University Project – Web Application April

By Damyan Kehayov and Aleksander Slavov

# Project Task 1

The first task of this project was to separate the given example application into micro-frontends. To do this we first examined the given example and tried to structure it into how we thought it would be working, frontend and backend wise.

## Micro-FEs

We identified the following Micro-FEs:

To qualify as a micro-frontend, any of the following must at least be able to be deployed independently from the entire application and be used in other applications.

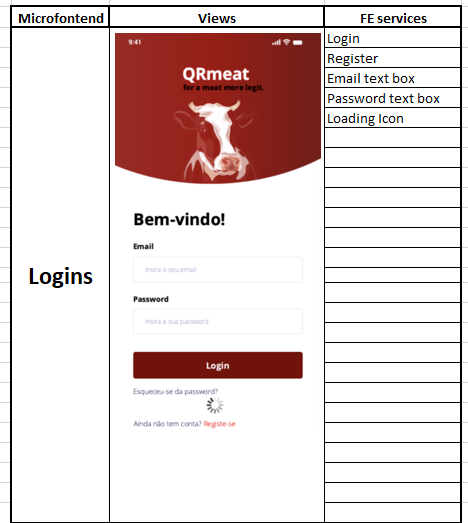
Login and Register pages are usually de-coupled from the rest of the logic, and as such can be completely independent, with minor tweaks such as changing the user model or authentication and authorization logic to match that of the respective application in which they would be used.

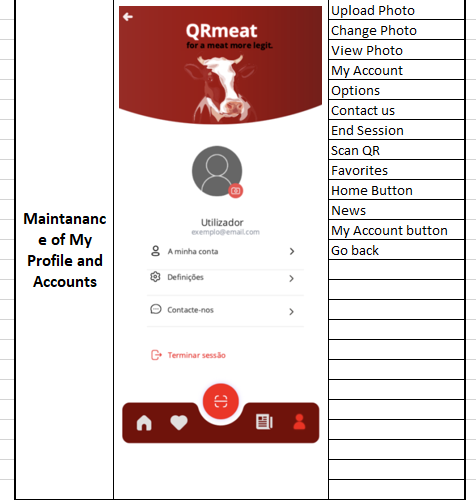
Assuming React.JS is being used for this application we can safely say that any page which’s only function is to list some models (such as meat products in this case) which can be handled by reusable components, can be considered independent and would therefore fit the definition of a micro-frontend. All the following pages are pages that fit this criterion:

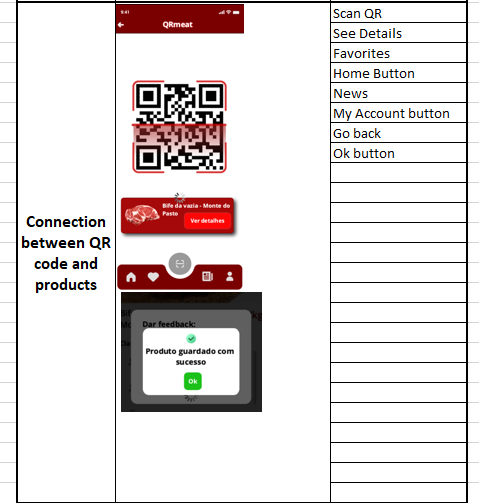
1. Product Page
2. User Details Page
3. News Page
4. Basket Page
5. Statistics Page

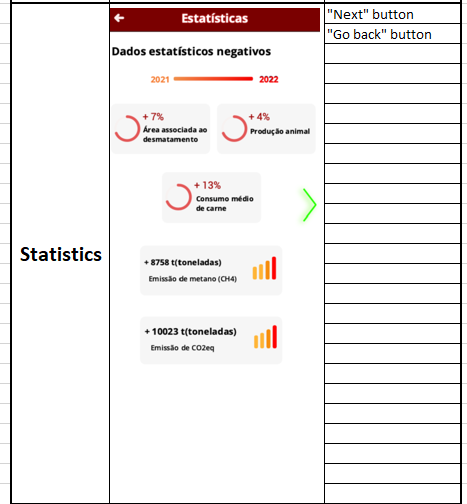
Additionally, any component relationships that they may have, which are not required in a specific scenario can be dropped without much issue.

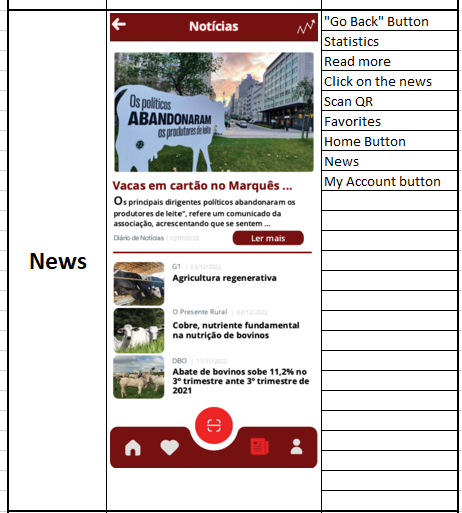
# Project Task 2

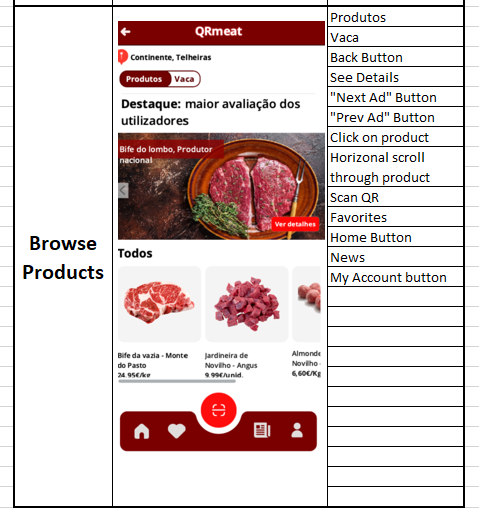












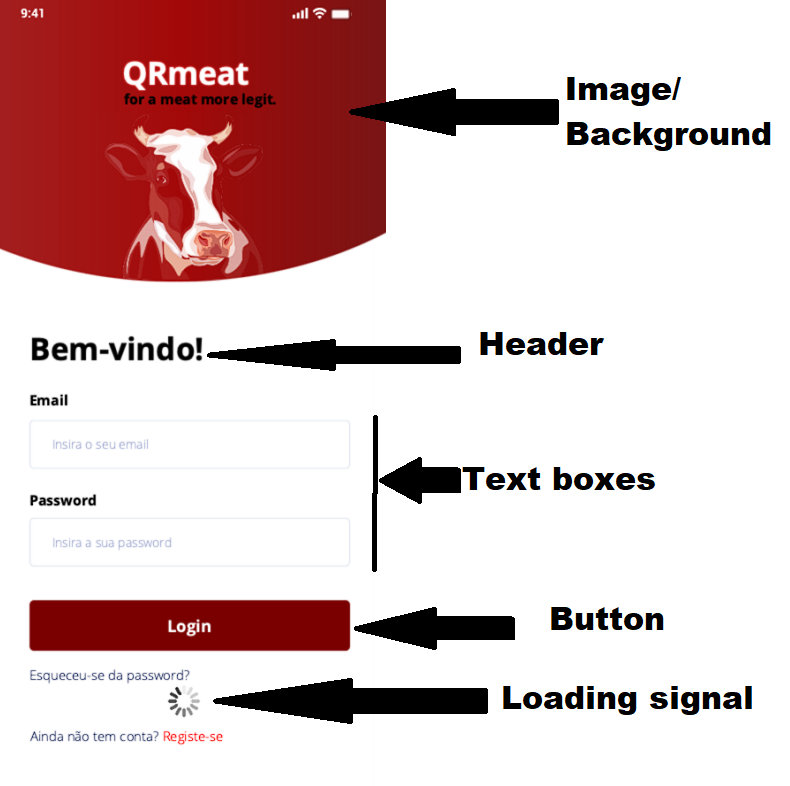
# Project Task 3

Assuming React will be used, we can surmise that components such as buttons or text fields, will be reusable, and as such will only be listed as one single component.

## Views:

### Login

#### Components:



This page is only shown if the user is not yet logged in, as such it can be said that the viewmodel *could be* a model like LoggedInUser depending on the logic of the application.

### Dashboard

While not a page by itself, the dashboard is definitely a micro-frontend, containing the routing logic of the application and multiple components.

A close-up of a red card

Description automatically generated

The navigation bar in this case is a component containing multiple button, which can be either hard coded or dynamic. It is one of the few micro-frontend services in this application which is only shown based on a state (logged in).

This micro-frontend could only have one ViewModel, that being a list of available pages, as well as their routes, though that is likely handled by the React.JS inbuilt routing.

### Home Page

#### Components:

A screenshot of a website

Description automatically generated

This page has no ViewModel

### All Products

#### Components:

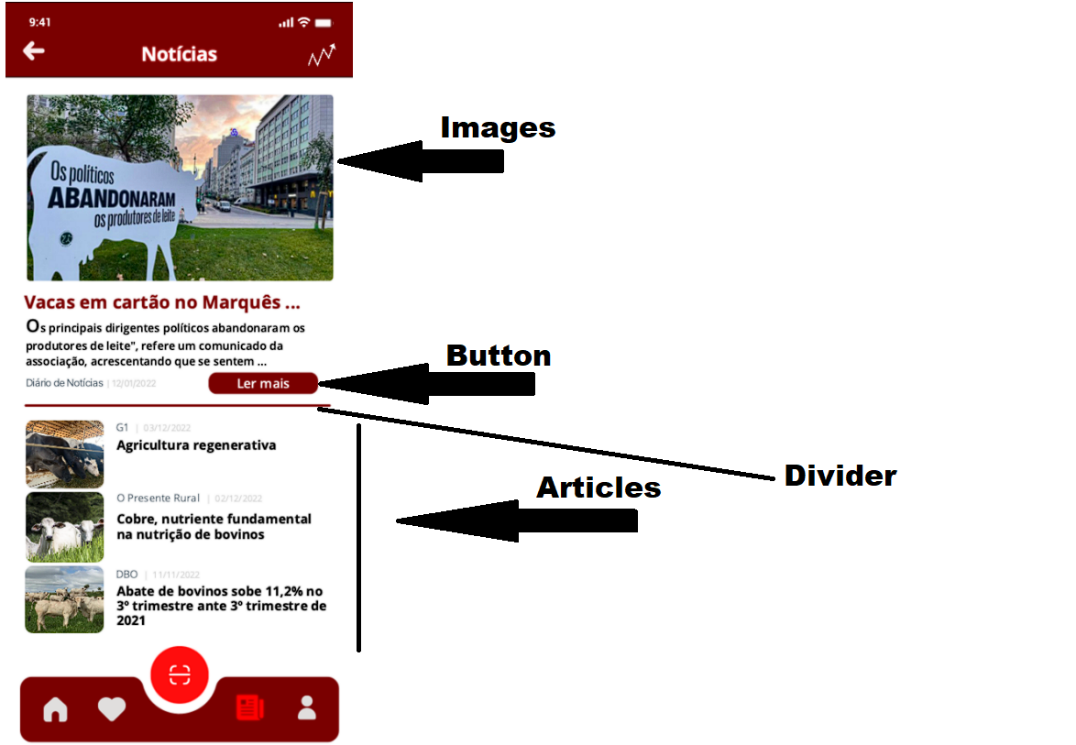


Some things to note here are that the Scroll Bar is likely part of the component containing all of the products, which is likely a table component containing images and text along with said images.

This page could have a ViewModel of a List of Products

### News

#### Components:

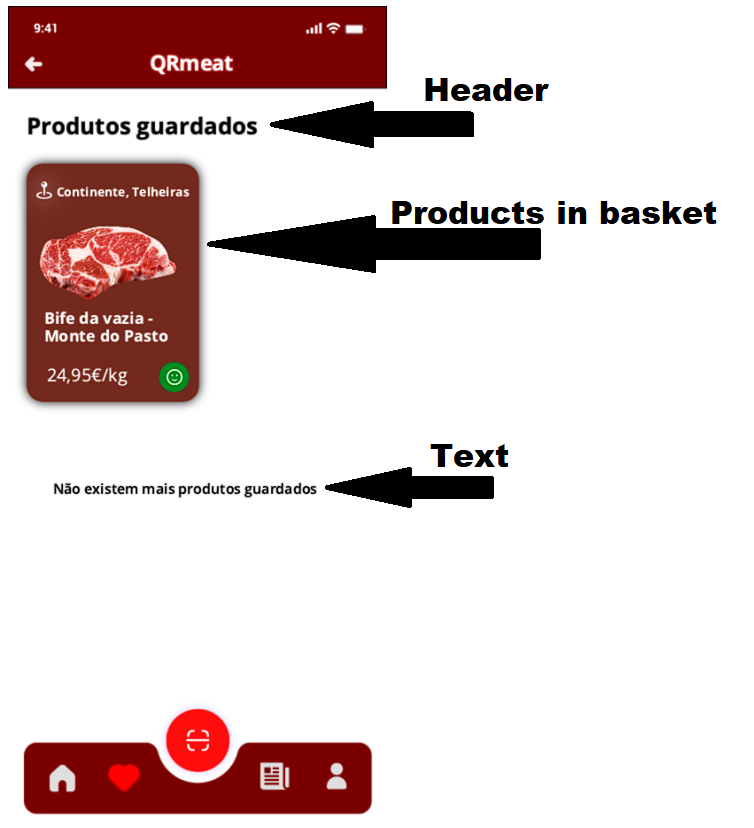


One thing to note here is that it is possible that the text and button are part of the same component.

This pages ViewModel is likely a list of the latest articles.

### Basket

#### Components:

  
This page’s ViewModel is likely a list of products, which are currently in a user’s basket. Perhaps something like UserID, ProductID or UserID, List<int> ProductIDs

### Statistics

#### Components:

A screenshot of a cell phone

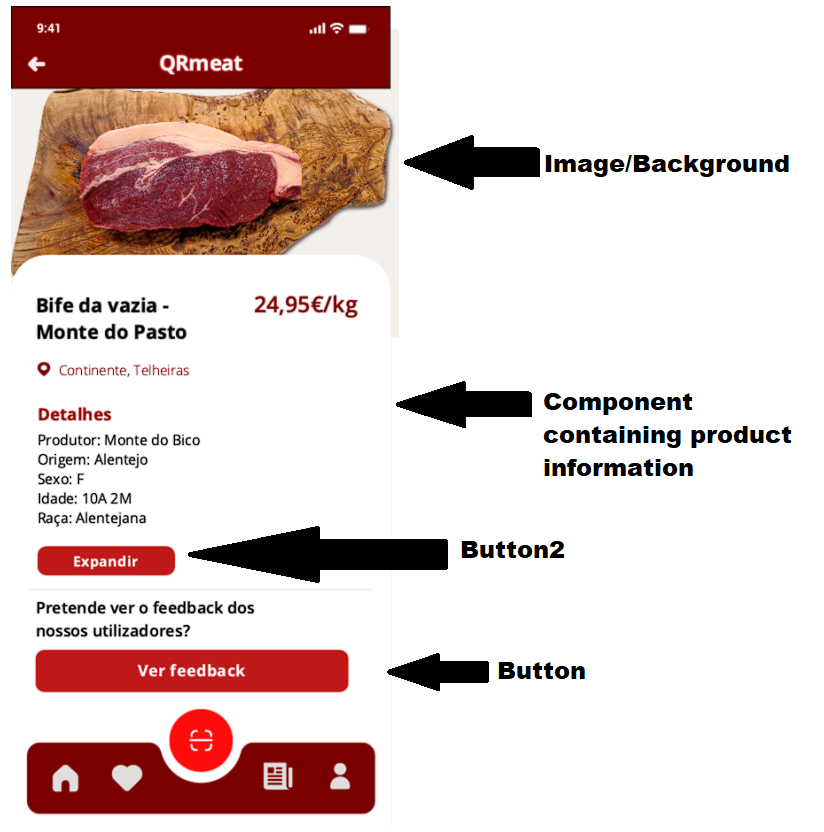
Description automatically generated

The statistics page seems to have multiple reusable subcomponents, used to measure different kinds of statistics. They are most likely components with some custom CSS which have some JS logic in them, used to determine the number shown, as well as the other things such as colour or how full the circle/ graph is.

Another interesting thing here is depending on the logic the green arrow marked as either a button or a toggle could be either be a simple button, used to change the page or a toggle, used to change the state, the main difference being that if it were a toggle, the state would likely persist and you would automatically be shown a page based on the toggle, instead of the hyperlink.

### Single Products

#### Components:



In this page there is another complex component, which contains other sub-components.

This page’s ViewModel is that of the Product shown on the page.

### Product QR Link

#### Components:



This page has a ViewModel which is likely generating a custom QR code, using some third party QR generator library along with data to link said QR code to a product, probably the ID of said product.

### User Details

#### Components:

A screenshot of a phone

Description automatically generated

This page is only shown if the User is already logged in, and it’s ViewModel is that of the User, which can run basic RUD(Read,Update,Delete) operations on the users (depending on the business logic of the application).

# Project Task 4

Add Item to basket Story Tell:

A diagram of a product purchase

Description automatically generated