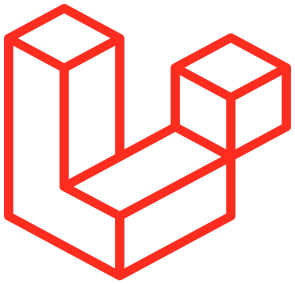


Social network with laravel

by Bernat and Luis

Competence analysis

Instagram

The instagram website presents a similar customer journey as the one introduced in the project . The main page is the login / register .

When the user is logged in, it is redirected to the home page where he or she can visualize his or her posts and the posts made by friends in a chronological order. The following things dragged our attention and gave us insight on how to approach the project.

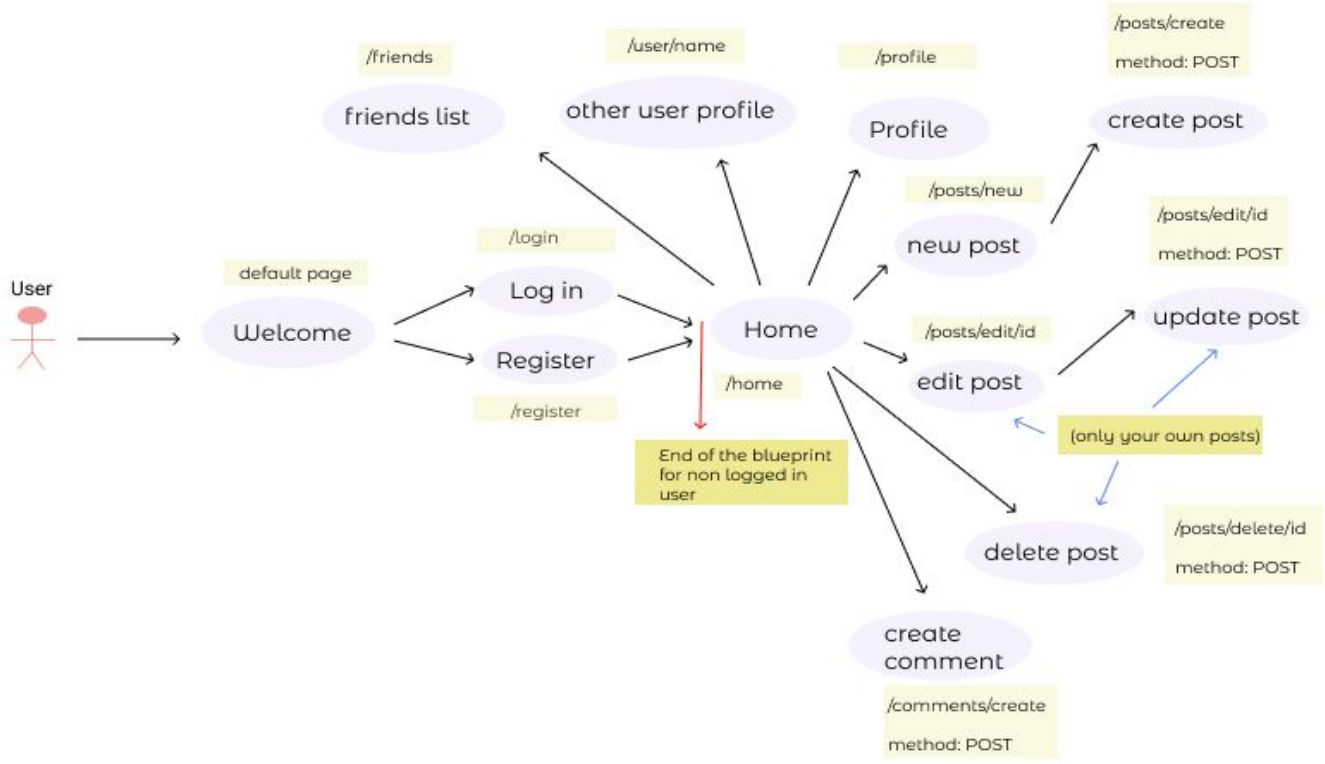
- Posts loading dynamically as the user scrolls down.

- Posts have a clear and rather simple structure, the actions one can perform have an icon that already tells about their usage. Such designs favour the user experience and are easier to maintain, as they follow a structured pattern .

- User friendships. Instagram follows a relationship structure between users that we did not completely engage on, as we found it a bit complicated. Instead, we followed the facebook pattern, for which the relationships are mutual, meaning that instead of following or being followed you establish a bidirectional friendship between another user.

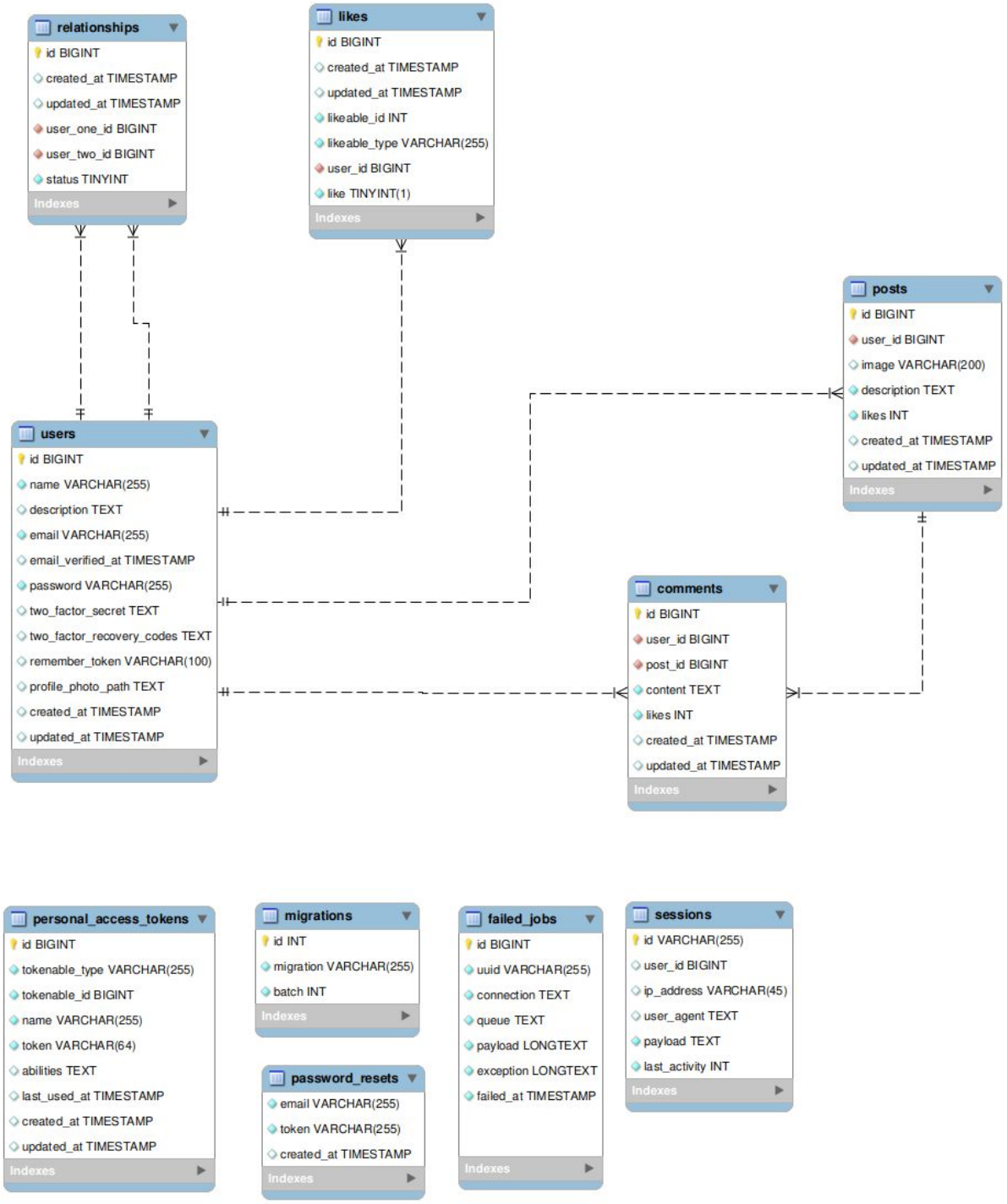
- Visiting other user's pages. When visiting another user page, the view that will be shown depends on the relationship we have with the user. If the friendship request has been accepted, you will be able to see the most complete version of the user profile. On the other hand, if that user does not appear as one of your friends, the view will be limited, displaying only a few information. Last , if the user blocked you, you won't be able to see anything. We took that idea and implemented it in our project, as it is a good solution when establishing the dependency between user relationships and user views .

Use case diagram



The diagram shows all the possible sites the user can visit. If nothing is stated, the method to access the site is GET . On top of each page, we have written the uri for which the page is accessed. The red arrow marks the end of a user journey that does not log in or register . From that point on, all sites have a middleware that checks user credentials. Comparing the initial diagram with the end result, we have created all the endpoints and pages stated here.

UML database diagram



The diagram in the top shows the end result of the database structure. Two relationships do not appear to be reflected here, since we use polymorphic relationships. In order to relate the likes with the comments and posts, we have a column called ‘ likeable id’ and another one called ‘likeable type’ . The first stores the primary key for comments or posts, depending where the likeable type column points at. Unfortunately , only the framework is aware of such relationship and thus, does not appear on the diagram

Wireframes

When designing the social network website, we took into account two facts. We knew that spending too much time in designing and wireframing would not bring much to the end result.Because of the fact of being beginners with laravel , it was difficult to foresee the problems we would have along the way. Thus, it brought much more value to us to make a really simple wireframe and deal with further usability problems along the way. The diagram below shows a schematic wireframe of some of the pages our site contains.

0 Welcome page

WE SHARE

log in | register

1 Login Page

username

password

cancel | log in

user logged in

3 Home Page

search | logout | notifications | profile

luis posted .. | ...

edit
delete

like | comment | share

+

4 Profile Page

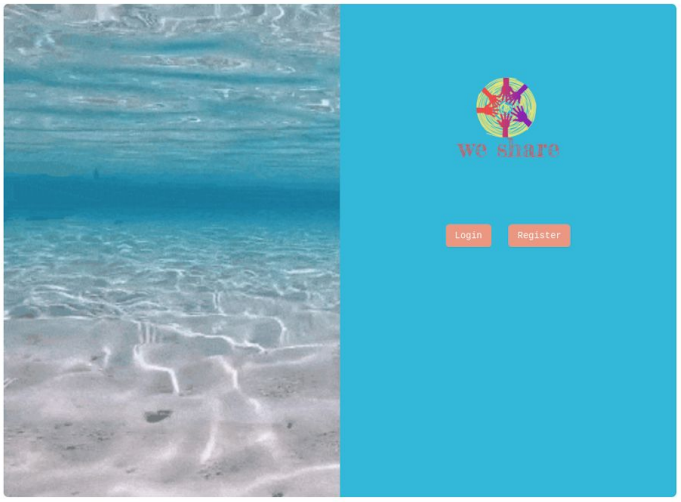
SEARCH

FOTO

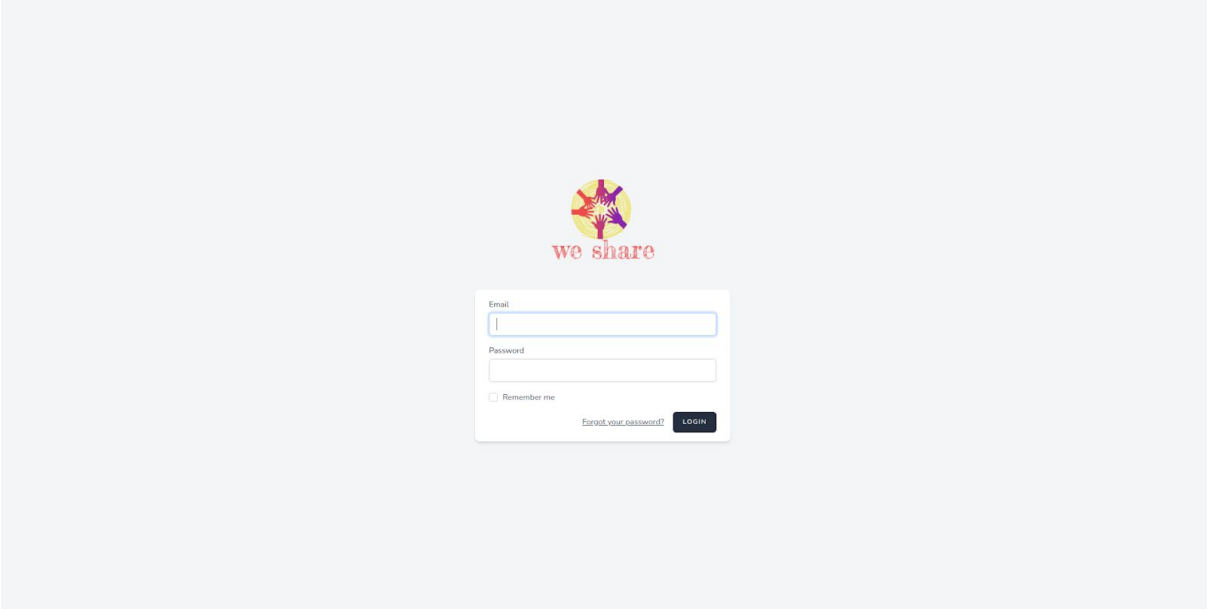
Comparison with the final result

Although the final result does resemble the wireframe in some way, it has key usability features that came along the way. We can say that the final result greatly improves our initial sketches. Below we can see some frame screenshots.

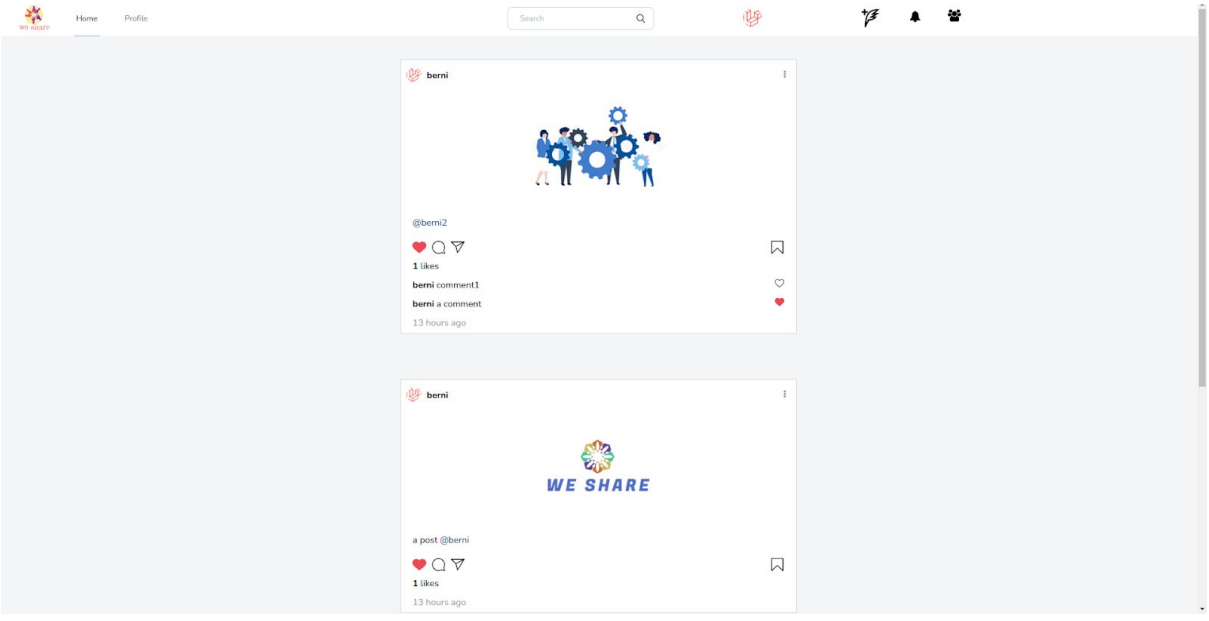
Welcome page



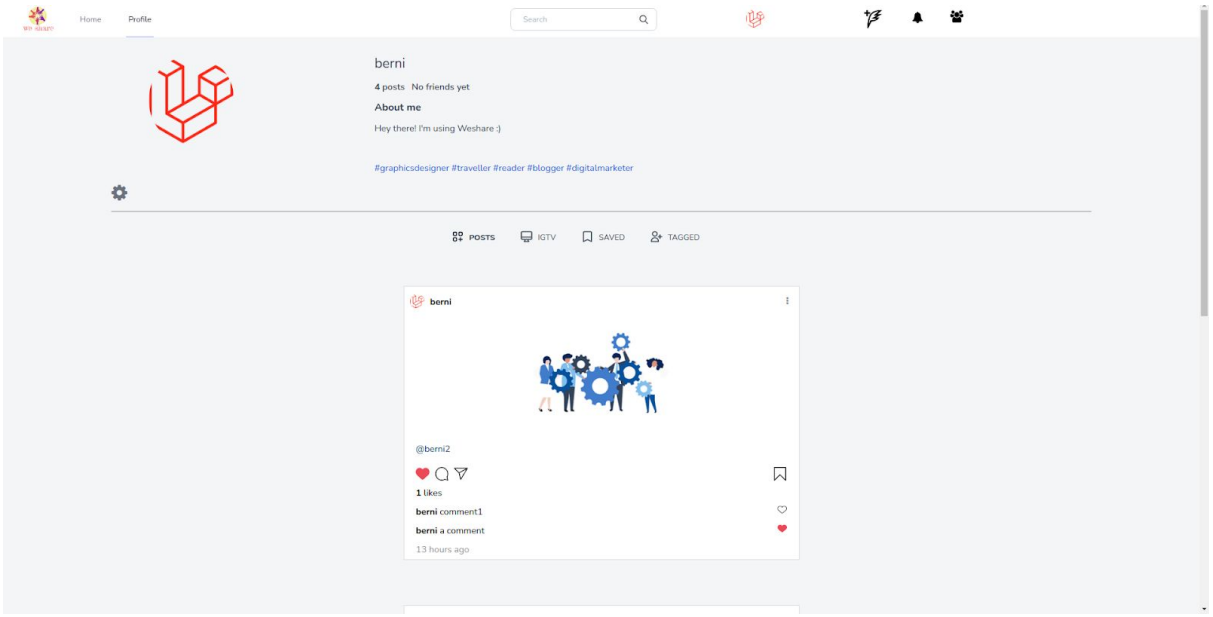
Login page



Home Page

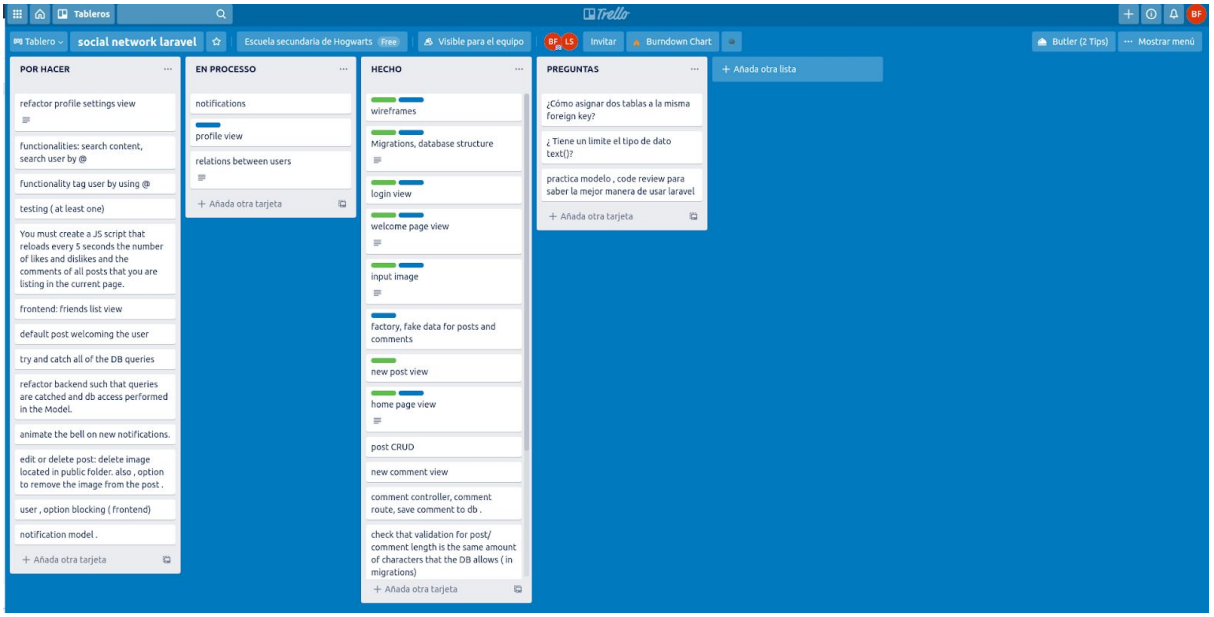


Profile page



Organization , task division

We organized the tasks by using a trello and dividing those between ‘ TODO’ , ‘IN PROCESS’ and ‘DONE’. Moreover, each task had its subtasks, defined in the card description . When picking a task, we would attach our name at the top of it and with a given color, so we would keep track of who was performing each task, which ones were done and which were still pending . We did a lot of pair programming, especially in cases where several parts of the program were involved. We did the frontend and backend at the same time, as developing the frontend would give us more insight on how the data should be retrieved from the backend, and developing the backend would give us clues on how the data should be displayed. Such feedback helped us a lot in the development process. Here you can find a snapshot on how the trello dashboard looked like.



Lessons learned

- Catch SQL query errors and display an user friendly message.
- Separate the views in components to be used in different parts of the project.
- Usage of middlewares to protect and verify the user data, as well as making sure the user only interacts with material he/she has access to .

- Dynamic loading of posts by using the scroll event listener in javascript.

- Creation of custom classes with useful functions available throughout the project.

Problems encountered

- Problem when storing the images uploaded by the user on the internet by using the imgur api. Solved by finally storing the images in the public folder of the project

- Problem when trying to change default jetstream routings, as those are defined on the vendor folder .

- Several problems related with task division and task definition, as we realized some functionalities were required once we already started the development phase of the project.