

# Provision of relevant data

## Internship report

By ECHAIB WALIM



# 1. Privacy statement



## FICHE DE CONFIDENTIALITE DES RAPPORTS ET MEMOIRES (SOUTENANCES)

CE DOCUMENT DOIT ETRE COMPLETE POUR TOUT RAPPORT OU MEMOIRE DIFFUSE A CESI EXIA ALGERIE ET CONTENANT DES INFORMATIONS SUR L'ENTREPRISE D'ACCUEIL

Titre du rapport ou du mémoire :

Année et filière Exia.Cesi Algérie : 2ieme année - Cycle supérieur Manager en Système d'information

Nom et prénom de l'étudiant : ECHAIB Walim

Date de la soutenance :

Nom du maître de stage : Imene Chibane

Structure d'accueil : Vibrand

Confidentialité du rapport ou du mémoire (soutenance)

(Cocher la case correspondante)

☒ Diffusion libre

Les rapports / mémoires sont conservés en archives et ils peuvent être librement consultés. Ils peuvent être utilisés par les destinataires, les études peuvent faire l'objet de publication ...

☐ Diffusion restreinte

Les rapports / mémoires sont ramassés à la fin de la soutenance et rendus à l'entreprise. Aucune reproduction n'est alors autorisée. La responsabilité de cette opération est confiée au stagiaire.

Dans le cadre de la politique de lutte contre le PLAGIAT, les rapports / mémoires seront susceptibles d'être analysés pour en vérifier les sources et ceci quel que soit le mode de diffusion ou ci-dessus.

Date : 25/12/2020

L'entreprise



Date : 26/12/2020

ECHAIB Walim

Date :

CESI EXIA Algérie



CESI ALGER: 60 rue du Kadous, Tixéraine, Birkhadem Alger - Algérie - T. +213 (0)21 40 55 00 Fax +213 (0)21 40 55 11.  
CESI ORAN: Résid. « Les Belles de Jour », Bat. C, Bloc D PLAZA millénium, Avenue Acimi Smâl, Hai Khemisti, Oran - Alger.  
T. +213 (0) 41 73 55 22 Fax +213 (0) 41 73 55 24- SPA au capital de 1 000 000 DA - RC 16/00-0963512 B03

## Table of content

1. Privacy statement.....	3
Table of figures:.....	6
2. Acknowledgements: .....	7
3. Introduction: .....	8
4. Introduction of the company: .....	9
5. Welcome in my new environment! .....	11
6. Specifications.....	12
7. Work performed: .....	14
I. Justification of technical choices: .....	14
a) Why Bootstrap? .....	14
b) React Native.....	14
c) Why SASS?.....	17
Why using a Framework? .....	17
d) NPM .....	18
e) Why Laravel? .....	18
f) Using Laratrust for role management: .....	19
g) Laravel Sanctum: .....	20
h) React Hooks:.....	20
i) Context API: .....	21
II. Initialization of the working environment:.....	21
a) Becoming familiar with GitHub: .....	21
b) Mobile project creation: .....	23
c) With using GIT: .....	23
III. Database: .....	24
a) Modeling: .....	24
b) Creation of migrations: .....	25
a) Implementation of seeders:.....	28
IV. Models: .....	29
a) Models creation:.....	29
b) Linking between the tables: .....	30
V. API: .....	32
VI. Controllers: .....	34
a) Routing: .....	34
b) Creation of a controller:.....	35

c) Project management: .....	35
VII. Functionalities: .....	36
a) Mobile: .....	36
1. API integration (mobile).....	37
2. Context Provider .....	38
3. Authentication: .....	40
4. Form submitting: .....	42
5. Display data (mobile).....	44
6. Get contacts from the device .....	46
7. External files upload .....	49
b) Web.....	51
1. Authentication .....	52
2. Email verification.....	53
3. Users monitoring.....	54
4. Contacts.....	55
5. Messages .....	56
6. Notes.....	56
7. Passwords.....	57
8. File storage.....	58
9. English Summary: .....	59
10. French summary:.....	60
11. Internship report: .....	62
12. Conclusion: .....	63
13. Resources:.....	64

## Table of figures:

Figure 1: Vibrand logo .....	9
Figure 2: Vibrand organization's chart .....	10
Figure 3: Functional analysis .....	13
Figure 4: Bootstrap logo .....	14
Figure 5 : React Native. ....	14
Figure 6: Sass logo.....	17
Figure 7: npm logo.....	18
Figure 8: Laravel logo .....	18
Figure 9: Laratrust logo.....	19
Figure 10: Laravel Sanctum logo .....	20
Figure 11: React Hooks logo .....	20
Figure 12: GitHub logo.....	21
Figure 13: repos GitHub .....	22
Figure 14: Publication frequency .....	22
Figure 15: Rating .....	22
Figure 19: MySQL Workbench logo.....	24
Figure 20: ERD Diagram .....	25
Figure 21: Token request with Insomnia (HTTP Client) .....	33
Figure 22: Kanban board .....	36
Figure 23: Prototype with Adobe XD .....	37
Figure 24: Login page.....	40
Figure 25: Menu page.....	41
Figure 26: Profile update page .....	42
Figure 27: Password page.....	46
Figure 28: Contacts page .....	47
Figure 29: File upload .....	50
Figure 30: Dashboard.....	52
Figure 31: Login page.....	52
Figure 32: Template for e-mail verification.....	53
Figure 33: Mailtrap.io Mailbox.....	53
Figure 34: Users list.....	54
Figure 35: Web contacts page .....	55
Figure 36: Web messages interface .....	56
Figure 37: Web notes page .....	56
Figure 38: Web form for submitting tasks .....	57
Figure 39: Passwords Hub .....	57
Figure 40: Web form for submitting passwords .....	58
Figure 41: Folders hub.....	58

## 2. Acknowledgements:

First, I would like to thank Mr. BENTALEB, CEO of Vibrand, for having integrated me into their team during these 2 months of internship where they volunteered to propose me a concrete project so that I could apply a good number of notions learned at CESI.

### 3. Introduction:

As part of my second year of advanced cycle at CESI Graduate School of Engineering, I chose to carry out my 2.5-month internship within an emerging start-up active in the audiovisual field under the supervision of Mrs. Imene Chibane, CEO of Vibrand.



## 4. Introduction of the company:



Figure 1: Vibrand logo

Vibrand is a start-up founded in 2019 by two experts, one in digital marketing and communication and the other in graphic design.

Its goal is to develop from an agency to a studio in order to diversify its services in the world of communication and audio visual whether on the national territory or outside the borders where the market is more fruitful and vast.

They offer various services mainly dedicated to companies that are characterized in:

- The management of the customer's identity on the Internet: that is to say that the image that people will have of this company will be defined by audio visual and informative campaigns that will aim to make the customer known.
- Elaboration of strategies related to the digital aspect of the box: which consists in establishing plans for the organization of sponsorship campaigns related to certain events organized for the promotion of products etc...
- Creation of IT solutions that can correspond to a specific need, tailor-made and easy to handle thanks to the know-how of developers who do everything to guarantee a better user experience.
- Photo shooting and filming of promotional clips.

**Organization Chart:**

Figure 2: Vibrand organization's chart

## 5. Welcome in my new environment!

Every working environment is different from the previous one and will be different in the future, this has to do with the people you work with (their involvement in what they do) but also with the workspace and the tools provided, in my case because of COVID-19 and the drop in orders I must telework, so this criterion is not really considered in my discovery report.

### **Observations**

#### **The recruitment process:**

Recruitment for the moment is closed, but their recruitment strategy is based on human criteria because, according to them, good manners take precedence over the professional aspect that is acquired through experience.

#### **The organization:**

For the organizational aspect, on receipt of a task a deadline is set for its completion, always one day before delivery to the client so that the work can be validated by the manager.

#### **Internal communication:**

Now, everything is organized in a Slack group allowing the sharing of ideas and answers to questions about a specific task or its realization, but also to schedule meetings to set up a workload plan.

#### **Inter-colleague relations:**

Now, there are not many collaborators within the start-up so that there is not much conflict or disagreement - everything is focused on the work that is done remotely.

#### **Relations with managers:**

As far as the relationship with the managers is concerned, there is a fairly good understanding because they consider their employees as partners and not as employees.

#### **The working environment:**

As said before, because of COVID-19 we are teleworking and do not share premises, but once a month we share the time of a week of the premises for a bootcamp and to establish a workload plan allowing us to consult on the arrival of new clients or to prepare meetings with the client.

#### **The tools at my disposal:**

No tools have been made available to me for this internship.

## 6. Specifications

This practical internship is for me an opportunity to apply the knowledge acquired at CESI but above all to learn new technologies so that they can be mastered by the end of the course.

The first thing that was done was to summaries all the tasks to be done to put some order into the management of the project by establishing rules.

### Objectives pursued:

1. To design a mobile utility for making important data available.
2. To make important data available on the device accessible on the web platform.
3. Manipulate extracted data to multiply its uses.
4. To be able to find contacts in case of theft or loss of the device.
5. Apply the knowledge learned at school.

### Details of the project:

Utility solution enabling a user to keep certain information (contacts, messages, notes, passwords etc....) accessible online in the event of loss of a device or emergency, he will be able to have it available using a web application.

### Functionalities:

#### Backend (server):

- Creation, deletion, modification, display of contacts.
- Creating, deleting, modifying, displaying messages.
- Creation, deletion, modification, display of notes.
- Creating, deleting, modifying, displaying passwords.
- Adding files involving folder creation operations or adding to an existing folder.
- Web authentication.
- Authentication using tokens (API).
- Role system.
- Account verification system.

#### Mobile application:

- System for centralizing data on all views using React Hooks and Context APIs.
- Ability to authenticate using a third-party application (Facebook, Google or other).
- Synchronization of contacts and messages.
- Password manager.
- Create, read and delete notes.
- Backup files.

- Saving albums for images.
- Access settings.
- Profile management.

### Dashboard (WEB):

- Access to messages
- Access to contacts
- Access to notes
- Access to passwords
- Access to folders and associated files
- Adding notes
- Adding passwords
- Data display

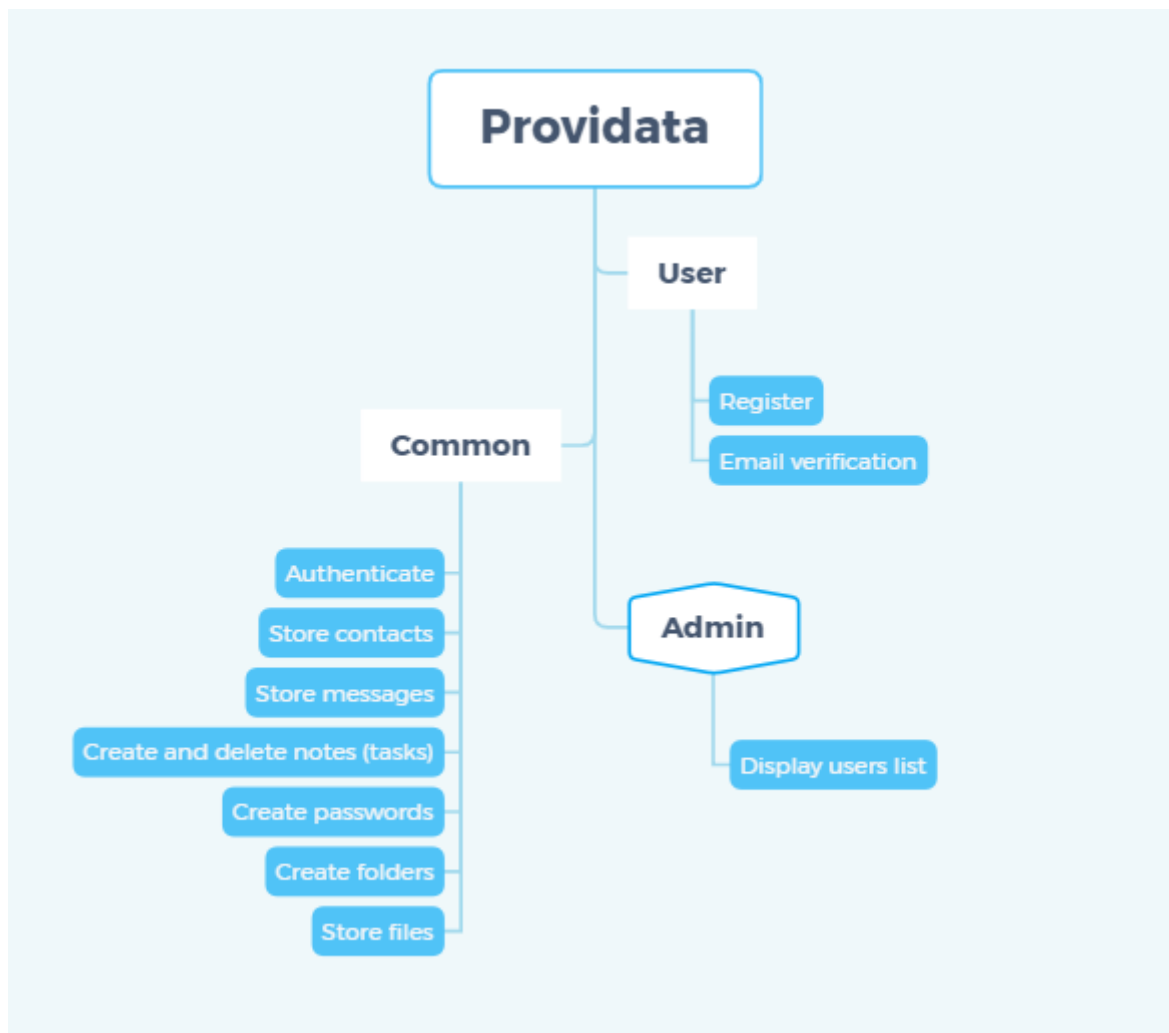


Figure 3: Functional analysis

## 7. Work performed:

### I. Justification of technical choices:

#### a) Why Bootstrap?

##### Bootstrap 4



Figure 4: Bootstrap logo

Well, because it is a Framework, an orchestra of components made available, which is used to create the foundations as well as the broad outlines of all or part of the architecture of a website, software, or application. For example, we have to do routing for a site, we take a component that is already ready and proven and we use it: saving time and reliability. The usefulness of a Framework is to avoid spending time developing what has already been done by others, often more competent.

With the technological advances and the different types of devices available on the market, we have been obliged to adapt our web interfaces to the different sizes in order to match them, it is possible to do this with simple CSS (Media Queries) but it would take much more time, this is where Bootstrap comes in, a Bootstrap grid has 12 columns by default. However, it should be known that it is flexible and extensible insofar as we can reduce the number of columns or insert new ones in each existing column and this just by modifying certain classes of our HTML.

The Bootstrap system has four predefined grid sizes so that the page modulates according to the screen size according to two hypotheses: either the elements resize while remaining positioned, or they stack up when the window becomes narrower and position themselves side by side when it widens.

#### b) React Native

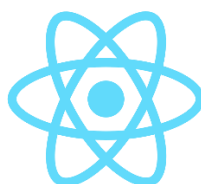


Figure 5 : React Native.

### **Allows Performing Development Faster, and, Therefore, It Costs Less**

It's no secret that every CEO wants to get the most benefit at a lower effort. With React Native the same code is used for deployment both on iOS and Android platforms. Business owner saves time and money by shortening the development

cycle and scaling the team which is involved in a project. It can be possible to cut development efforts by almost 50% without sacrificing either quality or productivity.

### **Provides Cross-Platform**

It means that it is possible to get an app for two platforms at once — iOS and Android.

React Native was created by the Facebook team. As you know, these guys will not suggest bad stuff. iOS support was released in early 2015, Android support — in autumn of the same year.

The framework integrates the pros of mobile app development with the native React environment agility and power.

### **The efficiency of Native Apps Development Increases Many-Fold**

React Native is based on React ideas and, therefore, allows creating strong mobile apps. Development of native apps is less efficient and productive. React Native shortens the development cycle, makes it possible to deliver products in the fastest way. Development of apps is adapted to the hybrid environment and has native results.

The framework uses famous ReactJS UI library developed by Facebook for user interfaces and apps creating and implements ReactJS under the hood. It transfers virtual DOM, improved app performance, and more simple programming processes from ReactJS.

Moreover, React Native has the 'live reload' feature, which isn't available for other native frameworks. It allows viewing the latest code changes at once. If two screens are opened, the first one shows the code, while the second one contains a mobile screen as a result of the code.

### **React Native Is Among the Leading Frameworks for Cross-Platform Mobile Development**

And its popularity is constantly growing. There is no need for Java, Swift, or Objective-C — for Android and iOS respectively. There is no need to hire a team of developers anymore. A JavaScript developer with expertise in native UI elements, design patterns for hybrid mobile app development, and APIs is enough. React Native has strong support and constant promotion by a huge community of developers. If problems arise, JS and native enthusiasts help to fix them quickly, as well as share unique development skills and required instruments.

### **Focus on UI and Access to Native API out of the Box.**

React Native focuses exclusively on a mobile UI building and compares favorably with other frameworks. It makes React Native look more like a JavaScript library than a framework.

A built mobile app is smoother and is loaded much faster than a classic hybrid one. As JavaScript interacts asynchronously with the native environment, UI feels fluid and is highly responsive.

### **Single Code Base for iOS and Android**

It means that a single code base is deployed to multiple mobile operating systems. Components are reused anytime at any level into existing code without you rewriting it and recompiling the app.

The framework is open source to be compatible with other platforms and available to the whole community of developers. It allows writing native module in a comparable language and linking it to React Native codebase in a simple way. It's needed in case you develop some features which aren't supported for now by React Native libraries.

### **Transform Any Web Project into a Mobile Decision Easily**

With React Native, a code is reusable. One update is needed for two platforms. It simplifies detecting bugs between codebases a lot. React Native interface is modular and intuitive. It means that developers who are not engaged in a project can easily understand it and take it as a basis. A team's flexibility increases, and web app updates are easier to make. QA engineers, in their turn, spend fewer hours delving into programming logic and writing relevant test cases. All this allows you to save time for transforming web project into a mobile decision.

### **App Performs as a Native App**

Building blocks of React Native are reusable native components and compile to native platform. Therefore, there is no longer need to use WebView system components. Native components which are used in iOS or Android platforms are comparable with React. As a result, an app performs as a native app with relevant functionality as well as look and speed.

The architecture of React Native is greatly adapted to the demands of mobile gadgets and has a strong performance for mobile environment.

It uses the graphics processing unit, whereas native platforms are more central processing unit intensive. In comparison with other hybrid frameworks and technologies, it allows developing apps ultra-fast and making them more agile.

With 3rd party plugins there is no longer a need for specific WebView functions. It's possible to link the plugin with a native module via the framework. These processes can be linked with an app features with smoother running, faster loading, and less memory needed.

### **React Native Will Not Disappear in the Years to Come**

React Native is widely adopted by developers. It is so because guys will not waste their time and efforts on technology that can disappear in a couple of years.



Even though the framework is still new, it continues to mature and strengthen positions. The proof is that the Facebook team has long-term plans to invest much more in its growth.

### c) Why SASS?



Figure 6: Sass logo

As you know, SASS is a CSS preprocessor, which means that it allows CSS files to be generated dynamically and aims to improve the writing of these files, providing greater flexibility.

This tool brings us a lot to our way of writing CSS by facilitating the stylization with the help of these different functionalities I quote:

#### **The variables**

This will serve us better when we want to respect the graphic charter by storing them in our variables, which will make it easier when we want to call them up, the same goes for fonts.

#### **The mixings**

Mixins are CSS fragments that can be used as functions.

Ease of writing, reading, and optimized code.

Writing with sass is better organized, more readable, less repetitive. This will make it easier to read our code.

### Why using a Framework?

#### **An organization to our project**

Based on the MVC model (Model - View - Controller), this clarifies our code and makes it more logical.

The model will take care of retrieving the data it will provide to the controller, the controller will perform several operations and display the result on the view.

#### **Reusable components and libraries**

Thanks to the Frameworks we will be able to use a number of packages containing predefined functions allowing us to avoid writing more lines of code, and to use functionalities that will allow us to progress more quickly in the design of our project.

### **A regularly updated database**

By choosing a Framework, one also chooses an active community that will detect and correct flaws or shortcomings in the Framework. In a way, the means of development are pooled. You will thus benefit from updates of the Framework with the set of improvements they contain.

#### **d) NPM**



*Figure 7: npm logo*

npm is the abbreviation for node package manager. It allows a transparent management of node.js packages. With this tool we can install, share and manage node.js packages, in our project it will allow us to use ready-made libraries of components facilitating the development and the integration of certain functionalities.

#### **e) Why Laravel?**



*Figure 8: Laravel logo*

### **Quality documentation when needed.**

Every web developer using Laravel has access to documentation that is particularly easy to understand, neat and much more organized.

Laravel adopts the controller-view-model. As it is an MVC-compliant framework, it comes with many built-in features and a robust and enhanced development architecture.

### **Updating the database**

Database updating is an automated process. Database tables can be updated with default data that can be used for preliminary configuration of the application or for testing the application.

### **Authentication and authorization by hierarchy**

It is a PHP framework that makes authentication of web applications very simple because everything is already integrated or configured in the framework. While

the main concern of most web application owners is to verify users and prevent access by unauthorized users, Laravel relieves them of this tension. It is relatively easy for owners to organize the authorization logic of their web application and restrict user access to all resources.

### Cache

We can take advantage of the basic caching system provided by Laravel to perform simple caching tasks such as storing objects in a file or database. At the same time, you also have the option of seamlessly integrating the web framework with popular caching systems such as Redis, APC and Memcached. Thus, you can simply increase the performance of the web application by integrating a robust caching system into Laravel.

### Artisan

Laravel offers an integrated tool called Artisan, which allows the user to carry out very quickly long programming tasks that can be easily done by Laravel's developers. In Laravel, the developer has to interact using a command line that manages the Laravel project environment. It is used to generate structured code and a database structure to facilitate the management of the database system.

### Templating Blade engine

Laravel comes with the Blade template engine. We can take advantage of Blade to include simple PHP code in the view and compile the views into PHP code. This way we can improve the performance of the website by compiling the views into PHP code, and cache the code until the views change. The blade template engine also allows you to display data more easily and extend the layout without affecting the speed of the application.

### Routing system

The routing system is the key feature of Laravel which is an easy-to-use routing method. The path can be defined on the application with a control layer and good flexibility. A directory is created to match the URI.

#### f) Using Laratrust for role management:



Figure 9: Laratrust logo

Laratrust is a Laravel package that allows you to manage very easily all the permissions (roles and permissions) in your application. Thanks to a very simple configuration process and API, this package will help us to create roles and assign them to our different types of users, to give them extra actions or to limit them.

This package creates tables that allow you to add or remove a role to a user at the command line.

### g) Laravel Sanctum:



Figure 10: Laravel Sanctum logo

Laravel Sanctum provides a featherweight authentication system for SPAs (single page applications), mobile applications, and simple, token-based APIs. Sanctum allows each user of your application to generate multiple API tokens for their account. These tokens may be granted abilities / scopes which specify which actions the tokens can perform.

Laravel Sanctum exists to solve two separate problems.

First, it is a simple package to issue API tokens to your users without the complication of OAuth. This feature is inspired by GitHub "access tokens". For example, imagine the "account settings" of your application has a screen where a user may generate an API token for their account. You may use Sanctum to generate and manage those tokens. These tokens typically have a very long expiration time (years) but may be manually revoked by the user at any time.

Laravel Sanctum offers this feature by storing user API tokens in a single database table and authenticating incoming requests via the Authorization header which should contain a valid API token.

### h) React Hooks:



Figure 11: React Hooks logo

#### What are React Hooks?

*Hooks* are a new addition in React 16.8. They let you use state and other React features without writing a class. It mainly uses to handle the state and side effects in react functional component. React Hooks are a way to use stateful functions inside a functional component. Hooks do not work inside classes, they let you use React without classes React provides a few built-in Hooks like *useState* and *useEffect*.

- **It enforces best practices.**

- **Easy to understand.**
- **Easy to test.**
- **It increases the performance and so on.**

### Why React Hook?

The first main reason is the Introduce state in a functional component. You know that the states cannot be used in functions. But with hooks, we can use states. Another reason is the handle side effect in react component. It means, now you can use newly introduced state such as useEffect.

But do you know for some scenarios, there are 3 places where react fails. While Reuse logic between components

- Has Huge components
- Confusing

#### i) Context API:

React Context API is a way to essentially create global variables that can be passed around in a React app. This is the alternative to "prop drilling", or passing props from grandparent to parent to child, and so on. Context is often touted as a simpler, lighter solution to using Redux for state management. I haven't used Redux myself yet, but every time I use React's Context API, I have to look it up because it doesn't seem obvious to me.

## II. Initialization of the working environment:

### a) Becoming familiar with GitHub:



Figure 12: GitHub logo

GitHub is a version management tool, so it allows several developers to work on the same project, on the same files, without any conflict with the changes made to the project.

For the structure of the project, I created a single repository containing the different applications (Platform, API and Mobile), this choice was regrettable because I had some conflicts when deleting some components to rebuild them, but it was nothing critical.









	therealwalim Passwords data fetched from API	e22d552 5 days ago	🕒 64 commits
	.idea	Email templating	last month
	Conception	Email templating	last month
	Mobiel	Passwords data fetched from API	5 days ago
	Rapport	Mobile profile	7 days ago
	Web	Passwords data fetched from API	5 days ago
	README.md	Update README.md	2 months ago
	package.json	Mobile profile	7 days ago

Figure 13: repos GitHub

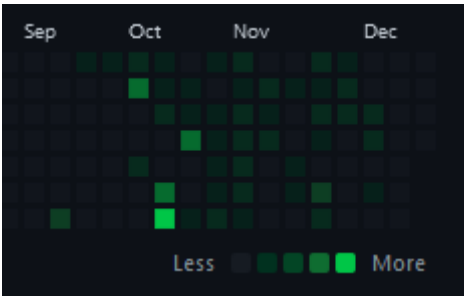


Figure 14: Publication frequency

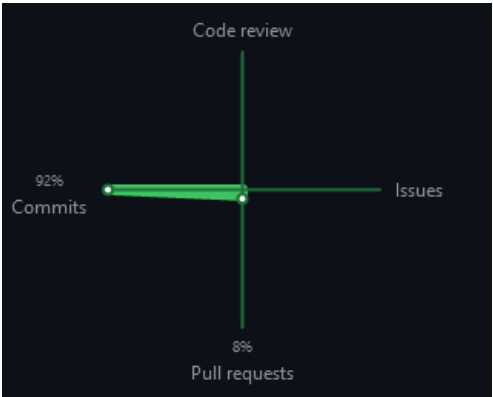
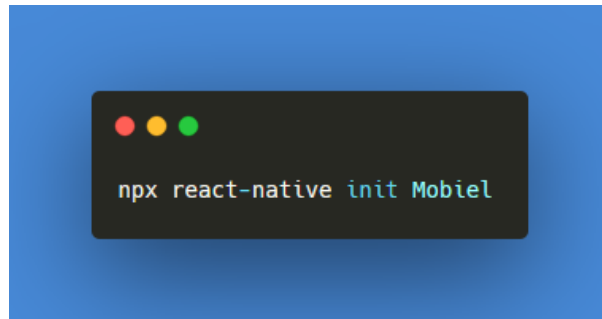


Figure 15: Rating

## b) Mobile project creation:

Before installing the React Native project (which is a boilerplate) you should first create a file locally, this is not a necessity, but it is a good practice to keep your desk tidy.

The installation was done using npm:

A terminal window with a dark background and three colored window control buttons (red, yellow, green) in the top left corner. The command `npx react-native init Mobiel` is entered in the terminal.

```
npx react-native init Mobiel
```

## c) With using GIT:

Once our project has been created, we will clone our project.

To link the file, we created to the directory created on GitHub as you can see below:

A terminal window with a dark background and three colored window control buttons (red, yellow, green) in the top left corner. The following commands are entered: `git clone https://github.com/therealwalim/gegevens-verstrekken.git`, `git remote add [-t <branch>] [-m <master>]`, and `C:\Users\ASUS\Desktop\gegevens-verstrekken (meester -> origin)`.

```
git clone https://github.com/therealwalim/gegevens-verstrekken.git
git remote add [-t <branch>] [-m <master>]
C:\Users\ASUS\Desktop\gegevens-verstrekken (meester -> origin)
```

Once the project has been recovered, it will need to be updated frequently in order to retain the changes made in the event of local file loss or accident.

Before sending our data, we will have to list them in order to know what changes we have made, the point ". "will be used to list all the changes in a single operation.

Then we will have to send the files to our repository with a message (hence the use of the -m parameter):

Then it will be necessary to send our information in our repository with the push command:



```
git add .  
git commit -m <Message>  
git push
```

### III. Database:

#### a) Modeling:



Figure 16: MySQL Workbench logo

For the modelling of our database, we used MySQL Workbench software, a tool that will allow us to create an Entity Relational Model. An entity-association diagram is a type of flowchart illustrating how "entities" such as people, objects or concepts are related to each other within a system. It uses a series of predefined symbols such as rectangles, rhombs and ovals connected by lines to describe the interconnections between entities, their relationships, and attributes. They mimic a grammatical structure, where entities are nouns and Relationship are verbs.

Here is a diagram of our database model:



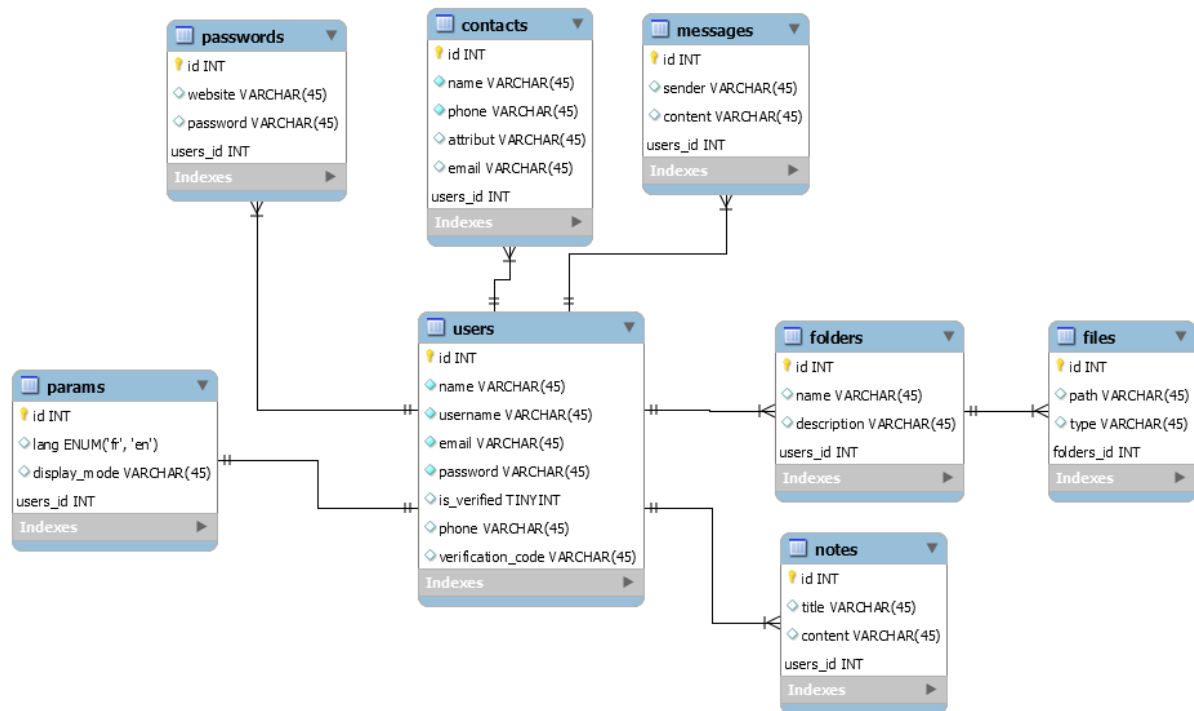


Figure 17: ERD Diagram

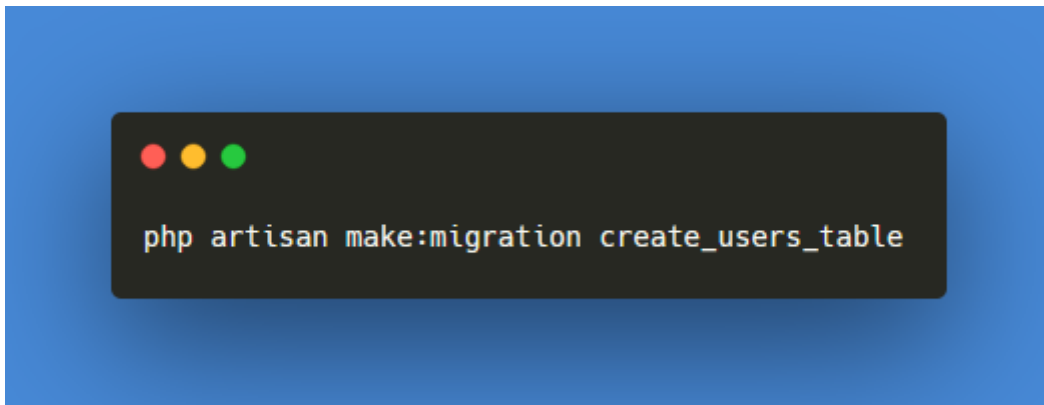
- For this modelling we favored an "Object" approach, for example, the attributes of a product had to be separated into tables or the types of point of sale or operations.
- For the relationships (n, m) we used pivot tables.
- The user table has a reflexive relationship because for each user created by another user, the name of the creator will appear with the user created.

## b) Creation of migrations:

Migrations are a method that allows us to deploy our database in one click.

Migrations are like a version control for our database, allowing a team to easily modify and share the database schema of the application. Migrations are usually associated with Laravel's schema builder to easily build the database schema of our application. And this will avoid us to recreate the tables each time for example in a team, if a member does not have such a table with the migrations everyone will have the same content.

To create a migration, you must use the following command:



It is important to know that a migration can contain two functions, the first one is UP to upload the table and another one is DOWN to overwrite the migration if it already exists.

Here is an example of the migration to create the user's table.

```
<?php

use Illuminate\Support\Facades\Schema;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Database\Migrations\Migration;

class CreateUsersTable extends Migration
{
    /**
     * Schema table name to migrate
     * @var string
     */
    public $tableName = 'users';

    /**
     * Run the migrations.
     * @table users
     *
     * @return void
     */
    public function up()
    {
        Schema::create($this->tableName, function (Blueprint $table) {
            $table->increments('id');
            $table->string('photo')->default('default.jpg');
            $table->string('name', 45);
            $table->string('username', 45)->nullable();
            $table->string('email', 45);
            $table->string('password', 255);
            $table->tinyInteger('is_verified')->default('0');
            $table->string('phone', 45)->nullable();
            $table->string('verification_code', 45)->nullable();
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists($this->tableName);
    }
}
```

The variable \$table is used to create a table, followed by the metadata or the type of fields filled in.

The index and foreign functions are used to index foreign keys and give their equivalent in other tables.

#### a) Implementation of seeders:

A seeder allows you to pre-add a user at the start of the migration to have some sort of default user, it is usually used as a super administrator, which will in turn be hidden because it has all possible permissions.

```
<?php

namespace Database\Seeders;

use Illuminate\Database\Seeder;
use App\Models\Role;
use App\Models\User;
use Illuminate\Support\Facades\Hash;

class UserSeeder extends Seeder
{
    /**
     * Run the database seeds.
     *
     * @return void
     */
    public function run()
    {
        $user = User::create([
            'username' => 'admin',
            'email' => 'admin@test.com',
            'password' => Hash::make('@TEST99'),
            'is_verified' => 1,
            'name' => 'Admin'
        ]);

        $admin = "administrator";

        $user->attachRole($admin);
    }
}
```

The Run() function allows us to create a new user by filling in his information and the super administrator role which will be unique.

To launch this Run() function we will have to call the class in another class which oversees launching these seeders functions:

```
namespace Database\Seeders;

use Illuminate\Database\Seeder;

class DatabaseSeeder extends Seeder
{
    /**
     * Seed the application's database.
     *
     * @return void
     */
    public function run()
    {
        // \App\Models\User::factory(10)->create();
        $this->call([
            LaratrustSeeder::class, //Generating the roles
            UserSeeder::class //seeding users
        ]);
    }
}
```

To launch this seeder, it will be necessary to use a command which is the following one:

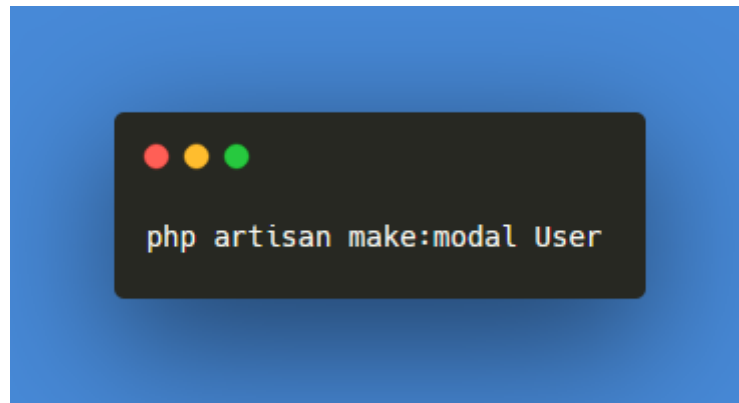
```
php artisan db:seed --class="NomDeLaClasse"
```

## IV. Models:

### a) Models creation:

A model is a class that will make our table an object, thanks to Eloquent we can manipulate tables as objects, instantiate and manipulate them and even create interactions between them thanks to the different types of existing relationships.

To create a model, we need to execute this command:



### b) Linking between the tables:

There are several types of relationships we mentioned earlier to describe our database schema, these relationships allow us to create relationships between models which will allow us to execute the queries we will need for our future operations:

```
class User extends Authenticatable
{
    use LaratrustUserTrait;
    use HasApiTokens, HasFactory, Notifiable;

    protected $fillable = [
        'name',
        'email',
        'password',
    ];

    protected $hidden = [
        'password',
        'remember_token',
    ];

    protected $casts = [
        'created_at' => 'date:d-m-Y',
    ];

    // Relationship
    public function Contact()
    {
        return $this->hasMany('App\Models\Contact');
    }

    public function Message()
    {
        return $this->hasMany('App\Models\Message');
    }

    public function Note()
    {
        return $this->hasMany('App\Models>Note');
    }

    public function Password()
    {
        return $this->hasMany('App\Models>Password');
    }

    public function Folder()
    {
        return $this->hasMany('App\Models\Folder');
    }

    public function Param()
    {
        return $this->hasOne('App\Models\Param');
    }

    public function Files()
    {
        return $this->hasManyThrough('App\Models\File', 'App\Models
\Folder');
    }
}
```

In the following example of the User model, we have created a \$fillable variable that will contain the fields of our table once the \$fillable table exists, Laravel will accept to send our data to the database, while for the password and the token we have chosen a \$hidden variable because it allows us to limit the content.

As for the relations between the tables:

hasOne/hasMany means that the template has a foreign key in the other table.

belongsTo means that the table has a foreign key from another table.

## V. API:

The creation of the API is done with the help of Sanctum, we will generate a token to each user who will connect, it will allow him to authenticate himself as a user and exchange with our API.

```
Route::post('/sanctum/token', function (Request $request) {
    $request->validate([
        'email' => 'required|email',
        'password' => 'required',
        'device_name' => 'required',
    ]);

    $user = User::where('email', $request->email)->first();

    if (! $user || ! Hash::check($request->password, $user->password)
    || $user->is_verified === 0) {
        throw ValidationException::withMessages([
            'email' => ['The provided credentials are incorrect.'],
        ]);
    }

    $token =
    $user->createToken($request->device_name)->plainTextToken;

    $response = [
        'user' => $user,
        'token' => $token
    ];

    return response($response, 201);
});
```



First, a call to the API is made to request a token, the data sent is subject to validation, then we will search for the first user who corresponds to the email address entered, a validation error will occur if the email is incorrect, the password does not match it, or the account is not verified, then the token is created and saved, the server will return a reply with the user's data and the new token if successful.

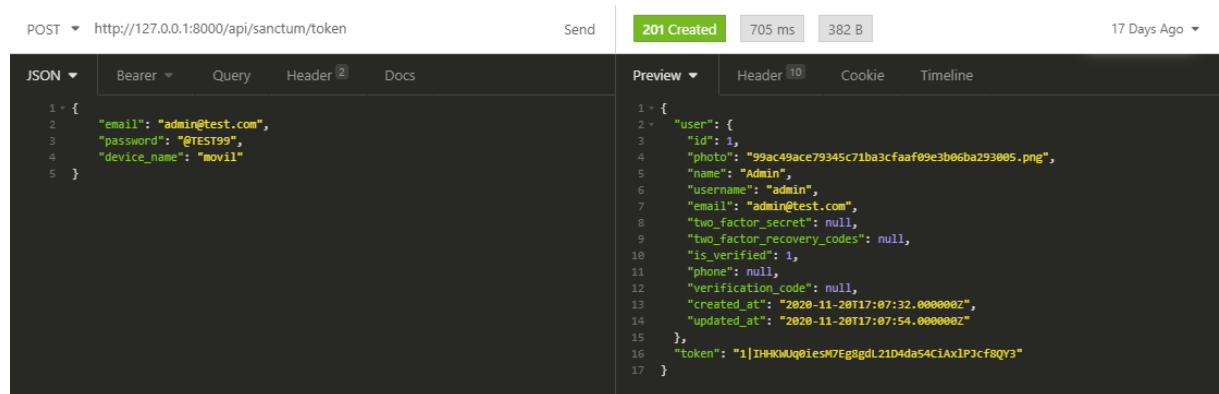


Figure 18: Token request with Insomnia (HTTP Client)

```
Route::middleware('auth:sanctum')->post('/logout', function (Request
$request) {
    $request->user()->tokens()->delete();

    return response('Loggedout', 200);
});
```

We have also set up a system of disconnection allowing to remove the token from the user so that the user no longer has access to his account after the disconnection.

```
Route::middleware('auth:sanctum')->get('/user', function (Request
$request) {
    return $request->user();
});
```

But also, to have access to users via a protected route.

## VI. Controllers:

### a) Routing:

A route is like a path that we give to a URL that we create, indicating in our case the method that will display, for example, our view.

```
Route::get('/', [DashboardController::class, 'index']);
```

We use a GET request for the URL which will call the index() method of the DashboardController :

```
class DashboardController extends Controller
{
    public function index(){
        $user = User::all();
        $message = Message::where('user_id', '=',
auth()->id()->get());
        $contact = Contact::where('users_id', '=',
auth()->id()->get());
        $note = Note::where('user_id', '=', auth()->id()->get());
        $password = Password::where('user_id', '=',
auth()->id()->get());

        $ucount = $user->count();
        $mcount = $message->count();
        $ccount = $contact->count();
        $ncount = $note->count();
        $pcount = $password->count();

        return view('pages.home',
['ucount'=>$ucount, 'mcount'=>$mcount, 'ccount'=>$ccount,
'ncount'=>$ncount, 'pcount'=>$pcount]);
    }
}
```

This method returns the Home view which is in the "Pages" folder, we can also send to the view variables or operations as shown in the demonstration.

## Middleware:

Middleware provides a convenient mechanism for filtering HTTP requests entering your application. For example, Laravel includes a middleware that verifies the user of your application is authenticated. If the user is not authenticated, the middleware will redirect the user to the login screen. However, if the user is authenticated, the middleware will allow the request to proceed further into the application.

```
Route::middleware('auth')->group(function () {  
    // Dashboard  
    Route::get('/', [DashboardController::class, 'index']);  
    Route::get('/logout', [DashboardController::class,  
    'logout'])->name('logout');  
  
    // Admin  
    Route::get('/users', ['middleware' => ['role:administrator'],  
    'uses' => 'App\Http\Controllers\AdminController@index']);  
});
```

## b) Creation of a controller:

The task of a controller is to receive a request (which has already been sorted by a route) and to define the appropriate response to it.

To create a controller, we need to execute the following command:

```
php artisan make:controller DashboardController
```

## c) Project management:

Using a Kanban board which is a working tool to apply the Kanban method to a workflow. Kanban boards use Kanban sheets to visualize the evolution of work units

in a production process, which will allow the assignment of specific tasks to one or more members of the development team.

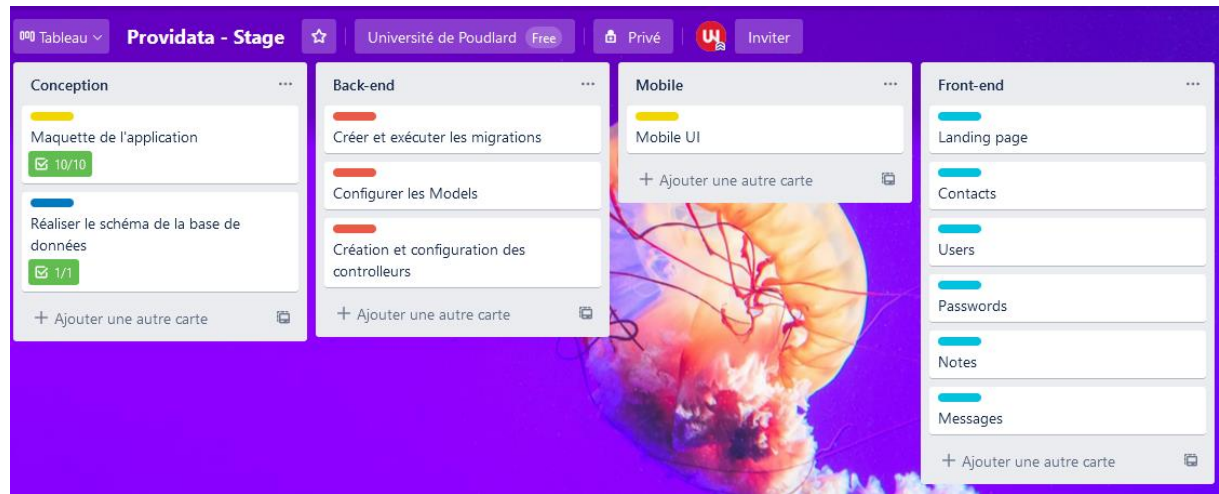


Figure 19: Kanban board

## VII. Functionalities:

### a) Mobile:

The role of the mobile application is initially to provide the essential data for backup, such as the contacts and messages that are present inside the phone. It also makes it possible to display the ordered and structured data in the various screens that make up the application, it also makes it possible to create data that it transmits to an API so that it can be accessed via a web interface.

For this part we used React Native, a way to develop multi-OS compatible applications. React Native is like React, but it uses native components instead of web components as building blocks. To understand the basic structure of a React Native application we therefore need to understand some of the basic concepts of React, such as JSX, components, status, and accessories. If you are already familiar with React, you still need to learn some React-Native specific tricks, such as native components.



Figure 20: Prototype with Adobe XD

## 1. API integration (mobile)

After the design of the mobile module, I moved on to the stage of integrating the API's that will allow communication between the module and the server.

The communication will be done thanks to AXIOS: a JavaScript library working as an HTTP client. It allows to communicate with APIs using all types of requests (POST, GET...).

Each user of the service has a unique ID which, if sent to our server, will be able to identify him/her and retrieve information specific to it:

```
axios.defaults.baseURL = 'http://10.0.2.2:8000';

const [password, setPassword] = useState([]);
const [name, setName] = useState('');

useEffect(() => {
  axios.defaults.headers.common['Authorization'] = `Bearer
  ${user.token}`;

  axios.get('/api/password')
    .then(response => {
      setPassword(response.data);
    })
    .catch(error => {
      console.log(error.response);
    })

  axios.get('/api/user')
    .then(response => {
      setName(response.data.photo);
      console.log(name);
    })
    .catch(error => {
      console.log(error.response);
    })
}, []);
```

## 2. Context Provider

React context allows you to share information to any component, by storing it in a central place and allowing access to any component that requests it (usually you are only able to pass data from parent to child via props).

The provider acts as a delivery service. When a consumer asks for something, it finds it in the context and delivers it to where it is needed.

In our case:

```

import React, { useState } from 'react';
import AsyncStorage from '@react-native-community/async-storage';
import axios from 'axios';

axios.defaults.baseURL = 'http://10.0.2.2:8000';

export const AuthContext = React.createContext({});

export const AuthProvider = ({children}) => {
  const [user, setUser] = useState(null);
  const [error, setError] = useState(null);

  return (
    <AuthContext.Provider
      value={{
        user,
        setUser,
        error,
        login: (email, password) => {
          axios.post('/api/sanctum/token', {
            email,
            password,
            device_name: 'mobile',
          })
            .then(response => {
              const userResponse = {
                email: response.data.user.email,
                token: response.data.token,
                name: response.data.user.name,
                phone: response.data.user.phone,
                password: response.data.user.password,
                id: response.data.user.id,
                photo: response.data.user.photo,
              };
              setUser(userResponse);
              setError(null);
              AsyncStorage.setItem('user', JSON.stringify(userResponse));
              console.log(response.data.user.name+" connected")
            })
            .catch(error => {
              const key = Object.keys(error.response.data.errors)[0];
              setError(error.response.data.errors[key][0]);
            })
        }, register: (email, name, phone, password) => {
          axios.post('/api/users', {
            email,
            name,
            phone,
            password,
          })
            .then(response => {
              console.log(response.message)
            })
            .catch(error => {
              console.log(error.response);
            })
        },
        logout: () => {
          axios.defaults.headers.common['Authorization'] = `Bearer ${user.token}`;

          axios.post('/api/logout')
            .then(response => {
              setUser(null);
              AsyncStorage.removeItem('user')
              console.log("User disconnected")
            })
            .catch(error => {
              console.log(error.response);
            })
        }
      }}>
      {children}
    </AuthContext.Provider>
  );
}

```

Using Context API, we can make the data available for all the screens by putting our Context above the entire content of the App:

```
import React from 'react';
import { AuthProvider } from './AuthProvider';
import Routes from '../Routes';

export const Providers = ({}) => {
  return (
    <AuthProvider>
      <Routes />
    </AuthProvider>
  );
};
```

### 3. Authentication:

To set up a consistent authentication system, we must first establish the connection with our API, generate a token and then save it within the application:

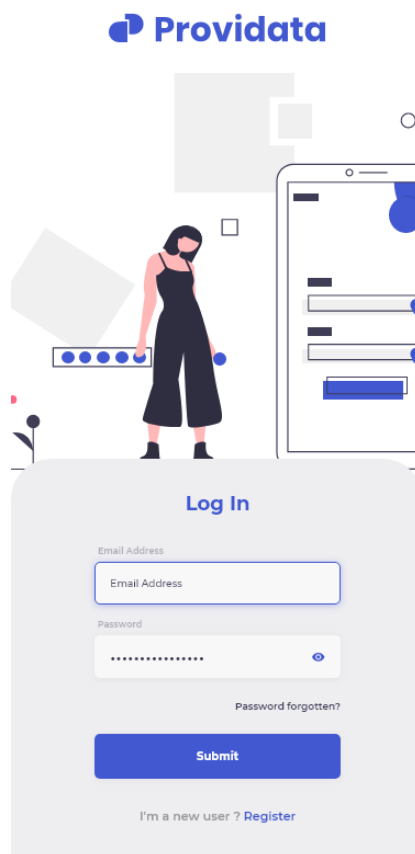


Figure 21: Login page



```

login: (email, password) => {
  axios.post('/api/sanctum/token', {
    email,
    password,
    device_name: 'mobile',
  })
  .then(response => {
    const userResponse = {
      email: response.data.user.email,
      token: response.data.token,
      name: response.data.user.name,
      phone: response.data.user.phone,
      password: response.data.user.password,
      id: response.data.user.id,
      photo: response.data.user.photo,
    }
    setUser(userResponse);
    setError(null);
    AsyncStorage.setItem('user', JSON.stringify(userResponse));
    console.log(response.data.user.name+" connected")
  })
  .catch(error => {
    const key = Object.keys(error.response.data.errors)[0];
    setError(error.response.data.errors[key][0]);
  })
})

```

You will be redirected to our contextual menu; you can see that user has been saved and the data are displayed, this menu page gives you access to some data like statistics and quick shortcuts actions you can use to achieve some tasks.

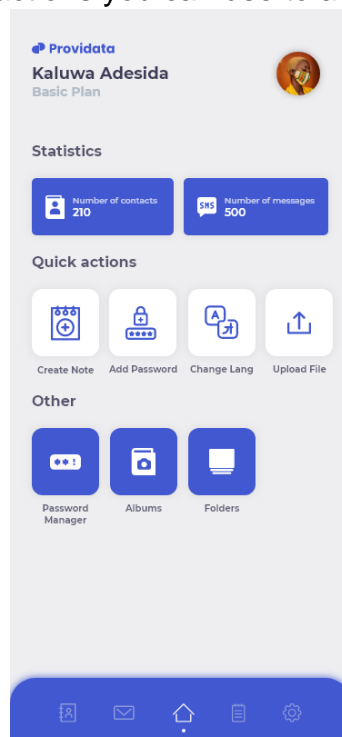


Figure 22: Menu page

After registering the token, the user will not need to log in again until logging out and decide to delete the token linked to the account.

```
logout: () => {  
  axios.defaults.headers.common['Authorization'] = `Bearer  
  ${user.token}`;  
  
  axios.post('/api/logout')  
  .then(response => {  
    setUser(null);  
    //SecureStore.deleteItemAsync('user')  
    AsyncStorage.removeItem('user')  
    console.log("User disconnected")  
  })  
  .catch(error => {  
    console.log(error.response);  
  })  
}
```

#### 4. Form submitting:

As far as sending the forms is concerned, I have created a function for each request sent to call from each form the required elements so that they can be sent to the server:

Providata  
Kaluwa Adesida  
Basic Plan

Kaluwa Adesida

New Password

Email

Phone Number

Save

Disconnect

Figure 23: Profile update page

```

// Function
profile: (password) => {
  axios.put('/api/users/' + user.id, {
    password,
  })
  .then(response => {
    console.log(response.message)
  })
  .catch(error => {
    console.log(error.response);
  })
},

// Profile.js
const { profile } = useContext(AuthContext)

const [password, setPassword] = useState('');

<TextInput
  style={styles.input}
  placeholder={user.password ? user.password : "Password"}
  onChangeText={text => setPassword(text)}
  secureTextEntry={true}
/>

<TouchableHighlight
  style={styles.button}
  onPress={() => profile(password)} // Call to the function
>
  <Text style={{color:'white',fontWeight:"bold",fontSize:15}}>Submit</Text>
</TouchableHighlight>

// Route on Laravel
Route::apiResource('users','App\Http\Controllers\UserController');

// Controller on Laravel
public function update(Request $request, User $user)
{
    if ($user->update($request->all())) {
        return response()->json([
            'success' => 'User updated with success'
        ], 200);
    }
}

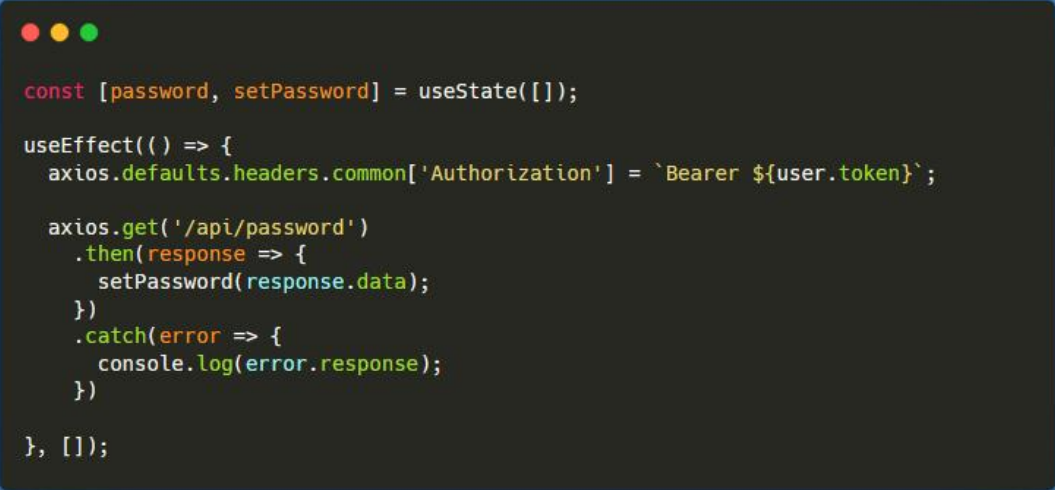
```

### Server side:

These two interfaces have a similar operating logic, once the fields are filled in a POST request is sent to the relevant APIs containing the data inserted by the user and his ID so that the server can check and save them. They can then be displayed on the platform.

## 5. Display data (mobile)

The display of data is done via calls to the API, which sends us back our data in JSON format, then we take care of processing it to display it.



```
const [password, setPassword] = useState([]);

useEffect(() => {
  axios.defaults.headers.common['Authorization'] = `Bearer ${user.token}`;

  axios.get('/api/password')
    .then(response => {
      setPassword(response.data);
    })
    .catch(error => {
      console.log(error.response);
    })
}, []);
```

First, we declare a variable and a function with the hook **useState**, and we send a GET request to the API with the correct headers, then we put the response which are our data array onto our variable.

```

//Data to display:
{
  "password": [
    {
      "id": 1,
      "service": "facebook",
      "serviceid": "walim@messenger.com",
      "password": "test999",
      "user_id": 1
    },
    {
      "id": 2,
      "service": "spotify",
      "serviceid": "walim@spotify.com",
      "password": "test999",
      "user_id": 1
    },
  ],
  "count": 5
}

//Flatlist
<FlatList
  data={password.password}
  keyExtractor={({item, index}) => index.toString()}
  renderItem={({item}) => <PasswordCard item={item} site={item.service} />
}
/>

//PasswordCard Component
export default function PasswordCard({item}) {

  const [hidePass, setHidePass] = useState(true);
  const test = `${item.service}`;

  return (

    <View style={{
      padding: 20,
      backgroundColor: service[test].color,
      elevation: 2,
      borderRadius: 5,
      flexDirection: "row",
      alignItems: "center",
      justifyContent: "space-evenly",
      marginTop: 5,
      marginBottom: 5,
    }}>
      <Icon style={styles.service} name={item.service} size={27}
color="white" />
      <View style={styles.TextContainer}>
        <Text style={styles.Text}>{item.serviceid}</Text>
        <TextInput
          style={styles.Password}
          value={item.password}
          editable={false}
          secureTextEntry={hidePass ? true : false}
        />
      </View>
      <TouchableHighlight style={styles.icon}>
        <BenjaminButton name={hidePass ? 'eye-off' : 'eye'} size={22}
color="white" onPress={() => setHidePass(!hidePass)} />
      </TouchableHighlight>
    </View>
  )
}

```

Like you can see, that is the kind of data we can get from our API, then we parse it to with key extractors so we can use it as key value and render the Password with our data so we can display it:

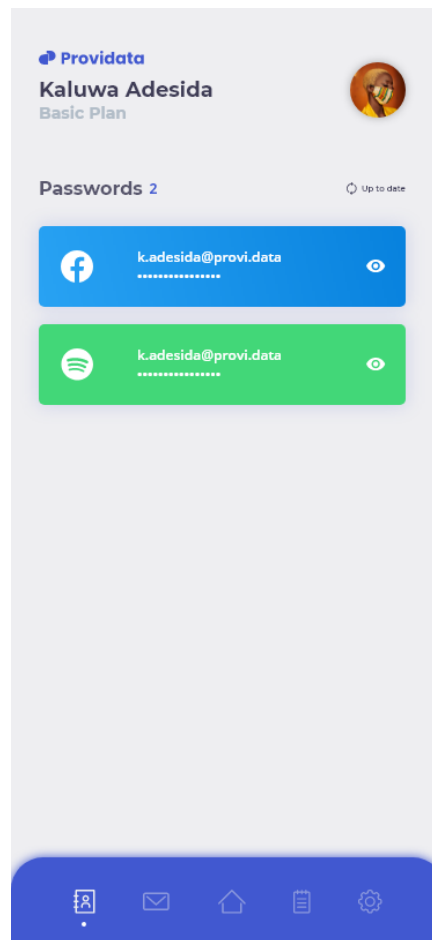


Figure 24: Password page

## 6. Get contacts from the device

With the help of the react-native-contacts library, we can retrieve the contacts present on the phone if we use another library that allows us to obtain permission.

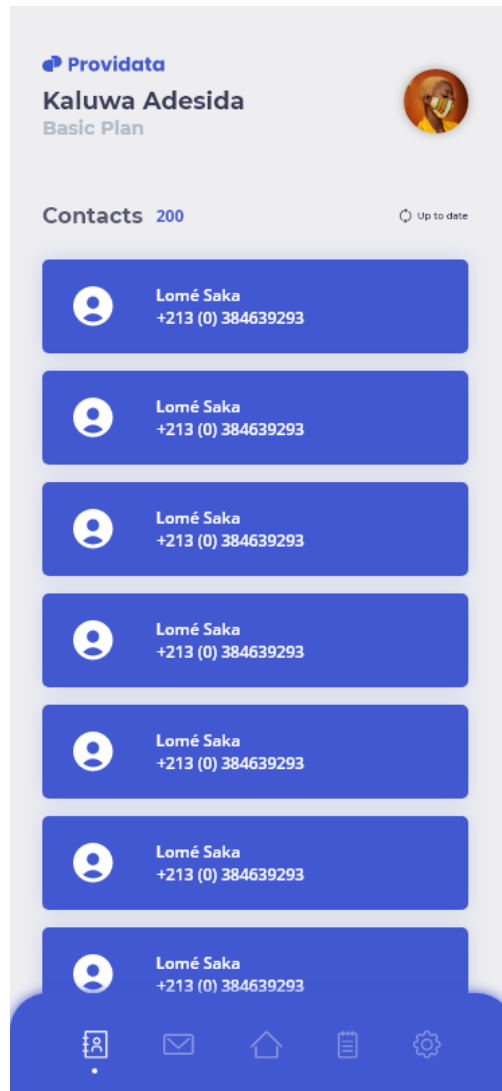


Figure 25: Contacts page

List information is mapped to components for customized display using FlatLists.

```
import con from 'react-native-contacts';

const [contact, setContact] = useState([]);
const [count, setCount] = useState('');

useEffect(() => {
  PermissionsAndroid.request(
    PermissionsAndroid.PERMISSIONS.READ_CONTACTS,
    {
      'title': 'Contacts',
      'message': 'This app would like to view your contacts.'
    }
  ).then(() => {
    con.getAll((err, contacts) => {
      if (err === 'denied'){
        // error
      } else {
        // contacts returned in Array
        setContact(contacts);
      }
    })
  })
  .then(() => {
    con.getCount((counts) =>{
      setCount(counts)
    })
  })
  .catch((err)=> {
    console.log(err);
  })
}, []);
```

In the first instance permission is requested to use the data present on the device, the response can take on two states, the first arises when the user refuses to allow the application to access the requested data, and the second returns the contacts and saves them in our table so that the data can be processed.



```
// Element
<FlatList
  data={contact}
  renderItem={
    ({item}) => <ContactCard item={item}/>
  }
/>

// Component
export default function ContactCard({item}) {
  return (
    <View style={styles.container}>
      <Icon name="users" size={27} color="white" />
      <View style={styles.TextContainer}>
        <Text style={styles.Text}>{item.givenName}</Text>
        {item.phoneNumbers.map(phone=>(
          <Text style={styles.Text}>{phone.number}</Text>
        ))}
      </View>
    </View>
  )
}
```

The data is thus transmitted to the component, which will first break down the raw data so that it can be restructured for access and display.

## 7. External files upload

It is easy to send text data to the server as it is just a matter of storing it in a variable and displaying it, but as far as the files are concerned, you have to specify the type and extension in order to give the servers the information they need to pass our request through and not reject it.

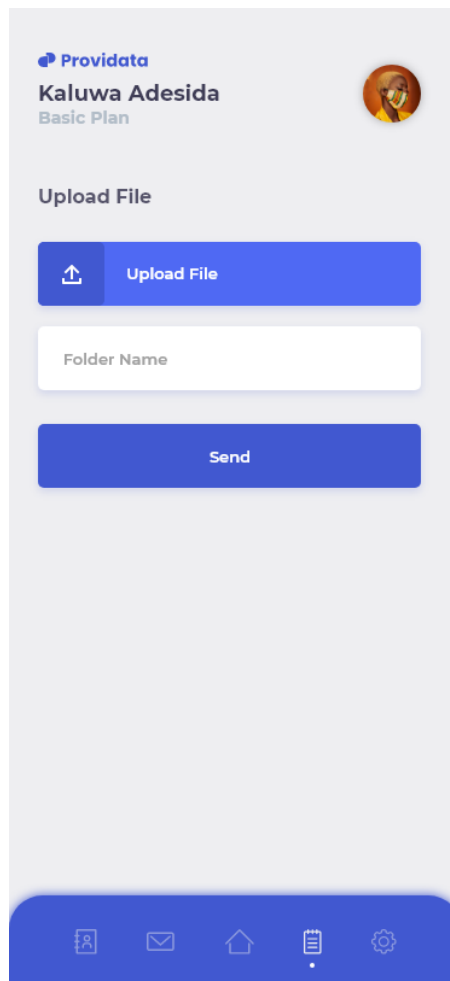


Figure 26: File upload

To send the request we are going to send what is called a multipart form, sending several types of data including files for the server to recognize them.



```
const formData = new FormData();
formData.append('Note', {
  uri: //Our image path
  type: 'image/jpeg',
  name: "imagenname.jpg",
});
axios({
  url : api,
  method : 'POST',
  data : formData,
  headers: {
    Accept: 'application/json',
    'Content-Type': 'multipart/form-data',
    'Authorization': 'Basic YnJva2VyOmJyb2tldl8xMjM='
  }
})
.then(function (response) {
  console.log("response :", response);
})
.catch(function (error) {
  console.log("error from image :");
})
```

## b) Web

As for the web version, we use it as a supervision support, its main role is the display of data and statistics but also for the administrator to have a view on the users.

This support will allow the user to access his data from any location, we have reduced the constraint to a simple internet connection which is nowadays accessible from any location in view of the general deployment.

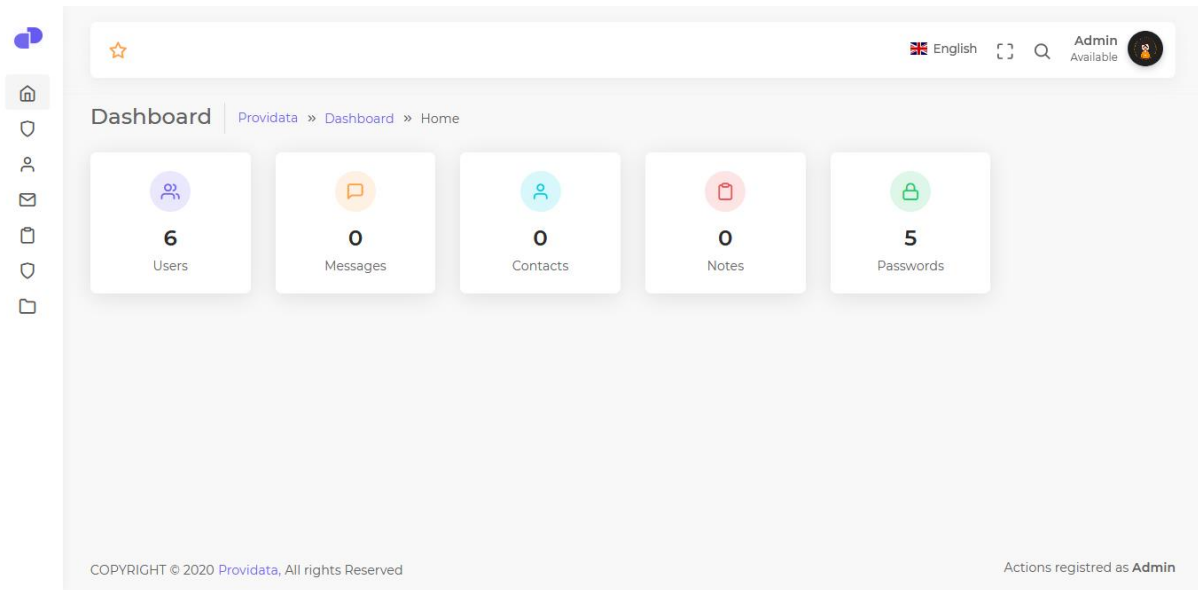


Figure 27: Dashboard

## 1. Authentication

Laravel Fortify registers the routes and controllers needed to implement all Laravel's authentication features, including login and registration.

Since Fortify does not provide its own user interface, it is meant to be paired with our customized user interface which makes requests to the routes it registers.

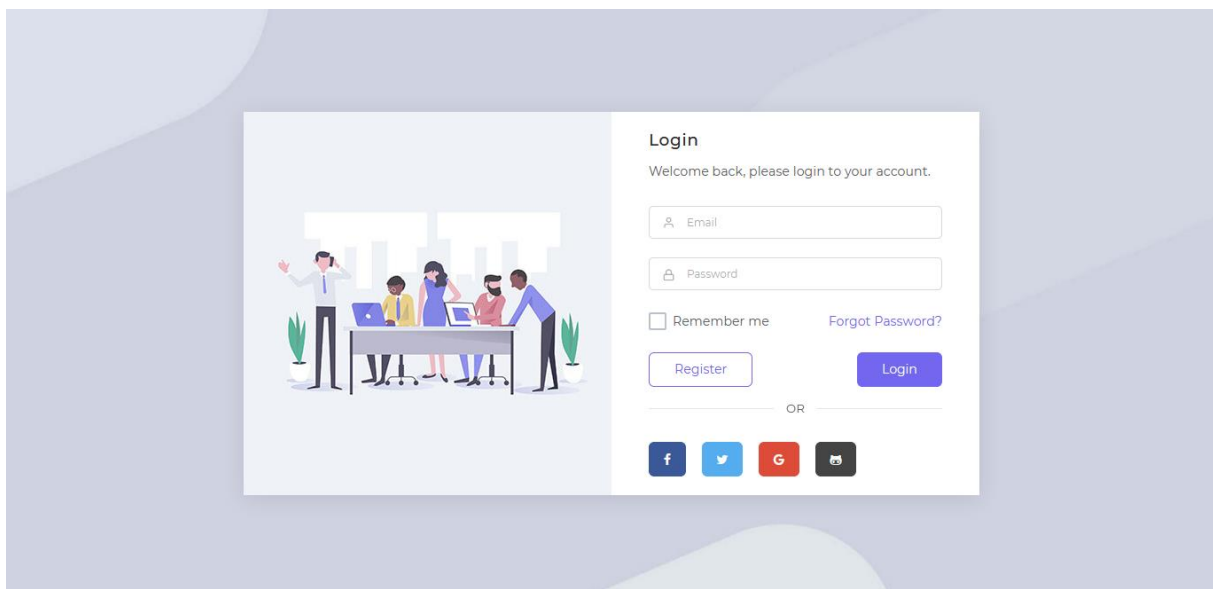


Figure 28: Login page

## 2. Email verification

When the user registers on the platform (or on the application) he automatically receives an e-mail asking him to check his account accompanied by a link with a token allowing him to do so.

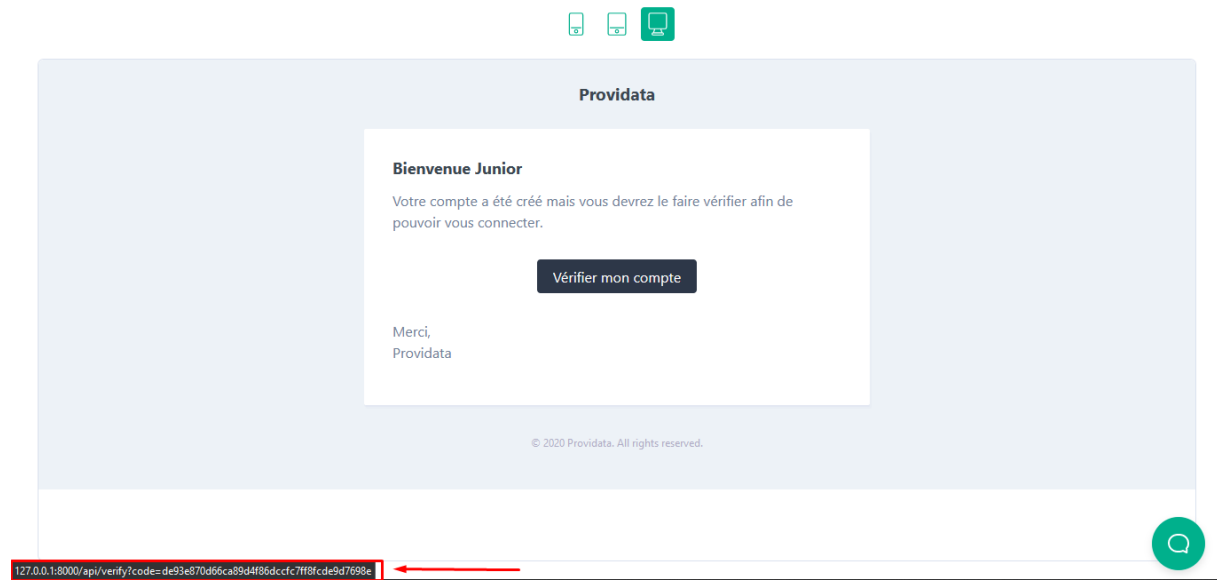


Figure 29: Template for e-mail verification

We use the Mailtrap tool to capture the e-mails sent by our platform, it allows us to preview the content but also to interact with it.

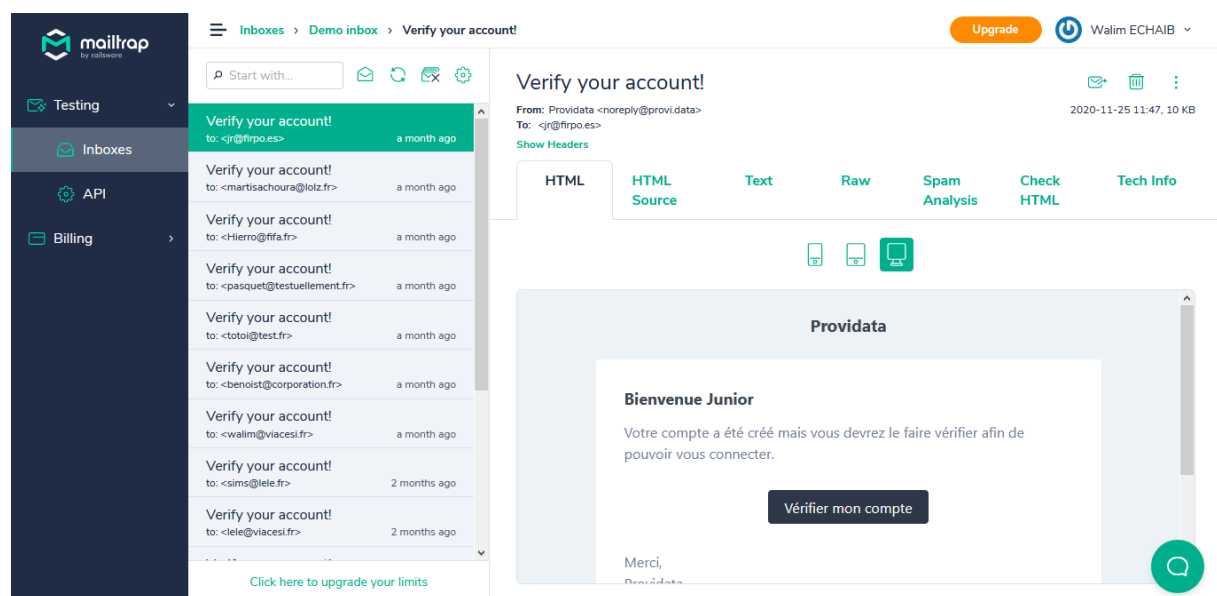


Figure 30: Mailtrap.io Mailbox

We have put in place a constraint to verify that the user is properly authenticated, otherwise he will not be able to access his account.

```
if ($user &&
    Hash::check($request->password, $user->password) && $user->is_verified === 1
) {
    return $user;
}else{
    throw ValidationException::withMessages([
        'unverified' => ['Please verify your account'],
    ]);
}
```

### 3. Users monitoring

The platform administrator has access to the complete list of users using the platform, thanks to the role system established on the platform using the Laratrust package.

The screenshot displays the 'Administration' section of a web application, specifically the 'Users' management page. The page features a sidebar with navigation icons and a top navigation bar with language and user settings. The main content area shows a table of users with the following data:

ID	Name	Email	Verified	Phone	Created At
1	Admin	admin@test.com	Yes	+213781879800	20-11-2020
3	loobar	martisachoura@lolz.fr	Yes		25-11-2020
5	test	testuelememnt@test.fr	No		25-11-2020
6	Junior	jr@firpo.es	No		25-11-2020

The interface also includes a search bar, a pagination control showing 'Page 1 of 1', and a footer with copyright information: 'COPYRIGHT © 2020 Providata, All rights Reserved' and 'Actions registered as Admin'.

Figure 31: Users list

To do this, we gather the necessary information to be displayed, to send it in JSON format, to provide a dataset that can be used by our board.

```
public function userdata()
{
    $users = User::select('id', 'name', 'created_at',
    'email', 'photo', 'is_verified', 'phone')->get();
    $users_coll = collect();

    foreach ($users as $user) {
        $users_coll->push([
            'id' => $user->id,
            'name' => $user->name,
            'email' => $user->email,
            'created_at' => $user->created_at,
            'phone' => $user->phone
        ]);
    }

    return response()->json($users);
}
```

#### 4. Contacts

Once retrieved, the contacts are put online in the database, they are then made available to the platform so that they can be accessed remotely.

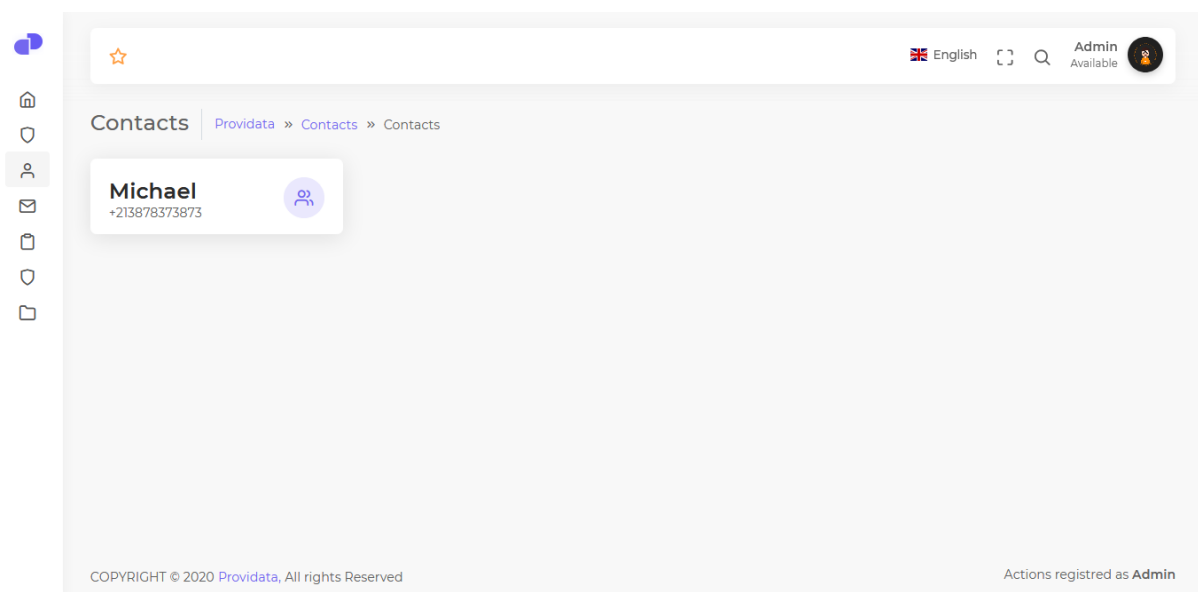


Figure 32: Web contacts page

## 5. Messages

It is the same for messages, it will be necessary to retrieve the data relating to the sender as well as the content of the latter to render the data in the appropriate component.

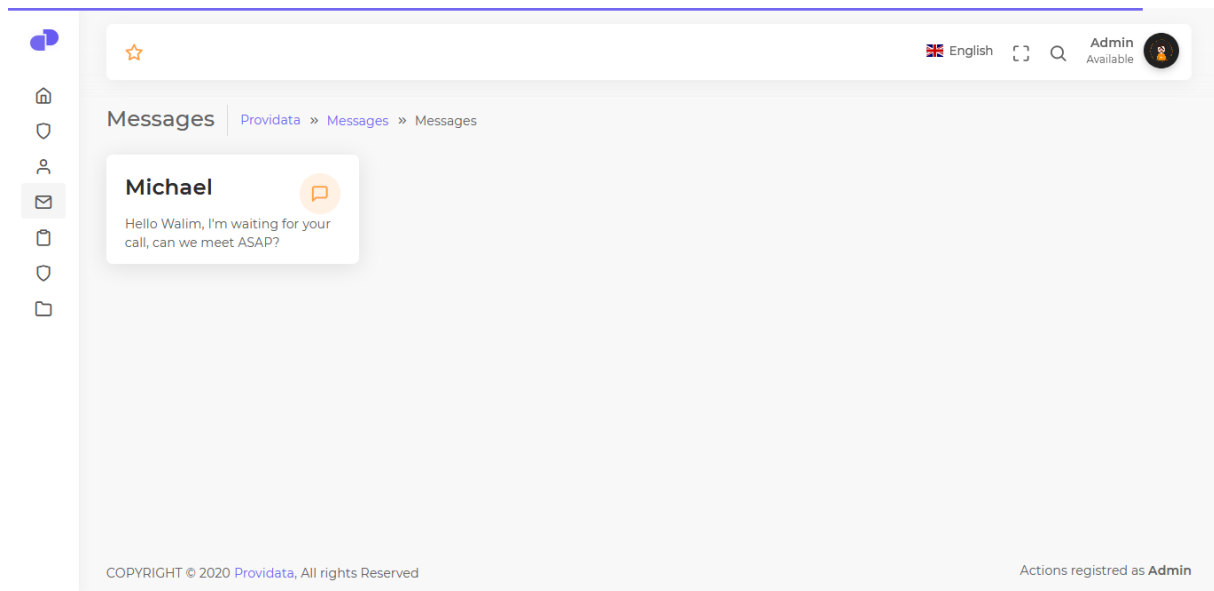


Figure 33: Web messages interface

## 6. Notes

The system of notes or tasks allows the user to save notes that he will be able to classify on the page to determine their priority, he will be able to create them via a form and delete them but in our case not modify them because each note must characterize a specific content and to give importance to this system, we have retracted the fact that they are modifiable.

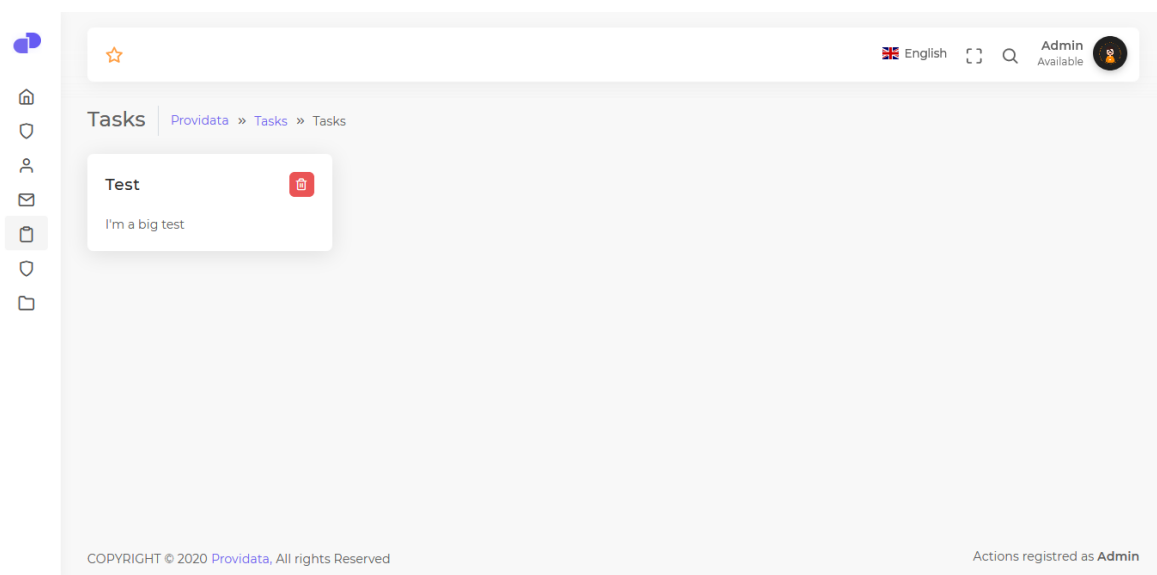


Figure 34: Web notes page



The screenshot shows the 'Add Task' interface. At the top, there's a navigation bar with 'English', a search icon, and a user profile 'Admin Available'. Below this, a breadcrumb trail reads 'Providata » Tasks » Add Task'. The main content area features a form titled 'Add a note' with two input fields: 'Title' and 'Content'. A blue 'Submit' button is at the bottom of the form. On the left, a vertical sidebar contains icons for home, tasks, profile, messages, and documents. The footer includes the copyright notice 'COPYRIGHT © 2020 Providata, All rights Reserved' and the text 'Actions registered as Admin'.

Figure 35: Web form for submitting tasks

## 7. Passwords

The password management system allows the user to save his passwords on his phone and then retrieve them on the platform so that they are accessible at any time, the user will also be able to create them via the form dedicated to this, we have taken care to assign the logo corresponding to the service in an automatic way so that there is a better recognition and accessibility.

The screenshot displays the 'Passwords Hub'. The top navigation bar is identical to the previous figure. The breadcrumb trail is 'Providata » Passwords » Passwords'. The main area shows a list of saved passwords, each in a card with a service logo: Facebook (walim@messenger.com), Spotify (walim@spotify.com), a generic service (walim@sp.com), and a test service (walim@testuellement). Below these, a password for 'test@ieus.fr' is shown with a Twitter logo. The footer contains the same copyright and user information as Figure 35.

Figure 36: Passwords Hub

The screenshot shows the 'Add Password' form within the Providata application. The breadcrumb trail is 'Providata » Passwords » Add Password'. The form contains three input fields: 'Service', 'ID', and 'Password'. A blue 'Submit' button is located below the 'Password' field. The footer of the application shows 'COPYRIGHT © 2020 Providata, All rights Reserved' and 'Actions registered as Admin'.

Figure 37: Web form for submitting passwords

## 8. File storage

This page allows the user to list the different folders he owns within the platform, each folder contains different types of files that the user can download, to display the files corresponding to a specific folder just press the button dedicated to that, and our interface will display the files contained in the folder in question, each file is downloadable in a snapshot via each button associated with it.

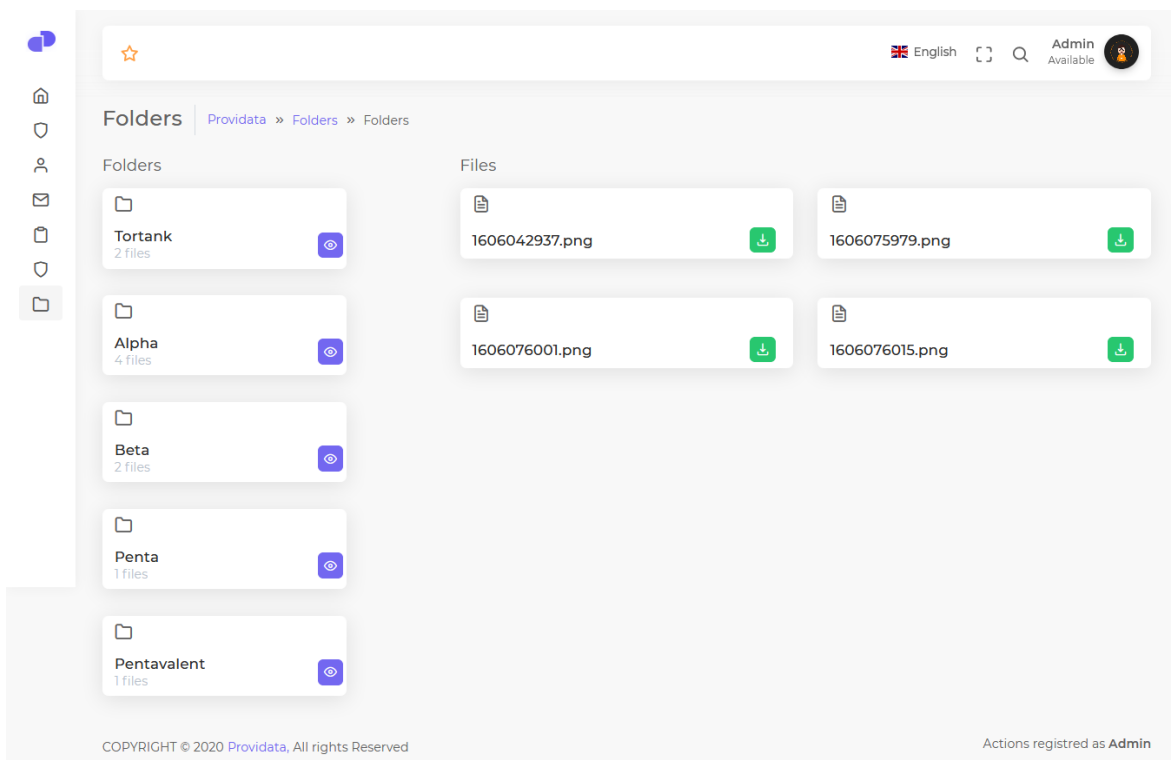


Figure 38: Folders hub

## 9. English Summary:

### Introduction:

I am a student in second year of engineering cycle in an IT engineering school at CESI Graduate School of Engineering, I did this internship as part of my formation in a big startup named Vibrand.

This internship allowed me to reuse and combine technologies that I worked with in the last two years, but with other means, because they improved a trough the different updates.

During this internship, my main mission was to combine two massive technologies on the web domain that are PHP and JavaScript to build a multi architectural ecosystem that relies on a REST API.

### Summary of my internship work:

To run an application, it needs a source from which to extract data to display which is a given database, to design it is necessary to create an ERD Diagram that will identify as needed the relational tables we will need as well as the attributes that will allow us to establish a relational database based on different kinds of relations.

From this schema we will be able to create what are called migrations which are designed to recreate our database automatically, defining the different indexes.

Then will come the creation of models that are classes allowing us to manipulate our tables as objects by creating relationships between them, to create it we will need to seed default data like super administrator account, the controller will allow the API to perform various operations that we will need on our platform and the mobile application.

Then will come our web platform made up of various components with instantaneous loading, it will allow users to perform various operations like retrieving their data or adding notes, passwords or downloading files.

As for the mobile application, it will establish a link with our API and interact with, so our API is the center of operations as well as it allows to the web platform and mobile to interact with each other.

### Conclusion

These two months of internship allowed me to learn more about a lot of new technologies that are in demand on the market while my main mission was to create a solution to allow users to retrieve their data no matter where they are.

## 10. French summary:

### Introduction:

Je suis étudiant en deuxième année de cycle d'ingénieur dans une école d'ingénieurs en informatique au CESI, j'ai fait ce stage dans le cadre de ma formation dans une grande startup nommée Vibrand.

Ce stage m'a permis de réutiliser et de combiner les technologies avec lesquelles j'ai travaillé ces deux dernières années mais avec d'autres moyens, car elles se sont améliorées au fil des différentes mises à jour.

Pendant ce stage, ma mission principale était de combiner deux technologies massives sur le domaine du web que sont PHP et JavaScript pour construire un écosystème multi architectural qui s'appuie sur une API REST.

### Résumé de mon travail durant le stage :

Pour faire fonctionner une application, il faut une source d'où extraire des données à afficher qui est une base de données donnée. Pour la concevoir, il faut créer un MCD qui identifiera au besoin les tables relationnelles dont nous aurons besoin ainsi que les attributs qui nous permettront d'établir une base de données relationnelle basée sur différents types de relations.

À partir de ce schéma, nous pourrons créer ce que l'on appelle des migrations qui sont destinées à recréer automatiquement notre base de données, en définissant les différents index.

Ensuite viendra la création de modèles qui sont des classes nous permettant de manipuler nos tables comme des objets en créant des relations entre elles, pour le créer nous aurons besoin de semer des données par défaut comme le compte du super administrateur, le contrôleur permettra à l'API d'effectuer diverses opérations dont nous aurons besoin sur notre plateforme et l'application mobile.

Ensuite viendra notre plateforme web composée de divers composants à chargement instantané, elle permettra aux utilisateurs d'effectuer diverses opérations comme la récupération de leurs données ou l'ajout de notes, de mots de passe ou le téléchargement de fichiers.

Quant à l'application mobile, elle établira un lien avec notre API et interagira avec elle, de sorte que notre API soit le centre des opérations tout en permettant à la plateforme web et au mobile d'interagir entre eux.

### Conclusion

Ces deux mois de stage m'ont permis d'en apprendre davantage sur un grand nombre de nouvelles technologies qui sont demandées sur le marché, tandis que ma mission

principale était de créer une solution permettant aux utilisateurs de récupérer leurs données où qu'ils se trouvent.

## 11. Internship report:

After about two months of internship, I can conclude that I was able to learn a lot of things, including technologies that we have not yet discussed at CESI, but that in the future could help me in my professional career.

## 12. Conclusion:

This short internship was beneficial for my learning, it allowed me to discover the work in a team of developers and the good practices to adopt.

I can say that this internship has taught me to develop in a more professional way.

## 13. Resources:

- <https://www.linguee.fr/>
- <https://reactnative.dev/>
- <https://www.npmjs.com/>
- <https://medium.com/>
- <https://software-mansion.github.io/react-native-gesture-handler/docs/getting-started.html>
- <https://reactnavigation.org/>
- [https://www.adobe.com/mena\\_en/products/xd.html](https://www.adobe.com/mena_en/products/xd.html)
- <https://stackoverflow.com/users/8592712/therealwalim>
- <https://www.youtube.com/>
- <https://dev.to/skydiver/using-laravel-fortify-to-restore-laravel-ui-functionality-7dc>
- <https://laravel.com/>
- <https://laraveldaily.com/>
- <https://github.com/facebook/react-native>