Web Programmierung - SolCity

Ein Webprojekt von Dominique Helfer und Patrick Schläpfer

Inhaltsverzeichnis

[Einleitung 3](#_Toc406590070)

[Anforderungen: 3](#_Toc406590071)

[Umsetzung: 3](#_Toc406590072)

[Nicht umgesetztes: 3](#_Toc406590073)

[Versionisierung 4](#_Toc406590074)

[Datenbankanbindung 4](#_Toc406590075)

[Kurze Erklärung der Tabellen: 4](#_Toc406590076)

[sc\_user 4](#_Toc406590077)

[sc\_image 4](#_Toc406590078)

[sc\_article 4](#_Toc406590079)

[sc\_article\_preview 5](#_Toc406590080)

[sc\_category 5](#_Toc406590081)

[sc\_subcategory 5](#_Toc406590082)

[sc\_comment 5](#_Toc406590083)

[sc\_webcrawler 5](#_Toc406590084)

[sc\_webcrawler\_import\_log 5](#_Toc406590085)

[sc\_migration 5](#_Toc406590086)

[Migrationen 5](#_Toc406590087)

[Composer 5](#_Toc406590088)

[Grobentwurf Architektur 6](#_Toc406590089)

[Framework 6](#_Toc406590090)

[Entscheid 6](#_Toc406590091)

[MVC 6](#_Toc406590092)

[Webcrawler 8](#_Toc406590093)

[Grundaufbau: 8](#_Toc406590094)

[RSS Feeds verwalten 8](#_Toc406590095)

[RSS Import Status 9](#_Toc406590096)

[RSS Artikel freigeben 9](#_Toc406590097)

[Schlusswort 10](#_Toc406590098)

# Einleitung

Dies ist eine Zusammenfassung zum durchgeführten Projekt im Kurs "Web Programmierung". Zunächst werden alle Anforderungen aufgelistet und je nach Umsetzung kurz beschrieben. Eine weitere Beschreibung zum Projekt folgt auf späteren Seiten noch.

# Anforderungen:

* Benutzer registrieren
* Benutzer einloggen
* Artikel pro Kategorie erstellen
* Mobiles Design
* Twitter Einträge als Referenzen
* Webcrawler
* Artikel von Supervisor gegenlesen

# Umsetzung:

* Benutzer registrieren
* Benutzer einloggen
* Artikel pro Kategorie erstellen
* Mobiles Design
* Webcrawler

Da in unserer Gegend (Freunde und Bekanntenkreis) vermehrt Blogs wie Tumblr genutzt werden, haben wir uns dazu entschieden den Webcrawler optimal umzusetzen, welcher solche Blogeinträge direkt in unsere Datenbank und somit auf unsere Seite importieren kann.

# Nicht umgesetztes:

* Twitter Einträge als Referenzen

Im Projekt Webprogrammierung haben wir das Eintragen von Twitter Feeds ausgelassen, da Twitter in unserer Gegend nur sehr beschränkt genutzt wird. Doch haben wir uns stärker auf die Optimierung des Webcrawlers konzentriert.

* Artikel von Supervisor gegenlesen

Da jeder Blogger für seine Blogs selbst verantwortlich sein soll, haben wir das Gegenlesen eines Artikels durch einen Supervisor ausgelassen. Je nachdem wie viele Blogger später diese Seite nutzen würden, wäre dies aus Zeittechnischen Gründen auch gar nicht mehr möglich.  
Das Veröffentlichen der Beiträge, welche durch den Webcrawler importiert werden, haben wir jedoch ermöglicht. Dies aus dem Grund, da wir nicht alles Mögliche importieren wollen. Später könnte es jedoch zu einer Menge an importierten Artikeln kommen, welche auch wieder aus Zeitgründen nicht mehr gegengelesen werden können. In diesem Fall kann man alle Beiträge mit nur wenigen Mausklicks gleichzeitig veröffentlichen.

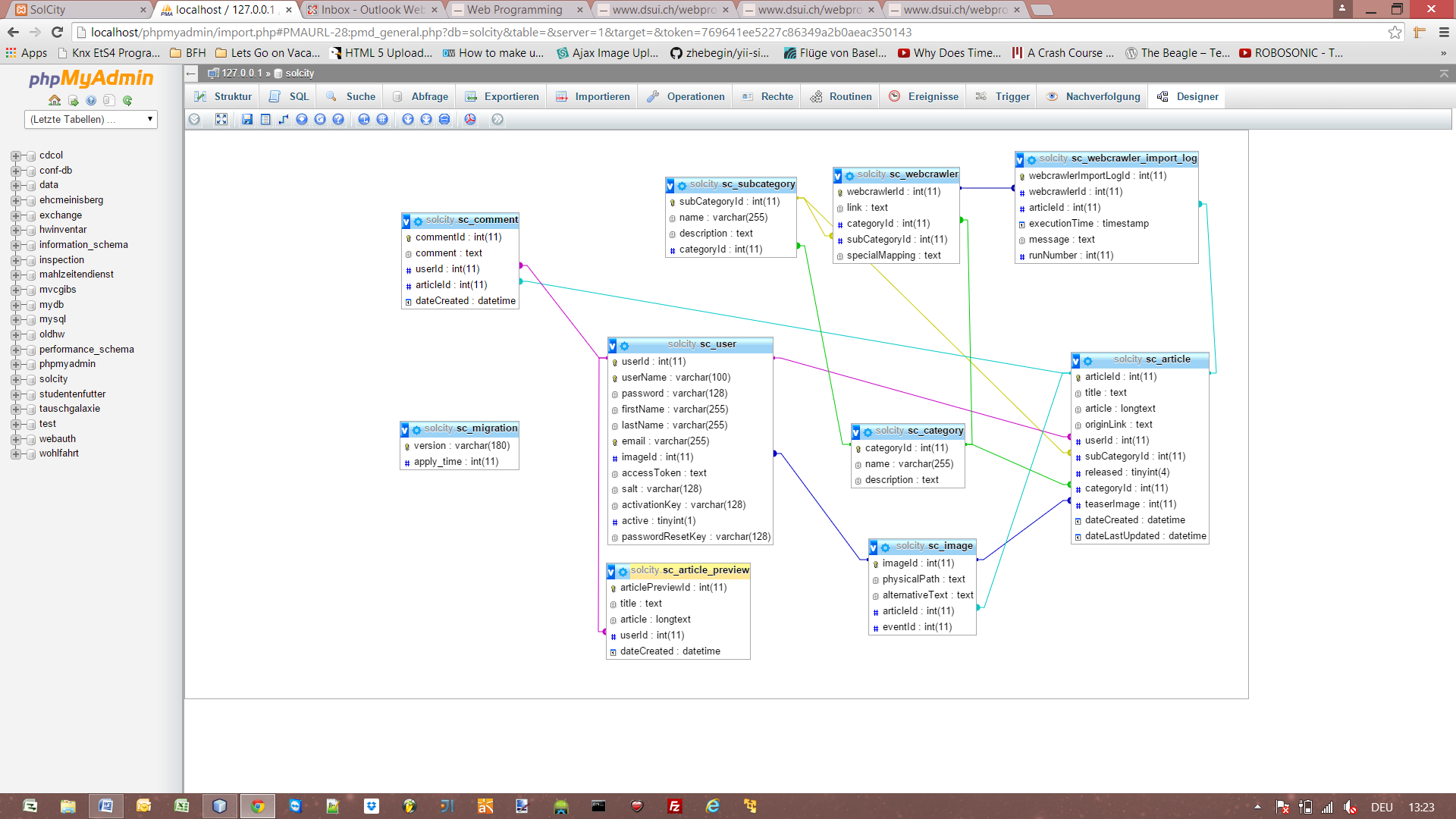
# Versionisierung

Da wir zu zweit an einem Projekt arbeiten, mussten wir für eine geeignete Versionisierung schauen. Da wir beruflich schon mit Git zu tun hatten, haben wir uns für diese Lösung entschieden. Eine andere mögliche Lösung wäre auch Subversion gewesen. Auf die effektiven Unterschiede sowie Vor- und Nachteile, werden wir hier allerdings nicht eingehen.

# Datenbankanbindung

Da unser Projekt über Persistente Objekte verfügt, wurde eine Datenbank angebunden. Dabei handelt es sich um eine klassische MySQL Datenbank, welche die Objekte darin speichert.

Unten ein ERD der Datenbank:



## Kurze Erklärung der Tabellen:

### sc\_user

In dieser Tabelle wird der Benutzer mit allen Eigenschaften gespeichert. Da der Benutzer auch einen Avatar hochladen kann, besteht eine Verbindung zur Tabelle sc\_image.

### sc\_image

In dieser Tabelle werden die Bilder abgespeichert. Jedoch haben wir keinen Blob verwendet, da sonst die Datenbank viel zu gross werden würde, sondern wir haben die Bilder lokal auf dem Computer oder Server abgespeichert und nur den Pfad zum Bild in der Tabelle abgelegt. Nebst des Bildes wird auch noch eine Artikel ID angegeben, falls es sich um ein Artikelbild handeln sollte.

### sc\_article

In dieser Tabelle werden die Artikel gespeichert. Ein Artikel befindet sich jeweils in einer Kategorie oder einer Subkategorie, daher besteht zu beiden Tabelle eine Verbindung. Zudem hat der Artikel einen Ersteller, welcher mit der ID des Benutzers gekennzeichnet wird. Ein Artikel kann auch über ein Vorschaubild verfügen, welches direkt mit dessen ID in der Artikeltabelle gespeichert wird.

### sc\_article\_preview

Diese Tabelle wird für die Vorschau eines gerade geschrieben Artikels verwendet, damit nicht gerade ein richtiger Artikel erstellt wird.

### sc\_category

In dieser Tabelle werden die Kategorien gespeichert. In einer Kategorie kann man eine Subkategorie oder einen Artikel erstellen. Wenn man weitere Subkategorien erstellt, so wird anschliessend in der Menüansicht der Webseite ein Dropdown dargestellt. Wenn dies der Fall ist und für eine Kategorie Subkategorien bestehen, so können direkt an die Kategorie keine Artikel erstellt werden. Dies wird progammatisch abgefangen.

### sc\_subcategory

In dieser Tabelle findet man die Subkategorien zu den jeweiligen Kategorien. An diese Subkategorien können dann lediglich Artikel gespeichert werden und keine weiteren Subkategorien

### sc\_comment

In dieser Tabelle werden die Kommentare zu den verschiedenen Artikeln gespeichert.

### sc\_webcrawler

In dieser Tabelle werden die Links, welche später durchforstet werden sollen, gespeichert. Dazu können auch gleich noch die Mappings gespeichert werden.

### sc\_webcrawler\_import\_log

In dieser Tabelle wird ein Logging des Webcrawler Imports gemacht. So kann genau nachvollzogen werden was importiert wurde.

### sc\_migration

In dieser Tabelle wird die Historie der ausgeführten Migrationen gespeichert.

## Migrationen

Damit man eine saubere Versionisierung hat, wurden für die Datenbank jeweils Migrationen erstellt. Dies wurde gemacht damit man nicht jedes Mal ein Dump and Reload der Datenbank machen muss. Diese Migrationen werden direkt von Yii zur Verfügung gestellt und sind für unser Verfahren daher sehr nützlich.

# Composer

# Grobentwurf Architektur

In diesem Abschnitt des Dokuments wird der Grobentwurf des Projektes und somit auch der Aufbau des Frameworks dargestellt

## Framework

### Entscheid

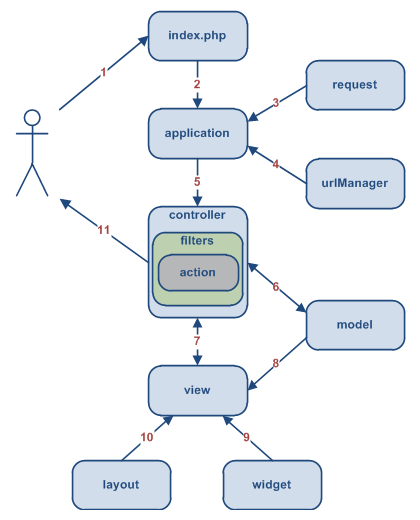
Die Anwendung wird unter Verwendung des Yii-Frameworks realisiert.

### MVC

Yii basiert auf der MVC-Architektur (**M**odel-**V**iew-**C**ontroller). Ein Controller versorgt dabei eine View mit entsprechenden Daten aus einem dazugehörenden Model.

**[[1]](#footnote-1)**

Yii-spezifisch betrachtet sieht diese Architektur folgendermassen aus:

**[[2]](#footnote-2)**

Die Anwendung wird immer über denselben Punkt betreten, nämlich über die index.php. Von dort wird die Applikation initialisiert, sowie die mitgegebenen Parameter verarbeitet.

Der entsprechende **controller** führt nun die angeforderte **action**, unter Beachtung der definierten Filter, aus, wobei vom **model** die Daten erhält, welche in der **view** dargestellt werden.

Die **view** verwendet ein spezifisches **layout** und bindet, falls definiert, verfügbare **widgets** ein.

# Webcrawler

Mit dem Webcrawler können Artikel von anderen Newsseiten wie auch von Bolgseiten direkt auf unsere Seite importiert werden. Dabei ist rechtlich allerdings festzuhalten, dass man die Artikel nicht komplett darstellen darf und immer ein Link zum Originalartikel vorhanden sein muss. Wenn dies so gemacht wird, dann gilt die Darstellung dieser Artikel quasi als Zitat und ist rechtlich abgesichert.  
Wer dann den Artikel komplett lesen möchte, muss auf die Ursprungsseite wechseln.

Der Webcrawler unterstützt das häufig verwendete Format der RSS. Fast jede News und Blogseite stellt einen solchen Link zur Verfügung. Wenn man nun diesen Link im Webcrawler speichert, so importiert er Artikel von dieser Seite.

Zum Webcrawler hat man allerdings nur mit einem speziellen Benutzer Zugriff:

Benutzername: SOLCITY\_RSS\_CRAWLER  
Passwort: 1234

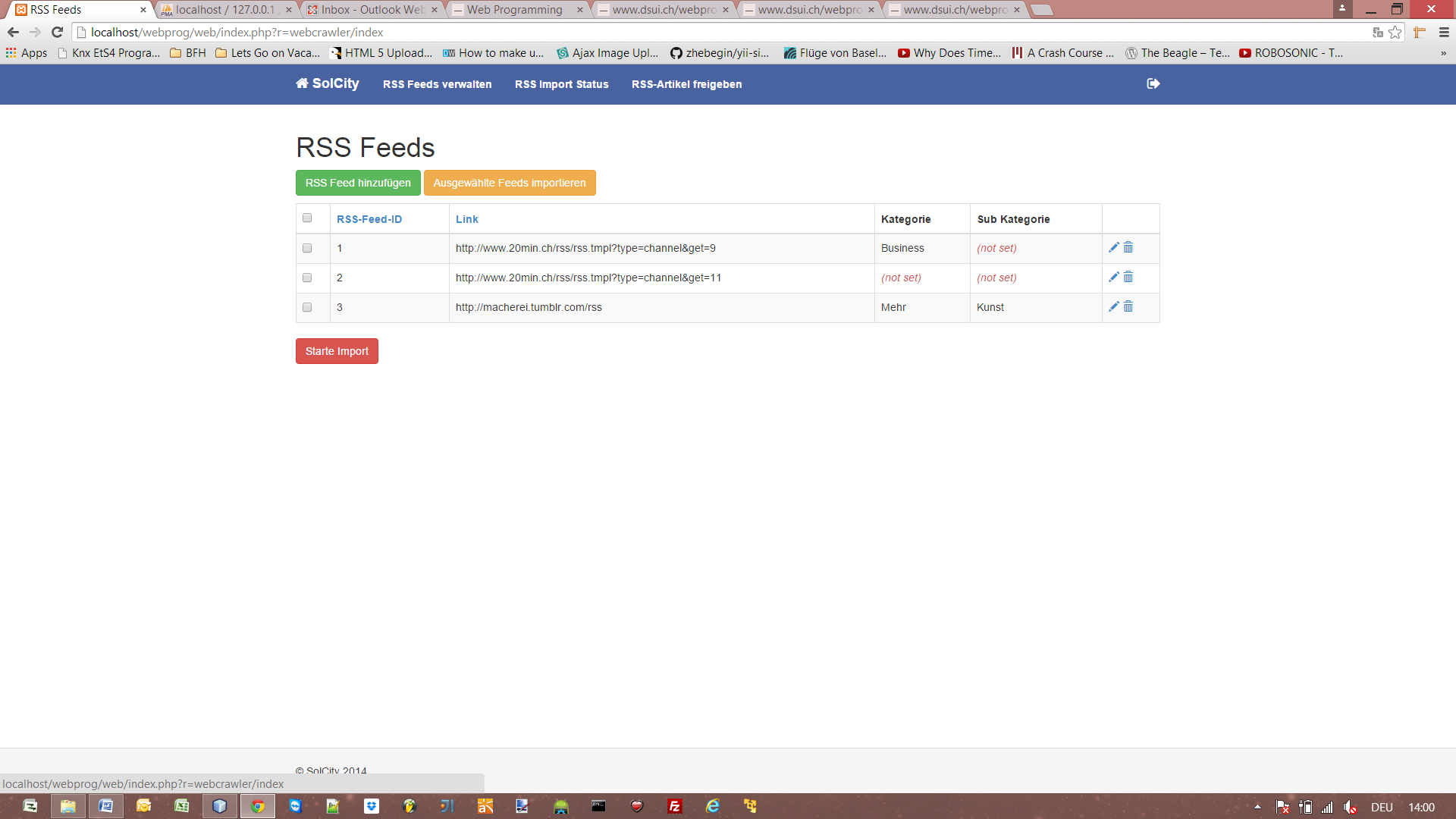
## Grundaufbau:

Wenn man sich mit dem Benutzer SOLCITY\_RSS\_CRAWLER einloggt, so kann man die RSS Feeds verwalten, einen Import Status abrufen und Artikel freigeben. Der Ablauf könne folgendermassen aussehen:

1. Es werden neue RSS Feeds hinzugefügt, welche anschliessend importiert werden sollen.
2. Man wählt die gewünschten RSS Feeds aus und starten mit dem gelben Button den Import. (Mit dem roten Import Button können auch gleich alle importiert werden)
3. Wenn der Import abgeschlossen ist, so wechselt es automatisch auf die RSS Import Status Seite. Auf dieser Seite wird nun der Status des Imports dargestellt.
4. Wenn alles in Ordnung ist, so muss man die Artikel für die Öffentlichkeit noch freigeben. Dies kann man gleich auf der Seite "RSS Artikel freigeben" machen.

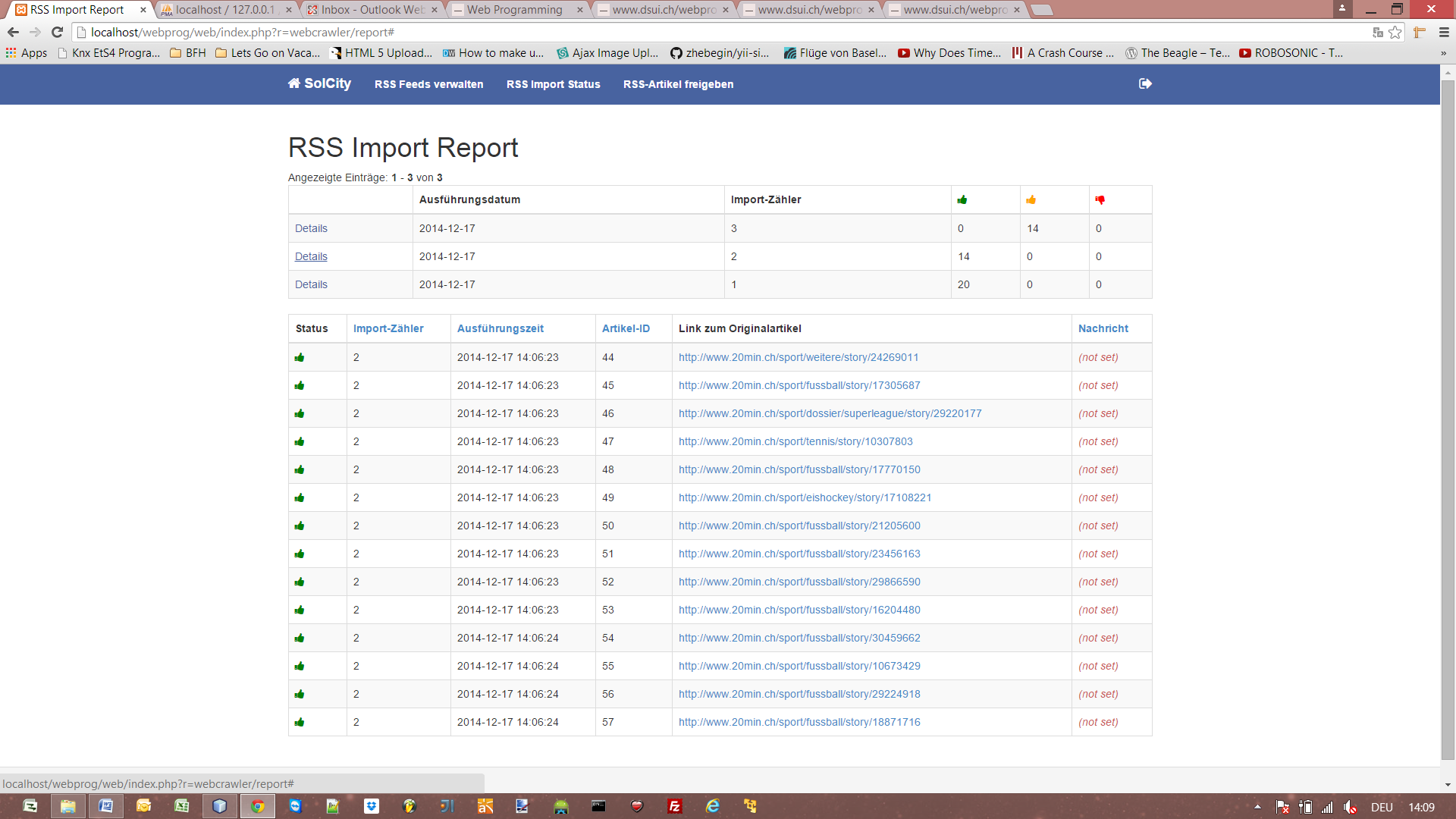
## RSS Feeds verwalten

Auf der Verwaltungsseite sieht man gleich alle RSS Feeds, welche dem Crawler schon hinzugefügt wurden. Diese Feeds kann man auch gleich bearbeiten oder löschen. Über den gelben Import Button kann man ausgewählte Feeds importieren.



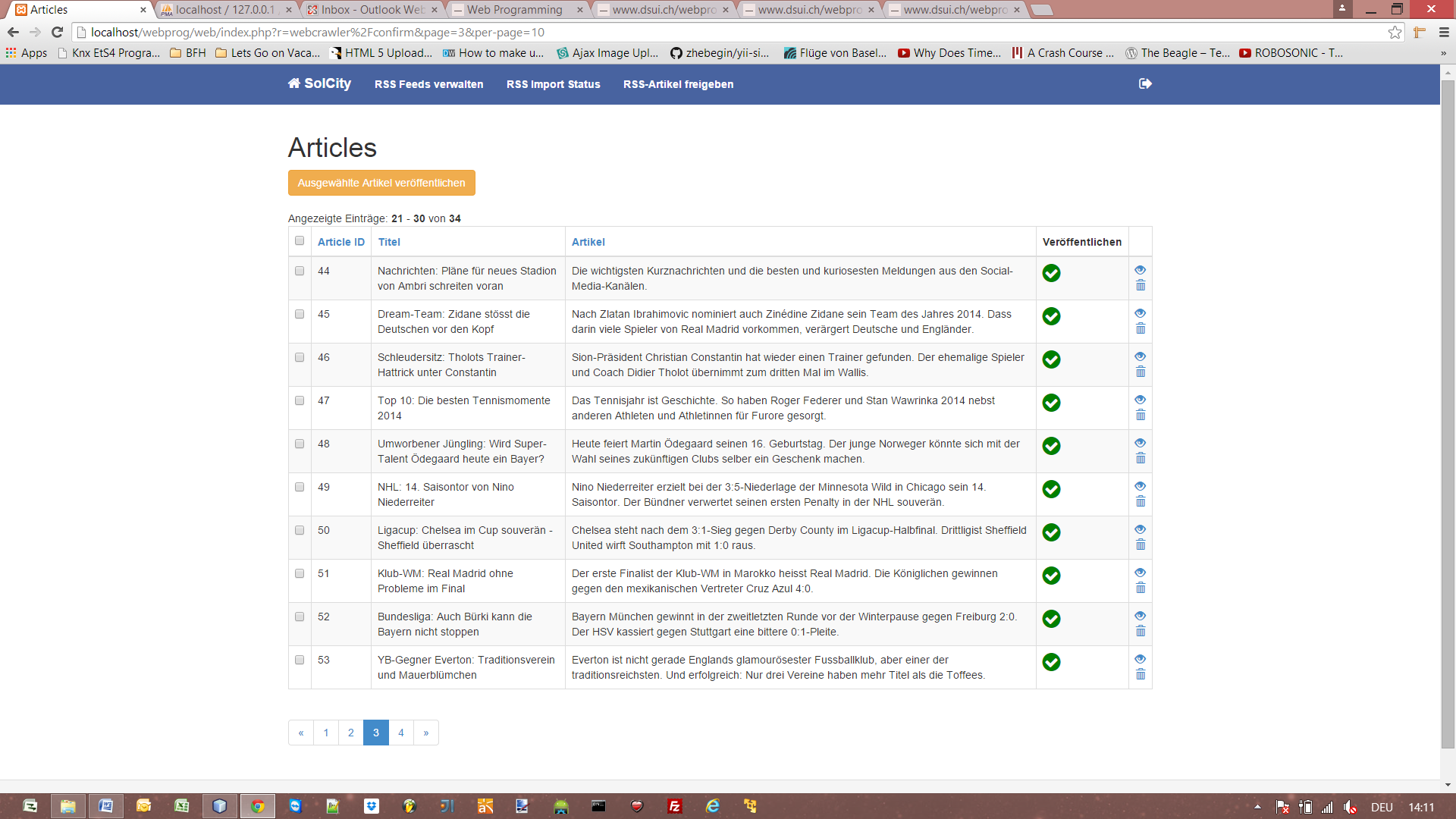
## RSS Import Status

Auf dieser Seite kann man nun die Imports überwachen. Man sieht welche Artikel importiert wurden, dann dies gemacht wurde und welche ID diesen Artikel jeweils zugeteilt wurden.  
Über den "Details" Button des jeweiligen Imports kann man diese Informationen ersichtlich machen.



## RSS Artikel freigeben

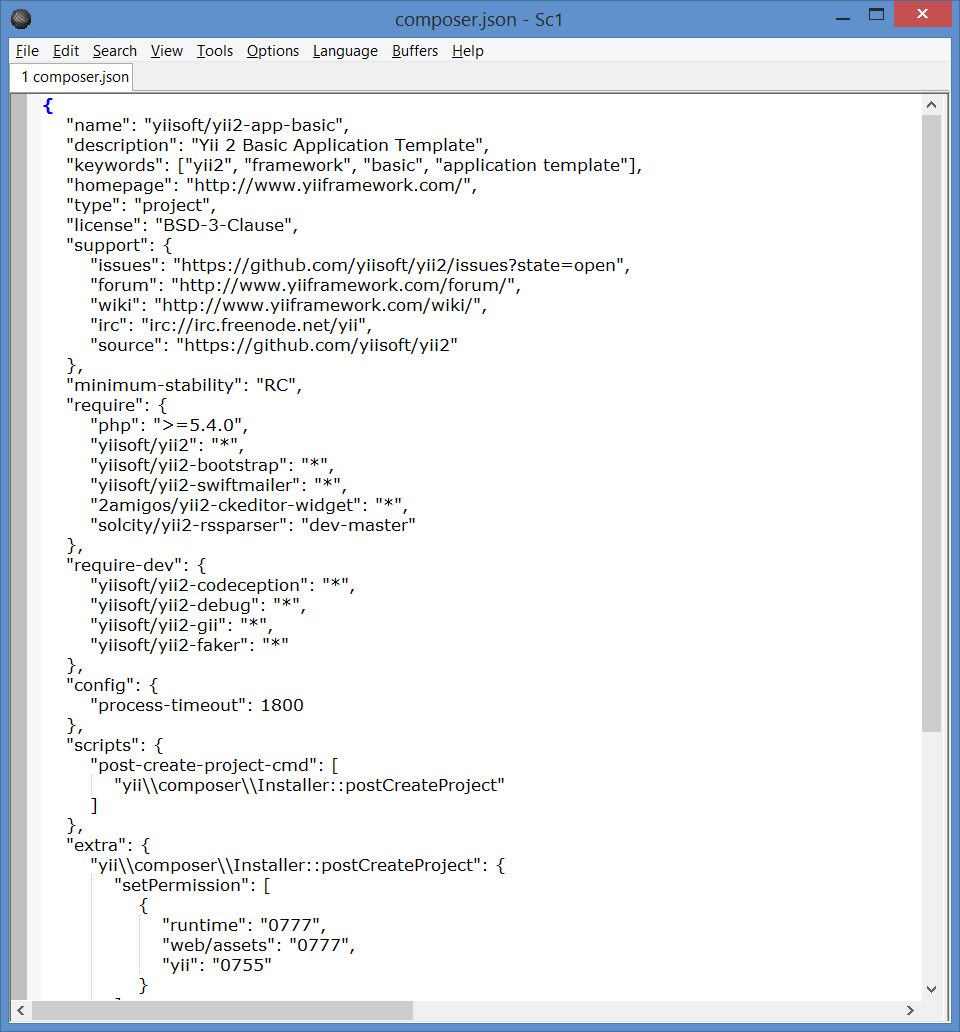
Auf dieser Seite müssen die Artikel nun jeweils noch für die Öffentlichkeit freigegeben werden. Damit hat man eine gewisse Sicherheit was auf der Seite alles publiziert wird. Mit den Checkboxen kann man die Artikel selektieren und über den gelben Button jeweils wieder veröffentlichen.



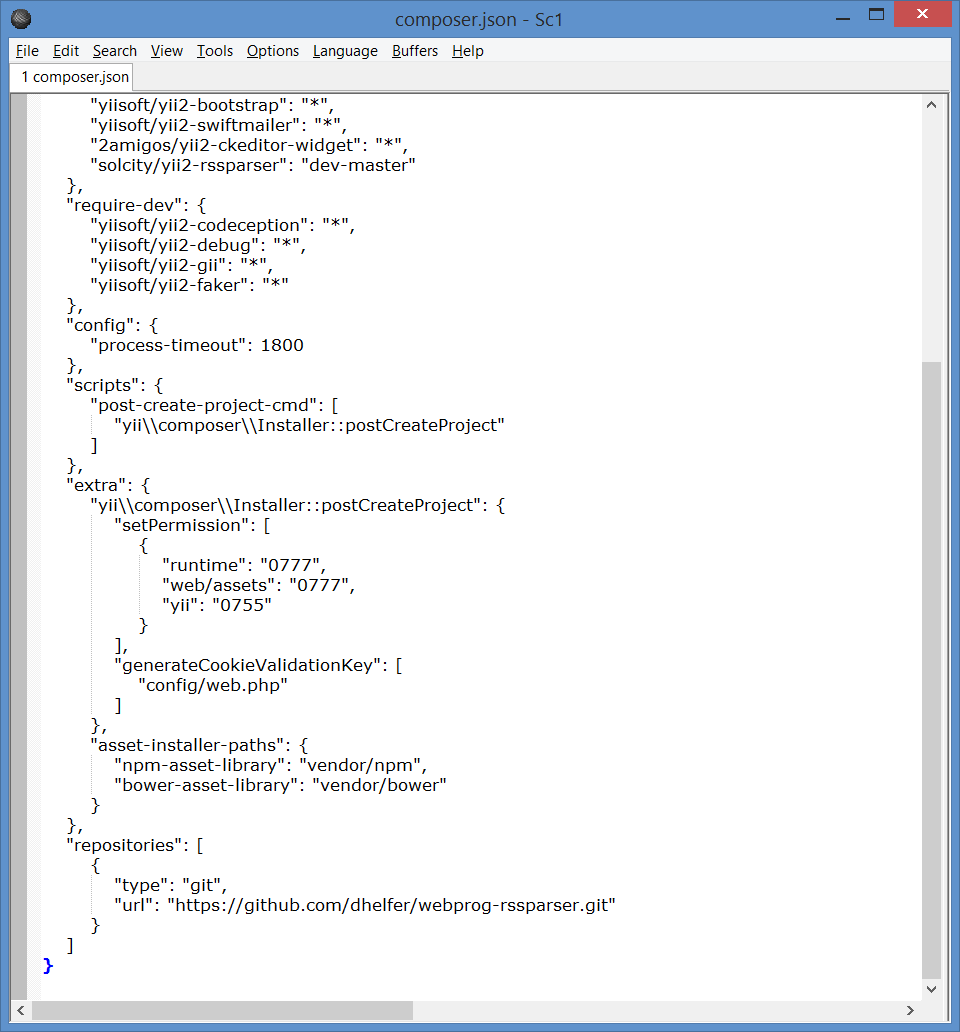
# Composer

Mithilfe von Composer können Packages, welche von einem Projekt benutzt und demzufolge benötigt werden, einfach installiert werden.

In einer composer.json-Datei wird in der root-Ebene des Projektes definiert, welche Abhängigkeiten des Projekt hat und von wo es diese benötigten Teilprojekte holen soll.



Beispielsweise der Webcrawler, welchen wir als Yii-Extension realisiert haben fügen wir dem Projekt mittels composer hinzu.

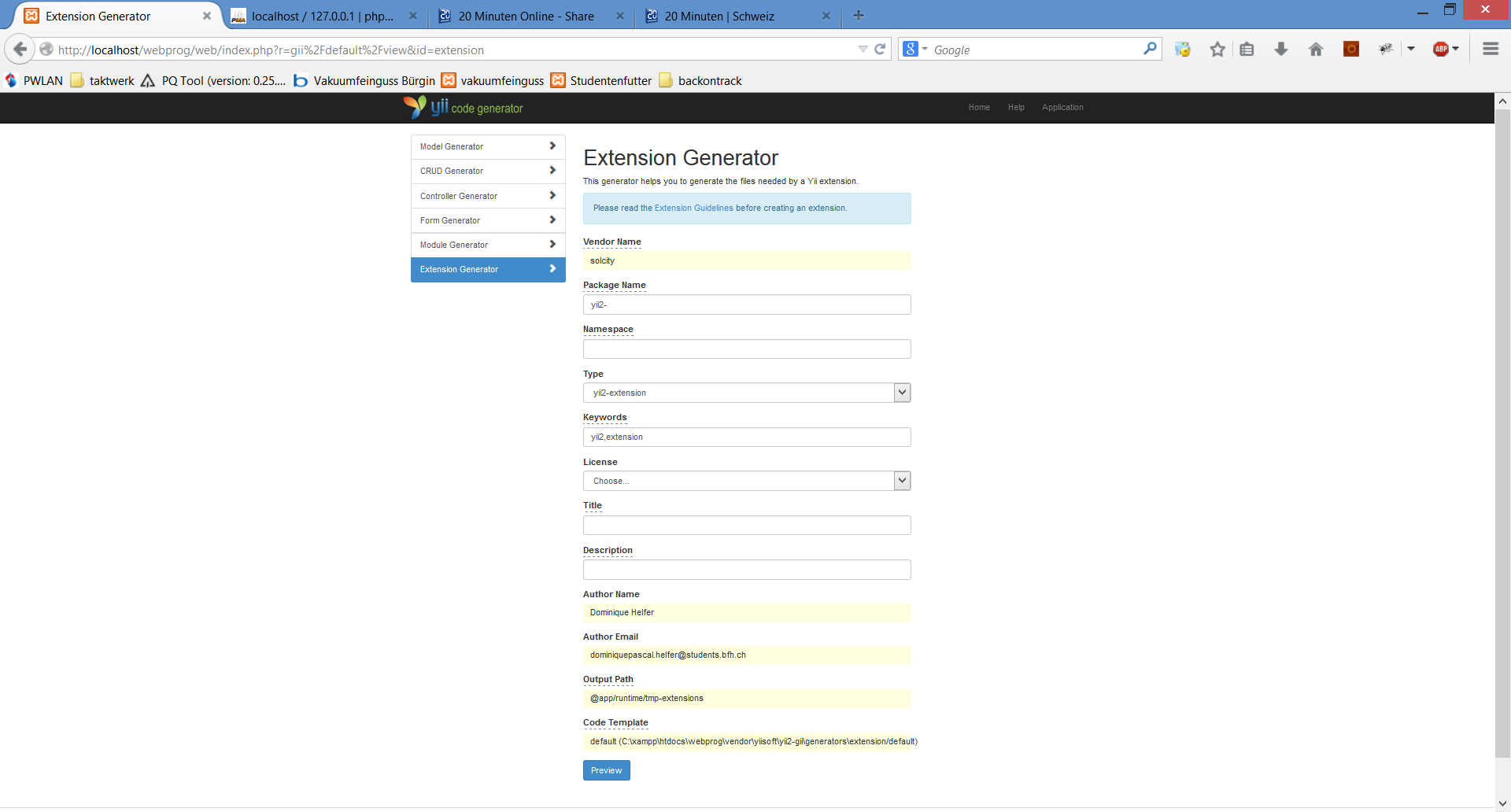


Bei dieser Extension handelt es sich um ein eigenes git-Repository. Dieser RSS-Crawler kann also auch ganz einfach von anderen Projekten benutzt werden. Auch ist es nun möglich, diesen Crawler bei Problemen zu bearbeiten und dann im Solcity-Projekt ein Update der Abhängigkeiten durchzuführen, um die neue, fehlerfreie Version einzubinden. Das Solcity-Projekt muss also in keinster Weise bearbeitet werden, sondern nur der Crawler.

Diese Modularisierung der Anwendung führt zu einer höheren Wartbarkeit.

# Extension

Yii bietet die Möglichkeit an, auf einfache Art und Weise Extensions zu erstellen.



# Schlusswort

Dieser Aufbau der Applikation wäre für eine Veröffentlichung durchaus denkbar und würde grundlegend auch funktionieren. Wenn man die Seite allerdings produktiv veröffentlichen möchte, so müsste man verschiedenste Funktionalitäten noch weiter ausbauen und absichern.  
Darauf würde anschliessend eine saubere Testphase wie eine Betaphase folgen um sicherzustellen, dass die Applikation funktioniert.

Was auch noch berücksichtigt werden müsste, wäre weitere Sicherheitsaspekte wie auch Performancekriterien. Diese wurden in unserem Projekt nicht vollkommen vernachlässigt, jedoch wurde auch nicht besonderen Wert darauf gelegt.

1. 1. <http://www.yiiframework.com/doc/guide/1.1/en/basics.mvc>

   [↑](#footnote-ref-1)
2. 2 <http://www.yiiframework.com/doc/guide/1.1/en/basics.mvc> [↑](#footnote-ref-2)