



# Pet Shop

Introduction to Web Development / 1s2020

Bruno Baldissera  
Bruno Gazoni  
Rafael Ceneme

10724351  
7585037  
9898610

# 1. Requirements

## 1.1 Two types of users:

Admins: manage other admins, customers, products and services.

Customers: normal users who access the site to acquire goods and services.

note: the application comes with an admin account already registered:

username: sindaco

password: bauret

## 1.2 Admin contains name, id, photo, phone and email data.

## 1.3 Customers contain name, id, photo, phone, email, address, an array of their pets and an array representing their product additions to cart..

## 1.4 Each pet is registered under a customer, containing name, id, photo, race and age

## 1.5 A product's record include name, id, photo, description, price, sold and in-stock quantities.

## 1.6 Services are made up of name, id, photo, description and price.

## 1.7 There is a calendar functionality for customers to schedule services:

Displays free slots for use (customers cannot cancel an appointment)

Covers 10 next weeks with 10 slots per day

Hours with already scheduled services are displayed with the animals name

## 1.8 To buy products, the customer can select them from a list, choose the quantity and add to a cart. The payment is done via credit card.

## 1.9 Admins can edit all products and services.

## 1.10 Admins can add new products and services

## 1.11 There is a "earnings screen" showing a list of products and services sold

## 2. Project Description

As per the specifications provided in the course's google classroom, this project's main goal was to build, using various resources required in the course, a website for a pet shop type of establishment. For simplicity's sake, we chose not to name the hypothetical company anything, maintaining the simple, self-evident name, "petshop", whenever the use of a specific company name was necessary (for example, in a logo). Furthermore, the website itself, as described in section 1 of this document, was expected to function as an online store for the company, and not only to promote a physical store, so, that naturally increased the level of complexity of the project, requiring more intelligent and broad solutions.

### Development Stack:

The stack used for the web app development was based in the MEVN core, which means we used mongoDB for the database, the Express framework for building our API and handling server-side structures, Vue.js for the front-end user interfaces and Node.js as the engine for the project.

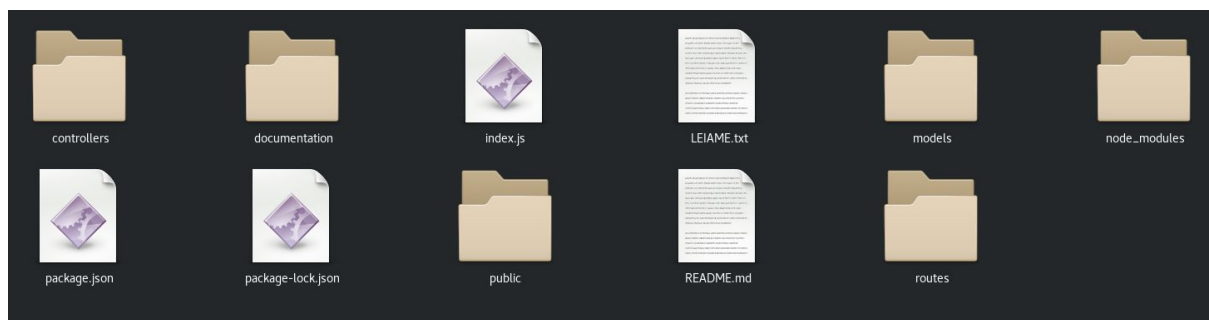
### RESTful API:

Much effort was moved into building our REST server-side architecture, creating routes and logical communication between the components of the web application, using the MEVN stack mentioned above.

The structure was based around an index.js file which established the router, includes the node modules, connects to the database and creates a static link to the public directory of the application where lies the html files.

The models folder retains the database models and schema methods, while the controllers folder is responsible for holding the more elaborate queries used in the application, communicating directly with the models, and finally the routes file links these to end-points accessible to the client side using axios requests to the custom urls created.

File structure:



### Authentication:

Initially we would use Passport and JWT-based methods for authenticating our users and admins, but since this feature was added in late stages of development, we opted to

handcraft the authentication methods using the crypto lib, which helped us in generating hashes and salt strings to store client data more securely in the mongoDB database.

Hashing and salting for the client schema:

```
clientSchema.methods.setPassword = function(password) {
  this.salt = crypto.randomBytes(16).toString('hex');
  this.hash = crypto.pbkdf2Sync(password, this.salt, 10000, 512, 'sha512').toString('hex');
};

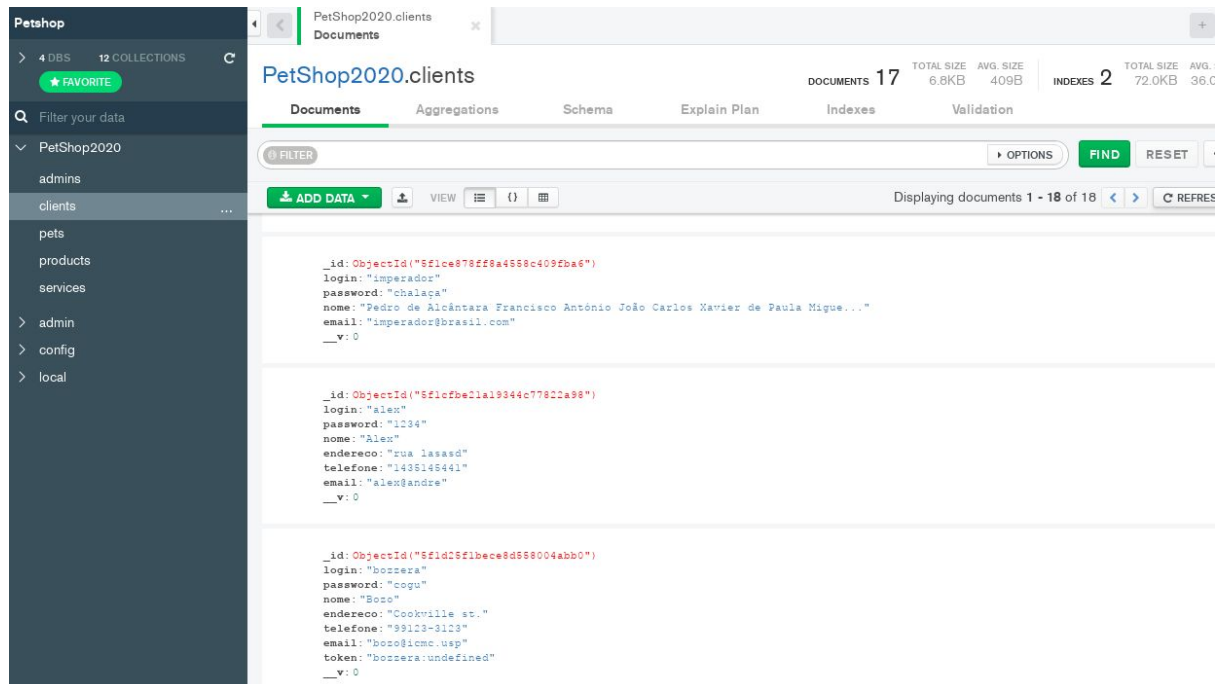
clientSchema.methods.validatePassword = function(password) {
  console.log("validating password");
  const hash = crypto.pbkdf2Sync(password, this.salt, 10000, 512, 'sha512').toString('hex');
  return this.hash === hash;
};
```

## Database:

For development reasons, we decided in favor of using a cloud database service (mongodb atlas). The database stores most of the data required by the functionalities of the project, including most of the attributes of the “objects” described in section 1 of this document (product, customer, etc).

Moongoose was the choice for modeling and facilitating the database interactions.

## MongoDB Compass UI:



## Handling Requests:

In addition, throughout the project we utilised the Axios javascript library, which allowed us to use the same code both to make HTTP requests in node and Ajax requisitions in the browser context. Axios was an integral part of this implementation, responsible for most of the communication with the mongodb atlas cloud.

Bit of an example request for a client login:

```
axios.post('/api/clients/login', {  
  login: self.login_c,  
  password: self.pass_c  
})  
  .then( (response) => {
```

### User Interfaces:

In the making of the front-end section of the project, we made use of the bootstrap toolkit both to save time and to prevent the reinvention of the wheel, allowing the team to hastily build responsive and *aesthetically* pleasing webpages, with carousels, footers and headers, intuitive buttons and convenient design. The images were curated from the most respectable stock photo sources, allowing us to build an entire marketing identity for the Pet Shop application, aimed at ensuring client loyalty to the business.

Within the project, Vue.js is utilised in the context of building functional user interfaces. The progressive framework was handpicked from a myriad of other available options because of its simplicity, elegance and ease of use. Also, the team was already semi-familiarized with the toolkit due to past assignments in the Introduction to Web Development course.

Eventually we found BootstrapVue to be useful in unprecedented ways, seamlessly integrating Vue.js - the progressive javascript framework for building user interfaces - with the most popular front-end CSS library: Bootstrap.

After months of development, hopefully, it will have been worth the wait. We can be reached at:

[bruno.baldissera@usp.br](mailto:bruno.baldissera@usp.br)

[bruno.gazoni@usp.br](mailto:bruno.gazoni@usp.br)

[rafael.ceneme@usp.br](mailto:rafael.ceneme@usp.br)

### **3. Test Plan**

Most of the testing was done manually. Every test was done on a need-to-know basis, that is, most of them were designed to simulate user behaviour, whether that user's an administrator or a client. For instance, when testing if a new user was properly added to the database, we simply used the registration form as any human being would, in order to create a new user account. Afterwards, using mongodb-compass, we could see that the new user was indeed saved on the cloud DB service.

## 4. Test Results

This image was taken during the registration process of user “norb”:

Cadastrar

Norbert Brandtner

Nome

norb

Login

norb@music.com

Email

.....

Senha

Sweden

Endereço

27651983

Telefone

Foto

Browse...

No file selected.

Cadastrar

After clicking the button “Cadastrar”, at the bottom of the registration form, an entry for user “norb” was added to the database, with its own unique, identifying ID:

```
_id: ObjectId("5f1e0642f5b9aa35f581716e")
login: "norb"
nome: "Norbert Brandtner"
endereço: "Sweden"
telefone: "27651983"
email: "norb@music.com"
salt: "3e6c3de36520554557f564027351da76"
hash: "55a9b14d8256be338e788adf53583c7b2ad8c55cdb8ae345f837f0bef730bd110757a8..."
```

Successful appointment for Mr. Penguin (placeholder):

  [Serviços](#) [Produtos](#) [Calendário](#) [Conta](#)

Confira seus horários!

Maio						
D	S	T	Q	Q	S	S
					1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30
31						

Junho						
D	S	T	Q	Q	S	S
	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30				



Horários do dia 1:

8:00

8:30

9:00

9:30

10:00

 Banho do Sr. Penguin



Basic shopping functionality was ensured through practical testing. By clicking on “adicionar ao carrinho” the user can stack products on a shopping cart:



which is then displayed at checkout (the fields contain placeholder text). Note that only the email field is actually used to query the database and add the item id to the client cart object:

## Dados para Cobrança

Nome

Miquéias Pinheiro

Email

miqueias@pinheiro.com

Endereço

Jazz St.

Estado

AM

CEP

10001

## Pagamento

Número do cartão

1111-2222-3333-4444

Expiração

Setembro

Ano de expiração

2018

CVV

352

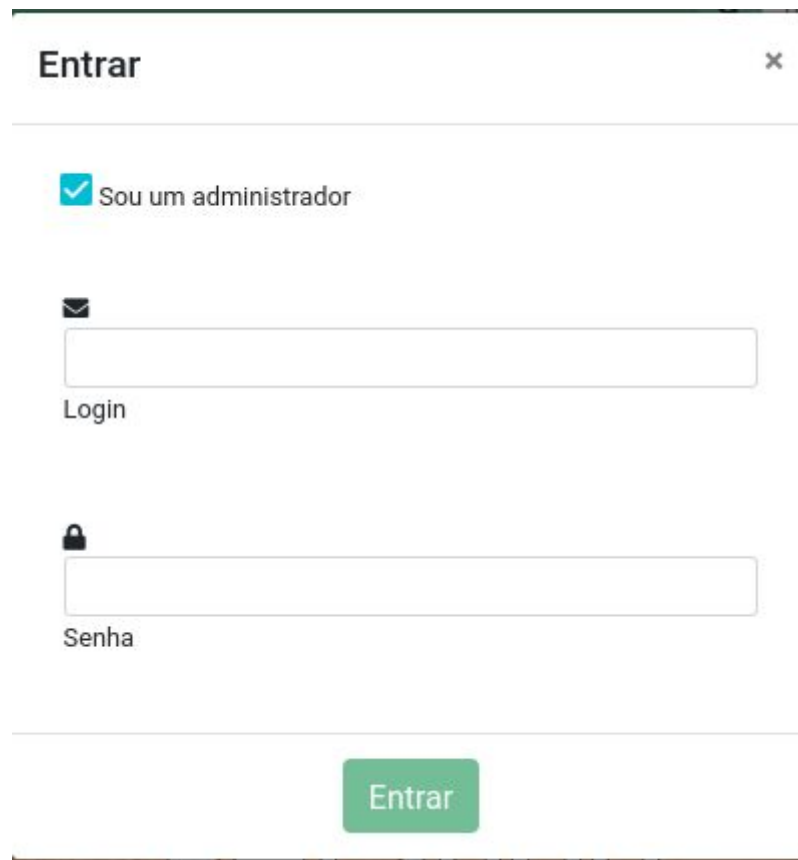
Adicionar

Cart  1

Bola \$5

Total \$5


The login screen has an admin checkbox which allows admins the sign into the privileged areas of the website, if the admin credentials were previously registered in the database via the “register admin” functionality:




The image shows a login form titled "Entrar" with a close button (X) in the top right corner. Below the title, there is a checkbox labeled "Sou um administrador" which is checked. Underneath this, there are two input fields: the first is labeled "Login" and has an envelope icon to its left; the second is labeled "Senha" and has a lock icon to its left. At the bottom of the form, there is a green button labeled "Entrar".

**Entrar** X

☒ Sou um administrador



Login




Senha

**Entrar**

Successful login:


**Entrar** ✕

☒ Sou um administrador



sindico

Login




.....

Senha

**Entrar**

**Login bem-sucedido! Bem-vindo(a) Tim Maia !**

Client actions page:


 **PetShop**

Buscar no site

ServiçosProdutosCalendário


Do que você precisa?


Conheça nossos serviços.



Agendar Horário


Agende um horário para atendermos o seu pet







Comprar Produtos

Confira nosso estoque de produtos








Pet profile:

Rex

Iti malia

Meu dono



ID

0072

Raça

Vira-Lata

Idade

4 Anos

Serviços Agendados


Um

Monte

De

Serviços

Admin actions page:


 **PetShop**

Buscar no site

ServiçosProdutosCalendário


## Bem-vindo de volta, administrador!


Confira ou edite.



### Registrar Administrador


Registre aqui alguém para se juntar à equipe







### Clientes

Acesse a nossa lista de clientes







Register Product (admin only):

## Registrar um novo produto



Nome

Descrição

Preço

Quantidade em estoque

## 5. Build Procedures

Some node.js modules must be installed to ensure proper functionality. To do so, you must follow the developer's instructions to install node, and then run the following commands on the project's root folder (/PetShop/) to install all the necessary packages.

```
npm install express --save
npm install mongoose --save
npm install body-parser --save
npm install axios
npm install vue bootstrap-vue bootstrap
npm install jsonwebtoken
npm install permit --save
npm install express --save
npm install crypto
```

Then, while on the same root folder mentioned above, use the following:

```
node index.js
```

You should now be able to open the project in a browser of your choice, by navigating to

```
http://localhost:8080
```

## 6. Problems

Because of the natural complexity presented by the PetShop application, many proposed functions were left unpolished and or disfunctional. The main challenge in the backend was balancing, establishing coherent communication and properly orchestrating the many external libraries and frameworks required to simplify javascript programming and data bank access.

It is hence advised to have familiarity with the frameworks listed on the buildup procedures so code comprehension might be attained.

Another great issue was the necessity that the group had to review and be able to understand every single component of the application across front and back ends for the development progress's sake, which slowed down itself. This made the teamwork a slow and confusing one.



 **147** commits

For instance, as of the time of writing this document, most of these commits consisted in individual works developed by each member which had to be thoroughly discussed in order to promote the understanding and usability of certain elements of the app to all other colleagues.

The unfamiliarity with the greater part of the tools, frameworks and even work flow also hindered the development and its efficiency.

A constant back-and-forth motion within deciding to use a certain framework or another to 'simplify' development was also present. For instance, the authentication method was rethought several times in different frameworks and abandoned only to be replaced by a more manual approach. Evidently this is also a consequence of low familiarity with web development as a whole and little knowledge of the area's concepts as well.

A large time consumption on the API understanding, projecting and building stages of development resulted in a lack of time for developing front-end and actual interactive user functionalities and features.

**Enjoy.**