Wisdom Of the Krowd – Architecture and engineering of a crowdsourcing application

Miguel Henrique Germano Garrido

Thesis to obtain the Master of Science Degree in

Information Systems and Software Engineering

Supervisors: Prof. Duarte Nuno Jardim Nunes
Prof. Sónia Isabel Ferreira dos Santos Rafael

Examination Committee

Chairperson: Alexandre Paulo Lourenço Francisco
Supervisor: Prof. Duarte Nuno Jardim Nunes
Members of the Committee: Augusto Emanuel Abreu Esteves

November 2019

Acknowledgements

I would like to express my thanks to my supervisor and co-supervisor for the opportunity of experiencing the business environment through this project, and all the support they gave during its development: supervisor Nuno Jardim Nunes from IST and co-supervisor Sónia Rafael from Faculdade de Belas Artes da Universidade de Lisboa;

I would also like to thank Unbabel for supplying us with the necessary tools to complete the project, for their many ideas and tweaks and for their help during various phases of the project, especially the User Testing phases.

I would also like to thank the experts involved during development for all their support, ideas and suggestions: João Ferreira from Faculdade de Belas Artes and Pedro Oliveira, Anthropologist;

To my team for their dedication and perseverance I would like to give big thanks: Lourenço Palma; David Lopes; Márcia Marranita;

For volunteering to our User Testing and giving us very useful suggestions I would like to thank the other IST-Scope Team: Carlos Silva; Diogo Lopes, Miguel Pinto; Ricardo Monteiro;

Finally I am very grateful to all the endless unconditional support from my family and friends

Resumo

Este relatório descreve o desenvolvimento da aplicação *Wisdom Of the Krowd*, uma aplicação que fornece tarefas para os seus utilizadores resolver. As respostas são processadas e a melhor resposta é retornada. Esta aplicação foi possível de se realizar graças ao Projeto IST-Scope, com o objetivo de incorporar colaboração interdisciplinar entre varias especialidades de engenharia para resolver problemas colocados por empresas. É complementado pelo modelo Capstone, um novo tipo de projeto que serve como base para a criação de uma tese, culminando em experiencia académica e individual para o estudante. O relatório explica as características da aplicação, focando-se no Back-end.

A aplicação utiliza uma Arquitectura de Micros-serviços de à sua flexibilidade e permitir escalabilidade, e consiste de três serviços: O Facebook Handler, responsável por comunicar e enviar tarefas aos utilizadores do Facebook; O Client-Side Handler, responsável por gerir a plataforma usada pelos clientes gerir as suas tarefas; O Main Server, responsável pela gestão de tarefas e respostas submetidas. A aplicação é hospedada no Heroku e usa as bases de dados do MongoDB para armazenar o conteúdo de cada serviço. É programado em Python e usa uma API Flask-RESTful e a biblioteca requests para comunicar entre os serviços.

Palavras Chave: Co-Worker, serviço, utilizador, cliente, comunicação, WOK

Abstract

This Project describes the development of Wisdom Of the Krowd, an application that supplies its users with tasks to answer. The responses are processed and the best response is returned. This application is made possible through the IST-Scope project, aiming to incorporate interdisciplinary collaboration across different engineer specialties to solve problems set by companies. It is complemented by the Capstone module, a new project approach that serves as a baseline for the creation of a thesis, culminating in academic and individual experience for the students. The Project goes into detail on the application's features, focusing on the Back-End.

The application consists of three services: The Facebook Handler, able to communicate with the Facebook users and send those tasks; The Client-Side Handler, able to supply the clients with a platform for managing its tasks; The Main Server, able to manage those tasks and responses. The application is hosted on Heroku and uses mongoDB databases to store each services data. It is programed in Python and uses a Flask-RESTful API and the requests library to communicate with the other the service.

Key Words: Co-Worker, service, user, client, communication, WOK

Contents

List of Figures

Acronyms

IST – Instituto Superior Tecnico

WOK – Wisdom Of the Krowd

mLab – mlab Cloud MongoDB Hosting

QE – Quality Estimation

TSV – Tab Separated Value

CSV – Comma Separated Value

JSON – JavaScript Object Notation

Chapter 1 – Introduction

This project is a SCOPE project by *Instituto Superior Técnico*, IST, aiming to incorporate interdisciplinary collaboration across different engineer specialties to solve problems set by companies. It is based on the CAPSTONE Model, a new project approach that serves as a baseline for the creation of a thesis, culminating in academic and individual experience for the students.

The project was made in collaboration with *Unbabel* [1], a Portuguese startup company pioneering in “Translation as a Service”, combining neural Machine Translation, advance Quality Estimation and a crowd of Editors to deliver professional quality translation at a massive scale.

The project is being developed by a group of four students: two, including myself, in software development, one in charge of the design & Branding and one managing the project.

Figure 1.1: SCOPE Team and designated roles

1.1 – Motivation

A problem with outsourcing is the inability to guarantee that the price to pay is consistent with the value of the tasks requested. Some simple checks, like small translations issues or small problems, can be solved without the need of hiring an expert. Typically an expert is hired to solve many tasks for a price, but if those simple checks become most of the tasks they solve then the company could stand to lose funds that would be better applied elsewhere.

With this in mind we were told by *Unbabel* to develop a task solving application able to resolve complex problems through the usage of Wisdom of the Crowd. From this application we could assess the advantages and disadvantages of crowdsourcing those small problems compared to hiring an expert.

1.2 – Objective

This Project report’s objective is to describe my work during the creation of the Wisdom Of the Krowd application and its integration with the features and innovation. My contribution in this work was the development of the Back-End and some small contributions in other aspects of the application.

Chapter 2 – State of the Art

This Chapter discusses crowdsourcing and the various strategies used while developing a crowdsourcing application. It discusses the Chatbot's benefits and various levels of complexity and the Microservices advantages in complex applications.

2.1 – Crowdsourcing

Crowdsourcing is a problem solving technique. It consists on taking a task usually performed by an employee or contractor and outsourcing it to a large group of people. It was coined by Jeff Howe and Mark Robinson to describe how businesses were using the Internet for outsourcing.

Crowdsourcing is not a simple procedure, having a multitude of approaches. When using it, one must consider the best way to achieve their objectives. Jeff Howe organizes those approaches into four basic strategies:

Crowdfunding – Tapping into the crowd's financial resources.

Crowdcreation – Using the crowd to create what you want to sell.

Crowdvoting – Using the crowd to sift through things and vote.

Crowdwisdom – Harnessing the collective intelligence of the crowd to solve complex problems. It is the strategy used in the development of WOK.

The Crowdwisdom Strategy, popularized as "Wisdom of the Crowd" by James Surowiecki, author of "*The Wisdom Of Crowds*" [2], states that the collective opinion of a group of individuals, usually less experienced, is more accurate than that of a single expert. Jeff Howe in his book "*Crowdsourcing: Why the Power of the crowd is driving the future of Business*" [3] describes the benefit of an inexperienced individual as "The untrained are also untainted. Their greatest asset is a fresh pair of eyes". It means that due to that inexperience he is able to view a problem or task in a new, possibly better angle than a trained expert, who would have more difficulty changing their normal perspective.

Examples of the practice of Crowdwisdom can be seen in systems like IEM (Iowa Electronic Markets) [4], a forecasting tool that allows university students to invest real money on predicting future events and gaining depending on their accuracy, for it has been used to correctly predict election results more efficiently than the traditional Polls, as shown in Figure 2.1.

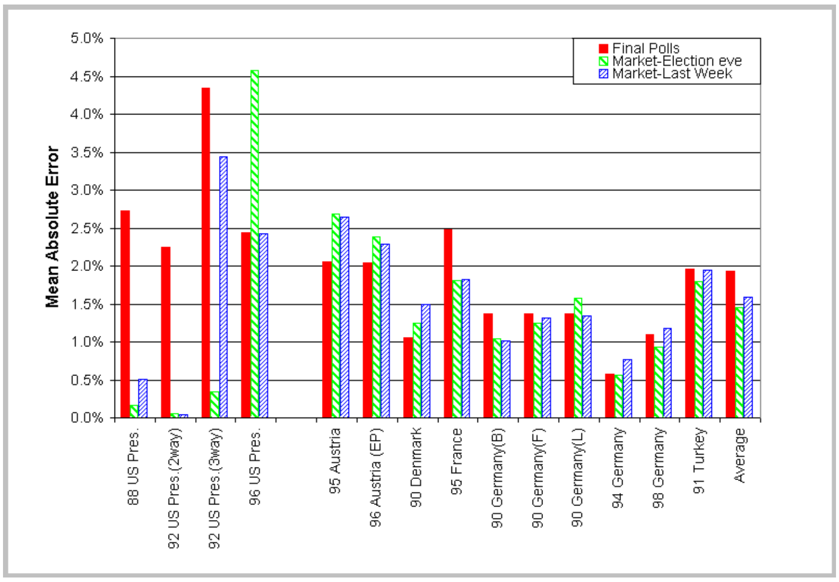


Figure 2.1: Mean Absolute Error of election results comparison between traditional polls(Red) and IEM predictions [5]

Typically, applications using Crowdsourcing strategies reward their users for the work they have done. Those rewards mainly come in the form of virtual rewards like discounts or cosmetics and physical rewards like gift cards or money. This is demonstrated in applications like *Swagbucks* [6], a pc and mobile application that rewards its users with gift cards for use in *Amazon* and other retailers or money through *PayPal* for performing a variety of tasks like surf the web, watch videos, play games, complete surveys and others.

2.2 – Chatbots

A Chatbot is a computer program designed to simulate interactive human conversations through a chat interface by using predefined messages. They have gained a great amount of popularity due to the increasing usage of messenger apps like Slack, Facebook Messenger and others. Chatbots are not limited by time or location, always being available to its users. This makes it appealing to many businesses that desire to improve relations with its users and reduce costs.

A Chatbot can be developed with various levels of interactive complexities:

A contextual Chatbot uses machine learning to study the user’s behavior and previous interactions to determine the best response. It is one the most complex implementations of a Chatbot;

A Keyword recognition Chatbot analyzes the messages received from the user and, based on specific keywords, determines the best reply;

A Buttons and Menu Chatbot is a glorified decision tree hierarchy where its actions are determined based on the buttons and menus accessed by the user. It is a simple implementation as every button corresponds to a specified action. This Chatbot implementation is used in the development of our application.

More often than not the users of a Chatbot tend to become tired of the same responses and patterns. That is why when creating a Chatbot it is given a personality that appeals to its target audience, keeping them motivated by being fun and helping them when requested. One of the first Facebook messenger interactive bots with a personality was *Poncho* the weather cat. It told its users the weather by pretending that it was writing as a cartoon cat and gained great amount of popularity. It has been discontinued but information about its behavior can still be found online [7].

2.3 – Microservices

An application using the Microservice Architecture Style is formed by a collection of loosely coupled services which can be developed, deployed and maintained independently. Each service has its own responsibilities and can communicate with the other services through simple API's to solve a larger more complex business problem.

With every service being independent, there are many benefits for the application:

Teams: Each service can be managed by a small team focused only on that service's objective. This allows for an easier scalability of the project if needed.

Maintenance: If a service is disabled for maintenance or fault detection it does not affect other services, allowing for temporary removal of specific featured from the system without disabling the entire application.

Language: Due to every service communicating with others through an API, every service can have its own technology (language, databases, etc) allowing for the creation of efficient services with the best technology to solve their problems.

.

Chapter 3 – WOK Application – General

This chapter summarizes the application, discussing its key features and system.

3.1 – WOK Branding

The WOK logo is inspired by the traditional Chinese frying pan *wok*. Noodles are usually cooked in woks and in our case represent the "entanglement" of the tasks we have to solve. Our job is to "untangle" those tasks, making difficult problems easier to understand and be dealt with.

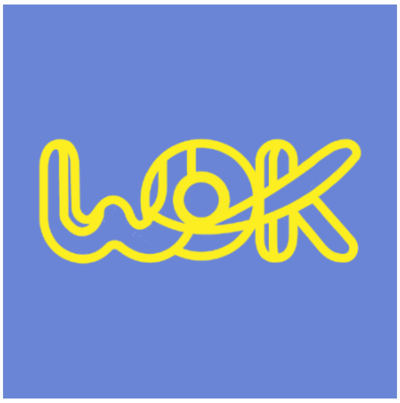


Figure 3.1: Wisdom Of the Krowd Logo

Figure 3.2: Wisdom Of the Krowd Facebook Cover

3.2 – Development Environment

By the suggestion of *Unbabel*, the application’s services are programed in Python 3.6.6 [8]. This language has the benefit of a large quantity of libraries and modules, files containing Python definitions and statements that facilitate the applications’ incorporation with the hosting and database platforms.

3.3 – Hosting

To host the WOK application Heroku [9] was chosen, a Platform as a Service based on a managed container system. It was chosen due to previous experience with the platform, compatibility with the language desired for the creation of the application and performing Load Balancing for the application’s requests through its dynos for scalability.

In Heroku, the deployment of the application is fast and easy to do, using Git [10] to manage App deployments. Heroku also allows for the creation of Environment Variables that are used to manage the application’s URLS and other values.

3.4 – Database

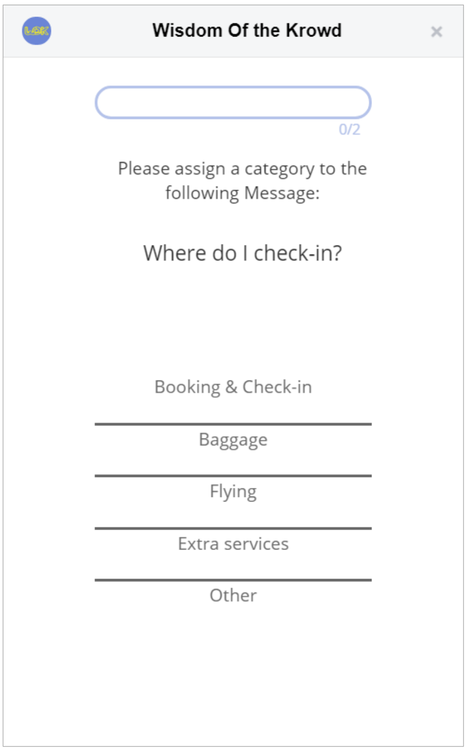
JSON, JavaScript Object Notation, Document-oriented databases were used to store the application’s data due to previous experience with the format and, as the needed data was simple, there was no need for a more complex database . For that mLab [11] was chosen, a “Database-as-a-Service” Platform for MongoDB [12]. It allows for a quick and secure connection to the databases and supports the JSON Format.

3.5 – Tasks

Each task sent through the WOK application is assigned to a limited number of users and given a time limit to solve. This technique allows us to keep the service cost effective. When a task is completed the user receives a reward in the form of virtual coins which can be exchanged for money. The payment system is currently not developed. There are currently two types of tasks supported by the application:

3.5.1 – Categorization Tasks

A user solving a category tasks must specify the task's category and subcategory. The client supplies all the possible categories used for identification and the message to identify.



Wisdom Of the Krowd

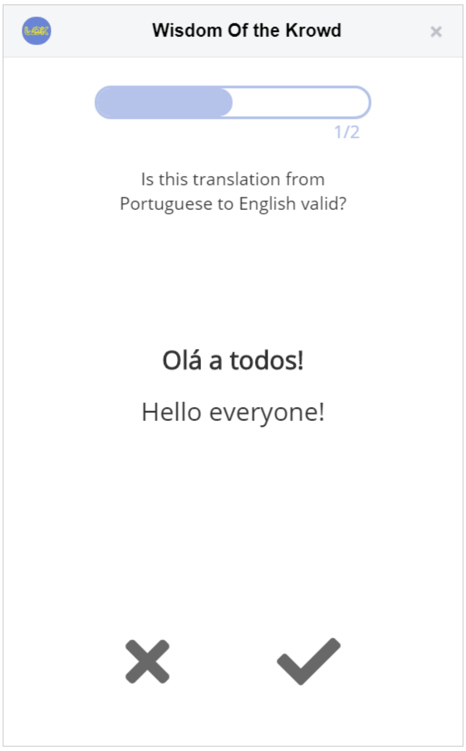
Please assign a category to the following Message:

Where do I check-in?

Figure 3.3: Category Task Example

3.5.2 – Quality Estimation Tasks

Quality Estimation, QE, tasks are simple yes or no questions intended in estimating if the task's submitted objective was accomplished. The client supplies the original objective and extra data, one to two messages depending on the objective. This type of task is flexible as it allows the client to supply tasks ranging from translation verifications to basic validations.



Is this translation from Portuguese to English valid?

Olá a todos!

Hello everyone!

Figure 3.4: Quality Estimation Task Example

3.6 – User’s Co-Workers

A new concept of the WOK application is the use of multiple Co-Workers that the user can choose from and bond with. Each Co-Worker has its own personality and interests studied to fit with the type of target audience that would use the application. This was done with the objective of establishing a friendly environment with our community and to motivate the users to keep resolving tasks.

When developing the Co-Workers’ personality a study of the demographic likely to use the WOK application was performed. It consisted on the creation of user examples based on the application’s target audience to define the ideal behavior of the personalities, those users being determined to be between the ages of eighteen and thirty five years. Each user was defined with their own ethnicity, profession, hobbies and motives for using the application.

To define the Co-Workers’ base personalities the Big Five Theory [13] was used. The Theory classifies peoples’ personality by five personality traits: Extraversion, Agreeableness, Conscientiousness, Neuroticism and Openness.

From the theory and the user’s data five Co-Workers were created, each focusing on a single personality trait and having behaviors that fit them.

The Co-Workers are represented on Facebook by a name and avatar:

3.6.1 – Gal

Gal is the default Persona associated to the user when he registers to the application. This Persona is identified as a genderless bot that gives long talks regarding the universe and life. This persona was based on the Openness personality trait, transparent and curious.

Figure 3.5: Co-Worker Gal’s Avatar

3.6.2 – Julian

Julian is identified as a caring bot that loves to motivate others. This persona was based on the Agreeableness personality trait, always willing to help others.

Figure 3.6: Co-Worker Julian’s Avatar

3.6.3 – Eli

Eli is identified as a very formal bot, work driven and focused on its goals. This persona was based on the Conscientiousness personality trait, hard-working and reliable.

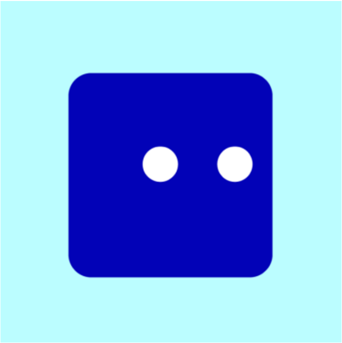


Figure 3.7: Co-Worker Eli’s Avatar

3.6.4 – Jesse

Jesse is identified as an outgoing bot, very communicative and friendly. This persona was based on the Extravert personality trait, being outgoing and social.

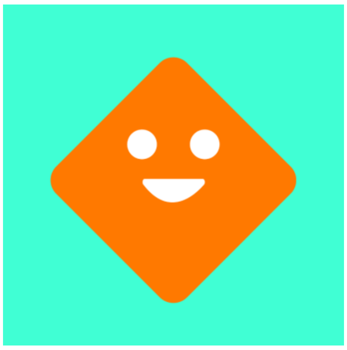


Figure 3.8: Co-Worker Jesse's Avatar

3.6.5 – Sam

Sam is identified as a shy bot, mostly keeping to himself. This persona was based on the Neurotic personality trait, being very self-conscious.

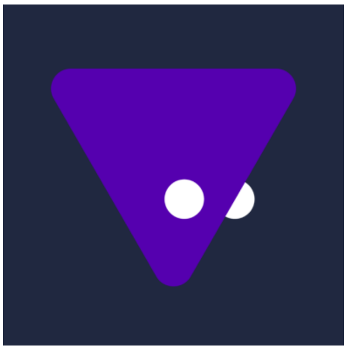
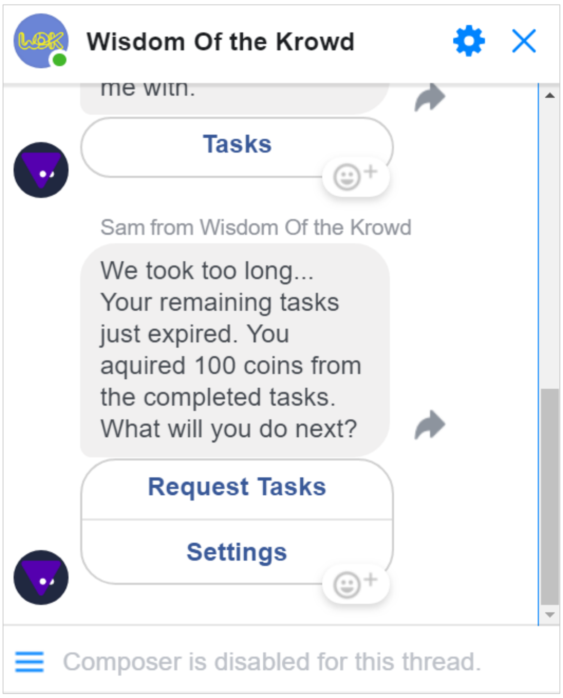


Figure 3.9: Co-Worker Sam's Avatar

3.7 – Notifications

The notification system helps the WOK users stay up to date on task solving by sending them a message reminding them of incomplete, pending and/or new tasks to solve. It also informs the user if the group of tasks submitted has expired by reaching its solving time limit. In the case of the user having partially solved the group of tasks it is sent a message informing them of the reward gained from those completed tasks.

The system sends the messages to the chat using the Co-Workers, each having their own way of notifying the users on any events. The user also has the option of disabling this function in the Settings page which will stop the notifications for the new tasks and pending tasks. It will not, however, disable the messages concerning the time limit.



Wisdom Of the Krowd

me with.

Sam from Wisdom Of the Krowd

We took too long... Your remaining tasks just expired. You aquired 100 coins from the completed tasks. What will you do next?

Composer is disabled for this thread.

Figure 3.10: Notification message of expired tasks

Chapter 4 – WOK Application – User Interaction

This chapter describes the application on the Facebook side, going into detail on the decisions made during its development and the behavior of the possible actions a user can take.

4.1 – Facebook

When searching for a possible chat platform to integrate the WOK application Slack [14] and Line [15] were considered for their popularity and friendly interface, however in the end the Facebook Platform [16] was chosen.

This was owing to Facebook having mostly the same qualifications as the others, namely an API used to send messages, notably the Send API [17]. To send the message to Facebook the Send API send a POST Request to the page and add to the payload the "*messaging_type*" property to specify the message being sent, the "*recipient*" property responsible for identifying the user and the "*message*" property containing the content that will be sent, ranging from button templates to simple messages.

Facebook was also chosen for having the API's and features that allowed the application to become more interactive as desired, such as the Messenger Sender Action [18], able to simulate seen and writing user signals through the bot. This is done through the "*sender_action*" property added to the Send API POST request payload; The Persona API [19], used to define and choose the various Co-Workers to use when sending the message. It associates a name and profile picture to a persona id and uses of the "*persona_id*" property when making a Send API POST request to specify the which Co-Worker should show up when the message is sent; The Messenger Webview [20] used to load web pages inside of the Messenger.

4.2 – Facebook Messenger

For a user to access the WOK application he must first go to its Facebook page, Wisdom Of the Krowd [21], and use the embedded messenger.

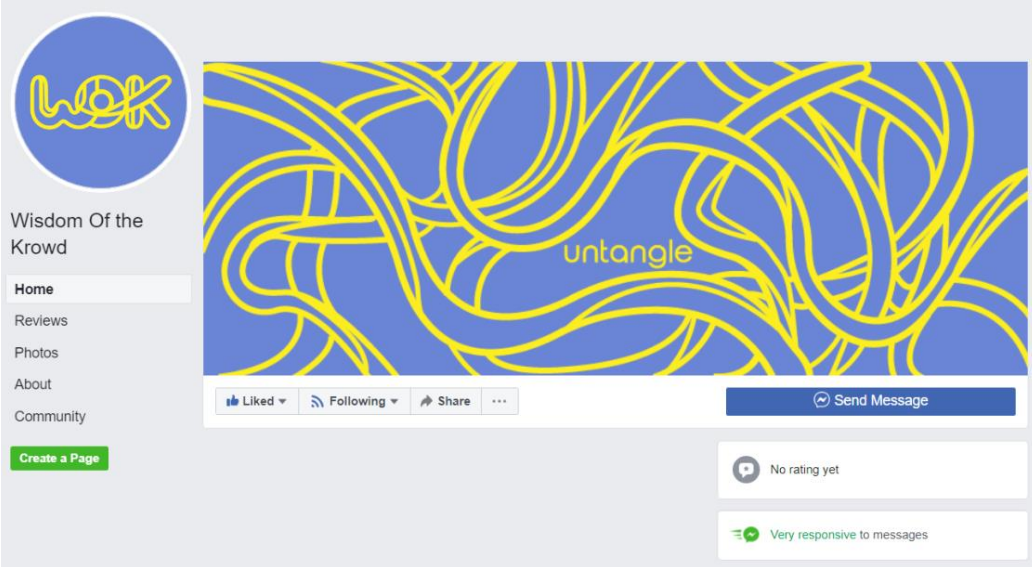
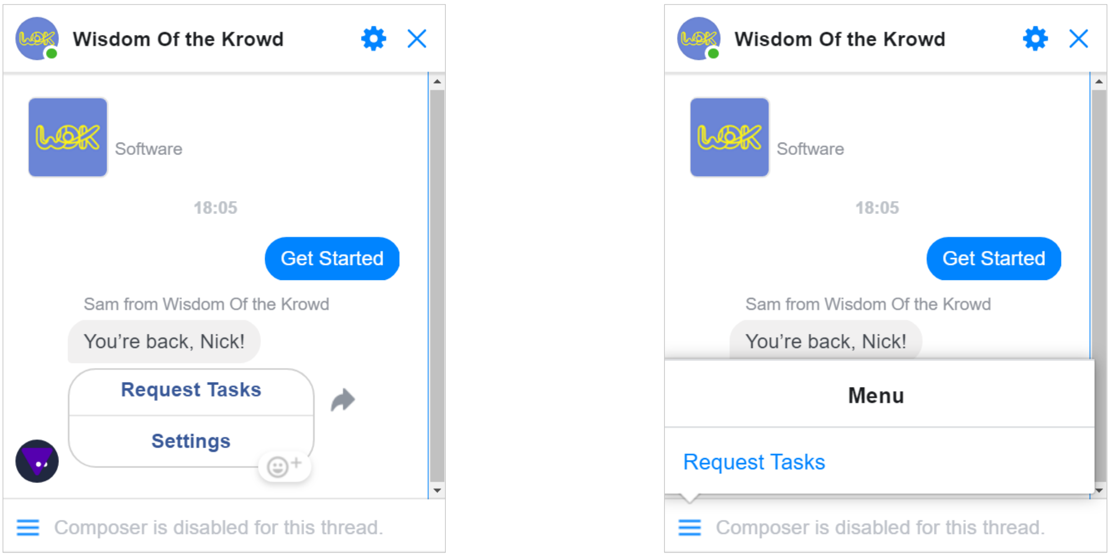


Figure 4.1: Wisdom Of the Krowd's Facebook page

The interaction is done by supplying the user with all the choices it can make either by messenger Buttons [22] or by the use of the Persistent Menu [23], an easily accessible and simple dropdown menu that provides the top-level bot actions to the users. Text Chat is disabled to prevent the users from spamming and overloading the server with requests. To allow for the disabled chat and persistent menu the Messenger Profile API [24] was used to send and set the desired configurations to the messenger.



Sam from Wisdom Of the Krowd

You're back, Nick!

Composer is disabled for this thread.

Sam from Wisdom Of the Krowd

You're back, Nick!

Composer is disabled for this thread.

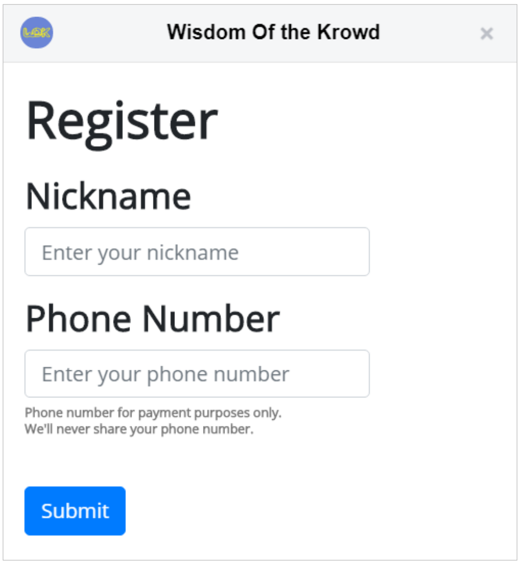
(a) Buttons

(b) Persistent Menu

Figure 4.2: WOK page messenger choices example

4.3 – Registering

When first using the messenger, the user will be sent to a web page through the Webview and will be prompted to fill a Registration Form in order to use the service. It consists on submitting a nickname, used by the Co-Workers to address them, and their Phone Number, used for identification in the system and for sending the money acquired through task solving, currently not developed. The user will receive a message from Gal, our default Co-Worker, greeting the user and showing him the possible actions it can take.



Wisdom Of the Krowd

Register

Nickname

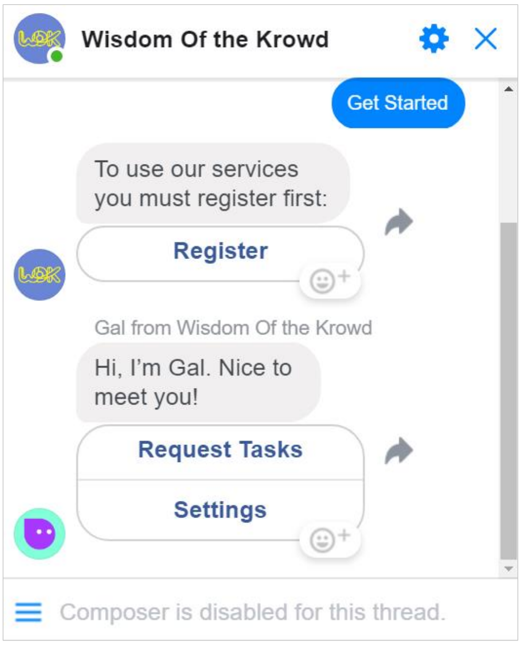
Enter your nickname

Phone Number

Enter your phone number

Phone number for payment purposes only.
We'll never share your phone number.

Figure 4.3: Register Web Page



Wisdom Of the Krowd

Get Started

To use our services you must register first:

Gal from Wisdom Of the Krowd

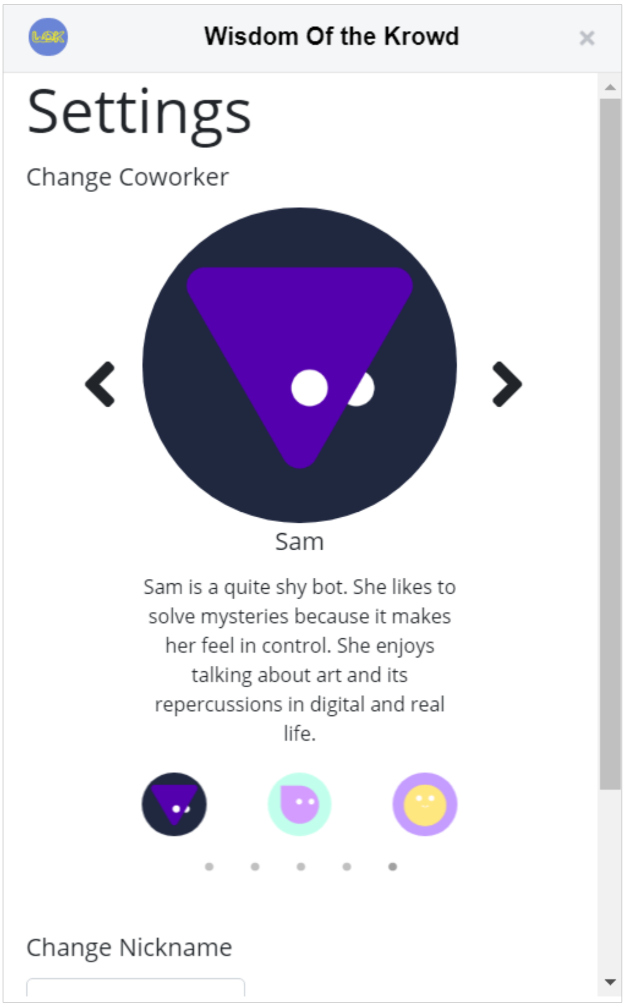
Hi, I'm Gal. Nice to meet you!

Composer is disabled for this thread.

Figure 4.4: Gal welcoming user after registering

4.4 – Settings

After registering, the user can click on the Settings button presented by the Co-Worker to open a web page with all the options it can change. These options allow the user to: Change between the available Co-Workers, each having their avatars displayed and a small description of their habits and personality; modify the user's Nickname used by the Co-Workers and enable or disable the notification messages for the user.



Wisdom Of the Krowd

Settings

Change Coworker

Sam

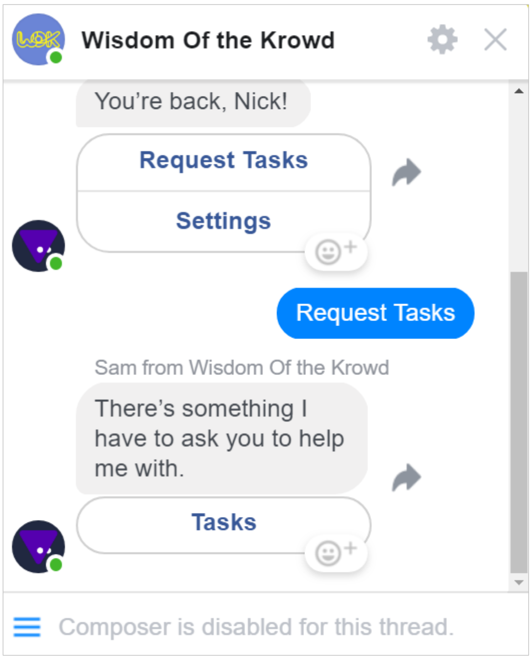
Sam is a quite shy bot. She likes to solve mysteries because it makes her feel in control. She enjoys talking about art and its repercussions in digital and real life.

Change Nickname

Figure 4.5: Settings Web Page showing the Co-Worker Options

4.5 – Request Tasks

When the Request Tasks action is taken by the user, the WOK application will process and submit through a Button and the Webview a group of tasks for him to solve. After the tasks are completed or if the time limit for the completion of the tasks is reached, the Co-Worker will inform the user of the quantity of coins gained through the solved tasks and will prompt him to continue solving tasks by sending him the Buttons with the next possible actions it can take.


Wisdom Of the Krowd

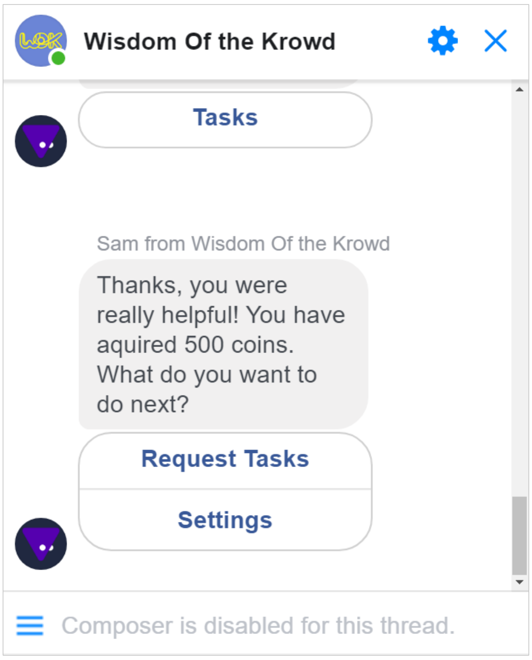
You're back, Nick!

Request Tasks

Sam from Wisdom Of the Krowd

There's something I have to ask you to help me with.

Figure 4.6: Wok page messenger request tasks example


Wisdom Of the Krowd

Sam from Wisdom Of the Krowd

Thanks, you were really helpful! You have aquired 500 coins. What do you want to do next?

Figure 4.7: Wok page messenger tasks completed Example

Chapter 5 – WOK Application – Client Interaction

This chapter mentions the development of the application on the Client's side, demonstrating the web page used by the client to review and submit the tasks to our application and detailing the formats used when exporting and importing tasks.

5.1 – Client Side Handler

The Client-Side Handler creates and manages a Web Page [25] accessible only to the clients, where they can add new tasks and check on the progress of their existing tasks. The site allows the client to filter according to the information they desire to see. They can change between seeing the Category or QE tasks and filter through the time data, description data and response data:

5.1.1 – Time Filter

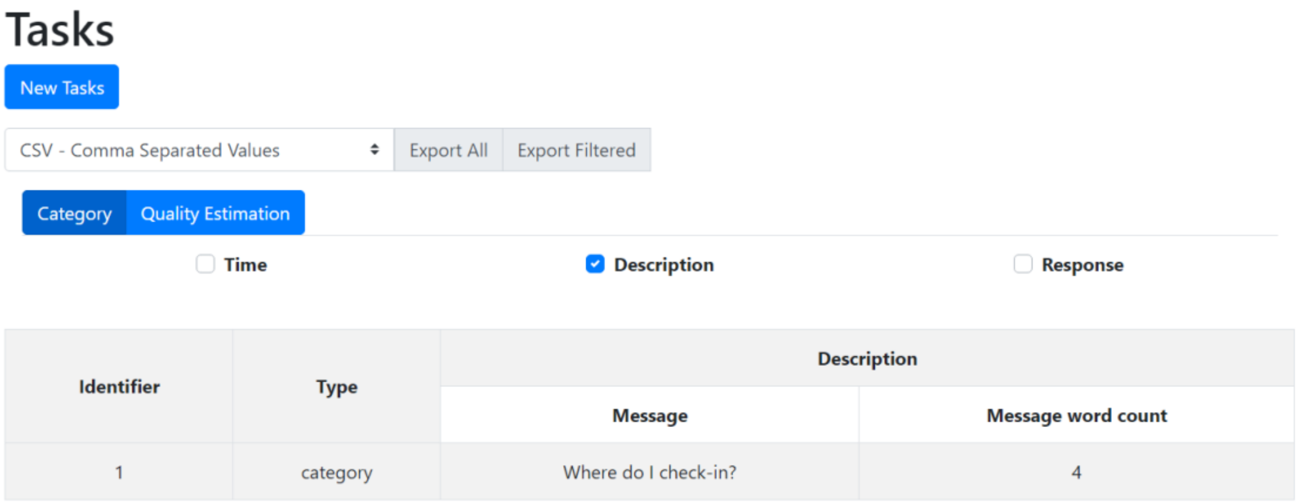
The Time Filter Contains all the time related data. It shows the data of submission of the task, the average response time of all users and the completion date, set when the last response is submitted. The data allows the client to evaluate the difficulty the users had while completing the task.

5.1.2 – Description Filter

The Description Filter contains the important data submitted by the client. In the Category Tasks it consists of the message to be categorized and its word count. The QE Tasks have a similar structure but contains two messages and their word counts. The world counts allows the client to make an estimate on the time it would take an average user to solve that task. This information can then be used in conjunction with the Time data to detect foul play or bot usage whilst solving the tasks and help prevent them.

5.1.3 – Response Filter

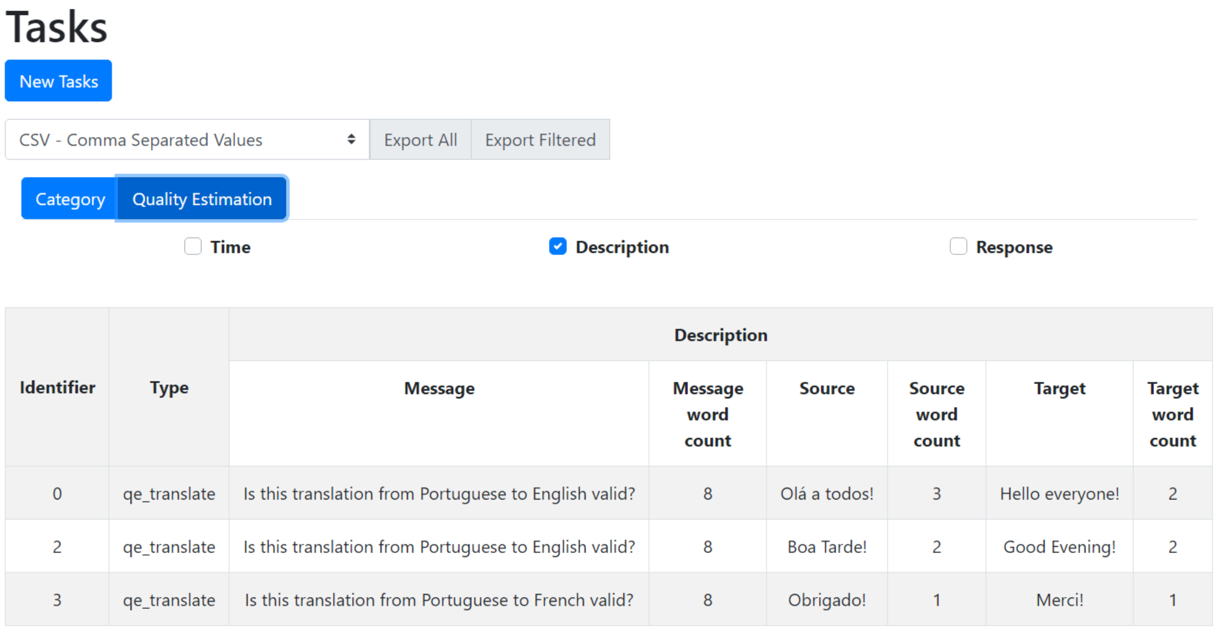
The Response Filter contains the task's completion progress and the results so far. The results cover the recommended response and correctness of the response based on the number of user's that answered it.



☐ Time ☑ Description ☐ Response

Identifier	Type	Description	
		Message	Message word count
1	category	Where do I check-in?	4

Figure 5.1: Client site category task list filtered by description



☐ Time ☑ Description ☐ Response

Identifier	Type	Description					
		Message	Message word count	Source	Source word count	Target	Target word count
0	qe_translate	Is this translation from Portuguese to English valid?	8	Olá a todos!	3	Hello everyone!	2
2	qe_translate	Is this translation from Portuguese to English valid?	8	Boa Tarde!	2	Good Evening!	2
3	qe_translate	Is this translation from Portuguese to French valid?	8	Obrigado!	1	Merci!	1

Figure 5.2: Client site Quality Estimation task list filtered by description

5.2 – Exporting Tasks

The client has the option of exporting its tasks, be them pending and/or solved. When doing so he can choose from exporting all the tasks, which will export two files, one for each task type, each containing all the data from every filter, or export only the filtered content displayed in the web page, exporting a single file containing only the previewed task type and filters.

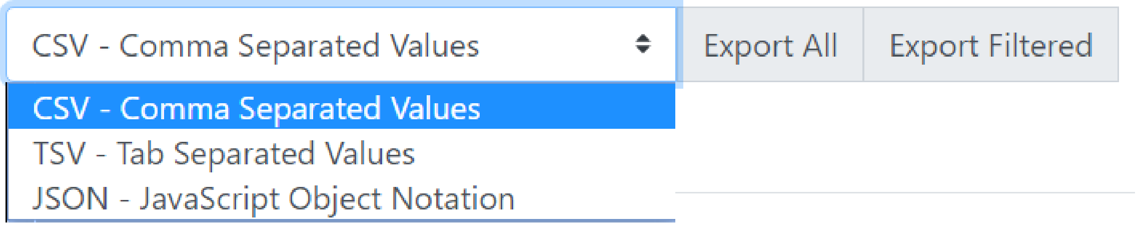
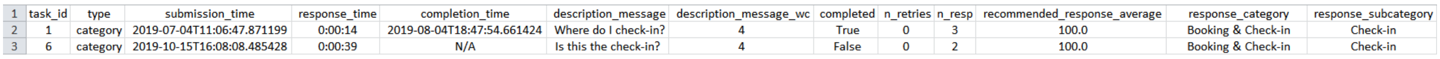

CSV - Comma Separated Values
Export All
Export Filtered
CSV - Comma Separated Values
TSV - Tab Separated Values
JSON - JavaScript Object Notation

Figure 5.3: Exporting options

The task data can be exported in three formats: CSV; TSV and JSON.

5.2.1 – TSV & CSV Format Tasks

The Comma Separated Value, CSV, Format and the Tab Separated Value, TSV format are arranged according to the site's table's Columns.

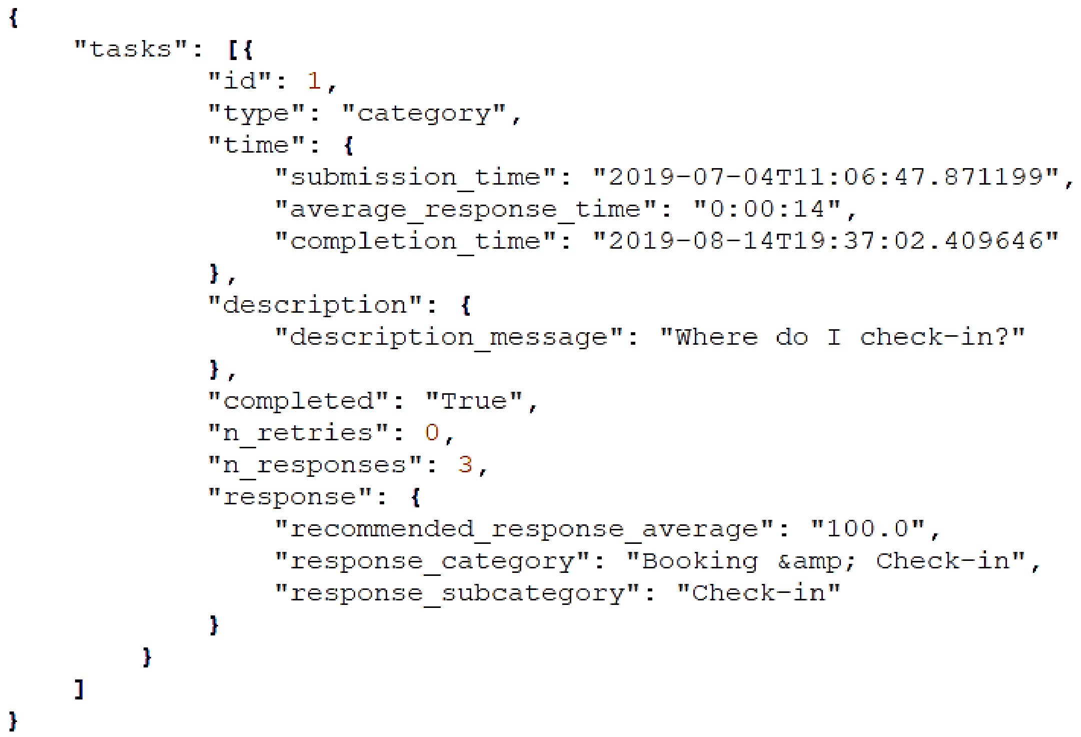


1	task_id	type	submission_time	response_time	completion_time	description_message	description_message_wc	completed	n_retries	n_resp	recommended_response_average	response_category	response_subcategory
2	1	category	2019-07-04T11:06:47.871199	0:00:14	2019-08-04T18:47:54.661424	Where do I check-in?	4	True	0	3	100.0	Booking & Check-in	Check-in
3	6	category	2019-10-15T16:08:08.485428	0:00:39	N/A	Is this the check-in?	4	False	0	2	100.0	Booking & Check-in	Check-in

Figure 5.4: Exported Category tasks in TSV/CSV Format

5.2.2 – JSON Format Tasks

The JSON format follows a similar export format as TSV and CSV, arranging the data according to the columns, but with the difference of storing the data in their respective filter sections.

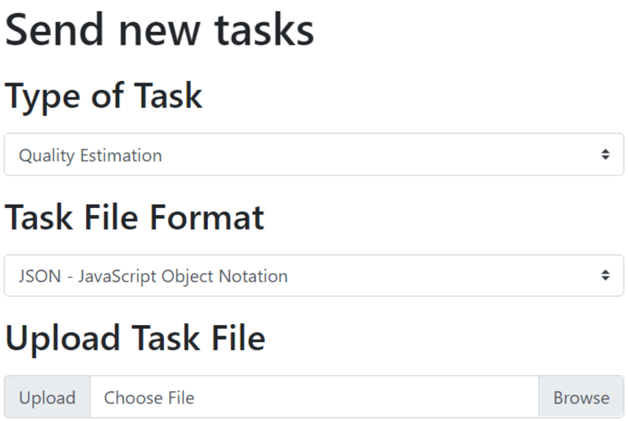


```
{
    "tasks": [{
            "id": 1,
            "type": "category",
            "time": {
                "submission_time": "2019-07-04T11:06:47.871199",
                "average_response_time": "0:00:14",
                "completion_time": "2019-08-14T19:37:02.409646"
            },
            "description": {
                "description_message": "Where do I check-in?"
            },
            "completed": "True",
            "n_retries": 0,
            "n_responses": 3,
            "response": {
                "recommended_response_average": "100.0",
                "response_category": "Booking &amp; Check-in",
                "response_subcategory": "Check-in"
            }
        }
    ]
}
```

Figure 5.5: Exported Category tasks in JSON Format

5.3 – Uploading New Tasks

By clicking the “New Tasks” or by going to our Upload Web Page [26], the client can upload new tasks to the service. Those tasks will then be processed, sent to the users and solved.



Send new tasks

Type of Task

Quality Estimation

Task File Format

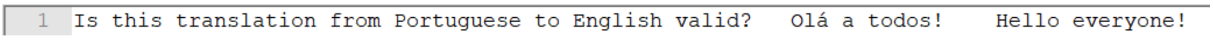
JSON - JavaScript Object Notation

Upload Task File

Figure 5.6: Upload Web Page

5.3.1 – TSV & CSV Format Tasks

Only the QE tasks can be uploaded using the TSV or a CSV Format, this is due to the complexity of the data required for Category tasks. It contains the description of the task in the first column and the QE message(s) in the next two columns. Figure 5.7 establishes the format needed to upload a Quality Estimation task in TSV Format. In this case the newly added task will ask the users to confirm if the translation of “Olá a todos!” in Portuguese to “Hello everyone!” in English was valid.

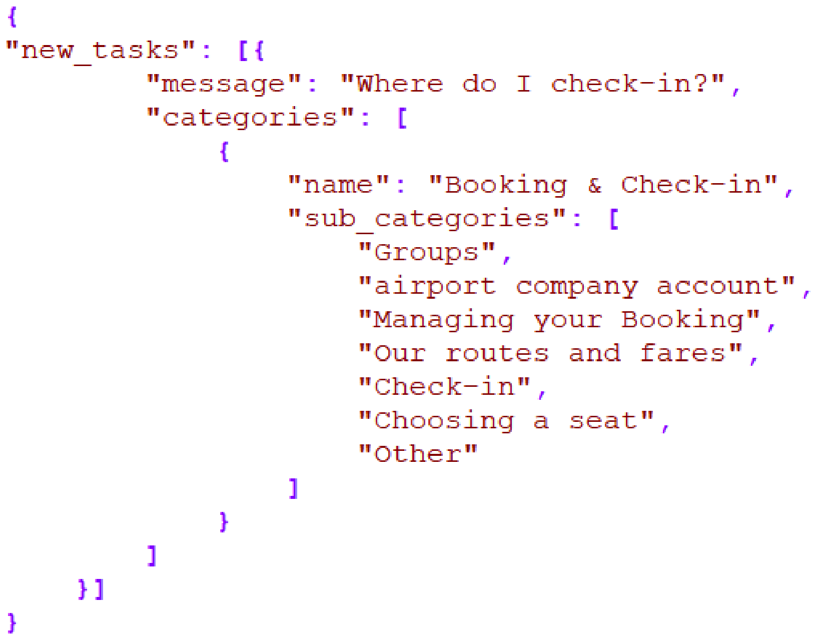


```
Is this translation from Portuguese to English valid?   Olá a todos!    Hello everyone!
```

Figure 5.7: Quality Estimation task for upload in TSV Format

5.3.2 – JSON Format Tasks

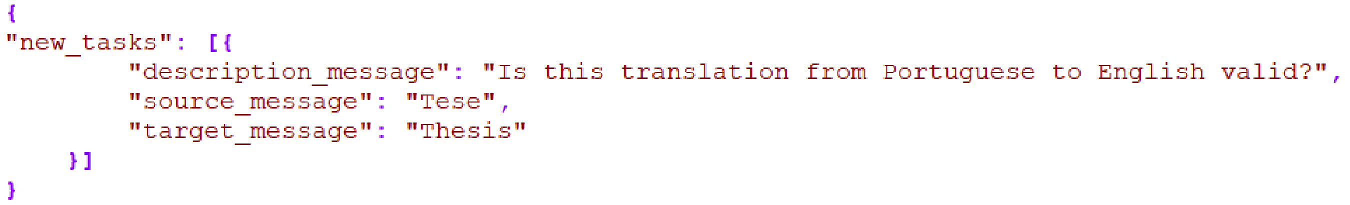
The JSON format can be used to upload all available Formats. Figure 5.8 demonstrates the format needed to upload a Categorization task in this Format. When sent to the application a new task asking to categorize the message “Where do I check-in?” will be created. That task will only contain as the possible categorization options the category “Booking & Check-in” with the sub-categories: “Groups”, “airport company account”, “Managing your Booking”, “Our routes and fares”, “Check-in”, “Choosing a seat” and “Other”.



```
{
"new_tasks": [{
        "message": "Where do I check-in?",
        "categories": [
            {
                "name": "Booking & Check-in",
                "sub_categories": [
                    "Groups",
                    "airport company account",
                    "Managing your Booking",
                    "Our routes and fares",
                    "Check-in",
                    "Choosing a seat",
                    "Other"
                ]
            }
        ]
    }]
}
```

Figure 5.8: Category task for upload in JSON Format

Figure 5.9 demonstrates the format needed to upload a Quality Estimation task in JSON Format. In this case the newly added task will ask the users to confirm if the translation of “Tese” in Portuguese to “Thesis” in English was valid.



```
{
"new_tasks": [{
        "description_message": "Is this translation from Portuguese to English valid?",
        "source_message": "Tese",
        "target_message": "Thesis"
    }]
}
```

Figure 5.9: Quality Estimation task for upload in JSON Format

Chapter 6 – WOK Application –Work Developed

My main responsibility in the project was its back-end, setting up the services and managing the communication with the databases. I also managed the communication between the service and the Facebook Messenger.

6.1 – Architecture

The WOK application is composed of three services, working together to from the application:

The Facebook Handler, responsible for the communications with Facebook, supplying the users with their requested tasks, managing their settings and responses and keeping them up to date by the use of the notification system;

The Client-Side Handler, in charge of the communications between the client web page and their clients, supplying them with the options for exporting or importing tasks to the application. Those imported tasks are sent to the Main Server to be used by the application;

The Main Server, responsible for the management of the tasks and user responses, receiving the tasks from the Client-Side Handler, supplying those tasks to the Facebook user through the Facebook Handler and processing and evaluating all the task's responses in order to send them to the clients through the Client- Side Handler.

Both the Facebook handler and the Main Server have databases containing the data used in their designated operations.

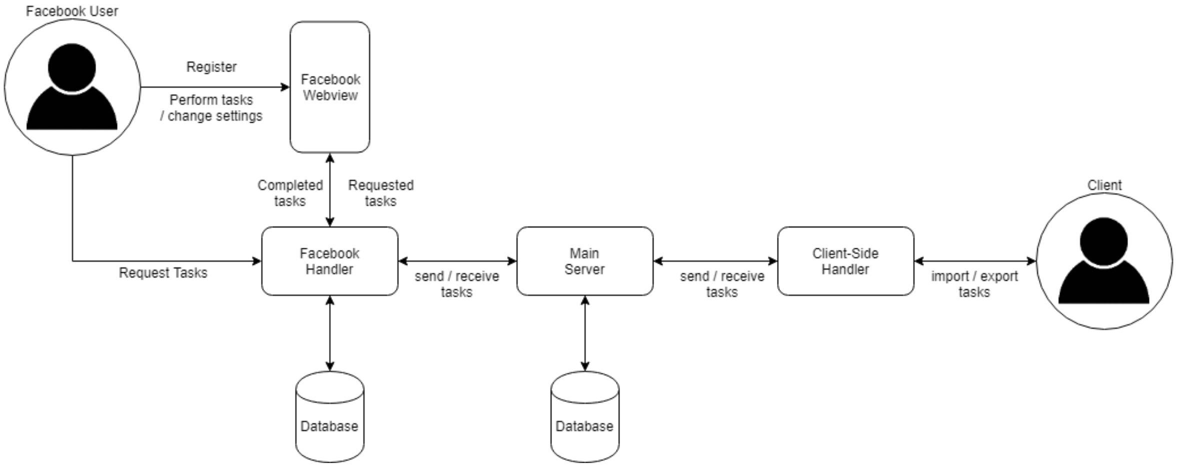


Figure 6.1: Application's architecture

Each service has its own API, developed using Flask-RESTful [27], an extension of the Flask [28] module, a lightweight WSGI web application framework for python. It allows to quickly build an API following the best practices and with minimal setup.

To send messages to the services it uses the facto standard library Requests [29], for there was no need for a more complex library to deal with the simple requests sent between the services and Facebook.

The communication between the databases and the services uses pyMongo [30], a module containing the tools needed to easily communicate with the MongoDB databases created for the application. It is one of the most recommended and simple libraries used to communicate with the MongoDB API.

6.2 – Facebook Messenger

To allow the Facebook Handler to communicate with the Facebook Page messenger a Facebook Bot was created by the name of “WOK Here4U”. It listens to the users requests through a Webhook and sends them on to the application. As a security measure it confirms the destination is valid by the use of an access token sent as a GET request to the root of the handler. If valid it sends the data as a POST to the root.

When a request for a web page is sent from the messenger the application generates a html from templates using Jinja2 [31], a fast and secure tempting language, and sends it to the Facebook Messenger’s Webview. To manage the behavior and responses processed through the Webview the Webview Messenger Extension SDK [32] is used. It allows the handler to control when the Webhook is closed automatically and identify the user that answered the tasks.

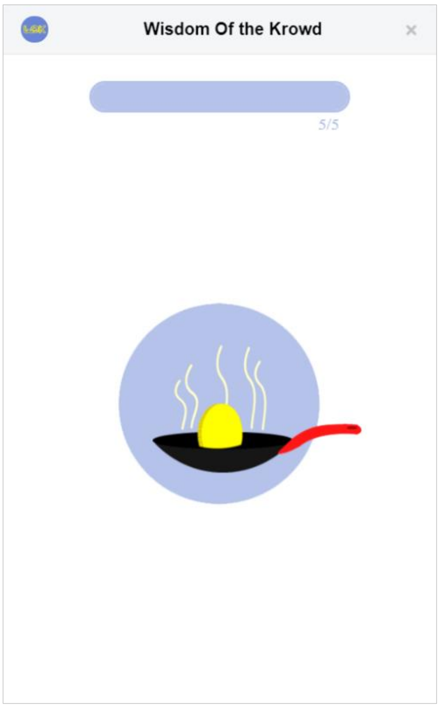


Figure 6.2: Tasks completion page will automatically close after animation is played out

6.3 – Facebook Handler

The Facebook Handler's Back-end is responsible for managing the user's interactions with the Co-Workers and Tasks. It receives user requests from the Facebook messenger and alters the user's data depending on the interaction. It also sends and receives data from the Main Server either to accommodate the user's changed data or for complementing the rendering of the html task templates, using Jinja2, sent to the Facebook's Webview.

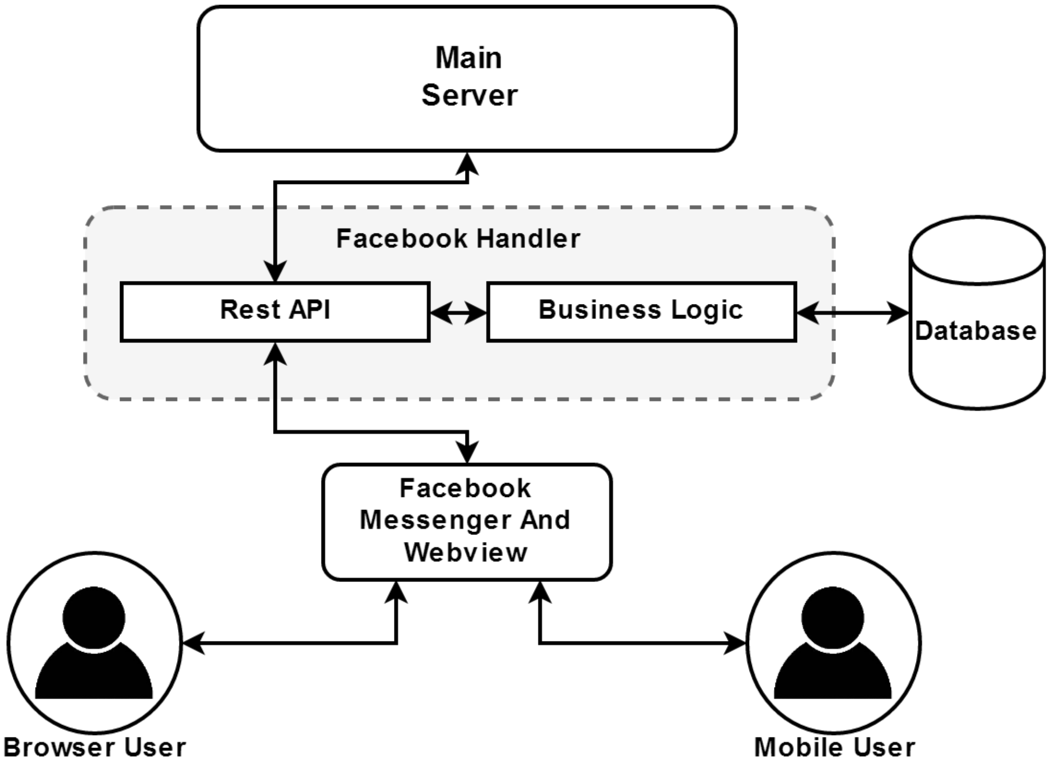


Figure 6.3: Facebook Handler's Business Logic

6.3.1 – Facebook Page Messenger - Handler Communication

When the user presses a button or uses the persistent menu, those requests are sent to the root of the Facebook handler’s API to be processed and replied. To allow the Facebook Handler to distinguish the type of request sent, an Id was associated with every available request sent to the root and added to the requests payload by the use of the Postback Button’s features.

Handler endpoints:

/ (GET) Root:

Used by Facebook to authenticate itself to the service. It receives in the payload a “hub” parameter containing the authentication token used for the validation and the response data needed to signify the validation’s success.

/ (POST) Root:

Used by Facebook to submit requests through its Postback Buttons. After the validation, the Facebook messenger sends to this endpoint the button’s data in the payload. In the button’s data contains an “id” parameter used by the service to identify specify its behavior. The user can subscribe to our service by setting “id” to “STARTED_PAYLOAD” and request or resume pending tasks by setting “id” to “REQUEST_TASKS_PAYLOAD”.

/settings (GET):

Used by Facebook’s Settings URL Button. It is responsible for generating an html page containing the user’s settings, allowing the user to change the CO-Worker being used, change its Nickname or enable/disable notification warnings.

/settings (POST)

Receives the new Settings data in the payload through the parameters “*new_persona_id*” “*new_nickname*” and “*has_notifications*” and changes the user’s settings to the new data. Sends messages to the user according to the changes made.

/web (GET):

Used by Tasks URL Button, a Button sent to the user when requesting tasks. It Generates and returns an html page containing the user's pending task.

/submit (POST):

Submits a payload containing the "*response*" parameter, a JSON object storing the task id and its response. The format of the response is based on the task solved, having "*category_id*" and "*sub_category_id*" in a Categorization task and "valid" in a Quality Estimation task. It sends the response to the Main Server and updates the database to match the coins gained and tasks solved.

/subscribe (GET):

Endpoint used by the Register URL Button. Used when the user is not registered in the application. Generates an html page containing the Registration Form.

/subscribe (POST):

Receives a payload containing the users registration data, sends a subscription request to the Main Server and stores the new user it in the database.

/notify (GET):

Endpoint responsible for the Notification System's behavior. When called, it sends to each user a notification message based on their situation. The situation is evaluated in the following order:

First, it checks if the user is solving tasks and if the time to solve them has expired, if so it sends a message informing that the attempt to solve the tasks have expired. It also informs the user about how much coins it gained by partially solving them.

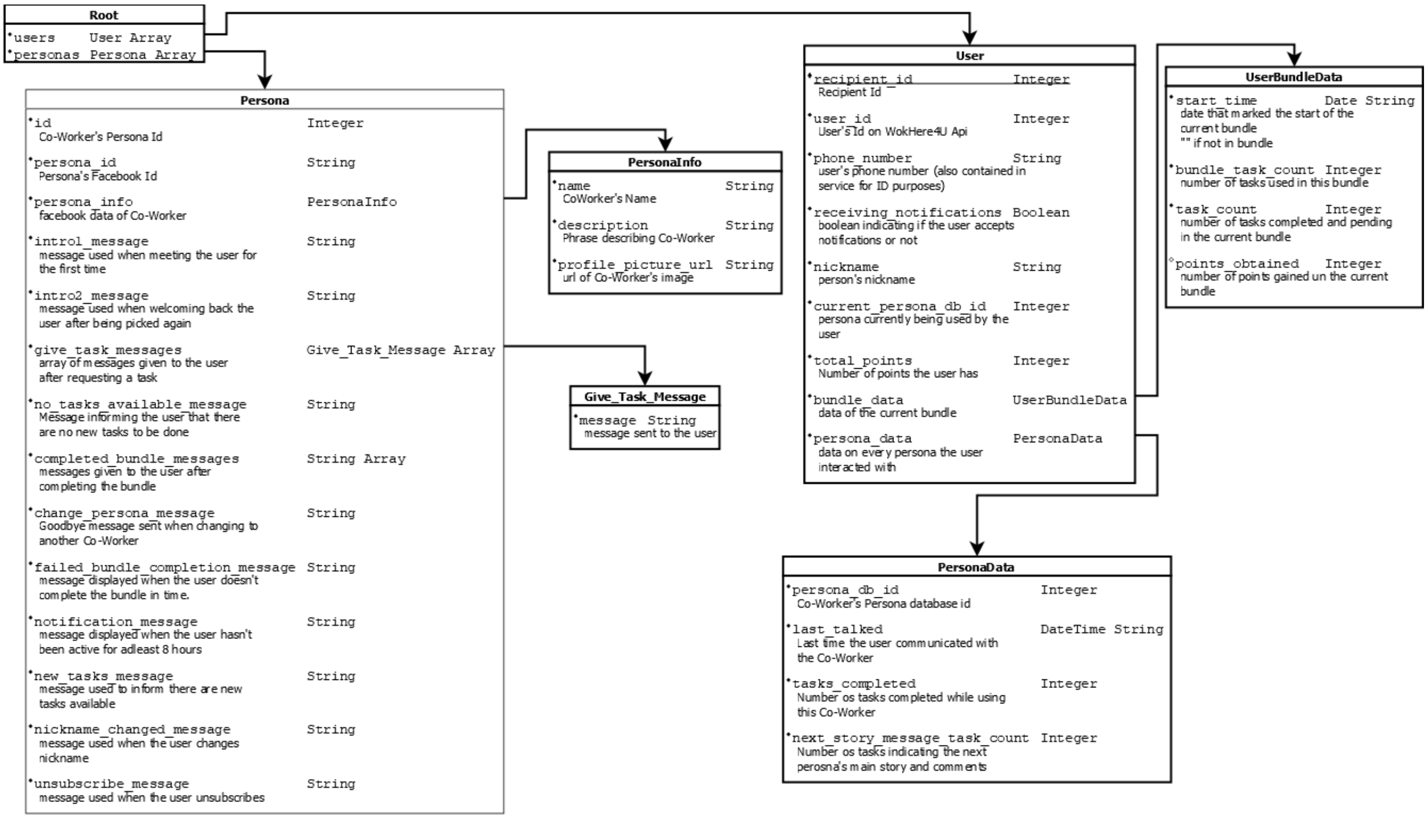
Second, it determined if new tasks were submitted to the application and if the user had solved all the tasks it could until that time, sending a message informing him of new tasks to solve.

Lastly if the user has tasks to solve and has been inactive for a while it gets sent a message informing him that there are still tasks to solve.

To reduce the chance of inconveniencing the user with endless notifications, a cooldown system was developed. If the user is notified, he will have a cooldown interval where notifications that are not related with expired tasks cannot be sent.

6.3.2 – Database Content & Management

Facebook Handler contains a small database, shown in Figure 6.4, focused on keeping track of the users and Co-Worker's interaction data. It stores the user's task progression and contains the user id used to communicate with the Main Server. The separate Id's between the handler and the Main Server allows for multiple handlers to share the same user data.



Root
users User Array
personas Persona Array
Persona
id Integer
Co-Worker's Persona Id
persona_id String
Persona's Facebook Id
persona_info PersonaInfo
facebook data of Co-Worker
intro1_message String
message used when meeting the user for the first time
intro2_message String
message used when welcoming back the user after being picked again
give_task_messages Give_Task_Message Array
array of messages given to the user after requesting a task
no_tasks_available_message String
Message informing the user that there are no new tasks to be done
completed_bundle_messages String Array
messages given to the user after completing the bundle
change_persona_message String
Goodbye message sent when changing to another Co-Worker
failed_bundle_completion_message String
message displayed when the user doesn't complete the bundle in time.
notification_message String
message displayed when the user hasn't been active for adleast 8 hours
new_tasks_message String
message used to inform there are new tasks available
nickname_changed_message String
message used when the user changes nickname
unsubscribe_message String
message used when the user unsubscribes
PersonaInfo
name String
CoWorker's Name
description String
Phrase describing Co-Worker
profile_picture_url String
url of Co-Worker's image
Give_Task_Message
message String
message sent to the user
User
recipient_id Integer
Recipient Id
user_id Integer
User's Id on WokHere4U Api
phone_number String
user's phone number (also contained in service for ID purposes)
receiving_notifications Boolean
boolean indicating if the user accepts notifications or not
nickname String
person's nickname
current_persona_db_id Integer
persona currently being used by the user
total_points Integer
Number of points the user has
bundle_data UserBundleData
data of the current bundle
persona_data PersonaData
data on every persona the user interacted with
UserBundleData
start_time Date String
date that marked the start of the current bundle
"" if not in bundle
bundle_task_count Integer
number of tasks used in this bundle
task_count Integer
number of tasks completed and pending in the current bundle
points_obtained Integer
number of points gained un the current bundle
PersonaData
persona_db_id Integer
Co-Worker's Persona database id
last_talked DateTime String
Last time the user communicated with the Co-Worker
tasks_completed Integer
Number os tasks completed while using this Co-Worker
next_story_message_task_count Integer
Number os tasks indicating the next perosna's main story and comments

Figure 6.4: Facebook Handler's database

6.4 – Main Server

The Main Server is the Core of the system, containing the important data of both the users and clients. It is responsible for keeping track of the user's task solving activities and the client's task submissions.

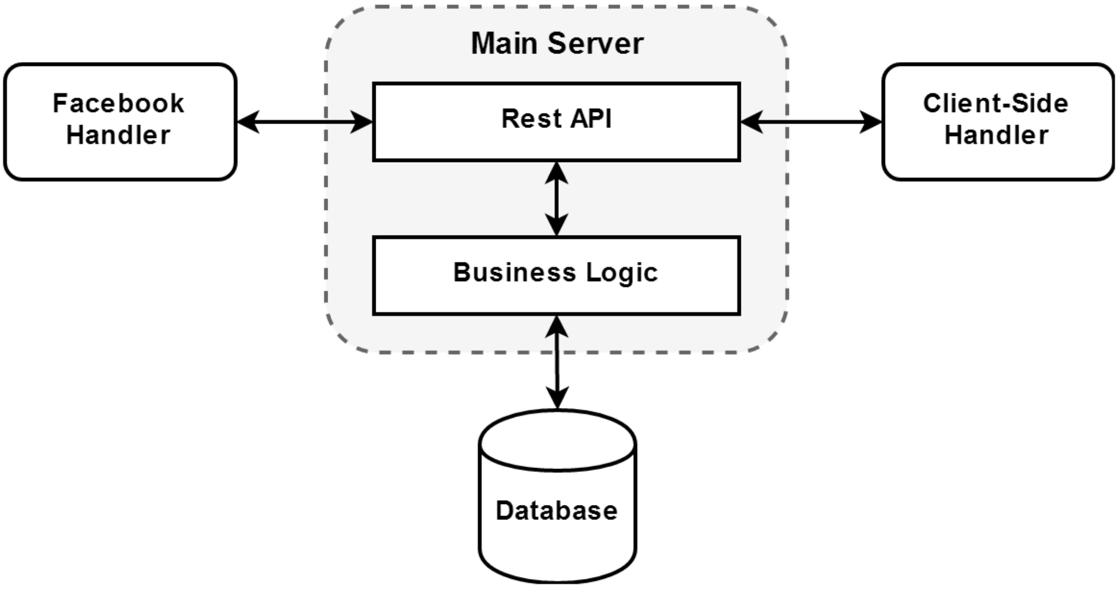


Figure 6.5: Main Server Business Logic

6.4.1 – Facebook Handler – Server Communication – Client-Side Handler

The Facebook Handler sends task requests to the server. It randomly chooses from the available tasks and sends them to the handler. It associates the job of solving that task to the user that requested it, so that even if the user does not solve the task immediately it can pick it up when possible or be reminded through the notification system.

Server Endpoints:

/subscribe (POST):

Adds a new user to the list of users. Used by the handler to register a new user.

/task (GET):

Obtain a simplified version of a task based on a specified user. The simplified version contains the information needed by the handler to render the task.

/task (POST):

Submits a user's response to a task. Used by the Facebook Handler.

/task/cancel (POST):

Cancel's the specified user's pending task, removing the user from the pending list of the task. Used by the Facebook's handler notification system when tasks expire.

/tasks (GET):

Evaluates and returns the specified client's tasks and their current status. The status consist of the evaluation made of the responses of the task: the average response time, number of times task expired, best response, percentage of users that chose the best response and the number of users who solved the task. Used by the Client-Side Handler.

/tasks/available (GET):

Obtains the number of tasks not yet completed, can be filtered by user and limited by a specified max size. Used by the Facebook handler to check a user's available tasks.

/newtask (POST):

Adds the received tasks to the task list and associates them to the client, allowing for users to request and solve them. Used by the client's from the client-side.

6.4.2 – Database Content & Management

The Database of the Main Server contains the tasks information and their user responses. It also contains the users' task completion and pending data.

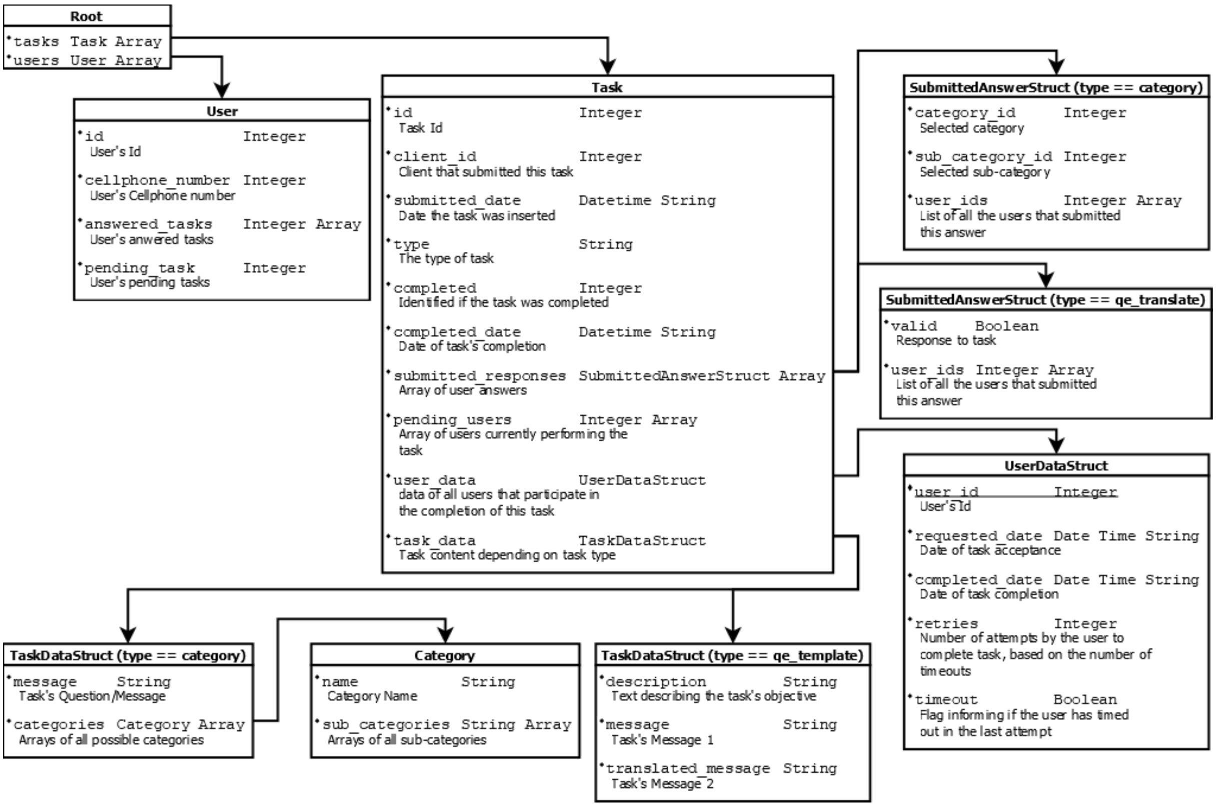

Root
•tasks Task Array
•users User Array
User
•id Integer
User's Id
•cellphone_number Integer
User's Cellphone number
•answered_tasks Integer Array
User's anwered tasks
•pending_task Integer
User's pending tasks
Task
•id Integer
Task Id
•client_id Integer
Client that submitted this task
•submitted_date Datetime String
Date the task was inserted
•type String
The type of task
•completed Integer
Identified if the task was completed
•completed_date Datetime String
Date of task's completion
•submitted_responses SubmittedAnswerStruct Array
Array of user answers
•pending_users Integer Array
Array of users currently performing the task
•user_data UserDataStruct
data of all users that participate in the completion of this task
•task_data TaskDataStruct
Task content depending on task type
SubmittedAnswerStruct (type == category)
•category_id Integer
Selected category
•sub_category_id Integer
Selected sub-category
•user_ids Integer Array
List of all the users that submitted this answer
SubmittedAnswerStruct (type == qe_translate)
•valid Boolean
Response to task
•user_ids Integer Array
List of all the users that submitted this answer
UserDataStruct
•user_id Integer
User's Id
•requested_date Date Time String
Date of task acceptance
•completed_date Date Time String
Date of task completion
•retries Integer
Number of attempts by the user to complete task, based on the number of timeouts
•timeout Boolean
Flag informing if the user has timed out in the last attempt
TaskDataStruct (type == category)
•message String
Task's Question/Message
•categories Category Array
Arrays of all possible categories
Category
•name String
Category Name
•sub_categories String Array
Arrays of all sub-categories
TaskDataStruct (type == qe_template)
•description String
Text describing the task's objective
•message String
Task's Message 1
•translated_message String
Task's Message 2

Figure 6.6: Main Server's database

6.5 – Client-Side Handler

This handler generates a web page used by the application's clients to send new tasks to the Main Server and receive information regarding its task's completion.

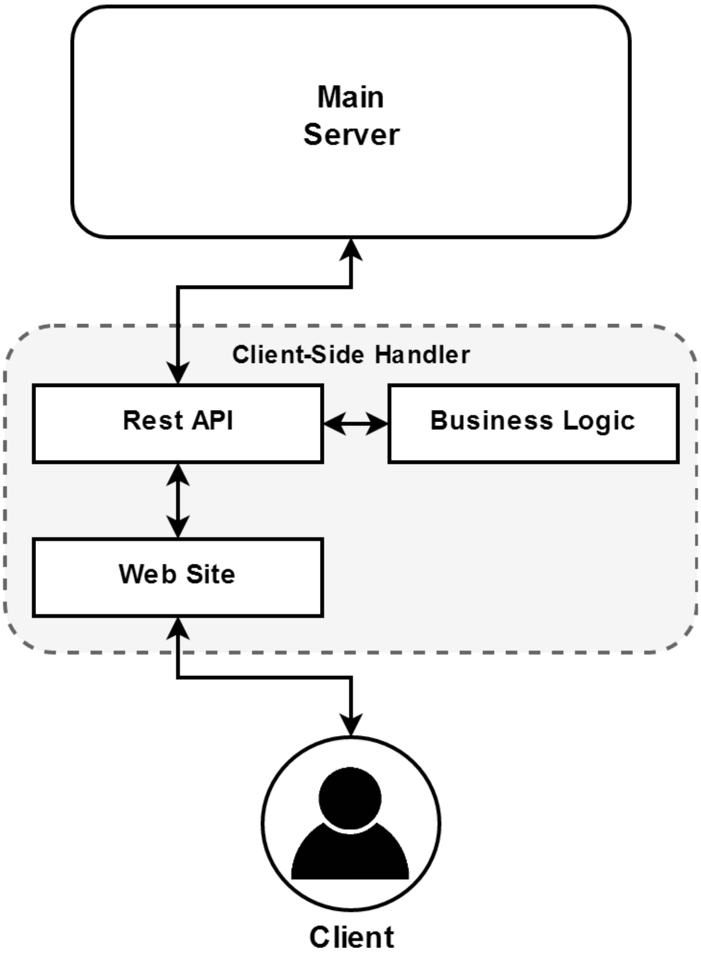


Figure 6.7: Client-Side Business Logic

6.5.1 – Main Server – Client-Side Handler Communication

When the Tasks web page is accessed by a client the Client-Side Handler obtains through a request to the Main Server all the tasks that the client submitted. To do so, it identifies and sends through the payload the client's id. If a task is incomplete the content displayed is adjusted to the available data, showing either the partial analysis made by the Main Server of not available ("N/A") due to insufficient data.

Client Endpoints:

/upload (GET):

Generates and returns an html for submitting tasks.

/upload (POST):

Submits the imported tasks to the Main Server using the “/newtask” endpoint.

/tasks (GET):

Generates and Returns an html page containing the client’s tasks and analysis. It uses the Main Server’s “/tasks” endpoint to obtain the client’s tasks and their analysis.

Chapter 7 – Discussion

The project was managed through the use of the iterative development process.

The prototype of the application had a monolithic architecture, but it was later changed to the micro service architecture due to all the benefits to the application and to allow for handlers for other platforms.

In the phase of the application where the hosting site and the database were not chosen, Ngrok [33] was used as a way to host the application online for testing purposes. A local JSON file was used as a temporary database during that phase.

The money given to the user for each task is 0,10 €. This value was determined from studying the competition and from the User testing responses.

There was some difficulty in the development of the Co-Workers, as creating them to be too human would lead to misrepresentation. For that the avatars were made to represent geometric figures and colors adjusted to their personality.

Due to Facebook's bot policies the bot didn't initially have the permission to communicate as freely as needed with our application, only allowing it to send messages to a user that had used the messenger within 24 hours, disabling the notification system. It also didn't allow for communication with the public unless the Facebook bot was evaluated and validated by Facebook. This was later resolved.

Heroku has a timeout for the activity in order to save processing time. While it reduces the systems consumption during the down time, when a request is made during the down time it will take a minute to boot and answer. This causes problems with Facebook since when he detects the task is taking too long to be answered it assumes it never reached its destination and sends it again, causing the system to process multiple cases of the same request.

The research, development and implementation of the application were done in parallel in order to reassess decisions through user testing and experimentation, allowing for the detection of mistakes more easily and improve the project.

Due to our inexperience our Team had some difficulties during the first User Testing phase. The company taught us the necessary skills to perform efficient User Tests. In our last phase we were able to perform it by ourselves.

Originally, the Co-Workers interacted not only in the chat but also in the Webview. This was scrapped later due to distracting the user in the middle of task solving and drastically increasing the task solving time.

Chapter 8 – Conclusion

The application Wisdom of the Krowd currently allows a client to submit Quality Estimation tasks and Categorization tasks and view their resolution progress. It also allows for Facebook users to subscribe and request tasks to solve.

While the user testing feedback was mostly positive, new user tests on a larger scale are needed to assess the viability of the application on the market.

The next step in the development of the application would be to identify the best payment method and the price value. This can be done by performing more user testing and making a survey on the user's favorite payment methods.

A possible future improvement to the application would be to the communication between the user's and the Co-Workers. This could be arranged by adding a trivia system to the Co-Workers where, after a user solves a number of tasks, the Co-Worker sends a message containing some trivia based on their personality and interests. The Co-Workers could also change their messages depending on events or time of day.

An improvement to the communication between the client and the application would be the addition of a priority and difficulty system to the tasks. The clients could define the priority and difficulty of each task or it could be defined by the use of an algorithm using the task's submitted time and/or similar previous tasks as a base. The higher priority and difficulty tasks could reward the users with extra coins.

Other improvements would be to the Security and performance of the application, optimizing task management for multiple handlers, identifying users using the application through various platforms and support for multiple clients.

This project was a challenging but enriching experience. It was my first time working in a business environment. It was also the first time in a group project where I collaborated with colleagues from other specialties.

Bibliography

[1] Unbabel - https://unbabel.com/about/ - Last seen 09-10-2019

[2] Surowiecki, J. (2005) The Wisdom Of Crowds

[3] Howe, J. (2009) Crowdsourcing: Why the Power of the Crowd Is Driving the Future of Business

[4] IEM - https://iemweb.biz.uiowa.edu/ - Last seen 21-10-2019

[5] IEM Polls accuracy - https://iemweb.biz.uiowa.edu/media/accuracy.html - Last seen 21-10-2019

[6] Swagbucks - https://www.swagbucks.com/ - Last seen 09-10-2019

[7] Poncho Information - https://phandroid.com/2016/05/27/download-poncho-for-android/ - Last Seen 22-10-2019

[8] Python 3.6.6 - https://www.python.org/downloads/release/python-366/ - Last Seen 08-07-2019

[9] Heroku - https://www.heroku.com/ - Last Seen 09-07-2019

[10] Git - https://git-scm.com/ - Last Seen 09-07-2019

[11] MLab - https://mlab.com/ - Last Seen 09-07-2019

[12] MongoDB - https://www.mongodb.com/what-is-mongodb - Last Seen 09-10-2019

[13] John, O. P., Srivastava, S. (1999) Big Five Inventory

[14] Slack - https://slack.com/intl/pt-pt/ - Last Seen 30-10-2019

[15] Line - https://line.me/en/ - Last Seen 30-10-2019

[16] Facebook Platform - https://www.facebook.com/ - Last Seen 09-10-2019

[17] Send API - https://developers.facebook.com/docs/messenger-platform/reference/send-api - Last Seen 09-07-2019

[18] Sender Actions - https://developers.facebook.com/docs/messenger-platform/send-messages/sender-actions/ - Last Seen 09-07-2019

[19] Persona API - https://developers.facebook.com/docs/messenger-platform/send-messages/personas/ - Last Seen 09-07-2019

[20] Webview - https://developers.facebook.com/docs/messenger-platform/webview/ - Last Seen 08-07-2019

[21] Wisdom Of the Krowd Facebook Page - https://www.facebook.com/Wisdom-Of-the-Krowd-1072603452901711/ - Last Seen 08-07-2019

[22] Facebook Buttons - https://developers.facebook.com/docs/messenger-platform/send-messages/buttons - Last Seen 08-07-2019

[23] Persistent Menu - https://developers.facebook.com/docs/messenger-platform/reference/messenger-profile-api/persistent-menu/ - Last Seen 08-07-2019

[24] Messenger Profile API - https://developers.facebook.com/docs/messenger-platform/reference/messenger-profile-api/ - Last Seen 09-07-2019

[25] Client-Side Handler Page - http://wokhere4u-client.herokuapp.com/tasks/ - Last Seen 09-10-2019

[26] Client-Side Handler Import Page - http://wokhere4u-client.herokuapp.com/upload/ - Last Seen 09-10-2019

[27] Flask-RESTful - https://flask-restful.readthedocs.io/en/latest/ - Last Seen 08-07-2019

[28] Flask - http://flask.pocoo.org/ - Last Seen 08-07-2019

[29] Requests - https://requests.kennethreitz.org/en/master/ - Last Seen 30-10-2019

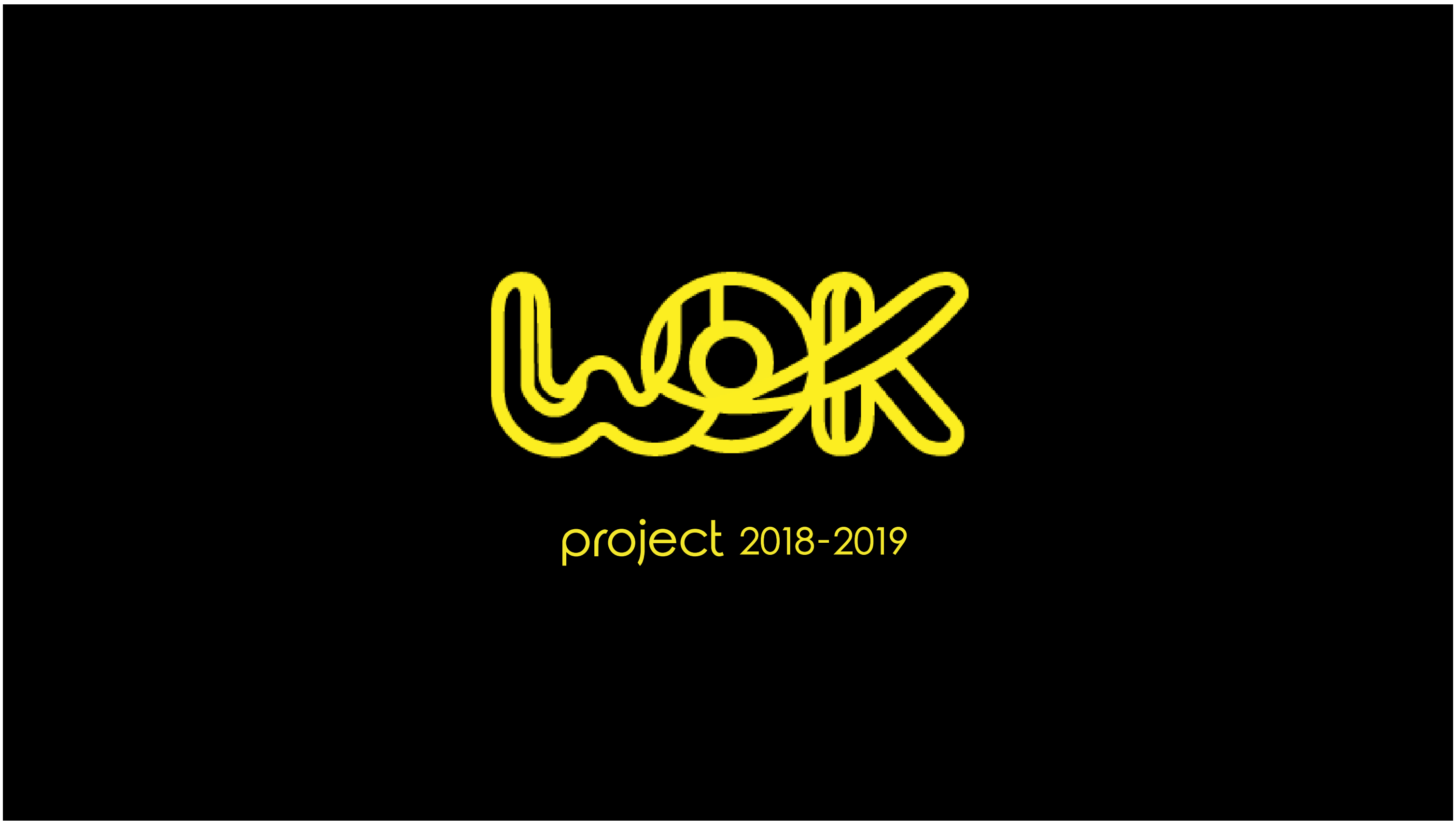
[30] PyMongo - https://api.mongodb.com/python/current/ - Last Seen 09-07-2019

[31] Jinja2 - https://jinja.palletsprojects.com/en/2.10.x/ - Last Seen 12-10-2019

[32] Messenger Extension SDK - https://developers.facebook.com/docs/messenger-platform/webview/extensions - Last Seen 12-10-2019

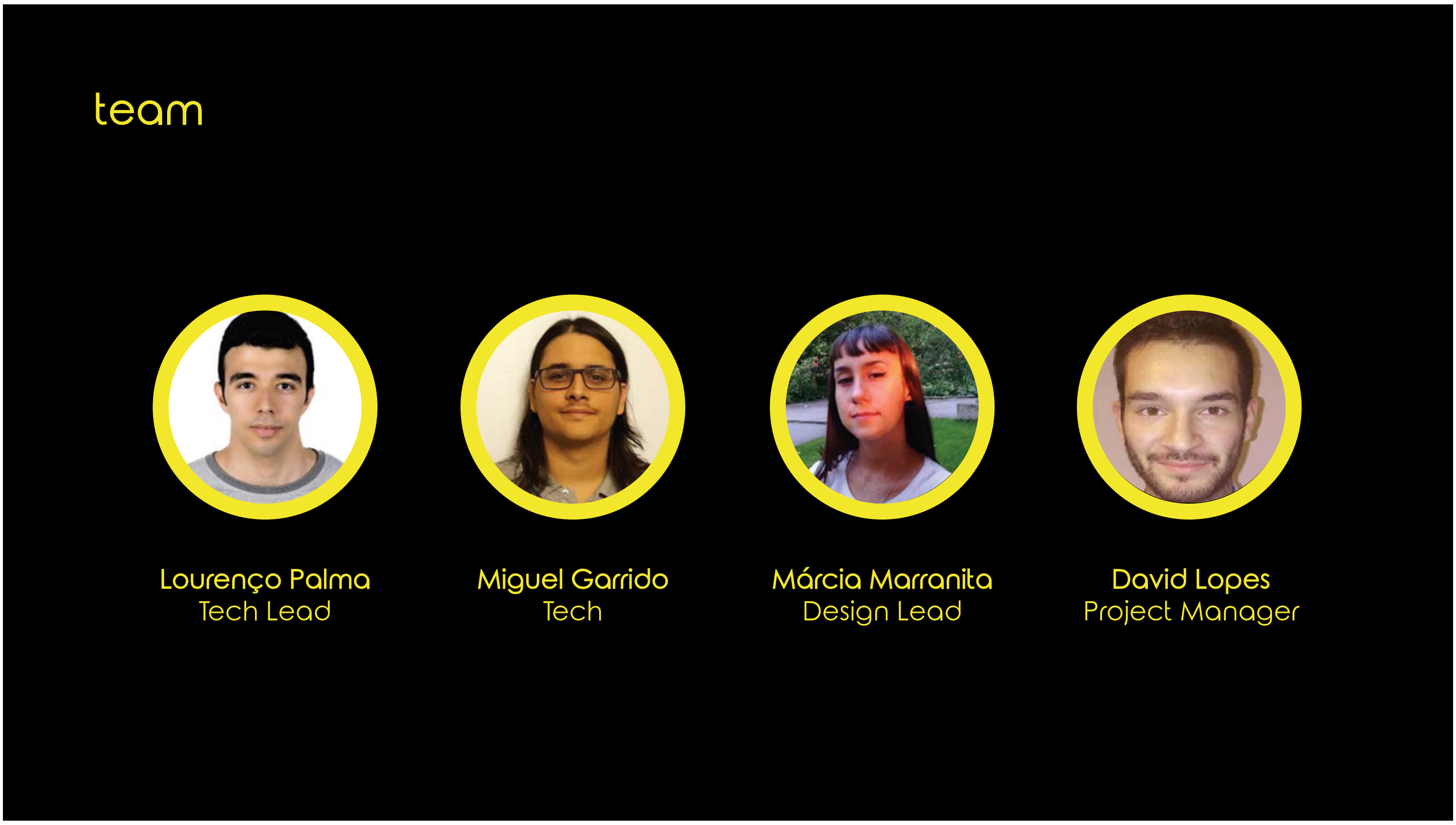
[33] Ngrok - https://ngrok.com/ - Last Seen 08-07-2019

Attachments – Group Report



WOK

project 2018-2019



team

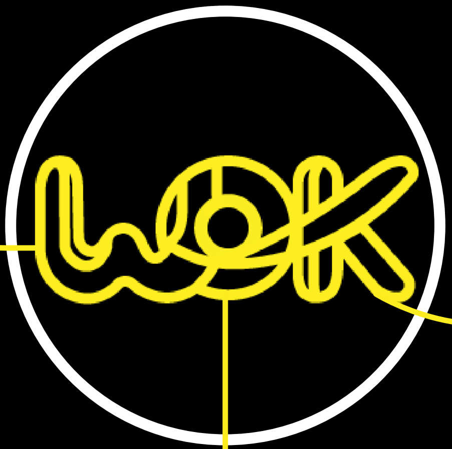
Lourenço Palma
Tech Lead

Miguel Garrido
Tech

Márcia Marranita
Design Lead

David Lopes
Project Manager

branding



shape
noodles

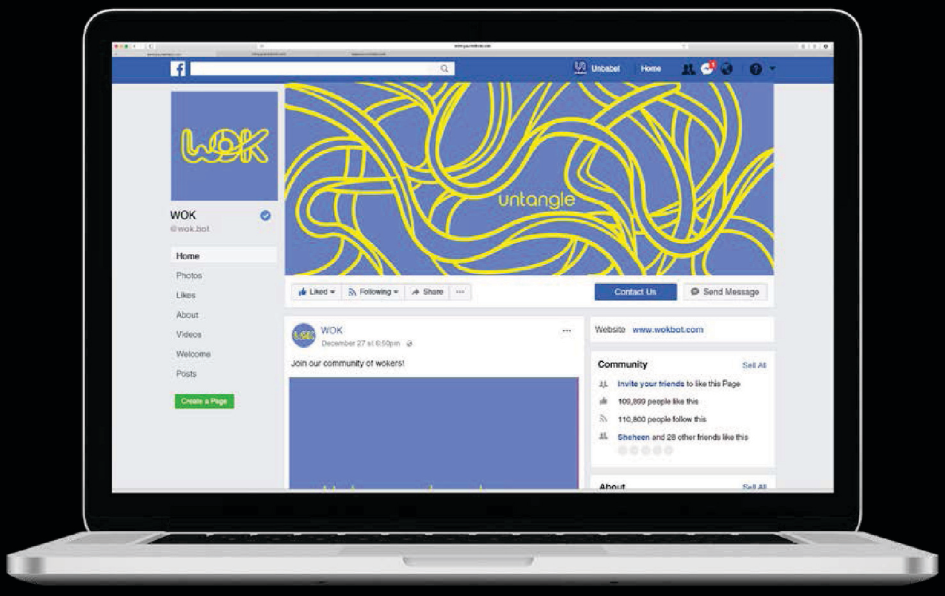
yellow colour
saffron

WOK
frying pan

untangle
(slogan)

free from a tangled or twisted state;
make (something complicated or confusing)
easier to understand or deal with.

facebook page



WOK
December 27 at 6:50pm

Join our community of wokers!

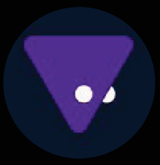
Invite your friends to like this Page

109,899 people like this

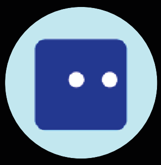
110,800 people follow this

Sheheen and 28 other friends like this

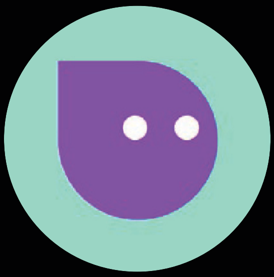
avatars



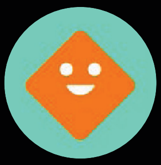
Sam



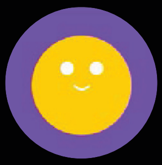
Eli



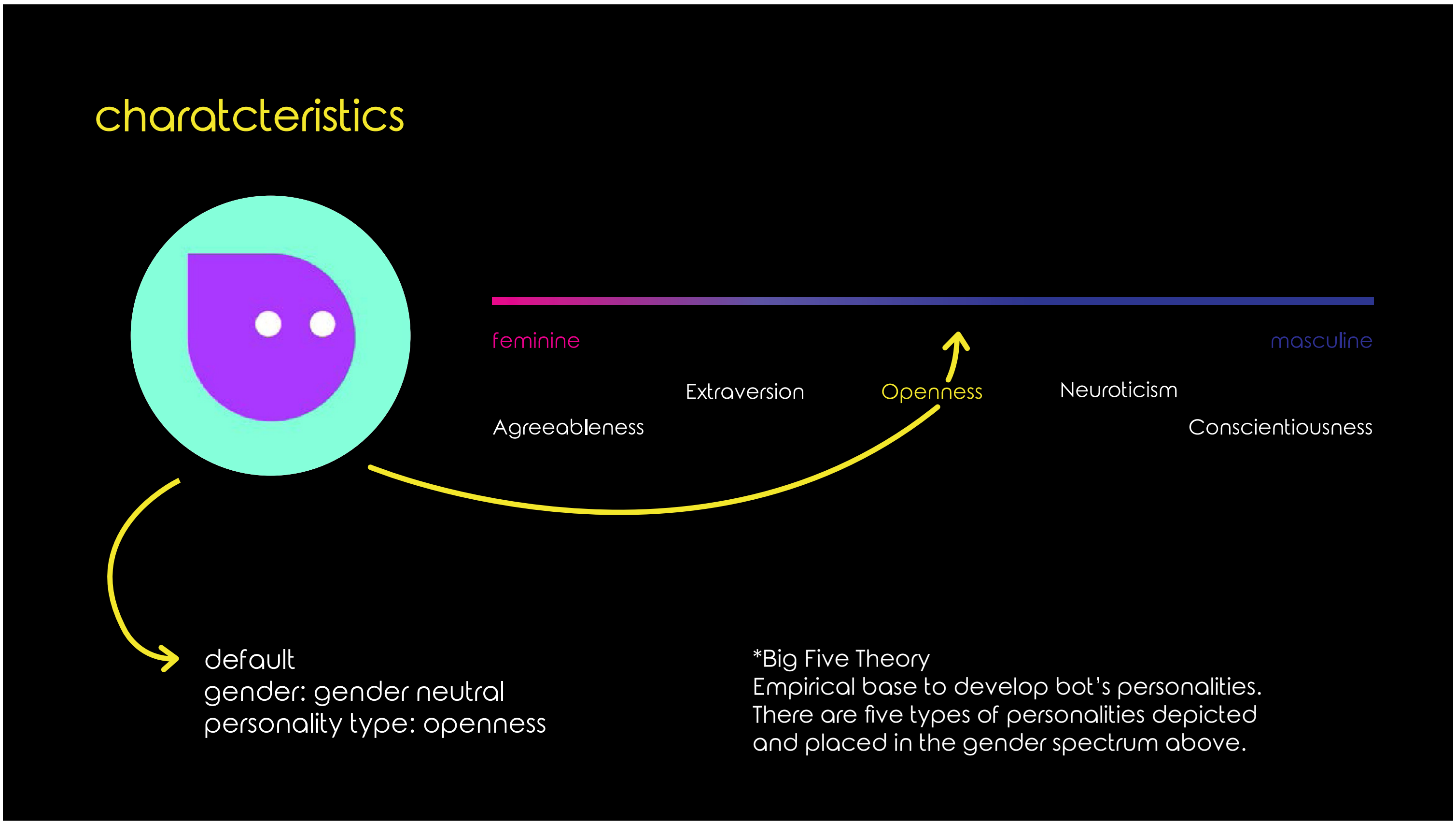
Gal



Jesse



Julian



charatcteristics

feminine

masculine

Extraversion

Openness

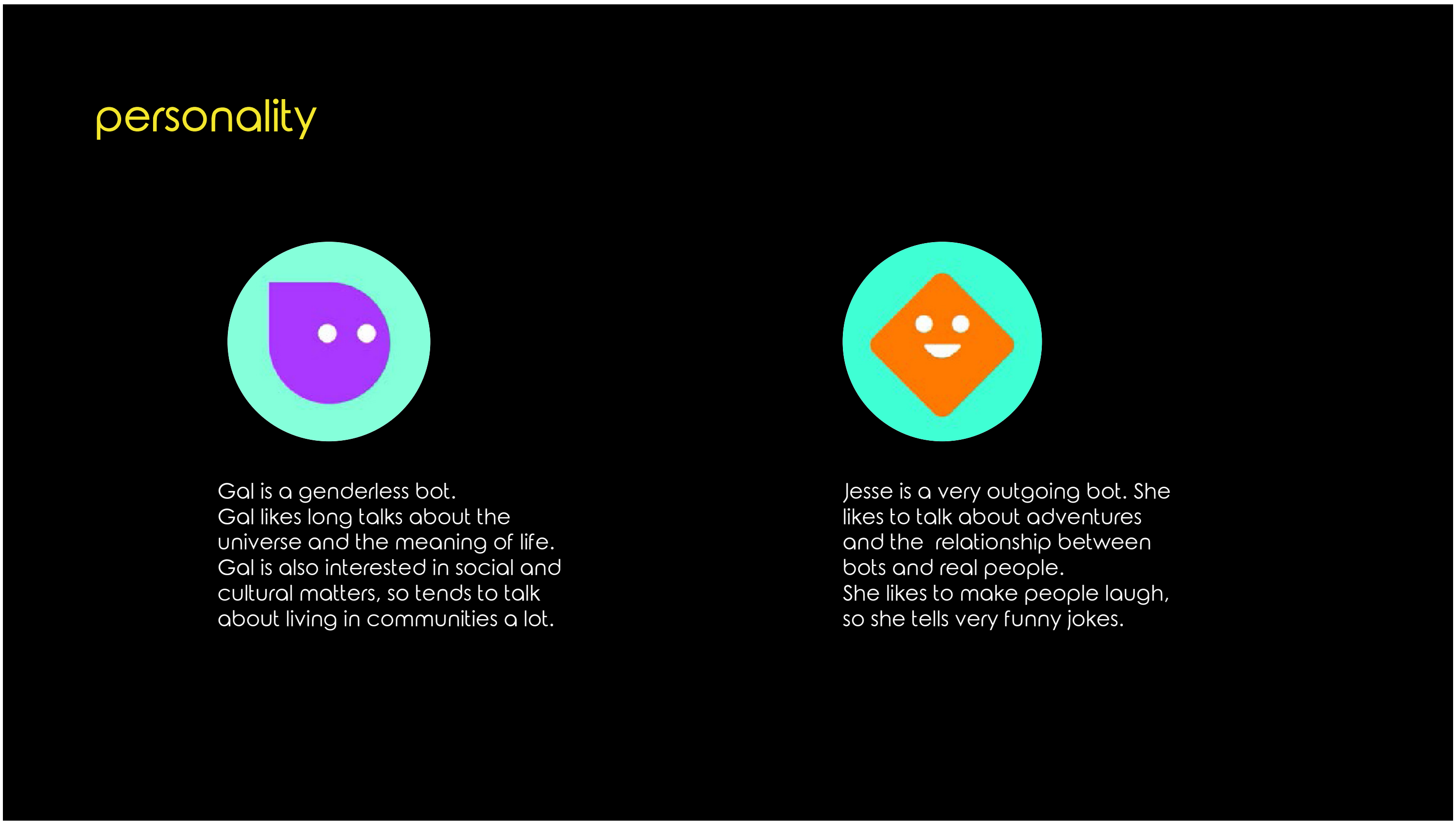
Neuroticism

Agreeableness

Conscientiousness

default
gender: gender neutral
personality type: openness

*Big Five Theory
Empirical base to develop bot's personalities.
There are five types of personalities depicted and placed in the gender spectrum above.

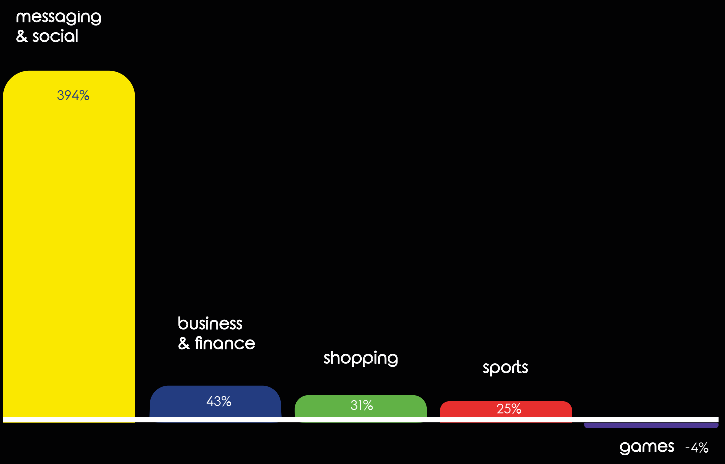


personality

Gal is a genderless bot.
Gal likes long talks about the universe and the meaning of life.
Gal is also interested in social and cultural matters, so tends to talk about living in communities a lot.

Jesse is a very outgoing bot. She likes to talk about adventures and the relationship between bots and real people.
She likes to make people laugh, so she tells very funny jokes.

messaging services



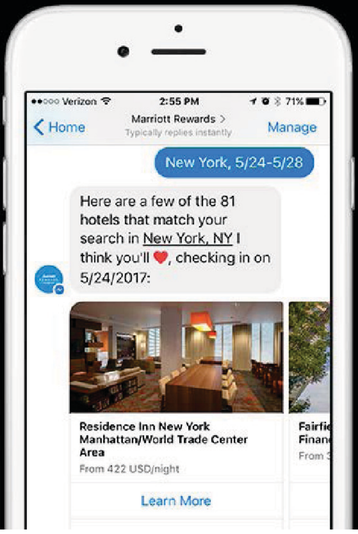
why messaging services?

1. app time spent growth 69%
2. user growth

chat bots

Bot applications are many:
1. customer support, for marketing;
2. share news;
3. financial market.

ex.: Marriott International



New York, 5/24-5/28

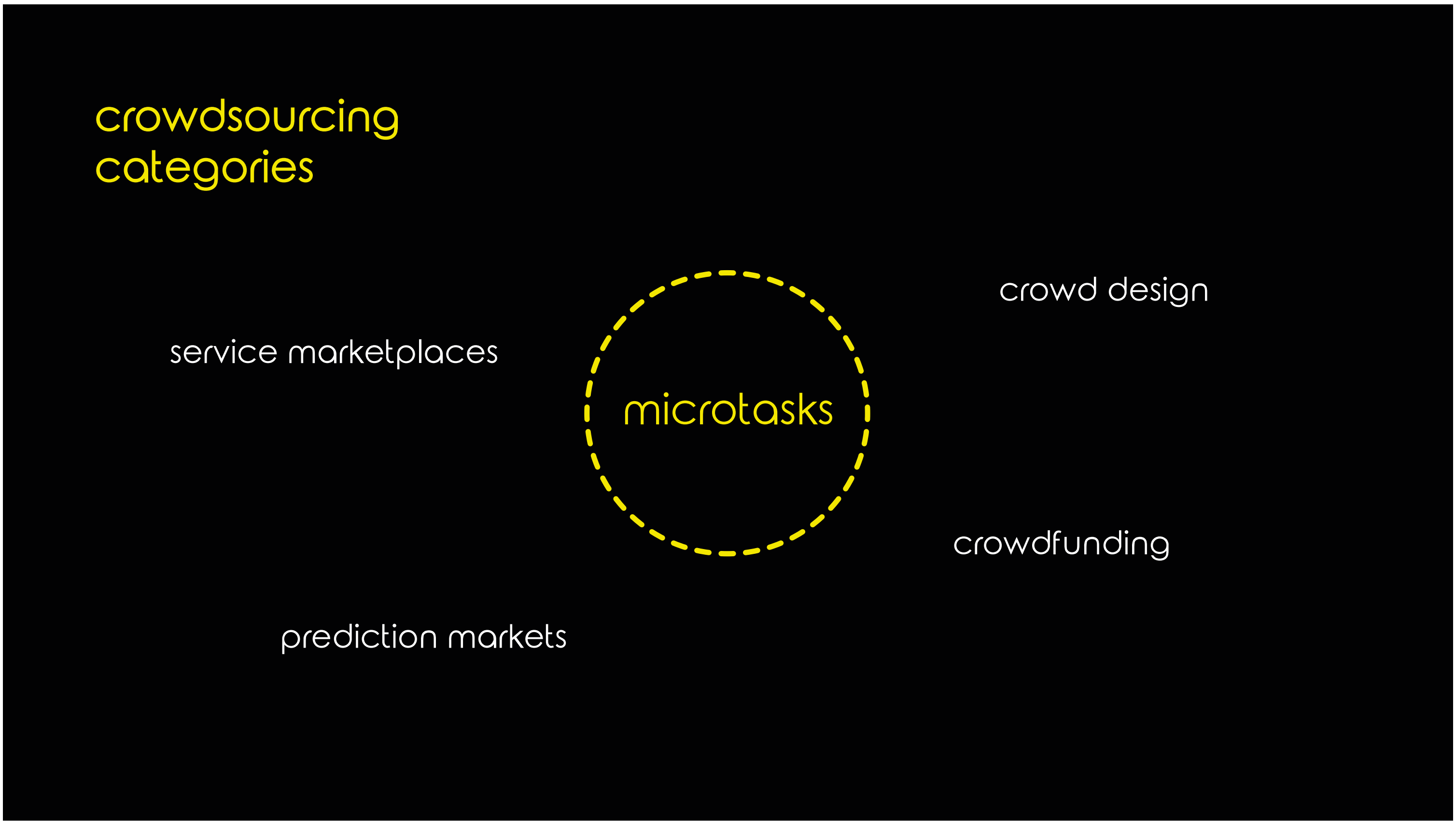
Here are a few of the 81 hotels that match your search in New York, NY I think you'll ❤, checking in on 5/24/2017:

Residence Inn New York Manhattan/World Trade Center Area
From 422 USD/night

crowdsourcing

"the act of taking a job traditionally performed by a designated agent (usually an employee) and outsourcing it to an undefined, generally large group of people in the form of an open call"

Jeff Howe



crowdsourcing categories

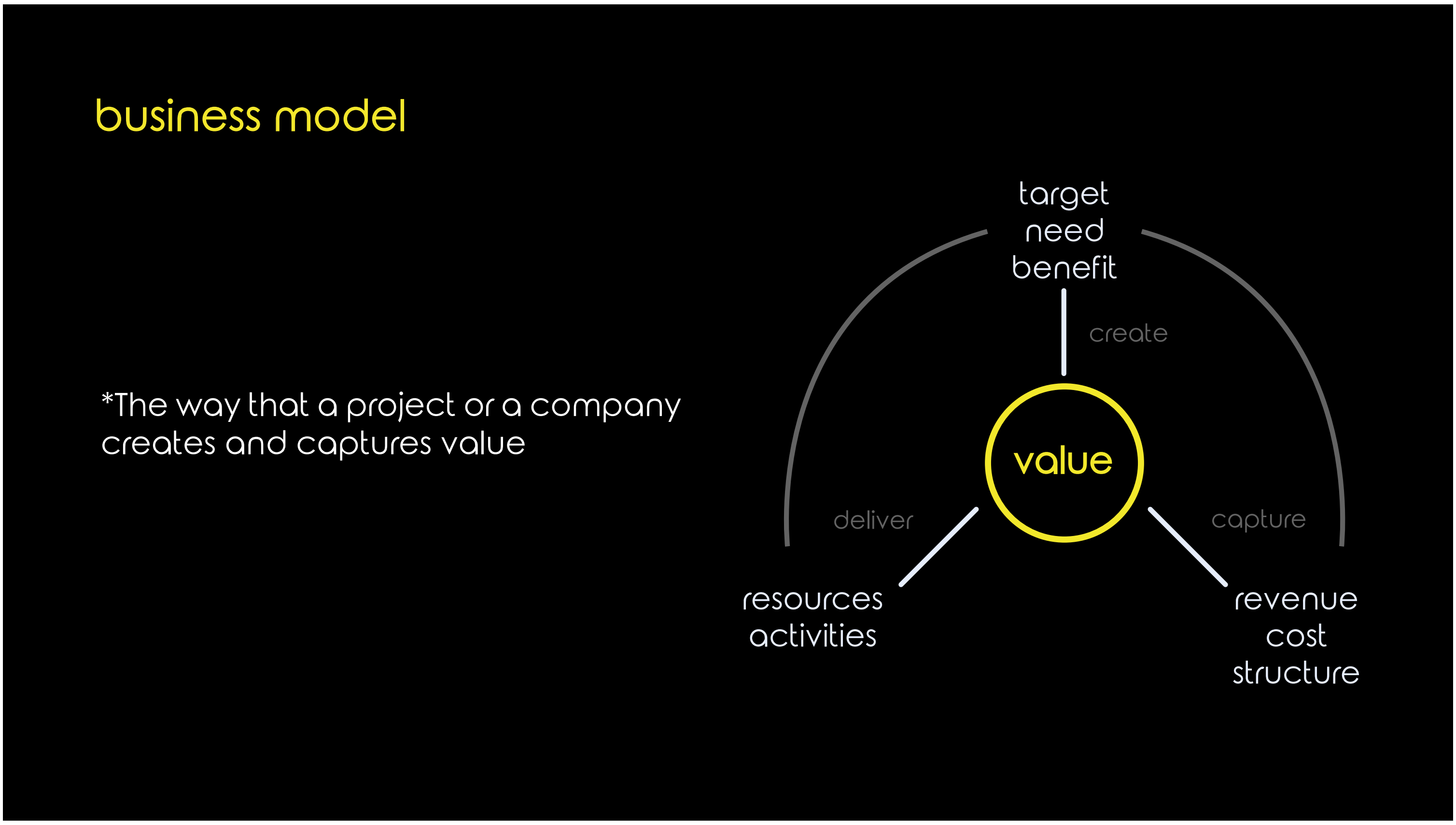
crowd design

service marketplaces

microtasks

crowdfunding

prediction markets



business model

*The way that a project or a company creates and captures value

target
need
benefit

create

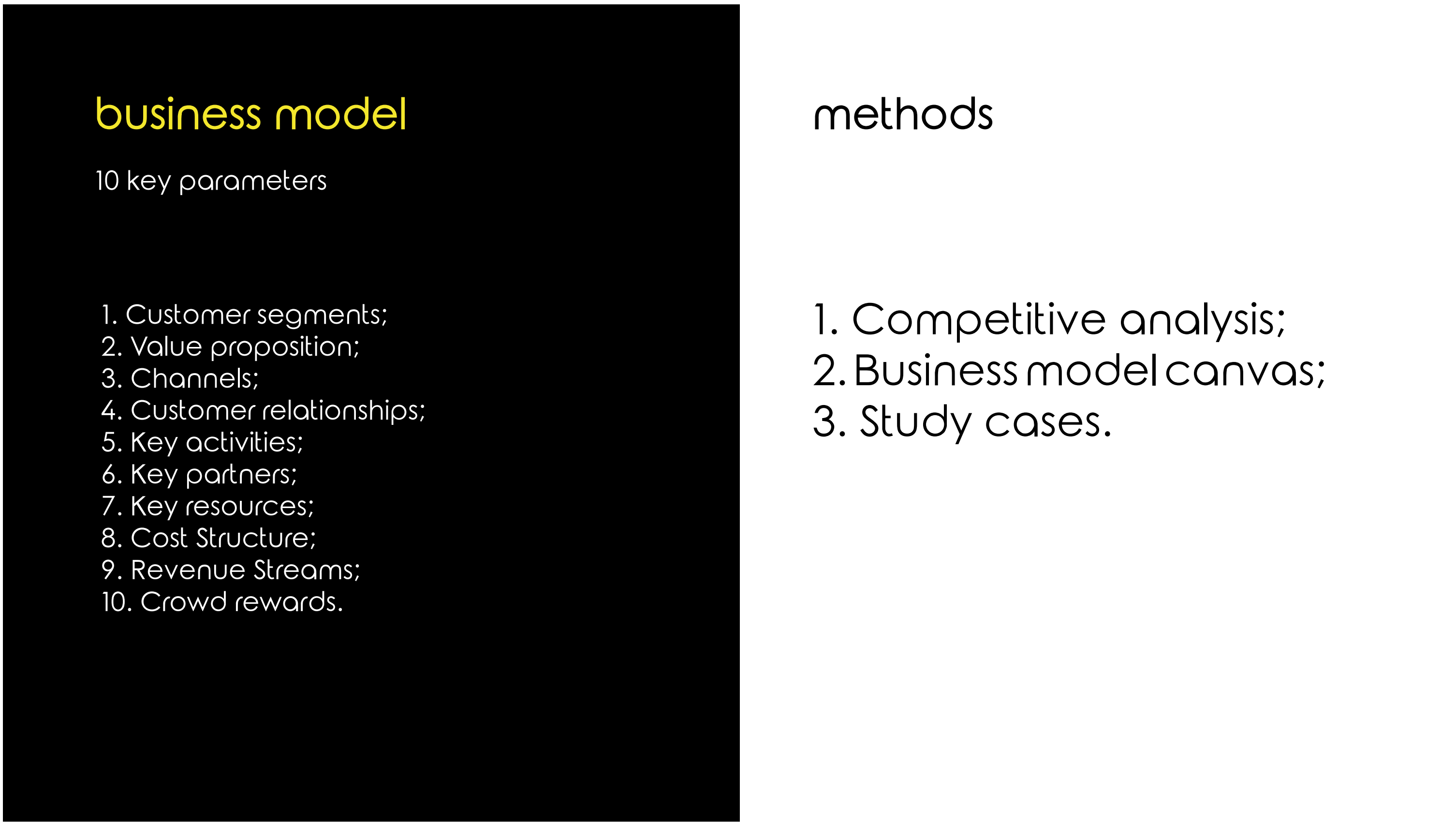
value

deliver

capture

resources
activities

revenue
cost
structure



business model

10 key parameters

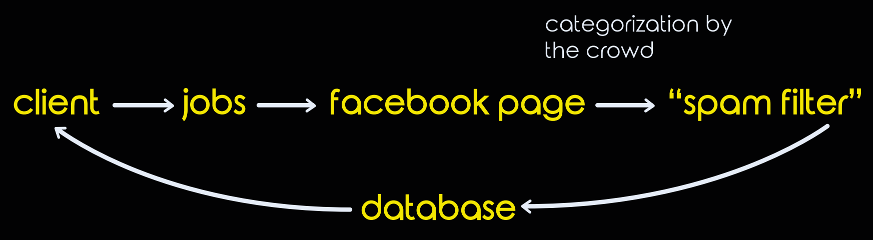
1. Customer segments;
2. Value proposition;
3. Channels;
4. Customer relationships;
5. Key activities;
6. Key partners;
7. Key resources;
8. Cost Structure;
9. Revenue Streams;
10. Crowd rewards.

methods

1. Competitive analysis;
2. Business model canvas;
3. Study cases.

business model

categorization tasks

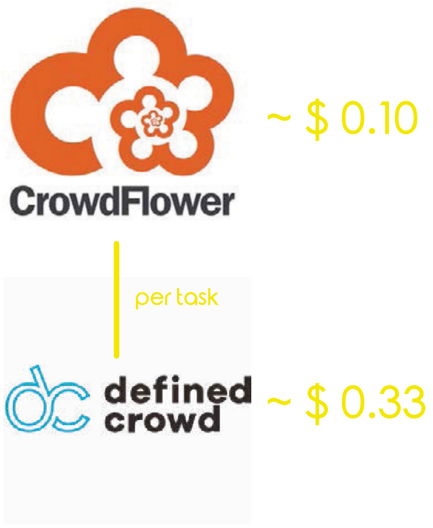

categorization by the crowd

client → jobs → facebook page → “spam filter”

database

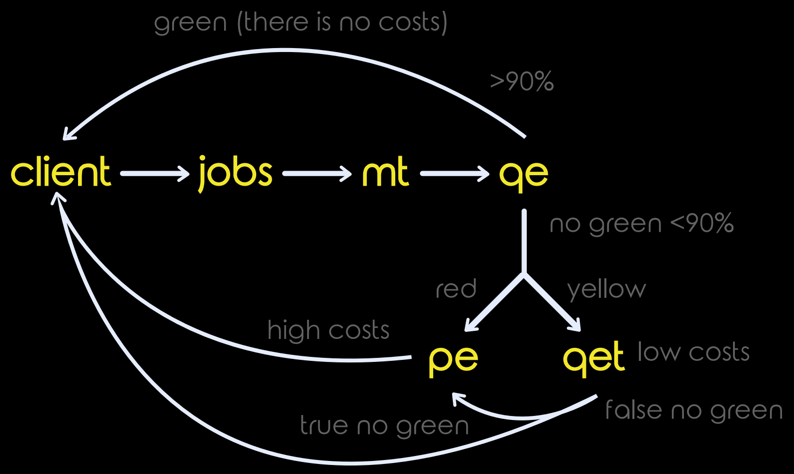
categorization tasks

price per task



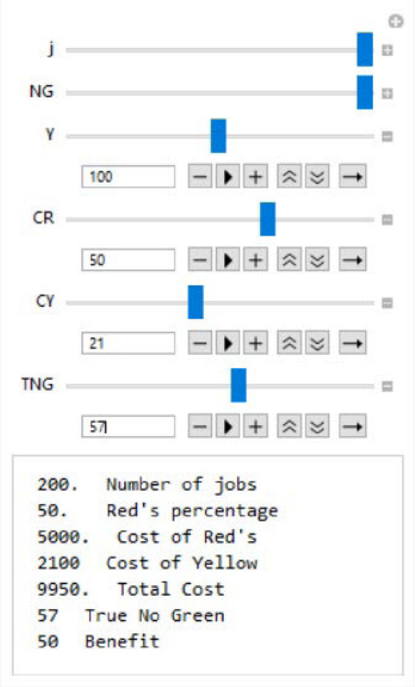
business model

quality estimation tasks



quality estimation tasks

simulation



Y 100

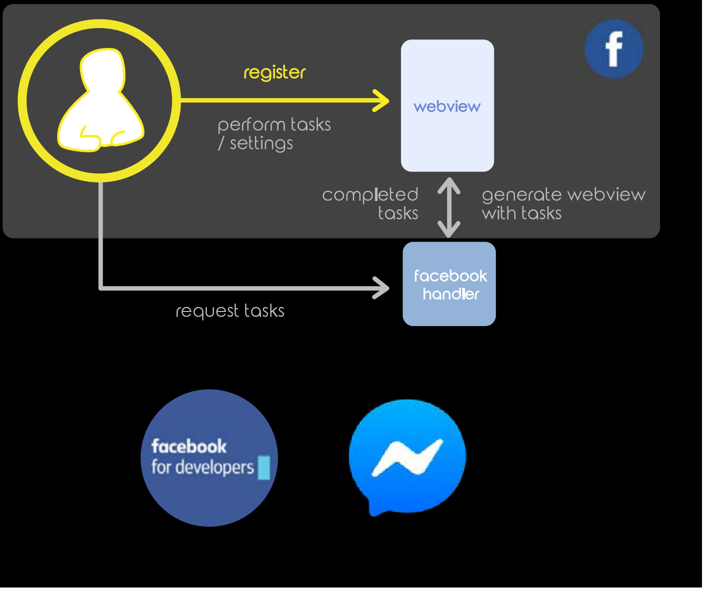
CR 50

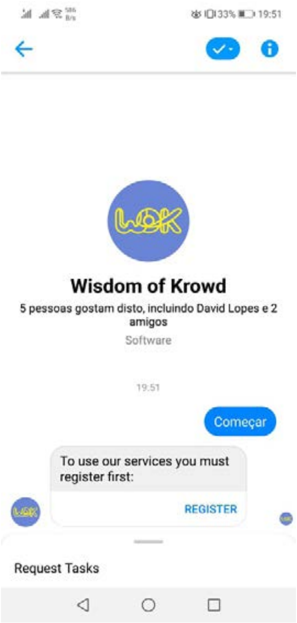
CY 21

TNG 57

```
200.   Number of jobs
50.    Red's percentage
5000.   Cost of Red's
2100   Cost of Yellow
9950.   Total Cost
57   True No Green
50   Benefit
```

user side architecture




Wisdom of Krowd

5 pessoas gostam disto, incluindo David Lopes e 2 amigos

Software

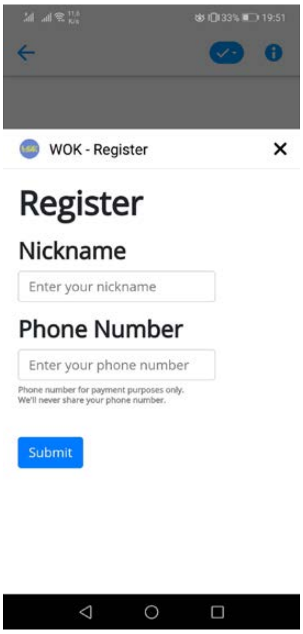
19:51

Começar

To use our services you must register first:

REGISTER

Request Tasks


WOK - Register

Register

Nickname

Enter your nickname

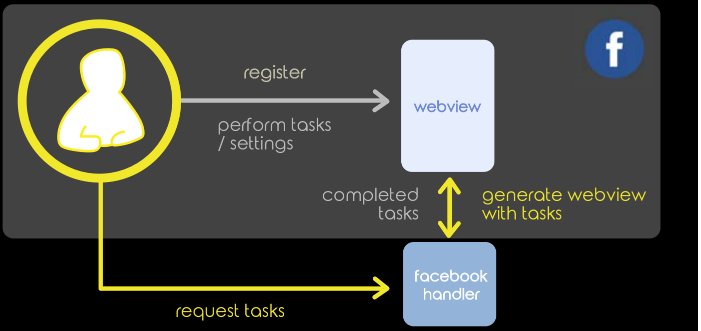
Phone Number

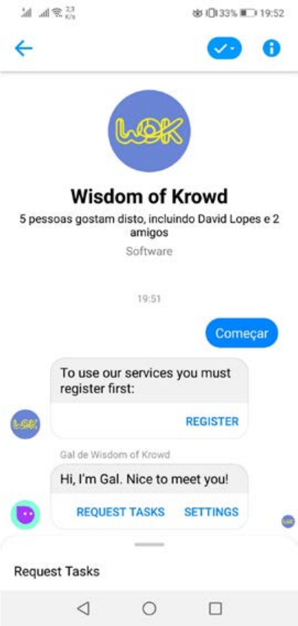
Enter your phone number

Phone number for payment purposes only.
We'll never share your phone number.

Submit

user side architecture




Wisdom of Krowd

5 pessoas gostam disto, incluindo David Lopes e 2 amigos

Software

19:51

Começar

To use our services you must register first:

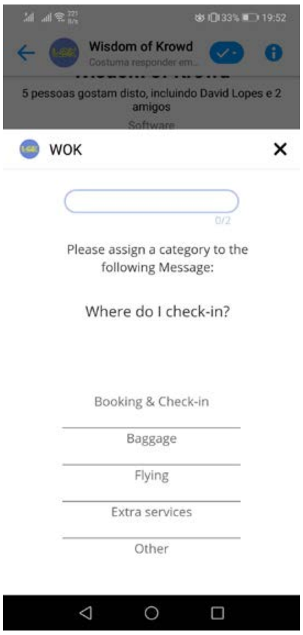
REGISTER

Gal de Wisdom of Krowd

Hi, I'm Gal. Nice to meet you!

REQUEST TASKS SETTINGS

Request Tasks


Wisdom of Krowd

WOK

0/2

Please assign a category to the following Message:

Where do I check-in?

Booking & Check-in

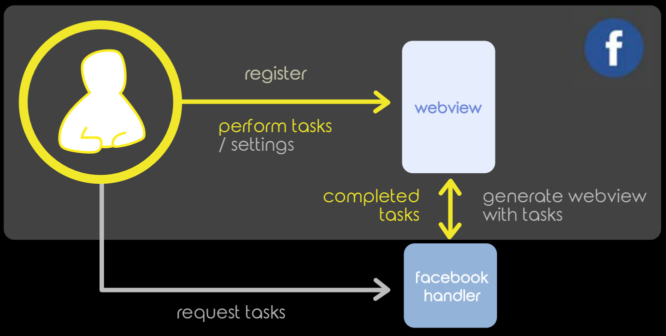
Baggage

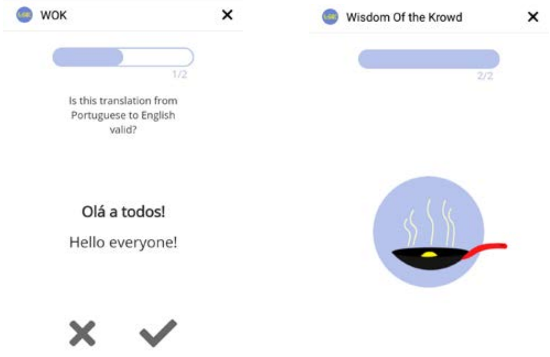
Flying

Extra services

Other

user side architecture

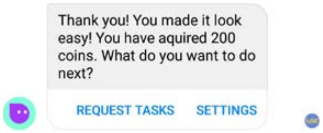




Is this translation from Portuguese to English valid?

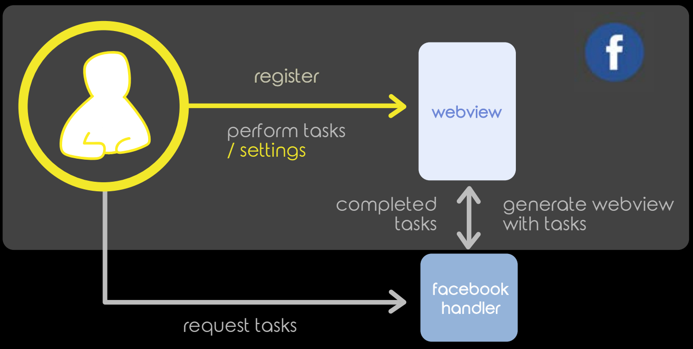
Olá a todos!

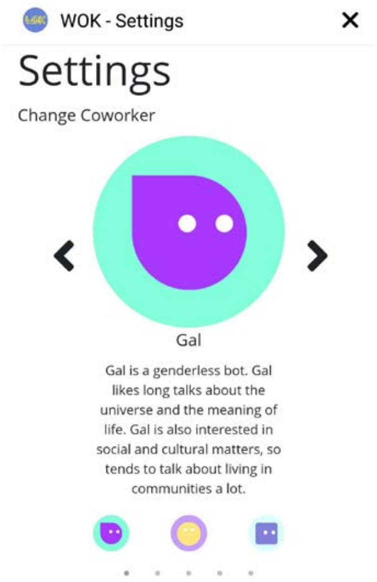
Hello everyone!



Thank you! You made it look easy! You have aquired 200 coins. What do you want to do next?

user side architecture



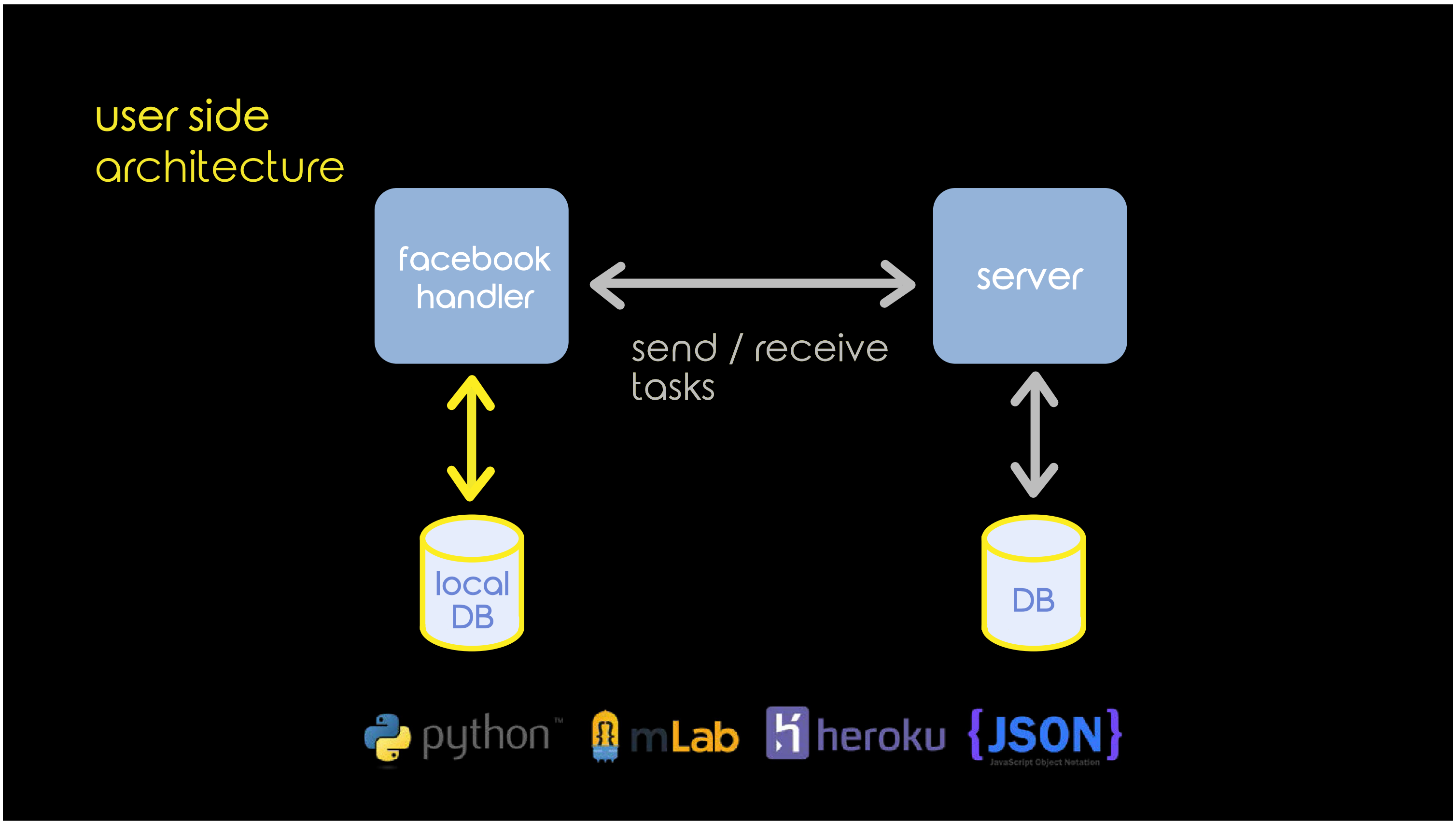


Settings

Change Coworker

Gal

Gal is a genderless bot. Gal likes long talks about the universe and the meaning of life. Gal is also interested in social and cultural matters, so tends to talk about living in communities a lot.



user side architecture

facebook handler

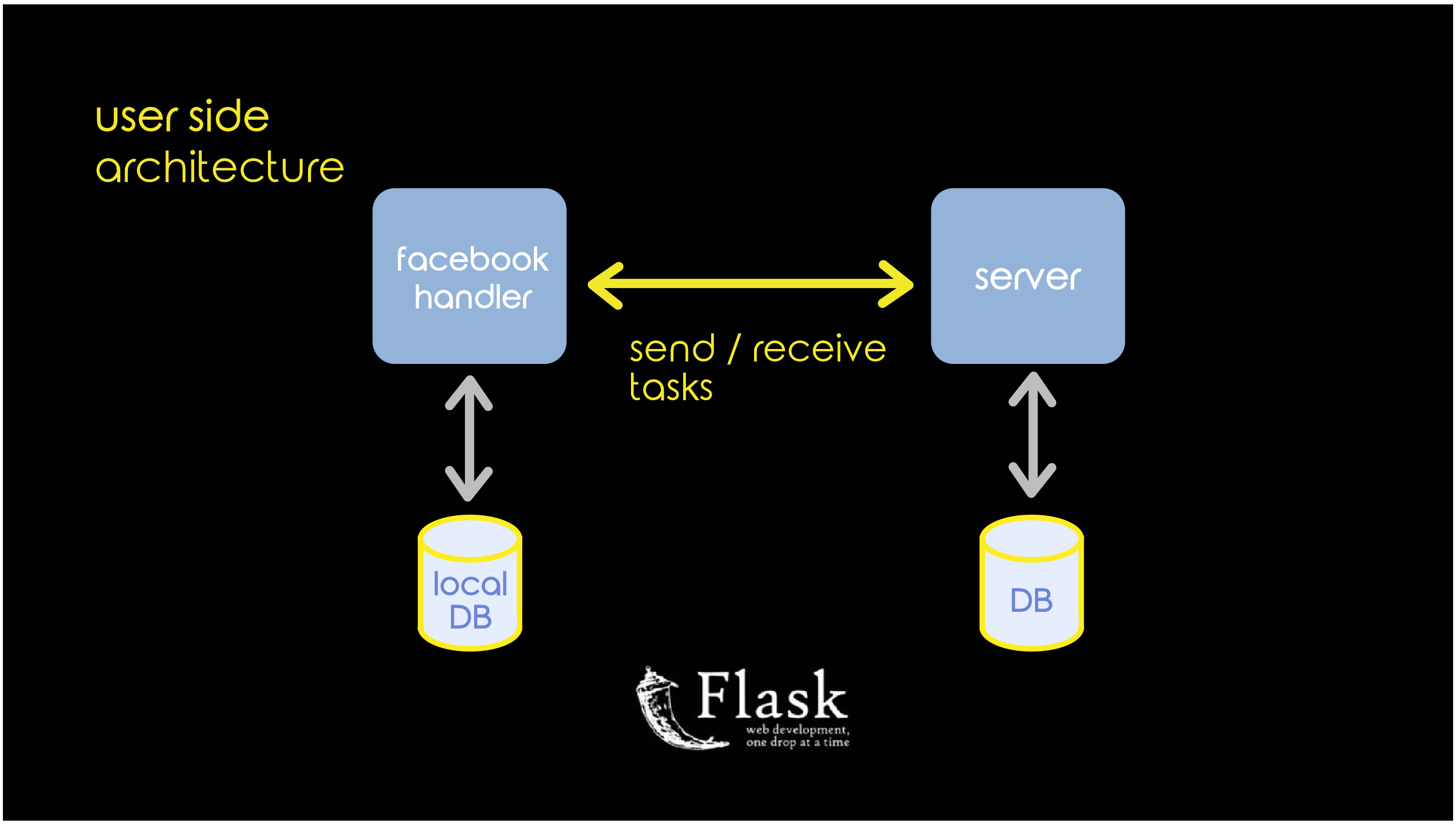
server

send / receive tasks

local DB

DB

python mLab heroku {JSON}



user side architecture

facebook handler

server

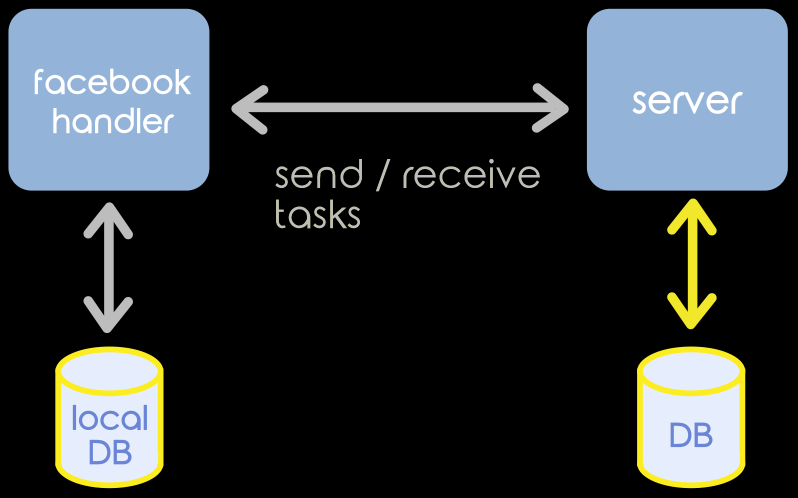
send / receive tasks

local DB

DB

Flask web development, one drop at a time

user side architecture



facebook handler

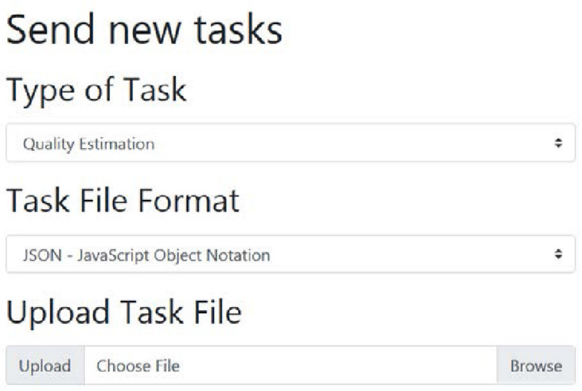
send / receive tasks

server

local DB

DB

client side architecture



Send new tasks

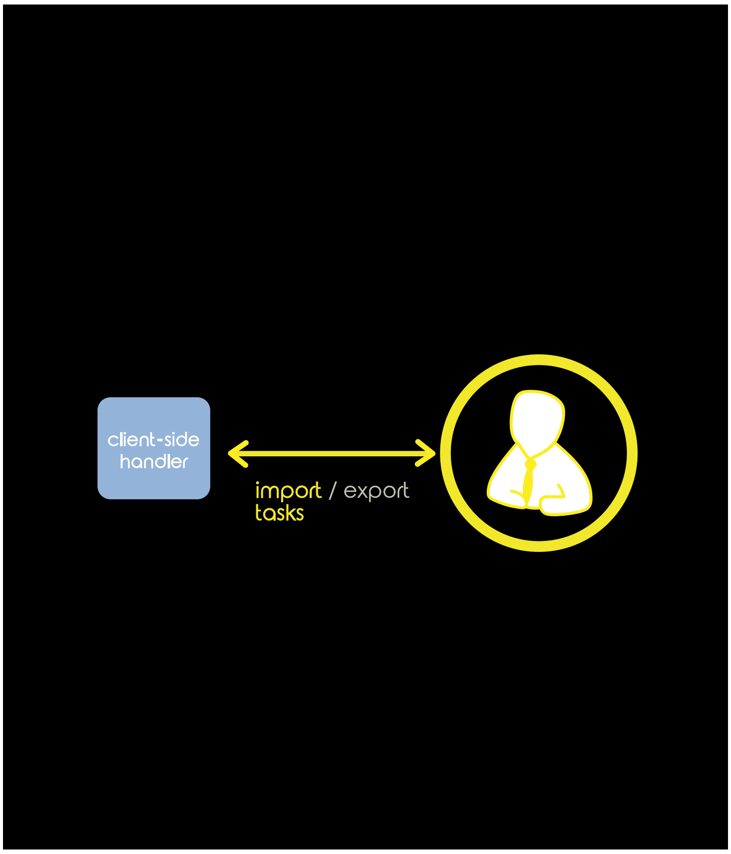
Type of Task

Quality Estimation

Task File Format

JSON - JavaScript Object Notation

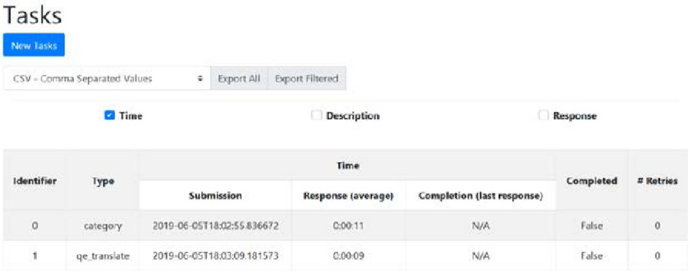
Upload Task File



client-side handler

import / export tasks

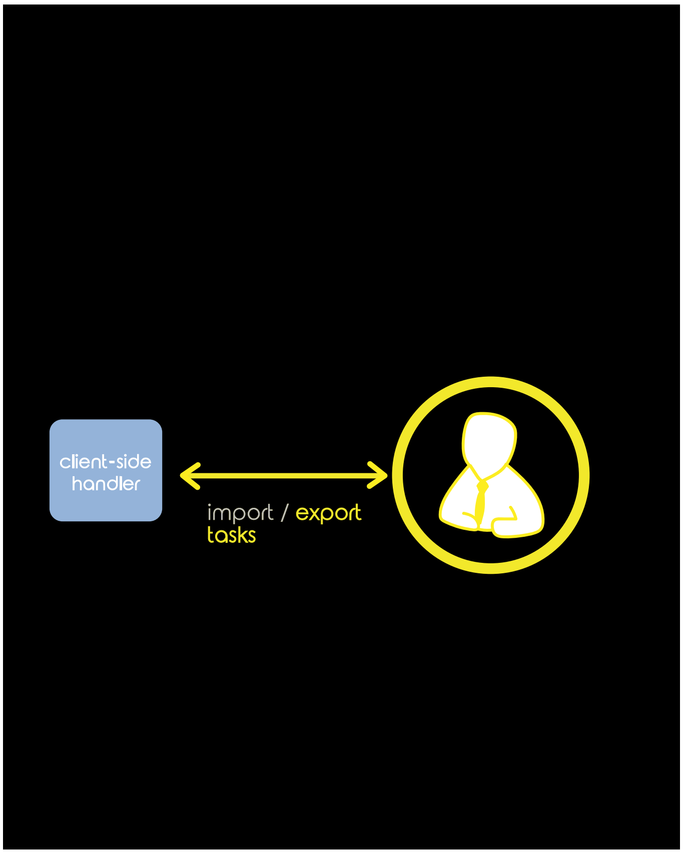
client side architecture



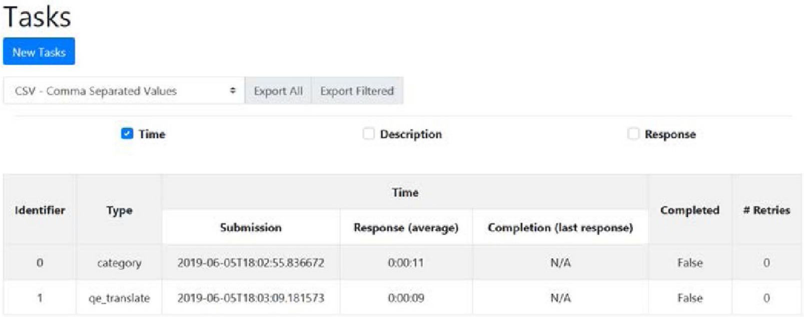
Tasks

☑ Time □ Description □ Response

Identifier	Type	Time			Completed	# Retries
		Submission	Response (average)	Completion (last response)		
0	category	2019-06-05T18:02:55.836672	0:00:11	N/A	False	0
1	qe_translate	2019-06-05T18:03:09.181573	0:00:09	N/A	False	0



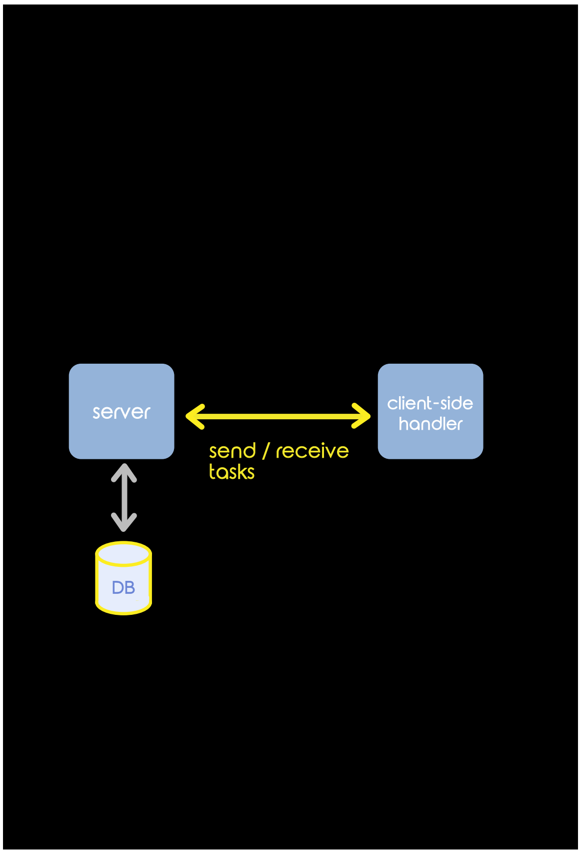
client side architecture



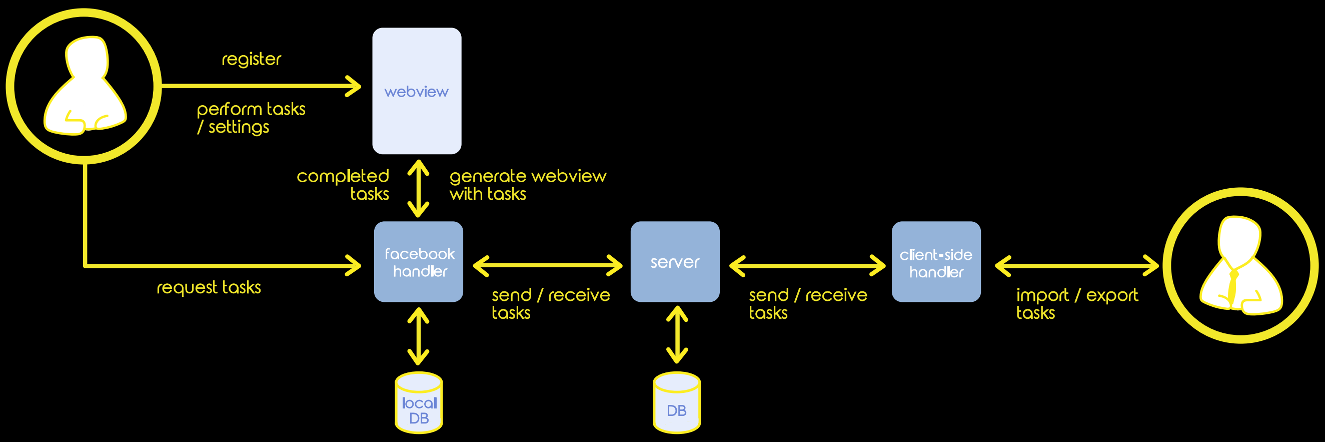
Tasks

☑ Time □ Description □ Response

Identifier	Type	Time			Completed	# Retries
		Submission	Response (average)	Completion (last response)		
0	category	2019-06-05T18:02:55.836672	0:00:11	N/A	False	0
1	qe_translate	2019-06-05T18:03:09.181573	0:00:09	N/A	False	0

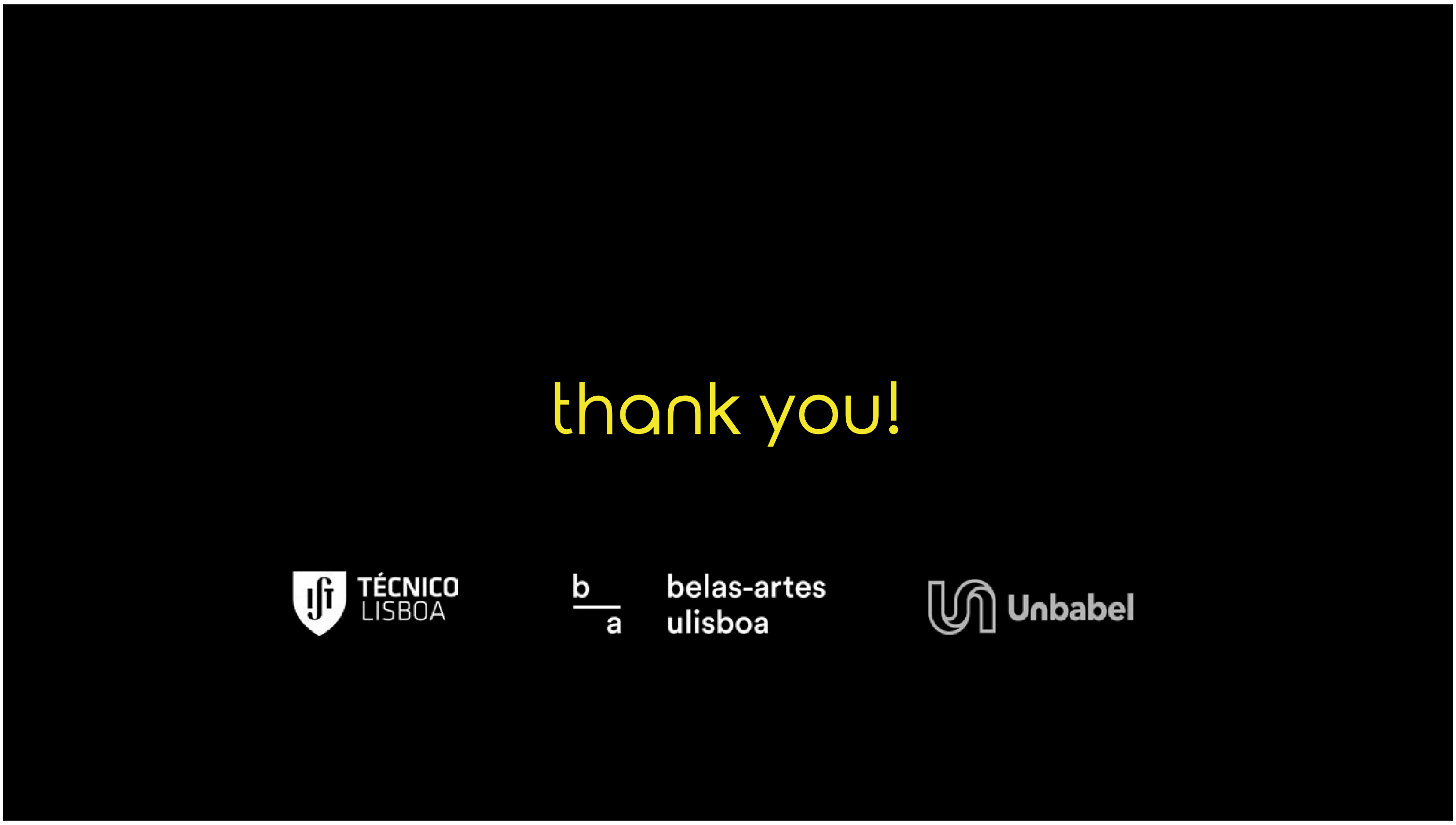


architecture



future work

1. Payment
2. Notifications
3. Client side authentication



thank you!

TÉCNICO LISBOA

belas-artes ulisboa

Unbabel