

# edi.substanceforms

---

## Dokumentation des Projekts

**Seppo Walther**

---

**Winterprüfung 2021**

---

Fachinformatiker Anwendungsentwicklung

**educorvi GmbH & Co. KG**

Karolinenstraße 17  
90763 Fürth  
[www.educorvi.de](http://www.educorvi.de)

**08.12.2021**

*Seppo Walther*

CC BY 4.0

# Inhaltsverzeichnis

---

1. Dokumentation des Projekts edi.substanceforms	3
2. Zusammenfassung (Management Summary)	3
3. Änderungsnachweis	5
4. Ausgangssituation	6
5. Zielsetzung	7
6. Anforderungsanalyse	8
6.1 Bedienbarkeit über das Web-CMS Plone	8
6.2 Responsive Webdesign	9
6.3 Barrierefreiheit gemäß BITV 2.0 (WCAG 2.1 Level AA)	9
6.4 Integrität der gespeicherten Daten	10
6.5 Gefahrstoffgemisch hinzufügen	10
6.6 Bildbehandlung	10
7. Bestandteile der Applikation (Module und Klassen)	11
7.1 Artikeltypen für die Pflege von Inhalten	11
7.2 Formulare	12
7.3 Views	12
7.4 Funktionale Views	13
8. Anwendungsarchitektur	14
8.1 Blockschaubild der Anwendung	14
8.2 Ebenen der Anwendung	14
9. Installation und Konfiguration im CMS Plone	17
10. Automatische Testverfahren	18
10.1 Scope der Funktionstests	18
10.2 Test-Verzeichnis im Package	19
10.3 Ausführung der Tests	19
11. Lizenzvereinbarungen für edi.substanceforms	20
12. Anhang	21
12.1 Anhang 1: Datenbankmodell	21
12.2 Anhang 2: Aktivitätsdiagramm	22
12.3 Anhang 3 Funktionstests	23
12.4 Anhang 4: Glossar	24
12.5 Anhang 5: Sequenzdiagramm	27
12.6 Anhang 6: Klassendiagramme	28

# 1. Dokumentation des Projekts edi.substanceforms

---

Autor: Seppo Walther [seppo.walther@educorvi.de](mailto:seppo.walther@educorvi.de)

IHK-Prüflingsnummer: 158 22139

Das hier dokumentierte Softwareprojekt wurde im Rahmen meiner Projektarbeit für die Abschlussprüfung zum Fachinformatiker für Anwendungsentwicklung bei der IHK Nürnberg für Mittelfranken erstellt. Bei meinem Antrag auf Genehmigung der Projektarbeit habe ich angekündigt, eine Technische Dokumentation für dieses Open Source Softwareprojekt einzureichen. Dieser Antrag wurde von der IHK genehmigt. Ich versichere hiermit, die Arbeit allein erstellt zu haben und das Dokument sowohl meiner Ausbilderin als auch dem Geschäftsführer meines Ausbildungsbetriebes zur Durchsicht und Freigabe vorgelegt zu haben. In dieser technischen Dokumentation werden keine Geschäftsgeheimnisse preisgegeben und es sind keine datenschutzrechtlich relevanten Daten oder Informationen enthalten. Die Gliederung meiner Dokumentation und die inhaltliche sowie die äußere Form als Online-Dokumentation mit speziellem PDF-Export wurde im Projektverlauf des agilen Entwicklungsprojektes sowohl mit dem Ausbildungsbetrieb, als auch mit dem Kunden als Anwender der Software abgestimmt und entspricht deren Anforderungen.

Fürth, 08.12.2021 Seppo Walther

Das Dokument wurde von uns gesehen und für die Übermittlung an die IHK Nürnberg für Mittelfranken freigegeben.

Fürth, 08.12.2021 Anke Lampe (Ausbilderin)

Fürth, 08.12.2021 Lars Walther (Geschäftsführer)

Die Software befindet sich hier: <https://github.com/educorvi/edi.substanceforms>.

Für die Dokumentation wurde ein eigenes Softwarepackage auf Basis von MKDocs (siehe <https://www.mkdocs.org/>) entwickelt.

Die Dokumentation befindet sich hier: <https://github.com/educorvi/edi.docTheSubstanceForms/>

## 2. Zusammenfassung (Management Summary)

---

Mit dem Add-On edi.substanceforms für das freie Web-Content-Management-System **Plone** wird die Online-Datenbank **Emissionsarme Produkte** der Berufsgenossenschaft Energie Textil Elektro und Medienerzeugnisse (BG ETEM) realisiert.

Mit Hilfe dieser Online-Datenbank können Unternehmerinnen und Unternehmer, Arbeitsschützer und Verantwortliche in den Betrieben der Branche Druck und Papierverarbeitung Chemikalienprodukte finden, deren Einsatz in der Produktionsumgebung mit möglichst geringen Gesundheitsgefahren für die damit in Kontakt stehenden Beschäftigten verbunden ist.

Die Entwicklung des Plone-Add-Ons war Gegenstand der Projektarbeit von Seppo Walther im Rahmen der Ausbildung zum Fachinformatiker für Anwendungsentwicklung und wird dem Prüfungsausschuss der IHK Nürnberg für Mittelfranken zur Bewertung vorgelegt.

Für die Durchführung des Entwicklungsprojekts wurde eine agile Projektorganisation gewählt. Diese Entscheidung hat sich vor allem unter den besonderen Bedingungen der Corona-Pandemie bewährt. Nach Vorbereitung des Softwarepackages mit den Generierungs-Werkzeugen von "Mr.Bob" (bobtemplates.plone, siehe <https://bobtemplatesplone.readthedocs.io/en/latest/>) wurden alle wesentlichen Bestandteile des Add-Ons in 3 zweitägigen Sprints entwickelt. Davon fand ein Sprint mit Kundenbeteiligung in Präsenz am Sitz der educorvi GmbH & Co. KG in Fürth statt. Für die Sprints wurden jeweils Kanban-Boards aufgestellt und erfolgreich abgearbeitet.

Auch wenn es unwahrscheinlich erscheint, dass die hier entwickelte Software 1:1 in anderen Umgebungen oder für vergleichbare Anwendungsfälle eingesetzt werden kann, so liefert das Projekt doch verschiedene Hinweise und Softwarebestandteile für vergleichbare Projekte. In Abstimmung mit dem Kunden und dem Ausbildungsbetrieb wurde deshalb entschieden, das Projekt als Open Source Software auf der Plattform GitHub(<https://github.com>) zu veröffentlichen.

Für die Dokumentation wurde entschieden, auf MKDocs zu setzen. MKDocs erlaubt die Dokumentation mittels Markdown-Dokumenten. Die Dokumentation kann mit dem gleichen Werkzeug wie die Erstellung des Softwarecodes erfolgen. Außerdem kann die Dokumentation ebenfalls über die Plattform Github versioniert werden und es können auf diese Weise Reviews durchgeführt werden. MKDocs erlaubt die Online-Veröffentlichung der Dokumentation über automatisch generierte HTML-Seiten. Mit dem für dieses Projekt angepassten Zusatzmodul mkdocs-with-pdf (<https://pypi.org/project/mkdocs-with-pdf/>) wurde die Dokumentation automatisch und ohne Zusatzaufwand als PDF-Dokument exportiert.

## **Fazit**

Im abschließenden Projektgespräch mit dem Ausbildungsbetrieb und dem Kunden am 02.12.2021 wurde festgestellt, dass die entwickelte Software den Kundenanforderungen entspricht und außerdem Potenzial für die Weiterentwicklung bietet. Für den aktuellen Softwarestand wird ein Branch mit der Version 1.0b1 gebildet und mit Hilfe von GitHub ein Release erstellt. Unmittelbar danach wird die Weiterentwicklung der Software anhand neuer Kundenanforderungen fortgesetzt. Dazu zählt beispielsweise die Integration der Gefahrstoffdatenbank GESTIS der Deutschen gesetzlichen Unfallversicherung (DGUV).

### 3. Änderungsnachweis

---

Version	Datum	Autor	Bemerkung
1.0b1	08.12.21	Seppo Walther	Releasekandidat für Kunde, Einreichung Projektarbeit IHK Nürnberg

---

## 4. Ausgangssituation

---

Die Berufsgenossenschaft Energie Textil Elektro Medienerzeugnisse (BG ETEM) pflegt für die Betriebe der Branche Druck und Papierverarbeitung eine Online-Datenbank mit speziellen Chemikalien und Chemikaliengemischen, die in den Produktionsprozessen der Branche Druck- und Papierverarbeitung zum Einsatz kommen. Zielsetzung des Bereiches Prävention der BG ETEM ist es, über diese Online-Datenbank auf die in den Chemikalien enthaltenen Gefahrstoffe hinzuweisen, und die Unternehmerinnen, Unternehmer und Arbeitsschützer zu motivieren, vor allem die Chemikalienprodukte einzusetzen, die mit möglichst geringen Gesundheitsgefahren für die Beschäftigten der Branche verbunden sind.

So ist beispielsweise bei Chemikaliengemischen darauf zu achten, dass ein möglichst hoher Flammpunkt für einen geringen Eintrag von giftigen Dämpfen in die Atemluft steht und dass Druckbestäubungspuder verwendet werden, die eine bestimmte Mindest-Körnungsgröße aufweisen, um nicht lungengängig zu sein.

Die Online-Datenbank trägt deshalb den Namen **Emissionsarme Produkte**.

Die Datenbank wurde auf Basis des Web-Content-Management-Systems Plone entwickelt. Die Eigenschaften der Chemikalienprodukte, deren Inhaltsstoffe, Hersteller und vieles mehr wurden bislang in einer objektorientierten Datenbank gespeichert. Die redaktionelle Arbeit zur Bearbeitung und Pflege sowie die Recherche der Endbenutzer erfolgte über die Leistungsmerkmale des CMS Plone.

Sowohl die Oberflächen für die Benutzer, Autoren und Redakteure als auch die Datenstrukturen in dieser Datenbank sind veraltet. Die Möglichkeit, Chemikalien gezielt nach gewissen Parametern zu durchsuchen, ist aufgrund der technischen Eigenschaften der objektorientierten Datenbank nur sehr eingeschränkt.

Zu den Chemikalienprodukten gehören:

- Wasch- und Reinigungsmittel für den Offsetdruck
- Sonderreiniger für die manuelle Anwendung im Offsetdruck
- Reinigungsmittel im Etikettendruck
- Staubarme Druckbestäubungspuder

## 5. Zielsetzung

---

Die Zielsetzung des Projekts besteht in der grundsätzlichen Anpassung und Modernisierung der zugrunde liegenden Softwarearchitektur. Vor allem die Umstellung von einer objektorientierten Datenbank auf eine relationale Datenbank bildet eine wesentliche Zielsetzung des Projekts. Daneben soll für die Endbenutzer eine verbesserte und vor allem schnellere Recherche über die in der Datenbank gespeicherten Chemikalienprodukte ermöglicht werden. Die redaktionelle Bearbeitung der Daten soll weiterhin über das bestehende Web-Content-Management-System Plone ermöglicht werden.

Im Rahmen eines Vorprojektes wurde ein ERM-Modell für eine relationale Datenbank entwickelt und anhand von Test-Datensätzen erprobt. Außerdem wurden im Rahmen des Vorprojekts die technischen Möglichkeiten zur Migration der Daten aus der bestehenden objektorientierten Datenbank in die relationale Datenbank getestet. Die im Vorprojekt erzielten Ergebnisse wurden als Proof of Concept gesichert und dokumentiert. Dementsprechend konnte in diesem Projekt auf die Ergebnisse des Vorprojekts sowohl in Form von Softwarebestandteilen als auch in Form von Dokumentation zurückgegriffen werden.

## 6. Anforderungsanalyse

---

Die Anforderungsanalyse ist für die technische Dokumentation vor allem vor dem Hintergrund des Verständnisses für die technischen Zusammenhänge der realisierten Softwarelösung von Bedeutung. Dieses Kapitel ist deshalb als Gegenüberstellung der Kundenanforderungen mit den im Projekt entwickelten Artefakten zu verstehen.

### 6.1 Bedienbarkeit über das Web-CMS Plone

---

Die folgenden Anforderungen werden durch die im Projekt entwickelten Nutzerformulare und Einzelansichten erfüllt:

- Formulare zum Anlegen von Herstellern und Chemikalienprodukten durch die Autoren und Redakteure mit integrierter Plausibilitätsprüfung und Speicherung im RDBMS
- Einzelansichten für Hersteller und Chemikalienprodukte und Auflistung von deren Eigenschaften für den redaktionellen Prozess
- Möglichkeit zur Löschung von einzelnen Datenbankobjekten aus den Einzelansichten
- Formulare zur Aktualisierung von Herstellern und Chemikalienprodukten
- Suche nach Produkten und Herstellern im Rahmen des redaktionellen Prozesses (keine Endbenutzersuche)

Außerdem waren die folgenden Punkte zu berücksichtigen und wurden entsprechend umgesetzt:

- Nutzung der Benutzerverwaltung des Web-CMS Plone und des Rechte- und Rollenkonzeptes von Plone
- Entwicklung der CMS-Artikeltypen "Datenbank" und "Tabelle". Die Inhaltstypen erlauben den Autoren und Redakteuren den freien Aufbau einer Navigationsstruktur innerhalb des CMS, stellen die technische Verbindung zu den Datenbankobjekten her und erlauben die Nutzung des Berechtigungskonzeptes von Plone bis auf die Ebene der Datenbanktabellen.
- Entwicklung von Methoden zur expliziten Prüfung der Rollen von Benutzern im Kontext der Formulare, mit denen einzelne Datensätze bearbeitet werden können:
- **userCanEdit**, prüft das Schreibrecht von Benutzern
- **userCanReview**, prüft das Recht von Benutzern zur Veröffentlichung



Für die Artikeltypen Datenbank und Tabelle verwenden wir einen eigenen Workflow, der die Berechtigungen für die Status wie folgt definiert:

- **privat:** Lese- und Schreibrechte für die CMS-Rollen Owner und Manager
- **Entwurf:** Lese- und Schreibrechte für die CMS-Rollen Editor und Manager, Leserecht für Reviewer
- **zur Veröffentlichung eingereicht:** Lese- und Schreibrechte für die CMS-Rollen Reviewer und Manager, Leserecht für Editor
- **Veröffentlicht:** Lese- und Schreibrechte für die CMS-Rollen Reviewer und Manager, Leserecht für Anonymous

## 6.2 Responsive Webdesign

---

Auch hier ist das CMS Plone hilfreich. Plone selbst unterstützt von Haus aus Responsive Webdesign. Das Webdesign des Kunden verwendet außerdem Bootstrap als CSS- und Javascript- Basis. Dementsprechend wurde für die Realisierung aller HTML-Templates im Projekt der HTML-Markup des Bootstrap Frameworks in der Version 4.6 (siehe <https://getbootstrap.com/docs/4.6/getting-started/introduction/>) verwendet.

Bootstrap teilt jeden Bildschirm in ein Raster aus 12 Spalten ein und ordnet den Endgeräten je nach Größe des tatsächlichen physikalischen Bildschirms eine von 4 Bildschirmgrößen zu:

- xs (extra-small),
- sm (small),
- md (medium) und
- lg (large)

zu.

In den Templates des Plone-Add-Ons wird definiert, welche Template-Elemente bei welcher Bildschirmgröße wie viele Spalten einnehmen sollen. So lässt sich beispielsweise an einem Arbeitsplatzcomputer ein Bild neben einer Tabelle anzeigen, während es am Smartphone nicht möglich ist und deshalb automatisch (responsive) unter der Tabelle angezeigt wird.

## 6.3 Barrierefreiheit gemäß BITV 2.0 (WCAG 2.1 Level AA)

---

Bei korrekter Verwendung des HTML-Markups von Bootstrap ist die Barrierefreiheit aus technischer Sicht gewährleistet. Ausserdem wird in den Formularen zum Upload und zum Update der Bilder zu Herstellern und Chemikalienprodukten sichergestellt, dass Texte für Titel und Bildbeschreibung angegeben werden können. Die Bilder selbst werden als Artikel im CMS Plone gespeichert. In den

Einzelansichten für Hersteller und Chemikalienprodukte werden die Titel- und Beschreibungstexte in den `<img>`-Tags verwendet:

- title = Titel des Bildes
- alt = Kurzbeschreibung des Bildes (außerdem Verwendung in der Bildunterschrift)

## 6.4 Integrität der gespeicherten Daten

---

Die Integrität der gespeicherten Daten wird durch Datentypisierung und fachliche Plausibilitätsprüfungen sichergestellt. Somit wird jeder von den Autoren und Redakteuren eingetragene Wert sowohl im Hinblick auf das Datenformat als auch im Hinblick auf die fachliche Richtigkeit geprüft.

## 6.5 Gefahrstoffgemisch hinzufügen

---

Eine Besonderheit bildet das Gefahrstoffgemisch. Während sonst stets das Produkt der Datenbanktabelle entspricht (beispielsweise wird ein Hersteller in der Hersteller-Datenbanktabelle angelegt), so fasst die Datenbanktabelle Gefahrstoffgemisch die Produkte Etikettenwaschmittel, Waschmittel für UV-Druck, Heatsetwaschmittel, Sonderreiniger und Offsetdruck-Reinigungsmittel zusammen. Daher muss darauf geachtet werden, dass in der Form zum Hinzufügen des Gefahrstoffgemisches die Entscheidungen zur Zuordnung getroffen werden können, die in dem Aktivitätsdiagramm [siehe Anlage 2](#) entsprechend analysiert wurden.

## 6.6 Bildbehandlung

---

Für einige Produkte (wie beispielsweise das Gefahrstoffgemisch) soll es möglich sein, ein entsprechendes Produktbild zu hinterlegen. Bei der Analyse dieser Anforderung wurde im Projektverlauf entschieden, das Bild nicht in der relationalen Datenbank PostgreSQL zu speichern, da die objektorientierte Datenbank Zope, die im verwendeten CMS Plone enthalten ist, hierfür besser geeignet ist. Beim Speichern eines Bildes muss also dafür gesorgt werden, dass das Bild einerseits in Zope hochgeladen wird, während eine Referenzierung des Bildes anhand einer ID in die relationale Datenbank erfolgt. Dieses Verfahren ist [Anhang 5](#) zu entnehmen.

## 7. Bestandteile der Applikation (Module und Klassen)

---

Alle entwickelten Bestandteile des Projekts befinden sich im Plone-Add-On **edi.substanceforms**. Dieses Add-On beinhaltet Content-Types (Artikeltypen), mit denen in Plone die Inhalte gepflegt werden, sowie alle benötigten Formulare und Ansichten (Forms und Views) für das Management der Chemikalienprodukte mit dem CMS Plone. Die Bestandteile der Applikation werden in den folgenden Abschnitten detaillierter beschrieben.

Für die Erstellung der Plone-Inhaltstypen (Content-Types, Viewklassen, Viewtemplates...) kommt der Generator [mr.bob](#) zum Einsatz. Dieser ermöglicht den Fokus auf die essentielle Programmierung und Anpassung der Inhaltstypen durch massive Zeitersparnisse gegenüber einer vollständigen manuellen Programmierung.

### 7.1 Artikeltypen für die Pflege von Inhalten

---

Die Content-Types haben für die Autoren und Redakteure eine besondere Bedeutung. Mit deren Hilfe werden in der Struktur der Plone-Site Inhalte angelegt, um diese für Benutzerinnen und Benutzer zu präsentieren.

Daneben können in den Content-Types Daten gespeichert werden, die für die Steuerung der Anwendung benötigt werden. Das Package `edi.substanceforms` beinhaltet folgende Content-Types:

#### 7.1.1 Datenbank

---

Die Datenbank **Emissionsarme Produkte** kann mit diesem Artikeltyp an beliebiger Stelle in der Inhaltsstruktur der Website eingebunden werden. Die Autoren und Redakteure können mit entsprechenden Feldern die Datenbank beschreiben und für die Benutzerinnen und Benutzer präsentieren. So stehen Textfelder zur Verfügung, um Informationen über die Datenbank bereitzustellen: was ist gespeichert? Wie interagiert der Nutzer mit den Informationen? Außerdem stellt die Datenbank die technische Verbindung zur physikalischen Datenbank her. D.h. es werden mit diesem Artikeltyp alle notwendigen Parameter für die Verbindung zur Datenbank gespeichert.

Aus der Sicht des Web-CMS-Plone stellt die Datenbank einen Ordnertyp dar, in dem Objekte vom Content-Type Tabelle gespeichert werden.

#### 7.1.2 Tabelle

---

Der Content-Typ Tabelle stellt die technische Verbindung zu den Tabellen der physikalischen Datenbank her. Bereits beim Anlegen des Content-Types in Plone wird der Autor und Redakteur gefragt, welche der Tabellen der Datenbank angezeigt werden soll. Mit dem Content-Typ Tabelle wird die Absicht verfolgt, im Kontext einer Tabelle die darin gespeicherten Datensätze anzuzeigen, als wären sie selbst als Objekte im CMS Plone gespeichert.

Aus der Sicht des Web-CMS Plone stellt die Datenbank ebenfalls einen Ordnerstyp dar, in dem Dateiobjekte (Bilder, PDF-Dateien) gespeichert werden, die für die Präsentation der Datensätze, also z.B. Hersteller, Chemikalienprodukte, benötigt werden. Für die Speicherung dieser Objekte wird ganz bewusst die objektorientierte Datenbank genutzt. In den Datensätzen werden Referenzen zu den Dateiobjekten gehalten.

## 7.2 Formulare

---

Formulare werden nach dem CRUD-Begriff (**C**reate, **U**ppdate, **D**eleate) für folgende Anwendungsfälle benötigt:

- **Hinzufügen** von Datensätzen in die Datenbank (create)
- **Aktualisierung** von Datensätzen in der Datenbank (update)

Das **Löschen** von Datensätzen wird mit einem funktionalen View realisiert.

Ausserdem wird ein Formular für die Suche von Datensätzen in der Datenbank benötigt.

## 7.3 Views

---

### 7.3.1 Einzelansicht für die Datenbank

---

In der Einzelansicht der Datenbank (datenbank\_view.pt) werden die vorher ebenfalls zum CMS hinzugefügten Tabellen aufgelistet. Die Tabellen sind die Kind-Elemente einer Datenbank, oder mit anderen Worten: die Inhalte des Ordners Datenbank. Eine Tabelle repräsentiert ihrerseits Hersteller oder Chemikalienprodukte.

### 7.3.2 Einzelansicht für die Tabelle

---

Die Einzelansicht für die Tabelle (tabelle\_view.pt) dient dem Listing aller Datensätze in der Tabelle und enthält entsprechende Suchparameter, nach denen die Trefferliste bei Bedarf gefiltert werden kann. So ist beispielsweise die Anzeige aller Gefahrstoffgemische möglich, die vorher als Sonderreiniger definiert wurden.

### 7.3.3 Einzelansicht für Hersteller und Gefahrstoffprodukte

---

Für jede Art der Gefahrstoffprodukte gibt es eine individuelle Ansicht (Viewtemplate). In diesem Viewtemplate wird eine HTML-Tabelle generiert, in der die entsprechenden Eigenschaften des Produktes zur Anzeige gebracht werden.

Beispiel:

<b>Merkmal</b>	<b>Eigenschaft</b>
Flammpunkt	> 70 Grad Celsius
Hautschutzkategorie	wasserlöslich
Inhaltsstoffe	Aceton > 2% < 10%

Außerdem erfolgt hier die Darstellung der gespeicherten Bilder. Folgende Viewtemplates sind vorhanden:

- `substance_view.pt`
- `substance_mixture_view.pt`
- `spray_powder_view.pt`
- `manufacturer_view.pt`

Viele der in der Datenbank gespeicherten Werte sind Schlüsselwerte, die nicht ohne Übersetzung lesbar sind. Daher erfolgt in der Viewklasse `single_view.py`, welche allen Einzelansichten zugrunde liegt (Prinzip Vererbung), die Schlüssel-/Wert-Zuweisung bzw. die Übersetzung von Werten für die Benutzer.

## 7.4 Funktionale Views

---

Ein funktionaler View zeichnet sich dadurch aus, entweder kein eigenes Template zu besitzen, oder eines, das auf die Bestätigung einer Aktion beschränkt ist. Zu den funktionalen Views gehören:

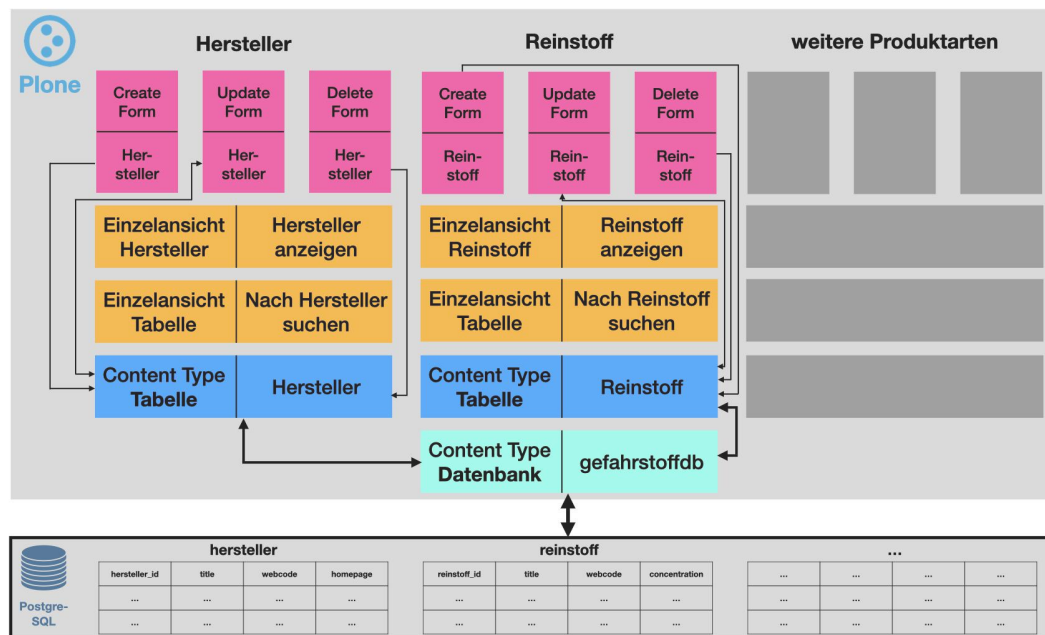
- `single_view`
- `selector_view`
- `delete_view`

So besitzt beispielsweise der Delete View ein Template zur Bestätigung des Löschens des Artikels per Auswahl einer Checkbox, während in den single-view alle Methoden der Einzelansichten ausgelagert wurden.

## 8. Anwendungsarchitektur

### 8.1 Blockschaubild der Anwendung

Hierbei handelt es sich um eine vereinfachte Darstellung des Zusammenspiels unterschiedlicher Komponenten des Projekts.



[Bild öffnen](#)

### 8.2 Ebenen der Anwendung

Die Anwendung besteht aus mehreren logischen Ebenen, die miteinander kommunizieren und Daten untereinander austauschen können. Im Folgenden wird das obige Bild, und damit die Funktionsweise von edi.substanceforms, von unten nach oben beschrieben.

#### 8.2.1 PostgreSQL

##### Relationales Datenbanksystem PostgreSQL

Die Basis der Architektur bildet ein relationales Datenbankmanagementsystem (RDBMS), basierend auf PostgreSQL. Da die Einrichtung dieser Datenbank nicht Teil des Projektes ist, wird hier nur soweit darauf eingegangen um darzustellen, dass alle Daten zu Herstellern und Gefahrstoffprodukten hier gespeichert werden. Dafür kommt jeweils eine Tabelle zum Einsatz, die in ihren Spalten alle Eigenschaften der Produkte abbildet. Eine Ausnahme bilden die Bilder zu den Gefahrstoffprodukten. Diese werden direkt im CMS gespeichert. In der Datenbank wird ein URL-Referenz auf das Bild gespeichert.

## 8.2.2 Plone

Der genaue Aufbau der Software ist obiger Grafik zu entnehmen, der Entwurf der Klassendiagramme ist in [Anhang 6](#) zu finden.

### Content-Type Datenbank

Der Plone Content-Type (CT) Datenbank interagiert direkt mit der zugrunde liegenden PostgreSQL Datenbank. Da im CT Datenbank die Anmeldedaten der Datenbank hinterlegt werden, ist es das Modul, welches die Verbindung zur PostgreSQL- Datenbank herstellt und für die Anwendung aufrecht erhält.

### Content-Type Tabelle

Auf dem CT Datenbank sitzt der CT Tabelle. Für jede PostgreSQL-Datenbanktabelle, die im CMS abgebildet werden soll, wird eine CT Tabelle angelegt. Darin wird neben der referenzierten Datenbanktabelle auch spezifiziert, welche Spalten später bei der Einzelansicht berücksichtigt werden sollen. Für Gefahrstoffgemische gibt es die Möglichkeit, nur bestimmte Typen dieser Mixtur anzuzeigen.

### Einzelansicht Tabelle

Die Standardansicht der Tabelle besitzt die Möglichkeit, alle Elemente der Tabelle (=Datensätze) anzuzeigen. Diese Standardansicht wird im Programmcode für den Hersteller und die Gefahrstoffprodukte weiter spezialisiert werden. Auf diese Weise wird der Benutzer in die Lage versetzt, die Datensätze nach entsprechenden Kriterien zu filtern. Diese Ansicht bildet eine Trefferliste, bei der die Benutzer auf einen Datensatz klicken können, um in die spezialisierte Einzelansicht des Herstellers oder Gefahrstoffproduktes zu kommen.

### Einzelansicht Hersteller oder Produkt

Für jede Tabelle und die darin gespeicherten Datensätze gibt es spezialisierte Einzelansichten. In diesen Einzelansichten werden alle Eigenschaften der Hersteller oder Gefahrstoffprodukte, welche aus der PostgreSQL Datenbank abgerufen werden, visuell aufbereitet und angezeigt. Hierfür stehen der Einzelansicht unterschiedlichste Übersetzungs- und Umwandlungsmethoden der Klasse `single_view` zur Verfügung, welche von allen Einzelansichten gleichermaßen genutzt werden können.

### Create-View

Der Create-View erlaubt es, der Datenbank einen neuen Datensatz hinzuzufügen. Er wird in der Ansicht der Tabelle über einen Button "Artikel hinzufügen" aufgerufen. Der Create-View ist ein HTML-Formular und beinhaltet Felder zu allen Attributen, die erfasst werden können bzw. müssen. Die eingegebenen Eigenschaften werden nach mehrerer Prüfung auf Plausibilität der Daten schließlich in die PSQL-Datenbank geschrieben.

### **Update-View**

Der Update-View wird aus der Einzelansicht des Herstellers oder Produktes über einen Button "Artikel bearbeiten" aufgerufen. Der Update-View ist technisch wie optisch nahezu identisch mit dem Create-View, beinhaltet ebenfalls ein HTML-Formular mit allen Feldern, jedoch werden hier die Felder bereits vorbelegt mit den vorhandenen Daten. Soll eine Änderung vorgenommen werden, so ändert der Nutzer beispielsweise im Feld "Titel" den Inhalt von "Silux Max Super" zu "Silux Super Max".

### **Delete-View**

Der Delete-View wird ebenfalls aus der Einzelansicht des Produktes aufgerufen, zum Einsatz kommt ein Button "Artikel löschen". Der Delete-View beinhaltet eine Abfrage, ob man sich sicher sei, das Produkt löschen zu wollen. Wird das Kästchen entsprechend angekreuzt, so wird der Datensatz aus der Datenbanktabelle von PostgreSQL entfernt.



## 9. Installation und Konfiguration im CMS Plone

---

### Technische Voraussetzungen für den Einsatz von edi.substanceforms:

Folgende Komponenten und Produkte sind Voraussetzung für die Anwendung von edi.substanceforms

- Lauffähige Datenbank auf Basis von PostgreSQL
- Installation von Python >= 3.8
- Lauffähige Installation (Buildout) des Web CMS Plone in der Version >= 5.2.4

Das Package **edi.substanceforms** wird auf der Plattform [GitHub](#) gehostet.

Um edi.substanceforms im Web-Content-Management-System Plone zu installieren, wird das Package zunächst via git clone in das Quellen-Verzeichnis (src) der Buildout-Installation heruntergeladen und für die Installation in der Datei **buildout.cfg** eingetragen. Die Installation erfolgt dann im Rahmen des Buildout-Prozesses

```
{buildout-home}$ cd src
{buildout-home}/src$ git clone https://github.com/educorvi/edi.substanceforms.git
```

Einträge in der Datei buildout.cfg der Plone-Installation:

```
eggs:
    ...
    edi.substanceforms

develop:
    ...
    src/edi.substanceforms
```

Durchführung eines neuen Buildouts im Wurzelverzeichnis der Plone-Installation:

```
{buildout-home}$ ./bin buildout
```

Anschließend kann der Plone-Client hochgefahren werden, dies geschieht mit dem Befehl:

```
{buildout-home}$ ./bin/{instance-name} start
```

Danach muss das Package in den Plone-Einstellungen installiert werden.

--> Einstellungen ({site-url}/@@plone\_control\_panel) --> edi.substanceforms --> **Installieren**

Das Package wurde nun erfolgreich installiert und ist einsatzbereit.

Nun kann mit dem Hinzufügen einer Datenbank begonnen werden.

## 10. Automatische Testverfahren

---

Die Entwicklung von edi.substanceforms wurde im grundsätzlichen Ansatz testgetrieben durchgeführt. Wie im Kapitel "Bestandteile" beschrieben, wurden die Inhaltstypen **Datenbank** und **Tabelle** sowie alle Ansichten (Views) und Formulare (Forms) zunächst mit dem Generator mr.bob generiert.

Im Zuge der Generierung werden bereits Templates für funktionale Tests im Package angelegt. Ausserdem werden in der Datei setup.py des Packages die Abhängigkeiten zur Durchführung automatisierter Tests vermerkt. Um die automatisierten Tests ausführen zu können, müssen diese Abhängigkeiten zunächst installiert werden. Dazu muss folgende Änderung in der buildout.cfg vorgenommen werden:

```
eggs =
    ...
    edi.substanceforms [test]
```

Danach muss der Buildout mit dem Befehl `./bin/buildout` im Wurzelverzeichnis der Installation erneut durchgeführt werden.

### 10.1 Scope der Funktionstests

---

Funktionstests im Web-CMS Plone verfügen über eine echte (objektorientierte) Datenbank und eine Komponentenarchitektur. Darüber hinaus kann mit Python-Code ein Browser simuliert werden. Wenn mit dem Python-Browser auf eine Seite zugegriffen wird, ist die komplette Transaktionsmaschinerie von Plone im Einsatz. Damit dies funktioniert, verpackt die Testschicht die Datenbank in einen Demospeicher, der einen regulären Speicher kapselt. Wird von den Tests etwas in die Datenbank geschrieben, speichert der Demostorage es in temporären Feldern des Arbeitsspeichers. Nach jedem Test wird der Demostorage gelöscht. Damit sollten die funktionalen Tests fast so schnell ablaufen wie Integrationstests oder gar Unit-Tests, und das trotz des zusätzlichen Overheads, der durch das Durchlaufen der Plone-Transaktionsmaschine aufgebaut wird.

Zu beachten ist dabei, dass der simulierte Browser reinen Python-Code darstellt. Javascript-Code kann somit im Rahmen der funktionalen Tests nicht getestet werden.

Ebenso wurden die Datenbank-Transaktionen zur relationalen Datenbank auf Basis von PostgreSQL nicht in die Funktionstests einbezogen. Die tatsächliche Kundenkonfiguration wird erheblich von einer Test- oder Entwicklungskonfiguration abweichen. Die generische Implementierung der notwendigen technischen Voraussetzungen für die Installation und Konfiguration der Testumgebung ist nicht möglich.

Die in edi.substanceforms implementierten Funktionstests umfassen die folgenden Themen:

- fehlerfreie Installation des AddOns im Content-Management-System Plone
- Anlegen der Inhaltstypen Datenbank und Tabelle
- Assoziation der View- und Formklassen zu den Inhaltstypen

## 10.2 Test-Verzeichnis im Package

Alle für die Ausführung der Funktionstests mittels Testroboter ((`plone.app.testrobot`)[<https://pypi.org/project/plone.app.robotframework/>]) notwendigen Konfigurationsdateien sowie die Dateien mit den Funktionstests befinden sich im Verzeichnis `../src/edi/substanceforms/tests/..`:

```
...src/edi/substanceforms/tests$ ls -la
total 132
drwxrwxr-x 4 plone_buildout plone_buildout 4096 Dec  4 13:03 .
drwxrwxr-x 9 plone_buildout plone_buildout 4096 Dec  4 12:28 ..
-rw-rw-r-- 1 plone_buildout plone_buildout   0 Nov 25 10:50 __init__.py
drwxrwxr-x 2 plone_buildout plone_buildout 4096 Dec  4 10:45 __pycache__
drwxrwxr-x 2 plone_buildout plone_buildout 4096 Nov 25 10:50 robot
-rw-rw-r-- 1 plone_buildout plone_buildout 3042 Nov 25 10:50 test_ct_datenbank.py
...
-rw-rw-r-- 1 plone_buildout plone_buildout 1564 Dec  4 11:33 test_view_update_mixture-form.py
-rw-rw-r-- 1 plone_buildout plone_buildout 1539 Dec  4 11:33 test_view_update_powder_form.py
-rw-rw-r-- 1 plone_buildout plone_buildout 1570 Dec  4 11:33 test_view_update_substance-form.py
-rw-rw-r-- 1 plone_buildout plone_buildout 1518 Dec  4 11:33 test_view_update_view.py
```

## 10.3 Ausführung der Tests

Die Tests für `edi.substanceforms` können aus dem Buildout-Verzeichnis heraus mit dem folgenden Befehl ausgeführt werden:

```
$ ./bin/test edi.substanceforms

...
Ran 65 tests with 0 failures, 0 errors, 0 skipped in 3.067 seconds.
Tearing down left over layers:
  Tear down edi.substanceforms.testing.EdiSubstanceformsLayer:IntegrationTesting in 0.000 seconds.
  Tear down edi.substanceforms.testing.EdiSubstanceformsLayer in 0.004 seconds.
  Tear down plone.app.contenttypes.testing.PloneAppContenttypes in 0.020 seconds.
  Tear down plone.app.event.testing.PAEventLayer in 0.004 seconds.
  Tear down plone.app.testing.layers.PloneFixture in 0.028 seconds.
  Tear down plone.testing.zope.Startup in 0.003 seconds.
  Tear down plone.testing.zca.LayerCleanup in 0.001 seconds.
```

# 11. Lizenzvereinbarungen für edi.substanceforms

---

Für jedes Softwareprodukt ist es von essentieller Bedeutung, Rechte und Ansprüche auf das geistige Eigentum zu definieren und zu schützen. Hierfür werden meist Lizenzen verwendet, die gewisse Rechte klar festlegen. Die educorvi GmbH & Co. KG folgt dem Open Source Gedanken und stellt entwickelte Software grundsätzlich frei, solange es keine anderslautende Vereinbarung mit dem Kunden gibt. Im Gegenzug nutzt die educorvi GmbH & Co. KG Projekte und Komponenten aus anderen Open Source Projekten, die als Grundlage für die Softwareprojekte dienen.

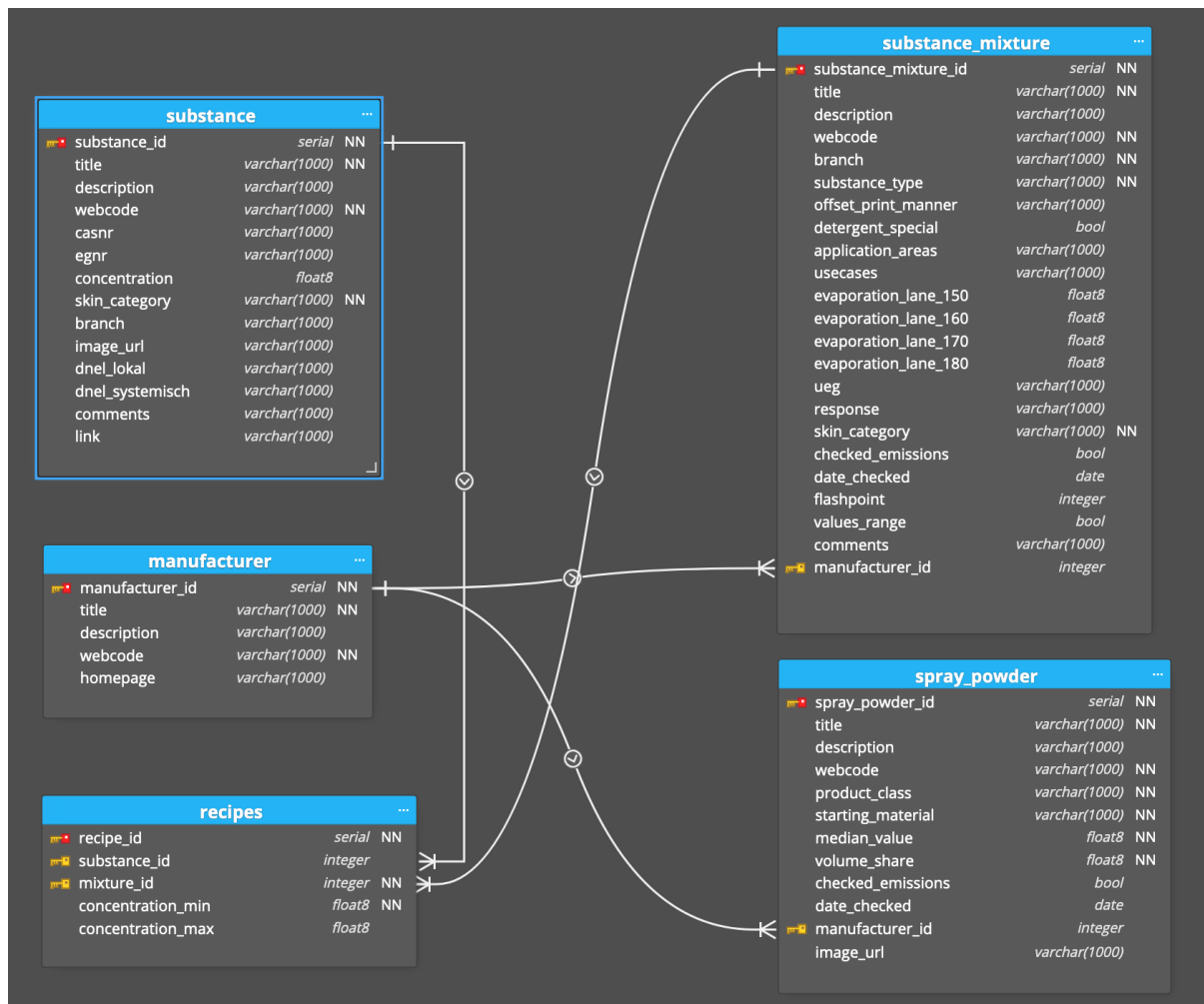
So kommt für die gesamte Software und die Dokumentation die MIT-Lizenz (siehe <https://de.wikipedia.org/wiki/MIT-Lizenz>) zum Einsatz. Für die MIT Lizenz haben wir uns gemeinsam entschieden, da sie Business- und Open Source-freundlich zugleich ist, und sie eine Verwendung durch Dritte ermöglicht, ohne deren gesamten proprietären Quellcode aufdecken zu müssen (eine Nennung in der Lizenzangabe genügt), gleichzeitig jedoch im Gegensatz zu GPL weiterhin eine Kommerzialisierung der Projekte ermöglicht.

Die Dokumentation in gedruckter Form (PDF Export), ist unter [CC BY 4.0](#) lizenziert. Diese Lizenz wurde gewählt, da Sie die Verwendung, Veränderung und Veröffentlichung durch Dritte (einschließlich die kommerzielle Nutzung) ermöglicht, jedoch eine Nennung des Urhebers voraussetzt.

# 12. Anhang

## 12.1 Anhang 1: Datenbankmodell

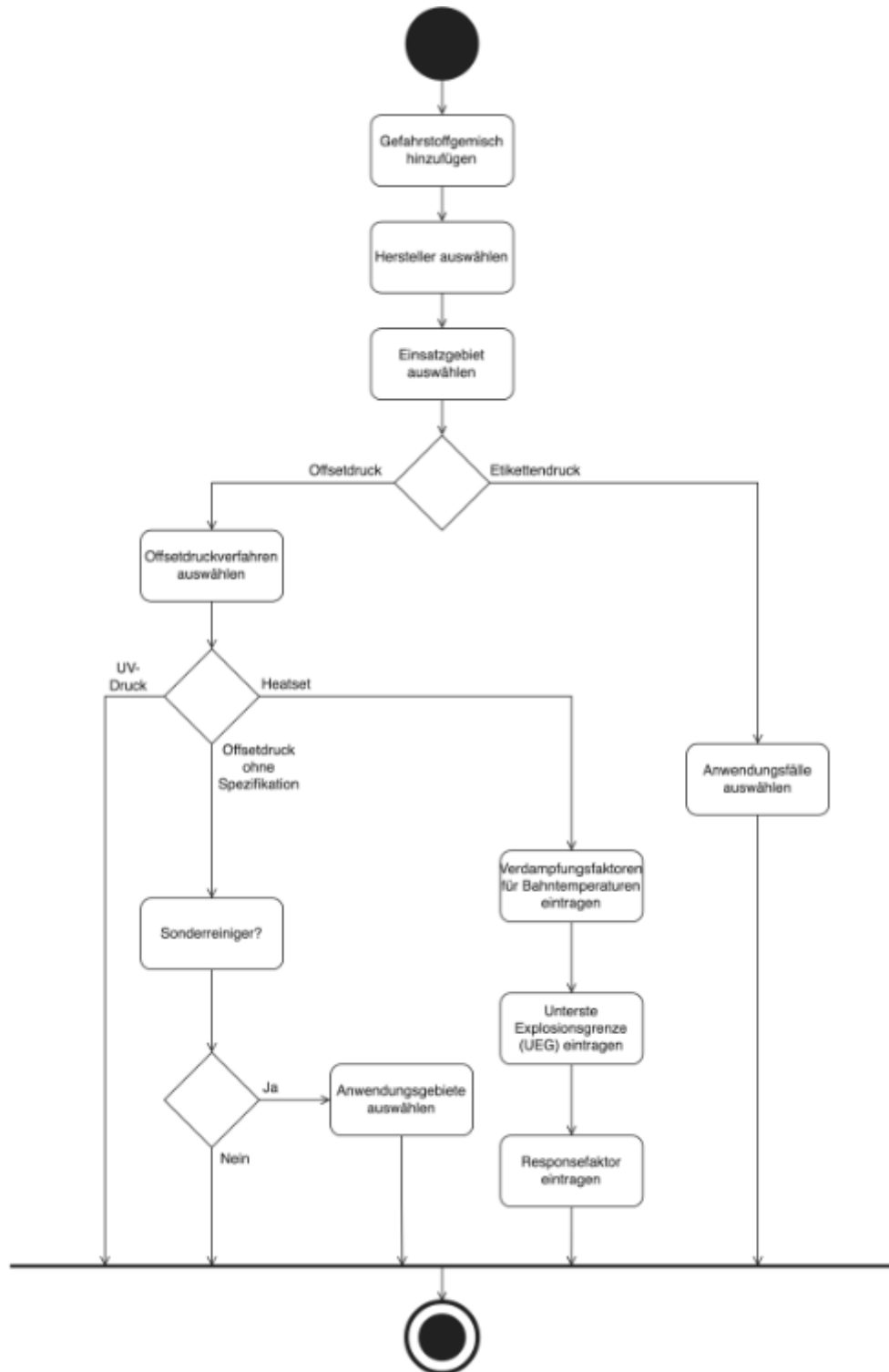
### 12.1.1 ERM Datenbankmodell



[https://doku.educorvi.de/wissensartikel/abbildungen-emissionsarme-produkte/datenbankmodell.png/image\\_view\\_fullscreen](https://doku.educorvi.de/wissensartikel/abbildungen-emissionsarme-produkte/datenbankmodell.png/image_view_fullscreen)

## 12.2 Anhang 2: Aktivitätsdiagramm

### 12.2.1 Aktivitätsdiagramm Gefahrstoffgemisch anlegen



[https://doku.educorvi.de/wissensartikel/abbildungen-emissionsarme-produkte/aktivitatendiagramm.png/image\\_view\\_fullscreen](https://doku.educorvi.de/wissensartikel/abbildungen-emissionsarme-produkte/aktivitatendiagramm.png/image_view_fullscreen)

## 12.3 Anhang 3 Funktionstests

---

### 12.3.1 Verzeichnis Funktionstests

---

Alle Funktionstests können unter der Domain <https://github.com/educorvi/edi.substanceforms/tree/main/src/edi/substanceforms/tests> vollständig eingesehen werden. Dieses Verzeichnis enthält alle Funktionstests, um die beiden Content Types Datenbank und Tabelle, sowie alle Views zu testen. Einzig die folgenden beiden Dateien im Verzeichnis sind nicht direkter Bestandteil der Tests, sondern sind für die Einrichtung des Testroboters notwendig, der die Tests dann automatisiert durchführt:

- test\_robot.py
- test\_setup.py

## 12.4 Anhang 4: Glossar

---

### 12.4.1 Glossar und Abkürzungsverzeichnis

---



<b>Abkürzung</b>	<b>Erklärung</b>
BG ETEM	Berufsgenossenschaft Energie Textil Elektro Medienerzeugnisse
CC	Creative Commons
CT	Content-Type (Inhaltstyp im CMS)
CRUD	Create, Read, Update, Delete
ERM	Entity–relationship model
IHK	Industrie- und Handelskammer
MIT	Massachusetts Institute of Technology
PDF	Portable Document Format
HTML	HyperText Markup Language
CSS	Cascading Style Sheets
PSQL	PostgreSQL
RDBMS	relational database management system —> Relationales Datenbankmanagementsystem
src	Sourcen- (Quellen) Verzeichnis
img-Tags	HTML image-Tags
(Web-) CMS	(Web-) Content Management System
Content Management System	Software, um digitale Inhalte zu erstellen, zu speichern und zu verwalten
Relationales Datenbankmanagementsystem	digitale, tabellenbasierte Datenbank, die auf der Grundlage des Konzepts der relationalen Datenbank, der Relation, beruht
PostgreSQL	Freies, Open Source relationales Datenbankmanagementsystem
Plone	Freies, Open Source Content Management System
View	Einzelansicht auf Element im Content Management System
Form	Webformular zur Erfassung von Daten, um sie anschließend weiterzuverarbeiten
Kanban-Board	Visuelles Hilfsmittel, das durch die Einteilung von Arbeitsschritten in ToDo/In progress/Done einen Überblick über den aktuellen Arbeitsstatus gibt und die Kommunikation im Team vereinfacht

**Abkürzung**

mr.bob

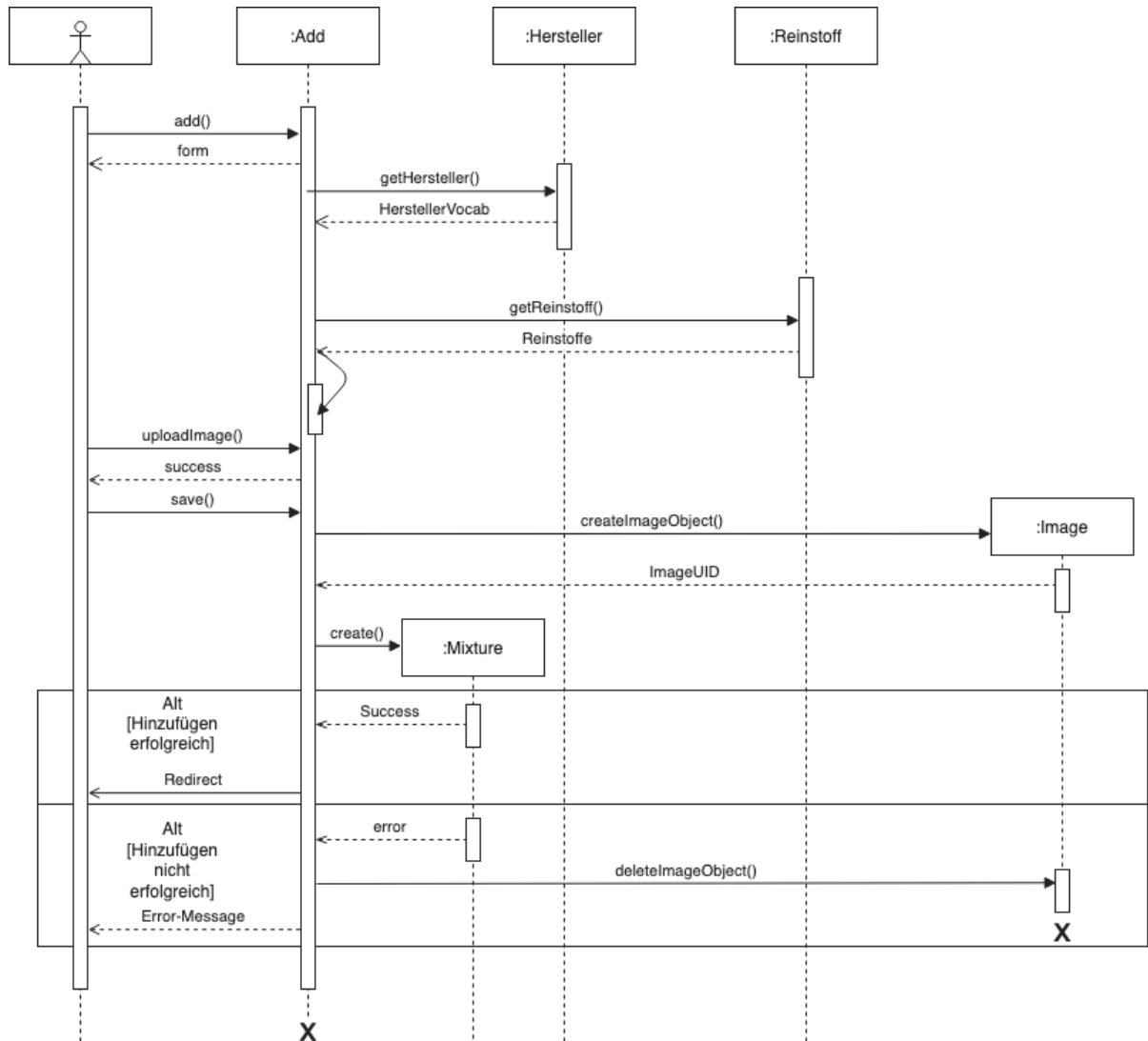
**Erklärung**

Werkzeug zur Generierung von Softwarepackages und weiteren Plone-Modulen mittels Schablonen (Templates)

---

## 12.5 Anhang 5: Sequenzdiagramm

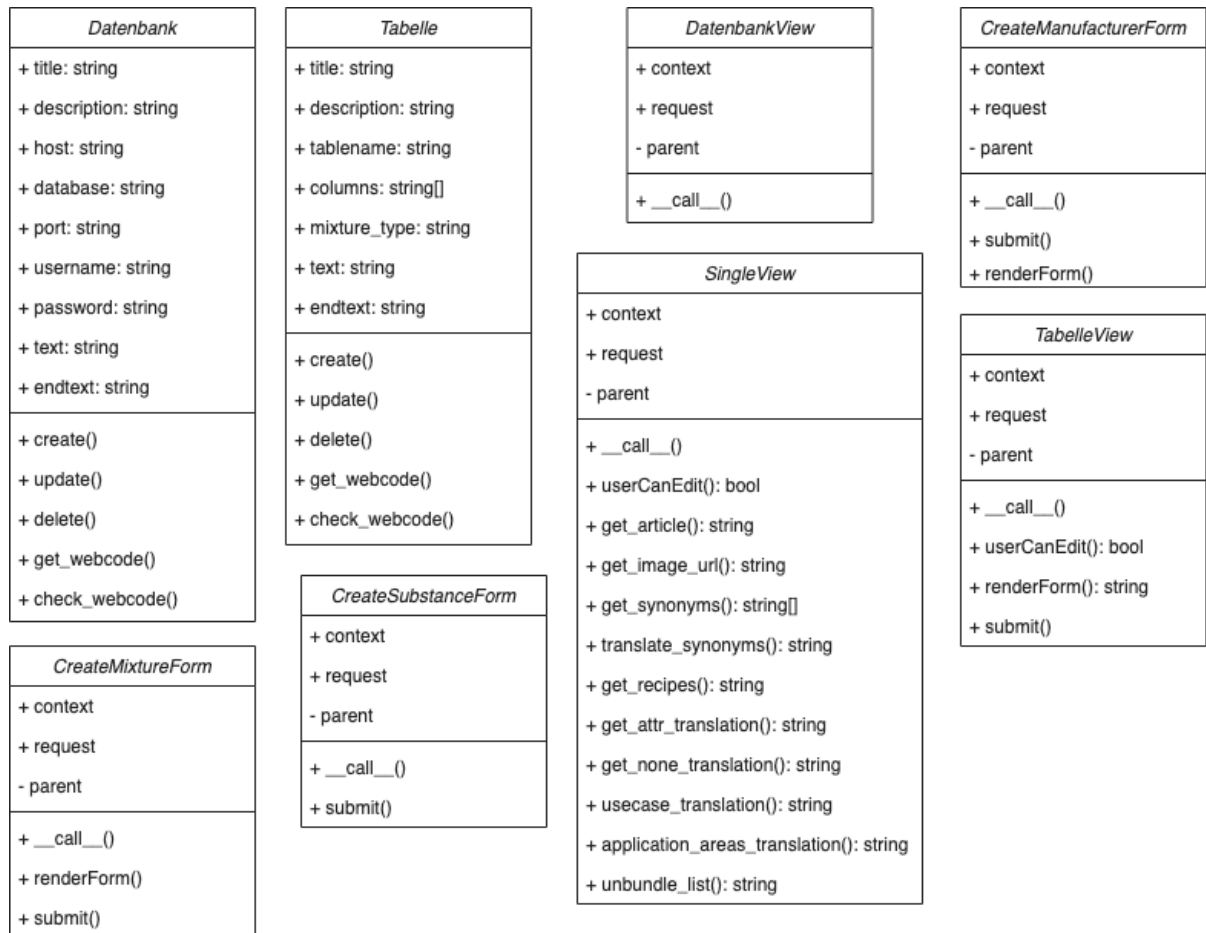
### 12.5.1 Sequenzdiagramm Bildbehandlung



[https://doku.educorvi.de/wissensartikel/abbildungen-emissionsarme-produkte/bildhandling.png/image\\_view\\_fullscreen](https://doku.educorvi.de/wissensartikel/abbildungen-emissionsarme-produkte/bildhandling.png/image_view_fullscreen)

## 12.6 Anhang 6: Klassendiagramme

### 12.6.1 Klassendiagramme



<https://doku.educorvi.de/wissensartikel/abbildungen-emissionsarme-produkte/klassendiagramme.png/>  
 image\_view\_fullscreen