

**PONTIFICAL CATHOLIC UNIVERSITY OF MINAS GERAIS**  
**DISTANCE EDUCATION CENTER**  
**Postgraduate *Summer of Sense* in Full Stack Web Development**

**Samuel Gallo Da Silva Jorge**

**ADMINISTRATIVE PANEL FOR E-COMMERCE**

Rio de Janeiro

2020

**Samuel Gallo Da Silva Jorge**

**ADMINISTRATIVE PANEL FOR E-COMMERCE**

Completion of course work presented to the Specialization Course in Full Stack Web Development as a partial requirement to obtain the specialist title.

Rio de Janeiro

2020

## **SUMMARY**

This application aims to implement an administrative panel for electronic commerce, so that people who do not have advanced knowledge can use it and that it is fast and makes use of modern technologies. With simple and intuitive usability, containing the essential resources for managing a virtual store and that can be used both in desktop environments and in mobile environments, a user can manage their products, their orders and their customers through the administrative panel. Using modern and easy-to-maintain technologies and the creation of new resources, this application is easily scalable and customizable, given that the architecture standard used is MVC (Model-View-Control), and also has a library of tests that verify correct operation. of the main functionalities of the application.

Keywords: Node.js. Express.js. MongoDB. Mongoose. Ecommerce. Web application.

# SUMMARY

<b>1. Presentation</b>	<b>5</b>
1.1. Context	5
1.1. Target Audience	5
1.2. Requirements	6
<b>2. Modeling</b>	<b>8</b>
2.1. Use Case Diagram	8
2.2. actors	8
2.3. Details of use cases	9
2.4. Interfaces	22
2.5. class diagram	38
<b>3. Project</b>	<b>40</b>
3.1. software architecture	40
3.2. Information Architecture	41
<b>4. Testes</b>	<b>43</b>
<b>5. URLs</b>	<b>45</b>
5.1. web application	45
5.2. source code repository	45
5.3. Job presentation video	45
<b>REFERENCES</b>	<b>46</b>

## **1. Presentation**

### **1.1. Context**

The purpose of the application is to provide a user-friendly interface for managing an e-commerce business, as the existing ones on the market are often very complicated to use, and have many features that are not used. The use of the application is made for people or small companies that want to simplify or even start their online business but with little investment power, the technologies chosen for the development of the application are the most current in the market and the simplest, thus giving powers to the owner of the code can scale the application according to his needs.

### **1.1. Target Audience**

In this application the target audience are people or companies that need an administrative panel for their online store, the application is easy to understand and requires little prior knowledge of e-commerce so that the application can be used in its entirety, knowledge such as the basics of SEO (Search Engine Optimization), knowledge in Marketing for better exposure and architecture of store products. The application has simple resources so that an e-commerce analyst or even the product owner can create, change and delete products and pages with an intuitive and limited interface so that even users who do not have prior knowledge can maintain the application. The expectation is that with a few hours of work you will have a ready-to-use e-commerce ecosystem with different user roles.

### **1.2. Requirements**

The table below presents the list of functional requirements for the application.

<b>ID</b>	<b>Functional requirement</b>	<b>Description</b>
RF-01	Login and Registration	Registration of new users. Registration is necessary to access the full application as well as define the roles of each user and manage the application's features.
RF-02	Functional access control	The application must have user roles distributed between Admin, Analyst and Assistant where the Admin has all the permissions the Analyst can register and make modifications and the Assistant can only see the data.
RF-03	application menu	The application must have a side menu that gives access to all application options.
RF-04	registration screens	In all application options, there should be a main screen where all the data of that entity and its options to add, change or delete can be listed.
RF-05	master/detail screen	The application must have a screen where the details of an entity can be registered.
RF-06	transaction screens	The application must be able to download and import products through .csv files. It should also have the option of registering data related to the business in question, such as name, telephone number and address.
RF-07	Statistics graphic report	The application must have graphics on the main screen that display quantities of products and the number of users and their roles.

The table below presents the list of non-functional requirements of the application.

<b>ID</b>	<b>Functional requirement</b>	<b>Description</b>
RNF-01	OAuth Integration	You must have the option to log in using a Google or LinkedIn account.

RNF-02	Responsiveness	The application must be accessible to all devices remotely, tablets, cell phones and computers.
RNF-03	Database	The application must use a non-relational database such as MongoDB.
RNF-04	server	The application must use Node.js and Express as the main web framework.
RNF-05	Endpoints	The application must have endpoints where the front-end can connect to display the site's data.
RNF-06	Availability	The application must be online 99% of the time.
RNF-07	Programming language	The application must be written in Javascript language.
RNF-08	Cross Browser	The application must work in the main browsers of today.
RNF-09	cloud storage	The download, upload and image files must be stored in the cloud using for example AWS S3.
RNF-10	Versioning	The application must have file version control.

## 2. Modeling

### 2.1. Use Case Diagram

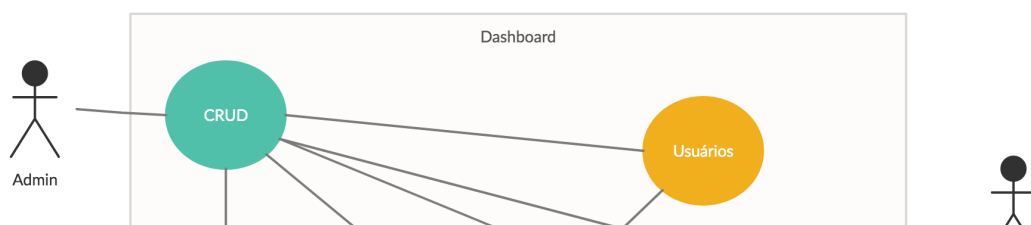


Figure 1: Use case diagram.

## 2.2. actors

The application foresees four actors for interaction with the developed functionalities:

- Admin - Has all permissions within the application, with this user it is possible to view, edit, register and delete freely within the application, this user also has the privilege to change or create the site configuration data.
- Analyst - Has permission to view, create and edit site data, but does not have permission to delete or change site configuration data.
- Assistant - This user can view site information and can create new data, but is not able to edit, delete, or access configuration information.
- Front-End (API) - It only has the option to view data from the main page, products, categories and pages.

## 2.3. Details of use cases

Table 3. Use case breakdown - Login

Name	Login
Description	User must provide email and password to access the application.
actors	Admin, Analyst, Assistant
Precondition	Valid email and password.
Postcondition	The application creates a session for the user.
Basic Flow	<ol style="list-style-type: none"> <li>1. The user enters the website.</li> <li>2. The user enters his email and password in the form fields or use the login option through social</li> </ol>



	<p>networks.</p> <ol style="list-style-type: none"> <li>3. The User clicks the Sign in button.</li> <li>4. The user is redirected to the main screen</li> <li>5. Closed use case.</li> </ol>
Alternative Flow	<ol style="list-style-type: none"> <li>3. The user has an incorrect email or password.             <ol style="list-style-type: none"> <li>3.1 The application returns to step 1 and displays an error message.</li> </ol> </li> </ol>
Exception Flow	<ol style="list-style-type: none"> <li>2. The user does not have a login in the application.             <ol style="list-style-type: none"> <li>2.1 The user must register or use the login through social networks.</li> </ol> </li> </ol>

Table 4. Details of use cases - Registration

Name	User registration
Description	The user logs into the application.
actors	guest user
Precondition	First name, Last name, Email and Password.
Postcondition	The application redirects to the login screen.
Basic Flow	<ol style="list-style-type: none"> <li>1. The user enters the website.</li> <li>2. Click on the Register button.</li> <li>3. The user enters his First Name, Last Name, Email and his password in the fields of the form.</li> <li>4. The User clicks on the Save button.</li> <li>5. The user is redirected to the login screen.</li> <li>6. Closed use case.</li> </ol>
Alternative Flow	<ol style="list-style-type: none"> <li>1. The user uses the login option through social networks.</li> </ol>
Exception Flow	<ol style="list-style-type: none"> <li>4. The user already registered in the application.             <ol style="list-style-type: none"> <li>4.1 The application redirects to the login screen and displays an error message.</li> </ol> </li> </ol>

Table 5. Use case breakdown - Login via social networks

Name	Login via social networks
------	---------------------------

Description	The user must login using Google or LinkedIn.
actors	guest user
Precondition	Registration on social networks.
Postcondition	The application creates a session for the user.
Basic Flow	<ol style="list-style-type: none"> <li>1. The user enters the website.</li> <li>2. The user selects one of the social network options and clicks the button.</li> <li>3. On the screen that opens, the user enters his data.</li> <li>4. The user is redirected to the main screen</li> <li>5. Closed use case.</li> </ol>
Alternative Flow	<ol style="list-style-type: none"> <li>1. The user clicks the Register button.</li> <li>3.1 The user fills in his data and registers in the application.</li> </ol>
Exception Flow	-

Table 6. Use case breakdown - Edit settings

Name	edit settings
Description	The user changes application information.
actors	Admin
Precondition	The user must be logged into the application as admin
Postcondition	The application reloads the edit page.
Basic Flow	<ol style="list-style-type: none"> <li>1. The user enters the website.</li> <li>2. The user logs in.</li> <li>3. The user clicks on the Settings option in the side menu.</li> <li>4. The user changes the data he wants.</li> <li>5. The user clicks save.</li> <li>6. Closed use case.</li> </ol>
Alternative Flow	-
Exception Flow	<ol style="list-style-type: none"> <li>5. An error occurs while saving the data. <ol style="list-style-type: none"> <li>5.1 The application redirects to the 503 error page and displays a message.</li> </ol> </li> </ol>

Table 7. Details of use cases - Download of registered products

Name	Registered product downloads
Description	The user downloads the products registered in the application.
actors	Admin, Analyst
Precondition	The user must be logged into the application as Admin or Analyst
Postcondition	The application reloads the page.
Basic Flow	<ol style="list-style-type: none"> <li>1. The user enters the website.</li> <li>2. The user logs in.</li> <li>3. The user clicks on the Products option in the side menu.</li> <li>4. The user clicks the Export link.</li> <li>5. The user saves the generated csv file.</li> <li>6. Closed use case.</li> </ol>
Alternative Flow	-
Exception Flow	<ol style="list-style-type: none"> <li>5. An error occurs while downloading the data. <ol style="list-style-type: none"> <li>1.1 The application redirects to the 503 error page and displays a message.</li> </ol> </li> </ol>

Table 8. Use case breakdown - Importing products

Name	Import of products
Description	User imports products from a csv file.
actors	Admin, Analyst
Precondition	The user must be logged into the application as Admin or Analyst
Postcondition	The application redirects to the products page.
Basic Flow	<ol style="list-style-type: none"> <li>1. The user enters the website.</li> <li>2. The user logs in.</li> <li>3. The user clicks on the Products option in the side menu.</li> <li>4. The user clicks the Import link.</li> <li>5. The user clicks the Choose File button.</li> </ol>

	<ol style="list-style-type: none"> <li>6. The user chooses the file to be imported.</li> <li>7. The user clicks the Upload button.</li> <li>8. Closed use case.</li> </ol>
Alternative Flow	-
Exception Flow	<ol style="list-style-type: none"> <li>7. An error occurs while downloading the data.             <ol style="list-style-type: none"> <li>1.1 The application redirects to the 503 error page and displays a message.</li> </ol> </li> </ol>

Table 9. Use case breakdown - Product listing

Name	product listing
Description	The user lists all the products registered in the application.
actors	Admin, Analyst, Assistant, Front-end
Precondition	The user must be logged into the application or access the application through the /api/products endpoint.
Postcondition	The application returns a list of products.
Basic Flow	<ol style="list-style-type: none"> <li>1. The user enters the website.</li> <li>2. The user logs in.</li> <li>3. The user clicks on the Products option in the side menu.</li> <li>4. Closed use case.</li> </ol>
Alternative Flow	-
Exception Flow	<ol style="list-style-type: none"> <li>3. An error occurs while saving the data.             <ol style="list-style-type: none"> <li>1.1 The application redirects to the 503 error page and displays a message.</li> </ol> </li> </ol>

Table 10. Use case breakdown - Product registration

Name	Product registration
Description	The user registers products in the application.

actors	Admin, Analyst, Assistant
Precondition	The user must be logged into the application as Admin or Analyst
Postcondition	The application redirects to the products page.
Basic Flow	<ol style="list-style-type: none"> <li>5. The user enters the website.</li> <li>6. The user logs in.</li> <li>7. The user clicks on the Products option in the side menu.</li> <li>8. The user clicks the New Product button.</li> <li>9. The user fills in the data.</li> <li>10. The user clicks the Save button.</li> <li>11. Closed use case.</li> </ol>
Alternative Flow	-
Exception Flow	<ol style="list-style-type: none"> <li>6. An error occurs while saving the data. <ol style="list-style-type: none"> <li>1.1 The application redirects to the 503 error page and displays a message.</li> </ol> </li> </ol>

Table 11. Use case breakdown - Editing products

Name	Product editing
Description	The user edits products in the application.
actors	Admin, Analyst
Precondition	The user must be logged into the application as Admin or Analyst
Postcondition	The application redirects to the products page.
Basic Flow	<ol style="list-style-type: none"> <li>1. The user enters the website.</li> <li>2. The user logs in.</li> <li>3. The user clicks on the Products option in the side menu.</li> <li>4. The user chooses the product to be edited and clicks on the edit button.</li> <li>5. The user fills in the data.</li> <li>6. The user clicks the Save button.</li> <li>7. Closed use case.</li> </ol>
Alternative Flow	-

Exception Flow	<p>6. An error occurs while saving the data.</p> <p>6.1 The application redirects to the 503 error page and displays a message.</p>
----------------	---

Table 12. Use case breakdown - Delete products

Name	delete products
Description	The user deletes the product in the application.
actors	Admin
Precondition	The user must be logged into the application as Admin
Postcondition	The application redirects to the products page.
Basic Flow	<ol style="list-style-type: none"> <li>1. The user enters the website.</li> <li>2. The user logs in.</li> <li>3. The user clicks on the Products option in the side menu.</li> <li>4. The user chooses the product to be deleted and clicks on the delete link.</li> <li>5. The user confirms the action.</li> <li>6. Closed use case.</li> </ol>
Alternative Flow	-
Exception Flow	<p>5. An error occurs while deleting the data.</p> <p>5.1 The application redirects to the 503 error page and displays a message.</p>

Table 13. Use Case Breakdown - Category Listing

Name	category listing
Description	The user lists all categories registered in the application.
actors	Admin, Analyst, Assistant, Front-end
Precondition	The user must be logged into the application or access the application through the /api/categories endpoint.
Postcondition	The application returns a list of categories.

Basic Flow	<ol style="list-style-type: none"> <li>1. The user enters the website.</li> <li>2. The user logs in.</li> <li>3. The user clicks on the optionCategories no menu lateral.</li> <li>4. Closed use case.</li> </ol>
Alternative Flow	-
Exception Flow	<ol style="list-style-type: none"> <li>4. An error occurs while saving the data. <ol style="list-style-type: none"> <li>1.1 The application redirects to the 503 error page and displays a message.</li> </ol> </li> </ol>

Table 14. Details of use cases - Registration of categories

Name	Category registration
Description	The user registers a category in the application.
actors	Admin, Analyst, Assistant
Precondition	The user must be logged into the application as Admin or Analyst
Postcondition	The application redirects to the category page.
Basic Flow	<ol style="list-style-type: none"> <li>1. The user enters the website.</li> <li>2. The user logs in.</li> <li>3. The user clicks on the optionCategories no menu lateral.</li> <li>4. The user clicks the New Category button.</li> <li>5. The user fills in the data.</li> <li>6. The user clicks the Save button.</li> <li>7. Closed use case.</li> </ol>
Alternative Flow	-
Exception Flow	<ol style="list-style-type: none"> <li>6. An error occurs while saving the data. <ol style="list-style-type: none"> <li>6.1 The application redirects to the 503 error page and displays a message.</li> </ol> </li> </ol>

Table 15. Use case breakdown - Editing categories

Name	category editing
Description	The user edits categories in the application.
actors	Admin, Analyst
Precondition	The user must be logged into the application as Admin or Analyst
Postcondition	The application redirects to the category page.
Basic Flow	<ol style="list-style-type: none"> <li>1. The user enters the website.</li> <li>2. The user logs in.</li> <li>3. The user clicks on the optionCategories no menu lateral.</li> <li>4. The user chooses the category to be edited and clicks on the edit button.</li> <li>5. The user fills in the data.</li> <li>6. The user clicks the Save button.</li> <li>7. Closed use case.</li> </ol>
Alternative Flow	-
Exception Flow	<ol style="list-style-type: none"> <li>6. An error occurs while saving the data. <ol style="list-style-type: none"> <li>6.1 The application redirects to the 503 error page and displays a message.</li> </ol> </li> </ol>

Table 16. Use case breakdown - Delete category

Name	delete categories
Description	The user deletes categories in the application.
actors	Admin
Precondition	The user must be logged into the application as Admin
Postcondition	The application redirects to the category page.
Basic Flow	<ol style="list-style-type: none"> <li>1. The user enters the website.</li> <li>2. The user logs in.</li> <li>3. The user clicks on the optionCategories no menu lateral.</li> <li>4. The user chooses the category to be deleted and clicks on the delete link.</li> <li>5. The user confirms the action.</li> </ol>



	6. Closed use case.
Alternative Flow	-
Exception Flow	5. An error occurs while deleting the data. 1.1 The application redirects to the 503 error page and displays a message.

Table 17. Use case breakdown - User listing

Name	user listing
Description	The user lists all users registered in the application.
actors	Admin, Analyst, Assistant
Precondition	The user must be logged into the application.
Postcondition	The application returns a list of users.
Basic Flow	<ol style="list-style-type: none"> <li>1. The user enters the website.</li> <li>2. The user logs in.</li> <li>3. The user clicks on the Products option in the side menu.</li> <li>4. Closed use case.</li> </ol>
Alternative Flow	-
Exception Flow	3. An error occurs while saving the data. 1.1 The application redirects to the 503 error page and displays a message.

Table 18. Use case details - User registration

Name	User registration
Description	The user registers a user in the application.
actors	Admin, Analyst
Precondition	The user must be logged into the application as Admin or

	Analyst
Postcondition	The application redirects to the users page.
Basic Flow	<ol style="list-style-type: none"> <li>1. The user enters the website.</li> <li>2. The user logs in.</li> <li>3. The user clicks on the Customers option in the side menu.</li> <li>4. The user clicks the New Customer button.</li> <li>5. The user fills in the data.</li> <li>6. The user clicks the Save button.</li> <li>7. Closed use case.</li> </ol>
Alternative Flow	-
Exception Flow	<ol style="list-style-type: none"> <li>6. An error occurs while saving the data. <ol style="list-style-type: none"> <li>6.1 The application redirects to the 503 error page and displays a message.</li> </ol> </li> </ol>

Table 19. Use case breakdown - Editing users

Name	User Editing
Description	The user edits categories in the application.
actors	Admin, Analyst
Precondition	The user must be logged into the application as Admin or Analyst
Postcondition	The application redirects to the users page.
Basic Flow	<ol style="list-style-type: none"> <li>1. The user enters the website.</li> <li>2. The user logs in.</li> <li>3. The user clicks on the Customers option in the side menu.</li> <li>4. The user chooses the data to be edited and clicks on the edit button.</li> <li>5. The user fills in the data.</li> <li>6. The user clicks the Save button.</li> <li>7. Closed use case.</li> </ol>
Alternative Flow	-
Exception Flow	<ol style="list-style-type: none"> <li>6. An error occurs while saving the data. <ol style="list-style-type: none"> <li>6.1 The application redirects to the 503 error page and displays a message.</li> </ol> </li> </ol>

Table 20. Use case breakdown - Delete user

Name	delete users
Description	The user deletes users in the application.
actors	Admin
Precondition	The user must be logged into the application as Admin
Postcondition	The application redirects to the users page.
Basic Flow	<ol style="list-style-type: none"> <li>1. The user enters the website.</li> <li>2. The user logs in.</li> <li>3. The user clicks on the Customers option in the side menu.</li> <li>4. The user chooses the user to be deleted and clicks on the delete link.</li> <li>5. The user confirms the action.</li> <li>6. Closed use case.</li> </ol>
Alternative Flow	-
Exception Flow	<ol style="list-style-type: none"> <li>5. An error occurs while deleting the data. <ol style="list-style-type: none"> <li>5.1 The application redirects to the 503 error page and displays a message.</li> </ol> </li> </ol>

Table 21. Use Case Breakdown - Page Listing

Name	Page listing
Description	The user lists all pages registered in the application.
actors	Admin, Analyst, Assistant, Front-end
Precondition	The user must be logged into the application or access through the /api/pages endpoint.
Postcondition	The application returns a list of pages.
Basic Flow	<ol style="list-style-type: none"> <li>1. The user enters the website.</li> <li>2. The user logs in.</li> <li>3. The user clicks on the Products option in the side</li> </ol>

	menu. 4. Closed use case.
Alternative Flow	-
Exception Flow	3. An error occurs while saving the data. 1.1 The application redirects to the 503 error page and displays a message.

Table 22. Use case breakdown - Page registration

Name	Page registration
Description	The user registers a page in the application.
actors	Admin, Analyst, Assistant
Precondition	The user must be logged into the application as Admin or Analyst
Postcondition	The application redirects to the page listing.
Basic Flow	<ol style="list-style-type: none"> <li>1. The user enters the website.</li> <li>2. The user logs in.</li> <li>3. The user clicks on the Pages option in the side menu.</li> <li>4. The user clicks the New Page button.</li> <li>5. The user fills in the data.</li> <li>6. The user clicks the Save button.</li> <li>7. Closed use case.</li> </ol>
Alternative Flow	-
Exception Flow	6. An error occurs while saving the data. 6.1 The application redirects to the 503 error page and displays a message.

Table 23. Use case breakdown - Editing pages

Name	Editing Pages
------	---------------

Description	The user edits pages in the application.
actors	Admin, Analyst
Precondition	The user must be logged into the application as Admin or Analyst
Postcondition	The application redirects to the page listing.
Basic Flow	<ol style="list-style-type: none"> <li>1. The user enters the website.</li> <li>2. The user logs in.</li> <li>3. The user clicks on the Pages option in the side menu.</li> <li>4. The user chooses the page to be edited and clicks on the edit button.</li> <li>5. The user fills in the data.</li> <li>6. The user clicks the Save button.</li> <li>7. Closed use case.</li> </ol>
Alternative Flow	-
Exception Flow	<ol style="list-style-type: none"> <li>6. An error occurs while saving the data. <ol style="list-style-type: none"> <li>6.1 The application redirects to the 503 error page and displays a message.</li> </ol> </li> </ol>

Table 24. Use Case Breakdown - Clear Page

Name	turn off pages
Description	The user deletes a page in the application.
actors	Admin
Precondition	The user must be logged into the application as Admin
Postcondition	The application redirects to the page listing.
Basic Flow	<ol style="list-style-type: none"> <li>1. The user enters the website.</li> <li>2. The user logs in.</li> <li>3. The user clicks on the Customers option in the side menu.</li> <li>4. The user chooses the data to be deleted and clicks on the delete link.</li> <li>5. The user confirms the action.</li> <li>6. Closed use case.</li> </ol>
Alternative Flow	-

Exception Flow	5. An error occurs while deleting the data. 5.1 The application redirects to the 503 error page and displays a message.
----------------	--

## 2.4. Interfaces

### 2.4.1 Login screen

The first screen presented when we access the application url is the login screen, this screen brings a form to login to the application using previously registered email and password, a button to register in the application and two buttons to login using the networks social media Google or LinkedIn.

Dashboard Ecommerce

Register

Or

Sign in

admin

.....

Sign in

Sign In with Google

Sign in with LinkedIn

© 2020 - Dashboard Ecommerce

Figure 2: Login screen.

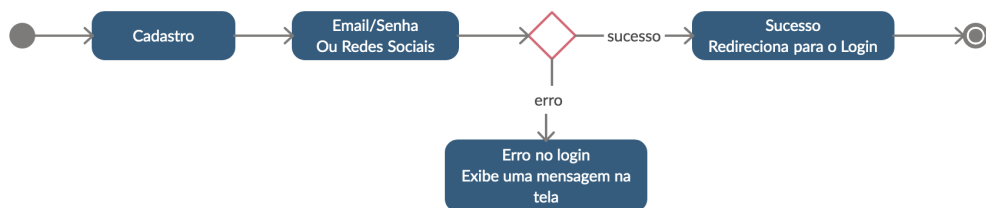


Figure 3: Diagram of States - Login.

## 2.4.2 Registration Screen


This screen presents a form to fill in the following data to register in the application: First name, second name, email and password, all registered users receive the role of analyst until someone with admin permission changes this role for that user in question .

First Name

Last Name

Email

Password

 Save

© 2020 - Dashboard Ecommerce

Figure 4: Registration screen.

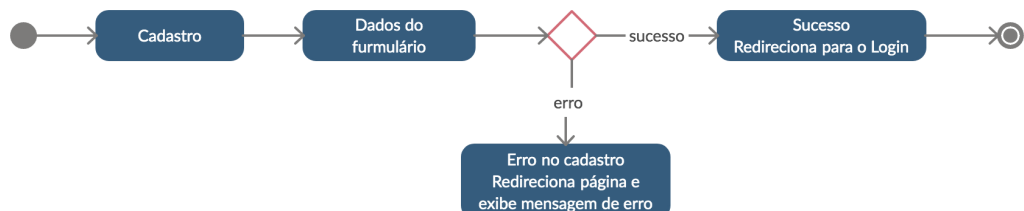


Figure 5: State Diagram - Registration.



### 2.4.3 Main fabric

This is the application's main screen where a welcome message is displayed. Just below this message, two graphs are presented, the first of which is a comparison of how many products in the application are active and how many are deactivated, and the other graph shows the average of roles of user this is how many users are admin, analyst and assistant, in addition to the side navigation bar where you can access other application options.



Figure 6: Main canvas, Desktop and Mobile.

### 2.4.4 Category list screen

Here are displayed the registered categories as well as their links to edit, delete and a button to register new categories.

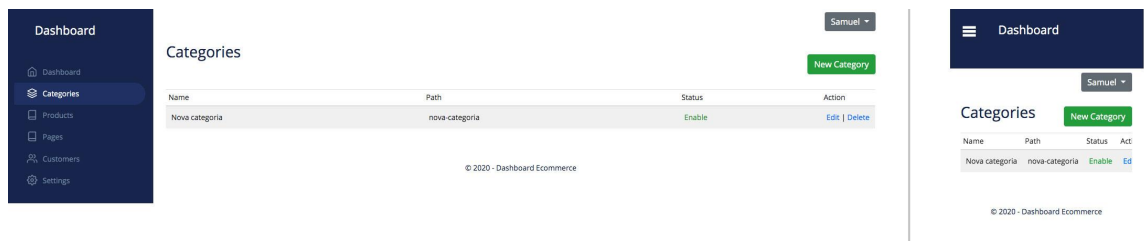


Figure 7: Category Canvas, Desktop and Mobile.



Figure 9: New category screen, Desktop.

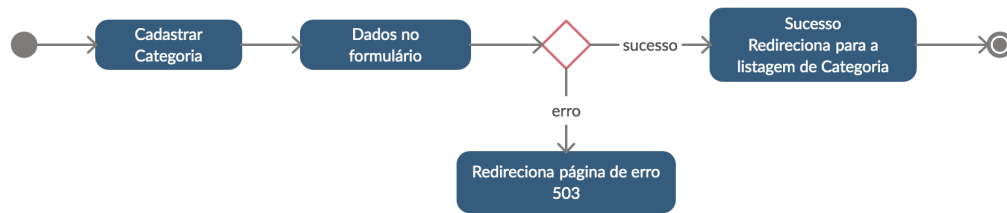





Figure 10: State Diagram - Registration of categories.

## 2.4.6 Category editing screen

On this screen, the user can change the data registered in a specific category.

The screenshot shows the 'Edit Category' interface within a dashboard. On the left is a dark sidebar with navigation options: Dashboard, Categories, Products, Pages, Customers, and Settings. The main content area is titled 'Edit Category' and contains the following elements:

- Name:** Nova categoria
- Url:** nova-categoria
- Content:** A rich text editor with a menu (File, Edit, View, Insert, Format, Table, Help) and a toolbar. The content area contains the text 'nova categoria' and a word count of '2 WORDS POWERED BY TINY'.
- Products:** A table listing products associated with the category.
 

Select	SKU	Thumb	Name	Price	Status
<input type="checkbox"/>	chain-bracelet		7 Shakra Bracelet	\$44.99	Enable
<input type="checkbox"/>	bangle-bracelet		Bangle Bracelet	\$43.99	Enable
<input type="checkbox"/>	bangle-bracelet-with-feathers		Boho Bangle Bracelet	\$44.99	Enable
- Status:** A dropdown menu currently set to 'Enable'.
- Save:** A blue button with a save icon.

© 2020 - Dashboard Ecommerce

Figure 11: Category editing screen, Desktop.

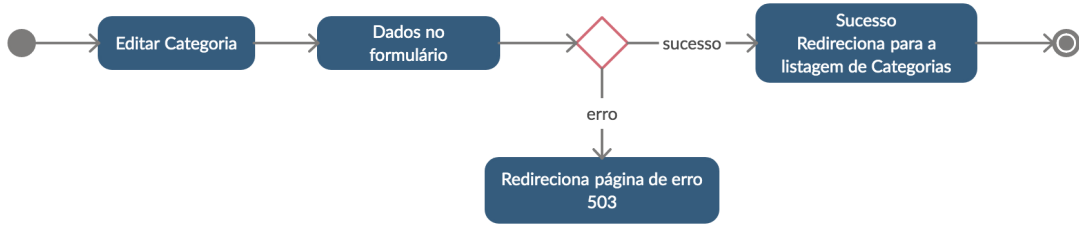


Figure 12: State Diagram - Editing Categories.

### 2.4.7 Product list screen

All products are listed on this screen, as well as links to actions on products such as deleting and editing, as well as buttons for registering new products, importing products and exporting products to a .csv file.

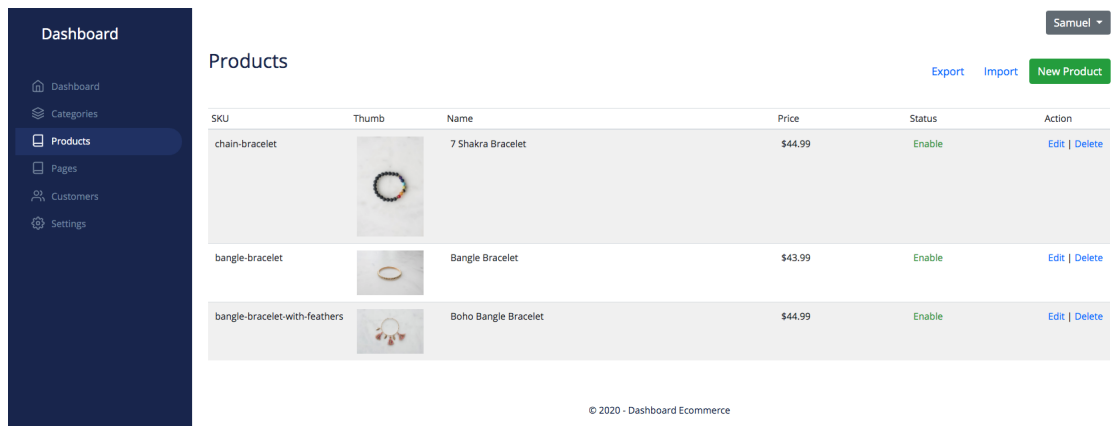


Figure 13: Product listing screen, Desktop.

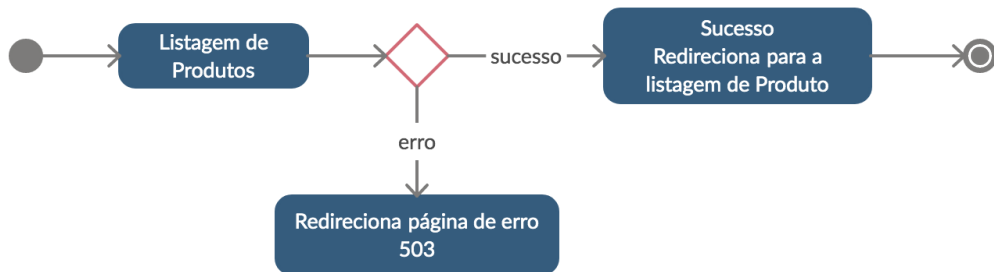


Figure 14: State Diagram - Product listing.

## 2.4.8 Product registration screen

On this screen, you can register a new product by providing the following data for registration: Product name, URL, SKU, Price, Special Price (if any), Description, Quantity, Status, image and a button to save the new product.

Figure 15: Product registration screen, Desktop.

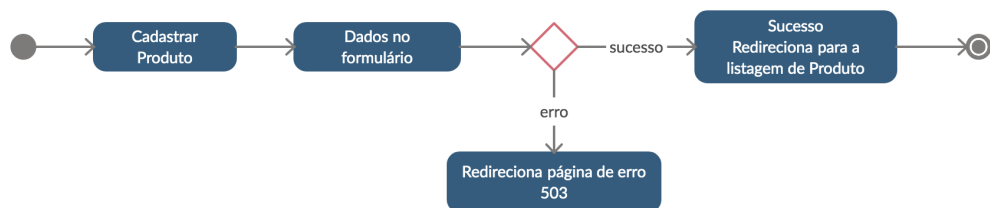


Figure 16: Status Diagram - Product registration.

## 2.4.9 Product edit screen

In this screen it is possible to edit the data of a previously registered product.

Figure 17: Product editing screen, Desktop.

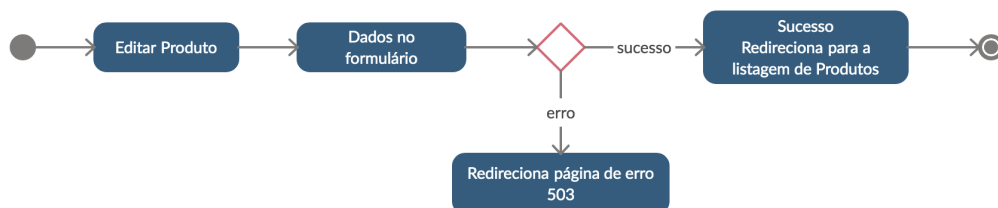


Figure 18: State Diagram - Product edition.

## 2.4.10 Page list screen

This screen lists all the dynamic pages registered in the application, it also has

buttons to edit, delete a product in question and a button to register a new page.

Title	Path	Status	Action
Minha Primeira Página	minha-primeira-pagina	Enable	Edit   Delete
Segunda página	segunda-pagina	Disable	Edit   Delete
Testee	teste	Enable	Edit   Delete
testando	testando	Enable	Edit   Delete

© 2020 - Dashboard Commerce

Figure 19: Page list screen.

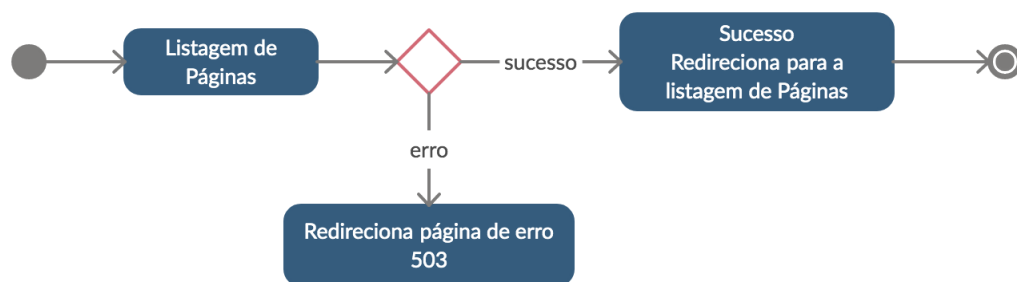


Figure 20: State Diagram - Page listing.

### 2.4.11 Page registration screen

In this screen it is possible to register a new page by inserting the following data: Title of the page, Url, content, status and button to save the new page.

Figure 21: Page registration screen, Desktop.

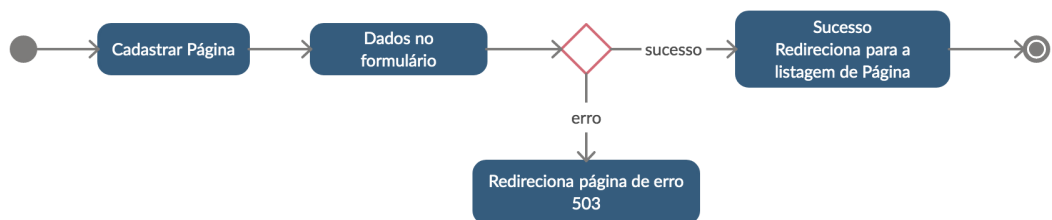


Figure 22: State Diagram - Page Editing.



### 2.4.12 Page editing screen

In this screen it is possible to edit an existing page by changing the data already inserted in the page.

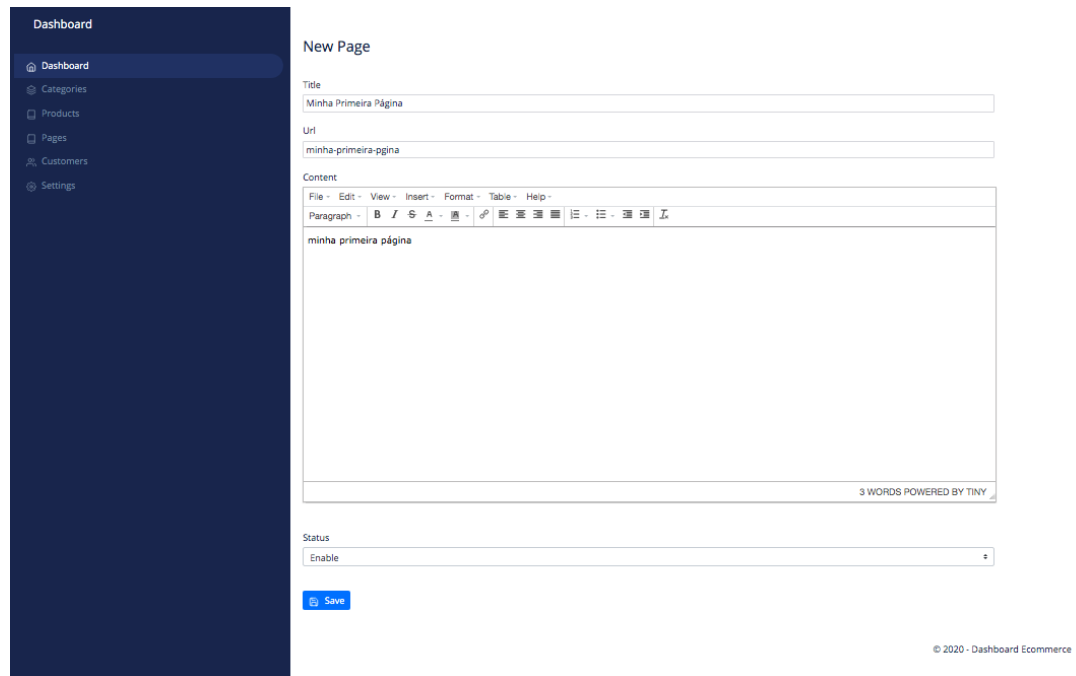


Figure 23: Page editing screen, Desktop.

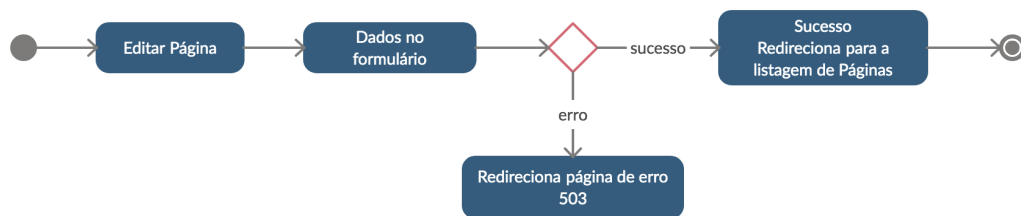
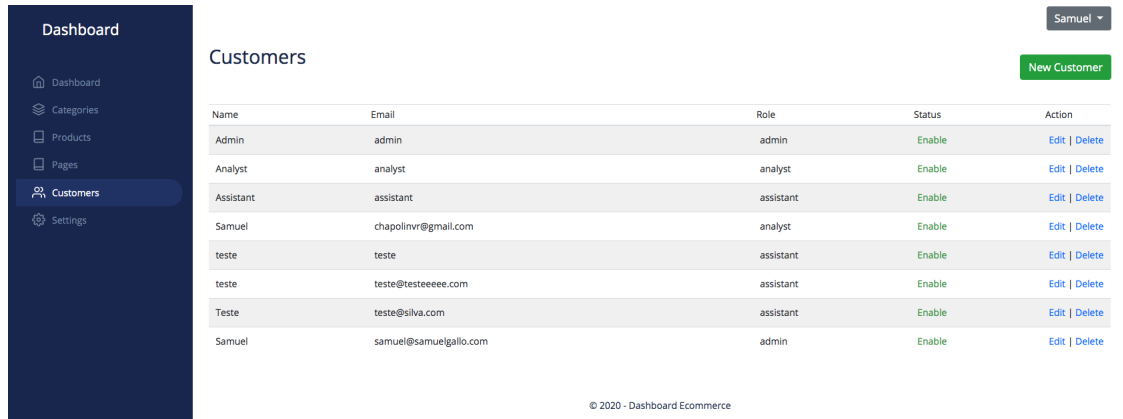


Figure 24: State Diagram - Editing page.

### 2.4.13 User list screen

This screen lists all users as well as links to user actions such as deleting, editing and even buttons to register new users.



Name	Email	Role	Status	Action
Admin	admin	admin	Enable	Edit   Delete
Analyst	analyst	analyst	Enable	Edit   Delete
Assistant	assistant	assistant	Enable	Edit   Delete
Samuel	chapolinrv@gmail.com	analyst	Enable	Edit   Delete
teste	teste	assistant	Enable	Edit   Delete
teste	teste@testeeeee.com	assistant	Enable	Edit   Delete
Teste	teste@silva.com	assistant	Enable	Edit   Delete
Samuel	samuel@samuegallo.com	admin	Enable	Edit   Delete

Figure 25: User list screen, Desktop.

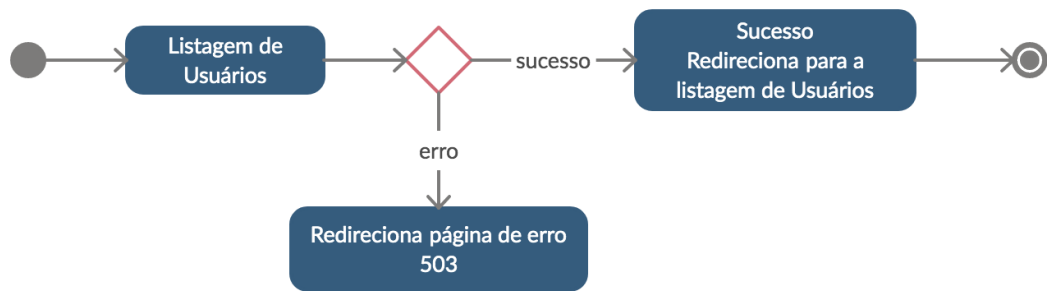


Figure 26: State Diagram - List of users.

### 2.4.14 User registration screen

On this screen, it is possible to register a new user by filling in the following data: first name, second name, email, telephone, date of birth, address, user status, user role and password.

Figure 27: Registration screen of user, Desktop.

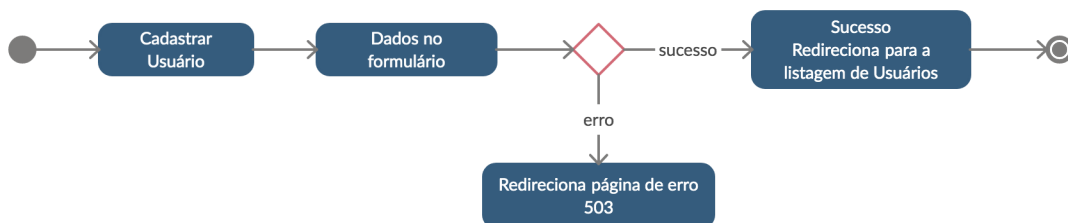


Figure 28: Status Diagram - User registration.

### 2.4.16 User edit screen

On this screen, it is possible to edit the data of an already registered user, such as the user's role in the application.

Figure 29: Edit screen of user, Desktop.

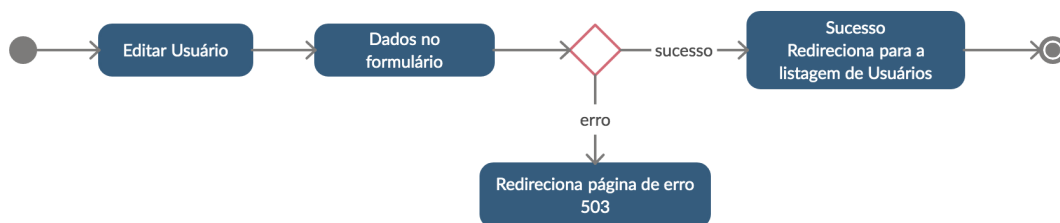


Figure 30: State Diagram - User edition.

### 2.4.17 Settings screen

All the application settings are displayed here, such as the name of the store, email, telephone, copyright, opening hours, address and logo, on this screen it is also possible to edit this information.

Figure 31: Settings editing screen, Desktop.

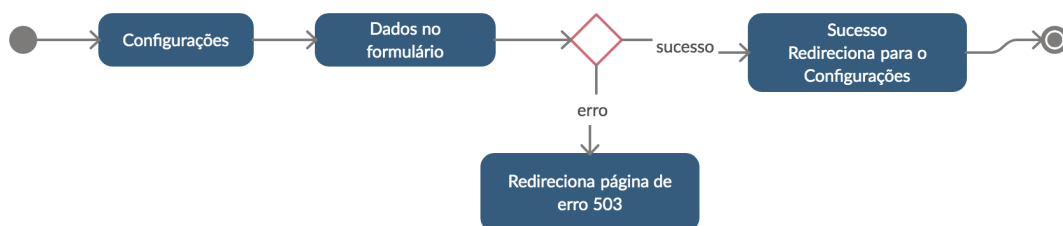


Figure 32: State Diagram - Settings.

## 2.5. class diagram

In the application, a NoSQL database was used because it has a more flexible structure, so we can conclude that the application code is responsible for the data relationship.

Composed of 5 Collections: categories, customers, pages, products, settings, and the only relationship found is categories, a category menu item where, within a category item, it is possible to link products to a category through the “\_id” of a product, a one-to-some (1:N) relationship. The NoSQL primary key is usually an Object that automatically receives an id when creating the data.

When creating new data or editing, a method called “timestamps” was included in the application, in which two fields are created, createdAt and updatedAt, these fields automatically insert the edit or creation date of the data.

The users class has a mechanism to encrypt the user's password at the time of creation, and every time the login is performed, this password is validated by the encryption application. This class has basic data and data that predict a future implementation of codes. In the settings class it is possible to define all the data related to the application, the class of products, categories and pages they receivedata formatted ofone WYSIWYG editor that allows the customization of information available for the front-end.

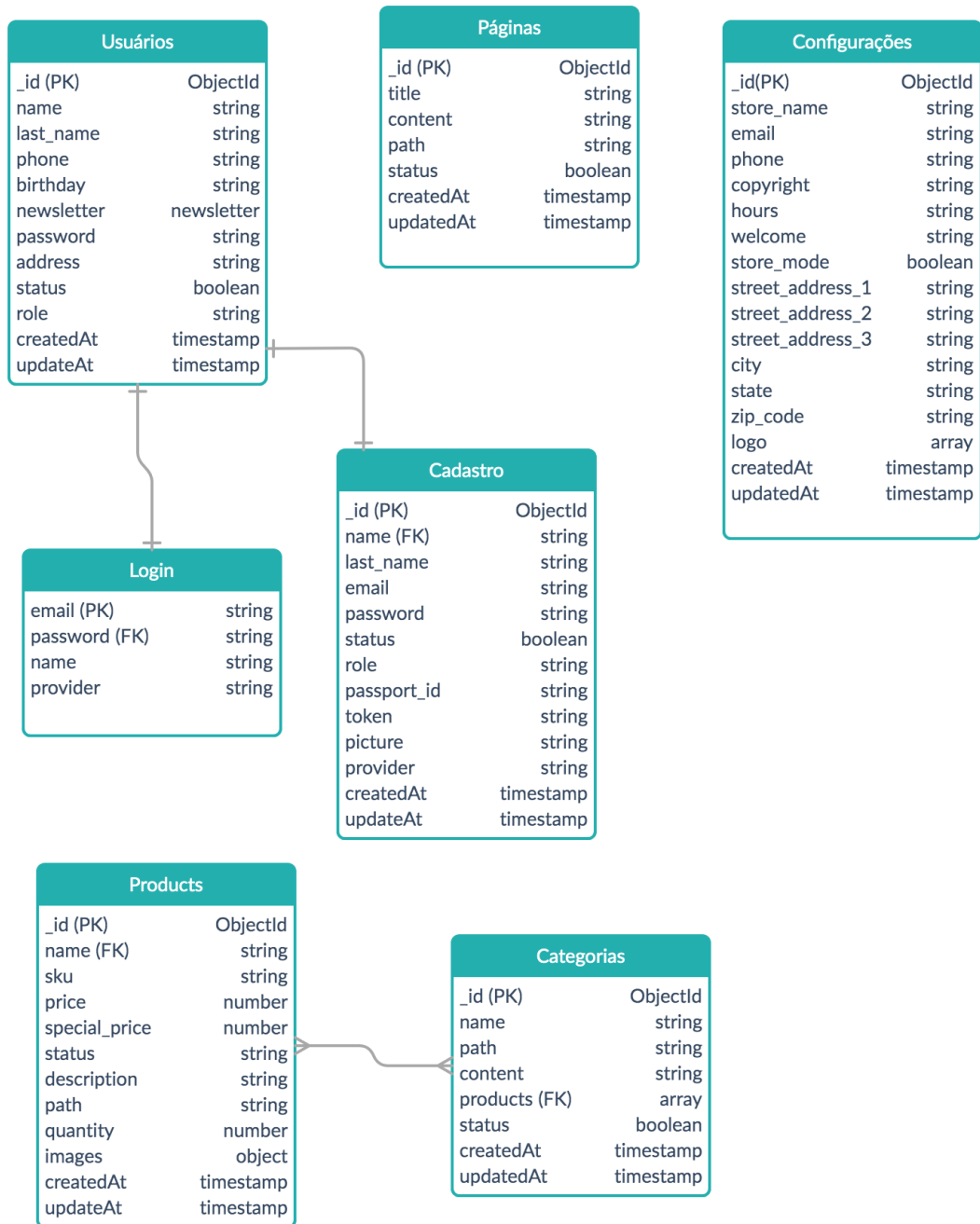


Figure 33: Class diagram.

### 3. Project

#### 3.1. architecture of software

The architectural pattern chosen was the Model View Controller (MVC) because it is a well-known pattern in application development. The programming language chosen was JavaScript because it has a complete Stack for building a complete application and with several open source resources available. The backend was built on top of Node.js because it has high scalability and performance, in addition to a package manager called NPM (Node Package Manager) that has many code library options, thus speeding up the development of an application. The web framework used was Express.js for being minimalist and having great adherence by the community, thus facilitating the support of the developed codes. To manage the application on the server, the PM2 package was used, whose main objective is not to stop the application in case of errors or a possible overload in the application.

The database used was MongoDB using the Mongoose library which is a powerful object data modeling library (ODM) based on schemas where Mongoose retrieves data from the database and converts it to javascript and can easily be used within the application. .

The login is done on top of a package called Passport.js which provides login capabilities to various social networks, and the native login was done using the bcrypt package to encrypt user passwords at registration, and user sessions use an Express.js framework add-on.

The application also has an integration with AWS S3, which is an Amazon storage service, where the images of the forms are stored as well as the “.csv” files, that is, if the application is not running locally, all images and files are sent to S3. The application uses a package to manage the environment variables where the password of the services used is located.



The pages displayed on the front-end are rendered on the back-end as they follow the EJS package standard (Embedded JavaScript templates), the application pages were made using HTML, there was the inclusion of the Bootstrap css library in order to facilitate the application of styles in the pages, and Google Fonts to apply new fonts to the texts of the application, the jQuery library was included for small interactions in the forms.

### **3.2. Information Architecture**

The information was arranged in a way that is simple to understand, with few colors to avoid confusion for the user. On the login screen, the buttons are positioned sequentially and the form fields have a description of what needs to be filled out. On the registration screen, the form fields have a label with the description of each item.

On the main screen, graphs of active pages and types of users are displayed, a welcome message for the user, his name and a button on the right side at the top with actions to edit profile and exit the application.

The side navigation menu has succinct items for navigating the system page options, and displays a different color on the active item to demonstrate the current location of the user, when the user is navigating through a mobile device the navigation menu is presented through a “hamburger” icon that by global convention represents a menu. Each item in this navigation has a self-descriptive word to access an application option, in addition the application presents some icons in order to make each action more dynamic.

The internal content of the application's pages mostly have a table listing all the data contained in that application's option. displayed and in the right corner of this information there are action links, which can be to edit, delete or preview a data completely, and at the top we can see a button to create a new data.

When the user clicks on an action link to register new data, a page is displayed that contains a form to be filled out, this form will have a label over the text or selection field, where the user must fill in what is asked to succeed in saving this new data.

When the user chooses to edit some pre-existing data, he will be directed to a page where a form with the fields already filled in will be displayed, thus only needing to change the data he wants.

The forms displayed on the site are labeled with words that are easy to understand so that it does not become something complicated to register or change. As it is a small application with intuitive buttons and does not have a search system, when saving some data from a form, a POST request is made to the controller of the data category in question where the treatment is carried out and this data is saved in the database of data, and then the application returns to the listing page, this listing page is returned through a GET type request for the specific controller of that category.

The application has a list of routes, that is, url paths for each functionality and within this route, the url is separated by type of request, be it GET to return data, POST to save data, DELETE to delete data, editing is done through the POST method as HTML forms do not have the PUT method as an option.

The data are grouped and accessible by their menu items so that when a menu item is accessed, the user has access to all the options available for that item on the list screen.

#### **4. Testes**

Some unit tests were performed using the Mocha test framework and the Chai library to verify the behavior of the results. The tests consist of checking the return of

the urls present in the application as well as their status based on the standardization of the HTTP Status Codes.

A login test was also carried out where the objective was to test a login in the application automatically and also test a login failure in the application returning the error code 401. And in the sequence a test is performed using the GET method to verify if the server is working and responding effectively. A test related to the products was carried out in order to verify the application's ability to automatically register a product and delete it from the database.

```
App
Mongoose is connected
(node:47254) [DEP0066] DeprecationWarning: OutgoingMessage.prototype._headers is deprecated
  ✓ should GET / (77ms)

Login Controller
  ✓ should fail POST /login/auth (59ms)
  ✓ should success login POST /login/auth (47ms)

Product Test
  ✓ Creates a product
  ✓ Remove a product

Routes
  ✓ should GET /login
  ✓ should GET /dashboard
  ✓ should GET /logout
  ✓ should GET /api

9 passing (307ms)
```

Figure 33: Tests - Test results.

The usability tests were carried out experimentally, aiming at the basic use of the application and considering the student nature of the application. The interface was inspired by other e-commerce applications available on the market, applications that have great user adherence, we can see this in front of the screens presented where the menu is allocated laterally to facilitate access, the login screen being displayed in a simple and lean as well as the other pages.

A quick assessment was carried out with 5 users, 3 of which have an advanced use profile, 1 intermediate and 1 basic knowledge.

1. For 4 users the interface is too simple and needs more usability.
2. 2 out of 5 users could log in using a social network, register a product and a dynamic page without much problem.
3. 1 user had problems with login due to not having the resource to redeem the password.
4. 3 users complained about the lack of messages when an action is performed or an error occurs.
5. 2 complained about the lack of resources, such as the possibility of registering more than one image for products and the lack of the application's final interface (Front-End) where pages and published products can be seen.
6. 3 users reported having no problems with the accessibility of the side menu.
7. 4 users did not have problems using a mobile device to access the application due to the interface.

In general, it was observed that the interface needs improvements and that new functionalities need to be implemented, such as integration with means of payment, documentation for using the application and customization of the graphics presented.

## **5. URLs**

### **5.1. source code repository**

<https://github.com/samuelgallo/ecommerce-dashboard>

### **5.2. Job presentation video**

<https://youtu.be/OmJCnhJKjag>

## REFERENCES

Powers, Shelley. **Learning Node: Using JavaScript on the Server**. Novatec Editora, 2017.

Stefanov, Stoyan. **JavaScript standards**. Novatec Editora, 2010.