Agrobox Documentation

Back-End Routes

In Production

- Users

POST:
{{baseUrl}}/auth/signup
{{baseUrl}}/auth/login

PUT:
{{baseUrl}}/auth/personalInfo
{{baseUrl}}/auth/address
{{baseUrl}}/auth/password
{{baseUrl}}/auth/promote
{{baseUrl}}/auth/demote/:id

GET:
{{baseUrl}}/auth/adminList
{{baseUrl}}/auth/user

- POST: {{baseUrl}}/auth/signup
  - Authentication Required: FALSE
  - Request body:
    ```
    {
      "name":"Full Name",
      "email":"some@mail.com",
      "password":"secretPassword",
      "phone":"787-123-4567"
    }
    ```
  - Response:
    - Body:
      - authenticationToken: [String]
    - StatusCode:
      - on success: 201
      - On fail: email already exists 409
      - On fail: server error 500

- POST: {{baseUrl}}/auth/login
    - Authentication Required: FALSE
    - Request Body:
      ```
      {
        "email":"someNew@mail.com",
        "password":"secretPassword"
      }
      ```
    - Response:
        - Body:
            - authenticationToken: [String]
        - StatusCode:
            - on success: 200
            - On fail: invalid credentials 403
            - On fail: server error 500

- PUT: {{baseUrl}}/auth/personalInfo
    - Authentication Required: TRUE
    - Role: user
    - Request Body:
      ```
      {
        "name":"New Name",
        "email":"pepito@somemail.cam",
        "phone":"7871230987"
      }
      ```
    - Response:
        - StatusCode:
            - on success: 200
            - On fail: Not Found 404
            - On fail: server error 500

    - PUT: {{baseUrl}}/auth/address
        - Authentication Required: TRUE
        - Role: user
        - Request Body:
          ```
          {
            "address":"Boulervard St. 96",
            "city":"Dorado",
            "zipcode":"00646",
            "state":"Puerto Rico"
          }
          ```

        - Response:
            - StatusCode:
                - on success: 200
                - On fail: Not Found 404
                - On fail: server error 500

- PUT: {{baseUrl}}/auth/password
    - Authentication Required: TRUE
    - Role: user
    - Request Body:

```
{
  "old_Password":"secretPassword",
  "new_Password":"newSecretPassword"
}
```

    - Response:
        - StatusCode:
            - on success: 200
            - On fail: Not Found 404
            - On fail: server error 500

- PUT: {{baseUrl}}/auth/promote
    - Authentication Required: TRUE
    - Role: admin
    - Request Body:

```
{
  "email":"pepito@somemail.cam"
}
```

    - Response:
        - StatusCode:
            - on success: 200
            - On fail: Not Found 404
            - On fail: server error 500

- PUT: {{baseUrl}}/auth/demote/:id
    - Authentication Required: TRUE
    - Role: owner
    - Request Params(id in url)
        - Id: user_id
    - Response:
        - StatusCode:
            - on success: 200
            - On fail: Not Found 404
            - On fail: server error 500

- GET: {{baseUrl}}/auth/adminList
    - Authentication Required: TRUE
    - Role: admin
    - Response:
        - Body:
```
[
   {
      "_id": "adminId",
      "name": "Full Name",
      "email": "someAdmin@mail.com",
      "phone": "787-123-4567"
   },
   ...
]
```
        - StatusCode:
            - on success: 200
            - On fail: server error 500

- GET: {{baseUrl}}/auth/adminList
    - Authentication Required: TRUE
    - Role: user
    - Response:
        - Body:
```
{
  "name": "New Name",
  "email": "pepito@somemail.cam",
  "phone": "7871230987",
  "address": "Boulervard St. 96",
  "city": "Dorado",
  "state": "Puerto Rico",
  "zipcode": "00646"
 }
```
        - StatusCode:
            - on success: 200
            - On fail: server error 500

- Products

POST:
{{baseUrl}}/product/

PUT:
{{baseUrl}}/product/:id

GET:
{{baseUrl}}/product/:id
{{baseUrl}}/product/

DELETE:
{{baseUrl}}/product/:id

  - POST: {{baseUrl}}/product/
    - Authentication Required: TRUE
    - Role: Admin
    - Request Body:

```
{
  "product_name": "Brocolii",
  "product_category": "Vegetales",
  "product_quantity_stock": 120,
  "product_units": "lbs",
  "product_price": 2.49
}
```

    - Response:
      - StatusCode:
        - on success: 201
        - On fail: conflict (product_name taken) 409
        - On fail: server error 500

- PUT: {{baseUrl}}/api/product/:id
  - Authentication Required: TRUE
  - Role: Admin
  - Request Params(id in url)
    - Id: productId
  - Request Body:

    ```
    {
      "product_name": "Brocoli",
      "product_category": "Vegetales",
      "product_quantity_stock": 120,
      "product_units": "lbs",
      "product_price": 2.49
    }
    ```

  - Response:
    - StatusCode:
      - on success: 201
      - On fail: product not found 404
      - On fail: conflict (product_name taken) 409
      - On fail: server error 500

- GET: {{baseUrl}}/product/:id
  - Authentication Required: FALSE
  - Request Params(id in url)
    - Id: productId
  - Response:
    - Body:

      ```
      {
        "_id": "6068ebc29d04e44dc7373c51",
        "product_name": "batata",
        "product_category": "viandas",
        "product_quantity_stock": "10",
        "product_units": "lbs",
        "product_price": "3.99",
        "product_image": "placeholder"
      }
      ```

    - StatusCode:
      - on success: 200
      - On fail: Product not found 404
      - On fail: server error 500

- GET: {{baseUrl}}/product/
    - Authentication Required: FALSE
    - Response:
        - Body:

```
{
    ‘Vegetales’: [
            {
                "_id": "607af298dd98c7002b2f7823",
               "product_name": "Tomate",
               "product_category": "Vegetales",
               "product_quantity_stock": 100,
               "product_units": "lbs",
               "product_price": 0.99
            }, …
      ], …
}
```

        - StatusCode:
            - on success: 200
            - On fail: server error 500
- DELETE: {{baseUrl}}/product/:id
    - Authentication Required: TRUE
    - Role: Owner
    - Request Params(id in url)
        - Id: productId
    - Response:
        - StatusCode:
            - on success: 200
            - On fail: Product not found 404
            - On fail: server error 500

- Boxes

POST:
{{baseUrl}}/box/

PUT:
{{baseUrl}}/box/enable/:id
{{baseUrl}}/box/disable/:id
{{baseUrl}}/box/addProduct
{{baseUrl}}/box/:id

GET:
{{baseUrl}}/box/products/:id
{{baseUrl}}/box/available
{{baseUrl}}/box/:id
{{baseUrl}}/box/

DELETE:
{{baseUrl}}/box/:id

  - POST: {{baseUrl}}/box/
    - Authentication Required: TRUE
    - Role: Admin
    - Request Body:

```
{
  "box_Name":"New Box",
  "box_Price":"99.99",
  "box_Content": [
    {
        "product_Id": "60693bf516b3995671468672",
        "product_Amount": 5
    },
    {
        "productId": "6068ebc29d04e44dc7373c51",
        "productAmount": 8
    },
    {
        "productId": "6068c3264fd9c0435c0c1f7c",
        "productAmount": 4
    }
  ]
}
```

        - Response:
            - StatusCode:
                - on success: 201
                - On fail: conflict (box_name taken) 409
                - On fail: server error 500

- PUT: {{baseUrl}}/box/enable/:id
    - Authentication Required: TRUE
    - Role: Admin
    - Request Params(id in url)
        - Id: productId
    - Response:
        - StatusCode:
            - on success: 200
            - On fail: Box not found 404
            - On fail: server error 500

- PUT: {{baseUrl}}/box/disable/:id
    - Authentication Required: TRUE
    - Role: Admin
    - Request Params(id in url)
        - Id: productId
    - Response:
        - StatusCode:
            - on success: 200
            - On fail: Box not found 404
            - On fail: server error 500

- PUT: {{baseUrl}}/api/box/:id
    - Authentication Required: TRUE
    - Role: Admin
    - Request Params(id in url)
        - Id: productId
    - Request Body:

```
{
  "box_Name":"New Box",
  "box_Price":"99.99",
  "box_Content": [
     {
          "product_Id": "60693bf516b3995671468672",
          "product_Amount": 5
     },
     {
          "productId": "6068ebc29d04e44dc7373c51",
          "productAmount": 8
     },
     {
          "productId": "6068c3264fd9c0435c0c1f7c",
          "productAmount": 4
     },
     ...
  ]
}
```

    - Response:
        - StatusCode:
            - on success: 200
            - On fail: Box not found 404
            - On fail: server error 500

- GET: {{baseUrl}}/box/box/:id
  - Authentication Required: FALSE
  - Request Params(id in url)
    - Id: productId
  - Response:
    - Body:
```
{
  "_id": "606a0e86e4ebdeef99bb5f1e",
  "box_name": "New Box",
  "box_price": "99.99",
  "box_image": "/img/url/path",
  "box_content": [
     "60693bf516b3995671468672",
     "6068ebc29d04e44dc7373c51",
     "6068c3264fd9c0435c0c1f7c"
  ]
}
```
    - StatusCode:
      - on success: 200
      - On fail: Box not found 404
      - On fail: server error 500

- GET: {{baseUrl}}/box/available
  - Authentication Required: FALSE
  - Response:
    - Body:
```
[
  {
     "_id": "606a0e86e4ebdeef99bb5f1e",
     "box_name": "New Box",
     "box_price": "99.99",
     "box_image": "/img/url/path"
  },
  {
     "_id": "606deead038da03b2af5c11e",
     "box_name": "La Verdadera Real Caja",
     "box_price": "99.99",
     "box_image": "/img/url/path"
  },
...
]
```

        - StatusCode:
            - on success: 200
            - On fail: server error 500

- GET: {{baseUrl}}/box/products/:id
    - Authentication Required: FALSE
    - Request Params(id in url)
        - Id: productId
    - Response:
        - Body:

```
{
  "potatoes": {
    "_id": "60693bf516b3995671468672",
    "product_name": "potatoes",
    "product_category": "viandas",
    "product_quantity_stock": "25",
    "product_units": "lbs",
    "product_price": "4.99",
    "product_image": "placeholder"
  },
  "batata": {
    "_id": "6068ebc29d04e44dc7373c51",
    "product_name": "batata",
    "product_category": "viandas",
    "product_quantity_stock": "10",
    "product_units": "lbs",
    "product_price": "3.99",
    "product_image": "placeholder"
  },
  "lemons": {
    "_id": "6068c3264fd9c0435c0c1f7c",
    "product_name": "lemons",
    "product_category": "fruits",
    "product_quantity_stock": "30",
    "product_units": "lbs",
    "product_price": "4.99",
    "product_image": "placeholder"
  }
}
```

        - StatusCode:
            - on success: 200

            - On fail: Box not found 404
            - On fail: server error 500

- GET: {{baseUrl}}/box/
    - Authentication Required: TRUE
    - Role: Admin
    - Response:
        - Body:

```
[
  {
    "_id": "606a0e86e4ebdeef99bb5f1e",
    "box_name": "New Box",
    "box_price": "99.99",
    "box_image": "/img/url/path",
    “available”: true
  },
  {
    "_id": "606deead038da03b2af5c11e",
    "box_name": "La Verdadera Real Caja",
    "box_price": "99.99",
    "box_image": "/img/url/path",
    “available”: true
  }
]
```

        - StatusCode:
            - on success: 200
            - On fail: server error 500

- DELETE: {{baseUrl}}/product/:id
    - Authentication Required: TRUE
    - Role: Owner
    - Request Params(id in url)
        - Id: boxId
    - Response:
        - StatusCode:
            - on success: 200
            - On fail: Product not found 404
            - On fail: server error 500

- Order

POST:
{{baseUrl}}/order/

PUT:
{{baseUrl}}/order/:id
{{baseUrl}}/order/manage/:id

GET:
{{baseUrl}}/order/:order_id
{{baseUrl}}/order/user/:user_id
{{baseUrl}}/order/city/:city

- POST: {{baseUrl}}/order
    - Description: Submit order. Must contain both the order information and the corresponding contents
    - Authentication Required: TRUE

- Request Body:

```
{
  "order": {
      "order_name": "order name here",
      "order_number": "787-123-4567",
      "delivery_address": "address",
      "delivery_city": "city",
      "delivery_zipcode": "deliveryZipcode",
      "total_Price": 99.99,
      "payment_Method": "cash"
  },
  "orderContent": [
          {
              "box_Id": "id",
              "boxName": "Box 1",
              "boxPrice": 37.99,
              "boxQuantity": 1,
              "boxContent": [
                  {
                      "productId": "prod 1 id",
                      "product_name": "Product 1",
                      "product_category": "Viandas",
                      "product_units": "lbs",
                      "product_price": "0.90",
                      "product_image": "placeholder",
                      "amount": 20
                  }, ...
              ]
          },
          {
              "box_Id": "id2",
              "boxName": "Box 2",
              "boxPrice": 37.99,
              "boxQuantity": 3,
              "boxContent": [
                  {
                      "productId": "prod 3 id",
                      "product_name": "Prod 3",
                      "product_category": "Viandas",
                      "product_units": "lbs",
                      "product_price": "0.80",
                      "product_image": "placeholder",
                      "amount": 20
                  }, ...
              ]
          }, ...
      ]
}
```

Selection in blue will become an order object stored in the object db, yellow will become an orderObject associated with the corresponding order

- PUT: {{baseUrl}}/order/:id

    - Description: update order. Mostly used to update status but can be used to optionally update any other order field.
    - Request Body:

```
{
  “order_status”: “status to change”,
  “Secondary_fields”: “value to change”
}
```

- GET: {{baseUrl}}/order/user/:id
    - Description: Get all orders from a user

- GET: {{baseUrl}}/order/:id
    - Description: Get order by order id

- GET:{{baseUrl}}/order/city/:city
    - Description: Get order by city (municipio); must receive city as a string.

- PUT:{{baseUrl}}/order/manage/:id
    - Description: Carries out inventory management for orders that have been confirmed/accepted for delivery. Will read id and auto manage stock.

- ## OrderContent

POST:
{{baseUrl}}/orderContent/

PUT:
{{baseUrl}}/orderContent/:id

GET:
{{baseUrl}}/orderContent/
{{baseUrl}}/orderContent/:id

- POST: {{baseUrl}}/orderContent/
    - Description: Inserts an orderContent object, used for testing. Look to the post method for orders to see how to properly insert an order’s contents.

- GET: {{baseUrl}}/orderContent/:id
    - Description: Get order by order id

- GET:{{baseUrl}}/orderContent/
    - Description: Get all order contents available within the system.

- PUT:{{baseUrl}}/orderContent/:id
    - Description: Testing method. All orders are assumed to be confirmed by the user and properly inserted into the system. Any error correction is left to the discretion of Ricardo.

Front-End

Specific Objectives

Easy - Access Authentication Process
Simplify the ordering process from the customer-side
Automate Order Processing from the admin-side
Provide efficient inventory management
Provide efficient order management
Aid in delivery management

Architecture (Diagram)

Natural Language Description

Technical Approach (Implementation)

All the screens and components presented in this document are divided into their own folder that contains the screen.js file and the screenStyleSheet.js file.

- **Screens**
    - Admin
        - **EditBoxScreen**

          Purpose:

          The main purpose of this screen is for the administrator to manage the quantity and product and to make boxes available/not available for the customers to purchase. This is also the screen used for the administrator to create a new box.

          Custom components in this screen:

          BackArrow - When pressed it redirects back to the ManageBoxScreen.

          FormInput - Is used to render the box's name and price, the admin can also make any changes to box's information, if desired.

Button - Its purpose is to enable or disable a specific box, delete a box and to save the changes made to the box and to send them to the BoxService and/or ProductService.

Loader - Used when navigating between screens (when network is slow), in this case when navigating from the ManageInventoryScreen into the EditBoxScreen.

DropDown - In this screen the dropdown is used to divide the information to edit. Four dropdown options are available, the 'Especias', 'Frutas', 'Vegetales' and ' Viandas', which are the four categories of food our client sells.

InteractiveProductCard - Through this component, the administrator is able to change the quantity of a specific product inside the box selected. It's shown when the corresponding category dropdown is pressed.

MediaUploader - Used to upload an image from the administrators phone and and set it as the box image.

Service(s) in this screen:

BoxService - Used to remove, add or update a box in the database.

ProductService - Used when the changes are made to any product in the specific box.

- **EditProductScreen**

  Purpose:

  The purpose of this screen is for the administrator to manage the quantity and units of the products available for the boxes a customer can purchase. This is also the screen used for the administrator to create a new product.

  Custom components in this screen:

  Loader - Used when navigating between screens (when network is slow), in this case when navigating from the ManageProductScreen into the EditProductScreen.

BackArrow - When pressed it redirects back to the ManageProductScreen.

FormInput - Is used to render the product’s name, category, quantity units and price (per unit).

Button - Used to either save the changes made or delete the product and send that information to the database.

MediaUploader - Used to upload an image from the administrators phone and and set it as the product image.

Service(s) in this screen:

ProductService - Used to send the changed products information to the database.

- **ManageAdministratorScreen**

  Purpose:

  Here the administrador can assign the admin role to another user using their email. Proving this role to the user of the owner's choice, can help them manage the rest of the application, in terms of the orders and establishing the deliveries, and updating the inventory.

  Custom components in this screen:

  BackArrow - When pressed it redirects back to the MenuScreen.

  Loader - Used when navigating between screens (when network is slow), in this case when navigating from the MenuScreen into the ManageAdministratorScreen.

  Service(s) in this screen:

  AdminService - used to fetch the users that have admin privileges and display them to the administrator. Additionally, when an admin is removed this service is used to update that information in the backend.

- **ManageBoxesScreen**

  Purpose:

This screen is meant for the admin to see and manage the inventory of the boxes, and to create a new box.

Custom components in this screen:

BackArrow - When pressed it redirects back to the ManageInventoryScreen.

Button - In this case, the button is used to redirect to the EditBoxScreen with the purpose of creating a new box.

BoxCard - Used to display the boxes name, price and image. When pressed goes to the EditBoxScreen to edit the content of the specific box.

Loader - Used when navigating between screens (when network is slow), in this case when navigating from the ManageInventoryScreen into the ManageBoxesScreen.

Service(s) in this screen:

BoxService - Used to save the box's information that is going to the backend.

- **ManageInventoryScreen**

Purpose:

This screen helps the administrator to select what they want to manage, whether it is the boxes or the products. It displays two buttons with the options that take them to the corresponding management screen (ManageBoxesScreen or ManageProductScreen). It's purpose is to facilitate the process inventory management of the company.

Custom components in this screen:

Button - Its purpose is to go to the next screen, when pressed, to continue with the management process, in this case two buttons are rendered with the option to go to the ManageBoxesScreen or the ManageProductScreen.

BackArrow - When pressed it redirects back to the MenuScreen.

Logo - The logo of the Agrobox company.

Service(s) in this screen:

No services are needed on this screen.

- **ManageOrdersScreen**

  Purpose:

  The purpose is to help the administrator to manage the placed orders by all the customers and help them organize them for preparation and delivery. This screen contains all the necessary information about the orders such as the customer's name, order number, date, delivery address (when pressed it opens google maps with the customers address pin), and the order status ('En Camino', 'Pendiente', 'Completada'), which can changed when needed.

  Custom components in this screen:

  OrderCard - This component holds the placed orders information, such as the customer's name, phone number, delivery address, payment information and the order id and status. When one of the options mentioned above are pressed, a different action occurs. When the phone number of the customer is pressed it gives you the option to call them. When the address is pressed, it opens the Google maps app with the customer's delivery location. Also, when the payment information is pressed, it gives the administrator the opportunity to copy the transaction id, additionally, when the status of the order is pressed, it allows the admin to change the status to 'Pendiente', 'En Camino' or 'Completada' and lastly when this component is pressed, the order details are shown.

  DropDown - In this screen the dropdown is used to divide the orders by their status. Three dropdown options are available, the 'Pendiente', 'En Camino' and 'Completada'.

  BackArrow - When pressed it redirects back to the MenuScreen.

  Loader - Used when navigating between screens (when network is slow), in this case when navigating from the MenuScreen into the ManageOrdersScreen.

  Service(s) in this screen:

OrderService - Used to fetch all the information regarding a specific order and to update the status in the database when the administrator makes the changes.

- **ManageProductScreen**

  Purpose:

  This screen is meant for the admin to manage the inventory of the product, create new products that can be added to boxes or used to 'Build your own' box.

  Custom components in this screen:

  DropDown - In this screen the dropdown is used to divide the information to edit. Four dropdown options are available, the 'Especias', 'Frutas', 'Vegetales' and ' Viandas', which are the four categories of food our client sells.

  ProductCard - Through this component, the administrator is able to see the displayed product name, price and image. When pressed goes to the EditProductScreen to edit the product.

  Button - In this case, the button is used to redirect to the EditProductScreen with the purpose of creating a new product.

  BackArrow - When pressed it redirects back to the ManageInventoryScreen.

  Loader - Used when navigating between screens (when network is slow), in this case when navigating from the ManageInventoryScreen into the ManageProductScreen.

  Service(s) in this screen:

  ProductService - Used to save the product's information that is going to the backend.

- <u>Customer</u>
  - **CartScreen**

    Purpose:

    This screen is considered part of the Payment process, and is used to store the information about the boxes a customer wants to buy, it stores the name of the box, the quantity, the accumulated price and total of the purchase. This screen is meant to give the customers a summary of their order.

Custom components in this screen:

Logo - The logo of the Agrobox company.

BoxCard - Used to render the information(image, box name, box quantity, box price and box accumulated price) of the box selected.

QuantitySpecifier - It allows the customer to change the quantity of a specific box from the same screen.

Button - Its purpose is to go to the next screen, when pressed, to continue with the payment process, in this case the next screen is the CheckoutScreen.

Service(s) in this screen:

UserService - Used to verify if the user is authenticated or not(if they are logged into their account).

CartService - This service is used to store the information of the cart screen at a specific time, meaning it will not be sent to the backend nor be removed until the customer finishes the payment process or it removes the boxes from the cart screen.

- **CheckoutScreen**

Purpose :

This screen is the second screen for the payment process. It renders the information of the customer like their name, phone number and the address where the order will be delivered to. Its purpose is that the customer can verify their information before placing the order.

Custom components it contains:

BackArrow - When pressed it redirects back to the CartScreen.

FormInput - Is used to render the user information (name, phone number and address) and for the customer to make any changes to their information, if desired.

Loader - Used when navigating between screens (when network is slow), in this case when navigating from the CartScreen into the CheckoutScreen.

Localizer - Used when the customer's address information is not recorded in the database, the customer can use the pin component to use their current address as the delivery address.

Button - Its purpose is to go to the next screen and to submit the customers information handler, when pressed.

Service(s) in this screen:

UserService - Used to fetch the users information (name, phone number and address) rendered in the FormInput component.

- **ContactUsScreen**

  Purpose:

  The ContactUsScreen renders the necessary information about where to contact directly the Agrobox administrators, through their social media and whatsapp.

  Custom components it contains:

  Logo - The logo of the Agrobox company.

  BackArrow - When pressed it redirects back to the MenuScreen.

  SocialIcon(React native element component) - These are the buttons used to connect all their social media to the app.

  Service(s) in this screen:

  No services are needed on this screen.

- **EditAccountScreen**

  Purpose:

  This screen's purpose is to ease the process of updating/changing the user information like their name, email, phone number, address and password (only if the user knows the current password to their account).

Custom components it contains:

FormInput - Used to render the user information (name, phone number and address) and for the customer to make any changes to their information, if desired.

DropDown - In this screen the dropdown is used to divide the information to edit. There are three dropdown options, the Customer information (name, email, phone number), Address information (street, city, state and zipcode) and Password (current, new password), where the user can change their password as long as they remember their current password.

Button - Its purpose is to update the customer information and send it to the UserService.

Localizer - Used when the customer's address information is not recorded in the database, the customer can use the pin component to use their current address.

Loader - Used when navigating between screens (when network is slow), in this case when in the TabBar where the person icon is pressed.

Service(s) in this screen:

UserService - Used to fetch the users information (name, email, phone number and address) rendered in the FormInput component.

- **EditCartScreen**

Purpose:

This screen is part of the payment process and can only be accessed if a box is added to the CartScreen and the BoxCard component is pressed. In this screen the customer can change a product quantity that belongs to a specific box(the box pressed in the CartScreen) and save the new information. These actions can help the customer to validate and make changes to the boxes they wish to purchase.

Custom components it contains:

BackArrow - When pressed it redirects back to the CartScreen.

Button - In this screen the button is used to save the changes made in this screen by sending the information to the CartService.

InteractiveProductCard - Through this component, the customer is able to change the quantity of a specific product inside the box selected from the cart.

Service(s) in this screen:

CartService - Used to fetch the box content(products and quantity) of a specific box.

- **HomeScreen**

  Purpose:

  The HomeScreen is the initial screen shown when the application is opened for the first time. It’s purpose is to show the customer all the boxes available for them to purchase. An image, the name and price of the box is what is rendered to the customer in order for them to navigate to the desired box.

  Custom components it contains:

  Loader - Used when navigating between screens (when network is slow), in this case when in the TabBar, where the home icon is pressed.

  BoxCard - Is the component that renders the box image, name and price for all the boxes available.

  Logo - The logo of the Agrobox company.

  Service(s) in this screen:

  BoxService - Used to fetch the box name and price of each box available.

- **LoginScreen**

  Purpose:

  This screen is meant for the user to login to their account, thus they can be able to continue with their payment process.

  Custom components it contains:

Loader - Used when navigating between screens (when network is slow), in this case when the user information is being validated.

FormInput - Meant to receive the email address and password of a specific user so they can login to their account.

Logo - The logo of the Agrobox company.

Button - In this screen the button is used to send the users credentials (email and password) to the backend to verify if they have an account and log them in.

Service(s) in this screen:

UserService - Used to fetch the user information and verify if it is in the database.

- **MenuScreen**

  Purpose:

  Is important to note that the MenuScreen renders different options for customers and admins. For the customers, less options are available (‘Comunícate con Nosotros’ and ‘Cerrar Sesión’), and for the admins most of the options that define the admin role are accessed through this screen, such like Inventory management, order management, atc.

  Custom components it contains:

  Logo - The logo of the Agrobox company.

  Button - Used to show all the options a user (whether an admin or customer) has, and when pressed, depending on the option, it performs an action. For example, when the ‘Cerrar Sesión’ option is pressed it performs the action that is to log the user out of their account.

  Service(s) in this screen:

  UserService - In this case the UserService is used to verify the role of each user, since the administrator role has different options than the customers.

- **OrderConfirmationScreen**

Purpose:

This screen is meant to show a message to the customer, letting them know that their order has been placed and confirmed. This screen will only be rendered when the payment and ordering processes culminate.

Custom components it contains:

Logo - The logo of the Agrobox company.

Button - Used to navigate to the ViewOrderScreen when pressed.

Service(s) in this screen:

No services are needed on this screen.

- **PaymentScreen**

Purpose:

The PaymentScreen can only be accessed when there are boxes in the cart, it shows the payment methods available (ATH movil, Paypal, Cash), which eases the payment process for the customers since all the payment methods are in the same place.

Custom components it contains:

BackArrow - When pressed it redirects back to the CheckoutScreen.

PopUp - Used for when the Paypal button is pressed this popup view takes the customer to Paypal so they can continue with the payment process and selecting the card they desire to use and to finalize and place the payment.

Loader - Used when navigating between screens (when network is slow), in this case when the order submission is being processed.

Logo - The logo of the Agrobox company.

Service(s) in this screen:

CartService - Used to refresh the cart when the order submission is being processed, in other words, is emptying the cart.

OrderService - Used to verify if the order was successfully submitted, therefore it can go on to emptying the cart and confirming the order.

- **RegisterScreen**

  Purpose:

  The main purpose of the RegisterScreen is that an user can be able to create an account in the application. This helps the administrator to manage orders, when they come in, and the customer who placed it.

  Custom components it contains:

  FormInput - Meant to receive the user information such as, Full name, email address, password and phone number of a specific user so they can register a new account.

  Logo - The logo of the Agrobox company.

  Button - In this screen the button is used to send the users credentials to the backend to add their account and log them in.

  Service(s) in this screen:

  UserService - This service is used to send the credentials to the backend and so they can be added to the database.

- **ViewBoxScreen**

  Purpose:

  This screen renders all the details each box contains, such as the name of the products and their quantities, and the price of the box. Additionally, the customers are able to change the quantity of the products they wish to have in their box. The purpose is to facilitate the customer’s shopping experience.

  Custom components it contains:

  Loader - Used when navigating between screens (when network is slow), in this case when navigating from the HomeScreen into the ViewBoxScreen.

  CachedImage - Used to render the image of the box by caching the image saved in the database.

Button - In this screen the button is used to add the selected box to the cart, it first asks the customer if they wish to add the box to their cart, if the customer says yes it will send the information to the CartService.

InteractiveProductCard - Through this component, the customer is able to see the quantity of a specific product that box contains and to change the quantity of a specific product.

QuantitySpecifier - In this case, this component is meant for the user to select the quantity of a specific box they want to add to their cart.

BackArrow - When pressed it redirects back to the HomeScreen.

DropDown - Only rendered when the 'Build Your Box' BoxCard (in the HomeScreen) is pressed.

Service(s) in this screen:

BoxService - Used to fetch the box content of the target box.

ProductService - Used only when the 'Build your own' BoxCard is pressed in the HomeScreen, it fetches the products catalog so it can be displayed on the screen to the customer.

CartService - Used to send the box current information to the cart so it can be rendered in the CartScreen.

ImageService - used to fetch the image of a specific box in the database and render it in this screen.

- **ViewOrdersScreen**

  Purpose:

  The purpose of this screen is for the customer to see their placed orders history and status. This will help the customer to visualize which orders are confirmed, on the way and delivered.

  Custom components it contains:

OrderCard - This component contains the details of each order a customer has made, their name, the date, order number, the address to be delivered to and the status (‘En Camino’, ‘Pendiente’, ‘Completada’).

DropDown - the dropdown is used to divide the orders placed by a customer. There are three dropdown options, the ‘En Camino’(on the way) orders, ‘Pendiente’ (pending) orders and ‘Completada’ (completed) orders, where the user can see the details of each order.

Loader - Used when navigating between screens (when network is slow), in this case when in the TabBar, where the order icon is pressed.

Service(s) in this screen:

OrderService - Used to fetch the order details (name, the date, order number, the address to be delivered to and the status) so they can be displayed on this screen.

UserService - Used to verify if the user is authenticated so the correct information is displayed to the customer.

- **<u>Components</u>**
    - BackArrow
    - BoxCard
    - Button
    - CachedImage
    - CheckoutButtons/ATHM
    - DropDown
    - FormInput
    - InteractiveProductCard
    - Loader
    - Localizer
    - Logo
    - MediaUploader
    - OrderCard
    - Popup
    - ProductCard
    - QuantitySpecifier

- Tab
- TabBar

- **Services**
  - AdminService - Used to handle all administrator operations. Methods included are:
    - getAdmins(): Returns a list of objects containing information about all the administrators. Requires the JWT to be passed as ‘x-access-token’ in request header.

    - addAdmin(admin_email): Grants administrator privileges to the user specified in admin_email. Requires the JWT to be passed as ‘x-access-token’ in request header.

    - removeAdmin(admin_id): Revokes administrator privileges to the user specified with admin_id. Requires the admin_id to be passed as query param and the JWT to be passed as ‘x-access-token’ in request header.

  - BoxService - Used to handle all operations regarding the Boxes. Methods included are:
    - getAllBoxes(): Returns a list of objects containing information about all the boxes stored in the system.

    - getAllAvailableBoxes(): Returns a list of objects containing information about all the boxes marked as available in the system.

    - getBoxContent(box_id): Returns an array of objects containing information about the content of the box specified with box_id. Requires the box_id to be passed as query param and JWT to be passed as ‘x-access-token’ in request header. Token must have role property with the value of “admin”.

    - addNewBox(box): Adds the box passed as parameter to the system. Requires the JWT to be passed as ‘x-access-token’ in request header. Token must have role property with the value of “admin”.

    - updateBox(box): Updates the box passed as parameter to the system. Requires the box_id to be passed as query param and JWT to be passed as ‘x-access-token’ in the request header. Token must have role property with the value of “admin”.

        - disableBox(box_id): Disables the box specified with box_id from the system. Requires the box_id to be passed as query param & the JWT to be passed as ‘x-access-token’ in request header.

        - enableBox(box_id): Enables the box specified with box_id from the system. Requires the box_id to be passed as query param & the JWT to be passed as ‘x-access-token’ in request header.

        - removeBox(box_id): Removes the box specified with box_id from the system. Requires the box_id to be passed as query param & the JWT to be passed as ‘x-access-token’ in request header.

    - CartService - Used to store the users cart locally, in the devices file system. Uses AsyncStorage to accomplish this feature. Methods included are:
        - getCart(): Returns a list containing all the items the users have stored in their cart.

        - setCart(cart): Updates the cart stored in the users file system with the cart passed as parameter.

        - addToCart(item): Adds the item passed as parameter to the users cart.

        - getCartTotal(): Computes the total price of all the items stored in the cart.

        - updateCart(box): Updates the specific box passed as parameter in the cart list.

        - refreshCart(): Refreshes (Clears) the content in the cart. Used when order is submitted, and the cart needs to be refreshed for the next order.

    - GeoCodingService - Used to convert coordinates to natural language address, or vice versa. Methods included are:
        - convertToAddress(coordinates): converts the coordinates passed as parameter to natural language format. Returns a string containing the converted address.
        - convertToCoordinates(address): converts the address passed as parameter to coordinates. Returns an object with the properties latitude and longitude.

- ImageService - Used to fetch images from the BackEnd asynchronously. Methods included are:
  - getURL(image_name): returns a string containing the URL for the image specified by the parameter image_name.

- OrderService - Used to handle all operations regarding the Orders. Methods included are:
  - getUsersOrders: Returns all orders of a specific user. Requires the JWT to be passed as ‘x-access-token’ in the header, and the response is an object of lists where each property is an order_status (‘Pendiente’, ‘En Camino’, ‘Completada’) and their values are their respective list of orders.

  - getAllOrders: Returns all orders recorded in the system. Requires the JWT to be passed as ‘x-access-token’ in the header. The response is an object of lists where each property is an order_status (‘Pendiente’, ‘En Camino’, ‘Completada’) and their values are their respective list of orders.

  - updateOrderStatus(order_id, status): Used to update a specific order’s status with the value passed as parameter in the status argument. Requires the JWT to be passed as ‘x-access-token’ in the header, and the order_id to be passed as query param. Request body is an object with a single property named order_status.

  - getOrderContent(order_id): Returns a list containing the content of the orders specified with the order_id parameter. Requires the JWT to be passed as ‘x-access-token’ in the header, and the order_id to be passed as query param.

  - submitOrder(order_info): Used to submit a new order to the system. Request body is specified by order_info, and requires the JWT to be passed as ‘x-access-token’ in the header.

- ProductService - Used to handle all operations regarding the Products. Methods included are:
  - getProductCatalog(): Returns an object of lists where each object property is a product category and the list are all the products within that category.

  - addNewProduct(product): Publishes the product passed as parameter in the system. Requires the JWT to be passed as ‘x-access-token’ in the header.

        - updateProduct(product): Updates the product passed as parameter in the system. Requires the JWT to be passed as ‘x-access-token’ in the header.

        - removeProduct(product_id): Removes the product specified in the product_id parameter from the system. Requires the JWT to be passed as ‘x-access-token’ in the header.

    - Service - Super class to all other services. Stores the API’s URL.

    - UserService - Used to handle all operations regarding the Users and authentication. Methods included are:
        - loadWebToken():
        - setWebToken(token):
        - isAuthenticated():
        - isAdmin():
        - logout():
        - sendLogin(data):
        - sendRegistration(data):
        - updateUserInformation(userData):
        - updateAddress(addressData):
        - updatePassword(passwordData):
        - getUserData():
        - sendForgotPassword(email):

Performance results

For Customer

1. Images were initially stored as *base64* string, but they were putting too much overhead on requests so they were taking too long to fulfill (> 5 seconds). Therefore, we switched to storing the image binary files directly in the database and then streaming them to our mobile app. Although this proved to be more efficient (< 3 seconds), it was not our desired result as they were still taking too long to load; thus, we ended up creating an image cacher component to store their references locally, maximizing image load time (< 1 second) after the initial load, allowing us to reach our desired goal.

2. Initially the app’s aesthetics consisted mainly of the color white as background, and the font size was average, but this proved to be inefficient as our audience is mainly

composed of older people. Therefore, we ended up changing the background color to a darker color (Burgandee, #), and increasing the font size to help with readability. After conducting more informal usability tests with our direct relatives, the feedback we received was favorable, as the dark background highlighted the font across the application, essentially achieving our goal.

For Admin

1. Currently, order processing time takes the administrators around 10 minutes per customer on average (write order per customer, verify available inventory, confirm order to customer). Since the application automates this process entirely, this provides a < % > increase in the administrator's available time, allowing them to focus on more important business processes, such as expanding their company. At the moment, publishing their available products (on their social media accounts) take around 10 minutes on average; meanwhile, with the application it would take around 11-12 minutes on average, depending on the use case (Creating/Updating Products or Creating/Updating Boxes). This represents a 17% (worst case) slight increase in management time with the application, but the benefits of the order processing time are far larger, representing a net increase of < % > - 17% = < % > in available time. Thus, allowing our client to achieve a satisfactory percentage for overall time management.

    ○