LISBON SCHOOL OF ECONOMICS & MANAGEMENT
UNIVERSIDADE DE LISBOA

MESTRADO

GESTÃO DE SISTEMAS DE INFORMAÇÃO

TRABALHO FINAL DE MESTRADO

TRABALHO DE PROJETO

WEREAD - DESENVOLVIMENTO DE UMA APLICAÇÃO EM LOW-CODE

DIOGO DIAS MENDES

OUTUBRO - 2019

LISBON SCHOOL OF ECONOMICS & MANAGEMENT
UNIVERSIDADE DE LISBOA

MESTRADO EM
GESTÃO DE SISTEMAS DE INFORMAÇÃO

TRABALHO FINAL DE MESTRADO

TRABALHO DE PROJETO

WEREAD - DESENVOLVIMENTO DE UMA APLICAÇÃO EM LOW-CODE

DIOGO DIAS MENDES

ORIENTAÇÃO:

PROFESSORA DOUTORA WINNIE PICOTO

OUTUBRO - 2019

Abstract

The mobile market is constantly growing and evolving, so the companies must develop new software that is unique and disruptive, pilot solutions.

The purpose of this work is to present and explain a pilot solution to perform water and electricity readings with a high degree of automation that I developed on Deloitte Portugal using OutSystems, a low-code platform.

To introduce the topics and models that I used to do this project it is presented a literature review on those same topics. After the literature review, the solution, and every phase of the project, is presented and described.

I end this work with a conclusion, where I talk about the difficulties of the project, the final considerations on working with a low-code platform and my satisfaction with the project itself.

The project consists of the development and implementation of two applications, a web portal (BackOffice) and a mobile application, called WeRead. The mobile application allows the user to read water and electricity meters with an Optical Character Reader. It offers a simple UI/UX so that any user with any degree of technological literacy can have a good experience while using the application. The web platform is available for management and administration only, where they can manage every information on the database.

Keywords: low-code platform, outsystems, mobile business

LISBON SCHOOL OF ECONOMICS & MANAGEMENT
UNIVERSIDADE DE LISBOA

Acknowledgment

I would like to thank teacher Winnie Picoto for being my mentor and for being always ready to help me.

I would also like to thank to ISEG, and every teacher of the GSI masters, for all of my knowledge that they passed onto me and the opportunity that they gave me.

I have yet to thank Deloitte for the opportunity to work with them and join CoE OutSystems (OutSystems Center of Excellence). Also want to thank all of my colleagues from the CoE for being always ready to help and to make me grow both personally and professionally.

Finally, but not least, I would like to thank to my girlfriend, all of my family and friends for always being there for me and available to help me at any time.

LISBON SCHOOL OF ECONOMICS & MANAGEMENT
UNIVERSIDADE DE LISBOA

Index

1. Introduction

Nowadays companies are interested fast software delivery that can be custom-made while meeting all of the requirements with ease. That is why low-code platform are quickly increasing their range of customers, their quantity of delivers while improving their own software (Richardson and Rymer, 2016).

Mobility and accessibility are important to most businesses and thus the mobile software market is steadily growing over the past few years. Every company wants to have their products always reachable and available to their customers, at work, at home at the public transports, everywhere. This is why low-code platforms are creating, developing and improving their software to accommodate web and mobile business, to offer to their clients a vast selection of technology so that they can deliver the product that best meet the requirements in less time.

In this report, it is going to be analyzed the growth of the mobile business as well as the low-code platforms, specifically OutSystems, and explain its advantages and disadvantages to the companies and users. Also it is going to presented a pilot solution to perform water and electricity readings with a high degree of automation, that I developed as a part of a project in a six months internship at Deloitte Portugal. It consists of two applications: one mobile for the regular users and one web focused for the admin users that works as a management/administration platform.

2. Literature Review

2.1 Mobile Business

A mobile phone used to be used mainly as a communication device, nowadays they are one of the most important work tools. It makes the business always reachable through the various applications, it also allows the companies and the users to buy or sell their own business as well as develop it (Sugai et al, 2010).

There are three major concepts that distinguish mobile business from traditional business: intimacy, time sensitivity and ubiquity (Paavilanein, 2001). Ubiquity because we have the ability of being everywhere and anywhere at the same time; Time sensitivity because we take so little to do things such as use our credit card or send an email or appoint a meeting; Intimacy because our phone *"always knows our needs",* if we let him to, he can always give us the best restaurant in the area or the directions to get to certain place (Paavilainen, 2001).

Mobile business can be defined as *"The exchange of goods, services and information using mobile technology"* (Paavilainen, 2001, p.1). Despite nowadays this market focuses on customer services it also values *"business-to-business and employee-to-employee"* emphaticizing the synergy between the costumer and the company's platform with the internet as the primary communication channel creating the flux between the mobile applications and the internet world.

Regarding the mobile internet has three main elements being *"communication, commerce and value added services"* (Paavilainen 2001, p.2). Those can be more specific if we add *"vertical target groups"* such as corporate and consumer.

One of the biggest *"assets"* of mobile business is the capacity to create value which means that it *"subsumes the concept of profit and that profit oriented viability is a necessity"*. It can be distinguished in three types of value: *"utility value, exchange value and essential value".* One good example on how mobile technologies can create value almost instantly is that you can *"enable a sales force"* simply by providing them with a mobile device where they can make business or create business, plus it adds *"social, ecological and otherwise essential revenue".* It is a certainty that mobile technologies offer an instant economic as well as social advantage however, those advantages can only be maintained when the technology is well thought and *"carefully incorporated with a long-term strategic view in mind"* (Unhelkar, 2009).

Mobile business offers the mobility aspect to a business and it extends to a whole lot of areas of expertise. In addition, it empowers the business quest for long-term profitability; it also plays a constant reminder in a business-to-costumer relationship offering a wide variety of marketing channels (Paavilanein, 2001).

"Does the mobile's phone limitations (a small screen, limited memory, etc) make it a different and more appealing marketing and business platform than a PC?" (Sugai, et al, 2010, p.72). The fact that it gives the possibility of permanent and constant communication combined with *"anytime, anywhere access"* automatically makes it completely different from a PC. Advanced mobile services help to create and maintain additional customer loyalty. As Frederick Reichheld stated in his *"Harvard Business Review"* articles, there is a direct link between customer loyalty and long-term profitability which means that *"attracting a new customer costs more than retaining a current one".*

2.2 Low Code Platforms

Low-code Development platforms are visual-based integrated development environments (IDE) where it is possible to develop applications through graphical user interfaces, drag & drop, instead of traditional programming. It reduces the delivery time and reduce both the time and the cost of the setup, training. It does not require any experience in programming but requires logical and problem solving thinking skills, and developing and deployment experience. It expands the barriers of programming enabling collaboration since multiple people can work on the same application at the same time.

The term *"low-code development"* was coined by Clay Richardson on a Forrester's article in 2014 (Richardson,2014).

Low-code development platforms have several advantages, being the most important the cost reduction, since you can develop more in less time, which means faster delivery speed (Appian, n.d.). Another big advantage is that the apps can be easily changed and adapted so that it can meet the new requirements with ease. It is the combination of these advantages that make low-code platforms highly productive, while improving the efficiency of a company and its releases and it gives a better customer experience (OutSystems, n.d.).

There are essentially two disadvantages, that are globally agreed on, customization and vendor lock-in. Every platform has its types of customization so before a platform is chosen, it is important to know the customization limits so that the correct choice is made so that the chosen platform can accommodate the requirements. Vendor lock-in is when you choose a platform and develop all your applications there. But you cannot access them outside the platform, so you are locked to the platform (OutSystems, 2018.). As Brendar Bank (CTO for MessageBird) once said

"The best low-code solutions are the ones that are flexible and framework/language/syntax-agnostic. Make sure to look for platforms that are webhook-based, where the HTTP call back requests are completely customizable by the customer." (Harris, 2019), it is the costumer that needs to search and analyze the market and figure out what is the best platform that suits his requirements and does not lock him in.

Despite all those *promises "impressive results in speeding application delivery, and high vendor growth rates.",* low code platforms have indeed risks, those being *"small vendors selling outside of tech management"* and customers with little knowledge about how low-code platforms will broaden, improve and accelerate their business (Richardson and Rymer, 2016).

2.3 Citizen Developers

These platforms are creating a new type of developer the "citizen developer". A citizen developer is a person that has a developer mentality without knowing any specific programming language. The citizen developer thinks like a pro developer but uses drag-and-drop tools.

To evaluate the definition and the impact of this new concept the analysis it is based on the following works: Everhad (2019), Warren (2019) and Gartner (n.d).

This new concept of allowing staff to become citizen developer, have some vast benefits such as cost reduction because the company doesn't have the need to recruit and pay more for expensive developers with the knowledge and experience on building enterprise applications. It allows the organization to be more agile and flexible regarding its approach to IT and developing application, it also speeds up the developing time as

Forrester claims that low-code platform can increase software development up to ten times faster than traditional methods. With that said, and since everyone can help each other and the company on its digital transformation productivity will skyrocket.

However, as stated earlier low-code platforms needs logical/problem solving thinking or a bit of programming experience, which many workers doesn't have and need to develop those certain skills, delaying the start of projects. It will also requires that workers are always well informed about the changes and upgrades of the platform and have consistent training and certifications.

2.4 Accessibility and Usability

Despite the continuous rise of the terms *usability* and *accessibility,* they still encounter many issues. All users have, at some point, encountered obstacles while interacting with a web app or a mobile app, not knowing the purpose or how it's operated. Also, there are two types of users, the *real user* and the *ideal user.* The *real user* is the one that only knows what was said, transmitted or taught by someone, which means he only knows what he perceives. The *ideal user* who just exists for the designer, knows the answer to all questions, has access in the ideal conditions to the best technology and information to perform the functions with the greatest success.

A reflection on usability and accessibility in web/mobile design becomes in a context in which the designer is unaware of all the current devices and future user preferences and needs. It should grant the user the best experience while navigating through the app/site without questioning himself, *"As designer professionals, we should be designing our content so it is globally accessible and meets the needs of as many*

people as possible and practical given our specific circumstances, regardless of their abilities or the type of device they choose to access the Web." (Clarke, 2006, p.14).

2.4.1 Usability

Usability studies and analyses the relationship between the platforms and the tools with their users. To consider a tool effective it should allow users to perform the tasks desired in the best possible way. There are five principal components of usability: 1) Learning Capacity; 2) Efficiency; 3) Ability to be memorized; 4) Security; 5) Satisfaction (Nielsen, 2003).

Usability can be defined as *"something that works well. That a person of average (or even below average) ability and experience can use the thing-whether it's a website, a fighter jet, or a revolving door-for its intended purpose without getting hopelessly frustrated."* (Krug, 2014, p.23).

2.4.2 Accessibility

Accessibility is considered a subclass of usability, seeks that any person, regardless of any possible sensory or motor limitations, can interact with the system.

The main concern in designing accessible products is to ensure their universality, making the product more flexible and complete. Accessibility aims, therefore, make the interfaces perceptible and understandable by people in various circumstances, environments, and conditions, *"Accessibility makes user interfaces perceivable, operable, and understandable by people with a wide range of abilities, and people in a*

wide range of circumstances, environments, and conditions." (Henry, 2007).

Accessibility can be defined as *"ensuring that a given page on the web is able to be accessed"* (Holzschlag, et al. 2006).

2.4.3 Usability and Accessibility - Conclusion

Both Usability and Accessibility are very important user wise – it evaluates the efficiency and the ease of the system – they are also important developer wise – it evaluates the success or the failure of the system. *"Somehow when a device as simple as a door has to come with an instruction manual – even a one-word manual- then it is a failure, poorly designed"* (Norman, 1998, p.87), as said previously we need to connect both concepts and use their good practices to create an interface and a system that works to every user to the full capacity.

2.5 Interface Design Heuristics

There are ten principles for the design of interfaces: 1) Visibility of the system status: the system must always provide users feedback of their actions and about what is happening; 2) Similarities between the system and the real world: the system must use familiar words, phrases, and concepts so that the user can easily understand it; 3) Control and freedom: the system should be equipped with *"Undo", "Redo", "Rewind" and "Forward"* buttons, to let the user roam around the interface without any *"chains"* on; 4) Consistency and standardization: all of the words, actions and buttons must be consistent throughout the system without leaving on the user any confusion; 5) Error Prevention: global help and examples should be available alongside the user navigation experience; 6) Flexible and efficient use: accelerators, such as shortcuts and macros,

should exist they should not be visible to not experienced users; 7) Recognize instead of remembering: the effort required to user memory should be minimized; 8) Aesthetic and minimalist design: the pages should not contain irrelevant information; 9) Help users recognize and recover from errors: the errors feedback messages should be displayed in clear colors and text that every user can comprehend and fix; 10) Help and documentation: the documentation should be easy to access and to search, focused on users actions and showing examples (Nielsen, 1994).

2.6 Development Methodologies

The concept of development methodology does not have a one hundred percent defined concept that all authors and scholars agree with. Therefore, there are going to be presented two different approaches by some of the most famous authors on this matter.

Different authors have different approaches to define development methodologies. Development methodologies can be defined as *an "application model of software engineering practices, which as the specific objective to provide the necessary means for the development of software systems"* (Ramsin and Paige, 2008). There are more authors with different approaches for the same concept, defining them as *"stages collection, procedures, rules, techniques, tools, documentation, management and training used to develop a software system"* (Avison and Fitzgerald, 2003).

For a software developing project be successful it should follow the principles of a development methodology(ies). It should be aligned with the project's needs in order to improve the efficiency of the project and consequently customer satisfaction, *"Using an appropriate software*

lifecycle model can improve efficiency and effectiveness of software development." (Guntamukkala, et al, 2006)

In the next chapter there is going to be made the analysis and the comparison between traditional methodologies (waterfall model is going to be used as example) and Agile ones (SCRUM is going to be used as example).

2.6.1 Traditional Development Methodology

Traditional methodologies are described as the ones that uses *"extensive planning, coded processes and rigorous reuse to make the development an efficient and predictable activity"* (Guntamukkala, et al, 2006). Establishing rules and different priorities so that the team is always aligned in all of the development stages until the delivery.

2.6.2 Waterfall Model

It is called "*waterfall*" because the way that the model works it's similar to a waterfall, the development team can only move on to the next phase when the current phase is over which makes all the work must be done in a linear way (Avison and Fitzgerald, 2003).

Based on the previous information it is safe to conclude that the concept of a development methodology using phases for the duration of the development process has more than sixty years.

The waterfall model has seven phases according to several authors (Munassar and Govardhan (2010), Avison and Fitzgerald (2003), Sheffield and Lemétayer (2013), Davis (1988) and Royce (1970)). It is based on those works that each of the waterfall model phase is described. Being: **(i)** System requirements **(ii)** Software Requirements **(iii)** Preliminary Design **(iv)** Detailed Design **(v)** Code and Debug **(vi)** Tests **(vii)**

Maintenance. The **(i)** system requirements and **(ii)** software requirements phases are similar and it's where is created the list with all of the system requirements and the tools that are needed to develop it as well as the software requirements and the expectations regarding the functionality of the application. On the third phase, **(iii)** preliminary design, the model and structure of the application is defined to meet the requirements that were defined in the first two phases. **(iv)** Detailed design is where it is defined how to implement every single aspect that was structured and defined on the previous phase. At the code and debug **(v)** phase the development of the components detailed previously start and while some of the elements of the team code, the others are running debugs. Nevertheless, to ensure that all of the functionalities are properly working and well implemented there is a need to test **(vi)** every requirement. Finally but not least **(vii)** Maintenance, where the problems that are left and some new updates or fixes are solved and implemented, for some authors this phase, besides being part of the *"waterfall model",* exists throughout the entire project. After the project ends, it is made a global evaluation that involves reviewing whether all the tasks have been completed with success and the development has achieved the objectives that set by the team.

2.6.3 Advantages and Disadvantages

Being a widely known and used model, it has its advantages and disadvantages according to several authors (Kumar et al, (2013), Alsharamni and Bahattab, (2015)). The advantages and disadvantages are describred based on those works.

Regarding the advantages, the waterfall model is simple and easy to understand, implement and manage; since each phase has

specific deliverables and review process all requirements should be clear before going to a next phase; phases are processed and completed one at a time and it is only possible to go to the next phase after the previous one is finished. At this point is where the authors differ on their opinions for Narush Kumar, A.S. Zadgaonkar and Abhinav Shukla one of the big advantages is that the model works well for projects where requirements are well understood while for Adel Alsharamni and Abdullah Bahattab the model works well on mature products and provides structure to inexperienced teams. Some more positive things about this approach identified by all authors is that the stages are well defined, it's easier to understand milestones, it minimizes overhead planning and both processes and results are well documented, both helping the final review and knowledge management to adapt and implement in a next project.

The major disadvantage of developing software using the waterfall model is that it does not allow revision or reflection of what has been done. It is very difficult to go back and change something that was not well documented. Also, it does not allow requirements changes as per clients request, no working software is developed until the late stages of the life cycle, which means that it's not ideal for long and ongoing projects. Additionally sometimes, the client does not get a clear view of what is being developed and wants to change some functions, requirements or even the scope of the project, which could kill it. Lastly, this model is it not a *"preferred model for complex and object-oriented projects"* (Alshamrani and Bahattab, 2015).

2.6.4 Agile Development

The agile methods are an answer to traditional ways of developing and acknowledge the *"need for an alternative to documentation driven, heavyweight software development processes"* (Beck, 2001). In the traditional methods, the kick-off starts with the documentation of a set of requirements followed by architectural design, development, and inspection. In the 1990s some people started getting frustrated and tired of the initial steps so they *"joined forces"* and started developing alternative methods (Highsmith, 2002).

Most of the Agile practices are nothing new, they are an evolution of the traditional processes (Highsmith and Cockburn, 2001).

In 2001, seventeen software developers met at a resort in Utah and together they published the Manifesto for Agile Software Development (Beck, et al, 2001). The manifesto has twelve principles: 1) *"Our highest priority is to satisfy the customer through early and continuous delivery of valuable software"*, 2) *"Welcome changing requirements, even late I development. Agile processes harness change for the customer's competitive advantage.",* 3) *"Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.",* 4) *"Business people a nd developers must work together daily throughout the project.",* 5*) "Build projects around motivated individuals. Give them the environment and support they need and trust them to get the job done.",* 6) *"The most efficient and effective method of conveying information to and within a development team is a face-to-face conversation.",* 7) *"Working software if the primary measure of progress.",* 8*)"Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.",* 9) *"Continuous attention to technical excellence and good design enhances agility.",* 10) *"Simplicity—the art of maximizing the amount of work not done—is essential.",* 11) *"The best architectures,*

requirements, and designs emerge from self-organizing teams.", 12) *"At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly."* (Beck, et al, 2001).

The Agile method has four really important ideas attached to it regarding software development, teamworking, and customer expectation, *"We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value: individuals and interaction over process and tools; working software over comprehensive documentation; customer collaboration over contract negotiation; responding to change over following a plan. That is, while there is a value in the items on the right, we value the items on the left more"* (Beck, 2001).

With this method, software development became more dynamic, practical and responsive while teamwork and communication became more valued in order to manage the needs and expectations of the customer.

What does it mean *"to be Agile"?* So being agile means being able to *"deliver quickly, change quickly and change often"* (Highsmith, 2001). Developing in iterations allows the development team to adapt quickly to changing requirements. Working in close location, near each other, and focusing on communication means that teams can make a decision and act on them immediately. Reducing intermediate artifacts and documentation that do not add value to the final deliverable means that the team doesn't waste time on that a more resources can be added to the development of the final product and it can be completed sooner, *"a great deal of the agile movement is about what I would call programmer power" (*Glass, 2001*).* The practices that Agile methods use are not new, but the idea and the *"recognition of people as the primary drivers of*

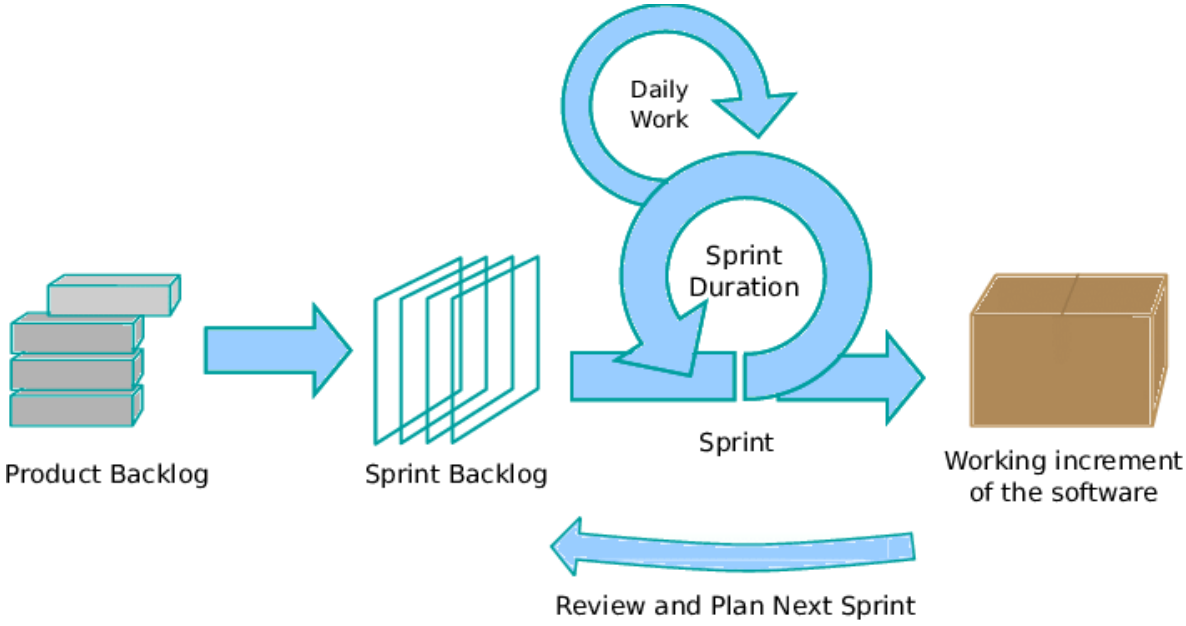
project success" is, plus the focus on effectiveness and maneuverability (Highsmith and Cockburn, 2001).

Although the agile development has many methods it will only be discussed one, SCRUM.

2.6.5 SCRUM

Scrum is one of the more used Agile methods. Scrum is described as a process that *"accepts that the development process is unpredictable"*, the term is lent from rugby (Schwaber, 1997).

Scrum hangs all of its practices on an iterative and incremental process (image 1).



(Image 1)

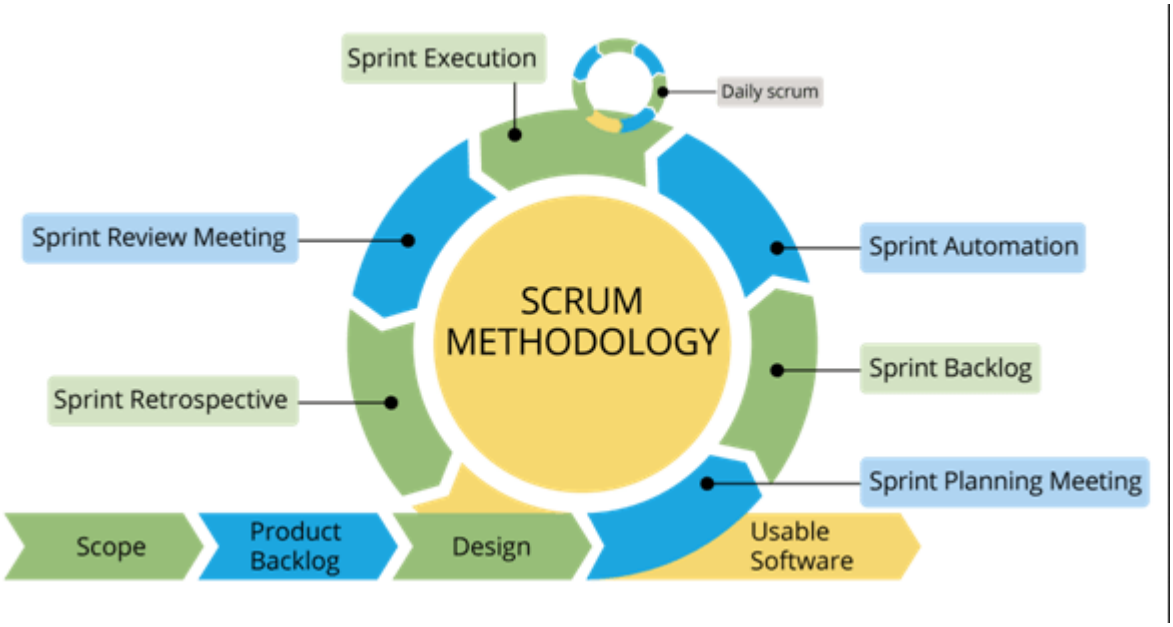
At the start of an iteration, the team reviews what is must do, it then identifies what functionalities are they able to finish until the end of the iteration keeping in mind all of the factors that can influence the process including their own skills and capabilities.

There are three scrum roles, the Product Owner, the Team and the ScrumMaster, and all of them have management responsibilities. The

Product Owner is responsible for representing the interests of everyone involved in the project. He creates the project initial list of requirements (product backlog) as well as return of investment (ROI) objectives and release plans. He also needs to make sure that the most valuable functionality is the first one being developed and built upon. The team is the developing end of the project and need to make sure that the functionalities meet the project's needs. It is self-managed, self-organized and cross-functional. Finally, the ScrumMaster is the responsible for the Scrum process, teaching, implementing and adapt the scrum method so that it fits the within the organization's culture and delivers the result that everyone wants (Schwaber, 2004).

The Scrum begins when the project starts, with a vision that will become clearer as the project advances. All work is done in Sprints (iterations). Each sprint is initiated with a Sprint Planning Meeting, where the Product Owner and the team plan what will they do on the next sprint, they select and sort the requirements from the Product Backlog by priority creating a development hierarchy. They agree on goals and deadlines. Sprint planning meetings cannot last longer than eight hours. The Daily Scrum is another important part of Scrum's workflow, every day all of the team's members get together and have a fifteen-minute meeting. Each member answers the same three questions: *"What have you done on this project since the last Daily Scrum meeting?", "What do you plan on doing on this project between now and the next Daily Scrum meeting?"* and *"What impediments stand in the way of you meeting your commitments to this Sprint and this project"*. *"The purpose of the meeting is to synchronize the work of all the team members daily and to schedule any meetings that the team needs to forward its progress"* (Schwaber, 2004). After a Sprint ends, a Sprint review meeting is held. It's an informal meeting where the team presents and discusses the result of the Sprint

to the Product Owner and to all the stakeholders that want to be present. Before the next Sprint starts the Scrum Master holds a Sprint retrospective meeting with the team. Together they reflect on the previous sprint and identify and agrees on actions that can improve the process (image 2).



(Image 2)

With the appearance of Scrum came the creation of new documentation, artifacts, that are used throughout the Scrum method. Being the main ones the Product Backlog and the Sprint Backlog. As said previously the Product Backlog has an ordered by priority list of requirements for the product. The common formats are user stories or use cases but it varies. And it is visible to everyone but it can only be changed with the approval of the Product Owner. The Sprint Backlog is the list of work that the team needs to do on the next Sprint. The list is created by the team with the items out of the Product Backlog and ordered, again, starting with the most important item (Schwaber, 2004).

Scrum works because *"moves control from a central scheduling and dispatching authority to the individual teams doing the work. The more*

complex the project, the more necessary it becomes to delegate decision making to independent agents who are close to the work" and also because *"shortens the feedback loop between the customer and the developer*" (Schwaber, 2004).

2.7 OutSystems

OutSystems is a Portuguese company based in Atlanta, USA, founded in 2001. Operates in 52 countries, 22 industries has more than 245 global partners and more than 210 000 members in the community. Provides a business platform for fast application delivery, through a low code platform. Its vision is "to fuel the future of digital innovation. A new world unbounded by traditional software and systems, where the creative potential in every organization is unleashed. A future with no limits" (Paulo Rosado, n.d).

This platform has been recognized by several entities. In fact, it was named leader in the 2017 Gartner's Magic Quadrant for mobile application development platforms and also in the 2018 Magic Quadrant for Enterprise High-Productivity Application Platform as a Service. It won a wide variety of awards including three times "Best Mobile Application Development Platform" by CODiE and "Top Rated Low-Code Platform" by TrustRadius.

Outsystems is a visual low-code platform that comes in two parts a server and a desktop application (Service Studio) for developers. The Service Studio is where the developers design and develop their mobile (iOS and Android) and web applications by drag & drop (visual programming). It allows the developer to model databases, create

workflows and rules with ease and then deploy them to production server within minutes.

Because of it being a low-code platform, it does not mean that we cannot control the source code. We can control HTML, CSS and JavaScript inside the OutSystems platform (Service Studio). HTML can be controlled through the "expression" widget, where we can inject our own HTML code into our pages. CSS can be manipulated in every aspect (create classes, modify existing CSS or create our own). We can select if we want to manipulate CSS of the current page or any other pages or the entire app or module.

JavaScript (JS) has different behaviors for mobile and web environments. On mobile apps, we can import and/or create our own JS scripts and then use them on our app. We can also insert scripts directly into our workflow. On Web apps, we can add JS scripts through the JS handlers inside every web block and screen, which will run directly on that page.

It is also possible to use JAVA, C# (C Sharp) and Microsoft .NET with an OutSystems platform (Integration Studio) where we can create our own functionalities and then use them on our OutSystems' applications. For that, we need to create an extension on integration studio. Integration Studio environment provides several features that accelerate and automate the development of integration components and add-ons. It also supports Microsoft SQL Server and Oracle Database.

OutSystems allows the user to reuse code and create modular apps be making elements of other modules available and ready to be used. With this practice, elements can be easily managed centrally in the module which provides them and, in the modules that use them as a reference, OutSystems can prompt you for changes and update them

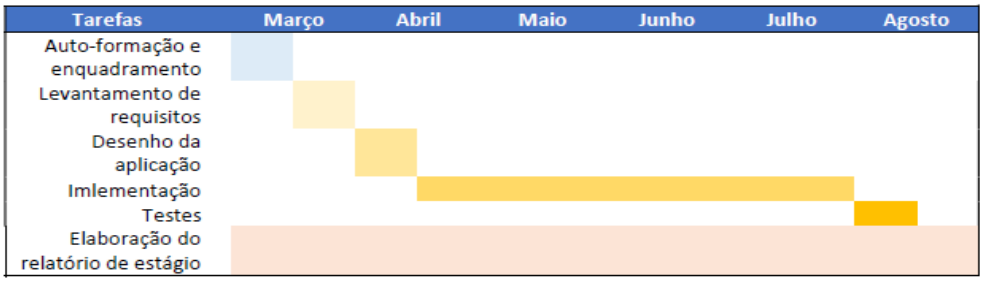
automatically at your command. Integrates natively with several of the major database systems: SQL Server, SQL Azure, Oracle, MySQL, and DB2 iSeries. It also has a project repository called Forge where users share and reuse open source software projects like applications, components, connectors, widgets, themes and sample code.

3. Project description

3.1 Introduction and Planning

This project was developed in Deloitte Consultores S.A (Deloitte Portugal) as I joined the OutSystems CoE (Center of Excellence, the only one in Portugal), a team of OutSystems specialists. The internship started on March 1 2018 and ended on August 31 2018. The objective was to develop and implement a pilot solution to take water, electricity and gas readings with a high degree of automation, named as WeRead.

Within the scope of the internship, six tasks were defined (image 3): 1) Self-training and contextualization, where I did two web courses certified by OutSystems and received contextualization about the platform; 2) Requirements specification, where I read the documentation regarding the project and defined all of the requirements; 3) Application Design, where I defined what were the elements and how to use them on the application, all of the graphic and technical design was made on this phase; 4) Development and implementation, where I develop and implemented all of the requirements with the designs created on the application design phase; 5) Testing, where I tested all of the functionalities and behavior of the application; 6) Elaboration of the internship report, the report was written after each phase was concluded. Image 3 illustrates the project planning.



(Image 3)

3.2 The WeRead Project

The project *WeRead* is a multiplatform application and it has two components: a mobile application and a web platform. Both were developed using *OutSystems* within the scope of this project. The mobile application allows the user to read the water and electricity meters through an OCR (Optical Character Reader) and it also gives the possibility to report any damage that the user sees in their city. It is a pilot solution in that target market because of its simple UI/UX (user interface and user experience) that offers the user a clean and simple interface throughout the whole experience.

It has two types of users, the clients and the agents. Clients are the users that want their meters read by the agents or by themselves. The agents are the users who go to the client's house to do the reading and report the damages called as *"Estragos"* (incidents) in the application's menus.

The web platform is available for management and administration only.It is on this platform where the administrator creates and manages everything in the database, regarding the types of users, the readings and the damages.

All of the app's design was created by me and one CoE (Center of Excellence) designer. Every single aspect of the design was thoroughly thought so that it could meet both our individual preferences and the application requirements, but it was thought so that it could provide the best user experience possible. I think with this design is possible to give the application to any user and it will always feel comfortable to navigate around the app and comprehend how it works without any problems at all (Appendix.III).

It is important to highlight that both the design and the behavior of the user interface were made to give the user the most comfortable experience as possible, because the target audience is mainly composed of people without technological literacy.

This application will be presented according to the view of each type of user (Agent, Client and Administrator) and after the explanation of the workflows will be presented the main use cases for those flows followed by an activity diagram explanation of some of those cases.

3.2.1 BackOffice – Web Portal

It is on the BackOffice (Appendix.I) that all of the information is managed, readings, users (agents and clients), incidents and images. To login on this web portal the user must have an account that has the role of administrator and access the link that was given to him, by the company. It is the responsibility of the company to give this link to the users that should have access to this portal.

The key action and navigation flows for the administrator are described on the Appendix.II.

The navigation on this portal is made by a top menu that has five links: "Clientes" (Clients), "Técnicos" (Agents), "Contagens" (Readings), "Estragos" (Incidents) and "Imagens" (Images), which will redirect the user to the correct screen. All of the navigation flows and screens are similar since it is an administration portal and the key requirement was to keep the UI/UX as simple and similar as possible. The administrator user will have the possibility to create (add), remove, edit and see lists of the users, agents, readings and incidents.

After the login is completed the user will be redirected to the first page of the portal "Lista de Clientes" (List of Clients), that corresponds to the Clients tap on the navigation menu.

On the list of clients page the administrator can see the list of clients that are registered on the platform and their main information. The administrator can also do two quick interactions with the table, first he can search for a client, by introducing his name on the search bar and click "Pesquisar" (Search) to refresh the table and he can delete the user through the "trash bin" icon. He can also see the detail page of a specific client by clicking on the id link. The user will be redirected to a page where he can see and edit all of the client's information.

The user can also add more clients to the platform by clicking on "Adicionar Cliente" (Add Client) and introduce their information on the form. To introduce the address he has two options, fill the fields manually or click on the map, it will show a red marker on the clicked location and the fields will be automatically filled.

The list of agents that are registered on the application can be accessed through the Agents tab on the navigation menu. Here the administrator can see the list of agents that are registered on the platform and their main information. The administrator can also do three quick interactions with the table, first he can search for an agent, by introducing his name on the search bar and click "Pesquisar" (Search) to refresh the table, also can see the photography of agent by clicking on pop-up icon and he can delete the user through the "trash bin" icon. He can also see the detail page of a specific agent by clicking on the id link. The user will be redirected to a page where he can see and edit all of the agent's information. He can also see the detail page of a specific agent by clicking on the id link. The user will be redirected to a page where he can see and

edit all of the agent's information and see what readings the agent did or have scheduled and all of its information.

The user can also add more agents to the platform by clicking on "Adicionar Técnico" (Add Agent) and introduce their information on the form.

Halfway through the top menu is the "Contagens" (Readings) tab, this list shows all of the submitted readings on the portal and the mobile application. It shows the main information, regarding each reading, such as the date, value, submission date, type, which client and agent it regards, address and the photography taken. The administrator can remove the reading through on the trash bin icon. To access the client's or agent's detail page it can be done by clicking on the on the respective name. To access the reading's detail page the administrator needs to click on the reading date. On this screen he can see all of the information regarding the specific reading, its date, value, type, client name which has a clickable link to the client detail page, agent name which has a clickable link to the agent detail page, submission date, photography, postal code, address (also visually represented by a red marker on the map). The administrator can edit all of the information and save the updates by clicking on "Editar" (Edit).

The user can also create readings by clicking on the "Adicionar Contagem" (Add Reading) link. On this page he needs to choose the date of the reading, the value, the photography of the meter and the submission date (if it is an already submitted reading), the type (water or electricity) and finally it needs to select the client and the agent that did the reading.

The list of incidents is accessible by the "Estragos" (Incidents) tab on the navigation menu. On this screen the user will see a list of all the incidents registered on the platform and all of their main information such

as, id, submission date, type, postal code, address, the agent that reported the incident and the photography. On this page the user can do four quick interactions with the table, he can search for an incident by filling the search input with addresses or submission dates, can see the incident's image by clicking on the zoom icon, remove the incident, by clicking on the trash bin icon, and assign a new status, "Por Resolver" (Unresolved) and "Resolvido" (Resolved). To access the incident's detail page the user needs to click on the link in the submission date column. On this screen he can see and edit all of the information regarding that specific incident, he also can change the status between resolved and unresolved.

The administrator can also report more incidents to the platform by clicking on "Adicionar Estrago" (Add Incident) and introduce their information on the form, including an image. To introduce the address he has two options, fill the fields manually or click on the map, it will show a red marker on the clicked location and the fields will be automatically filled, similar to the creation of a new client.

The last tab on the navigation menu is the imagens one. It redirects the user to the list of every image upload to the portal and application organized in tabs, "Contagens" (Readings), "Estragos" (Incidents) and "Técnicos" (Agents). These lists exist so that the administrator can control the type and content of images that are uploaded and delete the ones that are not correct or break the rules and conditions, per example obscene or bad taken images. The user can make two quick interactions with the table, access the images through a pop-up and delete the image by clicking on the trash bin icon. To access the page where the image is located just needs to click on the submission date link and will be redirected to the specific page, can be a client, agent or incident.

To summarize, the portal is coherent between all the processes and workflows. All screens are mirrors of each other so that there are not any obstacles on the user experience and navigation within the portal. This implementation happened because it was a requirement that the platform should be as simple and intuitive as possible. It is on this portal that the administrator can see, edit, add or remove readings, incidents, users (clients and agents) and images.

3.2.2 WeRead – mobile application

The WeRead it is a mobile application that accommodates two types of users: agents and clients, where they can manage their readings, incidents and profiles. The key action and navigation flows for both types of users are described on the Appendix.IV through activities diagrams.

When any type of user enters on the application the first screen that he sees is the loading screen, which contains the logo in the middle and the loading bar on the bottom. After the loading screen is finished the user is redirected to the login screen.

If the user is already registered, he can login on the application by introducing the email and password on the correct fields. If the user forgot about the password, he can recover it by clicking on "Esqueceu-se da password?" (forgot your password). He will then receive an email with a code and after a verification he will recover it. On the contrary, if the user is not yet registered on the application, he can register by clicking on "Registar" (Sign up) and then fill the form with the correct information.

Returning to the login screen, if both fields are filled with an existing and matching username and password, the login is successfully made, and the user is redirected to the homepage. The homepage it varies according to the user's type.

If the user is a "client" it will be directed to the client's homepage and respective flow. Here the he can see two reporting and summary areas, one regarding his spending, of water and electricity, and other regarding his readings, both submitted and scheduled. On the "Gastos" (Spending) tab he can see useful information regarding his water and electricity spending, such as, lowest and biggest consumption months and some statistics comparing his consumption with the national average. On the "Contagens" (Readings) tab the client can see all of the scheduled and already submitted readings as well as their detailed information, per example, date, the agent that did the reading and the value. To summarize, the client area has the reporting component and the action component, where the clients can see the most valued information of their readings.

On the other hand, if the login is made by an agent the flow and the interaction is completely different. An agent can see information regarding all of his clients, can report incidents and submit readings.

When an agent logs in on the application, he is redirected to his homepage screen. This screen is where the agent can see all of the information regarding his readings. The screen contains by a map, where the location of that agent readings are shown by red marker, a top menu, that filters the information shown on the map and a bot menu, that works as a navigation menu. The top menu has three filters that influences the information shown on the map, if the filter "Todos" (All) is active, both water and electricity readings will appear on the map, if the filter with the water logo is active only water readings will appear on the map, and lastly if the filter with the electricity logo is active only electricity readings will be shown on the map.

If the user clicks on the red marker it will be shown a pop-up containing all of the important information regarding that reading. By

clicking on "Proceder" (Proceed) the agent initiates the processes of concluding the reading, it will be redirected to the reading page, where he must take a photography of the meter, with the numbers visible, by clicking on the imagem that says "Tirar Foto" (Take Photo), introduce a value, either automatically, with the OCR, just by clicking "Tirar Leitura" (Read) or manually by clicking "Inserir valor manualmente" (Insert value manually) and then fill the mandatory field with the correct value.

To access the list of clients the agent must click on "Clientes" (Clients) on the bottom menu, accessible at all times. The user will be redirected to clients screen. On this screen the agent can see the most important information regarding each client but can also see a detailed page if he clicks on them, where he can also see all of their readings. He also has the possibility to search for a specific client by typing his name or surname on the search bar on the lateral menu triggered by a click on the magnifying glass icon.

The last tab on the navigation menu is the "Estragos" (Incidents) tab. The screen is similar to the readings one, in here the user can see a map with red markers that indicate the location of the incidents open by him. On the top exists a menu that filters the incidents shown on the map according to its type: collapse, flooding, electricity, vandalism an fallen trees. These incidents are situations that the agent sees on the street and reports them on the application so that the administrator on the BackOffice can report them to the correct entities so they can fix them.

To submit an incident the agent needs to click on the plus icon located on the right top side of the screen. Here the user needs to take a photo and submit the incident, he can also fill the input with the correct location, but it is not obligatory since automatically the application will send his current location using the GPS of his mobile phone. The objective of this functionalities is to significantly increase the repair speed of those

incidents by the responsible entities since they have an image and the precise location of the situation.

To summarize, the agent can see and report incidents, see and submit readings and access client's information.

4. Conclusion

Since the development and implementation of the project was mainly dependently on me, I had the responsibility to organize the developments the way that suited me and the project needs better. So this kind of development freedom really helped my growth both professionally (technically) and personally (soft skills). Because all of the thinking about how the app should work and look, I could combine the best technical solutions to accommodate the best design to facilitate the user's experience around both applications.

At the beginning I had some difficulties on changing my thinking and problem approaching mentality from traditional coding to low-code. Because it was required that I started thinking on a drag & drop development panorama instead of a traditional coding one.

I think that on the first few phases of the project were where I could apply the knowledge that this master's degree gave me. Specially on the phase 3 (Application Design), where I need to study both the state of the art of similar applications and the market, both mobile application market and this type of reading services market.

Regarding the technology, I think that didn't disappoint me at all, because with OutSystems the developers can always think logically speeding their deliveries since it involves less code. Nevertheless, it offers the ability for you to code whenever you want and enrich your applications with custom-made extensions created by you. And I think that these low code platforms can give way more that they are already giving to both developers and clients/users. Since I started developing without having no knowledge about the platform I have to say that, for me, two important advantages of OutSystems are that is an intuitive platform and working environment that any user can get hold of the concepts fast with the help

of the courses available on the website. The most crucial advantage is the speed that a developer can create content and implement solutions to the problem reducing the delivery time considerably.

This application is considered a pilot solution on this kind of market, particularly in Portugal and will be stated as ground zero for future applications. Will be also available on the Deloitte's CoE OutSystems factory so that it capacities and functionalities can be used for new applications developed by members of the CoE.

I am extremely grateful for the opportunity that ISEG, in particular GSI masters, and Deloitte gave me and I am really happy with the final result. I think that with my previous knowledge of OutSystem, that was none as referred earlier, and with what I achieved in only six months, the final result is really satisfiying. Both the mobile and web applications are doing what they were meant to do and all the requirements were met.

Professionally this internship opened me the doors to the job market since I'm now going to my 2nd year with Deloitte, on the CoE, and being promoted to Tech Consultant with two OutSystems certificates (Associate Web and Mobile developer).

References

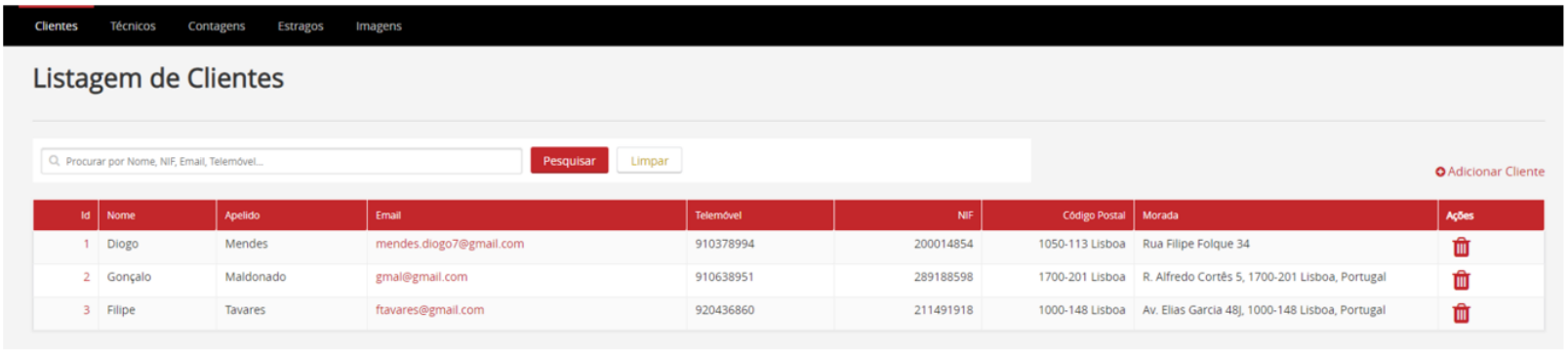
- Alshamrani, A. and Bahattab, A. (2015). A Comparison Between Three SDLC Models Waterfall Model, Spiral Model, and Incremental/Iterative Model.
- Appian. (n.d.). Benefits of Low-Code Development | Low-Code Basics, from https://www.appian.com/low-code-basics/benefits/
- Avison, D. E., & Fitzgerald, G. (2003). Where now for development methodologies?
- Clarke, A. (2006). Transcending CSS: The Fine Art of Web Design.
- Davis, A. M., Bersoff, E. H., & Comer, E. R. (1988). A strategy for comparing alternative software development life cycle models
- Everhard, J. (2019). The Pros And Cons Of Citizen Development, from https://www.forbes.com/sites/johneverhard/2019/01/22/the-pros-and-cons-of-citizen-development/#4c0577bc84fd
- Gartner (n.d.). Citizen Developer, from https://www.gartner.com/en/information-technology/glossary/citizen-developer
- Glass, R. (2001). Agile Versus Traditional: Make Love, Not War
- Govardhan, A & Munassar A. (2010). A Comparison Between Five Models Of Software Engineering.
- Guntamukkala, V., Wen H. J. and Tarn, J. M. (2006). An empirical study of selecting software development life cycle models
- Harris, R. (2019). Low code pros and cons, from https://appdevelopermagazine.com/low-code-pros-and-cons/
- Henry, S. L. (2007). Web Accessibility: Web Standards and Regulatory Compliance.
- Highsmith, J. and Cockburn, A. (2001) Agile Software Development: The People Factor
- Highsmith, J.A. (2002). Agile Software Development Ecosystems
- K. Beck, M. Beedle, A. van Bennekum, A. Cockburn, W. Cunningham, M. Fowler, J. Grenning, J. Highsmith, A. Hunt, R. Jeffries, J. Kern, B. Marick, R. Martin, S. Mellor, K. Schwaber, J. Sutherland, and D. T. (2001). Manifesto for Agile Software, from https://agilemanifesto.org/
- Krug, S. (2014). Don't make me think!: Web & Mobile Usability: A Common Sense Approach to Web Usability
- Kumar N., & Zadgaonkar, A. S. (2013). Evolving a New Software Development Life Cycle Model SDLC-2013 with Client Satisfaction.
- Nielsen, J. (1994). 10 Usability Heuristics for User Interface Design

- Nielsen, J. (2003). Usability 101: Introduction to Usability
- Norman, D. A. (1998). The Design of Everyday Things.
- OutSystems (n.d). What Is Citizen Development and How to Govern It, from https://www.outsystems.com/blog/posts/citizen-developer/
- OutSystems. (n.d.). Low-Code Development Platforms | OutSystems, from https://www.outsystems.com/low-code-platforms/
- OutSystems. (n.d.). Low-Code Myths, Fears, and Realities: Vendor Lock-in, from https://www.outsystems.com/blog/posts/vendor-lock-in/
- Paavilainen, J. (2001). Mobile Business Strategies: Understanding the Technologies and Opportunities
- Ramsin, R., & Paige, R. F. (2008). Process-centered review of object oriented software development methodologies.
- Richardson, C. (2014). New Development Platforms Emerge For Customer-Facing Applications, from https://www.forrester.com/report/New+Development+Platforms+Emerge+For+CustomerFacing+Applications/-/E-RES113411
- Richardson, C., & Rymer, J. R. (2016). Vendor Landscape: The Fractured, Fertile Terrain Of Low-Code Application Platforms The Landscape Reflects A Market In Its Formative Years.
- Rosado, P. (n.d.). About OutSystems | OutSystems, from https://www.outsystems.com/company/
- Royce, W.W. (1970). Managing the Development of Large Software Systems
- Schwaber, K. (1994). SCRUM Development Process
- Schwaber, K. (1997). SCRUM Development Process. In Business Object Design and Implementation
- Schwaber, K. (2004). Agile Project Management with Scrum
- Sheffield, J. and Lemétayer, J. (2013). Factors Associated with the Software Development Agility of Successful Projects.
- Sugai, P., Koeder, M. & Ciferri, L. (2010). The Six Immutable Laws of Mobile Business
- Unhelkar, B. (2009). Handbook of Research in Mobile Business: Technical, Methodological, and Social Perspectives
- Warren, N. (2019.). What Is Citizen Development and How to Govern It, from https://www.outsystems.com/blog/posts/citizen-developer/

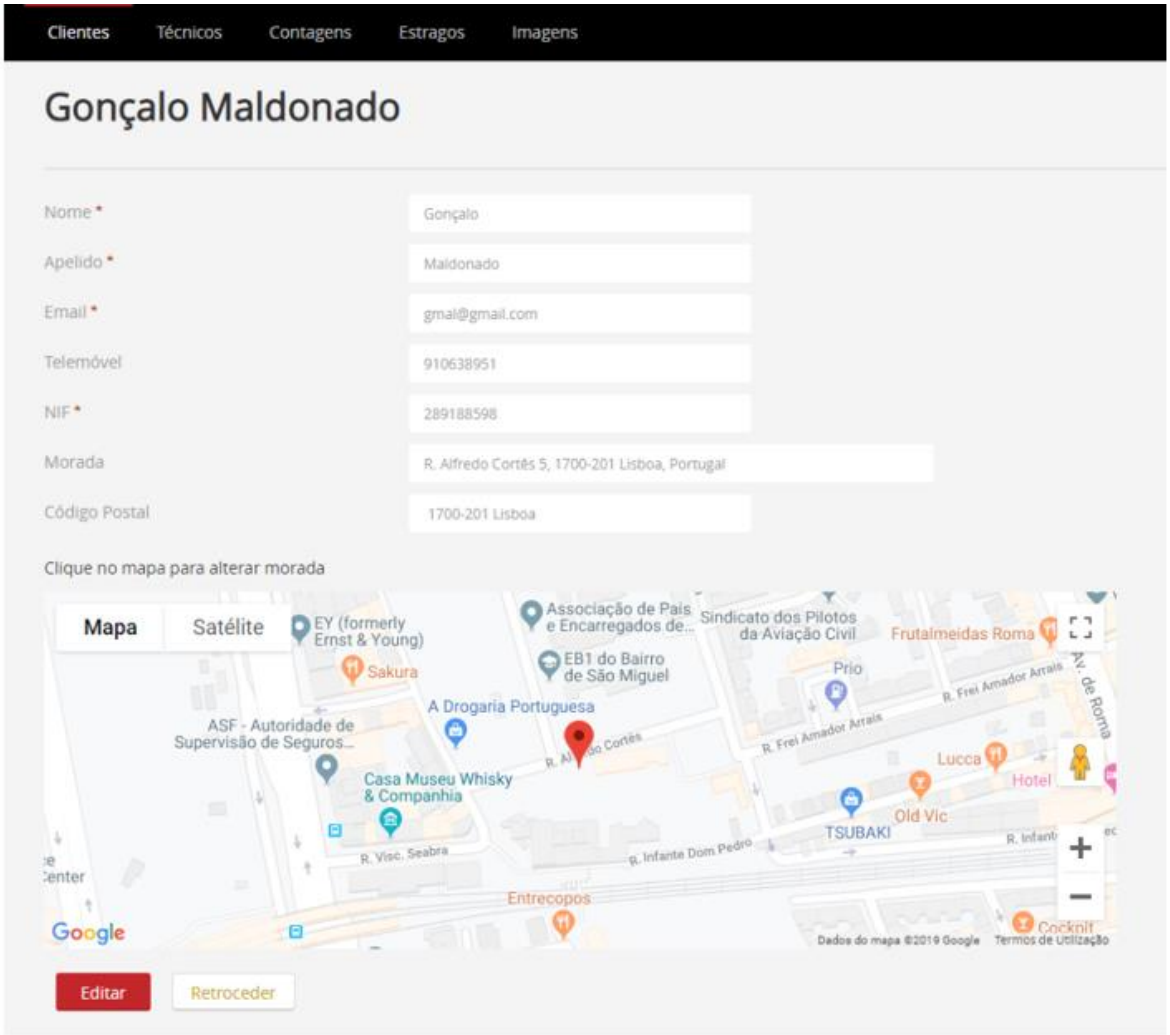
Appendix

Appendix I – BackOffice

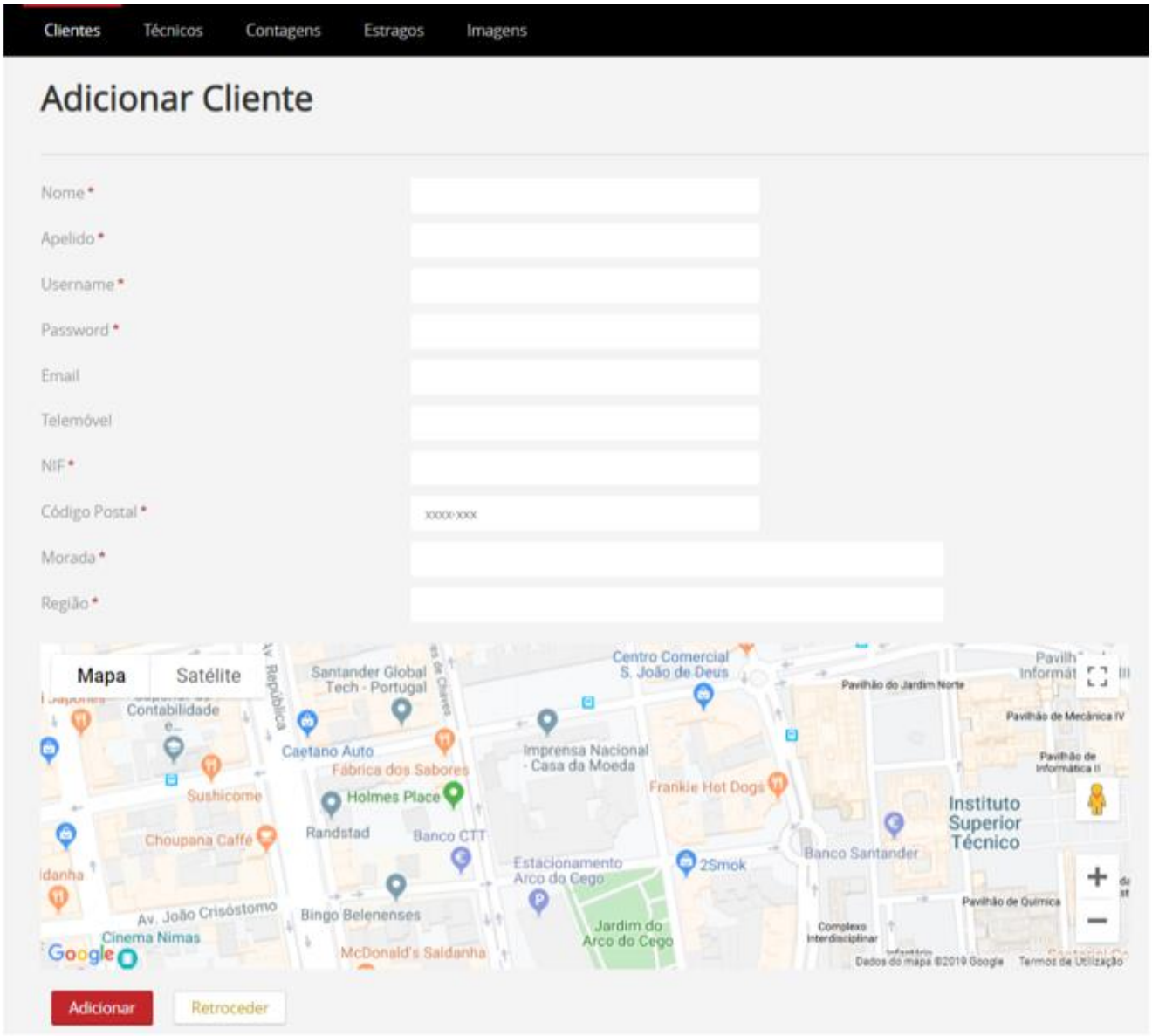
1. Client List



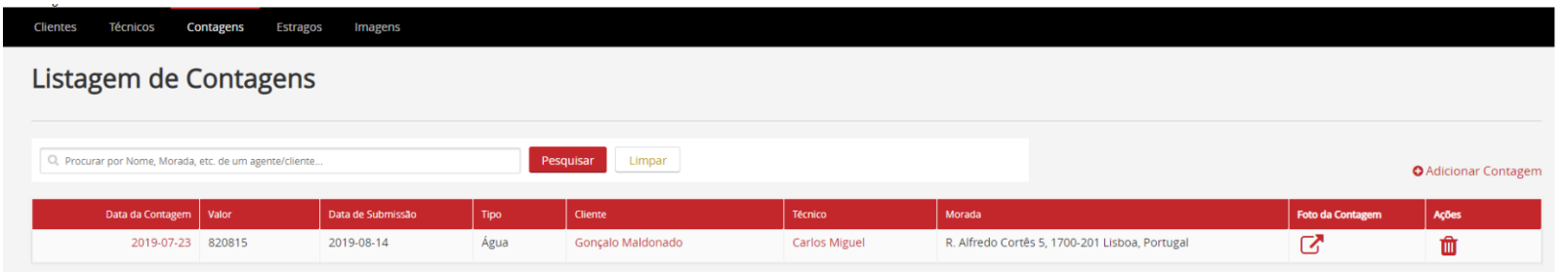
2. Client Detail



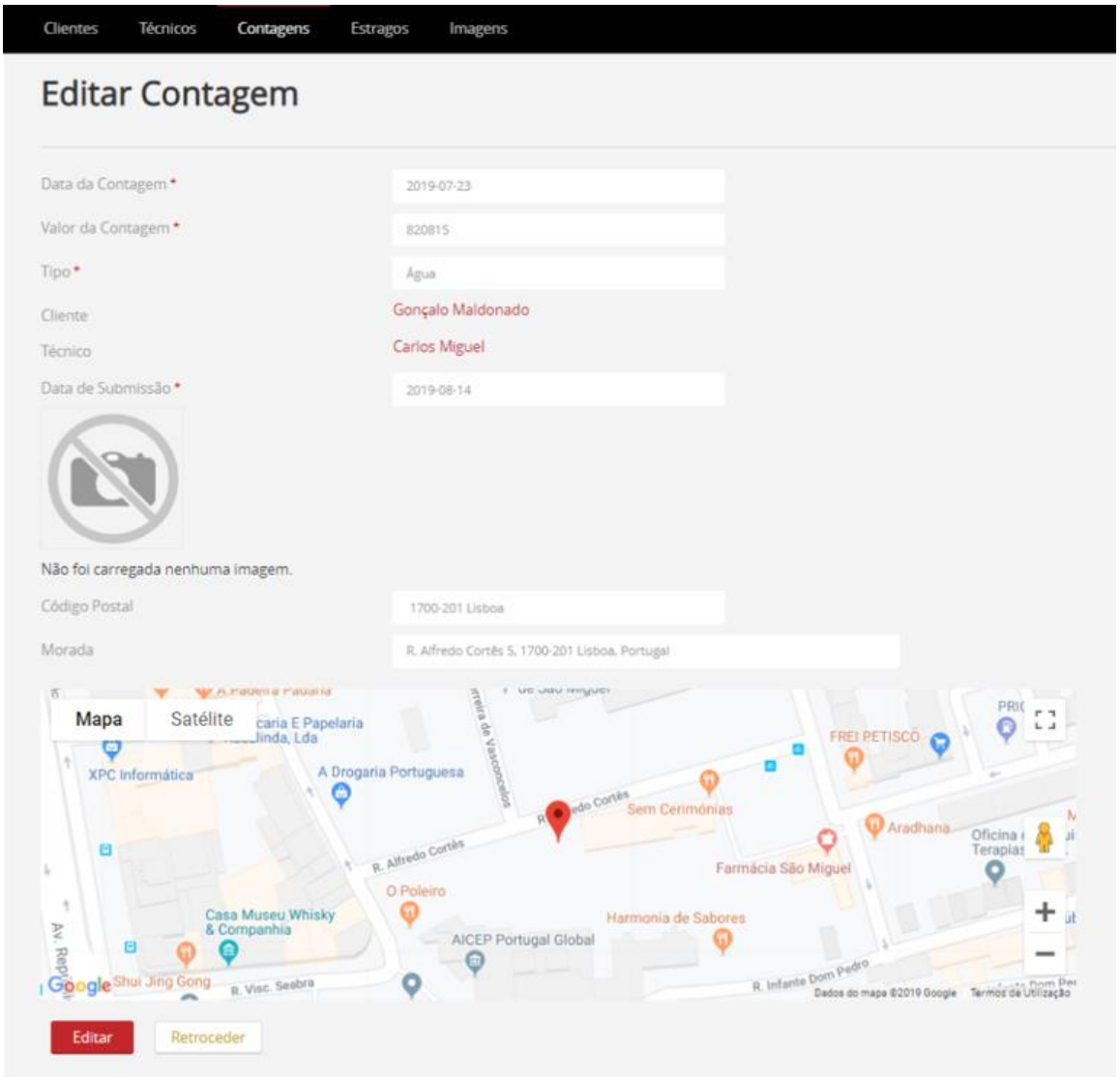
3. Add Client



4. Reading List



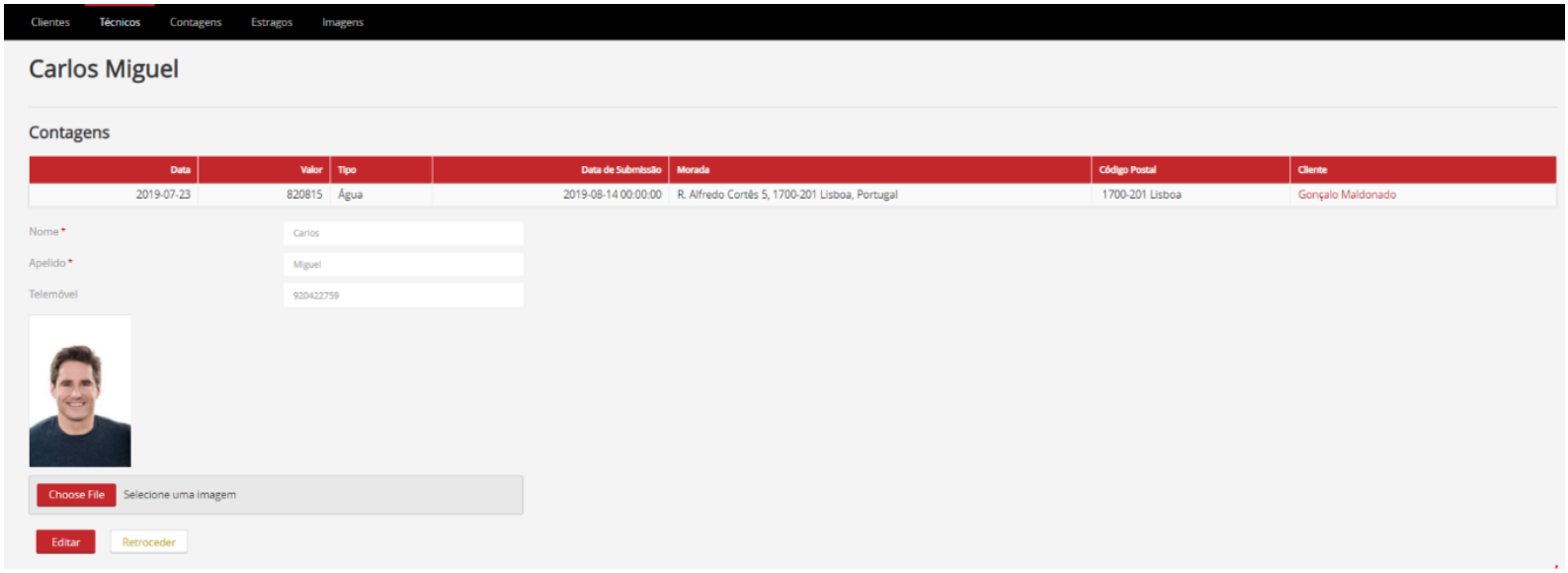
5. Reading Detail



6. Add Reading

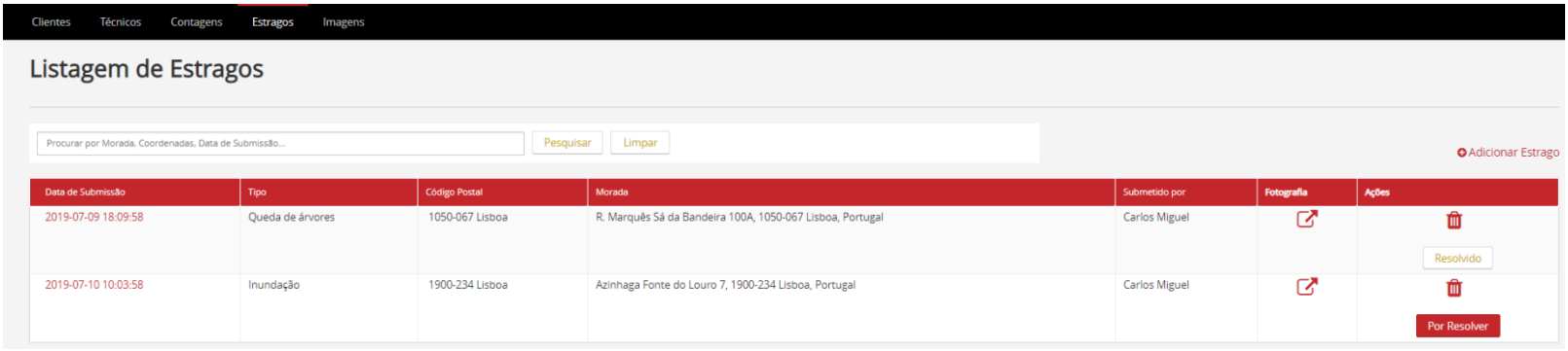
7. Agent List

8. Agent Detail

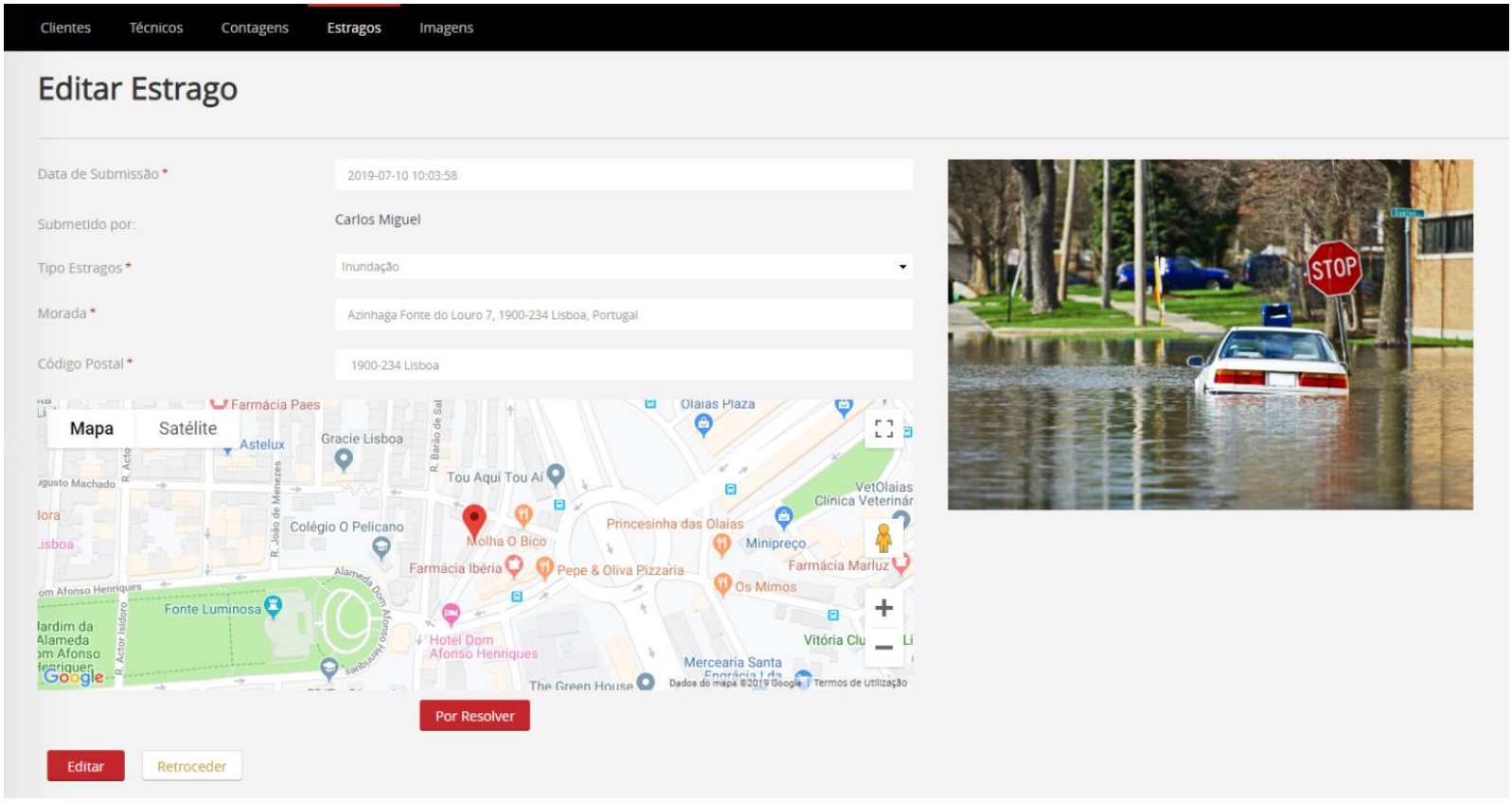


9. Add Agent

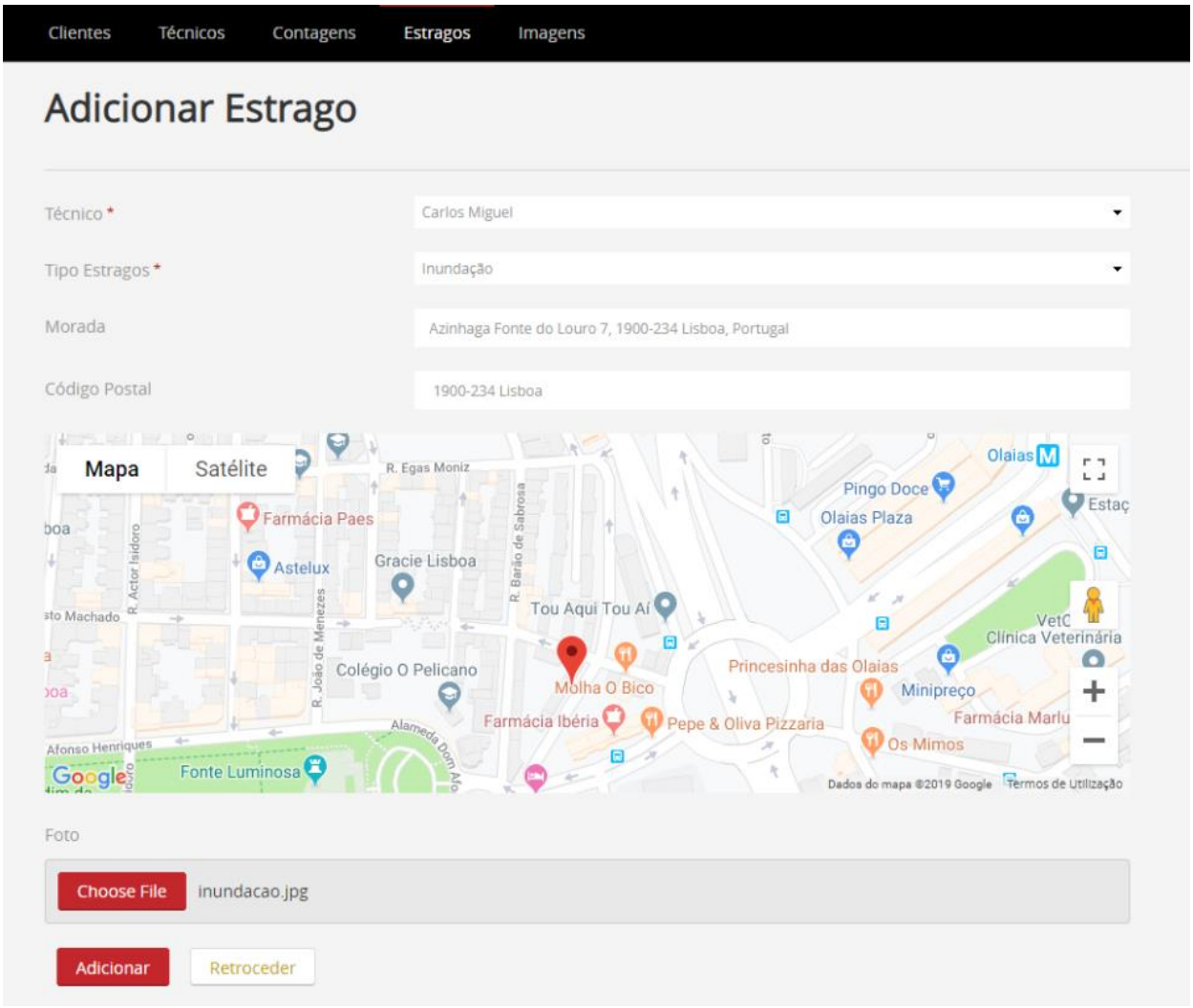
10. Incident List



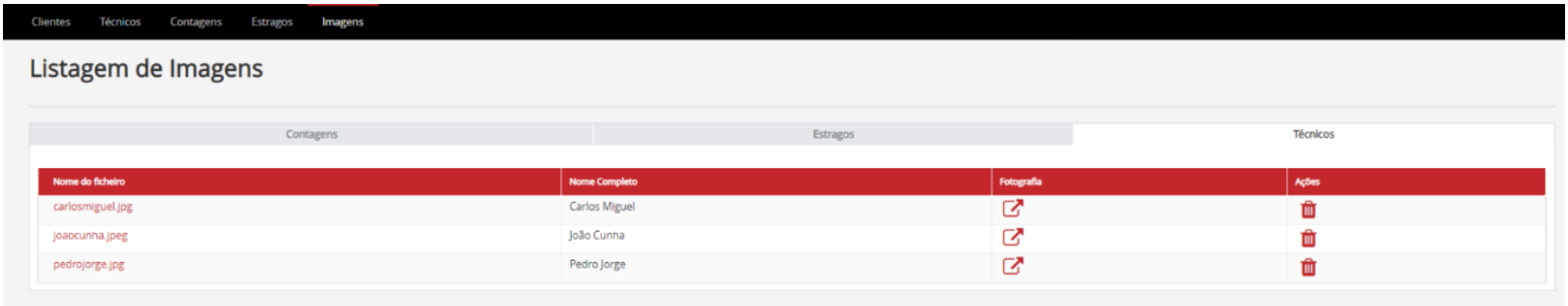
11.Incident Detail



12. Add Incident



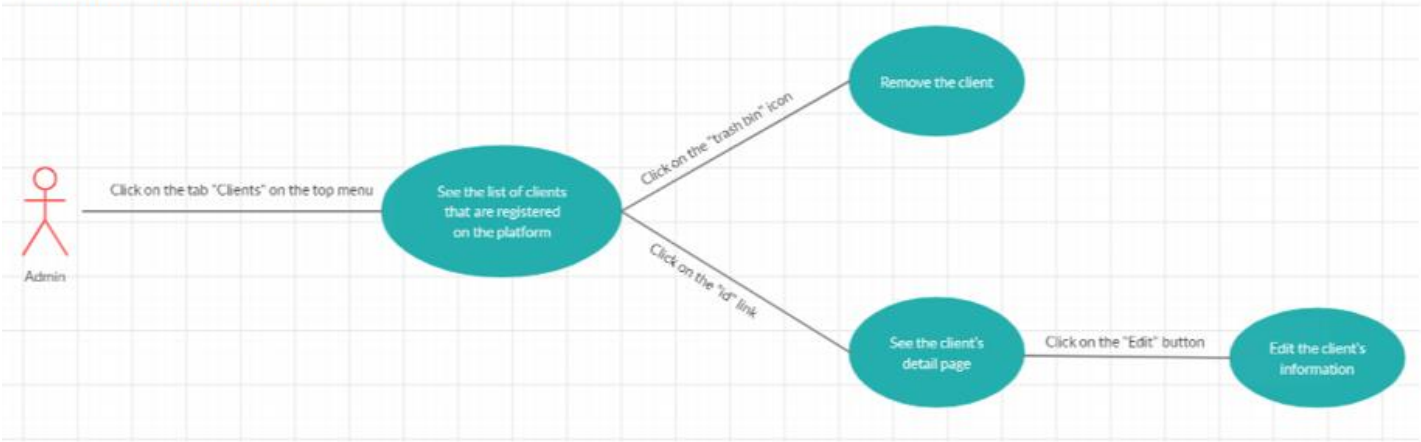
13. Image List



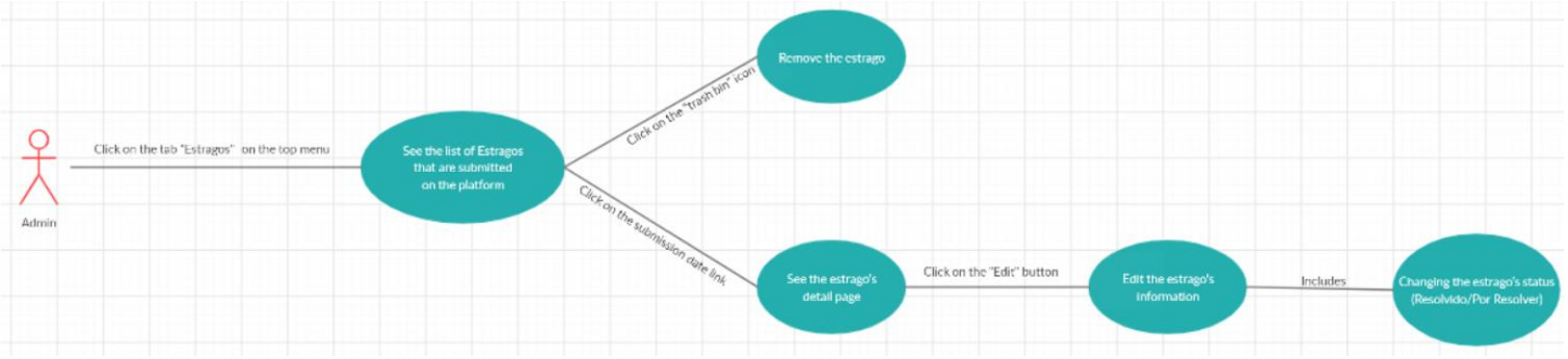
Appendix II – Activity Diagrams - BackOffice

1. Administrator - Agents Flow

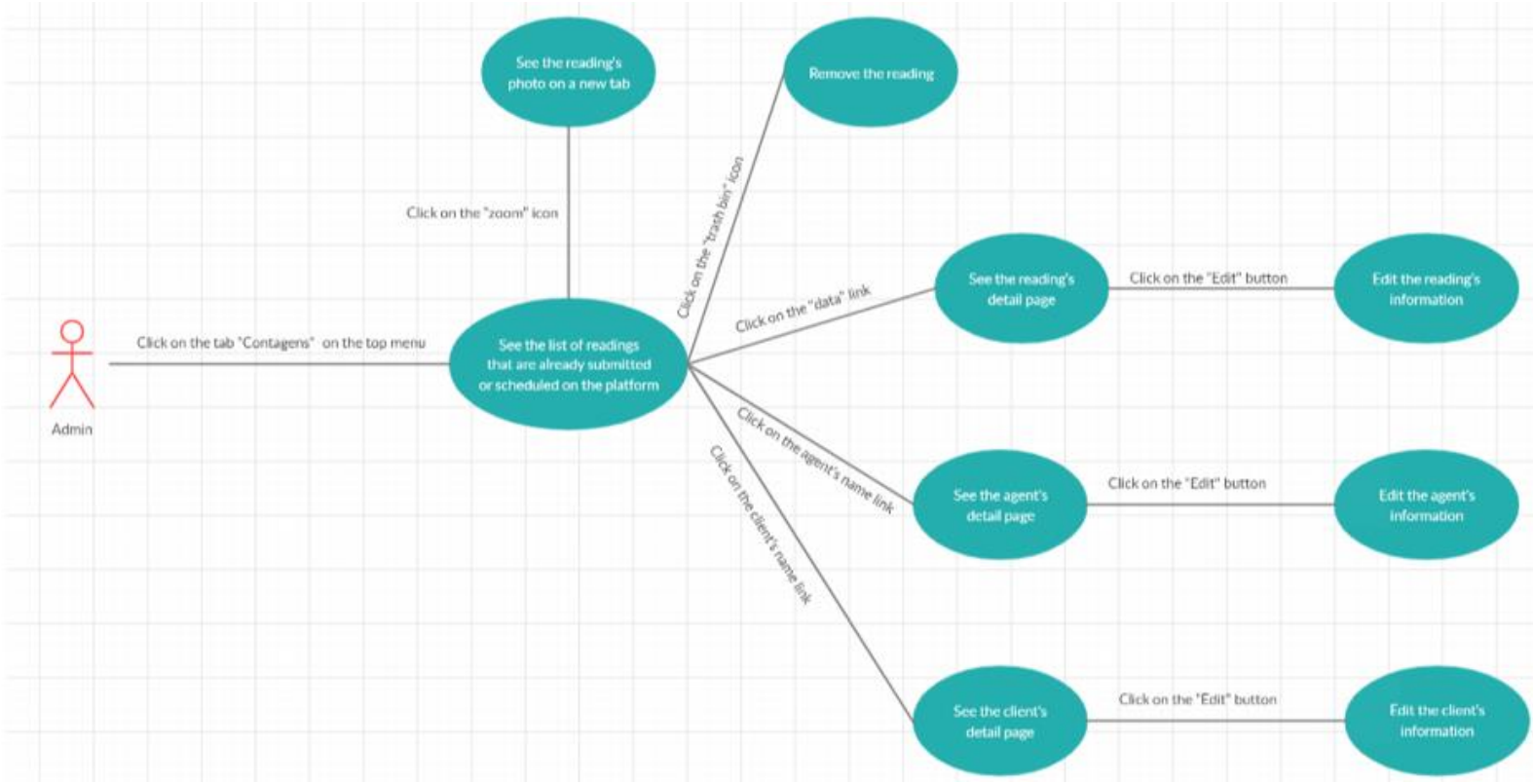
2. Admin - Clients Flow



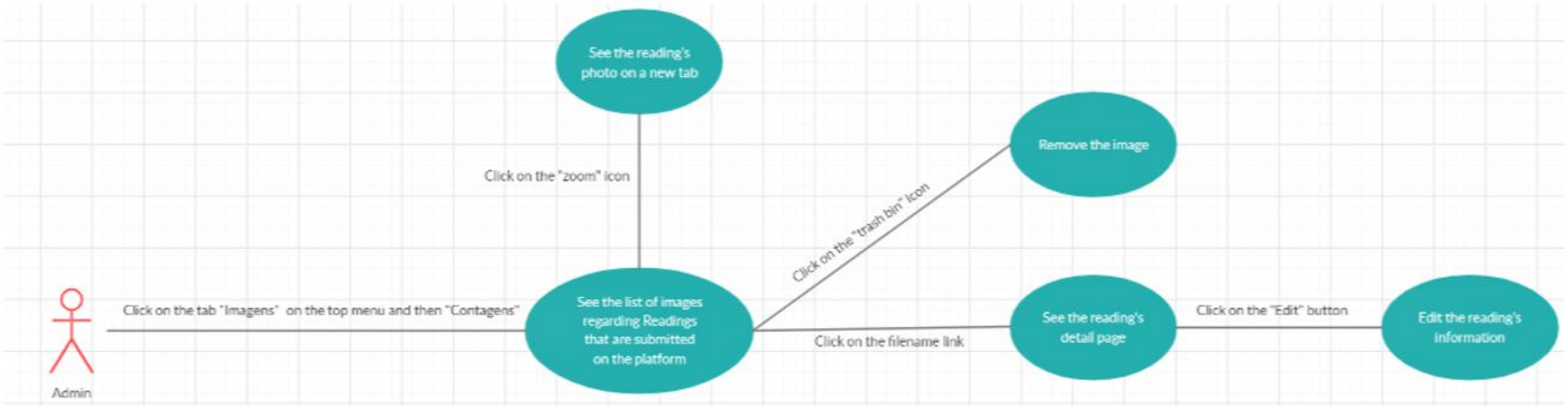
3. Administrator – Incidents Flow



4. Administrator – Readings Flow



5. Administrator – Images Flow



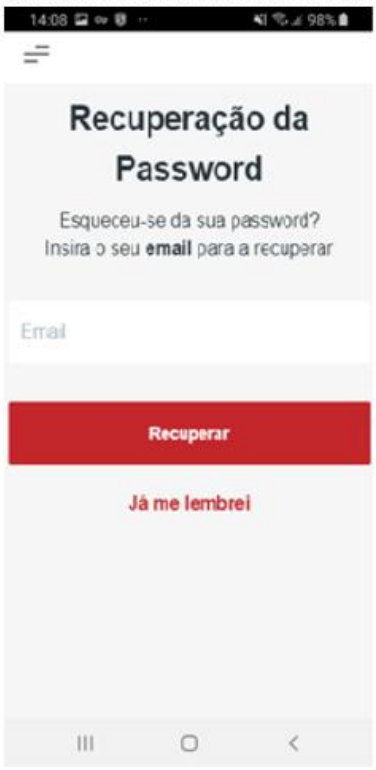
Appendix III – Mobile Application

1. Loading Screen

2. Login Screen

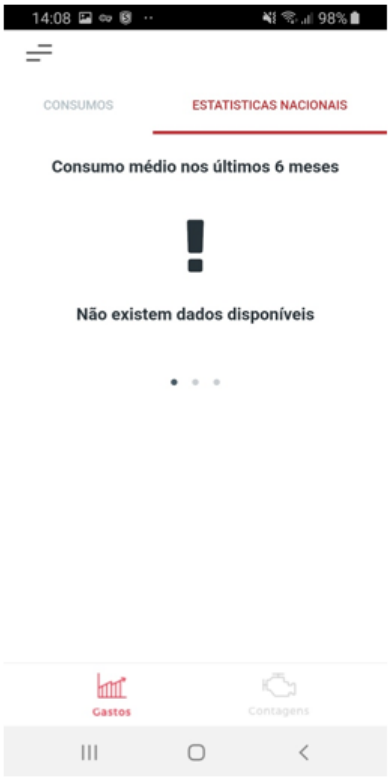
3. Signup Screen

4. Recover Password Screen

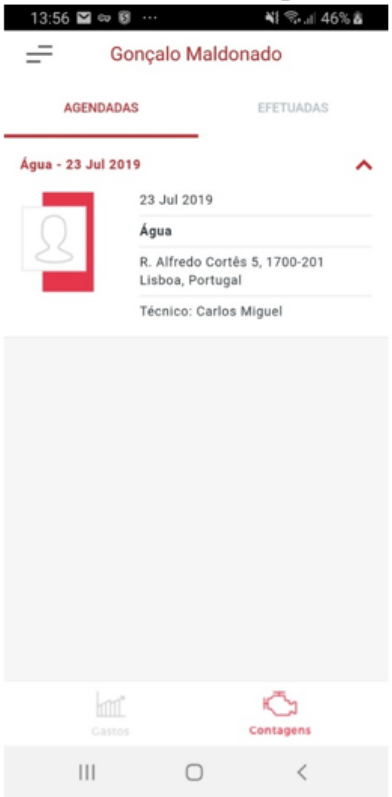


5. Spendings

6. Charts



7. Scheduled Readings



8. Submitted Readings

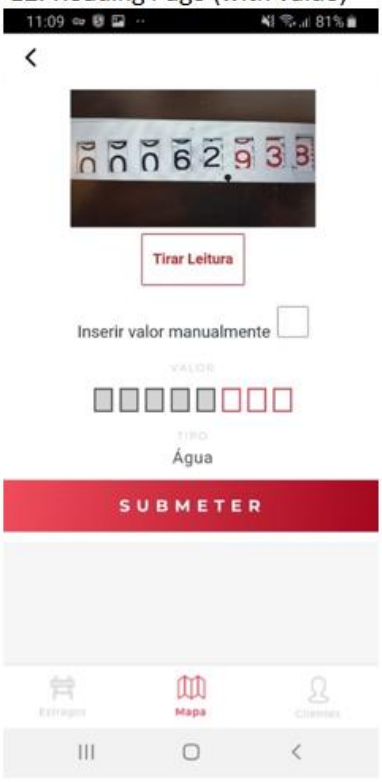
8. Submitted Readings

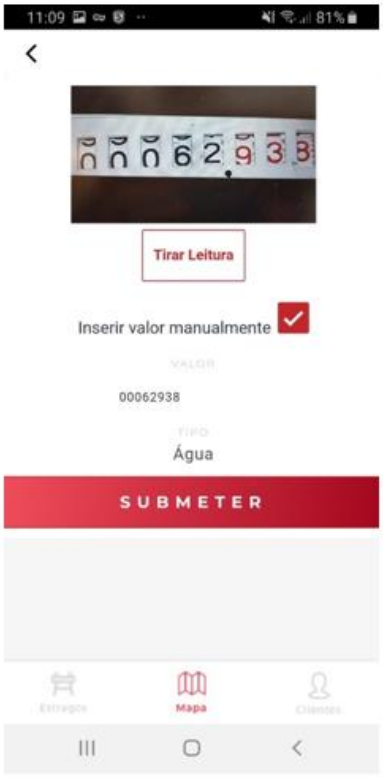
9. Agent Homepage

10. Pop-up Reading

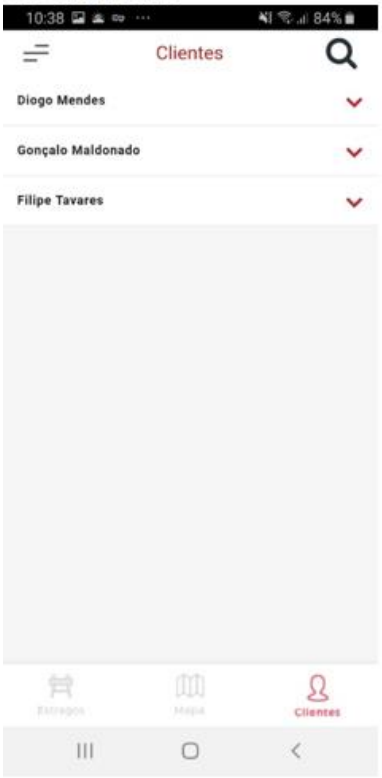
11. Reading Page

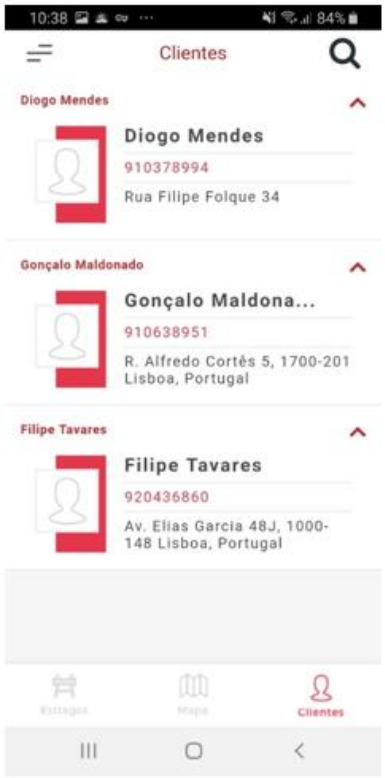
12. Reading Page (with value)



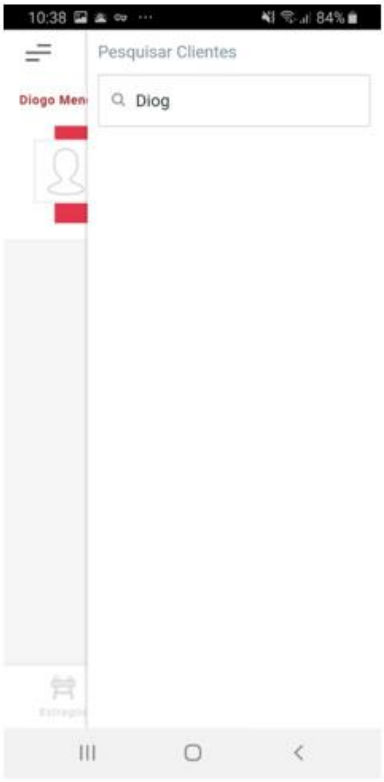


13. Clients List

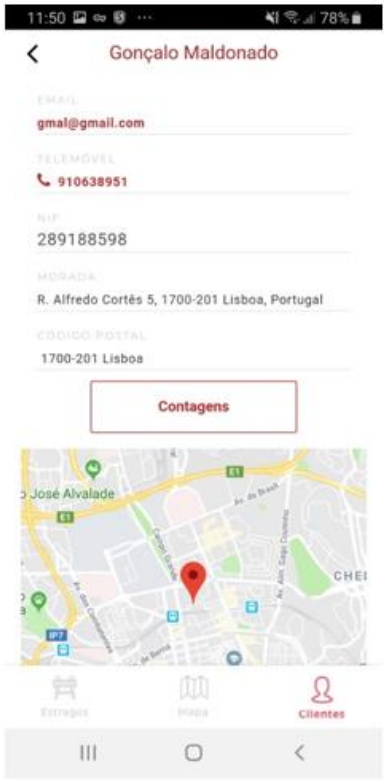




14. Client Search

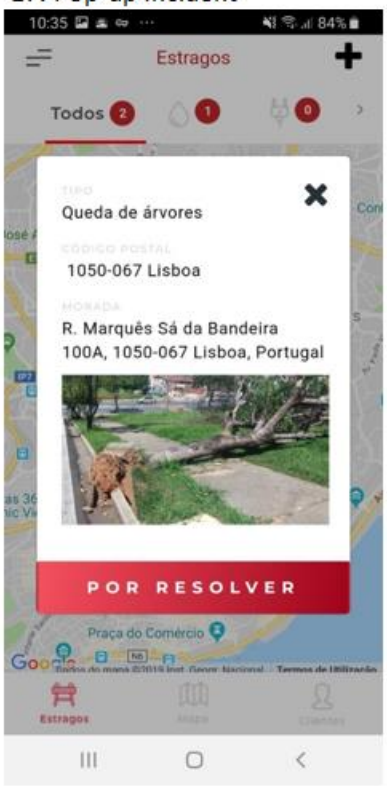


15. Client Detail



16. Incidents

17. Pop-up Incident

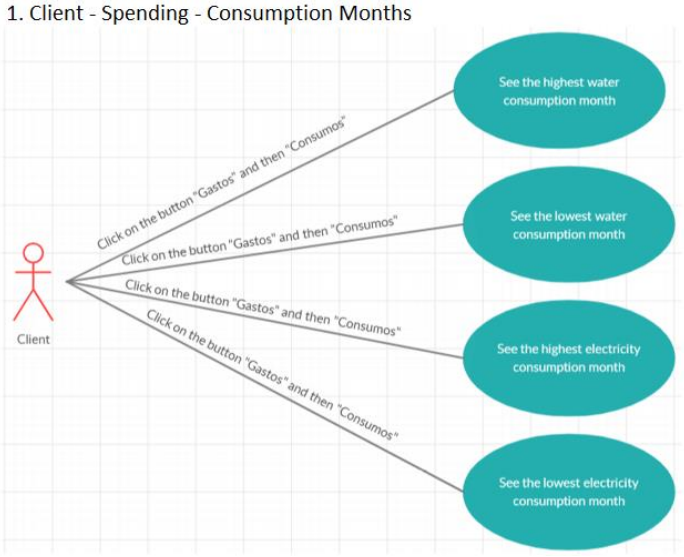


18. Filter menu incidents

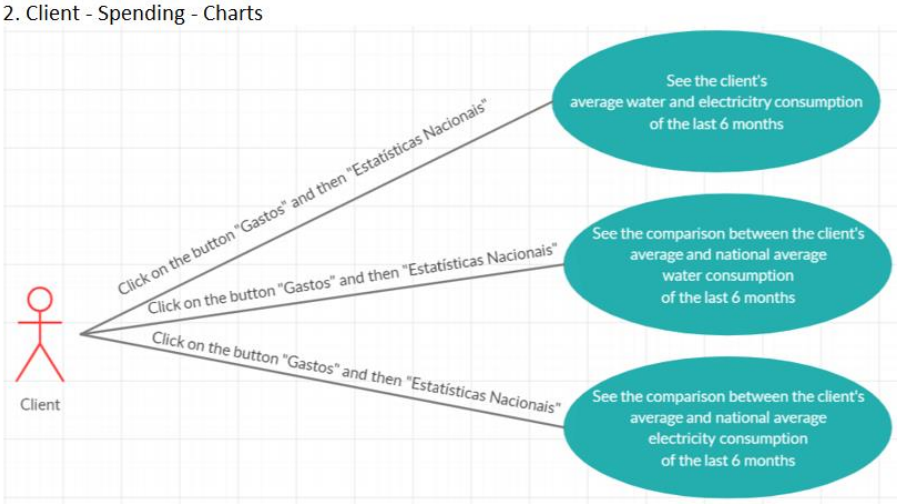
19. Add Incident

Appendix IV – Activity Diagrams – Mobile Application

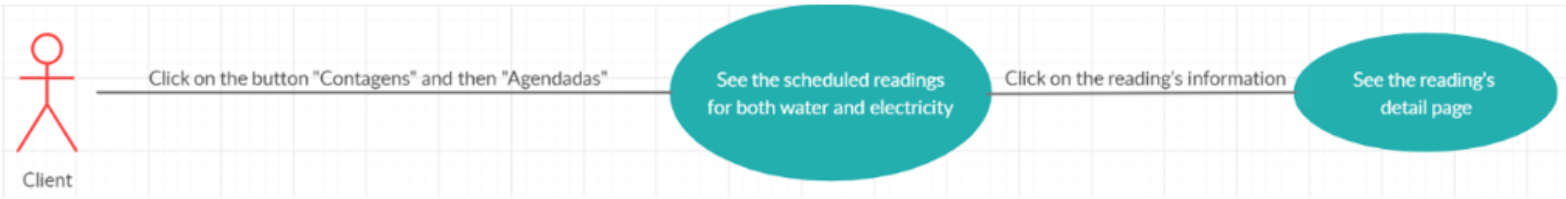
1. Client - Spending - Consumption Months



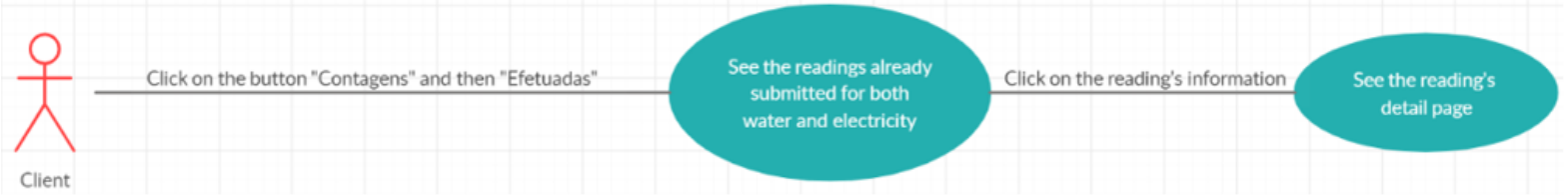
2. Client - Spending - Charts



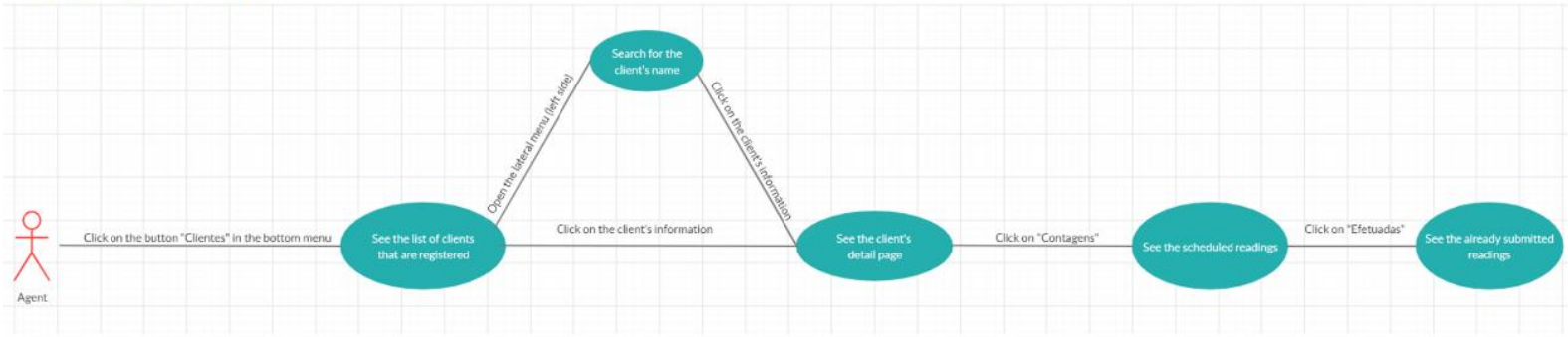
3. Client - Future Readings



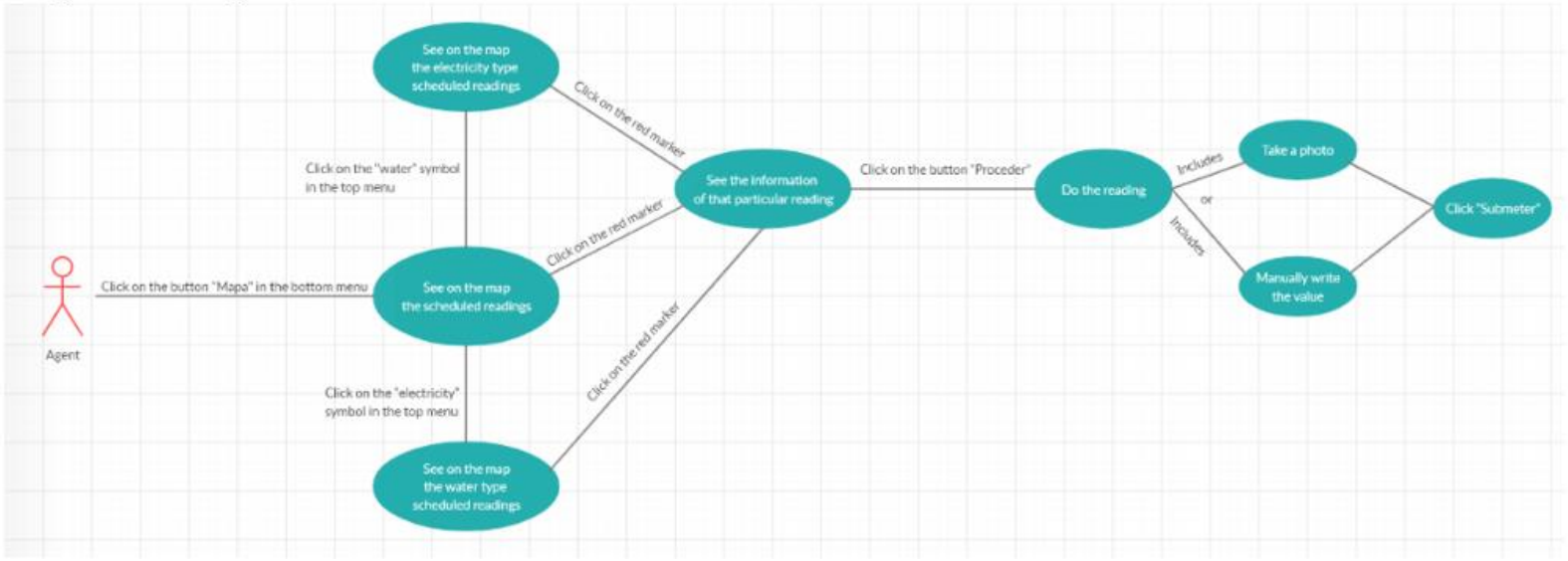
4. Client - Submitted Readings



5. Agent - Clients Flow



6. Agent - Readings Flow



7. Agent - Incidents Flow