsverzeichnisnummern
- `text`: der (gesäuberte) Text des Rechts / der Belastung aus Abteilung 2 / 3
- `re`: ein assoziatives Array mit allen `Regex`, indexiert nach `Regex-ID`

### Ausgabe der Funktion:

- eine Liste von `Spalte1Eintrag` - Beispiel:

### Beispiel: Funktion "Flurstück auslesen"

spalte_1	text
	"Nur lastend an Flur 2, Flst. 176 der Gemarkung Ludwigsburg:

spalte_1	text
5	Auflassungsvormerkung für ..."
return	[Spalte1Eintrag(5), Spalte1Eintrag(0).nur_lastend_an([FlurFlurstueck(2, "176", "Ludwigsburg")])]
17, Teil v. 20	"Auflassungsvormerkung für ... bezüglich BV-Nr. 20 nur lastend an Flur 5, Flst. 10/2"
return	[Spalte1Eintrag(17), Spalte1Eintrag(20).nur_lastend_an([FlurFlurstueck(5, "10/2")])]
4-6	"Auflassungsvormerkung für ..."
return	[Spalte1Eintrag(4), Spalte1Eintrag(5), Spalte1Eintrag(6)]

Diese Funktion macht üblicherweise starken Nutzen von regulären Ausdrücken, um die verschiedenen Formulierungen richtig auszulesen. Wenn ein Grundbuch offen ist, wird beim Testen automatisch das Bestandsverzeichnis des offenen Grundbuchs angenommen.

**Hinweis:** Wenn die Flur-Nr. "0" ist, werden alle Fluren nach dem Flurstück gefiltert, wenn die Bestandsverzeichnisnummer "0" ist, werden alle BV-Nummern nach Flur und Flurstück gefiltert:

### Beispiel: Filter der Funktion flurstuecke\_auslesen()

Bestandsverzeichnis	flurstuecke_auslesen()	gefilterte Flurstücke
lfd Nr. 1: Fl. 10, Flst. 11 lfd Nr. 1: Fl. 10, Flst. 12 lfd Nr. 1: Fl. 5, Flst. 13	Spalte1Eintrag(0) .nur_lastend_an([FlurFlurstueck(0, 12)])	lfd Nr. 1: Flur 10, Flurstück 12
lfd Nr. 1: Fl. 10, Flst. 11 lfd Nr. 4: Fl. 5, Flst. 11	Spalte1Eintrag(4) .nur_lastend_an([FlurFlurstueck(0, 11)])	lfd Nr. 4: Fl. 5, Flst. 11
lfd Nr. 1: Fl. 10, Flst. 11 lfd Nr. 4: Fl. 5, Flst. 11	Spalte1Eintrag(4) .nur_lastend_an([FlurFlurstueck(0, 11)])	lfd Nr. 1: Fl. 10, Flst. 11 lfd Nr. 4: Fl. 5, Flst. 11

## Abteilung 2: Klassifizierung RechteArt

Diese Funktion liest die `RechteArt` aus dem Text aus. Die verfügbaren `RechteArt`-Typen sind in der Auswahlbox unter der Überschrift aufgelistet, beim Ändern der Auswahl wird der `RechteArt`-Typ in die Zwischenablage kopiert.

### Klassifizierung RechteArt (Abteilung 2)

Abwasserleitungsrecht

```
def klassifiziere_rechteart_abt2(saetze: [String], re: Mapping[String, Regex]) -> RechteArt:
    recht = saetze[0]
    leitungsrecht = [
        "Leitungsrecht",
        "Leitungs-",
        "Leitungssystemrecht",
        "Trinkwasserleitung",
        "Leitungstrassenrecht",
        "Leitungsrecht",
        "Ferngasleitungen",
        "Kabelanlagenrecht",
        "-Leitungs-",
    ]
```

Test Eingabe...

Test Ausgabe der Funktion

### Parameter der Funktion:

- `saetze` : Liste an (gesäuberten) Sätzen des Rechtsabteilung 2 / 3
- `re` : ein assoziatives Array mit allen `Regex`, indexiert nach `Regex-ID`

### Ausgabe der Funktion:

- ist eine `RechteArt`, wobei die `RechteArt` folgende Werte annehmen kann:
  - `RechteArt.SpeziellVormerkung(x)` : nur bei Rechten "mit dem Inhalt des Rechts x"
  - `RechteArt.Abwasserleitungsrecht`
  - `RechteArt.AuflassungsVormerkung`
  - `RechteArt.Ausbeutungsrecht`
  - `RechteArt.AusschlussDerAufhebungDerGemeinschaftGem1010BGB`
  - `RechteArt.Baubeschränkung`
  - `RechteArt.Bebauungsverbot`
  - `RechteArt.Benutzungsrecht`
  - `RechteArt.BenutzungsregelungGem1010BGB`
  - `RechteArt.Bepflanzungsverbot`
  - `RechteArt.Bergschadenverzicht`
  - `RechteArt.Betretungsrecht`
  - `RechteArt.Bewässerungsrecht`
  - `RechteArt.BpD`
  - `RechteArt.BesitzrechtNachEGBGB`
  - `RechteArt.BohrUndSchuerfrecht`
  - `RechteArt.Brunnenrecht`
  - `RechteArt.Denkmalerschutz`
  - `RechteArt.DinglichesNutzungsrecht`
  - `RechteArt.DuldungVonEinwirkungenDurchBaumwurf`
  - `RechteArt.DuldungVonFernmeldeanlagen`
  - `RechteArt.Durchleitungsrecht`
  - `RechteArt.EinsitzInsitzrecht`
  - `RechteArt.Entwasserungsrecht`
  - `RechteArt.Erbbaurecht`
  - `RechteArt.Erwerbsvormerkung`
  - `RechteArt.Fensterrecht`
  - `RechteArt.Fensterverbot`



- RechteArt.Fischereirecht
- RechteArt.Garagenrecht
- RechteArt.Gartenbenutzungsrecht
- RechteArt.GasleitungGasreglerstationFerngasltg
- RechteArt.GehWegeFahrOderLeitungsrecht
- RechteArt.Gewerbebetriebsbeschränkung
- RechteArt.GewerblichesBenutzungsrecht
- RechteArt.Grenzbebauungsrecht
- RechteArt.Grunddienstbarkeit
- RechteArt.Hochspannungsleitungsrecht
- RechteArt.Immissionsduldungsverpflichtung
- RechteArt.Insolvenzvermerk
- RechteArt.Kabelrecht
- RechteArt.Kanalrecht
- RechteArt.Kiesabbauberechtigung
- RechteArt.Kraftfahrzeugabstellrecht
- RechteArt.LeibgedingAltenteilsrechtAuszugsrecht
- RechteArt.LeitungsOderAnlagenrecht
- RechteArt.Mauerrecht
- RechteArt.Mitbenutzungsrecht
- RechteArt.Mobilfunkstationsrecht
- RechteArt.Muehlenrecht
- RechteArt.Mulltonnenabstellrecht
- RechteArt.Nacherbenvermerk
- RechteArt.Niessbrauchrecht
- RechteArt.Nutzungsbeschränkung
- RechteArt.Pfandung
- RechteArt.Photovoltaikanlagenrecht
- RechteArt.Pumpenrecht
- RechteArt.Reallast
- RechteArt.RegelungUeberDieHöheDerNotwegrenteGemaess912Bgb
- RechteArt.RegelungUeberDieHöheDerUeberbaurenteGemaess912Bgb
- RechteArt.Rueckauflassungsvormerkung
- RechteArt.Ruckerwerbsvormerkung
- RechteArt.Sanierungsvermerk
- RechteArt.Schachtrecht
- RechteArt.SonstigeDabagrechteart
- RechteArt.SonstigeRechte
- RechteArt.Tankstellenrecht
- RechteArt.Testamentsvollstreckervermerk
- RechteArt.Transformatorenrecht
- RechteArt.Ueberbaurecht
- RechteArt.UebernahmeVonAbstandsflächen
- RechteArt.Umlegungsvermerk
- RechteArt.Umspannanlagenrecht
- RechteArt.Untererbbaurecht
- RechteArt.VerausserungsBelastungsverbot
- RechteArt.Verfuegungsverbot
- RechteArt.VerwaltungsUndBenutzungsregelung
- RechteArt.VerwaltungsregelungGem1010Bgb
- RechteArt.VerzichtAufNotwegerente

- RechteArt.VerzichtAufUeberbaurente
- RechteArt.Viehtrankerecht
- RechteArt.Viehtreibrecht
- RechteArt.Vorkaufsrecht
- RechteArt.Wasseraufnahmeverpflichtung
- RechteArt.Wasserentnahmerecht
- RechteArt.Weiderecht
- RechteArt.Widerspruch
- RechteArt.Windkraftanlagenrecht
- RechteArt.Wohnrecht
- RechteArt.WohnungsOderMitbenutzungsrecht
- RechteArt.Wohnungsbelegungsrecht
- RechteArt.WohnungsrechtNach1093Bgb
- RechteArt.Zaunerrichtungsverbot
- RechteArt.Zaunrecht
- RechteArt.Zustimmungsvorbehalt
- RechteArt.Zwangsversteigerungsvermerk
- RechteArt.Zwangsverwaltungsvermerk

### Beispiel: Funktion "RechteArt klassifizieren (Abteilung 2)"

Eingabe	Ausgabe
Trinkwasserleitungsrecht für ...	return RechteArt. LeitungsOderAnlagenrecht
Recht über das Betreiben einer Windkraftanlage für ...	return RechteArt. Windkraftanlagenrecht
Auflassung mit dem Inhalt des Rechts Abt. 2 Nr. 5 für einen noch zu benennenden Dritten...	return RechteArt. SpeziellVormerkung(5)
Übergabestationsrecht für ...	return RechteArt. SonstigeDabagrechteart

### Abteilung 2: Rechtsinhaber auslesen

Diese Funktion liest den Rechtsinhaber aus einem Recht der Abteilung 2 aus - ein leerer Rechtsinhaber ist nur bei Bodenreformvermerken erlaubt, da diese Rechte von Natur aus keine Person als Inhaber haben.

#### Rechtsinhaber auslesen (Abteilung 2)

```
def rechtsinhaber_auslesen_abt2(saetze: [String], re: Mapping[String, Regex], recht_id: String) -> String:
    recht = saetze[0].replace("für die Dauer", "", 1)
    inhaber = None

    if re["ZUGUNSTEN_1"].matches(recht):
        # "Widerspruch für XXX zugunsten YYY" - YYY ist der Rechteinhaber, nicht XXX
        inhaber = re["ZUGUNSTEN_1"].find_in(recht, 1)
        # "[RechteArt] für das Landesamt für XXX"
    elif "amt für" in recht.lower():
        inhaber = re["AMT_FÜR_1"].find_in(recht, 0) + "mt für " + re["AMT_FÜR_1"].find_in(recht, 1)
    else:
        pass
```

Test Eingabe...

Test Ausgabe der Funktion

Parameter der Funktion:

- `saetze` : die Sätze des Rechts der Abteilung 2
- `re` : ein assoziatives Array mit allen RegEx, indexiert nach RegEx-ID
- `recht_id` : Bei dem Inhaber "für noch einen zu benennenden Dritten" ist immer die ID des Rechts ("Ludwigsburg Blatt X, Abteilung 2 Recht Y") anzuhängen, damit die Inhaber verschiedener Rechte später separate Ordnungsnummern erhalten.

Ausgabe der Funktion:

- der Rechtsinhaber des Rechts - dieser entspricht einer Nebenbeteiligten-Ordnungsnummer und wird so später in LEFIS übernommen

**Beispiel: Funktion "Rechtsinhaber auslesen (Abteilung 2)"**

Eingabe	Ausgabe
BpD (Leitungs- und Anlagenrecht für die 220 kV-Leitung Neuenhagen-Pasewalk 303/304 /305/306) für die Vattenfall Europe Transmission GmbH, Berlin.	Vattenfall Europe Transmission GmbH, Berlin
Leitungsrecht nebst Bau- und Einwirkungsbeschränkung für einen von der Berechtigten noch zu benennenden Dritten der den Gestattungsvertrag übernimmt zugunsten des Berechtigten die Infrastrukturgesellschaft Schmölln GmbH & Co. KG, Wörrstadt, HRA 42676, Amtsgericht Mainz.	Infrastrukturgesellschaft Schmölln GmbH & Co. KG, Wörrstadt, HRA 42676, Amtsgericht Mainz

**Abteilung 2: Rangvermerk auslesen**

Diese Funktion liest den Rangvermerk aus Rechten der Abteilung 2 aus.

**Rangvermerk auslesen (Abteilung 2)**

```
def rangvermerk_auslesen_abt2(saetze: [String], re: Mapping[String, Regex]) -> String:
    recht = ". ".join(saetze)
    if not("Gleichrang" in recht or "Rang" in recht):
        return ""
    rvm = []
    for s in saetze:
        s = s.rstrip().rstrip()
        if "hat Rang vor" in s:
```

Test Eingabe...

Test Ausgabe der Funktion

Parameter der Funktion:

- `saetze` : die Sätze des Rechts der Abteilung 2
- `re` : ein assoziatives Array mit allen RegEx, indexiert nach RegEx-ID

#### Ausgabe der Funktion:

- der Rangvermerk oder "", wenn kein Rangvermerk vorhanden ist

#### Beispiel: Funktion "Rangvermerk auslesen (Abteilung 2)"

Eingabe	Ausgabe
BpD (Kabeltrassenleitungsrecht) für die ENERTRAG Netzinfrastruktur GmbH, Dauerthal im Gleichrang mit Abt. II Nr. 26.	im Gleichrang mit Abt. II Nr. 26

#### Abteilung 2: Text kürzen

Diese Funktion schreibt den Text eines Rechts der Abteilung 2 so um, wie er in LEFIS übernommen wird. Üblicherweise bedeutet das, dass nur der erste Satz übernommen und gekürzt wird.

##### Text kürzen (Abteilung 2)

```
def text_kuerzen_abt2(saetze: [String], rechtsinhaber: String, rangvermerk: String, re: Mapping[String, Regex]) -> String:
    erster_satz = saetze[0]
    erster_satz = erster_satz.rstrip().rstrip()

    if re["BEZÜGLICH BESCHRÄNKTE 1"].matches(erster_satz):
        erster_satz = re["BEZÜGLICH BESCHRÄNKTE 1"].replace_all(erster_satz, "$2")
    if re["BEZÜGLICH BESCHRÄNKTE 2"].matches(erster_satz):
        erster_satz = re["BEZÜGLICH BESCHRÄNKTE 2"].replace_all(erster_satz, "$2")
    if re["NUR_LASTEND 58"].matches(erster_satz):
        erster_satz = re["NUR_LASTEND 58"].replace_all(erster_satz, "")

    Test Eingabe...
    Test Ausgabe der Funktion text_kuerzen_abt2()
```

#### Parameter der Funktion:

- saetze : die Sätze des Rechts der Abteilung 2
- rechtsinhaber : der Rechtsinhaber, ausgelesen von rechtsinhaber\_auslesen\_abt2() (siehe oben)
- rangvermerk : der Rangvermerk, ausgelesen von rangvermerk\_auslesen\_abt2() (siehe oben)
- re : ein assoziatives Array mit allen RegEx, indexiert nach RegEx-ID

#### Ausgabe der Funktion:

- der Rechtstext, so wie er später in LEFIS (Alter Bestand) übernommen werden soll

#### Beispiel: Funktion "Text kürzen (Abteilung 2)"

Eingabe	Ausgabe
Beschränkte persönliche Dienstbarkeit (Kabel- und Leitungsrecht verbunden mit Bau- und Einwirkungsbeschränkungen), für die ENERTRAG Netz GmbH, Dauerthal. Gemäß Bewilligung vom 10.02.1990 eingetragen am 15.02.1990.	BpD (Kabel- und Leitungsrecht verbunden mit Bau- und Einwirkungsbeschränkungen) für die ENERTRAG Netz GmbH, Dauerthal.

#### Abteilung 3: Betrag auslesen

Diese Funktion liest den Betrag der Schuld aus der Spalte 2 der Abteilung 3 sowie dem Text aus.

### Betrag auslesen (Abteilung 3)

```
def betrag_auslesen(saetze: [String], re: Mapping[String, Regex]) -> Betrag:
    recht = saetze[0]

    waehrungen = [
        ('EUR', Waehrung.Euro),
        ('Euro', Waehrung.Euro),
        ('€', Waehrung.Euro),
        ('Deutsche Mark', Waehrung.DMark),
        ('DM', Waehrung.DMark),
        ('Mark', Waehrung.MarkDDR),
        ('M', Waehrung.MarkDDR).
```

Test Eingabe...

Test Ausgabe der Funktion

#### Parameter der Funktion:

- `saetze` : die Sätze der Belastung der Abteilung 3
- `betrag` : die Spalte "Betrag" der Belastung
- `re` : ein assoziatives Array mit allen RegEx, indexiert nach RegEx-ID

#### Ausgabe der Funktion:

- ein `Betrag` -Objekt, was mit `Betrag(vor_komma, nach_komma, waehrung)` konstruiert werden kann, wobei die Währung einer der folgenden Werte ist:
  - `Waehrung.Euro`
  - `Waehrung.DMark`
  - `Waehrung.MarkDDR`
  - `Waehrung.Goldmark`
  - `Waehrung.Rentenmark`
  - `Waehrung.Reichsmark`
  - `Waehrung.GrammFeingold`

### Beispiel: Funktion "Betrag auslesen (Abteilung 3)"

Eingabe	Ausgabe
1928,72 €	return Betrag(1928, 72, Waehrung.Euro)
300.000,00 M	return Betrag(300000, 0, Waehrung.MarkDDR)

### Abteilung 3: Klassifizierung SchuldenArt

Diese Funktion liest die `SchuldenArt` aus dem Text aus. Die verfügbaren `SchuldenArt` -Typen sind in der Auswahlbox unter der Überschrift aufgelistet, beim Ändern der Auswahl wird der `SchuldenArt` -Typ in die Zwischenablage kopiert.

### Klassifizierung SchuldenArt (Abteilung 3)

Grundschild

```
def klassifiziere_schuldenart_abt3(saetze: [String], re: Mapping[String, Regex]) -> SchuldenArt:
    recht = ". ".join(saetze)
    stichwoerter = [
        ('Gesamtgrundschild', SchuldenArt.Grundschild),
        ('Grundschild', SchuldenArt.Grundschild),
        ('grundschild', SchuldenArt.Grundschild),
        ('Hypothek', SchuldenArt.Hypothek),
        ('Rentenschuld', SchuldenArt.Rentenschuld),
        ('Aufbauhypothek', SchuldenArt.Aufbauhypothek),
        ('Sicherungshypothek', SchuldenArt.Sicherungshypothek).
    ]
    Test Eingabe...
    Test Ausgabe der Funktion
```

#### Parameter der Funktion:

- saetze : die Sätze des Rechts der Abteilung 3
- re : ein assoziatives Array mit allen RegEx, indexiert nach RegEx-ID

#### Ausgabe der Funktion:

- eine SchuldenArt, wobei die SchuldenArt folgende Werte annehmen kann:
  - SchuldenArt.Grundschild
  - SchuldenArt.Hypothek
  - SchuldenArt.Rentenschuld
  - SchuldenArt.Aufbauhypothek
  - SchuldenArt.Sicherungshypothek
  - SchuldenArt.Widerspruch
  - SchuldenArt.Arresthypothek
  - SchuldenArt.SicherungshypothekGem128ZVG
  - SchuldenArt.Hoehstbetragshypothek
  - SchuldenArt.Sicherungsgrundschild
  - SchuldenArt.Zwangssicherungshypothek
  - SchuldenArt.NichtDefiniert

### Beispiel: Funktion "SchuldenArt auslesen (Abteilung 3)"

Eingabe	Ausgabe
Sicherungshypothek gemäß §128 ZVG für ...	return SchuldenArt.SicherungshypothekGem128ZVG
Grundschild über 300.000 € für ...	return SchuldenArt.Grundschild

### Abteilung 3: Rechtsinhaber auslesen

Diese Funktion liest den Rechtsinhaber aus einem Recht der Abteilung 3 aus - ein leerer Rechtsinhaber ist nicht erlaubt.

### Rechtsinhaber auslesen (Abteilung 3)

```
def rechtsinhaber_abt3(saetze: [String], re: Mapping[String, Regex], recht_id: String) -> String:
    recht = saetze[0].replace("für die Dauer", "", 1)
    inhaber = None

    if re["ZUGUNSTEN_1"].matches(recht):
        # "Widerspruch für XXX zugunsten YYY" - YYY ist der Rechteinhaber, nicht XXX
        inhaber = re["ZUGUNSTEN_1"].find_in(recht, 1)
        # "[RechteArt] für das Landesamt für XXX"
    elif "amt für" in recht.lower():
        inhaber = re["AMT_FÜR_1"].find_in(recht, 0) + "mt für " + re["AMT_FÜR_1"].find_in(recht, 1)
    else:
        Test Eingabe...
```

Test Ausgabe der Funktion

#### Parameter der Funktion:

- **saetze** : die Sätze des Rechts der Abteilung 2
- **re** : ein assoziatives Array mit allen RegEx, indexiert nach RegEx-ID
- **recht\_id** : Bei dem Inhaber "für noch einen zu benennenden Dritten" ist immer die ID des Rechts ("Ludwigsburg Blatt X, Abteilung 2 Recht Y") anzuhängen, damit die Inhaber verschiedener Rechte später separate Ordnungsnummern erhalten.

#### Ausgabe der Funktion:

- der Rechtsinhaber des Rechts - dieser entspricht einer Nebenbeteiligten-Ordnungsnummer und wird so später in LEFIS übernommen.

### Beispiel: Funktion "Rechtsinhaber auslesen (Abteilung 3)"

Eingabe	Ausgabe
Grundsschuld ohne Brief über z w a n z i g t a u s e n d E u r o mit 15 % Zinsen für die Bausparkasse Schwäbisch Hall Aktiengesellschaft - Bausparkasse der Volksbanken und Raiffeisenbanken -, Schwäbisch Hall.	Bausparkasse Schwäbisch Hall Aktiengesellschaft - Bausparkasse der Volksbanken und Raiffeisenbanken -, Schwäbisch Hall

### Abteilung 3: Text kürzen

Diese Funktion schreibt den Text eines Rechts der Abteilung 3 so um, wie er in LEFIS übernommen wird. Üblicherweise bedeutet das, dass nur Betrag, Schuldenart und Rechtsinhaber übernommen werden.

#### Text kürzen (Abteilung 3)

```
def text_kuerzen_abt3(saetze: [String], betrag: String, schuldenart: String, rechtsinhaber: String, re:
Mapping[String, Regex]) -> String:

    gesamthaft = []
    for s in saetze:
        if "Gesamthaft" in s:
            gesamthaft.append(s)

    if schuldenart == "" or betrag == "":
        return ""

    gekuerzt = schuldenart + " über " + betrag + " für " + rechtsinhaber
    Test Eingabe...
```

Test Ausgabe der Funktion text\_kuerzen\_abt3()

#### Parameter der Funktion:

- **saetze** : die Sätze der Belastung in Abteilung 3

- `betrag` : der Betrag, ausgelesen von `betrag_auslesen()` (siehe oben)
- `schuldenart` : die SchuldenArt, ausgelesen von `schuldenart_auslesen_abt3()` (siehe oben)
- `rechtsinhaber` : der Rechtsinhaber, ausgelesen von `rechtsinhaber_auslesen_abt3()` (siehe oben)
- `re` : ein assoziatives Array mit allen RegEx, indexiert nach RegEx-ID

#### Ausgabe der Funktion:

- der Kurztext der Belastung, so wie er später in LEFIS (Alter Bestand) übernommen werden soll

### Beispiel: Funktion "Text kürzen (Abteilung 3)"

Eingabe	Ausgabe
Grundschild ohne Brief über z w a n z i g t a u s e n d E u r o mit 15 % Zinsen für die Bausparkasse Schwäbisch Hall Aktiengesellschaft - Bausparkasse der Volksbanken und Raiffeisenbanken -, Schwäbisch Hall. Gemäß Bewilligung vom 10.02.2003 (UR-Nr. 561/2003, Not.verwalter Max Mustermann in Berlin) eingetragen am 20.02.2003.	Grundschild über 20.000,00 € für Bausparkasse Schwäbisch Hall Aktiengesellschaft - Bausparkasse der Volksbanken und Raiffeisenbanken -, Schwäbisch Hall.

### Überprüfen der digitalisierten Informationen

Nachdem die Informationen digitalisiert und ausgelesen wurden, kann es trotz alledem vorkommen, dass sich Fehler eingeschlichen haben. Um diese Fehler zu erkennen, muss das Grundbuch nach der Digitalisierung und Analyse noch einmal mit dem Original verglichen werden. Ebenso müssen verfahrensbezogene Nebenbeteiligten-IDs (Stamm- / Unternummern) vergeben werden, um die Nebenbeteiligten in LEFIS einzupflegen.

Für die folgenden Schritte sollten zunächst alle Grundbuchblätter des Verfahrens geöffnet werden. Um eine Liste der Grundbuchblätter in einem Verfahren zu erhalten, folgen Sie bitte der Anleitung im Artikel [LefisUpload- AlleGrundbuchblätterImVerfahrenInListeAusgeben](#).

### Überprüfen der Nebenbeteiligten

Mit der Funktion "Nebenbeteiligte in CSV" können alle Rechtsinhaber (= Nebenbeteiligte) des gesamten Verfahrens in eine .tsv (tab-separated-values)-Datei exportiert werden (siehe Abb. A9). Die Spalten dieser Datei enthalten:

- `ORDNUNGSNUMMER` : die Stammnummer des Beteiligten - hier kann entweder eine existierende oder eine neue Nummer eingetragen werden.
- `TYP` : der "Typ" des Beteiligten: Beim Importieren der Nebenbeteiligten können Ordnungsnummern automatisch vergeben werden, *aber nur wenn die Spalte `ORDNUNGSNUMMER` leer ist*.
  - `OEFFENTLICH` : es wird eine Ordnungsnummer mit `810*` vergeben, z.B. `8100001`
  - `BANK` : es wird eine Ordnungsnummer mit `812*` vergeben, z.B. `8120001`
  - `AGRAR` : es wird eine Ordnungsnummer mit `813*` vergeben, z.B. `8130001`

A9: Nebenbeteiligte aus Verfahren in CSV-Datei exportieren



- **PRIVAT** : es wird eine Ordnungsnummer mit 814\* vergeben, z.B. 8140001
- **JEW-EIGENT** : es wird eine Ordnungsnummer mit 815\* vergeben, z.B. 8150001
- **LEITUNG** : es wird eine Ordnungsnummer mit 817\* vergeben, z.B. 8170001
- **GMBH** : es wird eine Ordnungsnummer mit 819\* vergeben, z.B. 8190001
- **<leer>**: es wird keine Ordnungsnummer automatisch vergeben
- **RECHT** : die Rechte, bei dem der Nebenbeteiligte auftaucht
- **NAME (GRUNDBUCH)** : Name (Text) des Nebenbeteiligten im Grundbuch
- **ANREDE** : LEFIS-Attribut: wird beim Hochladen in LEFIS berücksichtigt.
  - **HERR**
  - **FRAU**
  - **FIRMA**
- **TITEL** : LEFIS-Attribut: wird beim Hochladen in LEFIS berücksichtigt
- **VORNAME** : LEFIS-Attribut: wird beim Hochladen in LEFIS berücksichtigt
- **NACHNAME\_ODER\_FIRMA** : LEFIS-Attribut: wird beim Hochladen in LEFIS berücksichtigt - wenn diese Spalte leer ist, wird anstelle dessen die Spalte **NAME (GRUNDBUCH)** übernommen
- **GEBURTSNAME** : LEFIS-Attribut: wird beim Hochladen in LEFIS berücksichtigt
- **GEBURTSDATUM** : Geburtsdatum im Format DD.MM.YYYY, wird beim Hochladen in LEFIS berücksichtigt
- **WOHNORT** : LEFIS-Attribut, wird beim Hochladen in LEFIS berücksichtigt

In der Tabelle kann man relativ einfach die Schreibweisen aller Nebenbeteiligten überprüfen. Da auch verschiedene Formulierungen desselben Inhabers verschiedene Ordnungsnummern erhalten, ist es in Ordnung, wenn ein Name mehrmals mit verschiedenen Schreibweisen vorhanden ist.

Fehler kann man hier schnell erkennen, das zugehörige Recht in der Spalte **RECHT** finden, das Grundbuch öffnen und den Fehler überprüfen / korrigieren.

## Zuordnen von Nebenbeteiligten-IDs

Solange noch keine Nebenbeteiligten-IDs geladen sind, wird die Warnung "Konnte keine Ordnungsnummer finden" unter dem Recht in der Analyse angezeigt. Um diese Warnung aufzuheben, müssen die soeben erstellten Ordnungsnummern zugewiesen werden, was automatisch bei einer Übereinstimmung des Rechtsinhabers mit dem Eintrag in der Spalte **NAME (GRUNDBUCH)** geschieht. Mit der Funktion "Nebenbeteiligte entfernen" können zunächst alte Nebenbeteiligten-IDs entfernt werden, danach können neue Nebenbeteiligte mit "Nebenbeteiligte importieren" neu aus der .tsv-Datei importiert werden (siehe Abb. A10). Dabei können automatisch Ordnungsnummern basierend auf dem Typ (siehe voriger Abschnitt) vergeben werden.

Wenn das Grundbuch keine Fehler aufweist und alle Rechtsinhaber im Grundbuch eine Ordnungsnummer haben, ändert sich die Farbe des Häkchens in der Spalte der offenen Dateien von hellgrün auf dunkelgrün. Falls noch Fehler in der Datei vorhanden sind, wird anstelle dessen ein gelbes Warn-Dreieck angezeigt. Nach der Zuweisung wird die neue Ordnungsnummer vor dem Namen des Rechtsinhabers angezeigt.

A10: Nebenbeteiligte / Rechtsinhaber löschen und neu importieren

## Überprüfen von Rötungen und Rechten

Um Rötungen zu überprüfen, ist es am besten, sich das gesamte Verfahren noch einmal auszudrucken und - auf Papier - mit dem Original-Grundbuch zu vergleichen. Mit der Funktion "Export als PDF" können alle momentan geladenen Grundbuchblätter in ein PDF ausgegeben werden, sodass man nicht jedes Grundbuchblatt einzeln drucken muss.

## Überprüfen von Teilbelastungen

Da die Funktion "Abteilung 2 und 3: Belastete Flurstücke auslesen" viele verschiedenen Formulierungen ("nur lastend an ...", "lastend nur an ...", "bezüglich ... lastet an ...") auswerten muss, kann es sein, dass verschiedene Belastungen nicht richtig erfasst wurden. Um die Belastungen zu überprüfen, kann man über die Funktion "Alle Teilbelastungen speichern unter" alle Rechte im Verfahren, die eine Teilbelastung haben, in eine .html-Datei ausgeben und überprüfen.

## Überprüfen von Kurztext und Rangvermerken

Genauso wie die Teilbelastungen können auch alle Rechte (im gesamten Verfahren) in eine .html-Datei gespeichert werden. In dieser Übersicht sieht man sowohl den Original-Text als auch den gekürzten Text und den ausgelesenen Rangvermerk.

## Ausgeben von Fehlerberichten

Bei schwerwiegenden Fehlern (z.B. ein Recht referenziert ein Flurstück, das nicht im Bestandsverzeichnis vorhanden ist), sowie `Exceptions` in den Funktionen der Konfiguration, zeigt das Programm einen Fehler (in rot) unter dem Recht an und ändert das Icon in der ersten Spalte auf ein gelbes Warn-Dreieck. Um eine Übersicht aller noch zu korrigierenden Fehler im Verfahren zu erhalten, kann man die Funktion "alle Fehler speichern unter" benutzen, um die Fehler in eine .html-Datei auszugeben.

## Überprüfen von Abteilung 1 / Legitimation

Die Abteilung 1 ist wichtig, um später Eigentümer über das Grundbuch zu legitimieren. Mit der Funktion "alle Abt. 1 speichern unter" werden alle Einträge der geladenen Dateien in eine .tsv-Tabelle ausgegeben.

## Export und Weiterverarbeitung

Wenn alle Daten bereinigt, geprüft und Ordnungsnummern zugewiesen sind, kann das Endergebnis aller Grundbuchblätter in eine ".lefis"-JSON-Datei ausgegeben werden. Diese Datei enthält nicht mehr die Original-Daten, sondern nur noch ein Abbild der ausgelesenen und ausgewerteten Informationen und kann vom Programm [LEFIS-Upload](#) in die DHK eingespielt oder von anderen externen Programmen übernommen werden.

## Datenschema

### .GBX-Datei (JSON)

Da einige Objekte mehrmals auftauchen, sind sie hier vor dem Erscheinen im Grundbuch-JSON aufgelistet:

#### **StringOderMultiLine:**

Wenn der Text nur eine Zeile enthält, ist dieses Objekt gleich String. Bei mehrzeiligen Texten ist ein Array[String] vorhanden, um Probleme mit Zeilenendungen zwischen verschiedenen Betriebssystemen zu umgehen.

#### **PositionInPdf: Objekt**

- `seite`: Integer: Seitenzahl, auf der die Bestandsverzeichnis-Abschreibung gefunden wurde

- **rect** : Objekt: Wo steht der Text auf der Seite (von oberer linker Ecke, in mm)?:
  - **min\_x** : Integer
  - **max\_x** : Integer
  - **min\_y** : Integer
  - **max\_y** : Integer

### BestandsverzeichnisEintrag: Objekt

Das Objekt kann entweder ein Flurstück oder ein grundstücksgleiches Recht beschreiben.

#### Flurstück:

- **lfd\_nr** : Integer
- **bisherige\_lfd\_nr** : Optional[Integer]
- **flur** : Integer
- **flurstueck** : String
- **gemarkung** : Optional[StringOderMultiLine]
- **bezeichnung** : StringOderMultiLine
- **groesse** : Objekt: entweder mit "typ: m" oder "typ: ha"
  - **typ** : String:
  - **wert** : Objekt:
    - wenn **typ** = "m":
      - **m2** : Optional[Integer]: Quadratmeter Fläche
    - wenn **typ** = "ha":
      - **ha** : Optional[Integer]: Hektar Fläche
      - **a** : Optional[Integer]: Ar Fläche
      - **m2** : Optional[Integer]: Quadratmeter Fläche
- **automatisch\_geroetet** : Optional[Boolean]: ob der Eintrag vom Programm als gerötet erkannt wurde
- **manuell\_geroetet** : Optional[Boolean]: ob der Eintrag vom Nutzer gerötet wurde, überschreibt "automatisch\_geroetet"
- **position\_in\_pdf** : Optional[PositionInPdf]: Seite und Rechteck in mm, wo die Schrifterkennung das Objekt erkannt hat

#### Recht / Herrschvermerk:

- **lfd\_nr** : Integer: laufende Nummer in Bestandsverzeichnis Spalte 1
- **zu\_nr** : StringOderMultiLine: "gehört zu lfd. Nr." - bei Herrschvermerken in Bestandsverzeichnis Spalte 2 oder 1
- **bisherige\_lfd\_nr** : Optional[Integer]: bei fortgeführtem HVM die bisherige lfd. Nr.
- **text** : StringOderMultiLine: Text des Herrschvermerks
- **automatisch\_geroetet** : Optional[Boolean]: ob der Eintrag vom Programm als gerötet erkannt wurde
- **manuell\_geroetet** : Optional[Boolean]: ob der Eintrag vom Nutzer gerötet wurde, überschreibt "automatisch\_geroetet"
- **position\_in\_pdf** : Optional[PositionInPdf]: Seite und Rechteck in mm, wo die Schrifterkennung das Objekt erkannt hat

### Grundbuch: Objekt

- **datei** : String: Dateipfad zu der PDF-Datei, aus der dieses Grundbuchblatt entstanden ist
- **titelblatt** : Objekt:
  - **amtsgerecht** : String: Amtsgericht (von der Titelseite)
  - **grundbuch\_von** : String: Name des Grundbuchblatts / Grundbuchblattbezirks
  - **blatt** : Integer: Nummer des Grundbuchblatts
- **seitenzahlen** : Liste[Integer]: Liste aller PDF-Seitennummern

- **geladen** : Map[String: Objekt]: Liste aller automatisch digitalisierten Texte, indexiert nach PDF-Seitenzahl
  - **typ** : automatisch erkannter Typ der Seite (kann falsch / manuell korrigiert sein):
    - **bv-horz** : Seite zeigt Bestandsverzeichnis im Querformat
    - **bv-horz-zu-und-abschreibungen** : Seite zeigt BV-Zu- und Abschreibungen im Querformat
    - **bv-vert** : Seite zeigt Bestandsverzeichnis im Hochformat
    - **bv-vert-typ2** : Seite zeigt Bestandsverzeichnis im Hochformat, aber mit nur 3 Spalten (Gemarkung, Flur, Flurstück stehen untereinander)
    - **bv-vert-zu-und-abschreibungen** : Seite zeigt BV-Zu- und Abschreibungen im Hochformat
    - **abt1-horz** : Seite zeigt Abteilung 1-Formular im Querformat
    - **abt1-vert** : Seite zeigt Abteilung 1-Formular im Hochformat
    - **abt2-horz-veraenderungen** : Seite zeigt Abteilung 2-Veränderungen im Querformat
    - **abt2-horz** : Seite zeigt Abteilung 2-Formular im Querformat
    - **abt2-vert-veraenderungen** : Seite zeigt Abteilung 1-Formular im Hochformat
    - **abt2-vert** : Seite zeigt Abteilung 2-Formular im Hochformat
    - **abt3-horz-veraenderungen-loeschungen** : Seite zeigt Abteilung 3-Veränderungen / -Löschungen im Querformat
    - **abt3-vert-veraenderungen-loeschungen** : Seite zeigt Abteilung 3-Veränderungen / -Löschungen im Hochformat
    - **abt3-horz** : Seite zeigt Abteilung 3-Einträge im Querformat
    - **abt3-vert-veraenderungen** : Seite zeigt Abteilung 3-Veränderungen im Querformat
    - **abt3-vert-loeschungen** : Seite zeigt Abteilung 3-Löschungen im Querformat
    - **abt3-vert** : Seite zeigt Abteilung 3-Einträge im Hochformat
  - **texte** : Liste[Objekt]: Liste aller automatisch erkannten Textobjekte auf der Seite
    - **text** : StringOderMultiLine: Automatisch erkannter Text
    - **start\_y** : Float: Y-Position des Textanfangs in der Seite, gemessen von oberer linker Ecke, in mm
    - **end\_y** : Float: Y-Position des Textendes in der Seite, gemessen von oberer linker Ecke, in mm
    - **start\_x** : Float: X-Position des Textanfangs in der Seite, gemessen von oberer linker Ecke, in mm
    - **end\_x** : Float: X-Position des Textendes in der Seite, gemessen von oberer linker Ecke, in mm
- **analysiert** : ausgewertetes Grundbuch, nach automatischer Digitalisierung
  - **titelblatt** : Kopie test Root-titelblatt-Objekts
    - **amtsgericht** : String: Amtsgericht (von der Titelseite)
    - **grundbuch\_von** : String: Name des Grundbuchblatts / Grundbuchblattbezirks
    - **blatt** : Integer: Nummer des Grundbuchblatts
  - **bestandsverzeichnis** : Objekt: Bestandsverzeichnis des Grundbuchs
    - **eintraege** : Liste[BestandsverzeichnisEintrag]: Definition BestandsverzeichnisEintrag siehe oben
    - **zuschreibungen** : Liste[Objekt]:
      - **bv\_nr** : StringOderMultiLine: Spalte 1 der Bestandsverzeichnis-Veränderung
      - **text** : StringOderMultiLine: Spalte 2 der Bestandsverzeichnis-Veränderung
      - **automatisch\_geroetet** : Optional[Boolean]: ob der Eintrag vom Programm als gerötet erkannt wurde
      - **manuell\_geroetet** : Optional[Boolean]: ob der Eintrag vom Nutzer gerötet wurde, überschreibt "automatisch\_geroetet"
      - **position\_in\_pdf** : Optional[PositionInPdf]: Seite und Rechteck in mm, wo die Schrifterkennung das Objekt erkannt hat
    - **abschreibungen** : Liste[Objekt]:
      - **bv\_nr** : StringOderMultiLine: Spalte 1 der Bestandsverzeichnis-Abschreibung
      - **text** : StringOderMultiLine: Spalte 2 der Bestandsverzeichnis-Abschreibung
      - **automatisch\_geroetet** : Optional[Boolean]: ob der Eintrag vom Programm als gerötet erkannt wurde
      - **manuell\_geroetet** : Optional[Boolean]: ob der Eintrag vom Nutzer gerötet wurde, überschreibt "automatisch\_geroetet"

- `position_in_pdf` : Optional[PositionInPdf]: Seite und Rechteck in mm, wo die Schrifterkennung das Objekt erkannt hat
- `abt1` : Objekt: Abteilung 1 des Grundbuchs
  - `eintraege` : Liste[Objekt]: Einträge in Abteilung 1, Spalten 1 - 2
    - `lfd_nr` : Integer: laufende Nummer der Eintragung in Abteilung 1
    - `eigentuermer` : StringOderMultiLine: Eigentümer (Spalte 2)
    - `automatisch_geroetet` : Optional[Boolean]: ob der Eintrag vom Programm als gerötet erkannt wurde
    - `manuell_geroetet` : Optional[Boolean]: ob der Eintrag vom Nutzer gerötet wurde, überschreibt "automatisch\_geroetet"
    - `position_in_pdf` : Optional[PositionInPdf]: Seite und Rechteck in mm, wo die Schrifterkennung das Objekt erkannt hat
  - `grundlagen_eintragungen` : Einträge in Abteilung 1, Spalten 3 - 4
    - `bv_nr` : StringOderMultiLine: "BV-Nr.", Abteilung 1 Spalte 4
    - `text` : StringOderMultiLine: "Grundlage der Eintragung", Abteilung 1 Spalte 4
    - `automatisch_geroetet` : Optional[Boolean]: ob der Eintrag vom Programm als gerötet erkannt wurde
    - `manuell_geroetet` : Optional[Boolean]: ob der Eintrag vom Nutzer gerötet wurde, überschreibt "automatisch\_geroetet"
    - `position_in_pdf` : Optional[PositionInPdf]: Seite und Rechteck in mm, wo die Schrifterkennung das Objekt erkannt hat
  - `veraenderungen` : Liste[Objekt]: Veränderungsvermerke Abteilung 1
    - `lfd_nr` : Integer: Spalte 1 der Abt. 1 Veränderung
    - `text` : StringOderMultiLine: Spalte 2 der Abt. 1 Veränderung
    - `automatisch_geroetet` : Optional[Boolean]
    - `manuell_geroetet` : Optional[Boolean]
    - `position_in_pdf` : Optional[PositionInPdf]
  - `loeschungen` : Liste[Objekt]: Lösungsvermerke Abteilung 1
    - `lfd_nr` : Integer: Spalte 1 der Abteilung 1 Löschung
    - `text` : StringOderMultiLine; Spalte 2 der Abteilung 1 Löschung
    - `automatisch_geroetet` : Optional[Boolean]
    - `manuell_geroetet` : Optional[Boolean]
    - `position_in_pdf` : Optional[PositionInPdf]
- `abt2` :
  - `eintraege` : Liste[Objekt]:
    - `lfd_nr` : Integer
    - `bv_nr` : StringOderMultiLine
    - `text` : StringOderMultiLine
    - `automatisch_geroetet` : Optional[Boolean]
    - `manuell_geroetet` : Optional[Boolean]
  - `veraenderungen` : Liste[Objekt]:
    - `lfd_nr` : Integer
    - `text` : StringOderMultiLine
    - `automatisch_geroetet` : Optional[Boolean]
    - `manuell_geroetet` : Optional[Boolean]
  - `loeschungen` : Liste[Objekt]:
    - `lfd_nr` : Integer
    - `text` : StringOderMultiLine
    - `automatisch_geroetet` : Optional[Boolean]
    - `manuell_geroetet` : Optional[Boolean]

- `abt3` : Objekt
  - `eintraege` : Liste[Objekt]:
    - `lfd_nr` : Integer
    - `bv_nr` : StringOderMultiLine
    - `betrag` : StringOderMultiLine
    - `text` : StringOderMultiLine
  - `veraenderungen` : Liste[Objekt]:
    - `lfd_nr` : Integer
    - `betrag` : StringOderMultiLine
    - `text` : StringOderMultiLine
    - `automatisch_geroetet` : Optional[Boolean]
    - `manuell_geroetet` : Optional[Boolean]
  - `loeschungen` : Liste[Objekt]:
    - `lfd_nr` : Integer
    - `betrag` : StringOderMultiLine
    - `text` : StringOderMultiLine
    - `automatisch_geroetet` : Optional[Boolean]
    - `manuell_geroetet` : Optional[Boolean]
- `pdftotext_layout` : Objekt: enthält alle Texte, die aus dem PDF mit der Maus selektiert und kopiert werden können (vermeidet bei neuen Rechten fehleranfällige Texterkennung)
  - `seiten` : Map[Integer -> Objekt]: Seiten im PDF, nach Seitenzahl indexiert
    - `breite_mm` : Breite der Seite in mm
    - `hoehe_mm` : Höhe der Seite in mm
    - `texte` : Liste[Objekt]:
      - `text` : String,
      - `start_y` : Integer
      - `end_y` : Integer
      - `start_x` : Integer
      - `end_x` : Integer
- `seiten_versucht_geladen` : Liste[Integer]: Liste an Seiten, die automatisch klassifiziert wurden. Wenn eine Seite nicht automatisch klassifiziert werden konnte, wird sie trotzdem hier eingefügt.
- `seiten_ocr_text` : Map[Integer -> String]: reiner OCR-Text, der auf der Seite gefunden wurde - Cache für Performance
- `anpassungen_seite` : Map[Integer -> Objekt]: Zeilen / Spalten, die vom Anwender eingefügt wurden, indexiert nach Seitenzahl
  - `spalten` : Map[String -> Objekt]: Spalten, die vom Anwender mit der Maus verändert wurden, indexiert nach SeitenTyp
    - `min_x` : min-X-Wert der Spalte, von oberer linker Ecke, in mm
    - `min_y` : min-Y-Wert der Spalte, von oberer linker Ecke, in mm
    - `max_x` : max-X-Wert der Spalte, von oberer linker Ecke, in mm
    - `max_y` : max-Y-Wert der Spalte, von oberer linker Ecke, in mm
  - `zeilen` : Liste[Float]: Zeilen, die vom Nutzer eingefügt wurden, in mm vom oberen Rand der Seite
- `klassifikation_neu` : Map[Integer -> String]: manuelle Änderungen, die am Seitentyp vorgenommen wurden. Werte siehe `geladen.typ`
- `nebenbeteiligte_dateipfade` : Liste[String]: relative Dateipfade zu allen Nebenbeteiligten-.tsv-Dateien, werden beim Laden des Grundbuchs geladen

## .LEFIS-Datei (JSON)

Da einige Objekte mehrmals auftauchen, sind sie hier vor dem Erscheinen im Grundbuch-JSON aufgelistet:

### Nebenbeteiligter: Objekt

- `ordnungsnummer` : Optional[Integer]: Stammnummer
- `typ` : Optional[String]: Typ des Nebenbeteiligten - `GBH`, `OEFFENTLICH`, `LEITUNG`, etc.
- `name` : String: Name des Nebenbeteiligten laut Grundbuch
- `extra` : Objekt
  - `anrede` : Optional[String]: LEFIS-Attribut "Anrede"
  - `titel` : Optional[String]: LEFIS-Attribut "Titel"
  - `vorname` : Optional[String]: LEFIS-Attribut "Vorname"
  - `nachname_oder_firma` : Optional[String]: LEFIS-Attribut "Nachname oder Firma"
  - `geburtsname` : Optional[String]: LEFIS-Attribut "Geburtsname"
  - `wohntort` : Optional[String]: LEFIS-Attribut "Wohnort"

### FlurFlurstueck: Objekt

- `flur` : Integer: Flur des teilbelasteten Flurstücks
- `flurstueck` : String: Flurstück des teilbelasteten Flurstücks ("Zähler/Nenner")
- `gemarkung` : Optional[String]: Gemarkung des teilbelasteten Flurstücks
- `teilflaeche_qm` : Optional[Integer]: belastete Teilfläche (wenn nicht das ganze Flurstück belastet ist)

### LEFIS-Datei: Liste[Objekt]:

Achtung: Eine LEFIS-Datei kann mehrere Grundbuchblätter gleichzeitig enthalten!

- `titelblatt` : Objekt:
  - `amtsgericht` : String: Amtsgericht (von der Titelseite)
  - `grundbuch_von` : String: Name des Grundbuchblatts / Grundbuchblattbezirks
  - `blatt` : Integer: Nummer des Grundbuchblatts
- `rechte` : Objekt: Rechte in Abteilung 2 / 3 mit berechneten Flurstücken
  - `abt2` : Liste[Objekt]: ausgewertete und analysierte Abteilung 2-Einträge
    - `lfd_nr` : Integer: lfd. Nr. des Rechts Abteilung 2
    - `text_kurz` : String: gekürzter Text des Rechts Abteilung 2
    - `rechteart` : RechteArt (String): DABAG-RechteArt (siehe oben für mögliche Werte)
    - `rechtsinhaber` : String: Name des Rechtsinhabers
    - `rangvermerk` : Optional[String]: Rangvermerk
    - `spalte_2` : String: Text aus Spalte 2 (lasten an BV-Nummern)
    - `belastete_flurstuecke` : Liste[BestandsverzeichnisEintrag]: Liste aller belasteten Flurstücke
    - `lastend_an` : Liste[Objekt]: ausgelesene Belastungen und Teilbelastungen aus Spalte 2
      - `lfd_nr` : Integer: laufende Nummer des Grundstücks (Bestandsverzeichnis-Nr.)
      - `voll_belastet` : Boolean: ob dieses Grundstück voll belastet oder teilbelastet ist
      - `nur_lastend_an` : Liste[FlurFlurstueck]: "nur lastend an Flur X, Flurstück Y"-Teilbelastungen
    - `text_original` : String: gesäuberter Rohtext des Rechts
    - `nebenbeteiligter` : Objekt Nebenbeteiligter: siehe oben für Definition
    - `warnungen` : Liste[String]: Warnungen (werden unter dem Recht angezeigt)
    - `fehler` : Liste[String]: Fehler (werden unter dem Recht angezeigt)
  - `abt3` : Liste[Objekt]: ausgewertete und analysierte Abteilung-3-Einträge
    - `lfd_nr` : Integer: laufende Nummer der Schuld in Abteilung 3
    - `text_kurz` : String: gekürzter Text der Schuld in Abteilung 3
    - `betrag` : Objekt:
      - `wert` : Integer: Wert des Betrags vor dem Komma (300000 bei Spalte 2 = "300.000,00 EUR")
      - `nachkomma` : Integer: Nachkommastellen (0 bei Spalte 2 = "300.000,00 EUR")
      - `waehrung` : Waehrung (String): Währung, z.B. "MarkDDR", Werte siehe oben
    - `schuldenart` : SchuldenArt (String): Art der Schuld, Werte siehe oben



- `rechtsinhaber` : String: Begünstigter der Schuld (z.B. "Bank X GmbH")
- `spalte_2` : String: belastete BV-Nummern (Spalte 2 der Abteilung 3)
- `belastete_flurstuecke` : Liste[BestandsverzeichnisEintrag]: berechnete belastete Flurstücke
- `lastend_an` : Liste[Objekt]: ausgelesene Belastungen und Teilbelastungen aus Spalte 2
  - `lfd_nr` : Integer: laufende Nummer des Grundstücks (Bestandsverzeichnis-Nr.)
  - `voll_belastet` : Boolean: ob dieses Grundstück voll belastet oder teilbelastet ist
  - `nur_lastend_an` : Liste[FlurFlurstueck]: "nur lastend an Flur X, Flurstück Y"-Teilbelastungen
- `text_original` : String: gesäuberter Rohtext des Rechts
- `nebenbeteiligter` : Objekt Nebenbeteiligter: siehe oben für Definition
- `warnungen` : Liste[String]: Warnungen (werden unter dem Recht angezeigt)
- `fehler` : Liste[String]: Fehler (werden unter dem Recht angezeigt)

## Rechtliche Informationen

Das Programm selbst steht unter der MIT-Lizenz und wurde ursprünglich von Felix Schütt <Felix.Schuett@vlf-brandenburg.de> geschrieben. Icons wurden bereitgestellt von <https://icons8.com/>.

Bitte schreiben Sie mich bei Problemen oder Fragen zu dem Programm an.

---

Zeigt auf: [BB:Brandenburg](#)