

Politecnico di Torino  
Department of Automatics and Computer  
Engineering

**Bachelor's Degree in Computer Engineering**



## **Internship Report**

**Full Stack Web Development using React -  
Adhesion Backoffice Project**

**Jeyhun Yagublu S260229**

**Academic Tutor:** Prof. Alfredo Benso

**Company Tutor:** CTO Antonio Giordano - EFFE ERRE SRL

**Company:** EFFE ERRE SRL

**Period:** March 2021 - May 2021

## Contents

<b>1</b>	<b>Internship objectives</b>	<b>3</b>
<b>2</b>	<b>Used Technologies</b>	<b>4</b>
2.1	IDE . . . . .	4
2.2	Version Control . . . . .	5
2.3	React, MaterialUI, Redux, MongoDB . . . . .	6
<b>3</b>	<b>Adhesion Backoffice Front End</b>	<b>7</b>
3.1	First Steps . . . . .	7
3.2	React Hook Form . . . . .	8
3.3	Merchants Page . . . . .	9
3.4	Events Page . . . . .	10
3.5	Merchant Profile Page . . . . .	11
<b>4</b>	<b>Adhesion Backoffice Back end</b>	<b>12</b>
4.1	Introduction . . . . .	12
4.2	Models . . . . .	12
4.3	Authorization and Storage . . . . .	12
4.4	Endpoints . . . . .	13
4.4.1	GET Endpoints . . . . .	13
4.4.2	POST Endpoints . . . . .	14
4.4.3	PUT Endpoint . . . . .	15
4.5	Testing . . . . .	16

## 1 Internship objectives

Primary objective of this internship project was to create a back-office website platform for the already existing "Adhesion" application. Adhesion is an application for business owners and their customers for staying up to date in the latest offers and promotions. Customer subscribes to his/her favourite brand using Adhesion mobile application and gets all the latest promotions and offers directly to his phone by push notifications. Back-office platform is designed for the business owners to manage their shops and promotions. Additionally in future they can get comprehensive statistics about all subscribers and different promotions.

My first task was to learn JavaScript/Typescript programming languages after which I studied React library and I was assigned on a team developing the Adhesion-Backoffice project. During internship my team has developed both back end and responsive front end of the platform from scratch and successfully released it online:

Adhesion Backoffice



## 2 Used Technologies

### 2.1 IDE

We used Webstorm IDE as it was the perfect choice for our work. Webstorm has tons of plugins and it combines editor, terminal, file management, version control and many other useful features. Webstorm was used for developing both back end and front end.

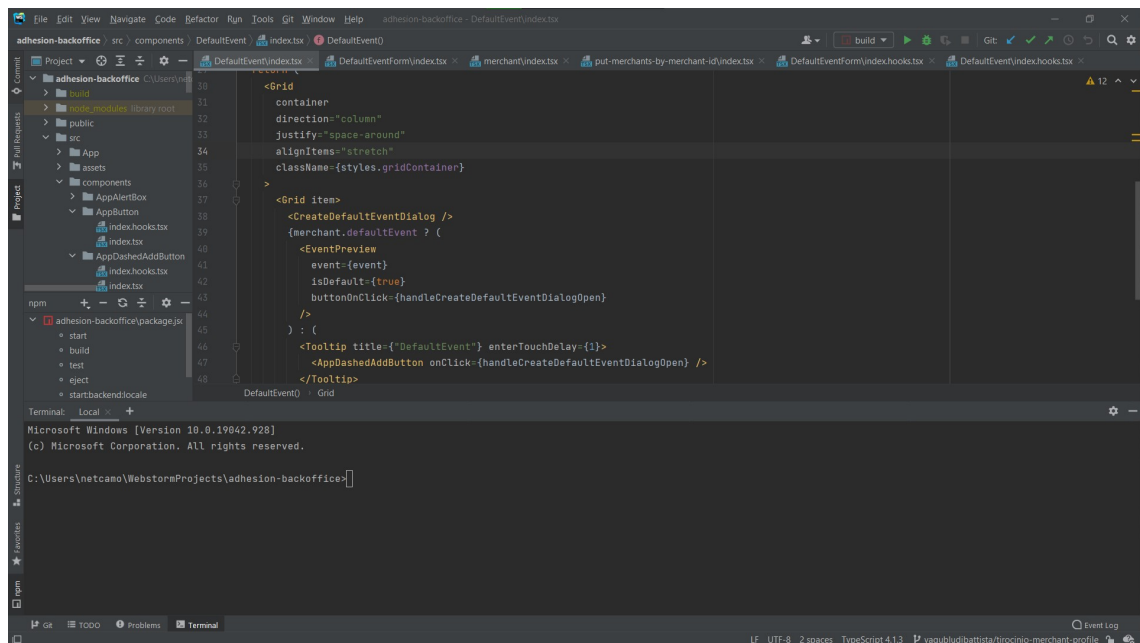


Figure 1: Webstorm IDE

## 2.2 Version Control

Our project was in a shared Github repository as Github has many advanced features that help teams to collaborate and create different parts of application without conflicts. We worked on the same project with different branches and our supervisor was able to check the changes to the project and review them. Webstorm was able to perfectly integrate our Github repository and this combination saved a lot of time and effort.

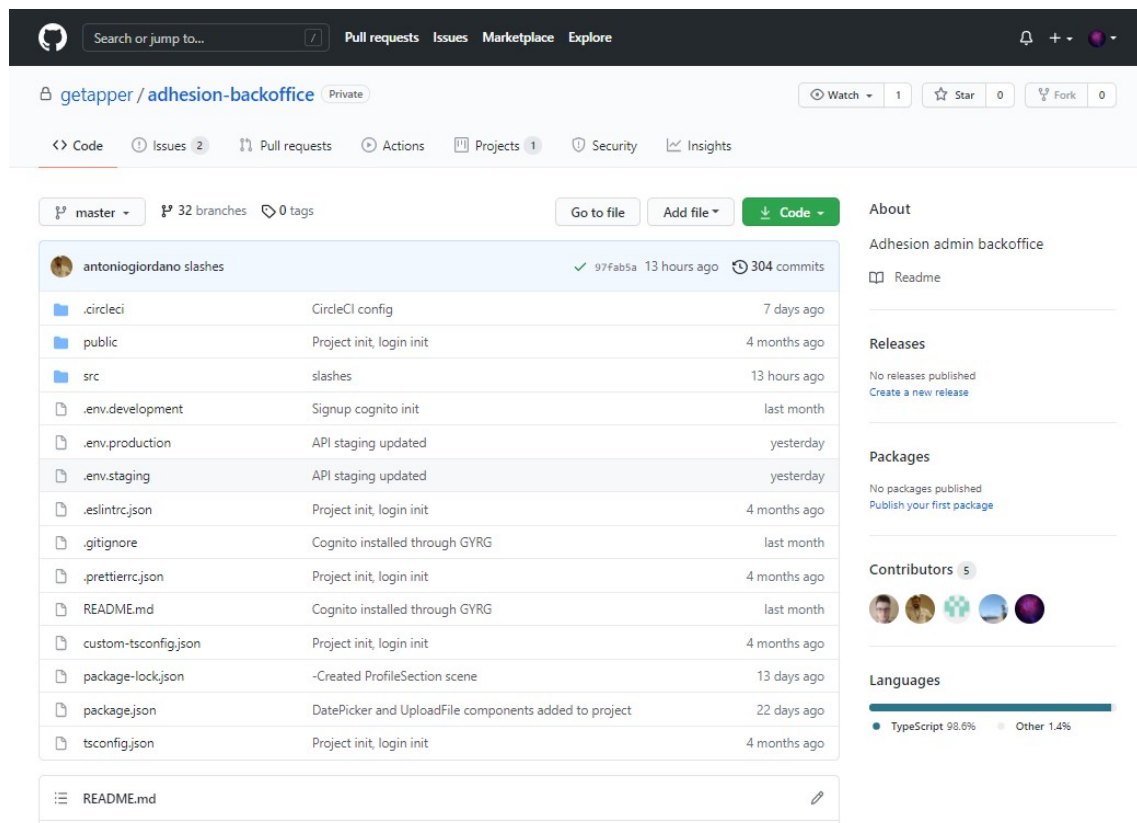


Figure 2: Adhesion-Backoffice Github page

### 2.3 React, MaterialUI, Redux, MongoDB



We wrote the platform using React library. React is an open-source, JavaScript library for building user interfaces or UI components. It is maintained by Facebook and a community of individual developers and companies. It made everything much simpler especially when it's combined with company's own Yeoman generator plugin created by my supervisor - Antonio Giordano.



Material-UI, the React component library based on Google Material Design, allows for faster and easier stylized web development. With basic React framework familiarity, you can build a deliciously material app with Material-UI, and it's almost like cheating. There are tons of ready to use responsive components that makes everything much more easier and faster.



Redux is an open-source JavaScript library for managing application state. It is most commonly used with React library for building user interfaces.



MongoDB is a general purpose, document-based, distributed database built for modern application developers and for the cloud era. No database makes you more productive.

## 3 Adhesion Backoffice Front End

### 3.1 First Steps

After learning Javascript/Typescript, React and MaterialUI basics I have been assigned to Adhesion project. My first task was to create "Landing Page" of Adhesion platform which should be minimalistic and should contain CTA(Call-to-action) button. With this first task not only I have applied what I learned but also I got to know about basics of Routing and how it works in React alongside with responsive design.

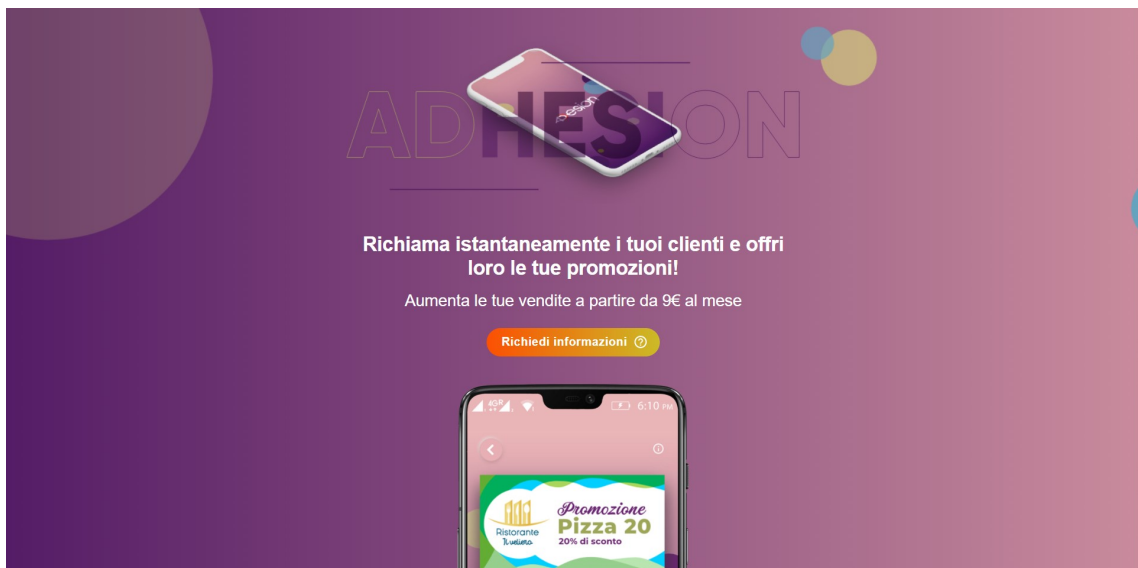


Figure 3: Landing Page

### 3.2 React Hook Form

CTA button takes user to SignUp page which contains registration form where I used React Hook Form to simplify validation and behaviour of my form. React Hook Form is a tiny library without any dependencies which handles errors, actions and validations of your form.

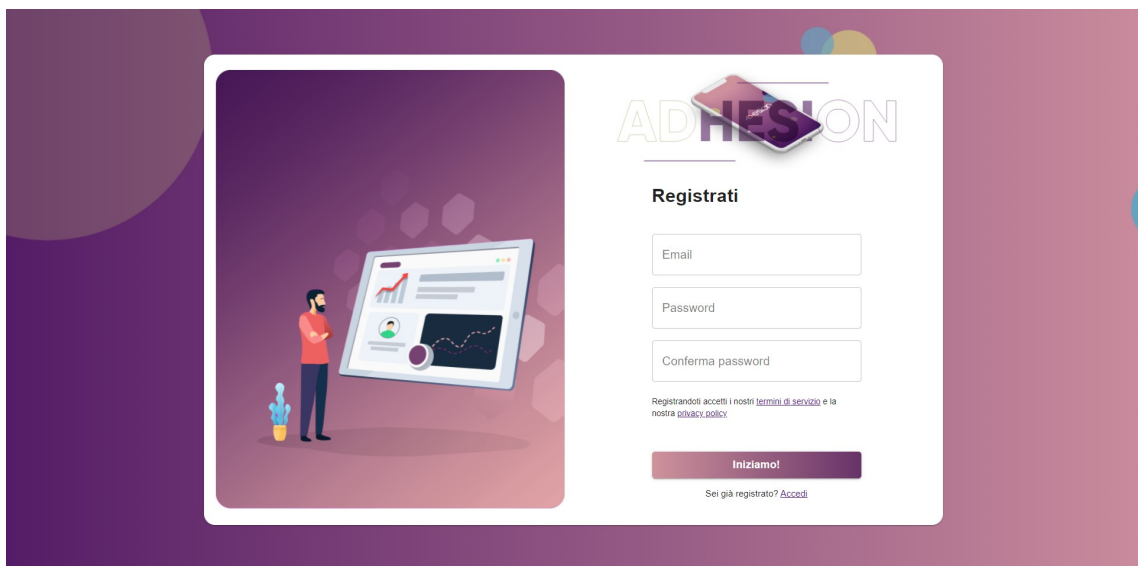


Figure 4: SignUp Page

```
const schema = yup.object().shape({
  email: yup
    .string()
    .required("Inserisci la tua mail")
    .email("Inserisci una mail valida"),
  password: yup
    .string()
    .required("La password è obbligatoria")
    .min(8, "La password deve avere almeno 8 caratteri"),
  passwordConfirmation: yup
    .string()
    .oneOf([yup.ref({ key: "password" })], "Le password non corrispondono"),
});
const { register, handleSubmit, errors } = useForm<{
  email: string;
  password: string;
  passwordConfirmation: string;
}>({ mode, reValidateMode, resolver, context, defaultValues, shouldFocusError, shouldUnregister, criteriaMode: {
  resolver: yupResolver(schema),
}});
const { register: register2, handleSubmit: handleSubmit2 } = useForm<{
  code: string;
}>();
```

Figure 5: React Hook Form takes care of almost everything for us



### 3.3 Merchants Page

Next step was to create the Merchants Overview page where there are MaterialUI Cards designed to show the shops that user owns. Add button opens MaterialUI Dialog to create new Merchant.



Figure 6: Merchants Overview

### 3.4 Events Page

Next step was to create the Events page where all promotions and events related to the shop are shown. Add button opens MaterialUI dialog for creating new Event. Above all, very unique feature of this dialog is that it shows a preview of event notification while you fill the form so, shop owner can see real time all the changes that he does:

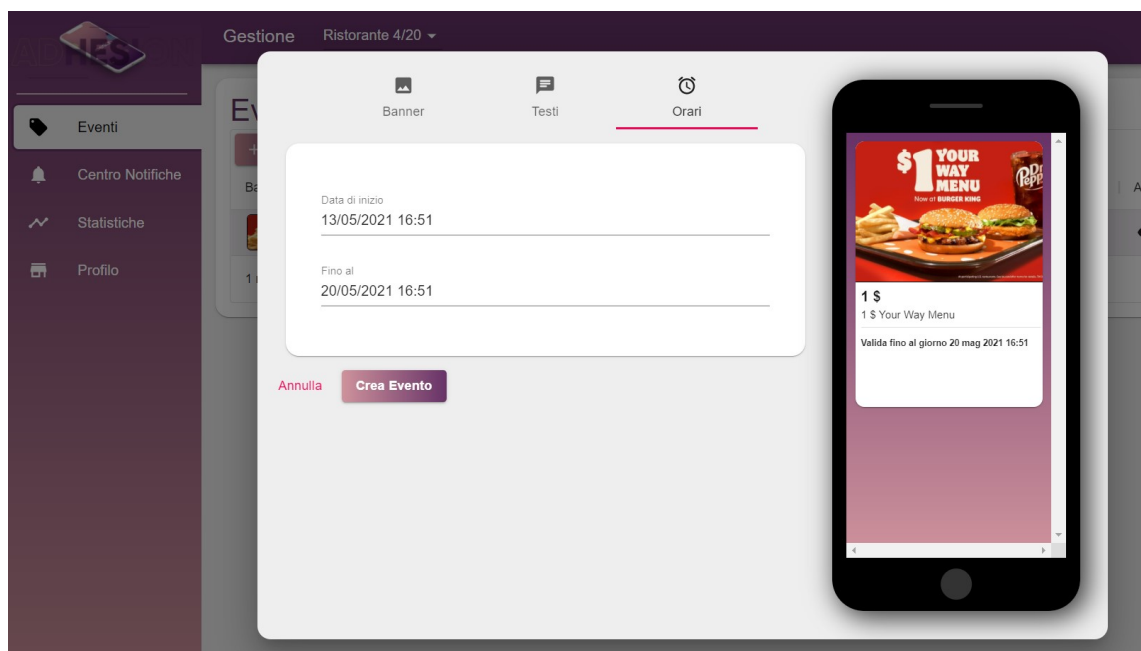


Figure 7: Events Page

In order to achieve this we have created an additional component called Event Preview which we have been reused in different components throughout the project.

### 3.5 Merchant Profile Page

We created this page for managing the different shops. Here shop owner can change shop's logo and its Default Promotion. Default Promotion is a special offer which is available to customers only first time that they subscribe and have special expiry time. In future this page will also create file which will contain unique QR code for subscribing to the shop and also the description of Default Promotion.

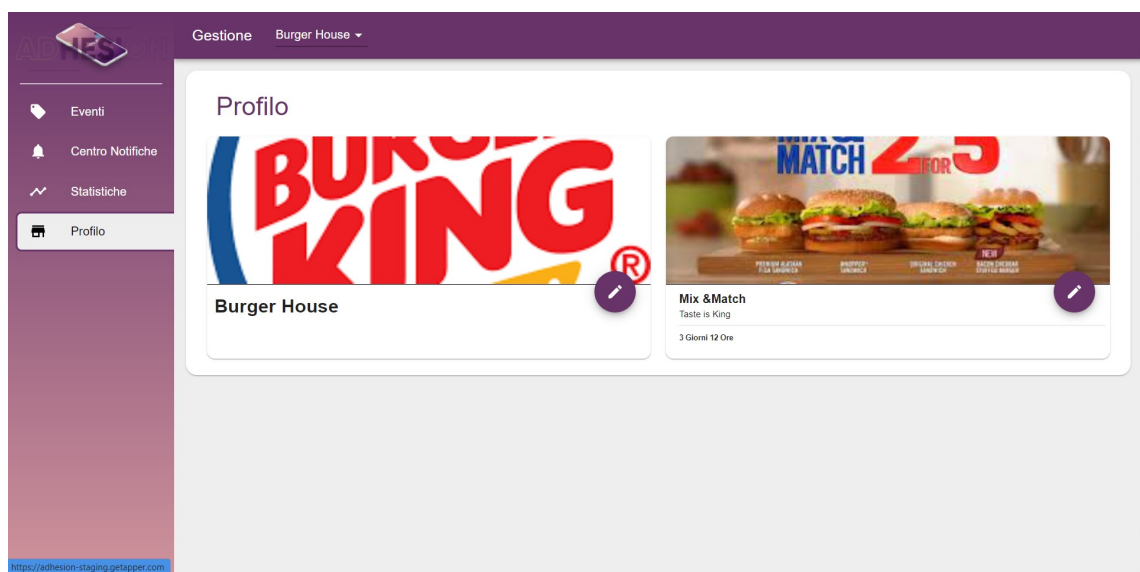


Figure 8: Profile Page

## 4 Adhesion Backoffice Back end

### 4.1 Introduction

Back end part of Adhesion project was written in Typescript in Node.js runtime environment as APIs that respond to the request. Database is managed by MongoDB. Since this project would be released using Amazon Web Services we also used Serverless API - open source project that has been developed by our company using Serverless.com Application Framework.

### 4.2 Models

We used only 3 main Models in BE which are following:

- User - this is the basic User model which contains only username and id.
- Merchant - this is the model of Shop which contains name, Shop logo, Default-Event interface ,and User ids that owns this shop.
- Event - this is the model for the promotions. It contains all the information about the Promotion such as name, content, expire date and Merchant's id.

### 4.3 Authorization and Storage

Authorization is done by AWS Cognito Service. Amazon Cognito User Pools provide a secure user directory that scales to hundreds of millions of users. As a fully managed service, User Pools are easy to set up without any worries about standing up server infrastructure.

Storage of all the images are done by Amazon Simple Storage Service (Amazon S3) which is very simple to use. It is an object storage service that offers industry-leading scalability, data availability, security, and performance.

## 4.4 Endpoints

Endpoints are created and managed by the Restlessness API. It has a graphic user interface for creating and managing services, endpoints and connecting them to Authorization services and database management services which also has in-built swagger.

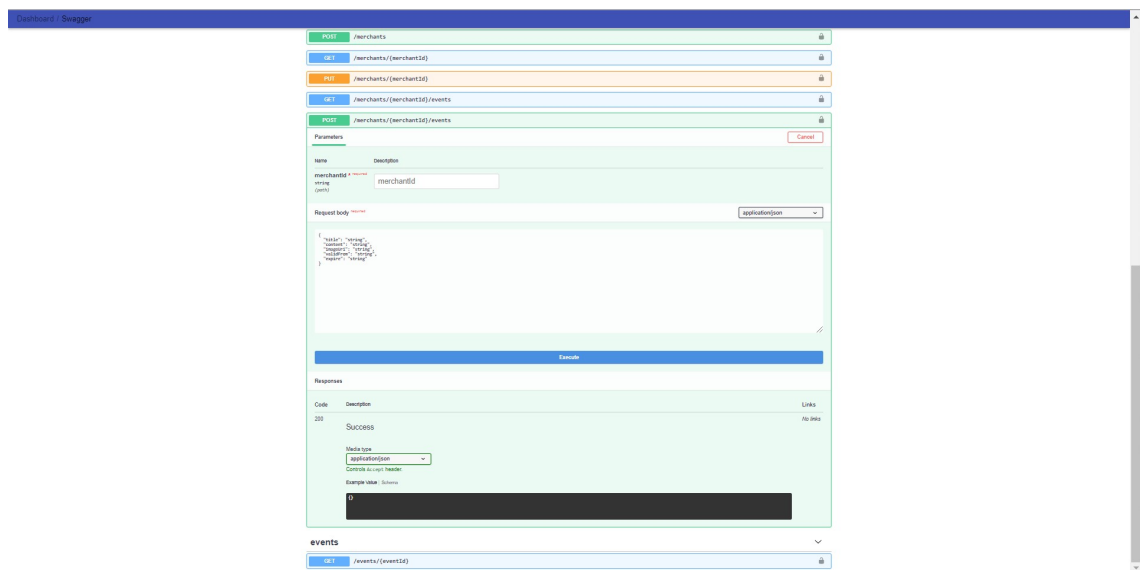


Figure 9: Restlessness GUI

### 4.4.1 GET Endpoints

GET Endpoints first check for the Authorization and then retrieves data for the related request. Restlessness creates all the basic structure for the endpoint and all left is to specify interfaces, validation of request and to develop the logical part of endpoint.

#### 4.4.2 POST Endpoints

POST Endpoints create new entry in database such as new promotion or user or merchant. For image storage special process has been decided. Front end application uploads image and converts it to base64 image string which is forwarded to Back end application. Back end application converts string to image file and uploads it to S3 bucket while it saves the address of image from the S3 bucket to the database. Front end Application then retrieves the link from the back end application and loads it to website real time.

```
let imageUuid;
let s3;
let imageBase64: string;
let imageBuffer;

if (payload.imageUri) {
  imageUuid = uuid();
  if (process.env.ENV_NAME !== "test") {
    s3 = useS3();
    imageBase64 = payload.imageUri.split(";base64,").pop();
    imageBuffer = Buffer.from(imageBase64, encoding: "base64");
    await s3.upload(
      getItemsBucket(),
      `${merchant._id}/${imageUuid}.png`,
      imageBuffer
    );
  }
  await merch.patch( fields: {
    imageUri: `${imageUuid}.png`,
  });
}
```

Figure 10: Storage process of Image

#### 4.4.3 PUT Endpoint

This was the most complex part of the Back end Application since there were many complex and optional parameters of the request. For managing the optional parameters I used Yup JavaScript schema builder for value parsing and validation:

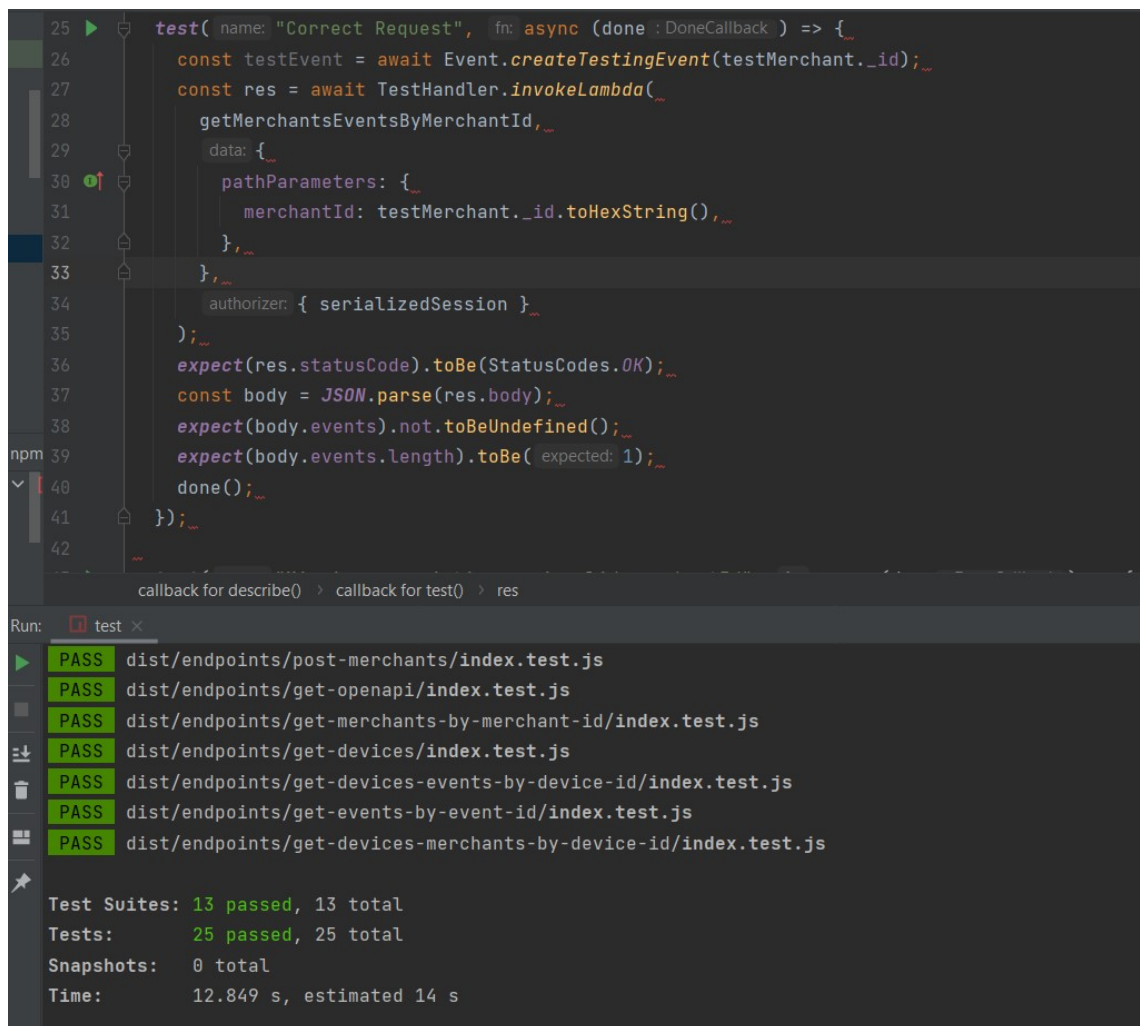
```
const payloadValidations = (): YupShapeByInterface<Payload> => ({
  defaultEvent: yup
    .object()
    .shape({
      title: yup.string().trim().min(3).max(30).optional(),
      content: yup.string().trim().min(0).max(200).optional(),
      imageUri: yup.string(),
      expiringMinutes: yup
        .number()
        .positive("Il scadenza è un parametro positivo")
        .integer(),
    })
    .optional(),
  imageUri: yup.string().optional(),
});
```

Figure 11: Yup validation

Expiration time of Default Promotion was created by user inserting in 3 different fields number of days, hours and minutes, however, in the back end it was saved as only one field called expiring Minutes so at every request the input from user is converted to total minutes.

## 4.5 Testing

Testing was created and done by Restlessness API which again creates all the basic structure for every Endpoint. This way developer can test his application in all possible requests which includes Authorized/Unauthorized sessions, wrong/correct request, Empty requests etc. and gets comprehensive feedback through terminal.



```
25 test( name: "Correct Request", fn: async (done : DoneCallback ) => {
26   const testEvent = await Event.createTestingEvent(testMerchant._id);
27   const res = await TestHandler.invokeLambda(
28     getMerchantsEventsByMerchantId,
29     {
30       data: {
31         pathParameters: {
32           merchantId: testMerchant._id.toHexString(),
33         },
34       },
35       authorizer: { serializedSession }
36     );
37   expect(res.statusCode).toBe(StatusCodes.OK);
38   const body = JSON.parse(res.body);
39   expect(body.events).not.toBeUndefined();
40   expect(body.events.length).toBe( expected: 1 );
41   done();
42 });
```

Run: test

- PASS dist/endpoints/post-merchants/index.test.js
- PASS dist/endpoints/get-openapi/index.test.js
- PASS dist/endpoints/get-merchants-by-merchant-id/index.test.js
- PASS dist/endpoints/get-devices/index.test.js
- PASS dist/endpoints/get-devices-events-by-device-id/index.test.js
- PASS dist/endpoints/get-events-by-event-id/index.test.js
- PASS dist/endpoints/get-devices-merchants-by-device-id/index.test.js

Test Suites: 13 passed, 13 total  
Tests: 25 passed, 25 total  
Snapshots: 0 total  
Time: 12.849 s, estimated 14 s

Figure 12: Testing with Restlessness