Learn Laravel With a Fun Coding Challenge: Finstagram (Fake Instagram)

Posted on June 14, 2022

When someone asks me **how to learn Laravel**, the first thing I do is point them at the docs, Tighten's free Laravel educational site Onramp, or video sites like Laracasts.

But if they really get serious about it, I'll give them an assignment: **create an application yourself**. Don't follow a tutorial; plan a feature, then do the work to learn how to implement that feature, then do it over and over again.

Of course, you can create any application you want; but my favorite has always been an Instagram clone, because it offers so many features you can add on progressively, eaech of which require familiarity with a new part of Laravel.

Are you looking to learn Laravel? Or have you struggled to move from tutorial to real life? I challenge you: take the time to create your own Finstagram. It'll take you far.

The application

We're going to build an app called "Finstagram" (sort of like "fake Instagram"). It's sort of like Instagram, except for a few important differences:

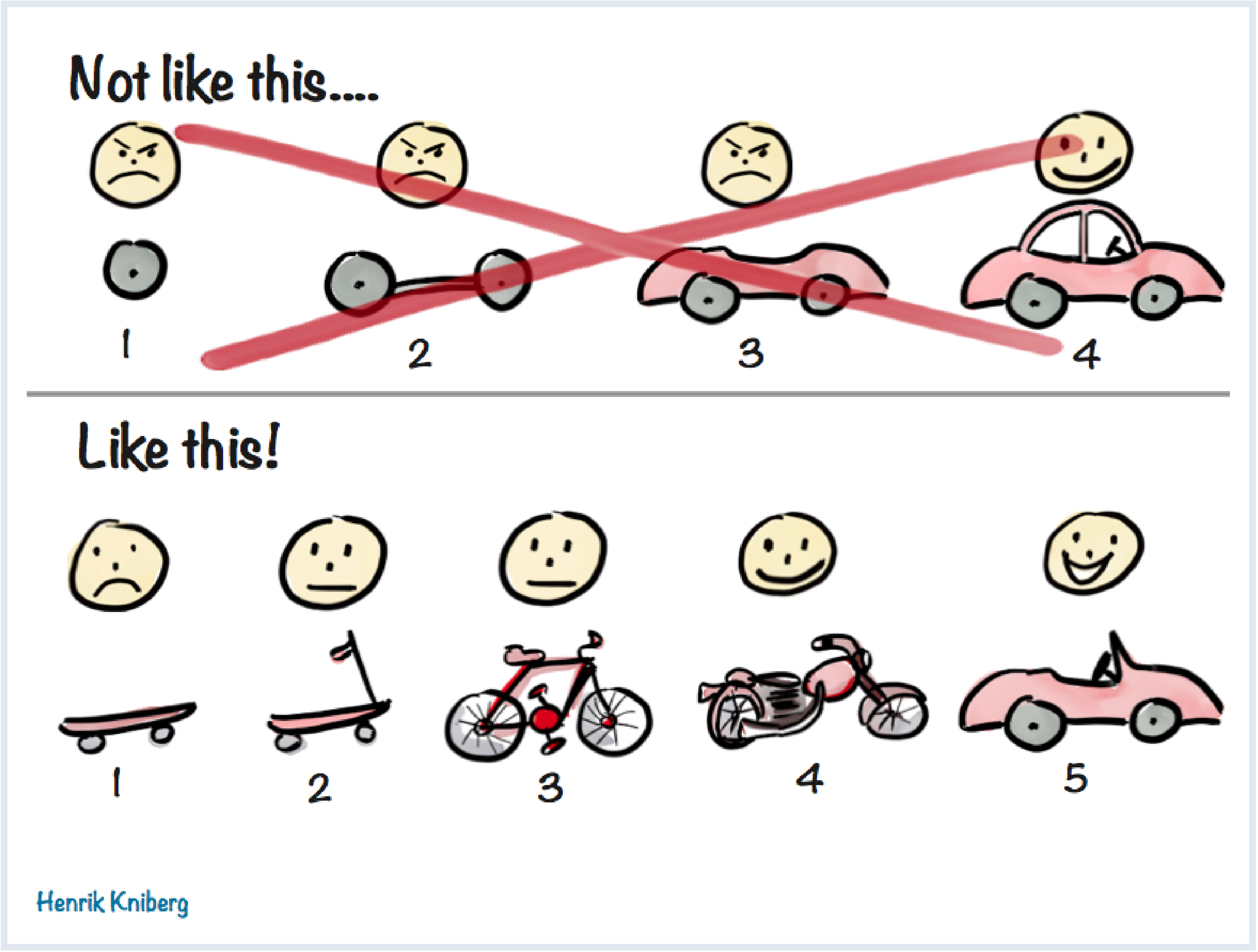
- It's going to start *very* simple and get better milestone by milestone—it won't be feature complete on day one
- It's going to be powered by Laravel with a light dusting of JavaScript (via Alpine)
- It's not going to give Mark Z. all of your personal data

The process

I will write out full tutorials for every step of the way *after this post*, but for now, I'm going to give you the same resources I've given every mentee I've sent on this challenge: A brief description of each milestone, and some resources for how to accomplish it.

The **milestones**, as I'm calling them, are more like iterations of a very simple photo-sharing web app as you build it slowly. It'll start as a rudimentary list of photos and move with each iteration toward a much more full-featured, Instagram-like tool.

If you're not familiar with this diagram, it's a great way to think about how to build any application iteratively:



Just like in this diagram, each of our milestones will be a *fully functional web app*. Each new milestone will add a feature or improve an existing one to make the app better and better with each release.

The milestones

Remember: each of these milestones will define *what you will have accomplished* once you've completed it.

The milestones will tell you the basic steps to complete them, but you still will likely have to do a big of digging to figure out all the details.

Milestone 0: Prep work

Goal

Create a GitHub repo and get a bare Laravel app in it; get that app's default Laravel home page showing in your browser. Nothing customized at all; just get a functioning web app that you can preview in your browser.

Learning

- Terminal/CLI
- Installing Laravel
- Git
- GitHub
- Composer
- Sail/Valet/Homestead

Steps

- Create a new project
- Serve your project locally using `php artisan serve` , and view it in your browser
- Create a new git repository and add all your files in the project to it; make a first commit
- Require *and install* Laravel Breeze; choose the default "Blade" stack
- Make a second git commit
- Create new repo in GitHub for this project and push your code up to it

Bonus goal

- Set up a more robust development environment using Sail (or, if you prefer, Valet, Homestead, or a similar tool)

Milestone 1

Goal

A simple HTML web site that has at least three images, each with a caption and a date and time, displayed top to bottom in a list, ordered most recently first. Straight HTML; no database, no JavaScript.

Learning

- Routing
- Blade templating

Steps

- Build the basic HTML for the simple page above in the editor of your choice
- Create a `posts` folder under the `resource/views` folder in your Laravel app, and create a new file in there named `index.blade.php`
- In that file, paste the HTML for the simple page descrived above
- Edit `routes/web.php` ; update the route definition for the home page (`Route::get('/')`) to point to the `index` view instead of the `welcome` view
- Test that viewing your web site (e.g. `http://localhost/`) now shows your HTML page

Bonus goals

- Use Tailwind to style the site instead of plain CSS
- Learn how Blade inheritance works by looking at `resources/views/dashboard.blade.php` and the docs' layout section; update your HTML page to use the `<x-app-layout>` component like the `dashboard.blade.php` file uses it
- Style the page to look a bit more like Instagram, with a little top nav and a constrained width and the content centered

Milestone 2

Goal

The same site as Milestone 1, but now the entries for each image (its image URL, caption, and date) are stored in a database, retrieved in the route definition, and passed to the view.

Learning

- Eloquent
- Migrations
- Seeders
- Passing data

Steps

- Create a `Post` Eloquent model and migration (`php artisan make:model Post -m`)
- Update the Post migration (`database/migrations/YYYY_MM_DD_create_posts_table.php`) to add three fields, all nullable:
  - A string field named `image_url`
  - A string field named `caption`
  - A datetime field named `published_at`
- Set up a local database server (if you're using Sail or Homestead, you already have a MySQL server running; if you're using Valet or `php artisan serve` , check out dbngin or Takeout)
- Create a database with the same name as the `DB_DATABASE` value in your `.env` file
- Run your migration (`php artisan migrate`)
- Put some real data in there (fake captions, fake dates, and fake image URLs using something like placeholder.com); you have three main options:
  - Create seeders (bonus goal)
  - Connect to your database with a tool like TablePlus and manually add data in there
  - Use Tinker to create instances of the `Post` model, fill them with data, and then save them to the database

Bonus goal

- Create seeders that automatically populate captions, dates, and image URLs so you can get real data with a simple seeder step

Milestone 3

Goal

Allow the user to click on each image, taking them to a fully separate URL for that post and its details.

Learning

- Routing
- Controllers

Steps

- Create a new template in `resources/views/posts/show.blade.php`

- Add a new URL definition to `routes/web.php` that represents the route to show an individual post
  - Set this URL definition to expect URLs like `http://localhost/123` where `123` is the ID of the database entry for that post: `Route::get('post')`

- In the definition for that route, first get the database entry for that Post (`Post::find($post)`) and then pass it to the view as a variable named `$post`

Bonus goal

- Use controllers instead of just defining the routes in the routes file.
- Use route model binding instead of looking up the post manually

Milestone 4

Goal

Allow anyone (no need for user accounts yet) to upload an image and a caption to the web site. When they upload the picture to the database, it's associated with the date and time they uploaded it.

Learning

- Form input
- Form validation
- File uploads
- Creating database entries

Steps

- Todo

Bonus Goal

- Validate stuff
- Store the files in S3 or a similar data store instead of the local filesystem Milestone 4 bonus: store the files in S3 or a similar data store instead of the local filesystem Learning: forms, file uploads, Eloquent

Milestone 5

5. Milestone 5 adds user accounts; link to the user signup and login pages, associate an uploaded image to the currently logged in user, and require a user to be logged in before they upload an image Learning: authentication, breeze, ACL/authorization

Milestone 6

6. Milestone 6 adds likes; add a button to each image and allow a logged in user to like it Learning: Eloquent relationships

Milestone 7

7. Milestone 7 adds processing the image; crop the uploaded image using Intervention Image Learning: Image processing

Other

Add testing in here somewhere todo

Optional Milestones

Want to keep going? Pick your poison!

- Convert the like button to be a Javascript trigger that calls an internal API endpoint instead of forcing a page refresh Learning: APIs, JavaScript
- Convert the image uploader to operate on a queue so the user doesn't have to wait Learning: queue
- Add comments Learning: forms, Eloquent relationships, validation
- Cache the results from the database queries to make the page load faster

Share:

Facebook Twitter LinkedIn

Comments? I'm @stauffermatt on Twitter

Tags: laravel

Subscribe

Quick links to fresh content, and more thoughts that don't make it to the blog

Your email

SIGN ME UP!