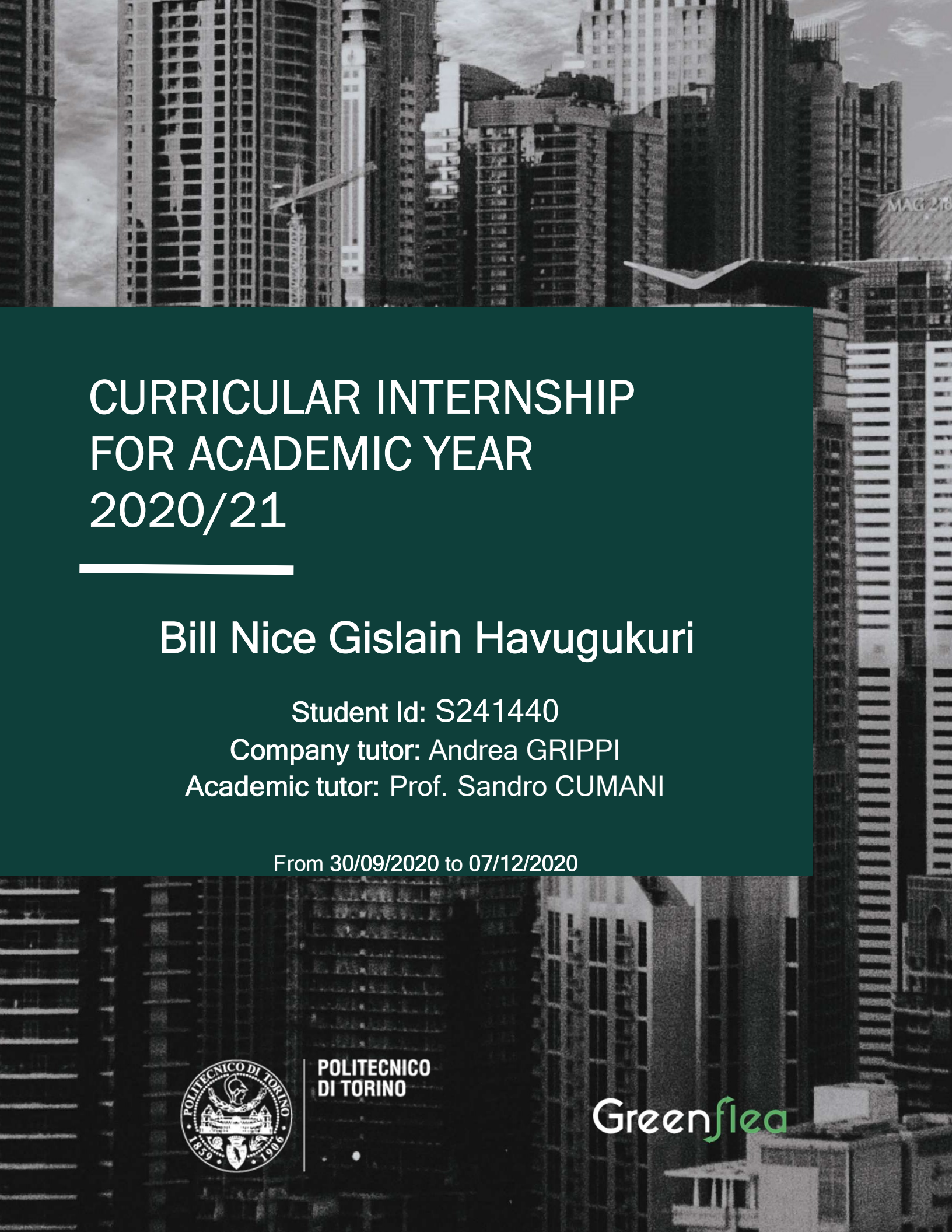


CURRICULAR INTERNSHIP FOR ACADEMIC YEAR 2020/21

Bill Nice Gislain Havugukuri

Student Id: S241440
Company tutor: Andrea GRIPPI
Academic tutor: Prof. Sandro CUMANI

From **30/09/2020** to **07/12/2020**

POLITECNICO DI TORINO

Greenflea

Contents

Introduction

Incorporated to the Computer engineering program, the professional training (known as Tirocinio)'s aim is to introduce to Bachelor students the main skills of a professional engineer.

The Polytechnic of Turin (Known as POLITECNICO DI TORINO) demands to from its student to fulfil 10 credits (250 hours) professional internship as part of their curriculum. It is very important for young engineers to know not only the theoretical part of its study but also the practical one.

I did my internship at **Greenflea**, a cutting edge digital eco-friendly marketplace for buying and selling green products between users and shops on the platforms and contributing to the protection of our planet by planting trees.

During this internship I had the opportunity to work on the marketplace mobile platforms (IOS/Android) by adding new features such as SCA complaint payments, a user review and rating system, FCM notifications, services (internet connection, HMS, and GMS) availability check, and enhancing others such as app language localization(en_it), navigation bar, optimizing and modularizing part of the code base.

The internship was from **30/September/2020** until **07/DECEMBER/2020**, due to COVID-19 pandemic the internship took place **remotely** for the most part using tools like (**Google Meet, Slack and** GitHub) and I worked an average of around 25 hours per week reporting directly to the company's CEO for supervision, mentoring and clarifications of any doubts I would have on a given task.

In the framework of this internship, I was introduced to cross platform mobile application development technologies such as **Google's Dart language and flutter UI toolkit**, **Firebase** firestore, cloud messaging, Analytics, cloud functions with **NodeJS** , **GitHub**, **Stripe** payment API, Zeplin for creating and sharing UI mocks and Play store private beta testing for the app.

Features, I worked on

During the internship, I learned the **Dart language**, **flutter framework**, **firebase cloud functions** (**Serverless** cloud computing) with **NodeJS** and **flutter packages**. With those newly acquired skills, I used them to work on many aspects of the mobile application, implementing new features and maintaining old ones.

Some of the more noticeable tasks were:

- Services availability check (internet connection, Huawei Mobile Services, and Google Mobile Services).
- Firebase Cloud Messaging notifications for chatrooms and product subscriptions.
- A new payment system for the platform with SCA (Strong customer authentication) conformity using the stripe API.
- Customer experience review and Sellers rating system.

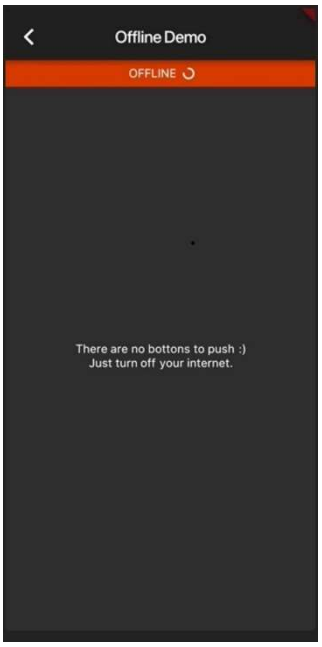
Tools used

- **Slack, Google Meet** and **GitHub** for collaborative remote working and team communications.
- **Android Studio** and **Visual Code** Integrated Development Environments
- **Zeplin** for Sharing UI/UX Mockups with the graphics design team.
- **Stripe** for managing the Stripe **API** authentication keys and monitoring payments during the development and debugging phase.
- **Firebase** using its tools like Cloud Firestore (a cross-platform document-oriented database program), Cloud Messaging (allow the developer to send notifications to all the users of his application meeting a certain criteria) and Cloud Functions (a serverless approach to google cloud computing).
- And finally, **Stack Overflow** which is a question-and-answer site for professional and enthusiast programmers.

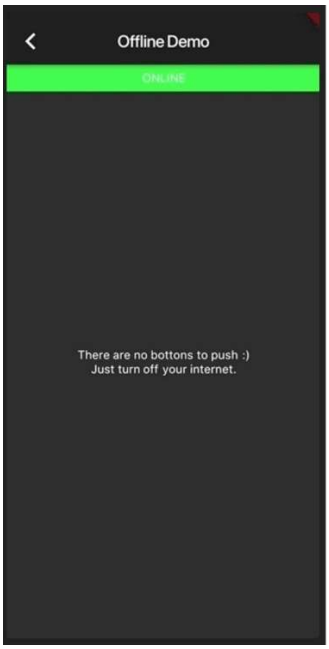
Main Features Description

✓ Services availability check (Internet connection, HMS, GMS)

Service Unavailable:



Service Available:



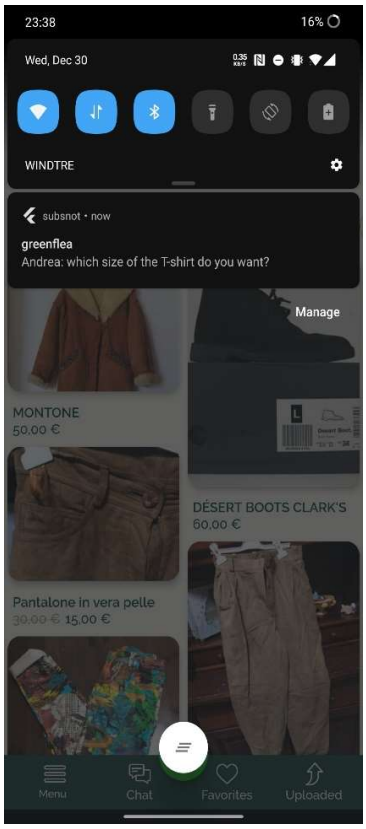
Purpose:
The objective of the task was being able to perform a check every time the app start running to make sure that critically required services are available and react accordingly, since their unavailability could potentially crash the app.

Implementation:
Integrating a flutter package into the app and using it to perform a check.

Code sample:

```
@override
initState() {
  super.initState();
  isLoading = false;
  pageController = PageController();
  //continous checking internet connection
  //a subscription would be needed
  netStatus = widget.internetStatus;
  Connectivity().onConnectivityChanged.listen(
    (ConnectivityResult result) async {
      //print("\nInternet status changed");
      netStatus = await checkInternetConnection();
    },
  );
  //standard
  if (netStatus == null) netStatus = true;
  if (netStatus) {
    trySilentLogin();
  }
}
```

✓ Firebase Cloud Messaging notifications for chatrooms and product subscriptions



Purpose:

The marketplace has chatrooms that a customer can use to ask the seller more information about the product they want to buy, I used the FCM tokens generated by each participant device to send FCM notifications whenever a user or a seller receives a new message.

Implementation:

Whenever a user sends a message:
On the client, retrieve the targeted device token and make a cloud function call to the server sending it the targeted device token and message payload.
On the server (cloud function), send the notification the device using the firebase FCM API.
On the other client, setup a cloud messaging subscription that listens for arriving notifications and handles them in depending on the devices current state (onMessage, onResume, onLaunch).

Source Code:

```
firebaseMessaging.configure(
  onMessage: (Map<String, dynamic> message) async {
    if (! ModalRoute.of (context).isCurrent) {
 showDialog(
   context: context,
   builder: (context) => GreenfleaPopup(
      title: message['notification']['title'],
      body: Column(children: <Widget>[
      Text(message['notification']['body'],
       style:TextStyle(fontSize: 16),),
 FlatButton(child: Text('View',
  style: TextStyle(color:Theme.of(context).primaryColor),
                  ),
   onPressed: () async {
     routingToMessagedChatroomPage(context, message, user);
                  },
               ),
             ],
           )),
      );
    }
  },
  onLaunch: (Map<String, dynamic> message) async {
    routingToMessagedChatroomPage(context, message, user);
  },
  onResume: (Map<String, dynamic> message) async {
    routingToMessagedChatroomPage(context, message, user);
  },
);
```

Android Studio: flutter

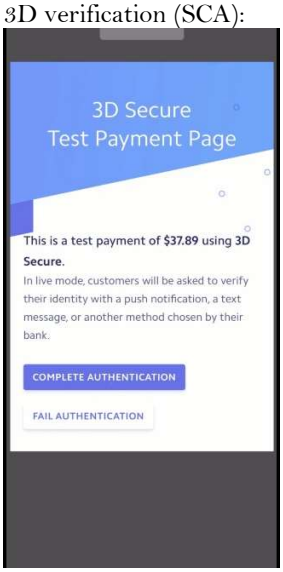
```
exports.sendNotificationToDevice = functions.https.onCall(
 async (data: any, context: any) => {
  if (context.auth.uid === null) {
   return { httpResponse: 401 };
  }
  try {
   const fcm = admin.messaging();
   const product: SoldProduct = data.product;
   const token: string = data.fcmToken;
   const message: string = data.body;
   const userName: string = data.safeUserName;
   const chatroomId: string = data.chatroomId;

   console.log(`the targeted token is : ${token}`);
   // @ts-ignore
   const payload: admin.messaging.MessagingPayload = {
     ...etc
   };
   const response = await fcm.sendToDevice(token, payload);
   console.log(response);
   console.log("Notification sent successfully");
   return { httpResponse: 200 };
  } catch (err) {
   console.log(err);
   console.log("Notification not sent");
   return { httpResponse: 500, message: err };
  }
});
```

Visual Code: NodeJS =>Cloud function

✓ A new payment system for the platform with SCA (Strong customer authentication) conformity using the stripe API

Payment Page:

3D verification (SCA):



Purpose:
Due to a new law passed in the **EU**. Starting with 2021 all online transaction will have to require a **3D verification(SCA)** for an increased security against credit cards fraud. So my task was to create a new payment system able to handle 3d verified credit cards since non SCA conformed card will no longer be processed after 2020.

Implementation:
On the client, the app uses a form to acquire payment details and pass them to a cloud function call.
On the server (cloud function), interacts with the stripe API and creates a payment_intent.
On the client, it then retrieves a reduced version of the **payment_intent** using a special **secret key** and processes the payment after handling the 3D verification.
This ensures that all critical operations like setting the product price are handled securely on Greenflea server.

Source Code:

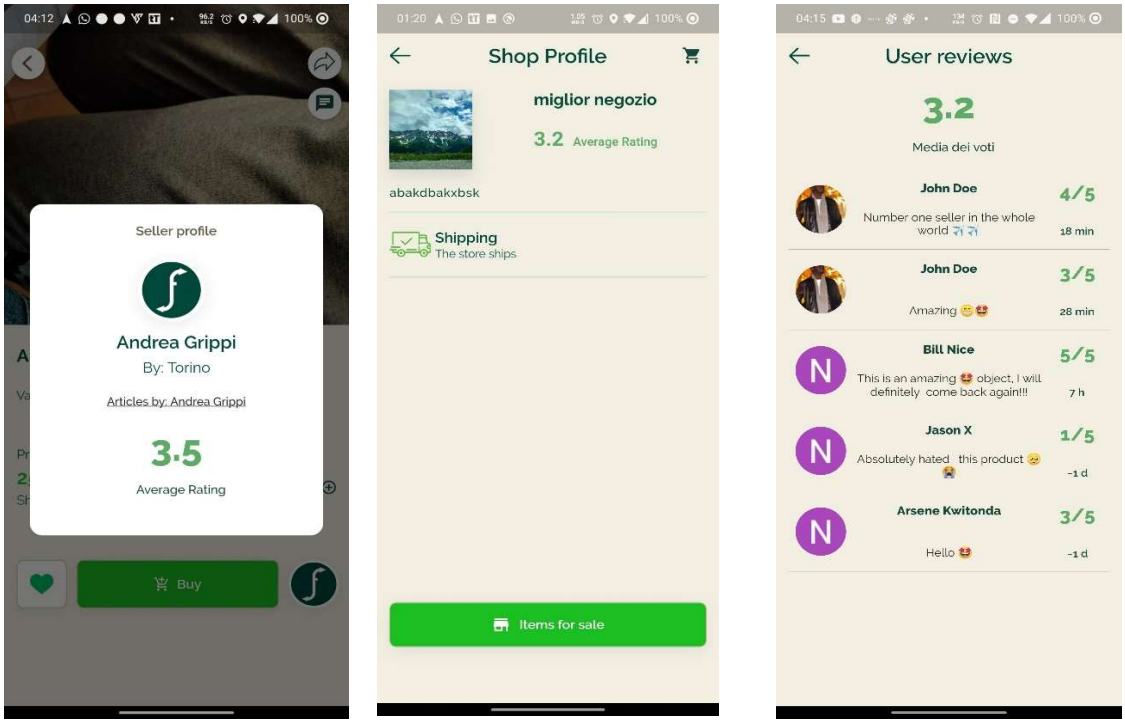
```
…etc

Map<String, dynamic> paymentIntentRes = await
createPaymentIntent(stripeAccountId,stripeCard,customerEmail
,context,storeId,order,user,itemName,transactionDetails);

String receiptUrl = "";
String orderId = "";
String clientSecret = paymentIntentRes['client_secret'];
String paymentMethodId =paymentIntentRes['payment_method'];
String status = paymentIntentRes['status'];

if (status == 'requires_action' ||
    status == "requires_confirmation") //3D secure is enable
in this card
  paymentIntentRes =
      await confirmPayment3DSecure(clientSecret,
paymentMethodId, context);

if (paymentIntentRes['status'] != 'succeeded') {
  showAlertDialog(
      context,
      "Error", payment failed");
  await increaseQuantityOnCancel(order);
}

if (paymentIntentRes['status'] == 'succeeded') {
  print(">>>>>>>>>>>>>>>>>>Payment Succeeded");
  print(">>>>>>>>>>>>>>>>>>" + storeId);

}

…etc
```

Android Studio: dart

```
exports.chargeCardWithStripeTest = functions_stripe.https.onCall(async (data, co
ntext) => {
  if (context.auth.uid === null) {return { httpResponse: 401 }; }
  let clientSecret;
  let totalAmount = order.totalNetAmount;
  const item_name = data.item_name;
  let feeableAmount;
  if (data.order.subtotal) {feeableAmount = data.order.subtotal; }
  else {  feeableAmount = data.order.totalNetAmount;  }
  if (data.shipping_costs) { totalAmount += data.shipping_costs;}
  const stripeAccount = data["stripeAccountId"];
  const payment_method_id = data["payment_method_id"];
  try {
    const greenfleaFee = ….;
    let stripeFee = …..; //PAYMENTS INTRA EU
    const account = await stripe_test.accounts.retrieve(stripeAccount);
    const fee = feeableAmount * (greenfleaFee - stripeFee);
    const vatAmount = fee * 0.22;
    const paymentIntent = await stripe_test.paymentIntents.create({
      amount: totalAmount,
      payment_method_types: ["card"],
      payment_method: `${payment_method_id}`,
      application_fee_amount: Math.floor(fee + vatAmount),
    }, {
      stripeAccount: stripeAccount,
    });
    clientSecret = paymentIntent.client_secret;
  }
  catch (error) {
    console.log(error);
    return { httpResponse: 500, message: "failed paymentIntent create " };
  }
  return clientSecret;
});
```

Visual Code: NodeJS =>Cloud function

✓ Customer experience review and Sellers rating system



Purpose:

The objective of the task was giving to the buyers the ability to review the seller after buying products from them, so that future customers can check the reliability of a seller before buying from them.

Implementation:

- ✓ O**n the client**, after a user receives the purchased item, they can review the seller giving him a rating between 1-5, after which the review data and product details are sent to a cloud function.
- ✓ **On the server** (cloud function), receives the data and creates a subcollection in the seller's collection(**Firestore Database**) and saves that review's details.
- ✓ O**n the client**, when a user navigates to another user or store's profile, they find the seller's current average rating.
  And if they click on the rating button, a list of all the reviews given to seller are retrieved from the database (Firebase **Firestore**) using **a Stream Builder widget** that is updated for every new review in real time.

Conclusion

Overall, I had a great experience, I feel extremely fortunate that I had the opportunity to do my internship at Greenflea. I would like to thank my company tutor, colleagues at the company (Andrea Grippi, Giulia Scomparin and Laura Lodi) and my academic tutor (Prof. Sandro CUMANI) for their help, support, and mentoring throughout the internship.

Bill Nice G. Havugukuri
31/12/2020