FINAL PROJECT

Title: **External services integration for internal usage of an organization, using Google Apps Script**

Degree: Technical Engineering in Telecommunications, speciality of Telematics

Author: Karina Díaz Arrunategui

Director: Jesús Alcober Segura

Date: 25/09/2015

Title: External services integration for internal usage of an organization, using Google Apps Script
Author: Karina Díaz Arrunategui
Director: Jesús Alcober Segura
Date: 25/09/2015

Overview:

In this project we have tried to pick an already existing software solution and give it new features as well as integrating it with third party services, maintaining the freeware philosophy it had since its beginning.

The main goal, anyway, is not to further extend the development of the application's core (the spreadsheet) but to boost its ability to communicate with other apps and services and to unify all the business intelligence and let the data flow.

Most businesses run multiple applications (CRM, ERP, calendars...) that run independently and therefore have to lead with data type incompatibilities, loss of data and in consequence a decrease in economical profits.

Integrating our services has many advantages, such as said flow of information between programs and people, consolidating our data storage and getting rid of duplicated resources or avoiding slow data allocation and access.

The result of our investigation and development is an improved business opportunity flow whose stages are fully covered in a managing site where the user can check incoming opportunities, update economic data, keep track of his daily tasks and maintenance duties and also have a quick access to some useful reports which are updated in real time as sales go by. This system is fully builded upon free resources, which means that with a simple setup the end user can benefit of our solution with no economic expense while saving also time and headaches.

Furthermore, this project in his «final» state still shows a huge potential for connecting to many other free services in the future, such as financial services, newsletter and marketing tools, booking engines, social media, business reviewing websites, invoicing apps... its scalability and low maintenance make it a good starting point in case someone wants to extend it or customize it even more.

Título: Integración de servicios externos para el uso interno de una organización, usando Google Apps Script
Autora: Karina Díaz Arrunategui
Director: Jesús Alcober Segura
Fecha: 25/09/2015

Resumen:

Para este proyecto hemos escogido una solución de software ya existente y le hemos dado nuevas características, así como su integración con servicios de terceros, manteniendo siempre la filosofía de software gratuito que tenía desde sus inicios.

Sin embargo, el objetivo principal no es extender aún más el desarrollo del núcleo de la aplicación (la spreadsheet), sino aumentar su capacidad de comunicarse con otras aplicaciones y servicios, para unificar toda la inteligencia de negocio y facilitar el flujo de datos.

La mayoría de los negocios integran múltiples aplicaciones (CRM, ERP, calendarios...) que se ejecutan de forma independiente y por lo tanto tienen que lidiar con tipos de datos incompatibles, pérdida de datos y, en consecuencia, una disminución en los beneficios económicos.

La integración de nuestros servicios tiene muchas ventajas, como el ya dicho flujo de oportunidades de negocio cuyas etapas están totalmente cubiertas en un site de gestión en el que el usuario puede comprobar oportunidades entrantes, actualizar los datos económicos, realizar un seguimiento de sus tareas diarias y de mantenimiento y también tener un acceso rápido a algunos informes útiles que se actualizan en tiempo real mientras las ventas se van produciendo. Este sistema está totalmente construido sobre recursos gratuitos, lo que significa que con una configuración sencilla el usuario final puede beneficiarse de nuestra solución sin ningún gasto económico ahorrando así tiempo y dolores de cabeza.

Por otra parte, este proyecto en su estado "final" sigue mostrando un gran potencial para la conexión a otros muchos servicios gratuitos en el futuro, como los servicios financieros, boletines de noticias para clientes y herramientas de marketing, motores de reservas, medios sociales, páginas web de opiniones sobre alojamientos, aplicaciones de facturación... Su escalabilidad y bajo mantenimiento lo convierten en un buen punto de partida en caso de que alguien quiera extender o personalizar aún más la implementación.

INDEX

1. INTRODUCTION

1.1 CONTEXT

WHAT WAS ALREADY DONE

This research project has its starting point right at the end of a related project titled “Resources management using Google Apps Script for internal consumption of a business”. That first research explored the possibility of using a Google Sheet as some sort of CRM tool for a holiday rental flat, where all the info of our customers as well as the flat management events, chores and financial information was unified and automated.

Some of the accomplishments of that project include Google Calendar and Gmail automation, employees planning and notification, economic reporting and historic data filing.

The philosophy of that proposal was very important, as it had to be something fully based upon free software, easy to use and free or very cheap to maintain.

THE NEW NEEDS OF THE USER

Following the above explained premises, some new needs for our user where introduced at the beginning of our research, mostly based on the goal of integrating and communicating our application with third party apps and services:

- Possibility of using Zapier, a hub-app with lots of pre made integrations with many free and premium apps
- The need for using Insightly -a cloud based CRM- to store our business intelligence as well as tracking our sales pipelines and tasks.
- The need for some improved and more visual economic reports, something more similar to the ones Insightly offers.
- The possibility of building a site or web interface to access all of the data in one endpoint
- The convenience of protecting the sheet in some manner, in order to avoid involuntary corruption of the format or breaking the script
- An automated flow, or some interface controls so the integration Insightly-Google Sheet can be done repeatedly in a timely manner or by request of the user.
- Improving of the Settings section which already existed, to have a fully translated tool that our customer can use either in English, Catalan or Spanish.
- Implementing some solution so one can always have notion of the integration state of the sheet (updated/not updated) and of every opportunity stage.

REASONS FOR INTEGRATING WITH THIRD PARTY SERVICES

First of all, a little background on the integration concept. Enterprise Application Integration -or EAI- was coined as technical concept in the early 2000's, although it responds to a much older issue. We could say that EAI is a kind of approach when dealing with the need for interoperability
between often numerous and very different systems that compound a standard enterprise infrastructure.

These architectures tend to depend on a lot of different systems and applications to carry on with their daily activities. Those separate systems can be developed in-house or licensed from a vendor, and they may have very different purposes: managing their supply chain, customer relationships, employee information, and business logic. This modularization is sometimes desirable because theory says that breaking big business areas and needs in smaller chores and functionalities usually makes them easier to be handled and evolved.

But that modularization also leads to problems, or in the least case challenges, when it's time to make them "talk" to each other via user interfaces, share their specifically formatted data or scale them. That is where Enterprise Application Integration finds its value.

In the past -before the EAI approach- the common solution was building specific components to connect points in a 1 -1 relationship. That is; your employee managing tool can send economic data to your ERP after payroll is set, but that info ain't arriving to the software where you calculate your production cost in order to add it to the price of the items you sell. Therefore this model can work in very simple scenarios with just two or three systems to connect… don't forget that those customized components will need to be maintained and scaled too.

What EAI proposes for this problem is to treat integration as system task -just like any other task your system does- rather than an ungovernable bunch of individual connections. Adapters are bundled together for connectivity, data transformation engines convert the data to the desired format for every system, etc.

Two typical and very popular elements of today's integration architectures are the API's and the ESB (Enterprise Service Bus).

An enterprise service bus (ESB) is a software architecture for middleware that provides fundamental services for more complex architecture: a mechanism that manages access to applications and services (especially legacy versions) to present a single, simple, and consistent interface to end-users via Web- or forms-based client-side front ends.

Meanwhile, an API (Application programming interface) it's a software-to-software interface that allows for separate parties to talk to each other without any previous user knowledge or intervention.

So we could say the main difference between them is that, while the API is like some kind of “window” where a service can access to make questions, write or delete data, a Service Bus is some central resource that’s not only able to receive all these queries, writings and order, but can also send messages, manage queues, balance server load etc.

For our particular purpose, clearly the API was the most fitting way of integrating our apps.

1.2 FUNCTIONAL REQUIREMENTS

A DETAILED LIST OF ALL THE FUNCTIONAL REQUIREMENTS

a. Management site

There must exist some kind of site or user interface so the manager can access and maintain the data in the sheet without being capable of altering the sheet's format or affecting the script in no manner. That interface should be therefore protected from its user.

b. Opportunities integration

For every new opportunity created in Insightly, a new row must be created as well in the management sheet. Any further update or change in the opportunity's details must be synchronized too. Fields relative to dates, customer data and stages must be edited ONLY in insightly and not in the sheet.

c. Interact with the sheet trough forms

There must exist some kind of form or dialog in order to interact with the sheet without opening it and accessing it directly. The manager needs to have the ability of making changes to only a specific set of fields and shall not be capable of affecting the others.

d. Report integration

Economic reports must also be available to be seen in the site, as it will be the center of intelligence for the manager.

1.3 TECHNICAL DETAILS

INVESTIGATION

The first integration approach was implemented through a Google Form, Zappier and the sheet.

This form was embedded in a Wordpress site which simulated a holiday rental web page. Customers could enter their booking data, that with the help of Zapier was injected in the sheet. This approach was discarded because one of the requirements is that the manager himself must enter the booking in Insightly and this option didn't consider the CRM.

IMPLEMENTATION

After further investigations, the final model is a multi system solution based in the integration of Google Sheets, Google Sites, JotForms, Cyfe and Insightly. Zapier was discarded in the last place because it was finally possible to integrate with Insightly's API using a script.

The resulting cycle is the following:

1. Opportunities are created in Insightly
2. The Google Sheet's script queries Insightly's API every minute to fetch opportunities' info in the form of JSON objects, and inserts them in the sheet, which is embedded in the Google Site
3. The JotForm form embedded in the Google dumps the user responses in a specific sheet. This sheet has another script bound to it that checks if the opportunities exist in the primary sheets and pushes the changes to them.
4. The Cyfe Dashboard is integrated with out primary sheets. It retrieves economic data and shows it as custom reports and graphs that are also embedded in the Google Site.

TOOLS

GOOGLE APPS SCRIPTS is a Javascript-based scripting language, not objected oriented. Its most important features are that it runs on the cloud and not on the developer/user's machine, it can be used for free and it is programmed in a cloud IDE developed by Google.
It is a clean, fast and easy to learn language but it also has a few disadvantages as for example running time and execution limitations, very rapidly evolving classes which often go deprecated or are substituted, and a community of developers much more small than other languages such

as Java or C# may have. This specially makes it more difficult to find support or examples when facing programming issues.

This language has been used in this project to develop the script in the main sheet, which calls to the Insightly API and files old bookings, and also to develop a second script in the JotForm responses sheet in order to read them and push them to the proper sheet.

Figure 1. Google Apps Script

INSIGHTLY is a cloud CRM intended for small businesses. It offers free plans as well as premium ones and has integration options with social media and some widely used apps, as well as access to a REST API for those who want to build custom integration with other apps or not supported services.

In this project Insightly is used as the starting point of our pipeline and the place where full info of the opportunities, their state, tasks and other process info is stored. Insightly's API is queried every minute for new data to push on the sheet.

Figure 2. Insightly: different types of entities supported.

GOOGLE SHEETS is a free spreadsheet program by Google. Its part of a *software as a service* office suite,it's free to use and runs in the cloud, although it's also available offline. It can be bound to one or more google apps scripts to extend its possibilities and shared and stored through google drive.
In this project Google Sheets could be considered the core of the system. It is the central point that receives data from the CRM and the forms, sends it to the reporting tool and is embedded in the site.

Figure 3. Google Sheets.

JOTFORM is a free online form creator. It features a drag and drop interface where the user can easily design his forms in a very visual and easy way. JotForm supports many built-in integrations such as the one with google sheets that we are using in this project. Also it requires no registration, which makes it very convenient.

We have used JotForm to design a form where the user can enter data relative to the opportunities that are already in the sheet. The form has some validation rules like entering only integers where needed or having some required fields which made coding much easier.

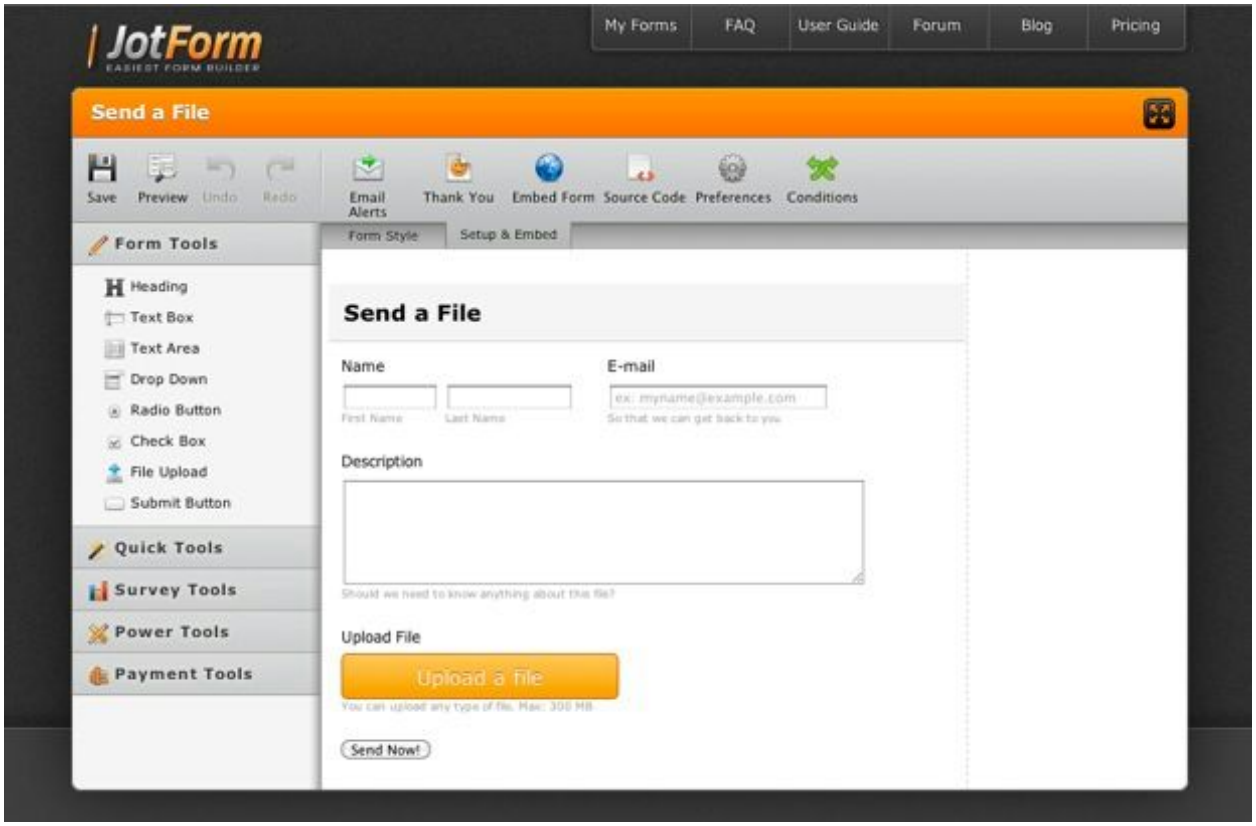

Figure 4. JotForm: visual form editor.

CYFE Is an all-in-one online dashboard that concentrates all the business data, events, calendars, etc in just one screen. It is very easy to set up -using pre-made widgets- and also a good tool to share real time info within our company. Cyfe can gather data from our accounts in Google Analytics, Salesforce, AdSense, MailChimp, Amazon, Facebook, WordPress, Zendesk, Twitter, etc and put it it into useful charts and graphs for easier analysis and understanding.

We have used Cyfe in its free version to process the raw data in the sheet and display it as visual and understandable reports. The reports are updated every 3 to 5 minutes so, as they are embedded in the site, we can say the site has a permanently updated report area.

Figure 5. Cyfe: Dashboard and its appearance on different platforms.

GOOGLE SITES is a web page creation utility part of the Google suite. Though it is a very simple product with a wiki-like interface, it can be escalated and enriched thanks to the growing community of widgets and add ons available for free.

Google Sites has been used to build the management web page where the rental manager can check data, enter updates and see the reports.

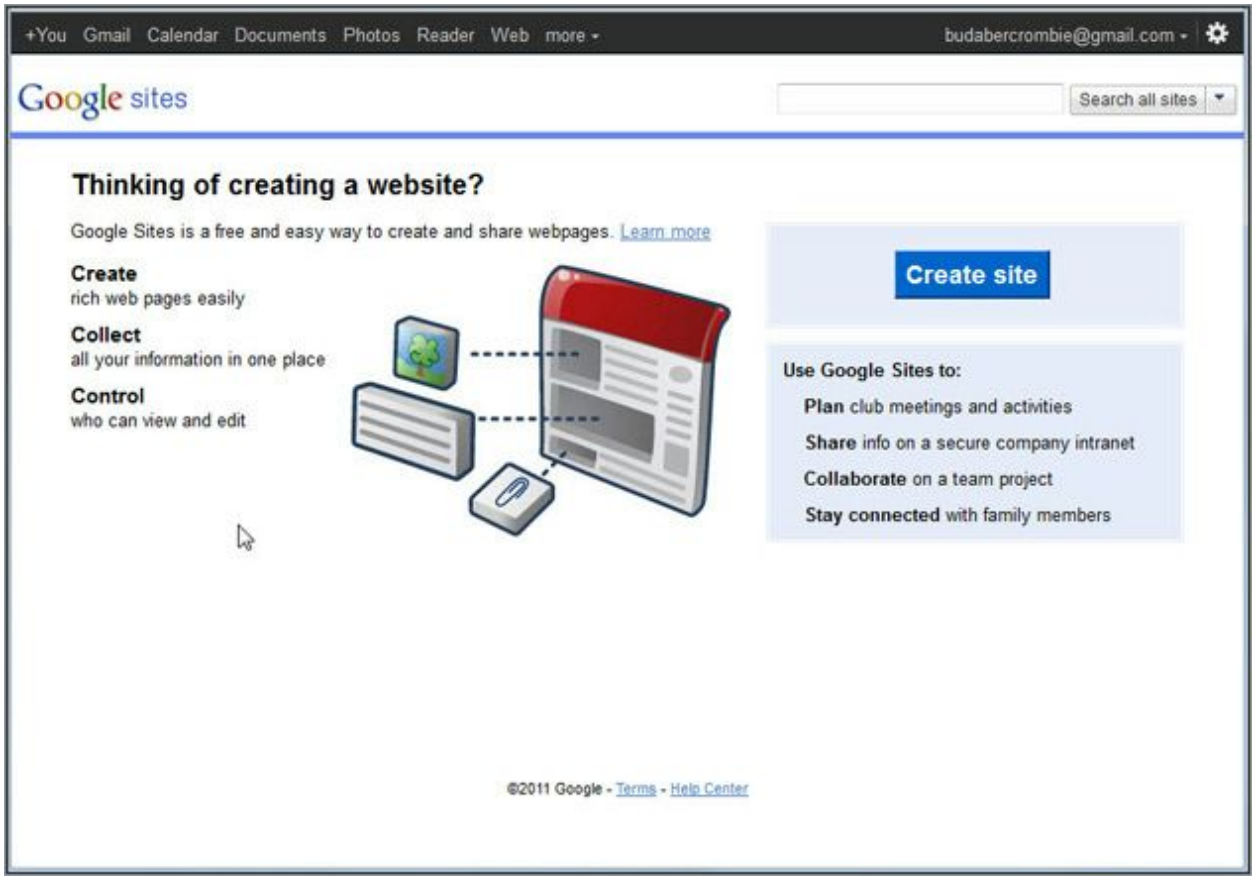

Figure 6. Google Sites: creating a site.

REST API IMPLEMENTATION

The main effort of this project has definitely been connecting to Insightly's API.
Although it is easily accessible, some programming skills are needed as well as understanding of the concepts of APIS and REST integration.

ABOUT REST

Representational State Transfer, or REST, is an architectural style that aims to build scalable web services.

RESTful systems communicate via HTTP using what is known like "rest verbs": POST, PUT, GET, DELETE, UPDATE… using these verbs and making the correct queries it is possible to acces the api's collection of resources and retrieve or edit the needed data.
If we want to build a RESTful service then there are some constraints we must respect:

- Client-server separation. Clients are not concerned with the database issues, and the server doesn't have to provide an interface to the user.
- Stateless. The communication consists on independent couples of query + response, therefore the server mustn't retain any state info or to persist any user or session data.
- Cacheable. A well designed web cache can save time in retrieving data that the client already had.
- Layering. Intermediary servers improve load balancing and system scalability.
- Uniform interface. This is a fundamental constraint for creating restful services: to simplify and decouple the architecture our interface will require identification of the resources, enable manipulation of resources via their representations -such as JSON objects- will send self explaining messages that contain themselves the data needed so the recipient knows how to process them, and use hypermedia as engine of the application state.

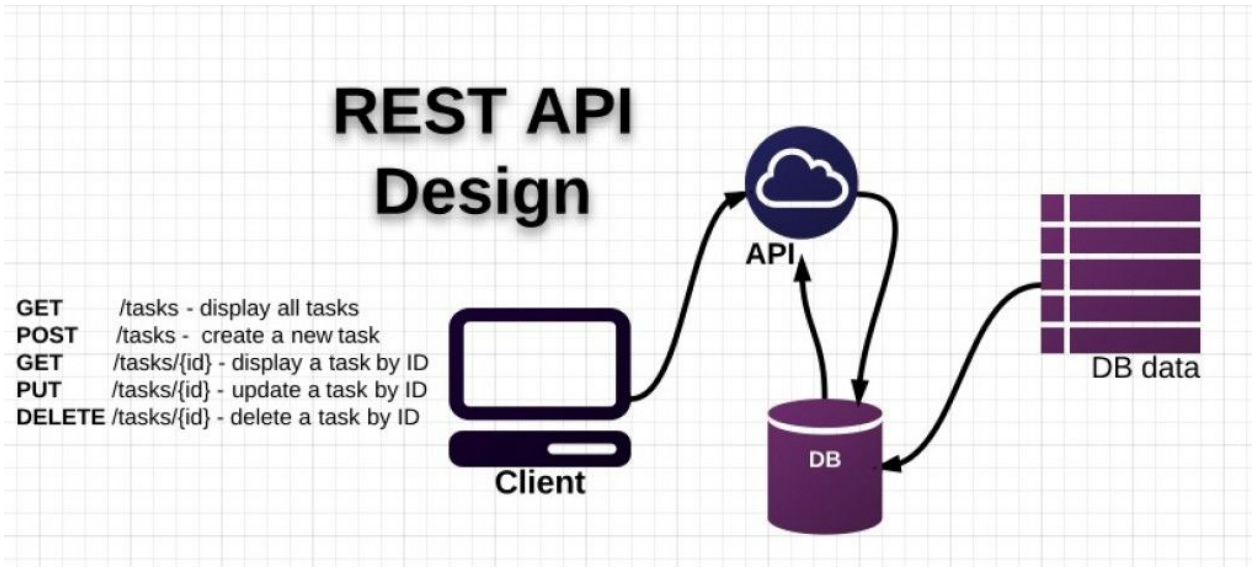

Figure 7. REST APIS: Schema, connexions and typical "verb" commands

These design patterns are more or less present in this implementation in many senses. For example, the sheet is bounded to the script but not dependant on it. The script itselfs makes certain queries to the API, accessing its resources in a way that resembles pure data base manipulation, but it does that in a way that makes it totally agnostic to what's behind the API.

1.4 THE SCRIPTS

We have two scripts implemented, one in the spreadsheet (with to versions, one for Barcelona and the other for Bristol) and a second one in the JotForm spreadsheet.

Bookings spreadsheet script

This script is based in three functions:
connectToInsighIty. This function sets the connection to the Insightly API. It contains the necessary parameters such as the manager authentication key, the endpoint with tag parameters (in order to query just for Bristol or Bcn content) and some formatting parameters such as muting http exceptions and expecting a JSON as response (the API can send us either an XML or a collection of JSON objects). This function has no trigger as it is called from the function getOpportunityDetails.

getOpportunityDetails. This function launches the query and stores the response in a json array. Then it loops trough the array that is written in a temporary sheet, splitting and separating its content and assigning it to variables which are later stored in the sheet within the correct column.
We can say this is the core function, fully responsible of asking for the correct info, determining how to handle it and writing it in its place. As we want this to be permanently updated, there are two triggers set upon this function: one running every minute and the other making it run after each sheet edit.

filePastBookings. This function loops through the active data range in the main sheet and when it finds an opportunity whose state is Won, Lost or Abandoned, it completes the task fields (review, checklist, etc) and files it in the Past_Bookings sheet. As this occurrence is much less frequent, this has a trigger set every 10 minutes.

JotForm sheet script

The script in the form sheet consists on just one function.
jotFormScript. This function loops through any row existing in the form responses sheet and evaluates if a response is relative to Bristol or Barcelona. After knowing this, it searches in that sheet for the opportunity ID and in case it exists, then it pastes the new info in the proper fields. If the data to paste is an integer it will paste it; if it's "0" it will any data existing in the destiny field. If it finds a null it won't affect the destiny field. After doing this it erases the response from the sheet to avoid having to make several changes for every script run.
This function has a trigger setup to make it run every minute.

1.5 SYSTEM QUALITY ASSURANCE

Quality assurance is a key point in every software project, but it is even more important in this one, as I am a Software Quality professional and have the ability to deliver my knowledge and experience in this particular field. Quality Assurance is gaining more and more recognition and space in the software lifecycle and so I wanted to reflect this as a core part of the implementation.
To ensure the system meets the requirements, a series of tests have been carried on the implementation. Here is documented, categorised by feature and with the low level description of the test cases.

Test Plan

1. A new opportunity in Insightly is pushed to the spreadsheet as a new row

i. Create a new opportunity in Insightly. Important: city tag (BRI or BCN) must be declared.
Write the customer's full name in the description field, and write the address without using any commas.

ii. Check that in a lapse of a minute the correct spreadsheet shows a new row with the new opportunity.

2. Opportunity updates made in Insightly are updated in the site

i. Access the opportunity created and edit any data: number of nights, description...

ii. Verify that the edition is shown in the site in about a minute after saving the changes in the crm

iii. Verify that, if we make annotations about the cost of the stay and later change something in Insightly, the integration of these changes won't affect the economic data.

3. Lost opportunities are filed

i. Mark every task of the opportunity as "done". Change the state of the opportunity as "lost", "suspended" or "abandoned".

ii. Verify that the corresponding row is filed in a lapse of about 5 minutes since the moment we save the new state. It must appear in the Past_Bookings sheet with every field related to a task (Fiança, Review, Checklist…) completed.

4. Re-opened opportunities appear as new again

i. Return to the opportunity you have closed and mark it as open again.

ii. Check that in a lapse of about 5 minutes it appears again in the main sheet. Task fields must appear blank again.

5. Won opportunities are filed

i. Mark every task of the opportunity as “done”. Change the state of the opportunity as “Won”.

ii. Verify that the corresponding row is filed in a lapse of about 5 minutes since the moment we save the new state. It must appear in the Past_Bookings sheet with every field related to a task (Fiança, Review, Checklist…) completed.

6. If a row is deleted by accident in the spreadsheet it will be restored

i. Access the spreadsheet directly (not through the site) and delete an opportunity row.

ii. Check that in about a minute the original row is recover. Note that if it had any economic data it cannot be recovered as it is not residing in Insightly’s database.

7. The sheets are protected in the management site

i. Verify that when accessing the site, in the management section of each city, we can see the sheet with the bookings but we are not able to write on it or make any changes.

8. The reports are visible in the management site

i. In the management site, go to the reporting section of every city’s section

ii. Verify that the cyfe dashboard appears, that it is fully editable and shows updated reports (note: seeing the dashboard may require additional login)

9. Economic data can be added through the site’s form

i. Check the id of the opportunity you created and re opened.

ii. In the management site, go to the homepage: you'll find a form

iii. Fill in the form with the city and opportunity id. Also fill any of the possible data, for example the cost one.

iv. Fill another field with any other number. Press SUBMIT.

v. Check that the new data appears in the site, in the row of the opportunity, cost column.

10. Economic data can be nulled through the site's form

i. Reload the homepage again to see the form.

ii. This time fill the field you submitted before with "0"

iii. Check that the field marked with "0" now appears null.

iv. Verify that the other field that had a value from the last test, and that has now been sent in the form as null, has suffered no variations. The old value is preserved.

11. Form data validation cases

i. Try to submit the form with a right opportunity id and correct values, but wrong city. Verify nothing new appears on the sheets.

ii. Try to submit the form with a right city and values but with a wrong opportunity id. Verify nothing new appears on the sheets.

iii. Try to submit a form without indicating any city in the dropdown. Verify the form can't be submited, a warning appears.

iv. Try to submit a form without indicating any opportunity id. Verify the form can't be submited, a warning appears.

v. Try to submit a form without indicating any value, but with correct city and opportunity id. Verify no change is made in the opportunity, at the sheet.

vi. Try to submit a form with correct city and opportunity id, but with a entering a value that is not numeric. Check that the form doesn't allow submitting not numeric values.

1.6 SYSTEM REQUIREMENTS

Hardware requirements are very low for this system. As everything runs in the cloud, the manager's machine doesn't need to be particularly modern or fast to enable a good user experience.
A broadband connection is mandatory to work online in Insightly, JotForm, Google Sites and Cyfe.
For Google Sheets, they can be accessed even offline if it's eventually needed, but user settings in Google Drive must have been changed to enable offline access to the sheet.

This system can also be accessed from mobile devices as cell phones or tablets, although it is not very recommendable as the layouts and themes are not designed as mobile-responsive and the UX would appear poor and a little bothering to use.

2. USER MANUAL

The sales process for the holiday rental business follows this simple diagram:

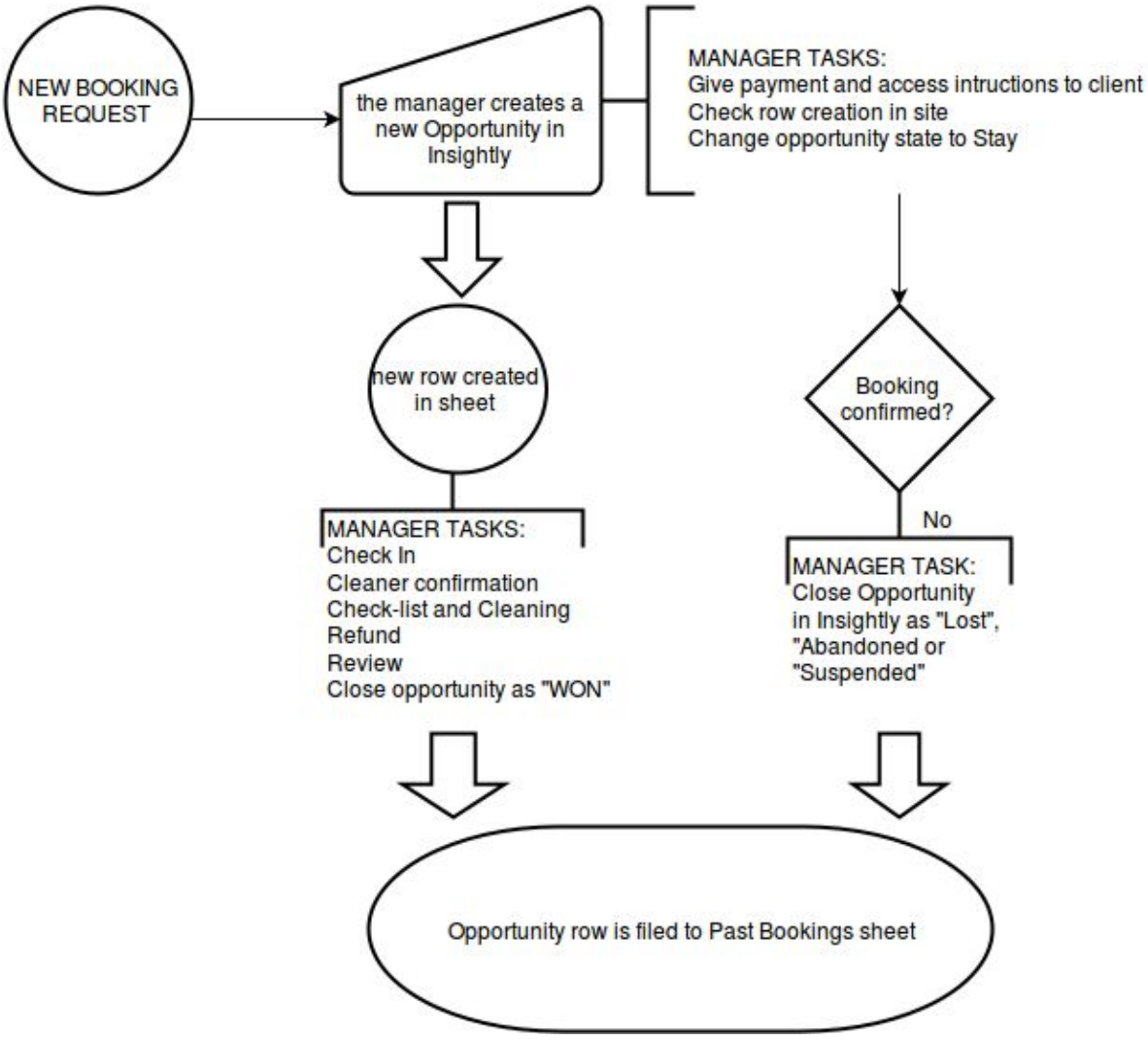

Figure 8. Opportunity pipeline: from the new booking to the sale closing.

To manage our business we will interact with Insightly, our cloud crm app, and our management site.

STAGE 1 : NEW BOOKING

CREATING THE BOOKING

Our flow starts with the notice of a new booking request. The flat manager talks to the customer to confirm pricing info and then enters the new opportunity at Insightly.

HOW TO CREATE AN OPPORTUNITY

Opportunities are the entities that enter our system as potential stays and are eventually closed as finished stays or cancelled requests.

To create a new one we must click on “Opportunities” in Insightly’s left column and then on top-right in the Opportunities screen, click on “NEW OPPORTUNITY”:

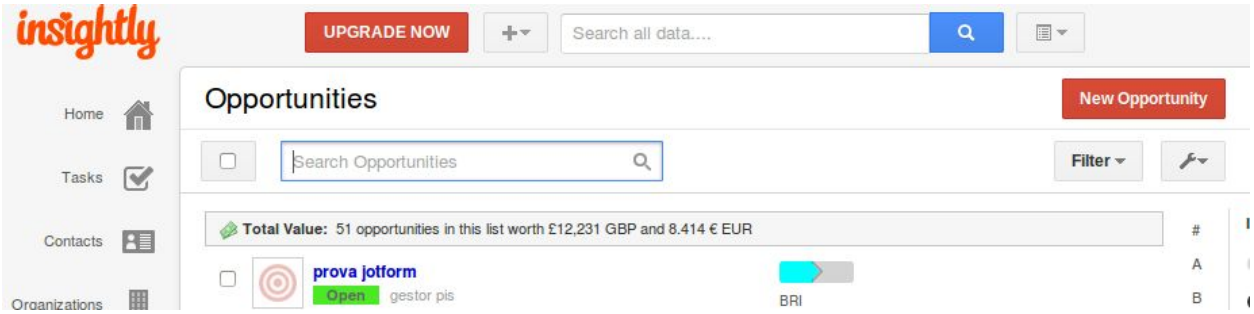

Figure 9. Button “New Opportunity” appearing on the top right of the Opportunities screen

After doing that a blank template appears with all the necessary fields to create a new opportunity. Please keep in mind not to use commas:

1. **Opportunity Details:** In this field we must enter a name for the opportunity. This should be something simple and self-explaining, such as “Name of the customer - DayIn/DayOut”, for example.
2. **Description:** In this field we write the customer's full name.
3. **Additional Info section**: There is nothing mandatory that we must write here
4. **Opportunity Pipeline:** A pipeline is a sequence of steps already defined for some kid of sale. We must choose “Bookings” here, and that will automatically set the stage to “Book”, which is correct.
5. **Opportunity custom fields:** these are some fields already set up in our system configuration; they contain specific data that we think is a must for managing our opportunities. All of them must be field so the proper information arrives later to the management site.
6. **Links/Permissions:** At the moment none of this sections is necessary.

After filling the form, we click on the “SAVE” button on the top of it, and the template form will turn into a view of the already saved opportunity.

Important: As a final step we must add a tag so we can now to which of our flats is the booking related. To do so, click on the “manage tags” link that appears just beside the opportunity’s title:

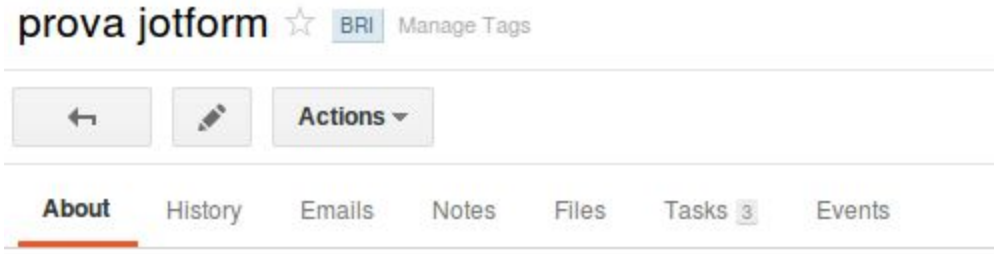

Figure 10. Where to add the opportunity tag.

And then on the little modal window that will pop up on the screen, write the appropriate tag (BRI for Bristol, BCN for Barcelona):

Figure 11. The modal window proposes already used tags.

Click on “SAVE” and we’re done.

After this is completed, you will notice that 3 new tasks have been created in that opportunity. Just follow along them, checking them as “Done” as they are completed.
More specifically, task n.2 states “Opportunity and excel row creation and calendars update”, you can check the row has been created in the management site, right on the “Management” section of the desired flat area.
Now we can go to the opportunity, open it and at the bottom-right of the page click on “Change State” to put it as “Stay”.

UPDATING AN OPPORTUNITY

A booking can be accessed whenever needed to change any of its details (customer name, dates, etc…), or to update its state. Changes will be integrated in the site in about a minute.

DELETING AN OPPORTUNITY

Opportunities can't be deleted, but their state can be changed to a final one in order to close them properly, as explained in the section "Closing an opportunity".

STAGE 2 : "STAY"

MANAGING THE BOOKING

After having created the booking in our system there are some tasks we as managers must to. Some of them are listed and ready to complete when we change the opportunity state from "Book" to "Stay":

- Cleaner confirmation
- Cleaning and Checklist
- Refund
- Review
- Close opportunity as WON
- Ready for check-in

But we also have to introduce the economic info in our site, which we can do with the form we will find at the Home page of the management site (https://sites.google.com/site/gestorpis/home):

Figure 12. Form to update the economic info.

From this form we can update the economic info for any opportunity already existing in our sheet. We must only insert numeric data; actually if we try to insert any other kind of characters we will see a notice like this:

Figure 13. Field validation in the JotForm form.

Instructions for filling this form are very simple:

First one must indicate if the row to update belongs to a Bristol or Barcelona reservation, and enter the Opportunity ID as stated in the sheet and Insightly. Please keep in mind that if a submission is made with a wrong ID or indicating a wrong city the data will not be inserted.

Then we can fill the rest of the fields, having the following options:

- We can write a value in a field in order to have it inserted in the sheet
- We can write “0” in a field and that with empty any info that was in that field
- We can leave the field blank and that won’t make any changes in that field (for example when you want to update just one field, you can leave the others blank not to lose the info already existing in the sheet for the rest of the fields).

Only economic data can be updated from the field, as the other opportunity’s details belong to insightly and must be maintained there.

There is a trigger set to run every minute for this form, so any change we make will be reflected in the site in about a minute.

CLOSING AN OPPORTUNITY

When a sale is finished or cancelled we must close the opportunity and give it a final state. We can do that pressing on the “Change” link in “Current State”:

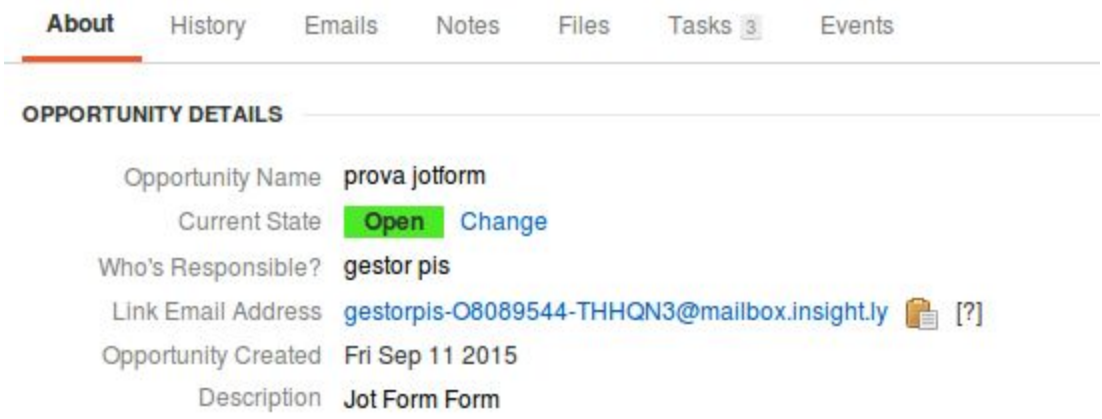

Figure 14. State info in an open opportunity.

There are some options available when closing an opportunity. The correct one according to the closing reason will be the following:

- Stay has ended successfully and every task is done: set the state to “Won”

- The customer has cancelled the booking or the booking hasn't been finally confirmed: set the state to either Lost, Suspended or Abandoned.

This change of state will appear in the site within one minute. Also, every ten minutes an additional script will pick any closed opportunity in the sheet, set some of its fields to "OK" (fields relating to tasks such as "checklist", "review", etc) and file it in a second sheet called "Past Bookings). If eventually a booking is re-opened back from a closed status it will result in duplication (will appear both in the main and the past bookings sheets).

3. REPORTING

The application -as the version that arrived to me when this project started- included a well designed module with economic reports. That reports had a disadvantage, that was the need of querying the specific data and ranges every time I needed to see them, and the impossibility to persist them in a protected and user friendly way or embedding them in the management site I had built.

Seeing that, and for the sake of investigating with integrations, that module was deactivated and Cyfe was set up to implement new graphical reports in the site.
This implementation is based in to desktop-looking dashboard that have been integrated in the site. Each one of them belong to one of the cities and can show up to five widgets -that is, in the free version we are using. Those widgets can be set up to show a calendar with future tasks and bookings, data grids, visual reports and many metrics widgets for controlling web analytics, marketing cost and return, and a wide quantity of useful tools. It also has some great configuration possibilities, like the ability of letting your colleagues or employees see just a part of the company's info and insights.

For what matters to us, the aim was to show the potential of integrating with an app specific for managing the reports and how useful it could be to see it from the same site. The dashboard was embedded in the Reports sections of every city's area, on its own subpage, and a simple calendar and a chart is currently configured.

The Cyfe charts resulted specially convenient because the app has its own integration ways. So everything the user has to do is to select a chart widget, drag and drop it in the desired place over the dashboard and configure his Google account to give the widget access to the spreadsheets. A special hidden sheet with a matrix of data has been prepared so the widget reads directly from it. What is shown currently is a simulation of real time data linked to a static data grid, but the possibilities are endless: we could have a matrix that is updated with the main or the Past_Bookings sheet, from another sheet nurtured through a script, etc.

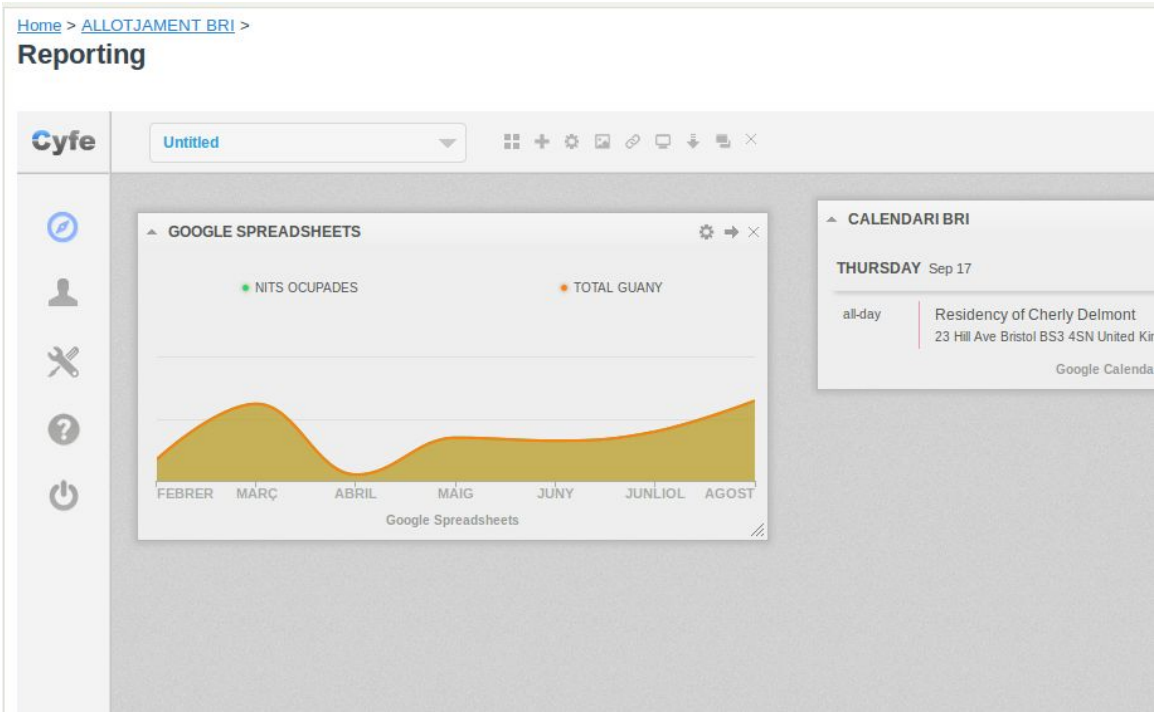

Figure 15. Cyfe dashboard with a calendar widget and a chart widget.

It is also important to note that, although the dashboard is embedded in the sheet, it has its own authentication path, so any visitor that could accidentally arrive to our site would not be able to see the reports and, even more delicate, editing them or affecting the Google authorization (as we said, the dashboard appears active and fully editable; the user doesn't really need to access cyfe directly again because the 100% of the options will already appear in the site).

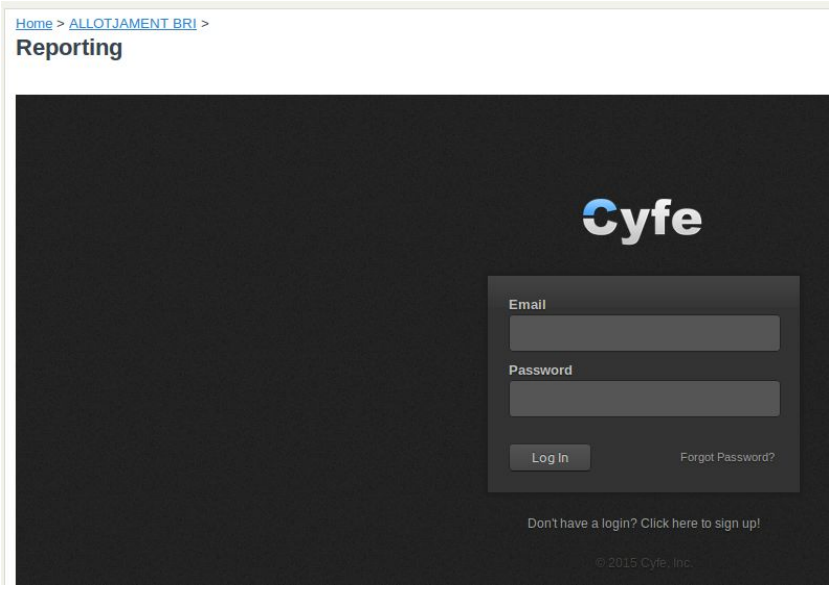

Figure 16. View of the Cyfe embedded dashboard while not logged in.

4. ENVIRONMENTALIZATION

This app can help reduce the overall environmental impact of our business, as the apartment manager just need a broadband connection to log into it: the access to the app is easy and decentralized. This advantage may save many unnecessary trips because the manager can work from anywhere while he's connected: home, an office, a coffee shop... Additionally it doesn't need investment in new and powerful machines or other devices, and even more it avoids dumping old machines with its inherent recycling costs.

Printouts are not necessary either; the system generates visual reports with the main parameters of the process that help the admin watching his monthly or yearly earning easily. This is a very important point too as it saves paper, trees and oxygen.

However, the app is related to tourism and its relationship with the environment is a complex item as it involves activities that can have adverse environmental effects. Much of this impact is linked to the construction of general infrastructures such as roads and airports, tourism facilities, restaurants, shops, amusement parks... This impact can gradually damage and destroy the resources on which the local and global balance rely.

On the other hand, tourism has the potential of creating beneficial effects on the nature by contributing to environmental protection and conservation. It's a way to raise awareness of good values and it can serve as a tool to finance preservation of natural areas and increase their economic value.

5. CONCLUSIONS

At the beginning of this project I faced a few obstacles. One of them was my lack of experience as a programmer and specially with the Google Apps Script language, as I had never seen it before nor had I work with Javascript, which is very similar. I also didn't have much experience working with Drive nor spreadsheets, so the first challenge was undoubtedly getting familiar with the app, the code editor and the language.

After this, also the limitations of the script presented many obstacles. One of them was the impossibility of creating a dedicated UI directly from the spreadsheet script with my own means and knowledge. I have to thank in this point the author of the former project, which helped me understanding what was already done by him and how it works, and knowing what his code did. Certainly manipulating and escalating another person's code is something delicate and that requires more learning than the language per se.

Other minor obstacles was fitting the requirements to the free versions of the chosen apps. They usually restrict some functionalities and some workarounds were necessary to fully obtain opportunities data from Insightly, from example.

Having overcome this little stones along the way, the results have not deceived me. What we have now is a more robust version of the prior app. The spreadsheet itself has more or less the same features, but appears fully protected from the user's possible mistakes. In the former version it was possible to drag a column, delete something or erase a formula by accident, all three actions that could break the script by making it incapable of finding a certain field. Now the sheet is shielded by a simple and light UI that offers protection without penalizing load time, velocity or interaction. The user can rely on a single point to control the business and doesn't have the need of visiting the reports site, opening one or another sheet, going to find the opportunities in the crm etc.

And more important, finally all of this has been achieved without altering the initial spirit of the project: something fully free, light, easy to use and that doesn't require learning, buying new devices or having a computer science degree. Something that can be shared with the community and be escalated and tailored in a continuous manner for his business or any other.

6. POTENTIAL FOR IMPROVING

I wanted to make a little final reflection to take a look at the possibilities of this system in the future.

There is still a huge lot of room for new integrations that could help boosting the business. This improvements were out of scope for this project and probably didn't deliver any more value in what refers to integrations (they would be exactly the same I had already done: integrating third party products). But they do bring creative ideas to grow the business in the same direction and with a more customer oriented approach.

So, some of those possibilities could be the following:

- **Mailchimp integration:** A new module could be added to the sheet, so when an opportunity is due Mailchimp sends an email to the customer. The mail could be sent without interaction from the user. It could make profit of a pre-made template (of many templates, according to the language of the user for example). That template with be filled with data from the spreadsheet like the customer name, check in dates, a weather forecast and some itinerary indications.

- **Social media + wifi integration:** the wifi installation in the flats could be free but linked to the customer accepting giving Like to our Facebook page. This like could be informed to the sheet and detailed in the opportunity row so we can thank him later.

- **Invoicing apps integration:** There are many options of free, cloud based invoice creators. Integrating the site with one of them could enable the manager to print an invoice with no effort just taking the data from the sheet. Or even more, we could add some invoice printer integration.

- **TPV/Paypal integration:** Communicating the site with some payment options could allow to enter the payment confirmation and sale balance in the sheet without typing. And reversely, the tpv could get the numbers from the sheet too.

- **Leisure offers:** Many ticketing services offer integration for the seller's websites. It could be great to offer your customers tickets for museums, concerts or other local attractions so the business can profit of a little fee and the visitors avoid waiting in line and get an extra service from us.

7. ADDENDUM

7.1 CODE

BRISTOL SHEET CODE:

```
//declare the sheet
var ss = SpreadsheetApp.getActiveSpreadsheet();



//:::::::::::::::::INSIGHTLY API CONNECTION BLOCK ::::::::::::::::::::::::::::::::::::::::::::::::::::::
// connectToInsightly: sets user and url parameters for API querying
function connectToInsightly() {
var key = "N2Y5MzgyOWItZjQyYy00MTFjLTk3YmQtNDYwMzFjZWVmOWM1";
var url = "https://api.insight.ly/v2/Opportunities?Brief=True&tag=BRI";
var response = UrlFetchApp.fetch(url, {
          muteHTTPexceptions: true,
          headers: {
          'Authorization': 'Basic ' + key,
          'Content-Type': 'application/json'
          }
  });
  return JSON.parse(response.getContentText());
}
//getOpportunitydetails: 1.bring API response 2. Dump to log 3. Write to new sheet 4. Check if
record already existed 5. Write new records on sheet 0.
function getOpportunityDetails() {
  ss.insertSheet("Temporal", 1)
  var endp = "/v2.1/Opportunities";
  var opportunities = [];
  opportunities = connectToInsightly(endp);

  var c = 1;
  var d = 0;
  for (var p in opportunities) {
          //throw JSON data into logger
  Logger.log(opportunities[p]);

  var range = "A"+c;
  var lc =  ss.getSheets()[1].getActiveCell();
```

```
  var currentRow = (opportunities[d]);
  ss.getSheets()[1].getRange(range).setValue(currentRow);
          //map JSON fields to spreadsheet concepts
          var cell1 = ss.getSheets()[1].getRange(range).getValues()[0];
          var sCell = cell1[0].split(",");
          var oName = sCell[17].split("=");
          var opportunityName = oName[1];
          var oID = sCell[20].split("=");
          var opportunityID = oID[1];
          var ciDate = sCell[18].split("=");
          var checkInDate = ciDate[1];
          var coDate = sCell[16].split("=");
          var checkOutDate = coDate[1];
          var fName = sCell[13].split("=");
          var fullName = fName[1];
          var dI = sCell[8].split("=");
          var daysIn = dI[1];
          var ad = sCell[12].split("=");
          var adults = ad[1];
          var inf = sCell[11].split("=");
          var infants = inf[1];
          var og = sCell[10].split("=");
          var origen = og[1];
          var origenReserva = "Insightly integration";
          var contact = sCell[15].split("=");
          var contactData = contact[1];
          var mail = sCell[14].split("=");
          var contactEmail = mail[1];
          var oState = sCell[32].split("=");
          var opportunityState = oState[1];
          //check opportunity doesn't already exist in sheet
          var lastWrittenRow = ss.getSheets()[0].getDataRange().getLastRow();
          var idRange = ss.getSheets()[0].getRange(1, 2, lastWrittenRow, 1);
          var idArray = idRange.getValues();
          var escr = true;
          ss.getSheets()[0].getRange(1, 1, ss.getSheets()[0].getDataRange().getLastRow(),
ss.getSheets()[0].getDataRange().getLastColumn()).setBorder(true, true, true, true, true,
true);
          for (var u = 1; u <= lastWrittenRow; u++){
          if (idArray[u] == opportunityID) {
          var escr = false;
          var rowNumber = u+1;
          }
          }
          if (escr){
          //if they aren't already there, paste new opportunities in main sheet
          var newRow = ss.getSheets()[0];
```

```
        //write in first columns
        newRow.appendRow([opportunityName, opportunityID, origenReserva, fullName,
contactEmail, contactData, checkInDate, checkOutDate, daysIn,
(parseInt(adults))+(parseInt(infants)), adults, infants, origen]);
        // write sate in col.24
        ss.getSheets()[0].getRange(lastWrittenRow, 24).setValue(opportunityState);
        } else if (escr == false) {
        ss.getSheets()[0].getRange(rowNumber, 1).setValue(opportunityName);
        ss.getSheets()[0].getRange(rowNumber, 4).setValue(fullName);
        ss.getSheets()[0].getRange(rowNumber, 5).setValue(contactEmail);
        ss.getSheets()[0].getRange(rowNumber, 6).setValue(contactData);
        ss.getSheets()[0].getRange(rowNumber, 7).setValue(checkInDate);
        ss.getSheets()[0].getRange(rowNumber, 8).setValue(checkOutDate);
        ss.getSheets()[0].getRange(rowNumber, 9).setValue(daysIn);
        ss.getSheets()[0].getRange(rowNumber,
10).setValue(parseInt(adults))+(parseInt(infants));
        ss.getSheets()[0].getRange(rowNumber, 11).setValue(adults);
        ss.getSheets()[0].getRange(rowNumber, 12).setValue(infants);
        ss.getSheets()[0].getRange(rowNumber, 13).setValue(origen);
        ss.getSheets()[0].getRange(rowNumber, 24).setValue(opportunityState);
        }
        c++;
        d++;
   }
  //Dump Sheet 1 after every script run
  if (ss.getSheetByName("Temporal")){
        ss.deleteSheet(ss.getSheetByName("Temporal"));
  }
}
//:::::::::::::::::END OF INSIGHTLY API CONNECTION BLOCK
::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::

  //This function files all the bookings with state = won, lost, abandoned or suspended.
  //If we re-open an opportunity it will return to the first sheet in the next script run.
function filePastBookings() {

  //declare the sheet
  var ss = SpreadsheetApp.getActiveSpreadsheet();
  var pastBookingsSheet = ss.getSheetByName("Past_Bookings");
  var BristolSheet = ss.getSheetByName("BRISTOL")

  //get all the written rows
  var lastOpportunity = ss.getSheets()[0].getDataRange().getLastRow();
  var lastCol = ss.getSheets()[0].getDataRange().getLastColumn();
  //get the State column and it's last row
  var stateColumn = ss.getSheets()[0].getRange(2, 24, lastOpportunity, 1);
  //get state values into array
```

```
  var stateArray = stateColumn.getValues();
  //loop trough the array checking for rows with state different than "OPEN"
  for (var y = 0; y < lastOpportunity-1; y++){
          if (stateArray[y][0] != "OPEN" && stateArray[y][0] != null) {
          //if state is not open or null, then write this values in sheet 0:
          var rowNumber = y+2;
          ss.getSheets()[0].getRange(rowNumber, 17).setValue("SI");
          ss.getSheets()[0].getRange(rowNumber, 23).setValue("OK");
          ss.getSheets()[0].getRange(rowNumber, 24).setValue("OK");
          ss.getSheets()[0].getRange(rowNumber, 26).setValue("retornat");
          ss.getSheets()[0].getRange(rowNumber, 28).setValue("OK");
          ss.getSheets()[0].getRange(rowNumber, 29).setValue("OK");
          ss.getSheets()[0].getRange(rowNumber, 30).setValue("OK");


          //find last written row in past_bookings sheet

          var destinationRow = pastBookingsSheet.getLastRow()+1;
          var actualRow = ss.getSheets()[0].getRange(rowNumber, 1, 1,
ss.getSheets()[0].getDataRange().getLastColumn());
//        var actualRowValores = ss.getSheets()[0].getRange(rowNumber, 1, 1,
ss.getSheets()[0].getDataRange().getLastColumn()).getValues();
          //and move the whole row to past bookings sheet

          var destinationRowRange = pastBookingsSheet.getRange(destinationRow, 1, 1, 31)

          actualRow.copyTo(destinationRowRange);


          // The code below will erase the copied row and positioned the next row in his spot
          ss.getSheets()[0].deleteRow(rowNumber);
          //Reset the counter 1 step above because we have 1 less row now
          }
  }
}
```

BCN SHEET CODE:

```
//declare the sheet
var ss = SpreadsheetApp.getActiveSpreadsheet();
```

```
//:::::::::::::::::INSIGHTLY API CONNECTION BLOCK ::::::::::::::::::::::::::::::::::::::::::::::::::::
// connectToInsightly: sets user and url parameters for API querying
function connectToInsightly() {
var key = "N2Y5MzgyOWItZjQyYy00MTFjLTk3YmQtNDYwMzFjZWVmOWM1";
var url = "https://api.insight.ly/v2/Opportunities?Brief=True&tag=BRI";
var response = UrlFetchApp.fetch(url, {
        muteHTTPexceptions: true,
        headers: {
        'Authorization': 'Basic ' + key,
        'Content-Type': 'application/json'
        }
  });
  return JSON.parse(response.getContentText());
}
//getOpportunitydetails: 1.bring API response 2. Dump to log 3. Write to new sheet 4. Check if
record already existed 5. Write new records on sheet 0.
function getOpportunityDetails() {
  ss.insertSheet("Temporal", 1)
  var endp = "/v2.1/Opportunities";
  var opportunities = [];
  opportunities = connectToInsightly(endp);

  var c = 1;
  var d = 0;
  for (var p in opportunities) {
        //throw JSON data into logger
  Logger.log(opportunities[p]);

  var range = "A"+c;
  var lc =  ss.getSheets()[1].getActiveCell();
  var currentRow = (opportunities[d]);
  ss.getSheets()[1].getRange(range).setValue(currentRow);
        //map JSON fields to spreadsheet concepts
        var cell1 = ss.getSheets()[1].getRange(range).getValues()[0];
        var sCell = cell1[0].split(",");
        var oName = sCell[17].split("=");
        var opportunityName = oName[1];
        var oID = sCell[20].split("=");
        var opportunityID = oID[1];
        var ciDate = sCell[18].split("=");
        var checkInDate = ciDate[1];
        var coDate = sCell[16].split("=");
        var checkOutDate = coDate[1];
        var fName = sCell[13].split("=");
        var fullName = fName[1];
        var dI = sCell[8].split("=");
        var daysIn = dI[1];
```

```
        var ad = sCell[12].split("=");
        var adults = ad[1];
        var inf = sCell[11].split("=");
        var infants = inf[1];
        var og = sCell[10].split("=");
        var origen = og[1];
        var origenReserva = "Insightly integration";
        var contact = sCell[15].split("=");
        var contactData = contact[1];
        var mail = sCell[14].split("=");
        var contactEmail = mail[1];
        var oState = sCell[32].split("=");
        var opportunityState = oState[1];
        //check opportunity doesn't already exist in sheet
        var lastWrittenRow = ss.getSheets()[0].getDataRange().getLastRow();
        var idRange = ss.getSheets()[0].getRange(1, 2, lastWrittenRow, 1);
        var idArray = idRange.getValues();
        var escr = true;
        ss.getSheets()[0].getRange(1, 1, ss.getSheets()[0].getDataRange().getLastRow(),
ss.getSheets()[0].getDataRange().getLastColumn()).setBorder(true, true, true, true, true,
true);
        for (var u = 1; u <= lastWrittenRow; u++){
        if (idArray[u] == opportunityID) {
        var escr = false;
        var rowNumber = u+1;
        }
        }
        if (escr){
        //if they aren't already there, paste new opportunities in main sheet
        var newRow = ss.getSheets()[0];
        //write in first columns
        newRow.appendRow([opportunityName, opportunityID, origenReserva, fullName,
contactEmail, contactData, checkInDate, checkOutDate, daysIn,
(parseInt(adults))+(parseInt(infants)), adults, infants, origen]);
        // write sate in col.24
        ss.getSheets()[0].getRange(lastWrittenRow, 24).setValue(opportunityState);
        } else if (escr == false) {
        ss.getSheets()[0].getRange(rowNumber, 1).setValue(opportunityName);
        ss.getSheets()[0].getRange(rowNumber, 4).setValue(fullName);
        ss.getSheets()[0].getRange(rowNumber, 5).setValue(contactEmail);
        ss.getSheets()[0].getRange(rowNumber, 6).setValue(contactData);
        ss.getSheets()[0].getRange(rowNumber, 7).setValue(checkInDate);
        ss.getSheets()[0].getRange(rowNumber, 8).setValue(checkOutDate);
        ss.getSheets()[0].getRange(rowNumber, 9).setValue(daysIn);
        ss.getSheets()[0].getRange(rowNumber,
10).setValue(parseInt(adults))+(parseInt(infants));
        ss.getSheets()[0].getRange(rowNumber, 11).setValue(adults);
```

```
          ss.getSheets()[0].getRange(rowNumber, 12).setValue(infants);
          ss.getSheets()[0].getRange(rowNumber, 13).setValue(origen);
          ss.getSheets()[0].getRange(rowNumber, 24).setValue(opportunityState);
          }
          c++;
          d++;
   }
  //Dump Sheet 1 after every script run
  if (ss.getSheetByName("Temporal")){
          ss.deleteSheet(ss.getSheetByName("Temporal"));
  }
}
//:::::::::::::::::END OF INSIGHTLY API CONNECTION BLOCK
::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::

  //This function files all the bookings with state = won, lost, abandoned or suspended.
  //If we re-open an opportunity it will return to the first sheet in the next script run.
function filePastBookings() {

  //declare the sheet
  var ss = SpreadsheetApp.getActiveSpreadsheet();
  var pastBookingsSheet = ss.getSheetByName("Past_Bookings");
  var BristolSheet = ss.getSheetByName("BRISTOL")

  //get all the written rows
  var lastOpportunity = ss.getSheets()[0].getDataRange().getLastRow();
  var lastCol = ss.getSheets()[0].getDataRange().getLastColumn();
  //get the State column and it's last row
  var stateColumn = ss.getSheets()[0].getRange(2, 24, lastOpportunity, 1);
  //get state values into array
  var stateArray = stateColumn.getValues();
  //loop trough the array checking for rows with state different than "OPEN"
  for (var y = 0; y < lastOpportunity-1; y++){
          if (stateArray[y][0] != "OPEN" && stateArray[y][0] != null) {
          //if state is not open or null, then write this values in sheet 0:
          var rowNumber = y+2;
          ss.getSheets()[0].getRange(rowNumber, 17).setValue("SI");
          ss.getSheets()[0].getRange(rowNumber, 23).setValue("OK");
          ss.getSheets()[0].getRange(rowNumber, 24).setValue("OK");
          ss.getSheets()[0].getRange(rowNumber, 26).setValue("retornat");
          ss.getSheets()[0].getRange(rowNumber, 28).setValue("OK");
          ss.getSheets()[0].getRange(rowNumber, 29).setValue("OK");
          ss.getSheets()[0].getRange(rowNumber, 30).setValue("OK");


          //find last written row in past_bookings sheet
```

```
        var destinationRow = pastBookingsSheet.getLastRow()+1;
        var actualRow = ss.getSheets()[0].getRange(rowNumber, 1, 1,
ss.getSheets()[0].getDataRange().getLastColumn());
//        var actualRowValores = ss.getSheets()[0].getRange(rowNumber, 1, 1,
ss.getSheets()[0].getDataRange().getLastColumn()).getValues();
        //and move the whole row to past bookings sheet

        var destinationRowRange = pastBookingsSheet.getRange(destinationRow, 1, 1, 31)

        actualRow.copyTo(destinationRowRange);


        // The code below will erase the copied row and positioned the next row in his spot
        ss.getSheets()[0].deleteRow(rowNumber);
        //Reset the counter 1 step above because we have 1 less row now
        }
  }
}
```

JOTFORM SHEET SCRIPT:

```
function jotformScript() {

  var jotSheet =
SpreadsheetApp.openById("1aBBdwvNBm97R4Yk12UFulcswZhMqU9mmticxYnvFN_Q").get
Sheets()[0];
  var bristolSheet =
SpreadsheetApp.openById("14HYwJvimhsLVfA1YqNyN94F4y6KmMt27AlPjU1Ikx7A").getSh
eets()[0];
  var bcnSheet =
SpreadsheetApp.openById("14HYwJvimhsLVfA1YqNyN94F4y6KmMt27AlPjU1Ikx7A").getSh
eets()[0]; //cambia el ID por el de barna

  var jotSheetValues = jotSheet.getDataRange().getValues();

  for (var c = 0; c < jotSheet.getLastColumn(); c++){
          if (jotSheetValues[0][c] == "Submission Date"){
          var jSubmissionDateCol = c;
          }
          if (jotSheetValues[0][c] == "ALLOTJAMENT"){
          var jALLOTJAMENTCol = c;
          }
          if (jotSheetValues[0][c] == "ID oportunitat"){
          var jIDoportunitatCol = c;
          }
          if (jotSheetValues[0][c] == "Cost pel Client"){
          var jCostpelClientCol = c;
          }
          if (jotSheetValues[0][c] == "Paga i senyal"){
          var jPagaisenyalCol = c;
          }
          if (jotSheetValues[0][c] == "Fiança"){
          var jFiancaCol = c;
          }
          if (jotSheetValues[0][c] == "Impostos"){
          var jImpostosCol = c;
          }
          if (jotSheetValues[0][c] == "Despeses gestió"){
          var jDespesesgestioCol = c;
          }
          if (jotSheetValues[0][c] == "Despeses neteja"){
          var jDespesesnetejaCol = c;
          }
          if (jotSheetValues[0][c] == "IP"){
```

```
        var jIPCol = c;
        }
        if (jotSheetValues[0][c] == "Submission ID"){
        var jSubmissionIDCol = c;
        }
        if (jotSheetValues[0][c] == "Edit Link"){
        var jEditLinkCol = c;
        }
}

for (var i = 1; i < jotSheet.getLastRow()+1; i++){
        var hojadestino = jotSheetValues[i][1];
        if (jotSheetValues[i][1] == "BRISTOL"){
        var destinationSheet = bristolSheet;
        var destinationSheetValues = bristolSheet.getDataRange().getValues();
        }

        if (jotSheetValues[i][1] == "BARCELONA"){
        var destinationSheet = bcnSheet;
        var destinationSheetValues = bcnSheet.getDataRange().getValues();
        }

        for (var c = 0; c < destinationSheet.getLastColumn(); c++){
        if (destinationSheetValues[0][c] == "Opportunity Name"){
        var dSubmissionDateCol = c;
        }
        if (destinationSheetValues[0][c] == "Num client "){
        var dNumclientCol = c;
        }
        if (destinationSheetValues[0][c] == "Origen reserva "){
        var dOrigenreservaCol = c;
        }
        if (destinationSheetValues[0][c] == "Nom i cognom "){
        var dNomicognomCol = c;
        }
        if (destinationSheetValues[0][c] == "Email client "){
        var dEmailclientCol = c;
        }
        if (destinationSheetValues[0][c] == "Dades de contacte (telfs, adreces,...) "){
        var dDadesdecontacteCol = c;
        }
        if (destinationSheetValues[0][c] == "Data d'entrada "){
        var dDatadentradaCol = c;
        }
        if (destinationSheetValues[0][c] == "Data de sortida "){
        var dDatadesortidaCol = c;
        }
```

```
if (destinationSheetValues[0][c] == "# dies "){
var ddiesCol = c;
}
if (destinationSheetValues[0][c] == "# persones "){
var dpersonesCol = c;
}
if (destinationSheetValues[0][c] == "# adults "){
var dadultsCol = c;
}
if (destinationSheetValues[0][c] == "# nens "){
var dnensCol = c;
}
if (destinationSheetValues[0][c] == "Pais"){
var dPaisCol = c;
}
if (destinationSheetValues[0][c] == "Cost pel client "){
var dCostpelclientCol = c;
}
if (destinationSheetValues[0][c] == "Paga i senyal "){
var dPagaisenyalCol = c;
}
if (destinationSheetValues[0][c] == "Saldo "){
var dSaldoCol = c;
}
if (destinationSheetValues[0][c] == "Pagat "){
var dPagatCol = c;
}
if (destinationSheetValues[0][c] == "Despeses de neteja "){
var dDespesesdenetejaCol = c;
}
if (destinationSheetValues[0][c] == "Despeses de gestió "){
var dDespesesdegestioCol = c;
}
if (destinationSheetValues[0][c] == "Impostos "){
var dImpostosCol = c;
}
if (destinationSheetValues[0][c] == "Guanys "){
var dGuanysCol = c;
}
if (destinationSheetValues[0][c] == "Fiança "){
var dFiancaCol = c;
}
if (destinationSheetValues[0][c] == "Checklist "){
var dChecklistCol = c;
}
if (destinationSheetValues[0][c] == "OPEN"){
var dOPENCol = c;
```

```
        }
        if (destinationSheetValues[0][c] == "Ressenya "){
        var dRessenyaCol = c;
        }
        if (destinationSheetValues[0][c] == "Calendaris actualitzats "){
        var dCalendarisactualitzatsCol = c;
        }
        if (destinationSheetValues[0][c] == "Retorn fiança "){
        var dRetornfiancaCol = c;
        }
        if (destinationSheetValues[0][c] == "Comentari "){
        var dComentariCol = c;
        }
        if (destinationSheetValues[0][c] == "Gestor Avisat"){
        var dGestorAvisatCol = c;
        }
        if (destinationSheetValues[0][c] == "Usuari Avisat"){
        var dUsuariAvisatCol = c;
        }
        }

        for (var j = 1; j < destinationSheet.getLastRow(); j++){
        if (jotSheetValues[i][jIDoportunitatCol] == destinationSheetValues[j][dNumclientCol]){

        //getRange empieza las columnas
        if (jotSheetValues[i][jCostpelClientCol] == "0"){
        destinationSheet.getRange(j+1, dCostpelclientCol+1).setValue("");
        } else if (jotSheetValues[i][jCostpelClientCol] != null){
        destinationSheet.getRange(j+1,
dCostpelclientCol+1).setValue(jotSheetValues[i][jCostpelClientCol]);
        }

        if (jotSheetValues[i][jPagaisenyalCol] == "0"){
        destinationSheet.getRange(j+1, dPagaisenyalCol+1).setValue("");
        } else if (jotSheetValues[i][jPagaisenyalCol] != ""){
        destinationSheet.getRange(j+1,
dPagaisenyalCol+1).setValue(jotSheetValues[i][jPagaisenyalCol]);
        }

        if (jotSheetValues[i][jFiancaCol] == "0"){
        destinationSheet.getRange(j+1, dFiancaCol+1).setValue("");
        } else if (jotSheetValues[i][jFiancaCol] != ""){
        destinationSheet.getRange(j+1,
dFiancaCol+1).setValue(jotSheetValues[i][jFiancaCol]);
        }

        if (jotSheetValues[i][jImpostosCol] == "0"){
```

```
          destinationSheet.getRange(j+1, dImpostosCol+1).setValue("");
          } else if (jotSheetValues[i][jImpostosCol] != ""){
          destinationSheet.getRange(j+1,
dImpostosCol+1).setValue(jotSheetValues[i][jImpostosCol]);
          }

          if (jotSheetValues[i][jDespesesgestioCol] == "0"){
          destinationSheet.getRange(j+1, dDespesesdegestioCol+1).setValue("");
          } else if (jotSheetValues[i][jDespesesgestioCol] != ""){
          destinationSheet.getRange(j+1,
dDespesesdegestioCol+1).setValue(jotSheetValues[i][jDespesesgestioCol]);
          }

          if (jotSheetValues[i][jDespesesnetejaCol] == "0"){
          destinationSheet.getRange(j+1, dDespesesdenetejaCol+1).setValue("");
          } else if (jotSheetValues[i][jDespesesnetejaCol] != ""){
          destinationSheet.getRange(j+1,
dDespesesdenetejaCol+1).setValue(jotSheetValues[i][jDespesesnetejaCol]);
          }
   //END OF EDIT
          }
          }
          //ERASE ROW
          if (i != 1){
          jotSheet.deleteRow(i);
          i--;
          }
  }
}
```

7.2 BIBLIOGRAPHY

Insightly API documentation: https://api.insight.ly/

Spreadhseet Service documentation: https://developers.google.com/apps-script/reference/spreadsheet/

Google Apps Script Guide: https://developers.google.com/apps-script/overview

JotForm user guide: http://www.jotform.com/help/

Google Sites API: https://developers.google.com/google-apps/sites/