A red and white logo

Description automatically generated with medium confidence

Digisay Assessment

Name: Esmail Shabayek

Email: [esmail.shabayek@gmail.com](mailto:esmail.shabayek@gmail.com)

Phone number: +201097072480

### Overview:

Create a web scraping tool for collecting news articles from different news websites then store them in a database in a structured way to retrieve and control them through the admin dashboard.

### Credential

Live project: <http://15.237.50.25/>

Email: [admin@example.com](mailto:admin@example.com)

Password: 12345678

### Requirements

* PHP and [Laravel](https://laravel.com/) framework
* [Adminlte 3](https://adminlte.io/themes/dev/AdminLTE/index.html)
* jQuery
* Ajax
* [Aws](https://aws.amazon.com/) ec2 Nginx web server – MySQL – phpv7.4 – composer v2

### Database ERD Diagram

**Articles**: news that we scraped

Article title - Article image - Article description (Excerpt, Full content) - Created at - Article link - Website name.

**Websites**: Hold the websites we will need to pull data from.

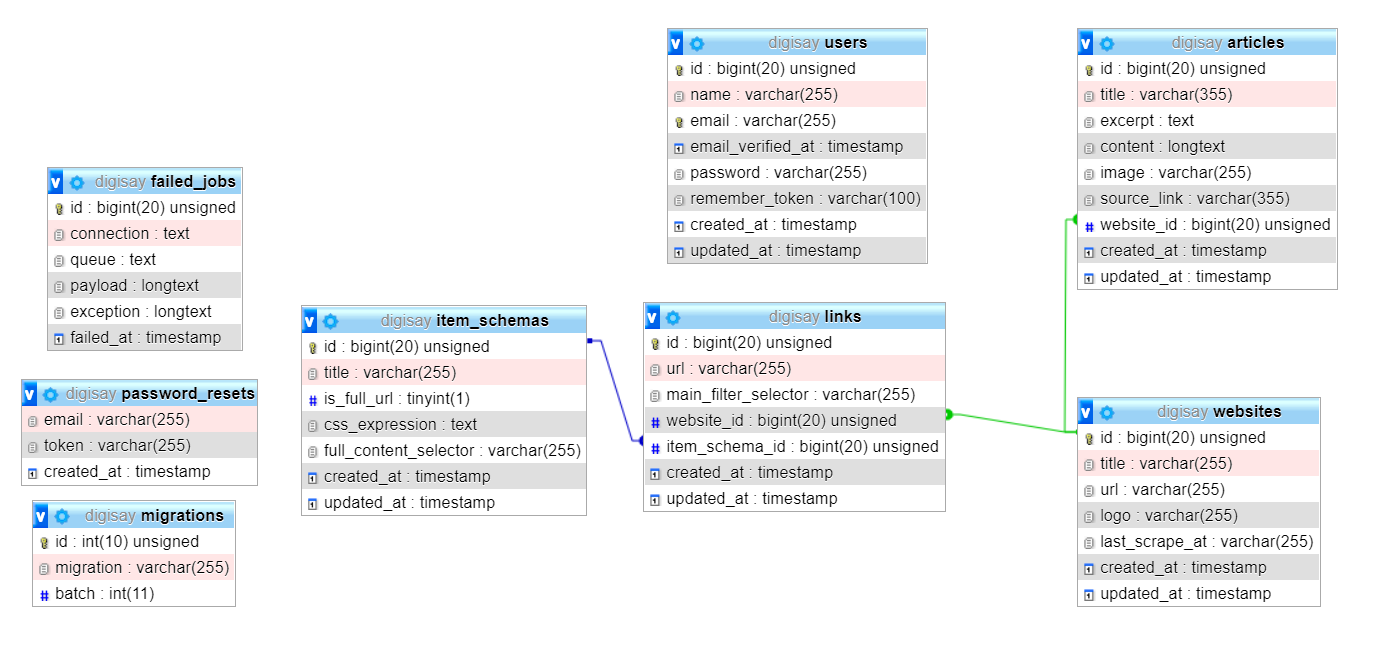
Website name - Website link - Created at - Last scraped at

**Links**: Hold the single links linked with the websites to pull data from.

**ItemSchemas**: This table define the schema for a single item in the list of items, it will contain the css expression to fetch individual pieces of data like title, excerpt, image etc.

**Users**: all admin registered

Name – email - password



**The website** table represent the websites that we will scrape data from.

**The links** table represent the links for specific website and we added category\_id to be assigned to, hence we say that link articles will be linked to the selected category id.The main\_filter\_selector defines the main css selector that will be passed to the filter() function so we say $crawler->filter(main\_filter\_selector)

**The item\_schema** table represent the schema structure for single items in an article list page, for example the article contain title, url, excerpt, image. the is\_full\_url defines whether the article uses a full url to the details page or partial url. the css\_expression attribute hold a special expression for all css selectors that represent those elements, as we will discuss this in the next sections. the full\_content\_selector defines the css selector of the item content in the detail page.

### Methods of Scraping:

* Regex (Regular expressions).
* CSS selectors.
* [XPath](https://www.hashbangcode.com/article/using-xpath-html-files-php) (a technique like querying with CSS selector).

### Scraping Process:

As shown in the figure below consider we have a list of articles, each article has image, title, excerpt and we need to retrieve each of those pieces.

so, using a technique like CSS selectors we can do something like this identified as pseudo code:

// this is just a pseudo code to clarify the process not actual code

foreach (<article> as node) {

$title = node.children('h2').text()

$excerpt = node.children('p').html()

$img = node.children('img').attr('src')

}

### PHP Goutte Package:

For the purpose of this tutorial, we will be using a php package called Goutte used in web scraping and this is the [GitHub](https://github.com/FriendsOfPHP/Goutte) repo for the package. This package internally based on symphony [dom crawler](https://symfony.com/doc/current/components/dom_crawler.html).

use Goutte\Client;

$client = **new** Client();

// create a crawler object from this link

$crawler = $client->request('GET', ' https://www.mklat.com/category/technology/computer-internet/');

// filter all elements that have class “post-item” and loop over them

$crawler->filter(‘.mag-box-container .post-item’)->each(function ($node) {

print $node->filter(‘h2.post-title’).text(); // get all h2 elements text inside each post-title class

});

The filter() function is one of the functions in symfony dom crawler, and it functions to filter elements by css selectors, there is also another function filterXPath() which filter elements with xpath expressions. So filter(selector) returns the selector if found as a crawler object or null of not found so that you can use that object to retrieve other nested elements

consider this sample html:

<ul>

<li class=”post-item”>

<img class=”wp-post-image” src=”url-image” />

<h2 class=”post-title”><a href=<http://url>>title</a></h2>

<p class=”post-excerpt”>excerpt</p>

</li>

<ul>

To get the first element

$crawler->filter(‘ul li. post-item’)->first();

To get element by specific index

$crawler->filter(‘ul li’)->eq(1);

To get children’s of an element

$crawler->filter(‘ul’)->children();

Refer to [symfony dom crawler](https://symfony.com/doc/current/components/dom_crawler.html) documentation to learn more about the other available functions.

### Preparing the Project:

We will build a news website where administrator can be able to add links for news websites and fetch articles with scraping.

### Preparing main layout

* Create the main layout template in resources/views/layouts/**app**.blade.php from [**adminlte**](https://adminlte.io/themes/dev/AdminLTE/index.html) template.
* Add resources/views/**dashboard** to add all module of admin panel
* Add crud blade for each module with name of function that refer it LIKE index for list, create for creating, edit for edit, form for all input fields

### Creating Controllers

* Create the required controllers **Websites Controller**, **Links Controller**, **ItemSchema Controller**
* Add the routes for those controllers in routes/web.php
* Modify app/Http/Controllers/**Controller**.php and add **uploadFile ()** function to upload image overall system.

### Classes

### WebScraper.php

**Actual job**: This class do the actual job of scraping which we will implement shortly to fetch and scrape news from a certain link.

**Functionality**: The main method in the above code is the **handle** () method. This method works on the Goutte client package. It takes a link object, creates a crawler object from the given URL.

Then it translates the CSS expression for the item schema attached with that link into an array of fields and their selectors with **translateCSSExpression** () method:

After we convert the expression into the array, we move into the filtering process passing the main filter selector to the **filter** () method, this will give us a collection of results we iterate over them using **each** () method, inside that function we get the different pieces of data.

Using the $node variable passed to the callback we can get and filter the sub elements we need to fetch such as titles and images. As a result, we looped over $**translateExpr** which is the translated CSS expression and return an array of $data to be saved into the database.

It’s better to put the code between try catch block as the fetching process may result in an error in any time due to many reasons like network loss or not found nodes matching the given expressions.

### Scrape Façade

**Actual job:** This class is custom Facades provide a static interface to a class that gives access to an object from the service container, let’s look at Laravel’s Custom Facades.

**Functionality**: getFacadeAccessor () method for a facade class to call WebScraper