



Nutri Selectif

Architecture.....	3
Objective.....	4
Succeeded features.....	4
Unsucceded features.....	5
Roadmap.....	6
Difficulties faced.....	6
Screenshots.....	7



Architecture

- Assets folder, which contains the static images which we use (no image placeholder, splash screen, logo)

- Rendu folder, which contains the document and the video presenting the application.

- StyleSheet.js

File containing every custom styles that were created in order to style our application

- CustomComponents.js

File containing every custom component, which was usually used several times, FoodItem for example, which is used to display the results of an API Query.

- UsefulFunction.js

A file containing some useful functions

It could have been deleted since it wasn't as useful as expected. It still contains the ignorePress function and the BMR calculation function.

- App.js

The first file the application reads.

It contains the Navigation interface and is responsible for the display of each screens

- HealthGoalScreen.js

Contains the form in which the user enters its information, which then calculates the recommended daily caloric intake.

It implements a few TextField, PickerField and DisplayField, which are custom components imported from CustomComponents.js

Thanks to AsyncStorage, every information and the recommended daily caloric intake are stored.

- FoodPageScreen.js

Contains a text search field which has an autocomplete implemented to it.

Once the user enters what he wants to search for, an API call is made. A maximum of 50 food items are then displayed.

When clicking a food item, a modal containing additional information on the food item is displayed.

It also provides a form where the user can enter the quantity, the meal of the day and the date at which the user wants to plan the current food item.

When adding it, AsyncStorage stores the food item object necessary information in order to display the food item on the meal planning screen.

The object stored has the following structure:

[something] meaning that something is variable.

```
{
  [dateString] : {
    breakfast:{
      [foodId]:{
        [foodItem]
        quantity:100
      }
    }
  }
  lunch:{...}
  snack:{...}
```



```

    diner:{...}
  }
  [dateString] : {
    breakfast:{...}
    lunch:{...}
    snack:{...}
    diner:{...}
  }
}

```

Since the quantity of data that is stored is not so important in our case, every food item is stored in a single item, which reduces the amount of calls to AsyncStorage necessary for data collection. With a bigger use of the application, each date could be a different item : when watching the 17/06/2023, only the food items for this date would be collected.

The foodId is unique, adding the same food item twice in the same meal of the same date only changes the quantity for this food item.

MealPlanningScreen.js

Displays the food items that the user added to their meal plan and allows the user to remove any displayed food item.

Clicking the date on top of the screen allows the user to select the date which he wants to display. The button is surrounded by arrows, which are used to display the previous or next day.

For each day, the screen contains 4 horizontal ScrollViews, in which every food item is added. Each ScrollView represents a meal of the day.

For each meal, the caloric intake of the selected food items is displayed, making it easier for the user to balance the calories he will consume through the day

The caloric intake is also calculated for the all day and displayed next to the recommended daily caloric intake calculated from the HealthGoal screen.

Objective

The main objective of the project was discovering how React Native works and how an Expo project is set up. While working on the project, we also had the opportunity to work with CSS and improve our skills with JS.

We also had a few secondary objectives which we set up for ourselves. These objectives will be presented in the next parts of the report, depending on whether or not we were able to successfully develop them.

Succeeded features

- The application should work properly



- Data persistence was implemented thanks to AsyncStorage. This may have not been the best way to go, but it worked and this didn't require us creating and hosting a database.
- A splash screen and a logo were designed (not only by hand), and they give a more "finished" aspect to the application
- Connection of the application with edamam food and groceries API.
- Connection to the auto-complete food and groceries API, which allowed us to help the user when searching for food. 5 options are given to the user when he starts typing.
- The application is in a state where in our opinion, it could be released. It is not perfect and could be improved in many ways of course, but we are still happy with the result.
- Implementation of an effect (useEffect()) in order to get the meals planned by the user every 5 seconds. A refresh button and a callback of the navigation "Meal planning" button could have been enough but some of the things we couldn't implement would have made these solutions obsolete.

Unsuccessful features

- We had the idea to create many custom components at first, and then use them in our application, this would have made the development faster. The problem being that in every screen, the design was different, thus requiring us to use customized components.
This can be seen with the HealthGoalScreen Component which only contains about 10 lines of custom components (TextField, PickerField and DisplayField). The other components contain a lot more lines of code.
- The application was tested both on Android and IOS at first. The absence of anyone with an iPhone in the last weeks of development made it harder to have a look at our application on IOS. Doing so would have required us to set up an emulator, which we didn't do. Even though it should work properly, we have no idea what the application looks like on IOS.
- The application design could have been better. At the moment, it somehow looks "old". It could definitely be improved by taking some time to make analysis of modern applications and use some of their practices, such as rounded up buttons, better onPress() animations...
- The meal planning screen is missing the "Add" button on each row (breakfast, lunch, snack, diner). If the application was to be developed further, this would be a priority. Yet, this does not prevent it from being used
- There is close to no error detection mécanismes on the application, which would have been necessary when using AsyncStorage, for example.
- Modification of the quantity of each food item from the meal planning screen.



- Ability to move each food item from a meal to another with a drop-down mechanism
- Display of the protein and fat intake with the selected food items for a day / each meal.
- Making a build (creating an apk) in order to use the application without Expo
- Parameter interface with a few customizing options (main color, dark/light mode)

Roadmap

- Environment setup
- Project creation
- Git repository setup
- Creation of the navigation interface
- Creation of the architecture (the different files)
- Implementation of the Health Goal Screen
- Choice of a nutrition API
- Implementation of the Food Page Screen
- Setup of the data persistence for Health Goals data
- Implementation of the Meal Planning Screen
- Setup of the data persistence for the planned meals
- Creation of the logo
- Creation of the Splash Screen
- Filming of the demonstration video
- Writing of the Readme document
- Writing of this documentation

Difficulties faced








The difficulties we faced were mostly due to the fact that this was the first time we used React Natives. At first, it wasn't easy but with some practice it became easier and easier. In the last days of development every features that needed to be developed didn't take long at all

A few examples:

- Using states and understanding everything that happens around them.
- Passing functions as properties in custom components, which is required to create two components with a different behavior, only using a single custom component.
- Making the application IOS friendly, which isn't that difficult but takes some time we didn't plan
- Communication between screens, which we didn't really implement since we used AsyncStorage when it was necessary. This probably makes the application slower.



Screenshots

16:32       

Age

23

Gender

Male ▼

Height (cm)

175

Weight (kg)

75

Activity Level




Moderate Exercise ▼

Health Goal

Weight Maintenance ▼

Your BMR is 1802.39

Recommended daily calories intake
:2793.7 kcal

 Health Goals  Food Page  Meal Planning



Banana

Search

banana

bananas

banana cue

banana jam

banana pie



d

ds

0.85g prot 81.09g fat

aped, With Salt

ds

731 kcal 0.49g prot 78.3g fat

Butter Oil, Anhydrous

Generic foods

100g :

876 kcal 0.28g prot 99.5g fat

Cheese, Blue

Generic foods

100g :

353 kcal 21.4g prot 28.7g fat



Health Goals



Food Page



Meal Planning



Banana

Bana

Bandana

a¹ z² e³ r⁴ t⁵ y⁶ u⁷ i⁸ o⁹ p⁰q[@] s[#] d[€] f⁻ g[&] h⁻ j⁺ k⁽ l⁾ m[/]w^{*}x["]c[']v[:]b[;]n[!]

,

.



?123

,



.



bread

Search

**Sesame Bread**

Generic foods

100g :
252 kcal 12.4g prot 3.5g fat

**Oat Bread**

Generic foods

100g :
236 kcal 10.4g prot 4.4g fat

**Multigrain Bread**

Generic foods

100g :
265 kcal 13.36g 4.23g fat
prot

**White Wheat Bread**

Generic foods

100g :
238 kcal 10.7g prot 2.15g fat

**Honey Wheat Bread**

Generic foods

100g :
252 kcal 12.4g prot 3.5g fat

**Buckwheat Round Bread**

Generic foods

100g :
164.55 kcal 5.06g prot 0.63g fat



Health Goals



Food Page



Meal Planning

Bread Machine Bread

Close

Oat Bread

Generic foods



per 100g

236 kcal

10.4g prot

4.4g fat

Add it to my Meal Plan

Quantity (g)

Meal

Breakfast

Lunch

Snack

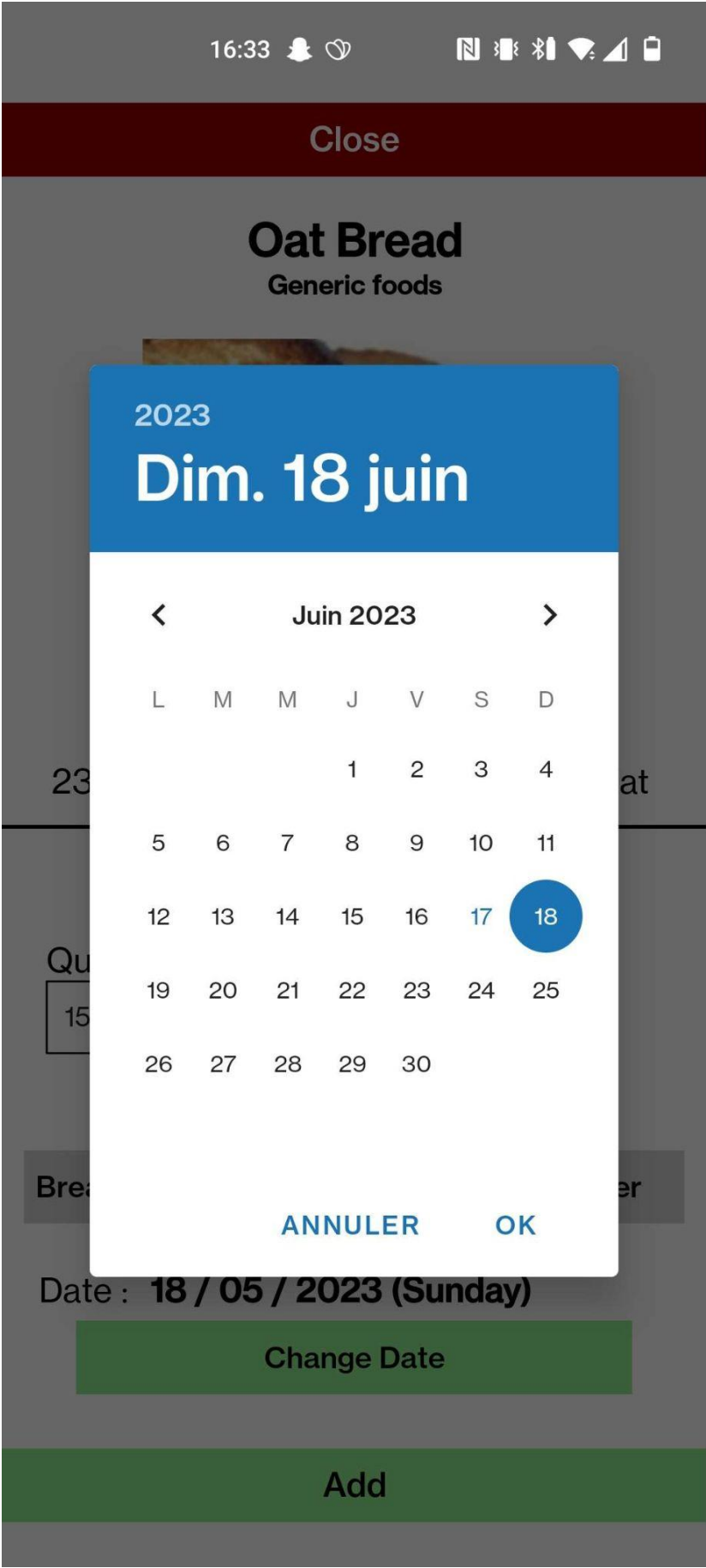
Diner

Date : **17 / 05 / 2023 (Saturday)**

Change Date

Add









←


17 / 05 / 2023


Saturday


1016.25 / 2793.7kCal

→

<div>Breakfast</div> <div>354 kCal</div>	<div>  <div> <div>Oat Bread</div> <div>354 kCal</div> <div>15.6g prot</div> <div>6.6g fat</div> </div> <div>150 g</div> <div>Remove</div> </div>	
<div>Lunch</div> <div>115.5 kCal</div>	<div>  <div> <div>Potato</div> <div>115.5 kCal</div> <div>3.075g prot</div> <div>0.135g fat</div> </div> <div>150 g</div> <div>Remove</div> </div>	
<div>Snack</div> <div>546.75 kCal</div>	<div>  <div> <div>Chocolate</div> <div>480 kCal</div> <div>4.2g prot</div> <div>30g fat</div> </div> <div>100 g</div> <div>Remove</div> </div>	<div>  <div>75 g</div> </div>
<div>Diner</div> <div>0 kCal</div>		

 Health Goals

 Food Page

 Meal Planning

