UNIVERSIDADE FEDERAL DE CAMPINA GRANDE
CENTRO DE ENGENHARIA ELÉTRICA E INFORMÁTICA
UNIDADE ACADÊMICA DE SISTEMAS E COMPUTAÇÃO
CURSO DE BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

GABRIEL FERNANDES DA SILVA

CONSUMERHUB:
A PRODUCT'S REVIEW PLATFORM

CAMPINA GRANDE - PB

2020

GABRIEL FERNANDES DA SILVA

CONSUMERHUB:
A PRODUCT'S REVIEW PLATFORM

Trabalho de Conclusão Curso apresentado ao Curso Bacharelado em Ciência da Computação do Centro de Engenharia Elétrica e Informática da Universidade Federal de Campina Grande, como requisito parcial para obtenção do título de Bacharel em Ciência da Computação.

Orientadora: Professora Dra. Eliane Cristina de Araújo.

CAMPINA GRANDE - PB

2020

S586c   Silva, Gabriel Fernandes da.
Consumerhub: a product's review platform. / Gabriel Fernandes da Silva. – 2020.

13 f.

Orientadora: Profa. Dra. Eliane Cristina de Araújo.
Trabalho de Conclusão de Curso - Artigo (Curso de Bacharelado em Ciência da Computação) - Universidade Federal de Campina Grande; Centro de Engenharia Elétrica e Informática.

1. Consumerhub – web aplication. 2. Product's review platform. 3. E-commerce. 4. Plataforma colaborativa. 5. Avaliação de produtos. I. Araújo, Eliane Cristina de. II. Título.

CDU:004(045)

Elaboração da Ficha Catalográfica:

Johnny Rodrigues Barbosa
Bibliotecário-Documentalista
CRB-15/626

GABRIEL FERNANDES DA SILVA

CONSUMERHUB: A PRODUCT’S REVIEW PLATFORM

Trabalho de Conclusão Curso apresentado ao Curso Bacharelado em Ciência da Computação do Centro de Engenharia Elétrica e Informática da Universidade Federal de Campina Grande, como requisito parcial para obtenção do título de Bacharel em Ciência da Computação.

Trabalho aprovado em: 2020.

CAMPINA GRANDE - PB

Consumerhub: A products' review platform

Gabriel Fernandes da Silva
Federal University of Campina Grande
gabriel.fernandes.silva@ccc.ufcg.edu.br

Eliane Araújo (mentor)
Federal University of Campina Grande
eliane@computacao.ufcg.edu.br

ABSTRACT

When looking for a new product to buy, consumers are many times faced with different options available, but how to choose the best? Review sections on e-commerce websites can help by providing product reviews and ratings from consumers, and new consumers can use them to base their buying decisions. In that approach, reviews are made only by the website's clients and only referring to the products they sell. This work proposes a solution that widens that approach. It is a community-driven platform where users collaborate to build a database of products' reviews and ratings to help consumers make a better buying decision.[1]

Repository

https://github.com/gabrielfern/consumerhub

1. INTRODUCTION

Today, the competition between products is high. Take, for example, this comparison [1] of the cost-benefit smartphone by a specialized comparison website. It is a comparison of 6 models of smartphones, and it could have been a much bigger list. As with smartphones, most of the products have options from competitors, and the consumer sometimes might get lost on what product to buy.

Many e-commerce websites provide a section for product reviews to help consumers. This product page [2] from Amazon is an example. A buyer might consider those reviews when deciding which product to buy between similar options available. The product with best reviews has a good chance of being one of the best.

For convenience goods that are sold typically only in supermarkets or grocery stores, consumers do not have a place like review sections of e-commerce websites to know what is the public opinion on a certain product. Because of this they usually ask people they know about the quality of the products they want to buy.

People are also relying on different kinds of platforms to get information about products, like the Youtube video platform [3]. These platforms are also useful to discover new products. One kind of videos found on Youtube that helps consumers is videos of reviews, where the youtuber tests and gives their opinion on a certain product. Another type of video about products is *unboxings*, where the product is shown being removed from its original package. Both of these kinds of videos can greatly help consumers, as the consumer can see the opinion of another person in the product they are interested in and see how the product looks when it is first opened.

The problem of gathering information of a product from other consumers is what we are trying to solve. And for that we propose an innovative platform where users can find opinions of other consumers about products they are considering to buy. This platform will leverage the creation of a community around product reviews.

This work introduces a platform named Consumerhub. It is a web application, open for all users to collaborate with new products and reviews. It has review sections on products, similar to those of e-commerce websites, but differently from those websites, this platform is open for any user, and products that do not exist in the platform can be added by any user. This way, products that are only found in grocery stores can also be added and receive reviews.

2. SOLUTION

2.1 Overview

Consumerhub is a web platform that is a hub for consumers. By hub we mean a place where consumers go, in this case navigate to, when they want to know about a product they want to buy. This platform is not a place they go to buy products, but rather a place they go to acquire information about products.

It is a collaborative platform, open, similar to what Wikipedia is. The catalog of products can be increased by any user, as users can create new pages on Wikipedia. Users can make

[1] The authors retain the rights, under a Creative Commons Attribution CC BY license, to all content in this article (including any elements they may contain, such as pictures, drawings, tables), as well as all materials produced by authors that are related to the reported work and are referenced in the article (such as source code and databases). This license allows others to distribute, adapt and evolve their work, even commercially, as long as the authors are credited for the original creation.

additions, or edit, products, as users in Wikipedia can. In a similar way that Wikipedia keeps its content from being vandalized, Consumerhub has a hierarchy of users: users, moderators and administrators. Each one of them with more powers regarding the platform.

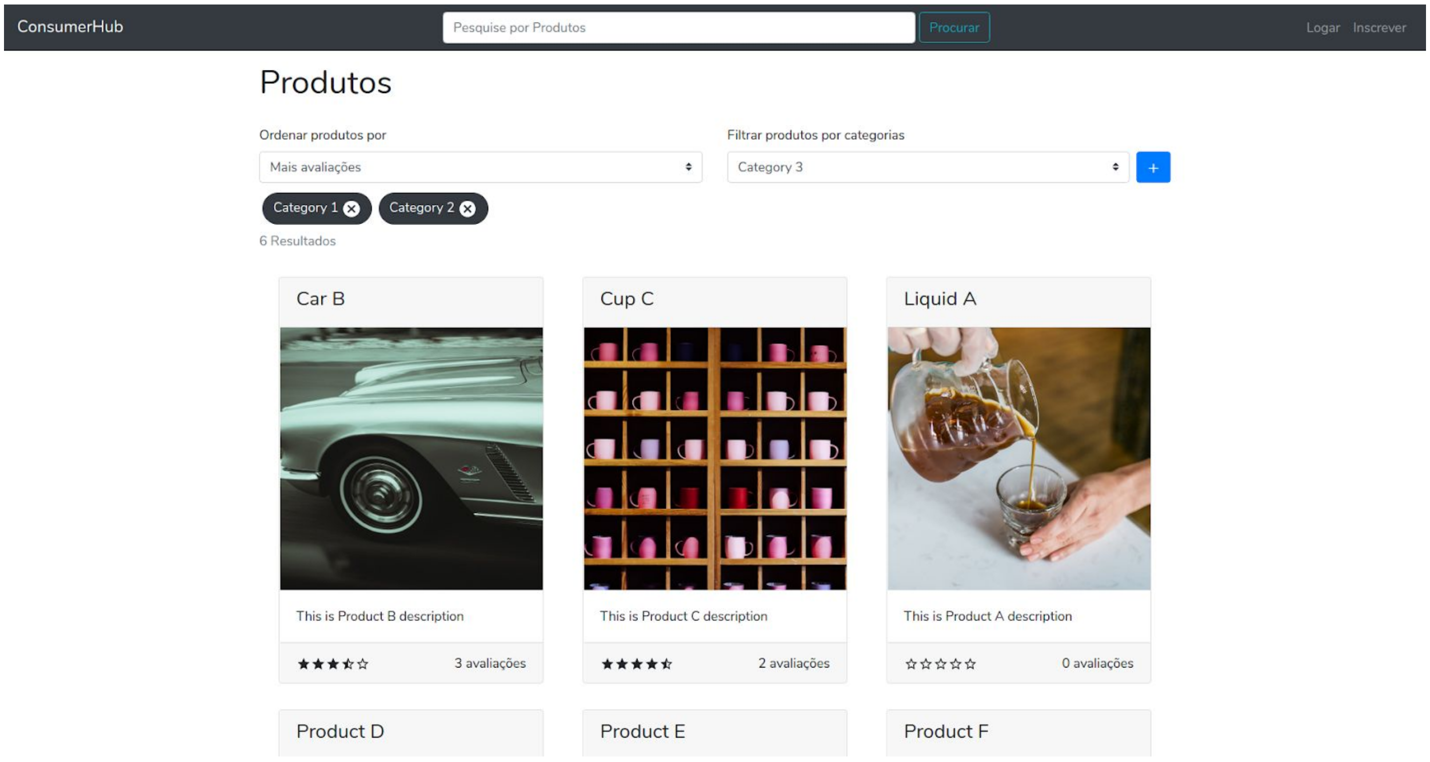
Each product can be reviewed by users. A review is an optional commentary, and a rating, the rating goes from 1 to 5. Those reviews are the source of knowledge that visitors can use to get more information on products. Aside from reviews, products have more information, they are:

- Title
- Description
- Images (up to 5)
- Links (up to 3)

Images are a good way to visualize a product, especially if the user has not seen it already. Links are a way of providing further information about a product to the users. A link can be a url to the product page on the manufacturer's website, e-commerce websites selling the product, or any other website that talks about the product.

Any visitor can navigate and search for all the products, and on each product page see all the reviews the product has. If the visitor wants to leave a review on a particular product, submit a new product or edit a current product, the visitor needs to create an account on the platform. That can be done with a name, email and password (a profile picture is optional) or if the person prefers, with a click of a button using a Google account.

Products are grouped into categories. Each product can belong to more than one category. A category is a type or class of products, for example, the category Smartphones would consist of several products that are smartphones. A smartphone could belong to the category of Smartphones and also belong to the category Technology, and so forth. These categories can be used to filter products in the product list, so only products of a certain category would show up in the search. *Image 1* shows the product list page and how categories are used to filter products.



Produtos

Ordenar produtos por: Mais avaliações

Filtrar produtos por categorias: Category 3 +

Category 1 ⊗ Category 2 ⊗

6 Resultados

Car B
This is Product B description
★★★⯪☆ 3 avaliações

Cup C
This is Product C description
★★★★⯪ 2 avaliações

Liquid A
This is Product A description
☆☆☆☆☆ 0 avaliações

Product D

Product E

Product F

Image 1: Shows the product list page, demonstrating how product categories can be used to filter products. From the perspective of a visitor, without being logged in. Pictures used here are Public Domain images, source [4]. Products used in this image are not real products, to avoid problems with intellectual property.

Users can interact with other users' reviews. A user can upvote or downvote a review. Reviews can later be sorted by reviews with the most upvotes, by date and so forth, as shown in *Image 2*.

Another way users can interact with other reviews is by reporting them. A report can be used to notify moderators of content that should not be on the platform (e.g. inappropriate images). Each report has a message left by the user that made the report (i.e. the reason). Users can also report products and other users. Those reports can be listed by moderators, and if they think the report is reasonable they can take an action, like removing the review for example.

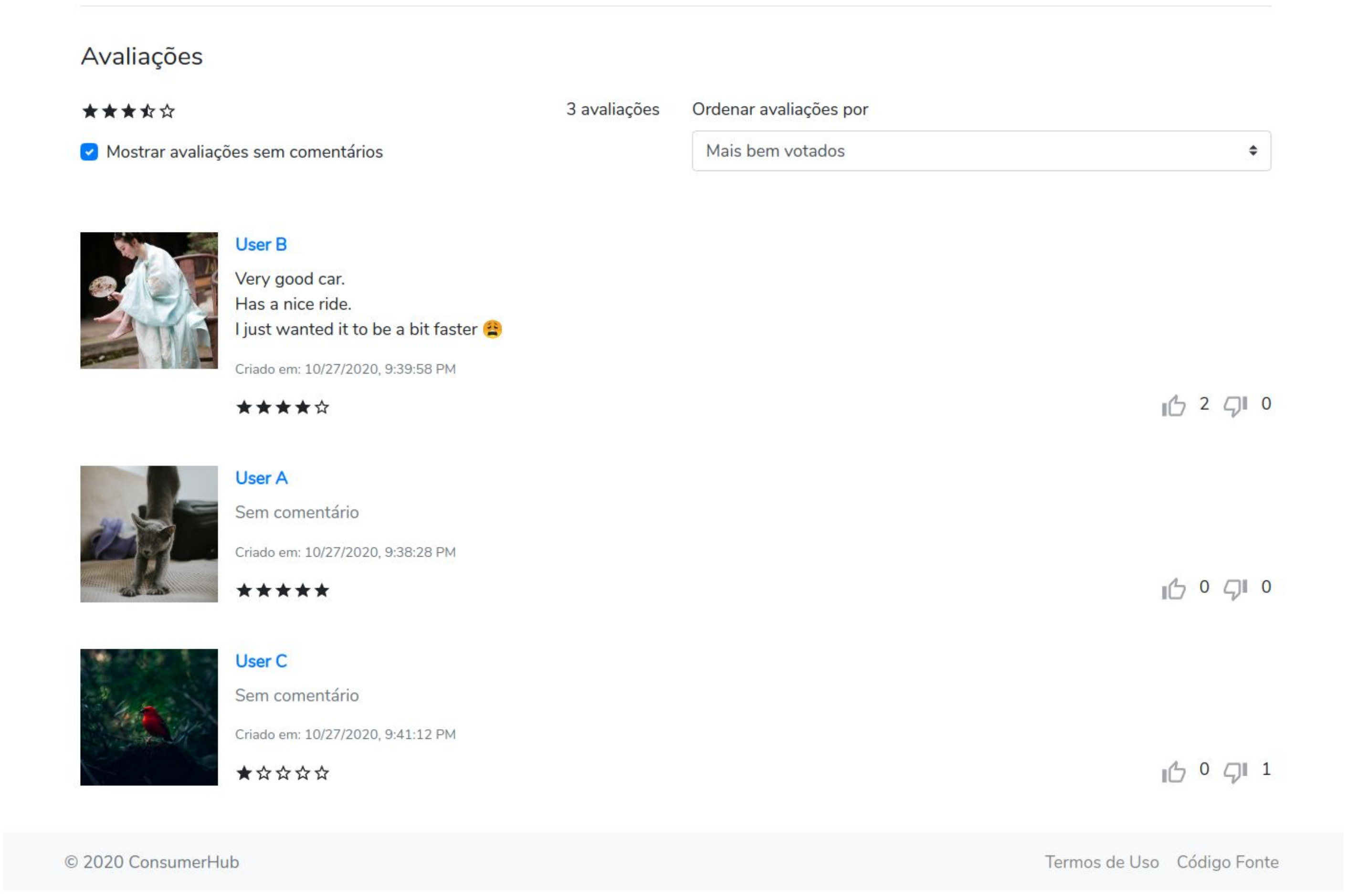
Users have a profile page. On a user's profile page, their reviews are listed. Also on a user's profile page, other users can send friendship requests or report the user. Only users that are friends, on the platform, can see each other's email. Users can use email for further communication, for example, to share more information about a review.

When new users create their account, they are assigned the user type "user". Users of this type can create new products and edit current ones, but their contribution needs to be accepted by one moderator to become part of the product catalog. Moderators can do everything a user can do, with the following additions:

- Accept product submissions from other users, including themselves.
- See all the reports made by users.
- Remove products, reviews or users from the platform. (A moderator can only remove users of the type "user", i.e. they cannot remove another moderator.)
- See the email of any user. (Normal users can only see the email of friends.)
- Edit product categories.

Administrators can (in addition to what moderators can do):

- Change the user type of any user.
- See a list of all the users currently registered on the platform.
- Remove any user. (Including other administrators.)
- Send email to individuals or all users at the same time, using the email configured in the server.


Avaliações

★★★⯪☆ 3 avaliações

Ordenar avaliações por: Mais bem votados

☑ Mostrar avaliações sem comentários

User B
Very good car.
Has a nice ride.
I just wanted it to be a bit faster 😩
Criado em: 10/27/2020, 9:39:58 PM
★★★★☆ 👍 2 👎 0

User A
Sem comentário
Criado em: 10/27/2020, 9:38:28 PM
★★★★★ 👍 0 👎 0

User C
Sem comentário
Criado em: 10/27/2020, 9:41:12 PM
★☆☆☆☆ 👍 0 👎 1

© 2020 ConsumerHub

Termos de Uso Código Fonte

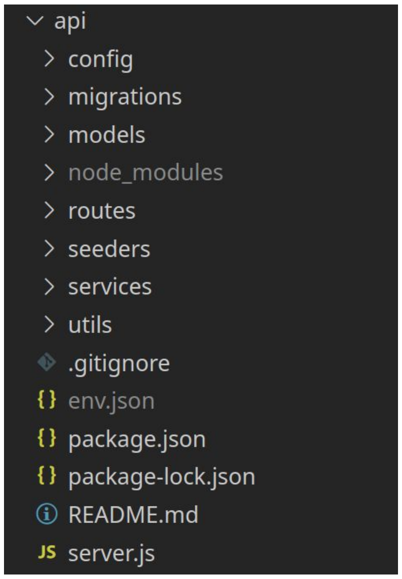
Image 2: Produtct's review section

2.2 Architecture

There are 3 main components in our system: client, server and database. The client module is our frontend code, or application, that runs on the user's browser. A different client can also be used to access our http api. The client communicates with the server, that is an application that runs on a single machine, and is responsible for responding to requests made by clients. The server is responsible for the business logic of the system, and connects to a database to store data. The database can be a process running on the same machine of the server, or it can be a service running elsewhere. A relational database was chosen for this project.

The codebase, which is found on Github (link in the repository section), is divided into two main directories, api and app. The api directory is our server, or backend. The app directory is our client, or frontend. In the next 2 subsections we will discuss the structure of these directories.

2.2.1 Backend structure



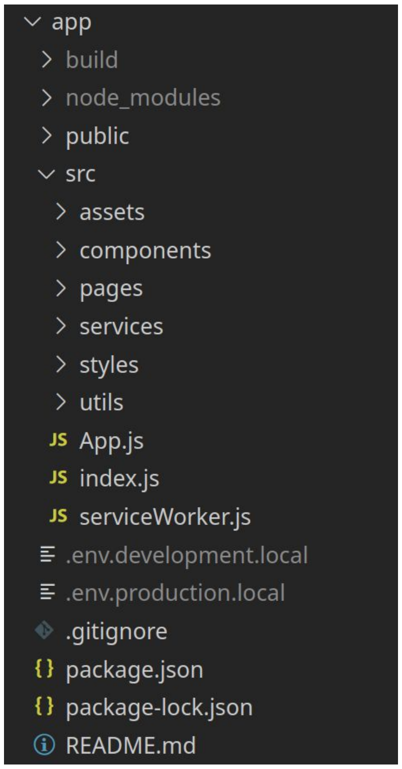
```
api
  config
  migrations
  models
  node_modules
  routes
  seeders
  services
  utils
  .gitignore
  env.json
  package.json
  package-lock.json
  README.md
  server.js
```

Image 3: Backend file structure

Image 3 shows the file structure of the backend. Each file goes in a subdirectory depending on its purpose. The database configuration goes into the config folder. The migrations folder has the source files to create all necessary tables in the database. The models are the mapping of database entities to the language objects. Routes, which can be seen as controllers, are responsible for handling http requests. Seeders are used to populate the database, in our case we create the first administrator in a seed file. Services provide additional functionality, for example notifications. Utils are for auxiliary functions. The *server.js* file is our entry point code.

2.2.2 Frontend structure

Image 4 shows the frontend file structure. At the top level, there are two main directories, the *public* directory that contains static files and the *src* directory that contains the source code. Inside *src*, the assets folder serves the same purpose of the public folder. It contains mainly images. Components and Pages have React components, with the difference between them being that Page components represent a webpage, having their own url. Services directory contains the api service that abstracts http requests out of the components. Styles are for CSS files, and utils have helper functions.



```
app
  build
  node_modules
  public
  src
    assets
    components
    pages
    services
    styles
    utils
    App.js
    index.js
    serviceWorker.js
  .env.development.local
  .env.production.local
  .gitignore
  package.json
  package-lock.json
  README.md
```

Image 4: Frontend file structure

2.2.3 Database entities

Consumerhub data model is composed of several different entities, that are translated to tables in the database along with their constraints. In the application server, the database tables are abstracted as model objects. *Image 5* shows the Entity–relationship diagram of the system:

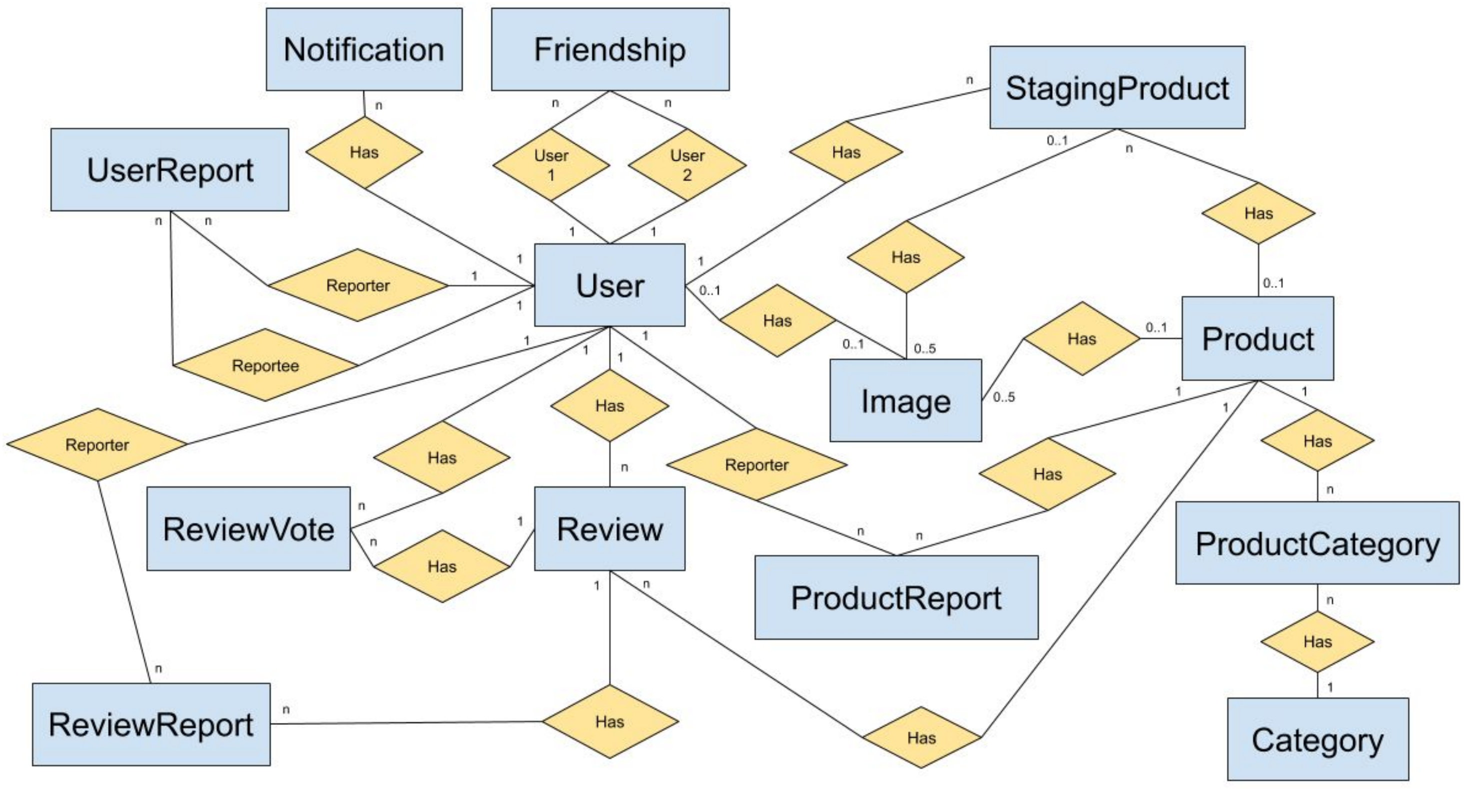


Image 5: Entity-relationship diagram of Consumerhub.

As a study case, the entity *ReviewVote*, that represents the vote a user gave to a review has associations with the entity *User* and the entity *Review*. One instance of *ReviewVote* has one *User*, the user who voted, and the instance also has one *Review*, the review that the vote was given to. One *User* can have zero or more *ReviewVotes*, and one *Review* can also have zero or more *ReviewVotes*.

2.3 Technologies

It was decided to use the Javascript programming language for both backend and frontend. The reason behind this decision is that it makes the development easier, as there is little overhead in switching from backend to frontend development.

We chose Nodejs [5] to run on the server-side. It is a Javascript runtime built on Chrome's V8 JavaScript engine. For the frontend, Javascript already runs natively on today's browsers.

In the next subtopics, we will explain details of technologies used in the backend and in the frontend.

2.3.1 Backend

As we were building a web server, the first decision made was what web framework to use. The most used web framework for Nodejs is Express [6]. It has a great amount of online resources and integrations, allied with previous experience of the author with it; it was a natural choice. Express makes development of http apis easier, providing a layer of fundamental web application features on top of Nodejs.

We used a Nodejs Object–relational mapping [8] (ORM) called Sequelize [9] to integrate with Postgresql [7] database. An ORM is an abstraction on top of the database, that translates database relations (or tables) to objects in the used language. Sequelize, among many different SQL [10] based databases, supports Postgresql, making the development of a Nodejs application that uses Postgresql very straightforward.

For the generation of authentication tokens, the library jsonwebtoken [11] was used.

Still in the topic of security, bcrypt [12] was used to hash user passwords. For the feature of emailing, nodemailer [13]. And to support login using Google accounts the google-auth-library [14].

2.3.2 Frontend

Javascript library React [15] was chosen to build the user interface. It is a very well known frontend library, and as it was the case with Express, the author had prior knowledge and experience with it. React makes it easier to create reusable code and interactive UIs. All of which were requirements for the frontend code, making it a perfect fit.

When it comes to styling, we used the Bootstrap [16] library together with React Bootstrap [17]. The Bootstrap library exports CSS, and for some of their interactive components (e.g. modals), it requires the inclusion of their Javascript library; React Bootstrap then replaces the Javascript part of Bootstrap, making React Components available for a seamless integration with React.

Our web application is a single page application [18] (SPA); SPAs increase responsiveness by avoiding page loadings and that translates into a better user experience. SPAs usually use client-side routing to give the impression of different pages to the user, and for that React Router [19] was used. React Router is a library for client-side routing, and as with React Bootstrap, it works very well with React.

3. USER EVALUATION

Consumerhub was deployed in Heroku [20], which is a *platform as a service* (PAAS). In Heroku, each deployed application gets a public URL. The URL for Consumerhub, https://consumerhub.herokuapp.com, was shared with a group of people for them to use and test the platform. After one week, an evaluation questionnaire was sent by email to the users registered on the platform at that moment.

The questionnaire had 12 questions. The first question had the objective of assessing the usefulness of a platform like Consumerhub. The next 10 questions were the System Usability Scale [21] questionnaire. The last question was the only optional. It asked the users for suggestions for the improvement of Consumerhub.

The questionnaire did not ask for the respondent's email. This was done to make the responses anonymous, as a way to avoid biases of the respondents when knowing they would be identifiable. The questionnaire was made through a Google Form, and it was configured to only allow respondents that were logged in; this was a measure against the same person responding to the form more than once.

The users invited to use the platform were of a wide range of ages, with all of them being adults. They had different backgrounds, some of them having great experience and technical knowledge in computing, while others having no technical knowledge, but all of them being consumers of products to some extent.

The questionnaire was made in Portuguese, the native language of the users. For presenting the results in this work, the Portuguese content was translated to English.

3.1 Results

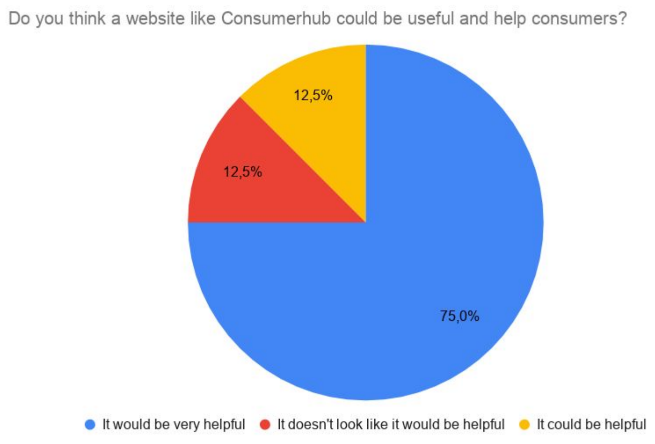
A total of 8 users participated in the questionnaire. The responses, and the original questions in Portgeuse, can be found at https://gist.github.com/gabrielfern/63b10dfe4981e1c681c31af841970bec.

The first question asked about the usefulness of a platform like Consumerhub. The objective of this question was to answer the main point of this work, that is if an approach of an open and collaborative platform for user reviews of products would help consumers.

This question gave the users 4 options, and they had to choose one. The options were:

1. It would be very helpful
2. It could be helpful
3. It doesn't look like it would be helpful
4. I don't think it would be helpful

The results can be seen in the following pie chart:



6 out of the 8 responses said that a website like Consumerhub would be very helpful. The next 2 options each one got 1 response. The last, and most negative, option did not get any answer.

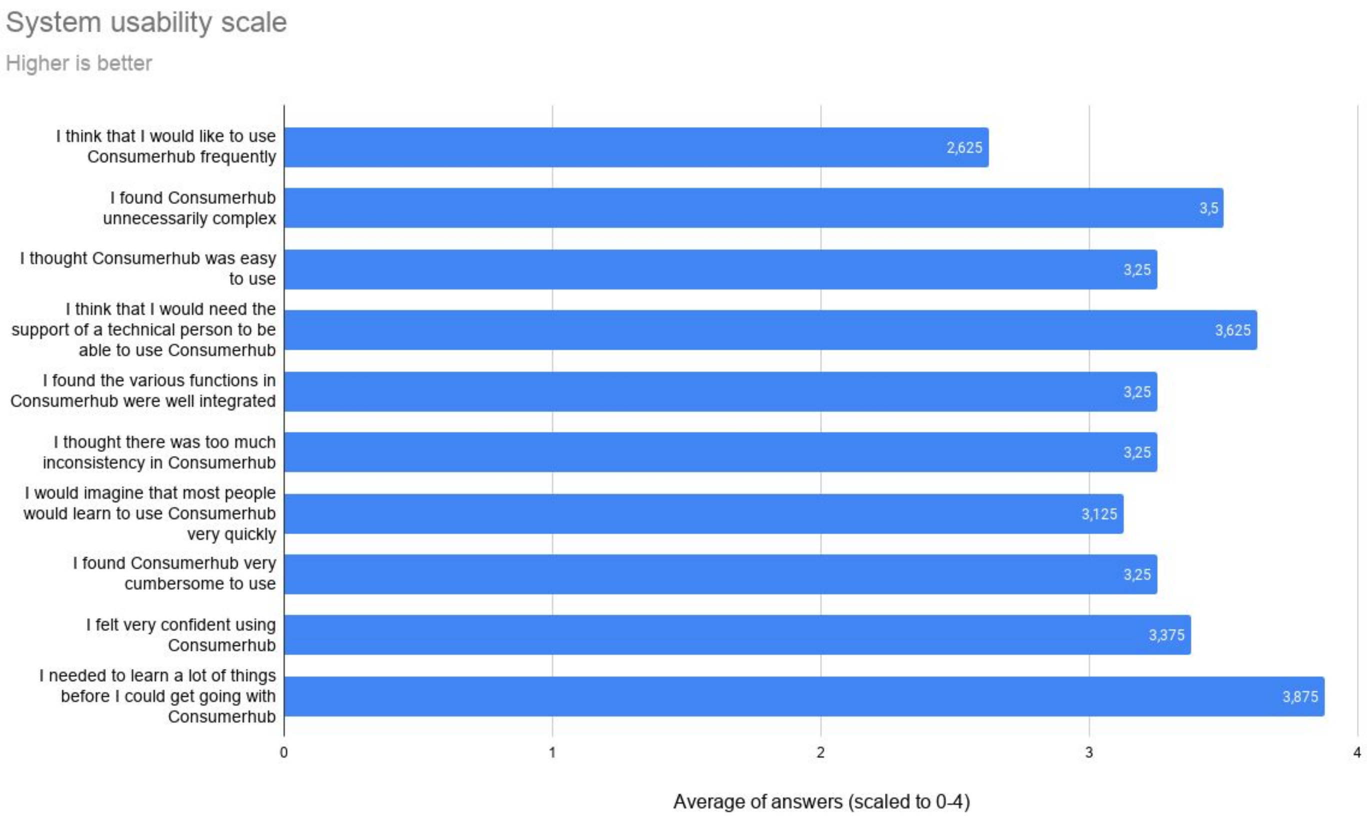
3.1.1 System usability scale

The System Usability Scale (SUS) is a reliable, low-cost usability scale that can be used for global assessments of systems usability [22]. It yields a single value representing the overall usability of the system.

SUS is a Likert scale [23] with 10 questions. Respondents choose a value from 1 to 5 for each question. The questions are sentences, and 1 is chosen if the person strongly disagrees with the sentence, and 5 if the person strongly agrees. There are 5 positive sentences and 5 negative sentences.

To calculate the SUS score, each response value from 1 to 5 is converted into a value that goes from 0 to 4, which is the question's contribution to the score. In this new scale, a value of 4 in a negative question means the user disagreed the most, and a value of 4 in a positive question means the user agreed the most.

The following chart shows the 10 SUS questions used in the questionnaire, along with the averages of all responses for each question. These are the averages of the converted values.



The sum of all of these averages is a number between 0 and 40. For our results the sum is 33.125. To get the SUS score the sum is multiplied by 2.5, which yields a value between 0 and 100, the closer the value is to 100 the better usability the system has. The average SUS score for Consumerhub then is 82.812, with a 95% confidence interval ranging from 70.33 to 95.28. This score is above the average found by Bangor et al. [24], which is 70.1 in 2324 SUS questionnaires, and by Sauro and Lewis [25], which is 62.1 in 324 SUS questionnaires.

3.1.2 Improvement suggestions
The last question was an input field for the users to write suggestions of improvements for the platform. This was an optional field. Out of the 8 users that answered the questionnaire, 6 left any text in this field.

Most suggestions were related to UI/UX, 3 out of 6. UI meaning user interface and UX meaning user experience. One of the suggestions was "Make the UX better". The other 2 suggested to make some UI elements more noticeable or more intuitive.

4. EXPERIENCE AND LESSONS LEARNED
4.1 Development Process
At first, a document of requirements was made, containing the core functionality expected and desirable ones. This requirements analysis together with raw diagrams, including an initial entity–relationship and a homepage wireframe, were the resources used in the system design phase.

Finished the system design phase, the development began. The approach used for the development, or rather the development process, resembles the incremental build model [26], where the product is designed, implemented and tested incrementally; and the product is defined as finished when it satisfies all of its requirements.

Every new module of the system had its backend part completed first, then its frontend counterpart. Modules here can be seen as a distinctive feature, for example user friends. The final stage of development was to revamp the webapp appearance, as the UI at that moment was mainly concerned with tests and the frontend logic, instead of a good looking and final user experience and user interface.

Tasks in the development phase were created as Github issues[2] in the repository of the source code. A Github Project was created to keep track of the issues, with the name of Consumerhub MVP[3], also available in the repository. Issues could be in 3 different states: To do, In progress and Done; as the tasks were being completed the respective issue was moved to Done. Issues could be a new functionality, a problem/bug, or some other improvement.

4.2 Main challenges and their solutions

4.2.1 User's product submissions

One of the requirements of the platform was to permit all users to submit new products. Product submissions would not go live for all the platform's visitors to see, but would go to a list of submissions that moderators would have to accept for it to go live. More than just new products, users would also be able to edit current products, and product edits would have to be accepted by a moderator the same way a new product is.

There would have to be products in two different states, accepted and waiting acceptance ones. The solution for this problem was to have a new database table, or a new model, to represent products awaiting acceptance. The name given to this model was "StagingProduct". For the accepted products, the model name was "Product".

This ideia was roughly based on git. When changes are made to a git repository, one can git add them, moving the changes to an area called staging area. Changes in the staging area can then be committed. With products, submissions of new or edited products are like changes to a git repository, they can be seen as being in the staging area, when accepted, in our analogy committed, they become products.

Products are always created from staging products, when that happens, the staging product is deleted, and a product is created with the staging product values. When a user clicks to edit a product, that product is cloned into a new staging product, which can be edited, and when a moderator later accepts the changes, the original product is updated to the values of the staging product, and then the staging product gets deleted.

4.2.2 Images

Images were challenging in two ways for this project; first was how to store them; secondly was how to deal with browser caching.

It is usually recommended to not store images in relational databases. As images take up more space, and creates overhead in the database or network. One other way of storing images is file storage services. To avoid the complexity of managing another service for images, Postgresql was also used to store the images, as blob data. Through tests in the infrastructure used to deploy the application, the latency to access the images was not a problem, and Postgresql supports up to 1 GB in a blob data type field, much more than the maximum of 5 MB per image that this application was configured to use.

In modern web browsers, images are cached after the first time they are downloaded. The first solution to store images in the database stored images as fields in the models they belonged to, for example, a User had a field "image" that was of the type blob and was the actual image data. With the first solution, the URL to get a user's image was always the same, even after the user changed its image, the same URL was seen by the browser as having the same data, so the browser did not show the updated user image because of the image URL always being the same (e.g. /api/user/image).

The problem of browsers caching the images also affected product images. A solution for this problem was later implemented. The solution was to have all images in a different table, called Images, where they have random ids, and for each update of an image a new instance of Image is added to Images, and this new instance gets a new id. The Image id then is used in the path to get the image, for example "/api/images/otisWtcMbpta". Having different URLs when images are updated makes the browser always download the image, without using the cache, everytime the URL of the image changes. For images that did not change, they keep the same URL, and browser caching works as expected.

4.2.3 Email

One of the desirable features was for Consumerhub to be able to send emails to users. The First use case for this functionality was to let users reset their passwords, through email, and to require users to confirm their email addresses when they create their account. The second use case was to have a way for administrators to send email to all users, or to one user, using the official email address of Consumerhub.

For an application to be able to send emails, the application needs to create an email server or to use an email

[2] Issues for Consumerhub
[3] Consumerhub MVP tasks

service. The second option was used, for its simplicity. The solution implemented uses a Gmail account and a library that connects to Gmail and sends email automatically. This solution has many problems though, as a Gmail account is not a proper email service, for sending emails via an api. Gmail provides an option for enabling access to other applications, but this access is not reliable, as Gmail sometimes blocks access to the account based on their heuristics. These problems were described in the work [27]. Because sending emails this way was not reliable enough, the only feature implemented was the support for administrators to send email to users.

4.3 Limitations

One of the platform's limitations is that actions that users take are not recorded, aside from logs from proxy servers that can keep a history of all the requests made to the server. Some of the actions that could be of great importance to record are:

- Moderator's removal of a user
- Moderator's removal of a product
- Moderator's removal of a review
- History of product modifications, linking to the user that made the modification.
- Moderator that accepted a product

Without knowing the moderator that removed reviews, it can be very difficult for administrators to demote or remove a bad intentioned moderator. Another difficulty would be to know which user made a certain modification to a product.

A second limitation is that if a user's submission of a product is rejected, all of the work the user put on that product is lost; if the user sees the rejection message from the moderator that rejected the product, and the user wanted to fix what the moderator said was wrong, the user has to make all of the changes again, or create a new product if the submission was for a new product.

Another shortcoming of the product system is that multiple users editing the same product can lead to a user's work being overwritten. Suppose two different users started editing the same product, they don't see each other's work, as submissions are private to the user. When a user starts editing a product, a clone of the product is created at that time. The two users would make their modifications, the first could change the product's title and the second could have added a new link to the product. When the first user's submission is accepted, the product title of the original product is changed, but if the submission of the second user is accepted, that submission has the product in the state it was before the first user's addition, leading to the link being added, and the title of the product going back to what it was before.

The user evaluation made in this work had the participation of only 8 users, which is a fairly small number for drawing conclusions. Although more than 10 users had signed up on the platform, the process of reaching out to the users for them to respond to the questionnaire might have been inefficient. For more conclusive results of the effectiveness of this system, a bigger evaluation can be made in the future.

4.4 Future work

The application produced in this work was a minimum valuable product (MVP). The core features thought to be needed were all completed. But there is a lot of room for improvement and evolution in this application. Some of the possible improvements are:

- User activity history
- Limiting moderator's power
- Bidirectional communication between moderator and user that submitted a new product or product changes
- Let users edit products together (i.e. shared editing).
- Email confirmation and resetting of password by email

5. REFERENCES

[1] Qual é o melhor celular custo benefício? [Guia 2019]. https://escolhasegura.com.br/qual-melhor-celular-custo-beneficio/

[2] Lessons Learned in Software Testing: A Context-Driven Approach. https://www.amazon.com/Lessons-Learned-Software-Testing-Context-Driven/dp/0471081124

[3] More people than ever are turning to YouTube for product reviews. https://medium.com/sponsokit/more-people-than-ever-are-turning-to-youtube-for-product-reviews-4956d3647e34

[4] Public Domain Pictures. https://unsplash.com/images/stock/public-domain

[5] Node.js. https://nodejs.org/

[6] Express - Node.js web application framework. https://expressjs.com/

[7] PostgreSQL: The world's most advanced open source database. https://www.postgresql.org/

[8] Object–relational mapping. https://en.wikipedia.org/wiki/Object%E2%80%93relational_mapping

[9] Sequelize ORM. https://sequelize.org/

[10] SQL. https://en.wikipedia.org/wiki/SQL

[11] JsonWebToken. https://github.com/auth0/node-jsonwebtoken

[12] Node bcrypt. https://github.com/kelektiv/node.bcrypt.js

[13] Nodemailer. https://nodemailer.com/

[14] Google Auth Library for Node.js. https://github.com/googleapis/google-auth-library-nodejs

[15] React – A JavaScript library for building user interfaces. https://reactjs.org/

[16] Bootstrap · The most popular HTML, CSS, and JS library in the world. https://getbootstrap.com/

[17] React Bootstrap. https://react-bootstrap.github.io/

[18] Single-page application. https://en.wikipedia.org/wiki/Single-page_application

[19] React Router: Declarative Routing for React.js. https://reactrouter.com/

[20] Heroku: Cloud Application Platform. https://www.heroku.com/

[21] System usability scale. https://en.wikipedia.org/wiki/System_usability_scale

[22] Brooke, J.: SUS: A “quick and dirty” usability scale. SUS - A quick and dirty usability scale

[23] Likert scale. https://en.wikipedia.org/wiki/Likert_scale

[24] Bangor, A., Kortum, P. T., Miller, J. T.: An Empirical Evaluation of the System Usability Scale. International Journal of Human-Computer Interaction. 24, 574--594 (2008)

[25] Lewis, James R., Sauro, Jeff: The Factor Structure of the System Usability Scale https://measuringu.com/wp-content/uploads/2017/07/Lewis_Sauro_HCII2009.pdf

[26] Incremental build model. https://en.wikipedia.org/wiki/Incremental_build_model

[27] Oliveira da Silva, Eri Jonhson: Um sistema para automatizar os processos gerenciais da Olimpíada Paraibana de Informática. https://drive.google.com/file/d/1oSiBBxZkFgPVKgRcqZ8rqaHC6O_Nu9jm/view