Skill Search

Master degree in Computer Engineering – Mobile Computation

Rui Pedro Santos Lavos

Leiria, September of 2019

Skill Search

Master degree in Computer Engineering – Mobile Computation

Rui Pedro Santos Lavos

Project Report under the supervision of Professor Micaela Esteves, and Professor Ângela Pereira

Leiria, September of 2019

Originality and Copyright

This dissertation/project report is original, made only for this purpose, and all authors whose studies and publications were used to complete it are duly acknowledged.

Acknowledgments

I am very thankful to all my friends, my family and everybody that supported me at Isobar Switzerland during the period in which I was developing this project. Without them, I could not succeed on this and for that, I cannot be more grateful.

I am thankful to my coordinators for their help, patience, and support.

I am also thankful for every opportunity that ESTG gave me, allowing me to complete this project and to have taught me everything that I know.

Abstract

To improve collaborators' work life and their performance on the projects they are assigned to, it is very important to have them allocated to the correct project or projects inside the company.

This is where talent management applications come into play, allowing managers to find the best solution for their collaborators and to allow them to keep on improving their skills continuously while working.

Isobar was in need to have such type of management application since the managers in the company are still using Excel files to manage collaborators' allocations, certifications, skills, and other relevant information. To improve this, an application was created using the Salesforce Platform as a development and deployment tool. The result is a new application, Skill Search, that can be used to do the required management tasks in a simple and easy way.

Keywords: Talent Management Software, Salesforce, Human Resources Management Applications

Contents

List of Figures

List of Tables

List of Abbreviations and Acronyms

API	Application Programming Interface
CRUD	Create, Read, Update, and Delete
CRM	Customer Relationship Manager
DOM	Document Object Model
ESTG	School of Technology and Management
LMS	Learning Management System
SLDS	Salesforce Lightning Design System
SPAs	Single Page Applications
SSO	Single Sign-On
SS	Skill Search
SaaS	Software as a Service
URL	Uniform Resource Locator

1. Introduction

1.1. Background and Problem Definition

Nowadays companies around the world require different kinds of collaborators. The kind of collaborator that knows its value, its knowledge and that wants to keep on improving himself/herself. From the company's point of view, it is preferable to have collaborators that are valuable in terms of knowledge and that want to keep on improving themselves [1]. This is due to the fact that technology is constantly evolving, and companies need their collaborators to be able to follow the technology's trend.

It is not easy however to have collaborators motivated to keep learning. So, companies are adopting a methodology that keeps on evaluating the collaborators and defining objectives for them. This methodology works by defining a career plan for the employee according to his / her preference. Then, collaborators auto-evaluate themselves and get their evaluation checked and discussed with a designated manager. After this, the same manager helps defining objectives for the collaborator to accomplish at a personal level. These objectives can include certifications, exercises to improve soft skills, etc. Near the limit date to accomplish the objectives, new objectives are defined and, depending on the previous results, the collaborator can be rewarded.

The manager's job is also to keep the collaborators motivated by what they are doing. It can become stressful to be working continuously in the same project when the collaborator does not enjoy doing such tasks. However, with poor organizational tools, the manager will find a hard time reallocating the collaborator, since this can include a switch with another collaborator or even a rotation between three or more collaborators. At this point, the problem becomes noticeable since, to allow collaborators to keep improving themselves, and to be satisfied in the project or projects they are working, it is necessary to have a good management vision over everyone. This includes having all the data about everyone in terms of project allocations, skills, etc., as well as a way to read that data easily. While a company has few employees, this is an easy task to do and can be solved by using a simple excel file, for example. But when the number of collaborators starts growing, this information can become very difficult or impossible for the manager to keep.

One of the companies that felt the necessity to solve such a problem was blue-infinity – Linked by Isobar, referred to as Isobar in this document since that at the start of this year it became Isobar Switzerland [2], [3]. Isobar is part of the group Dentsu Aegis Network, a multinational media, and digital marketing communications company headquartered in London [4].

In Isobar teams are organized by division and practice where a division contains one or more practices. The Division Managers coordinate the division in terms of client acquisition. The Practice Managers coordinate collaborators of their practice, assigning them to new and existing projects and make sure that everyone is being productive, that everybody is learning and that their practice is working smoothly. In Portugal, there are different teams working in different areas like Salesforce Platform, Salesforce Commerce, Quality Assurance, Business Automation, Front-End & Mobile and some others. Usually, each collaborator completes projects that belong to their practice, however, it is becoming common to have collaborators working together from different practices.

Practice managers have been using different methods to manage the collaborators of their practice in different ways. There is no standard defined by the company and so each one of them is using what is more convenient. Most of the practice managers have one method in common, the excel file. Each has its own version of it, storing different data in different ways. Despite this, they all agree that it is not the best method, knowing that it is not intuitive, easy to keep up to date, and that could be more functional.

At Isobar, Excel is the software that is currently being used by the managers in the talent management process of their collaborators. Managers keep the information of their collaborators and their skills locally, as well as the certifications that each collaborator has. Manually, they add new certifications acquired by collaborators to the file. Additionally, they also add the information about the projects that the collaborators are allocated to. With this information, they can use cross searches to find the necessary or allocated skills to each project. However, it would make it more difficult to update the information about the skills each collaborator acquires while allocated to a project. Also, every time that a new project surges, the manager needs to know what its collaborators are doing and what their skills are. Using Excel, this entire process is time-consuming, but this process could be done in a matter of seconds with the proper tool.

1.2. Objectives and Goals

The objective of this project is to develop a tool using the Salesforce Platform that can aid the practice manager to do manage employees' allocations, skills, certificates and other information related to them in the most generic and simple way possible.

Aside from transforming the existing excel files into a working database with a visual layer to work with, the project should also include a way to visually understand all the information related to the collaborators, using reports and the application layouts intuitively. To help answer questions related to the collaborators' expertise, an application that makes intelligent use of the database should be built as well, fetching collaborators by applying filters. The search should not be restricted to the defined filters. It should retrieve the best matching collaborators. This is an important feature since it will help find the best collaborators to work in new projects or projects that are in need of more help.

The application should also be easy to use so that the practice manager can have a clear view of the information. Managing this kind of information should be a task that is easy and fast, for the manager so that the best decisions can be easily made. Since an important part of the project concerns managing collaborators' allocations and absences, the manager should also be capable of reading this information visually.

It should also be possible to extend these functionalities to collaborators with other higher roles in the company like Technical Leads or Developer Leads. It can be of interest in some practices to have these types of lead collaborators managing the collaborators on their projects so that they can aid the practice manager.

From the description of the project objectives, the requirements were defined alongside the client. The list of requirements is then composed of:

- Create the application using the Salesforce Platform product.
- Transform the existing excel file containing employees' allocations and skills into a functional database that, besides this information, also stores certificates, absences, and clients. The application should be easily customized.
- Use Salesforce reports to create some standard reports that can be run every day, week, or month to find the skill gaps between the collaborators.
- Create a page within the app where collaborators can be filtered when a collaborator with a specific set of skills is required.

- Create a visual calendar where all the allocations and absences of a certain collaborator can be found.
- Create different user profiles in the platform with distinct permissions associated.

1.3. Report Structure

This document is composed of 7 chapters, including this first one:

- The present chapter contains the background of the current problem as well as an explanation of how Isobar teams are divided.
- The second chapter presents the state of the art related to the problem at hand.
- The third chapter contains all the methodology applied to this project, as well as the release planning.
- In the fourth chapter, all the architectural decisions in terms of development are made, including the decisions in terms of the data model. A description of each object and its purpose is done throughout the chapter alongside the functioning of Salesforce databases.
- Chapter five is where all the development process is described. It starts with the Salesforce Platform explanation and then, an explanation of all the development decisions.
- Testing is the topic discussed in the sixth chapter and includes both unit testing in salesforce with an explanation of how it works and how necessary it is, as well as user acceptance testing by some individuals inside Isobar.
- The seventh chapter is the conclusion and it includes the project conclusions as well as future work that could be done in order to improve the application.

2. State of the Art

One of the most common problems within different companies is the management of employees' allocations, skills, absences, certificates, etc. In order to find the correct employee to allocate to or rotate from a project, or even to make reports on which skills are the most required to fulfill the company needs, managers have been using software like Microsoft Office Excel. Powerful software like this can handle some Human Resource Management tasks, however, they can be poorly intuitive and functional while completing them.

This chapter contemplates the description of some applications and software that are currently available online and could be used to solve the problem either partially or even fully. These applications allow the user to execute similar functionalities to the ones that will be developed thus allowing for further analysis of the requirements.

The criteria used to select the applications described in this chapter includes both recommendation by Isobar collaborators and a selection of the best results while researching terms like "Skill Management Software" or "Talent Management Software".

At the end of the chapter, the different software on the chapter are compared with each other.

2.1. Skills Base

Skills Base [5] is a talent management software that is cloud-based and in which it is possible to track and report on skills and interest levels of the company's collaborators. This software is built using a data model allows the user to add locations of different offices, add people, qualifications, roles, skills, and team. An example of the form used to add a person to the system can be seen in Figure 1. This allows for a good talent management experience with various types of reports at the click of a button. It is also possible to export all the data in ".csv" files either by using filters and options on what to export or simply by exporting a different file for each "item" of the data model.

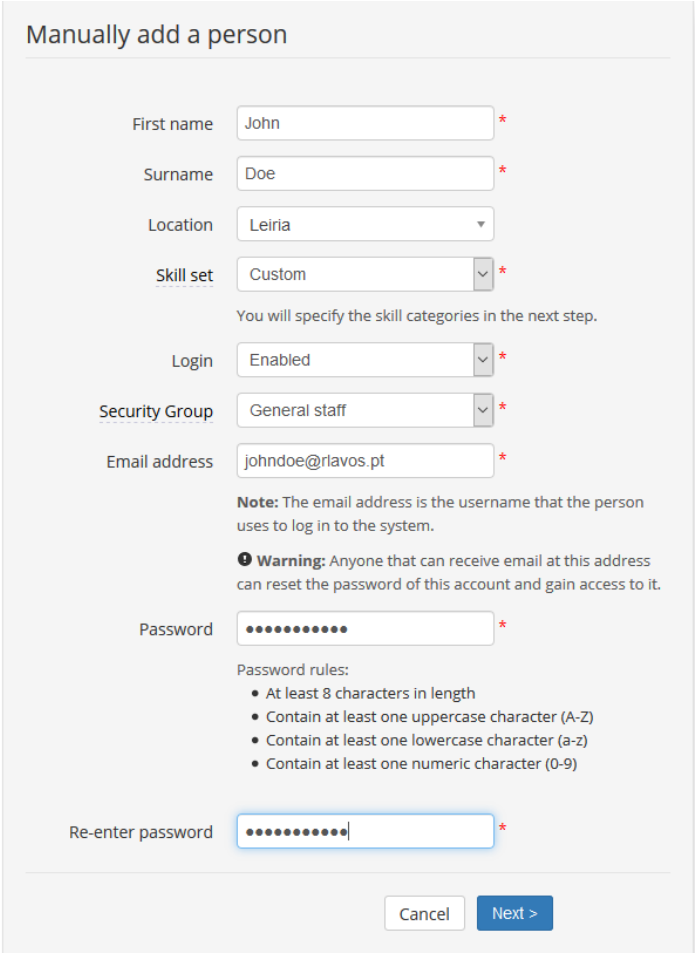


Figure 1- Inert Person on SkillsBase Example

This software is free up to twenty-five people in an organization, and this allowed to test the application and conclude that it is easy to use and configurable. Unfortunately, many interesting features are only available on the paid version.

The paid version of this software allows collaborators to participate in the experience as it is possible to give each one of them a Single Sign-On (SSO) permission for them to log in to an online portal. Here, the collaborators can evaluate themselves in terms of their skills, qualifications and other personal information. However, this is one of the many features that are not covered in the free version. Some other examples of the paid version features are Application Programming Interface (API) access, scheduled reports, data migration, skill set analysis, etc.

Examples of companies that use this software are SafeNet Consulting, Sendero Consulting, and Eltherington group ltd [6].

2.2. m|ployee

m|ployee is a paid AppExchange application that runs either on a computer or mobile device browser and promises to deliver recruitment tools, human resources management tools and, some other features. As this is a paid application, none of its features were tested. Despite this, m|ployee provides a list of features and some illustrative images of what their software is about [7].

Some of the Human Resources Management Tools include Onboarding, offboarding, Organization, Job Role, Remuneration, and Employee management. As for the Recruitment tools some examples are Social Media Integration, Skill Search and Matching, Interview Management and a Candidate Portal.

2.3. Docebo Perform

Docebo [8] is a Learning Management Software. It is a cloud-based software and allows integration with Salesforce. It started as a Learning Management System (LMS), but in the meanwhile, a new "add-on" was created.

Docebo Perform [9] is a Skills Management Software that was created with the intention of being used along with the LMS. Basically, Docebo Perform is intended to be used as a skill gap finder between collaborators. With Docebo it is possible to add collaborators, skills and many other types of records to the database that will help to better

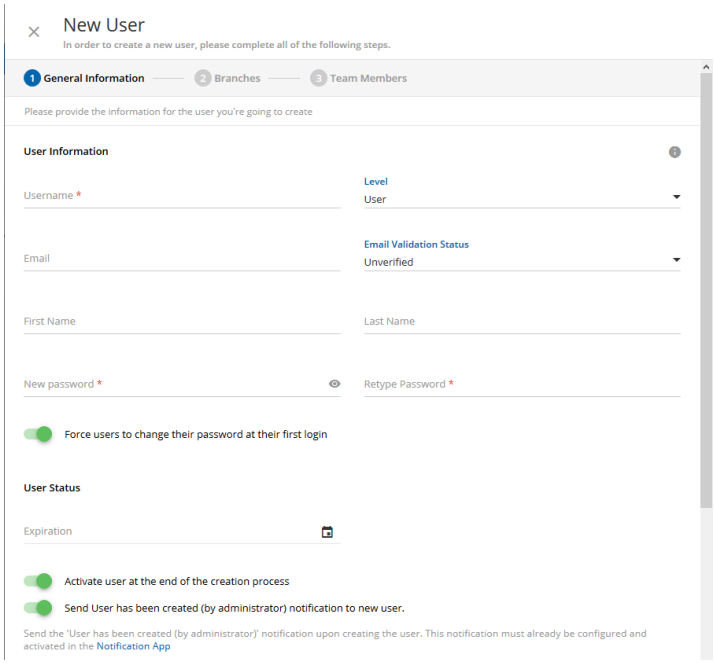


Figure 2 - Add New User to Docebo Platform Example

understand the skills of each collaborator as well as other information that might be crucial when using the application for any reason. Figure 2 contains the form to add a new collaborator to the database in order to exemplify the fields that are requested.

The functionality that stands out when researching about this software is that, as a manager, by choosing a set of skills that is appropriate for the company in which the software is being used, collaborators gain the ability to self-assess themselves on that same set of skills.

The self-assessment process is done by, after logging in to the platform, dragging skills into different boxes on the screen containing the different levels of expertise. After this process is finished, the manager evaluates the collaborators by selecting a skill, and then, dragging the collaborators to the level of expertise in that skill. This helps to find if any collaborator has a different opinion about their expertise from their own manager.

After this process, it is possible to create reports about the data and discover the skill gaps that exist between collaborators on different skills. The LMS offers the opportunity to create courses for the collaborators that need them to learn and improve their skills. In Figure 3, there is an example of the form used to create a new course.

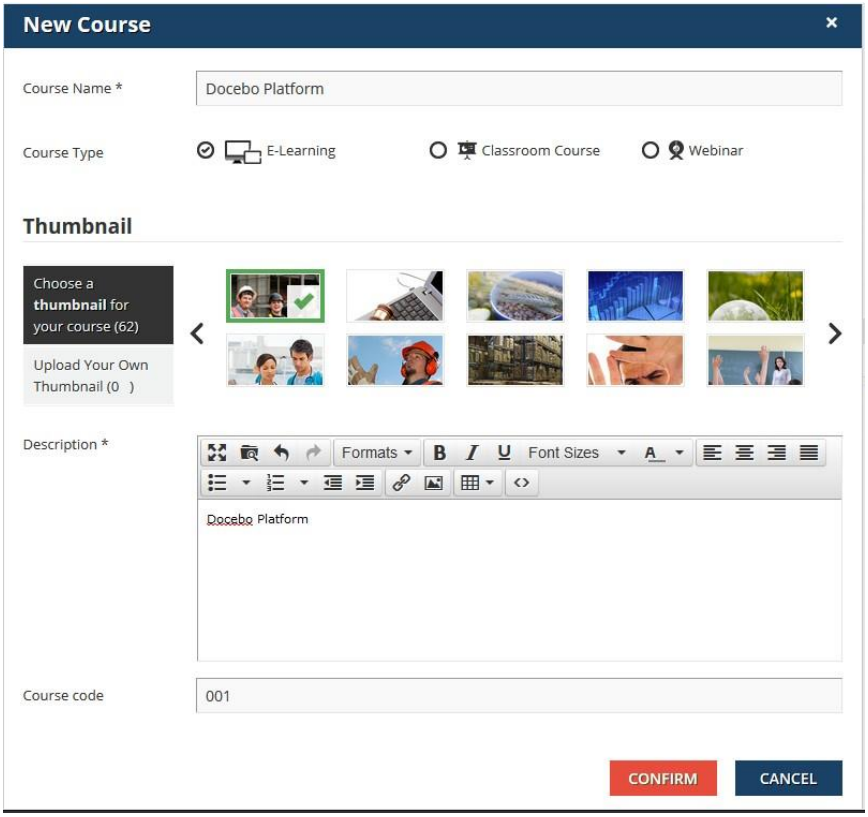


Figure 3 - Add Course on Docebo LMS

Unfortunately, this is a paid software and the free trial only includes the LMS, so the Docebo Perform was not fully tested to its potential.

Docebo is recognized every year by the industry with many awards. Some companies that recognize Docebo as the software with best use of training, learning or other similar features are L'Oréal and Swisscom.

2.4. Microsoft Excel

Microsoft Excel [10] is a very powerful software that enables the user for a great number of different features. While managing human resources, this software can do many important things like generating reports with charts, saving data, use filters to gather specific information, but, unfortunately, not many other features will be helpful while managing this sort of data. Making this the incorrect software to do so.

As described in chapter 1, Excel is being used at Isobar as a management tool. However, it reveals many limitations as, for example:

- Manual update of the skills acquired by the collaborators.
- Information update about which projects, if any, a specific collaborator is allocated to.
- Skills required for a collaborator to have in a new project, as well as the verification of which collaborators have such skills and if they are available to integrate the new project.
- Allow the manager to analyze the missing skills in its collaborator. This way, the manager can foresee and bridge the lack of skills for new projects.

2.5. Float

Float [11] is a powerful cloud-based management software that aims to manage employees' project schedules as well as project timelines and allocation planning. Has integrations with commonly used software like Outlook Calendar, Slack or Google Calendar, for example, and allows SSO with Okta [12] or OneLogin [13].

With Float, it is possible to have a calendar displaying all the allocations to projects for multiple employees at the same time, as well as drag these allocations or events to update their dates or hours dynamically, with alerts sent to all the collaborators involved. These were the main features that the company wanted to replicate in the long term since one practice is already using it and rating it as the powerful tool that it is.

Not every feature that Float has is listed in the requirements but, to have a calendar view similar to the one in Figure 4, is a requirement.

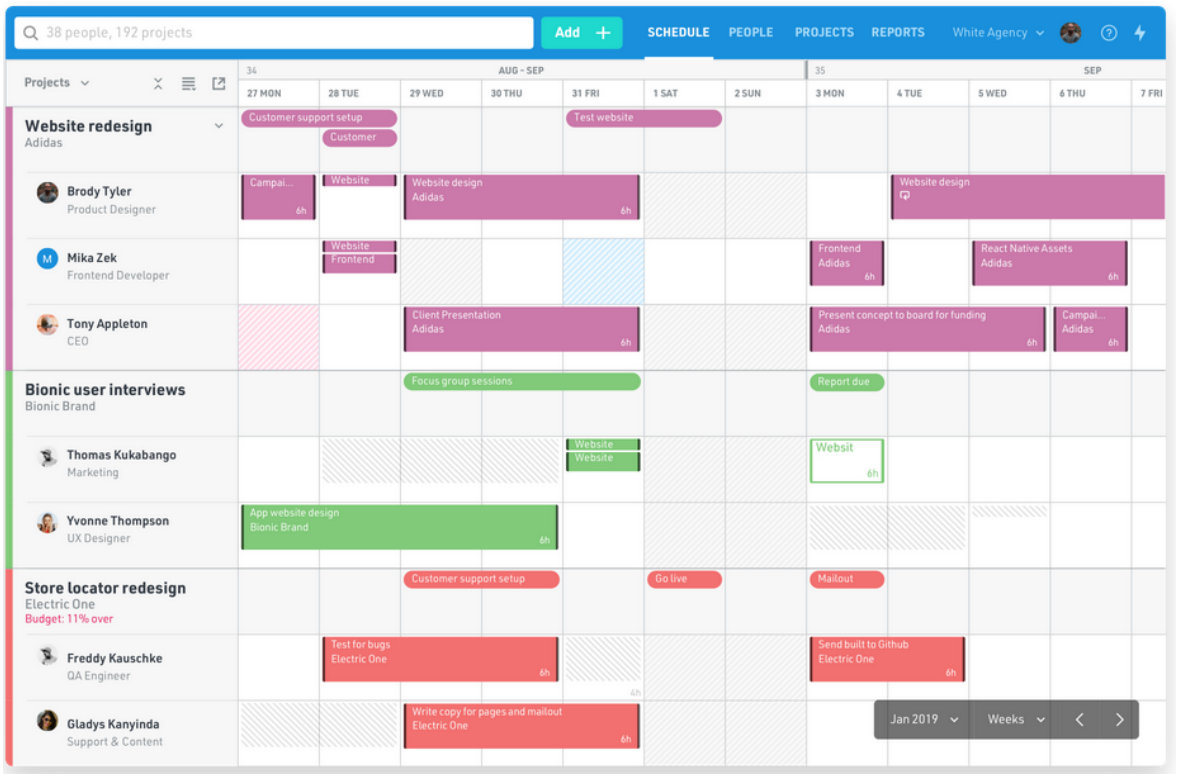


Figure 4 - Float Multi-Project Planning example (https://www.float.com/how-it-works/project-planning)

Float also has some other features such as reporting and forecasting project budgets seen in Figure 5.

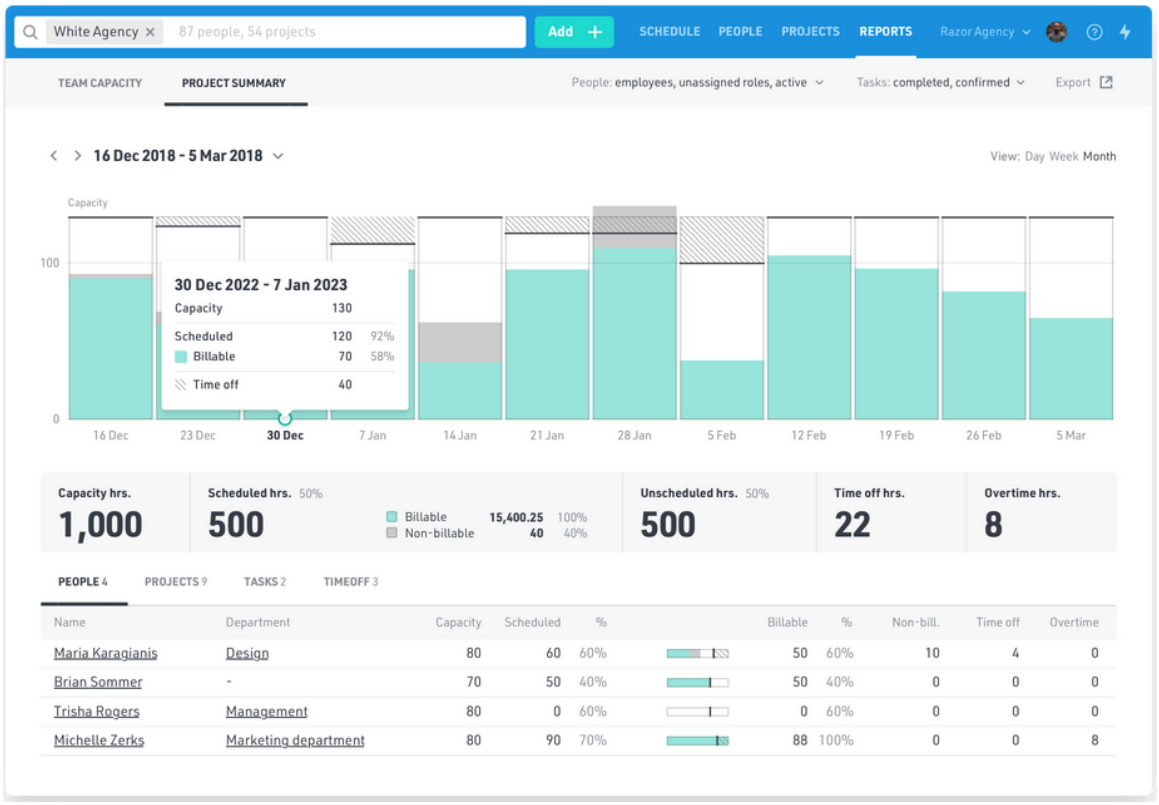


Figure 5 - Float forecasting feature (https://www.float.com/how-it-works/reports)

However, this software was not considered a possibility to fulfill the requirements related to its functionalities as it has a cost of five dollars per person, per month. This means that with around 200 collaborators, just in the Portugal offices, it would cost one thousand dollars each month just to have all the employees in Portugal in the application. As mentioned before, one practice inside the company is already using this software but since it is the practice with fewer collaborators, the cost is affordable.

2.6. Software Comparison

After all the software were analyzed, Table 1 was created comparing against each other using the requirements defined in chapter 1.2 as required features that they should have. This allows to understand which application(s) would best suit the requirements given at the beginning of the project.

Float won't be considered in this comparison since its features are more organization oriented than the ones of the rest of the applications.

The requirements present in this table are:

- Salesforce Platform, or Salesforce Integration, since it is required for the final product to be developed using the Salesforce Platform and installed in a Salesforce environment.
- Data Management, because management of information about the collaborators was the requirement that created the necessity to define this project and so it is the most important requirement to be considered on the comparison table.
- Reports, as it is very important to have the possibility to create records that can even notify the user.
- Login or Single Sign-On due to the fact that Salesforce allows users the possibility to log in and have their own environment according to their profile/preferences, which was also considered for this comparison.
- Custom Fields, since the application may need customization for different practices or companies.
- Intuitive, as the application must be intuitive and easy to use.

Table 1 - State of the Art Comparison

	Skills Base	m|ployee	Docebo Perform	Microsoft Excel
Salesforce Integration	By API		X	
Salesforce Platform		X		
Data Management	X	X	X	X
Reports	X	?	X	X
Single Sign-On	X			
Login		?	X	
Custom Fields	X	X		X
Intuitive	X	?	X	

Every time that, by using an application or learning from its details that it can achieve a requirement, a cross is placed on the table. The question marks presented mean that it is not clear whether the application achieves the requirement or not.

By comparing the amount of requirements fulfilled by each application present in Table 1, it becomes obvious that Docebo Perform and SkillsBase are the best software to complete the requirements, even if they do not fulfill every requirement, they do achieve the most important ones like data managing, report creation and have either login or SSO features.

While Docebo Perform announces Salesforce Integration, meaning that Salesforce will recognize Docebo's data as Salesforce objects without the need of mapping the data, SkillsBase does not offer this unless the developer creates a connection via API between SkillsBase and Salesforce, which is a much more complex task.

Even though the m|ployee application is the only other researched software for this project, that exists inside the Salesforce Platform context, this was the least desirable one as there is not enough information to fully understand how it works or what its capabilities are.

3. Methodology

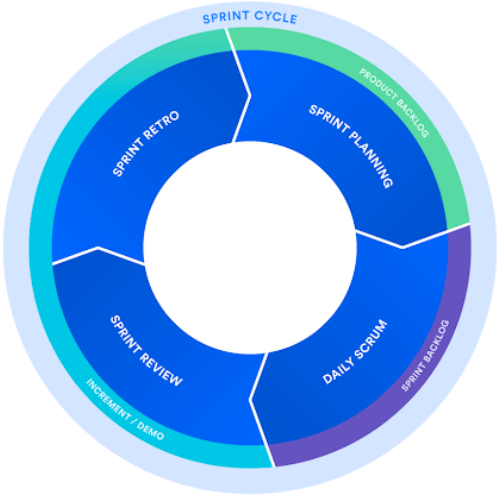
While planning the project development details, some key aspects needed to be defined. Such aspects include the development methodology, the actual requirements of the project and their conversion into user stories. Methodologies allow the optimization of the project development. In this chapter, a briefing about agile methodologies can be found, as well as an explanation of the framework used to manage the project as well as the release planning executed.

In order to have a successful management of the project development, it is key to define a good methodology. There are many development methodologies that can be used and each of them has its pros and cons. In some methodologies, there is the possibility to keep adapting the methodology itself until it gets optimal for the project. This is called Agile [14]. Agile methodologies can be used in many ways, having different rules and ways of doing specific things, these are called frameworks and the framework selected for this project was the Scrum framework [15]. Scrum is inserted into the agile group of methodologies and is the most utilized to manage projects in the Salesforce Platform practice in Isobar. This sufficed to select the Agile methodology and Scrum as a framework.

In Scrum, there are three main roles: the development team, the product owner or owners and the scrum master. While the product owner is the one who says what the application should be doing and defines the requirements for it, the development team oversees the development of those same requirements. The scrum master [16] is responsible for holding everything in place by making sure that the developers are not being impeded by anything and that their work is progressing, as well as scheduling the different meetings like sprint reviews, sprint planning, and standups, and alerting in the case of requirements being updated by the client [17].

Scrum is composed of a series of events (as seen in Figure 6) that happen in each iteration. Each iteration, also called sprint, usually lasts two to four weeks [18]. Before the project starts, a project backlog is defined with the requirements [19]. These requirements are divided into features that are referred to as user stories. Then, at the beginning of each sprint, a sprint backlog is defined with some stories that the development team aims to complete before the sprint ends. During the sprint, there is a daily meeting (or standup) [20]

between the development team and the scrum master. It is possible for people with other roles to join these meetings, but these never include the client. Each person explains what they have been doing in the previous workday and what they aim to do in the current day, considering that the meeting is in the morning. If it is at the end of the day, then each one explains what they have been doing and what they will do in the following day. This meeting helps each person to understand what the rest of the team is doing, to ask for help or the resources that are needed if that is necessary and to understand the status and direction of the project. At the end of each sprint, or when a considerable number of stories are completed, there is a demonstration of the developed work to the product owner, who can refine some requirements and approve or disapprove what has been done. Sprint reviews usually include the client in order to give them a summary of what has been done and what is intended to be done in the following sprint [21]. Sprint planning is done usually without



the client and it is where user stories are estimated and discussed [22].

Figure 6 - Sprint Cycle (https://www.atlassian.com/agile/scrum)

In this project, even though the rules described above should be followed for every project that follows the scrum framework, there were some changes to those rules. There were no daily meetings. These were done biweekly instead. The sprints did not always have the same duration, as they ranged from two to four weeks, depending on the difficulty estimated for each of the user stories. This was done as an adaptation since the first stories defined in the product backlog were of an easier difficulty and required different capabilities to accomplish. Also, before any of the sprints started, there was a course to better understand some advanced strategies in Salesforce projects architecture. After that, there were 8 sprints in total.

3.1. Task Management

In order to make better use of the scrum framework, one task management tool is commonly used. There are many different tools to accomplish this such as Trello [23] or JIRA [24], the one selected for this project. The choice of this tool was because it is the one that is most used in the Salesforce Platform practice. This type of tool allows the scrum master to manage the tasks for the developers to complete and to measure of the developers' performance.

To track the status of each story, there is a *kanban* board [25] divided in different statuses logically ordered. These statuses can be selected depending on the project needs or the scrum master preference and the ones used in this project were "Awaiting Development", "In Development", In Demo" and "Completed". They can be seen as places where the stories are. Primarily, all the stories sit inside the project backlog, and then, when it makes sense to start working on them or they gain some sort of priority, they go into the "Awaiting Development" column. As soon as a developer chooses a story to start working on, the same developer should move that story into the "In Development" column. At the end of the development, the story is put into the "In Demo" column. At this point, there are two options, either the functionality of the story gets approved by the owner and is moved into the "Completed" column, or it is not approved and goes back into the previous column to be improved or modified. Alongside the story's different status options, there is also the possibility to add comments and other indicators and information to each story. These are helpful when there are questions, doubts or even other necessities in terms of management of the stories.

3.2. Release Planning

As a best practice for release planning, Salesforce recommends the usage of at least two sandboxes before the final release of the developed functionalities. The first sandbox if for the development and the second one for testing [26]. This allows the quality assurance team to test all the functionalities in a different environment than the one used by the developers while developing functionalities. Only after the functionality tests are successfully completed and approved, the functionalities are migrated into the production sandbox where they can be used by the final users.

In the scenario of this project, since the functionalities are not meant to go into the existing production sandbox during the time given to complete the requirements, all the development, and testing of the functionalities were planned to be executed in the same sandbox. This allowed saving time creating and configuring a second sandbox just for testing purposes.

4. General Architecture

Like any other technology, Salesforce does not escape the rule of having its own aspects that allow a distinct experience when developing it and using its applications. Salesforce uses its own language to query the database. It even has two, Salesforce Object Query Language (SOQL) and Salesforce Object Search Language (SOSL) [27], but for this project, only SOQL will be used. There are also many known patterns that can be used to structure code in terms of separation and architecture, in this case, since Salesforce is the context, the used pattern was Separation of Concepts (SOC) [28], which Salesforce recommends.

The content of this chapter includes the description of the code structural architecture used in this project, as well as the data model created. Detailed explanation of how it can fulfill all the user requests, alongside the selected coding patterns is described too.

4.1. Architecture

As for this project's architecture (in terms of code structure), the best option is, currently, and recommended by senior developers and code architects, the SOC business pattern. SOC is also highly recommended by Salesforce in the Trailheads courses [29]–[31]. Since there were no special requirements at the pattern level or even the necessity to build this project on top of an existing one, there was no issue with selecting the code structure desired. By doing this, there was an opportunity to enforce the Apex Best Practices, allowing great scalability and performance. Another pattern that is used in this project was the Trigger Handler pattern. This one, despite not being documented by Salesforce in its trailheads, it is highly recommended by the senior developers of Isobar and has many variations according to what the developer needs.

The SOC pattern consists of three layers: the service layer, the domain layer, and the selector layer. Each of these layers is applied onto each object allowing for three SOC classes per object. This might look like too many classes for each object, knowing that aside from these, there will also be a "trigger handler" class for each object that requires so. However, this structure allows for an easy understanding of the code and great scalability. Each SOC class has a different purpose. The service class is where all the logic related to the object is stored, allowing for code modularity, thus reusing it whenever it is needed. The domain layer

is where the logic of the Data Manipulation Language (DML) operations is found. This is used to keep the code clean, in addition to having logic that enforces sharing rules, allowing for an easy data privacy structure. The selector class contains all the different SOQL or SOSL queries for the specific object, again allowing for modularity and clean code.

Figure 7, Figure 8, and Figure 9 [32]–[34] contain a representation of the service layer, domain layer, and selector layer respectively.

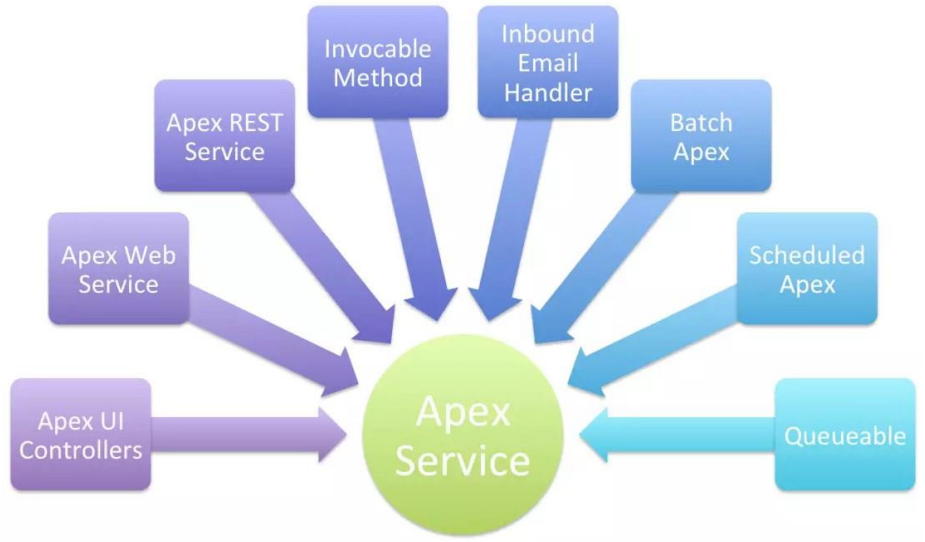


Figure 7 - Service Layer Representation

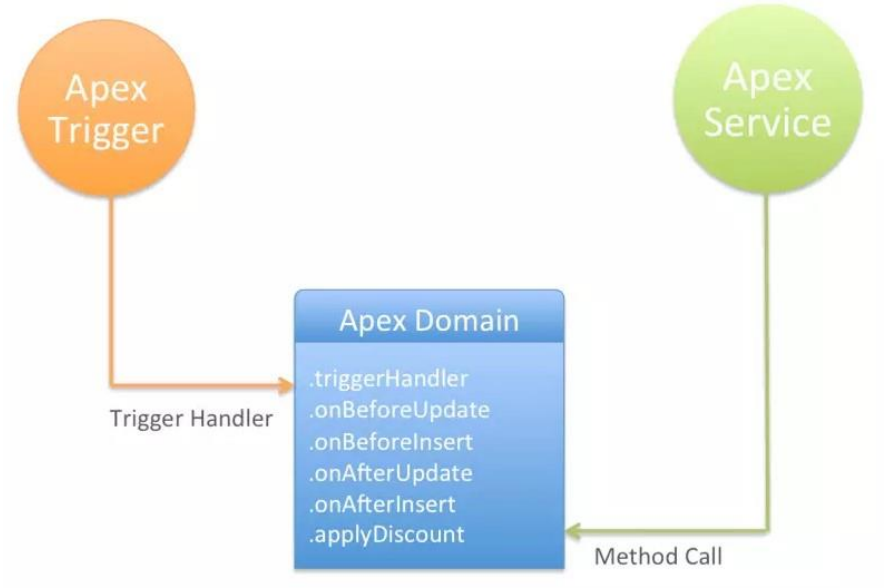


Figure 8 - Domain Layer Representation

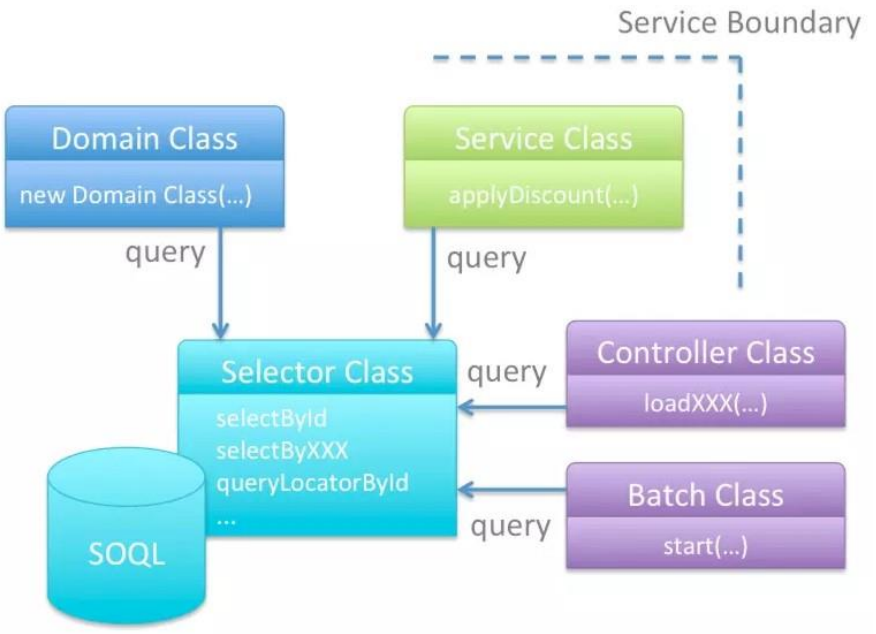


Figure 9 - Selector Layer Representation

Additionally, there is the possibility of one more SOC class in case of the necessity to have a service class that does not enforce sharing rules (assuming that all the "main" service classes are enforcing sharing rules as recommended).

Salesforce sharing rules are exceptions to the default level of access users have to each other's records, which is called Organization-wide defaults. Sharing rules are usually only used to give users additional access to records and cannot be stricter than the Organization-wide defaults [35].

The trigger handler pattern selected for this project is a simple one, meaning that its purpose is to manage the triggers for the different objects, manage when those same triggers should be bypassed or not and their maximum loop count. There are many more different trigger handler patterns that include other features prepared for bigger projects and other requirements. However, those would not make this project better due to its low dimension. The trigger pattern and Salesforce triggers are going to be better discussed in chapter 5.

4.2. User Stories

Based on the general requirements given by the client in one of the early meetings, a set of user stories was created to fulfill and validate those same requirements. After further investigation and, considering the meetings during the development, some user stories were adjusted, and others were created to keep the kanban board concise.

In Table 2, there is a list of all the final user stories. These are grouped in sets and do not include a business value since it was not estimated, meaning in each set, there is no specific order or logic path to complete the user stories. All the user stories start with the name of the project: Skill Search (SS).

Table 2 - User Stories

User Stories	Description
SS-1.1	As a user with administrator privileges, I want to be able to carry out Create, Read, Update, and Delete (CRUD) operations on the Employee object so that I can keep my database updated all times.
SS-1.2	As a user with administrator privileges, I want to be able to carry out CRUD operations on the Skill object so that I can keep my database updated all times.
SS-1.3	As a user with administrator privileges, I want to be able to carry out CRUD operations on the Certificate object so that I can keep my database updated all times.
SS-1.4	As a user with administrator privileges, I want to be able to carry out CRUD operations on the Project object so that I can keep my database updated all times.
SS-1.5	As a user with administrator privileges, I want to be able to carry out CRUD operations on the Client object so that I can keep my database updated all times.
SS-1.6	As a user, I want to be able to carry out CRUD operations on the Absence object so that I can keep my database updated all times.
SS-1.7	As a user with administrator privileges, I want to be able to carry out CRUD operations on the Experience object so that I can keep my database updated all times.

User Stories	Description
SS-1.8	As a user with administrator privileges, I want to be able to carry out CRUD operations on the Client object so that I can keep my database updated all times.
SS-1.9	As a user with administrator privileges, I want to be able to carry out CRUD operations on the Program object so that I can keep my database updated all times.
SS-1.10	As a user with administrator or manager privileges, I want to be able to carry out CRUD operations on the Employee Certification object so that I can keep my database updated all times.
SS-1.11	As a user with administrator privileges, I want to be able to carry out CRUD operations on the Certification Skill object so that I can keep my database updated all times.
SS-1.12	As a user with administrator or manager privileges, I want to be able to carry out CRUD operations on the Project Skill object so that I can keep my database updated all times.
SS-2.1	As a user with administrator or manager privileges, I want to have a Skill Search tab so that I can navigate to the Skill Search application by clicking on it
SS-2.2	As a user with administrator or manager privileges, I want to be able to filter employees by skill, certificates or previously business areas that they have worked on so that I can have a results table with the employees ordered by the best match.
SS-2.3	As a user with administrator or manager privileges, I want to be able to give different priorities to the filters I have applied so that I can influence the results retrieved by the search.
SS2.4	As a user with administrator or manager privileges, I want to be able to interact with the results table so that I can navigate to the employees' page layout or send them an email.

User Stories	Description
SS-2.5	As a user with administrator or manager privileges, I want to have the Skill Search application resizable so that I can access it on a mobile device.
SS-3.1	As a user, I want to be able to see a calendar and navigate on it on the Employee layout page so that I can see their absences and allocations more easily.

4.3. Data Model

As explained by Salesforce in their trailheads, using the analogy to a spreadsheet, a Salesforce object is like a table and a field is like a column. So, instead of using a table, an object is used [36]. There are many types of objects, standard and custom objects being the main ones and it does not matter if they are used in a specific framework such as Visualforce or Lightning, they are part of the Salesforce backend. An example of a standard object is the Account, or the Contact, for example. These objects are already defined in a sandbox and have some standard fields. However, it is possible to add custom fields (some standard objects may not be available in all types of sandboxes). On the other hand, custom objects are defined by the user and can represent whatever the user wants. These come with some obligatory fields like Name or Created Date, but besides those, it is possible to create as many custom fields as needed.

At the beginning of the designing process for the data model, the most obvious type of information that could be stored was the collaborators' data. To store their data, there was a need to create an object. However, since the standard object Contact would be so similar to this new one, a decision had to be made upon using the already defined object or not. On one hand, the Contact object already had some of the fields that the new custom object would have. On the other hand, the Contact object would have some unnecessary fields that are not required according to the data that needs to be stored.

The decision was ultimately made based on uncertainty about this project's future updates. Using the standard object would possibly represent confusion in terms of repository management and bigger loading times for the developers. It would also imply the necessity to have Record Type management, as the Employee would now be a Record Type of the

Contact. A Record Type, in summary, is a type of record of an object and it can be used to filter records or organize them as well as manage permissions to records of a specific record type. It can also be used to customize layouts, having a different layout for each record type, for example, or even to decide which values are possible for a certain picklist [37] [38].

Since any other object considered for the data model had any possible usage in a standard object that would make sense, every object in the data model is custom. Custom fields can be of various types and even allow relationships between objects. Some examples of the non-relationship fields are Number, Percent, Text, Text Area, Picklist, Date, Time, Uniform Resource Locator (*URL*), Currency, Checkbox, Email, etc.

Relationship fields are fields that connect objects together. For example, using the Salesforce CRM context, it is easy to imagine a relationship between the Account and Contact objects where each Contact can have an Account and each Account can have multiple Contacts. This way, it is possible to find a related list in the Account layout that lists every Contact "attached" to it and on the Contact layout is possible to find a link that redirects the user to that Contact's Account.

There are some types of fields that allow relationships between objects. These are, mainly, the Master-Detail Relationship and Lookup Relationship. Lookup Relationships can be one-to-one or one-to-many relationships and are typically used when objects are not mandatorily linked. In Master-Detail Relationships one object is the master and the other is the detail. The master object controls the behaviors of the detail object and if a master record gets deleted, all the detail object records are deleted as well. There are more, such as the External Lookup Relationship and the Hierarchical Relationship, where the first allows data integration with data stored outside Salesforce, and the second is only available for the User standard object, being used to create management chains between users. However, they are not going to be used for this project's scope.

4.3.1. Entity Relationship Diagram

The Data Model created for this project is composed of twelve Salesforce objects. The Program, Project, Client, Skill, Certification, Holiday, Employee and Absence objects are considered regular custom objects whereas the Allocation, Experience, Certification Skill, Project Skill and Employee Certification are considered junction objects. These junction objects are always between only two regular objects as seen by the Master-Detail

fields present in each one of the junction objects. It is also important to mention that some picklists are dependent on other picklists, meaning that, using the City and Country picklists on the Employee object as an example, the values in the State picklist depend on the selected value of the Country picklist. In order to have a better understanding of the objects and their relationships between one another, the Entity Relationship Diagram in Figure 10 (displayed larger in Appendix 1) was create, using crow's foot notation [39] as Salesforce uses to illustrate its own data models.

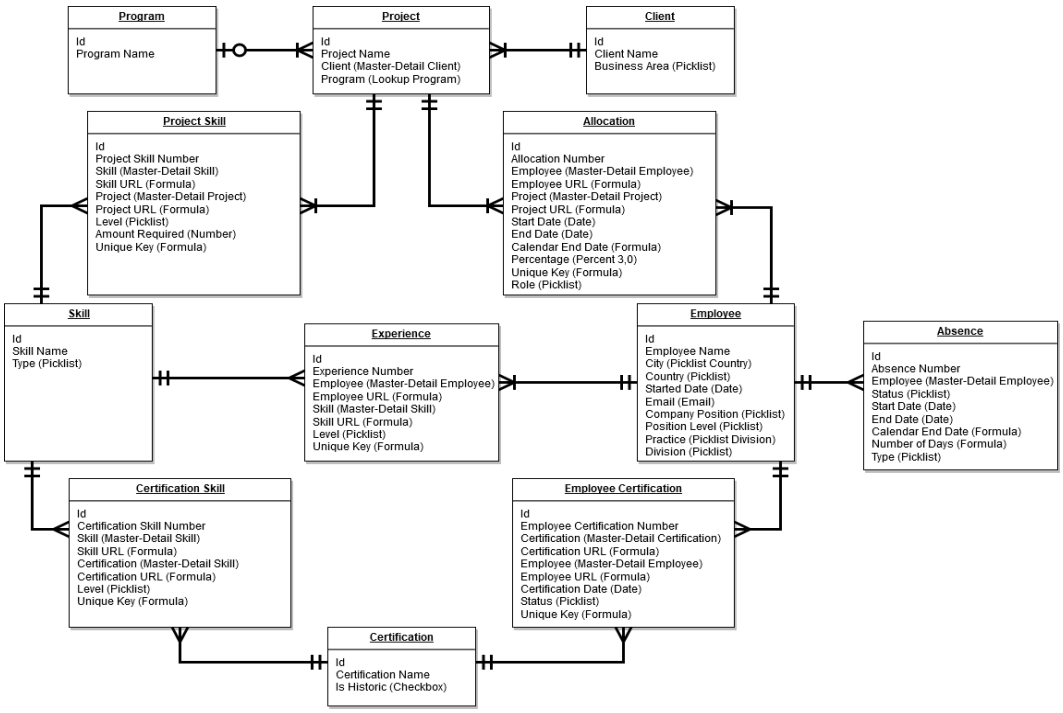


Figure 10 - Skill Search Entity Relationship Diagram

4.3.2. Data Logic Model

The data logic model's purpose is to display the fields in a different way, highlighting the primary and foreign keys by underlining them. Regular underline is for the primary keys (Id fields) while the dotted underline is for the foreign keys (relationships fields).

- Program [Id, Program Name]
- Project [Id, Project Name, Client, Program]
- Client [Id, Client Name, Business Area]
- Project Skill [Id, Project Skill Number, Skill, Skill URL, Project, Project URL, Level, Amount Required, Unique Key]
- Skill [Id, Skill Name, Type, Status]

- Certification Skill [Id, Certification Skill Number, Skill, Skill URL, Certification, Certification URL, Level, Unique Key]
- Certification [Id, Certification Name, Is Historic]
- Employee Certification [Id, Employee Certification Number, Certification, Certification URL, Employee, Employee URL, Certification Date, Status, Unique Key]
- Employee [Id, Employee Name, City, Country, Started Date, Email, Company Position, Position Level, Practice]
- Allocation [Id, Allocation Number, Employee, Employee URL, Project, Project URL, Start Date, End Date, Calendar End Date, Percentage, Role, Unique Key]
- Experience [Id, Experience Number, Employee, Employee URL, Skill, Skill URL, Level, Unique Key]
- Absence [Id, Absence Number, Employee, Status, Start Date, End Date, Calendar End Date, Number of Days, Type]

4.3.3. Data Domain

In order to better explain what every object purpose is and their fields, a table will be displayed for each object containing all the fields, their types, their description, and some observations. Many fields in the tables present in this chapter contain picklist fields with some default values. These values are not final and are just in place for the demo purposes of the application.

The Employee object (described in Table 3) illustrates an employee and is represented by their name, email, division, position in the company and level (internal value inside the company, which dictates whether a collaborator is ready to move from the actual position). Other important data considered are the country, city and the date in which the employee started working for the company. Since this object contains two dependent picklists fields [40], after Table 3, there are two figures (Figure 11 and Figure 12) to better understand the possible dependent values.

Table 3 - Employee Object

Field	Data Type	Description	Observations
Id	Id	The id of the employee.	Unique
Employee Name	Text	The name of the employee.	
City	Picklist (Country)	The city of the employee.	Picklist values in the following Figure 11
Country	Picklist	The country of the employee.	Picklist values in the following Figure 11
Started Date	Date	The date when the employee joined the company.	
Company Position	Picklist	The position of the employee inside the company.	Picklist values: "Trainee", "Junior Consultant", "Consultant", "Senior Consultant", "Lead Consultant", "Managing Consultant", "Senior Manager", "Manager"
Division	Picklist	The division from which the employee belongs	Picklist values in the following Figure 12
Email	Email	The company email of the employee.	Unique
Position Level	Picklist	The level in which the employee finds himself in his current position.	Picklist values: "A", "B", "C"
Practice	Picklist (Division)	The practice where the employee is inserted.	Picklist values in the following Figure 12

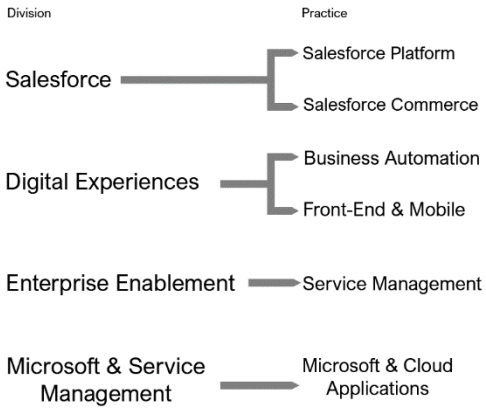


Figure 11 - Division and Practice Dependent Picklist Values Graphic

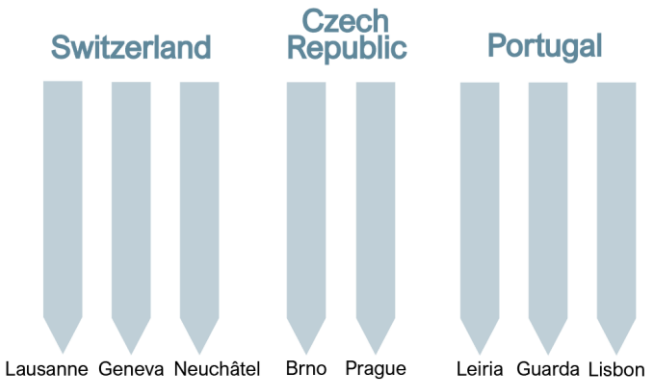


Figure 12 - Country and State Dependent Picklist Values Graphic

The Project object (described in Table 4) illustrates a project that has been won by the company and will be or is being completed by employees. The details of each project include the name, a Master-Detail relationship to the Client object and a Lookup relationship to the Program object. Each project is only owned by one client and can belong to a list with other projects.

Table 4 - Project Object

Field	Data Type	Description	Observations
Id	Id	The id of the project.	Unique
Project Name	Text	The name of the project.	
Client	Master-Detail (Client)	The owner of the project.	One-To-Many Relationship
Program	Lookup (Program)	The program where the project is from.	

The Allocation object (described in Table 5) represents all the allocations to a project or multiple projects in which an employee participates. For every project that an employee is working on, one Allocation record is created linking the two.

Additional information kept on Allocation records include the start and end date of the allocation, the role that the employee is playing in the project, and the percentage of time dedicated to that project during the time frame.

Table 5 - Allocation Object

Field	Data Type	Description	Observations
Id	Id	The id of the allocation.	Unique
Allocation Number	Text	The number of the allocation record.	Generated automatically
Employee	Master-Detail (Employee)	The employee in the allocation.	Many-To-One Relationship
Employee URL	Formula (Text)	A URL to this record but displaying the employee name.	Formula: HYPERLINK('/' & Id, Employee__r.Name)
Project	Master-Detail (Project)	The project in the allocation.	Many-To-One Relationship
Project URL	Formula (Text)	A URL to this record but displaying the project name.	Formula: HYPERLINK('/' & Id, Project__r.Name)
Start Date	Date	The start date of the allocation.	Used in conjunction with the Unique Key to determine uniqueness
End Date	Date	The end date of the allocation.	Used in conjunction with the Unique Key to determine uniqueness
Calendar End Date	Formula (Date)	The end date displayed on the calendar component.	Formula: End Date + 1
Percentage	Percent	The percentage of work hours the employee is allocated to the project. The total value between	

		all allocations of the same employee should add up to 100.	
Unique Key	Formula (Text)	A hidden value to avoid duplicate records between the same project and employee. The same employee cannot be allocated to the same project twice in the same time frame.	Formula: 'Employee + Project'
Role	Picklist (Multi-select)	The role of the employee in the project they are allocated to.	Values: "Developer", "Business Analyst", "Administrator", "Technical Architect", "Delivery Manager", "Scrum Master", "Technical Lead", "Practice Group Manager", "Practice Manager", "Practice Lead", "Customer Success Manager", "Project Manager", "Integration Specialist", "Quality Assurance", "Release Manager", "DevOps", "Solution Architect"

The Skill object (described in Table 6) illustrates a skill that can be mastered by many employees. It is described by its name and type of skill. For this projects' scenario, there are no specific sets of skills that employees can learn. It is considered that any employee can learn or master any skill.

Table 6 - Skill Object

Field	Data Type	Description	Observations
Id	Id	The id of the allocation.	Unique
Skill Name	Text	The name of the skill.	
Type	Picklist	The type of skill	Values: "Soft", "Techincal"

The Experience object (described in Table 7) represents the skills knowledge of each employee as it is the junction between Employee and Skill. An experience record exists for every skill that every employee has mastered, and it is used to check which skills each employee has. It only saves the highest level of knowledge that an employee has in order to avoid "duplicate" records. For example: if an employee has a Low Apex level but gets certificated and, the level, according to that certification, was to be increased, the level field of the experience record gets updated.

Table 7 - Experience Object

Field	Data Type	Description	Observations
Id	Id	The id of the experience.	Unique
Experience Number	Text	The number of the experience record.	Generated automatically
Employee	Master-Detail (Employee)	The employee in the experience.	Many-To-One Relationship
Employee URL	Formula (Text)	A URL to this record but displaying the employee name.	Formula: HYPERLINK('/' & Id, Employee__r.Name)
Skill	Master-Detail (Skill)	The skill in the experience.	Many-To-One Relationship
Skill URL	Formula (Text)	A URL to this record but displaying the skill name.	Formula: HYPERLINK('/' & Id, Skill__r.Name)
Level	Picklist	The highest level obtained by the employee for the given skill.	Values: "Low", "Medium", "High" and "Very High"
Unique Key	Formula (Text)	A hidden value to avoid duplicate records between the same skill and employee.	Formula: 'Employee + Skill'

The Project Skill object (described in Table 8) represents the required skills for an employee to take part in a certain project. When creating a new project record in the application, the user should create as many Project Skill records as many different skills are

required. This will link the project to the different skills required to accomplish it, thus, helping to understand which employees are best suited to do the project.

Table 8 - Project Skill Object

Field	Data Type	Description	Observations
Id	Id	The id of the allocation.	Unique
Project Skill Number	Text	The number of the project skill record.	Generated automatically
Skill	Master-Detail (Skill)	The skill required in the project.	Many-To-One Relationship
Skill URL	Formula (Text)	A URL to this record but displaying the skill name.	Formula: HYPERLINK('/' & Id, Skill__r.Name)
Project	Master-Detail (Project)	The project record linked to the skill record.	Many-To-One Relationship
Project URL	Formula (Text)	A URL to this record but displaying the project name.	Formula: HYPERLINK('/' & Id, Project__r.Name)
Level	Picklist	The minimum level required for the employee to be able to work on the project.	Values: “Low”, “Medium”, “High”, “Very High”
Amount Required	Number	The number of employees required to have a certain skill while working on the project.	
Unique Key	Formula (Text)	A hidden value to avoid duplicate records between the same skill and project.	Formula: ‘Project + Skill’

The Program object (described in Table 9) represents a collection of projects. This is only used when there are multiple projects that are in any way related to each other. This information can help when the manager is rotating employees from one project to another or while researching projects inside the application.

Table 9 - Program Object

Field	Data Type	Description	Observations
Id	Id	The id of the allocation.	Unique
Program Name	Text	The name of the program.	

The Client object (described in Table 10) illustrates a client and is composed of the name and area of business. Every project in the application must have a client. This allows to gather information regarding the number of employees that are working for the same client, as well as to give experience to the employees in the business areas of the client. The manager can also use the client object to rotate employees to different projects while maintaining them working for the same client.

Table 10 - Client Object

Field	Data Type	Description	Observations
Id	Id	The id of the client.	Unique
Client Name	Text	The name of the clients' company.	
Area of Business	Picklist (Multi-select)	The areas of business of the client.	Values: "Consumer Goods", "Finance", "Information Technology", "Luxury", "Life Science", "NGO", "Telecommunication", "Transportation"

The Certification object (described in Table 11) illustrates a certification that an employee can take. Sometimes, certain certifications cease to exist and are replaced by new ones with updated information. The checkbox named "Is Historic" can help the user to continue tracking old certifications in order to maintain the data up to date.

Table 11 - Certification Object

Field	Data Type	Description	Observations
Id	Id	The id of the client.	Unique
Certification Name	Text	The name of the certification.	
Is Historic	Checkbox	In the scenario of an old certification that cannot be taken anymore, this checkbox should be marked as true.	

The Employee Certification object (described in Table 12) represents a certification that an employee has. The purpose of this object is to track the certifications each employee has, and, for example, to know how many they took each year. This can also be helpful in cases where the client requires an employee to have a specific set of certifications in order to work on one of their projects. Combined with the Certification Skill object that will be explained further, it is possible to determine every skill that an employee has by the certifications he/she has.

Table 12 - Employee Certification Object

Field	Data Type	Description	Observations
Id	Id	The id of the client.	Unique
Employee Certification Number	Text	The number of the employee certification record.	Generated automatically
Certification	Master-Detail (Certification)	The certification that the employee has.	Many-To-One Relationship
Certification URL	Formula (Text)	A URL to this record but displaying the certification name.	Formula: HYPERLINK('/' & Id, Certification__r.Name)
Employee	Master-Detail (Employee)	The employee that took a specific certification.	Many-To-One Relationship

Field	Data Type	Description	Observations
Employee URL	Formula (Text)	A URL to this record but displaying the employee name.	Formula: HYPERLINK('/' & Id, Employee__r.Name)
Certification Date	Date	The date in which the employee took the certification	
Status	Checkbox	The status of the certification for the employee.	“Active”, “Expired”
Unique Key	Formula (Text)	A hidden value to avoid duplicate records between the same employee and certification.	Formula: ‘Employee + Certification’

The Certification Skill object (described in Table 13) represents all the skills granted to a collaborator that took a certain certification. Each record is a combination of one skill record and one certification record. It also contains other information like the skill level granted. It is used as soon as an employee gets a certification in the application to create or update the employees’ experience records.

When a certification is added to an employee, all the certification skill records containing that certification are listed. Then, all the skills in that list will be used to create or update experience records, linked to the employee. This will be explained in detail in chapter 5.

Table 13 - Certification Skill Object

Field	Data Type	Description	Observations
Id	Id	The id of the certification skill.	Unique
Certification Skill Number	Text	The number of the certification skill record.	Generated automatically
Certification	Master-Detail (Certification)	The certification that grants the skill.	Many-To-One Relationship
Certification URL	Formula (Text)	A URL to this record but displaying the certification name.	Formula: HYPERLINK('/' & Id, Certification__r.Name)

Field	Data Type	Description	Observations
Skill	Master-Detail (Skill)	The skill that is granted by the certification	Many-To-One Relationship
Skill URL	Formula (Text)	A URL to this record but displaying the skill name.	Formula: HYPERLINK('/' & Id, Skill__r.Name)
Level	Picklist	The level of the skill granted by the certification.	Values: “Low”, “Medium”, “High”, “Very High”
Unique Key	Formula (Text)	A hidden value to avoid duplicate records between the same skill and certification.	Formula: ‘Employee + Skill’

The Absence object (described in Table 14) represents all kinds of absences that an employee can take. This can be a holiday, maternity leave or other type of absence. Absence records help the manager to have a clear view of when a collaborator is not available.

Table 14 - Absence Object

Field	Data Type	Description	Observations
Id	Id	The id of the project skill.	Unique
Absence Number	Text	The number of the absence.	Generated automatically
Employee	Master-Detail (Employee)	The employee that the absence concerns about	Many-To-One Relationship
Status	Picklist	The status of the absence.	Values: “Pending Approval”, “Approved”, “In Progress”, “Completed”, “Declined”
Start Date	Date	The start date of the absence.	
End Date	Date	The end date of the absence.	

Field	Data Type	Description	Observations
Calendar End Date	Formula (Date)	The end date displayed on the calendar component.	Formula: End Date +1
Number of Days	Formula (Number)	The total number of days in which the employee is or will be absent, including weekends.	Formula: End Date – Start Date +1
Type	Picklist	The type of absence.	Values: “Accident”, “Birth”, “Civil Service”, “Compensation”, “Holidays”, “Maternity Leave”, “Move”, “Personal Loss”, “Sickness”, “Unpaid Leave”, “Wedding”

5. Project Development

When all the architectural decisions are made and all the methodology processes are implemented and ready to be used, the next logical step to follow is the actual development of the project.

In this chapter, the entire process of development of the application is explained. This will include the creation of the data model inside the newly created Salesforce sandbox and a detailed explanation of the multiple functionalities that were implemented as well as the software used to accomplish the project development.

5.1. Version Control Software

As a rule of thumb, every project should be kept safe in a repository of some sort. This adds an extra layer of safety in case the code is lost (when dealing with offline programming), helps to maintain the entire development team coordinated and with the correct version of the code where each member must develop. Many other benefits include keeping track of the code history in order to have a better understanding of what is being changed and why.

Even though Salesforce development is online, and the project was developed solely by one person, in order to make use of all the benefits previously mentioned, the code was regularly saved into a repository. In order to do so, the software used were: Git [41] and SourceTree [42].

Git is version control software that is used to create the connection between the computer and an existing repository and allows the user to have many operations that manage the files in the repository. SourceTree is a simple yet effective software that adds a visual layer on top of the Git software. It was used to simplify the tasks of repository management. The repository used to store the application code was hosted in BitBucket [43] and is owned by the company.

5.2. Development Software

The main software for this project was the Salesforce Platform as it was a requirement, but besides Salesforce itself (which will be discussed next in chapter 5.2.1) there were some other tools and applications used to develop the project.

MavensMate [44] was the software used to retrieve all the possible metadata files and Apex files from the Salesforce platform. In order to do so, the user must connect the development sandbox with MavensMate via a Salesforce login, and then, it is simply a matter of selecting which files to retrieve and the folder to save them. By using the same folder of destination on the MavensMate and Git, any changes to the code or to the metadata were easily tracked in the SourceTree. This software also allowed to create classes, triggers, VisualForce pages, and components, or even Aura Components and Applications directly inside the editors. Since MavensMate acts as a plugin for both Sublime Text or VisualStudio Code, it allows to open the retrieved Salesforce projects in any of these editors and program from there.

At the start of the application development, as explained later in chapter 5.7, a decision was made to develop using Lightning Web Components. This had a great impact on the project since MavensMate (which, unfortunately, was discontinued during the development) was no longer an option to develop because Lightning Web Components require SalesforceDX [45] to be created or edited.

In Salesforce it is always possible to develop directly in the sandbox using tools like the Developer Console or even the Apex Class editor. However, to develop Lightning Web Components, Salesforce DX is required as it is the only tool capable of doing so at this moment. SalesforceDX is a plugin with many of the functionalities of the MavensMate plugin and some others more, but it uses a different folder structure when retrieving the sandbox files, thus, requiring updating the folder structure of the project in the repository when the change was made. Also, instead of providing its capabilities to both editors mentioned above, it only supports the VisualStudio Code editor [46].

5.2.1. Salesforce

Salesforce is a company that specializes in Customer Relationship Manager (CRM) and was the first to do so using cloud computing and Software as a Service (SaaS) technologies combined.

A CRM System is a software used to keep track of information about business customers. By tracking information like emails, phone calls, meetings, and so forth, a CRM can provide valuable information to create, follow and close a business opportunity. The goal of any CRM is to help the business grow.

According to Salesforce, SaaS is a way of delivering centrally hosted applications over the Internet. There is no need to install these applications as they are web-based (hosted in the cloud) and most of the time these can be accessed via a web browser. This way there is no hassle with hardware management as the provider is the one that has to worry about providing access to the application, security, performance, and availability. The main advantages are the lower initial costs (depending on the case, obviously), no need for upgrades by the client and seamless integration with many other software services. With SaaS, Salesforce uses the Multitenant paradigm in which all the users share a common infrastructure, but each has a different type of access to it. Much like a building with different apartments, different companies can be using the same hardware infrastructure to use different software instances, yet the isolation from one another is virtual. In terms of software, one of the main benefits of using a multitenancy architecture is that with the feedback of all the different users, it is possible to upgrade incrementally everyone's system as problems start to appear [47].

This combination of technologies allowed Salesforce to take the lead in the CRM market because their software would not need installation or infrastructure maintenance by the clients on their side. Being a SaaS allows for faster development of applications without the need to have their own infrastructure, thus, allowing clients to save some costs. On top of this, by being fully cloud-based, the only requirement to access every Salesforce application or data is an internet connection and a browser.

Salesforce has many integrated products like Sales, Service, Marketing, Commerce, Engagement, Platform, Integration, Analytics, Industries, Communities, Enablement, and Collaboration. Salesforce claims that, on average, customers using the CRM and other Salesforce products see a 25% increase in revenue and are 38% faster while decision making [48].

The product used to develop the application was the Salesforce Platform. It has its own proprietary coding language, which is Apex, but allows the creation of applications in any coding language and has some proprietary frameworks like Lightning Component.

Apex is a strongly typed, object-oriented procedural programming language and it is very similar to the Java language. With Apex, it is possible to build scripts that execute on-demand or triggers that are fired before or after a database manipulation happens. When an Apex routine is executed for the first time, the compiled version is saved in a form of cache to be used in the next usages of the same routine, allowing memory savings.

Lightning Component framework is a UI framework used to develop Single Page Applications (SPAs) for mobile and desktop devices. Lightning Components (also known as Aura Components), makes use of Apex on server-side logic, JavaScript on client-side logic and Salesforce Lightning Design System (SLDS) to provide a style to the custom stand-alone Lightning applications that is consistent with Salesforce. As it is customizable, it is also possible to use custom CSS to style the applications in order to make them look as needed.

Between the end of 2018 and the beginning of 2019, Salesforce launched a new framework denominated Lightning Web Components [49]. Although Lightning Web Components were created to replace Aura Components, both can coexist on the same pages because they share the same base and underlying services. In other words, Aura Components can include Lightning Web Components. Also, it will still be possible to continue developing Aura Components. In Figure 13 there is an image explaining the implementation of the Lightning Web Components framework [50].

Figure 13 - Lightning Web Components Architecture

5.3. Sandbox Creation

The first step taken in order to start implementing this project was the actual creation of the Salesforce sandbox where the development would take place. In order to achieve this, there was the need to create a Developer Edition account [51]. Only with such an account, it is possible to create a Developer organization. There are three types of organizations: Production, Developer, and Sandbox. A production org contains users accessing the data and applications and the sandbox org is a copy of production where all the developments are made, since it is not possible to develop Apex code in any production org. The Developer organization purpose is to create packages that are put on the AppExchange [52] store or simply distributed online.

After a few forms were filled in and validated, the sandbox was created. To reach this form the following website was used: https://developer.salesforce.com/signup. After having access to the sandbox and configuring secure access to it, with a mobile number, the first step was to replicate the data model mentioned in chapter 4.3, inside the sandbox.

5.4. Data Model Implementation

Following the data model table, previously showed in Figure 10, the first step was to create all the objects and fields in the sandbox. This task was done by navigating to the Object Manager page inside the Salesforce Setup page.

To create an object, the following steps are needed:

- Inside the Object Manager Menu, click on Create and chose Custom Object.
- Populate mandatory fields like Label, plural Label, Object Name (API Name), Record Name and Data Type (Text or Auto-Number).
- In this step, it is also possible to choose if the records of this object are searchable, shared and accessible via Bulk API and Streaming API.
- Click the “Save” button at the bottom of the form.

And then to create each field inside an object, the following steps are needed:

- Select an object inside the Object Manager Setup.
- Select the Fields & Relationships menu on the left.
- Click on the “New” button.

- Select the Data Type of the field.
- Choose the Label and Field Name of the field. Then, depending on the previous selection, other options will vary.
- Select which profiles have visibility in the field. “Visible” and “Read-Only” are the option for each profile on this menu.
- Select in which layouts of the object the field appears in. (Every object can have multiple layouts to display the data of its records. It is possible to select a different layout for each record type of the object, for example.
- Click the “Save” button.

After all the objects and fields are correctly configured it is time to create all validation rules that the object needs. Validation rules are used to prevent records with bad data to be created or updated. They can be replaced with triggers, even though, they are created without any Apex code, just the validation formula.

In the Salesforce order of execution, the validation rules are fired after the “before triggers”, allowing “before triggers” the chance to apply a correction to the records before they are validated. If the validation fails, the entire transaction is rolled back and the specified error will be prompted to the user, stopping the record from being created. Table 15 contains all the validation rules that were implemented for each object:

Table 15 - Skill Search Validation Rules

Object	Validation Rule Name	Error Condition Formula	Error Message
Absence	Valid_Date_Period	Start_Date__c > End_Date__c	End Date cannot be before the Start Date
Allocation	Valid_Date_Period	Start_Date__c > End_Date__c	End Date cannot be before the Start Date

One of the next steps before moving to the triggers is to configure all the duplicate rules that will prevent records from being duplicated. It is possible to mark some types of fields as unique. Thus, not allowing two records of the same object to have the same value in that same field. However, the error message that is presented to the user on the process of trying to create a duplicate record is not very clear or even dynamic. With the duplicate rules,

even before the user clicks on the button to save record that would be duplicated, an error message will show up giving a hyperlink to the record that already exists and matches the criteria. Every duplicate rule is followed by a matching rule that is used to create the criteria that will match the record that is being created or updated and all the other existing records of the same type.

Table 16 contains all the duplicate rules that were created to avoid duplicated records in the application.

Table 16 - Skill Search Matching Rules

Object	Matching Rule Criteria	Considerations
Certification	Certification: Name EXACT	Block on create and update
Client	Client: Name EXACT	
Program	Program: Name EXACT	
Project	(Project: Client EXACT) AND (Project: Name EXACT)	
Skill	Skill: Name EXACT	

Therefore, if the user tries, for example, to create a Client with the same name as an existing one, even before the click to confirm the operation, a message will be prompted with a hyperlink to the existing client with the same name. Additionally, if the user clicks the save button, nothing will happen.

Duplicate rules would have been great to avoid duplicate records of the junction objects as well, however, matching rules do not allow two master-detail relationship fields as matching criteria. Since it is not possible to use them for this purpose, a decision was to be made between two options. Either create a text formula field that is unique, so that an error would show up when creating a new record of that junction object or, create that same text formula field but instead of making it unique, use triggers to validate that new records are not duplicated. The second choice seems to be more complex, nonetheless, only with that option displays an error message that is not confusing for the user.

For each object, the layout to display the records was modified to make it more readable for the user. These modifications included removing standard buttons like shortcuts to create contacts or cases that salesforce includes by default in every layout, removing standard related lists and improving the required ones. This improvement is where the fields that end in “URL” come into play. By default, each related list only contains a link to the related record, but it uses the Name field as the link, and this, for the objects that have an auto-number as a name, can be very confusing to the user. So, the solution was to create a formula field that would use a more logical description, maintaining the same functionality. The record layout can be seen in Figure 14.

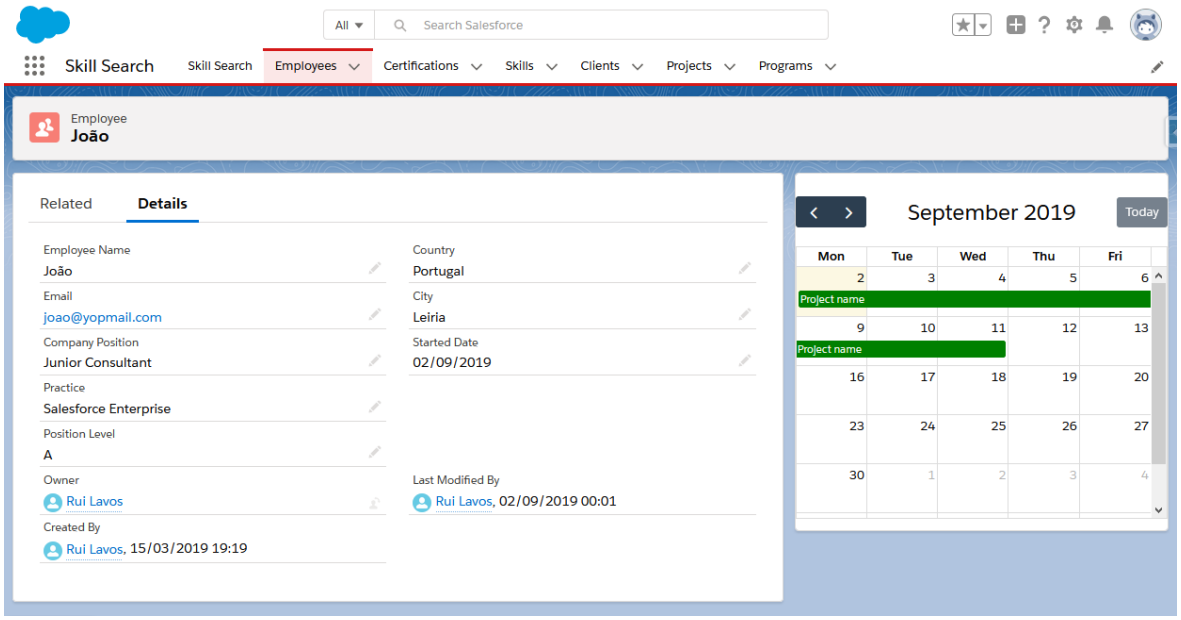


Figure 14 - Employee Layout (left side)

It is important to note that if a user has no visibility, in terms of permission, of a certain field of an object or even to an object itself, the field or tab, respectively, are hidden. This may cause slight variations in the layout according to the profile of the user that is logged in. However, it promotes security in the sandbox since the data that is supposed to be kept private for some users will never be displayed to them if they have the correct profile for themselves.

5.5. Trigger Pattern

For every object in Salesforce, there is the possibility to have multiple triggers. However, best practices dictate that there should be only one trigger per object. Salesforce explains this since with multiple triggers, there is no guarantee of a specific order of

execution between all the triggers. A trigger will execute its actions upon records and can be fired at different events. With these, it is possible to execute custom Apex code according to a specific situation. The different events are:

- Before insert
- Before update
- Before delete
- After insert
- After update
- After delete
- After undelete

"Before" events are usually used to update or validate record values before the final commit into the database while after triggers are usually used to access fields that are set by the system and to affect changes in other records.

Before the creation of the triggers for the objects, the trigger pattern class was first implemented. The trigger pattern is composed of one single class denominated *TriggerHandler* in this project.

The goal of this Apex class is to have better control over all the existing triggers for all objects, as these triggers are prone to escalate in terms of the number of functionalities and, therefore, it is important to keep having control over the logic being executed in all scenarios. With this class, it becomes easier to manipulate which events are being fired when running the triggers. It is important to note that as a best practice, the triggers should have the least amount of Apex code possible since they cannot be tested, and it is easier to delegate certain actions that might apply to multiple objects and to multiple events using generic methods outside triggers. Therefore, for each existing trigger, an Apex class containing the helper methods is created, as an extension from the *TriggerHandler* class.

In the next two figures, there is an example of how the trigger (Figure 15) and *TriggerHandler* class (Figure 16) for the Allocation object are defined making use of the Trigger Pattern.

```
1  trigger Allocation_Trigger on Allocation__c (before insert, before update) {
2      new Allocation_TriggerHandler().run();
```

Figure 15 - Allocation Trigger

```
1  public class Allocation_TriggerHandler extends TriggerHandler {
2      private List<Allocation__c> triggerNew = (List<Allocation__c>) Trigger.new;
3      private List<Allocation__c> triggerOld = (List<Allocation__c>) Trigger.old;
4      private Map<Id, Allocation__c> newMap = (Map<Id, Allocation__c>) Trigger.newMap;
5      private Map<Id, Allocation__c> oldMap = (Map<Id, Allocation__c>) Trigger.oldMap;
6
7      public override void beforeInsert() {
8          Allocation_Service.getInstance().blockDuplicates(triggerNew);
9      }
10
11     public override void beforeUpdate() {
12         List<Allocation__c> updatedAllocs = new List<Allocation__c>();
13
14         for (Allocation__c alloc: triggerNew) {
15             if (alloc.Start_Date__c != oldMap.get(alloc.Id).Start_Date__c
16                 || alloc.End_Date__c != oldMap.get(alloc.Id).End_Date__c) {
17                 updatedAllocs.add(alloc);
18             }
19         }
20
21         if (!updatedAllocs.isEmpty()) {
22             Allocation_Service.getInstance().blockDuplicates(updatedAllocs);
23         }
24     }
25 }
```

Figure 16 - Allocation trigger Handler class

As shown, the trigger is a very short file with just one line of code and the trigger handler contains all the logic separated in order to have that same logic tested and reused by other handlers. Also, in the *Allocation_TriggerHandler* figure, line 22 shows an example of a service class being used.

5.6. SOC Pattern

Following the explanation given in the previous chapter 4.1, for each object existing that required a Service, Selector or Domain class, those classes were created. Contrary to what was generally explained in the previous chapter, there is no strict need to create all those three classes per object if only one of them is required to exist. And so, if any query on a determined object was needed, a Selector class would be created for that object, and if any DML operation for a given object was needed, a Domain class would also be created. The same is applied for the Service class when there are logic operations associated to an object.

In order to have a better performance, all the Service classes use the Singleton Design Pattern [53]. With this pattern an instance of the class is created the first time it is called, and then, for the rest of the current operation or transaction, that same instance is used, avoiding to execute the class constructor more than one time, mitigating the impact of governor limits.

The Service classes is where all the logic operations regarding a specific object lies. Those operations can be used either by the TriggerHandler, by page controllers or even by flows. Since some objects require similar logic, a *SkillSearch_Service* class was built with the generic logic that is required by multiple objects. An example of this can be seen in Figure 16 as the function *blockDuplicates* (line 12) is using the method with the same name in the parent class. Every Service class in the project extends from the *SkillSearch_Service* class.

The Selector classes are where all the queries related to a specific object should be. This can help to maintain the code clean, avoiding code repetition throughout the different classes in the sandbox. In order to make better use of the Domain classes, all the queries are first validated with two methods: *isAccessible* and *isQueryable*. With both these methods, a layer of security is added not allowing a user without permissions for the given object to retrieve information about its records. Otherwise, there could be queries impacting governor limits that were retrieving no results if the user had no permissions on that object.

Lastly, the Domain classes, as previously stated, is where all the DML operations for a given object are stored. Similarly, to the Selector classes, there is a layer of protection in each of the different types of operations. Methods like *isCreatable*, *isUpdateable* and isDeletable allow a security check before executing the actual operation that could impact the governor limits.

In the following figures, there is an example of the Certification Skill object Service class (Figure 17), an example of the Selector class (Figure 18) for the same object and an example of the Domain class for the Employee Certification object (Figure 19).

```
1  public class CertificationSkill_Service extends SkillSearch_Service {
2      private static CertificationSkill_Service instance = null;
3
4      public static CertificationSkill_Service getInstance() {
5          if (instance == null) {
6              instance = new CertificationSkill_Service();
7          }
8
9          return instance;
10     }
11
12     public void blockDuplicates(List<Certification_Skill__c> certSkillsToCheckToCheck) {
13         super.blockDuplicates(certSkillsToCheckToCheck, Certification_Skill__c.Unique_Key__c, 'Certification_Skill__c');
14     }
15 }
```

Figure 17 - Service Class Example

```apex
public class CertificationSkill_Selector {
    public static List<Certification_Skill__c> getCetificationSkillsByCertifications(Set<Id> certificationIds){
        try {
            SObjectType sObjectType = Schema.getGlobalDescribe().get('Certification_Skill__c');

            return sObjectType.getDescribe().isAccessible() && sObjectType.getDescribe().isQueryable() ?
                [SELECT Id, Skill__c, Level__c, Certification__c
                FROM Certification_Skill__c
                WHERE Certification__c IN :certificationIds] : null;
        } catch (Exception e) { throw new CertificationSkill_SelectorException(e); }
    }

    public class CertificationSkill_SelectorException extends Exception {}
}
```

Figure 18 - Selector Class Example

```apex
public class EmployeeCertification_Domain {
    public static final String EMPLOYEECERTIFICATION_OBJECT = 'Employee_Certification__c';

    public static final String MSG_ISNT_GENERIC = 'You don\'t have permissions to {0}. Contact your Admin for more information.';
    public static final String MSG_ISNT_CREATEABLE = String.format(MSG_ISNT_GENERIC, new List<String>{'Insert Employee Certifications'});
    public static final String MSG_ISNT_UPDATEABLE = String.format(MSG_ISNT_GENERIC, new List<String>{'Update Employee Certifications'});
    public static final String MSG_ISNT_DELETABLE = String.format(MSG_ISNT_GENERIC, new List<String>{'Delete Employee Certifications'});

    public static void insertEmployeeCertifications(Employee_Certification__c record) {
        insertEmployeeCertifications(new List<Employee_Certification__c> {record});
    }

    public static void insertEmployeeCertifications(List<Employee_Certification__c> records) {
        if (Schema.SObjectType.Employee_Certification__c.isCreateable()) {
            try {
                insert records;
            } catch(DMLException e) { throw new DMLException(e.getMessage(), e); }
        } else {
            throw new EmployeeCertification_DomainException(MSG_ISNT_CREATEABLE);
        }
    }

}
```

Figure 19 - Domain Class Example

5.7.Skill Search Page

Before starting to develop the actual application containing all the user stories from SS-2.1 to SS-2.4, there was some indecision on which framework to use in its development. On one hand, there was the possibility to develop it using Aura Components which are well documented, and well-known across the Salesforce community of developers, but were not the latest trend of developing and would not bring anything new into the company. On the other hand, there are the Lightning Web Components that, at the time being, are not very well documented, so the Salesforce community is still learning the best ways and practices of how to use them and this could become an issue. And so, in order to make the project more challenging and to use it as a source of knowledge in the new framework, the Lightning Web Components was the chosen framework.

Each lightning web component is composed of a ".html" file, a ".css" file and a JavaScript file. Inside the JavaScript file is where the logic of the page is and, the bridge between the front-end and back-end is done. The Lightning Web Components HTML

templating system uses the virtual Document Object Model (DOM) to render components smartly and effectively [54].

For the main application, the .html file was built using several elements that were made available by Salesforce in their Component Library page and it was made to be responsive in order to be usable in the Salesforce1 application for mobile devices [55].

In Figure 20, a screenshot of the application divided into colored sections can be seen, in order to make it easy to explain in the following paragraphs.

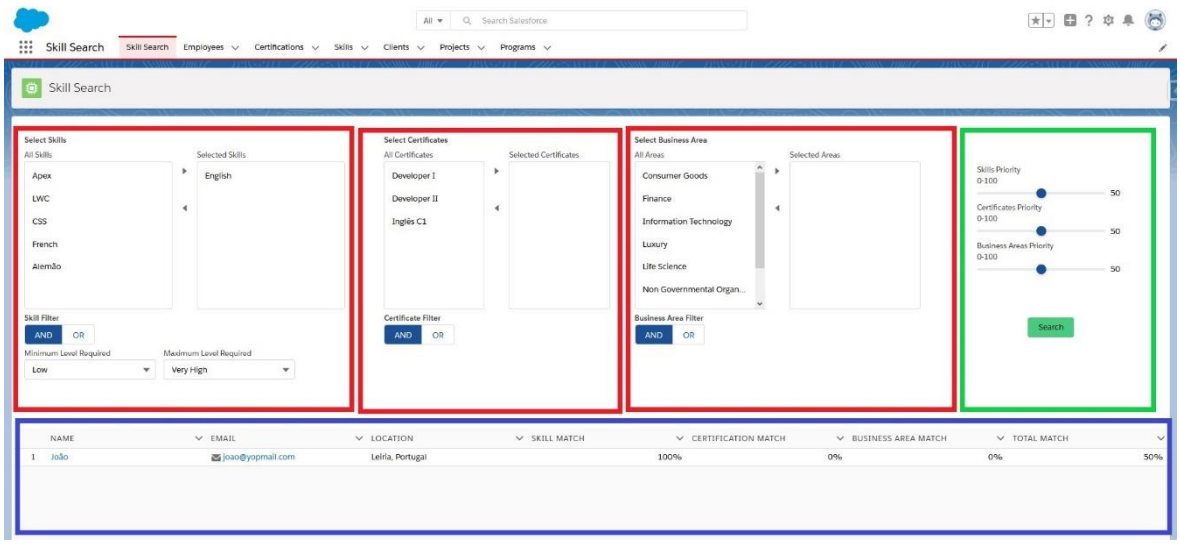


Figure 20 - Skill Search Application layout

The first elements to be built were the “Filter by Topic Elements” (red-colored in the figure) that are composed of a *lightning-dual-listbox* [56] and a lightning-radio-group each [57]. Except for the Skill Filter in which two *lightning-combobox* [58] were also added to select the minimum and maximum level of the Skills to be added into the search.

In order to select one of the listed items, the user must select it by clicking on it and then choose the right-pointing arrow to move it to the right column. This can be done to multiple items at the same using the “Ctrl” button on the computer keyboard. The user will then select whether the employee must have all those selected items, or if the employee can have only a subset of them with the “AND / OR” radio button. It is important to note that the radio button filter will apply that condition to the selected items if there is more than one for that topic selected. This will influence the match score given to a certain employee because the “AND” value implies that the employee must have both skills, for example, and if they only have one, the score will be lower, logically.

The second element to be made was the “Priority and Search Element” (green-colored in the figure). This element was built using three *lightning-slider* [59] components and one *lightning-button* [60] for the search button. The sliders allow the user to decide which of the parameters is the most important for them when doing a search based on multiple filters. Basically, it will influence the order of the results or even the actual employees that are returned, depending on the existing data of the database and on the selected options of the filters. The search button will only be enabled if there is at least one filter with an option selected. If nothing is selected, the user will not be able to do a search since there are no criteria for that search.

The third element that was built was the “Results Table Element” (blue-colored in the figure) that is composed by a *lightning-datatable* [61] that is only displayed when the user executes a search on the application and contains the following columns:

- Name
- Email
- Location
- Skill Match Percentage
- Certificate Match Percentage
- Business Area Match Percentage
- Total Match Percentage

The first two columns of the Results Table are interactive in the sense that, if the user clicks on the name of the employee, a new tab on the browser will be opened, displaying the employee record. If the user clicks on the employee’s email, a pop-up will be displayed prompting the user to send that employee an email.

Lightning Web Components applications need an Apex controller if there is the necessity to fetch data or use any defined method from an existing Apex class. Since this application needed to find the employees that are best suited to the parameters defined by the user, one Apex class was created as the controller class for the application. In order to expose the methods that are used in the controller to the Lightning Web Components, each method must have the annotation @AuraEnabled as seen in Figure 21.

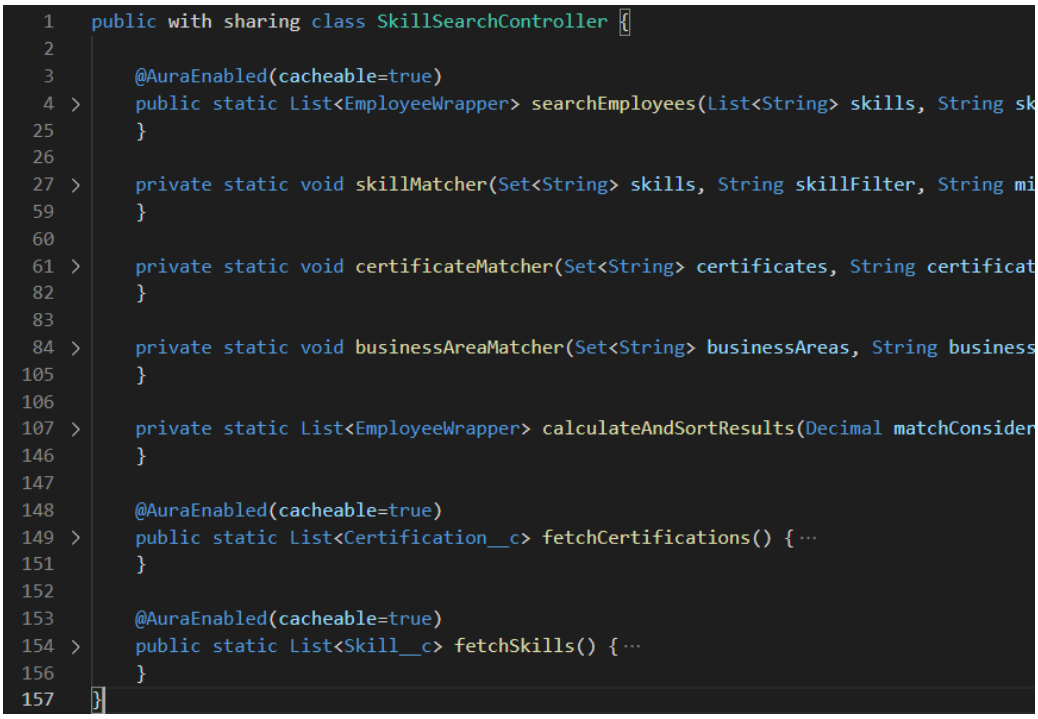


Figure 21 - SkillSearchController class

Another noticeable feature in this controller is the usage of "(cacheable=true)". This annotation means that once the data is loaded the first time into the application, it is saved into cache memory, making the application faster by avoiding multiple calls to the server. A good example of this is the search functionality. If the user searches for all the employees that know Apex, for example, the results are saved into the cache, and so, if the user decides to repeat this action, there will be no call to the Apex Controller once again.

This application was transformed to be responsive, making it usable in tablets and mobile phones with screen sizes above 6.4 inches using responsive CSS features like flexbox [62]. Then, while testing the application in different devices, it was found that the lightning-slider components do not work either in the Salesforce Application for iOS [63] or in the Safari browser if accessed on any iPhone device. However, the rest of the application is working as intended and it should be a matter of time until Salesforce fixes this.

To wrap this section up, a brief explanation of all the methods used:

- *fetchCertifications*(): populates the available certifications in the certifications filter topic. Makes use of the Certification Selector class.
- *fetchSkills*(): populates the available skills in the skill filter topic. Makes use of the Skill Selector class.

- *searchEmployees*(): action called when the user clicks on the search button. Runs the *skillMatcher*(), *certificationMatcher*(), and *businessAreaMatcher*() methods and then the *calculateAndSortResults*(). Then, returns the ordered list of employees.
- skillMatcher(): method called if the user selected any skills in the skill filter option. Creates a map of the experiences of each of the employees and then calculates the match percentage in terms of the skills found for each of them. The percentage is based on the fact of the number of skills that are a match and that the employee has, as well as their level. If the user selects "OR" on the certificate filter radio button, the number of skills has less impact on the matching percentage.
- *certificationMatcher*(): method called if the user selected any certifications on the certification filter option. Creates a map of employee certifications by employee and then calculates the match percentage in terms of the certifications found for each of them. The same calculations apply to this scenario as the skill scenario, excluding the selected level range impact on the percentage.
- *businessAreaMatcher*(): method called if the user selected any skills on the skill filter option. Creates a map of previous and current business areas by employee and then calculates the match percentage in terms of the business areas found for each of them. The same calculations apply to this scenario as the skill scenario, excluding the selected level range impact on the percentage.
- *calculateAndSortResults*(): method called that makes use of the sliders on the application to determine the final order of the employees and the ones to be returned. Makes use of a wrapper class named *EmployeeWrapper* to do the sorting operation.

The *EmployeeWrapper* is a wrapper class containing information about the employee and the order in which the employee will be displayed on the results table. It implements the Comparable class to do so and in order to have the values displayed on the table of the application, all its attributes have the @AuraEnabled annotation.

5.8. Calendar Component

The calendar component was first intended to be a separate application, like the first one in terms of filters, but with the difference of instead of displaying an ordered list with

the results, it displays a calendar with the allocations and absences of the selected employees. However, instead of creating a separate page, the page layout for the employee record was used, displaying the calendar directly next to the employee's data.

And so, the first step, after creating the lightning web component, was to download the FullCalendar [64], a JavaScript API that allows the creation of full-size calendars and events. Even though this API was selected because it is used by some collaborators of the company, there was the precaution of knowing what the API license allowed. FullCalendar has an MIT License that allows for Commercial user, Modification, Distribution, and Private use. Then, after the download, a static resource with the package was uploaded into the sandbox.

Static resources allow files of any kind to be uploaded and saved inside the sandbox. Some common usage examples include images to be used in application's pages or communities, data files that can be used in test classes and packages with JavaScript APIs or libraries to be imported into applications or pages [65].

The component HTML template is simply composed of a regular container where the calendar is injected with JavaScript on the controller. The controller makes use of the *connectedCallback* method to load everything and subscribe to the platform events that are sent through the Allocation and Absence triggers. This way, every time that an Allocation or Absence is created, updated or deleted, the trigger sends a platform event that is interpreted by the callback function of the subscription. Then, depending on the operation, the calendar updates itself if it needs to do so. Platform events are used to deliver notifications within Salesforce or external sources [66].

To get all the allocations and absences of the current employee when the component is loaded and to insert them into the calendar, an Apex controller was created similarly to the previous one:

- fetchAllocationsAndAbsences(): creates a list of CalendarEvent_Wrapper with all the allocations and absences of the employee and returns them to the calendar as events.

The CalendarEvent_Wrapper class is a wrapper class containing all the required information by the FullCalendar API that is present in both the Allocation and Absence objects. The allocations on the calendar will show up with the project name and the absences

will show the phrasing “Absence” plus the type of absence. Besides this, the wrapper class only randomly selects the color of the Allocation and sets all the Absences to orange. As mentioned before, all the attributes are marked with the @AuraEnabled so that the information is presented in the component.

This component does not render on mobile devices due to its size, as it would become imperceptible. Due to Firefox cache problems with Lightning Web Components, it is recommended to use this application in the Google Chrome browser.

5.9. Reports

Salesforce reports [67] help the user have a better understanding of the data that exists allowing, for example, easier decisions to be taken. In this project, there are some reports. An example of a report included in the application is a simple one where it is possible to count how many employees have a certain skill, then group the skills and have a donut chart showing the result visually. The user can run the report at any time to visualize the current data. A run of this report can be seen in Figure 22.

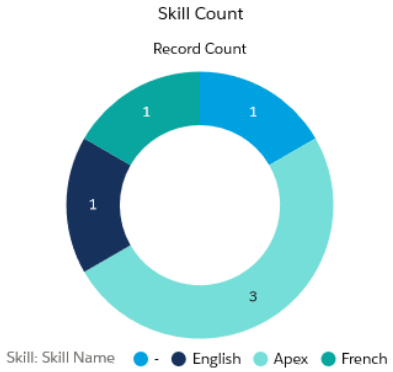


Figure 22 - Skill Count Report Example

In Salesforce it is also possible to subscribe to reports. If a user wants to get notified by a condition on a group of data, it can usually create a report and subscribe to it. This allows the user to receive an email weekly, daily or monthly, for example, alerting for an irregularity or simply giving some information. And so, the second report created for this project is a little bit less visual, since there is no grouping of data, but helps the user that subscribes to it to have the list of employees that will have end an allocation to a project in less than two weeks.

By running the report manually, it is possible to see a list of all the allocations that will be over in the next two weeks (with the possibility of having the list empty), but by subscribing it and defining that the email is only sent if the report contains any data, the user will only be notified in this scenario and does not have to concern with this more often.

5.10. Other Functionalities

Besides the creation of the data model with its validations and layouts, the skill search application and the calendar component, there were some other features that were implemented to improve the usability of the application.

One of the features is the automatic creation of Experience records. This feature was implemented to help the user when adding new Employee Certificates records. As seen in the data model explanation, these records link employees to certificates and represents all the certifications that employees already have. Since each certification is also linked to one or many skills, it is only logical that as soon as an Employee Certificate record is added to an employee record, that same employee should automatically have all the skills provided by that certificate granted. If the employee already possesses any of those skills, the experience will be updated in terms of level if that is the case (only updating to higher values). This was implemented simply using the triggers that were already created.

The second feature is more of a helping tool. It is external to Salesforce and was created in order to help develop both the Skill Search Application and Calendar Component. Since each Lightning Web Component does not allow to have any different files from the ones specified later, it became difficult to use, for example, a ".scss" file that would compile into a ".css" file. Sass [68] is a CSS extension that allows the developer to have an easier experience while styling any page. However, a compiler is required to compile the ".scss" file. This is usually done with a VisualStudio Code extension named "Watch Sass" [69] that compiles the file dynamically into one CSS file in the same folder. With the impossibility of using this extension, a compiler was created using the Node-Sass. Node-Sass is a library that provides binding for Node.js [70] to LibSass, the C version of the stylesheet preprocessor, Sass [71]. This way, it was possible to create a file mapping each Sass files to the different Lightning Web Components and, instead of using regular CSS, Sass was used to style the components. Node-Sass also has an MIT License that allows for Commercial user, Modification, Distribution, and Private use.

6. Testing

In order to keep an application working correctly, it is necessary to have it tested and approved by the users that will use it. Even though currently there are many types of tests, this project only uses two types: unit tests and acceptance tests. In this chapter, both these types will be discussed and described.

6.1. Unit Tests

As a Salesforce rule of thumb, in order to have an application deployed into a production sandbox, there must be enough unit tests that cover all the Apex classes used by at least 75% in each, not globally. Ideally, the unit tests would be the first thing to be implemented, and only then the applications, triggers, or etc, would be implemented, making use of the test class to find when the logic is working correctly. However, this idea is very hard to implement in most scenarios since sometimes there is not much certainty on how the final product will look like.

Even though Salesforce makes use of this type of test to make sure that the users are covering the lines of the classes, it is possible that these do not actually test the application. This is a bad practice and instead, the final result of the test classes should provide the certainty that the code logic is working as intended.

Test classes are simply Apex classes with some annotations that allow them to be executed and to save the test results into the sandbox. These annotations are @isTest and @testSetup and an example of them being used is in Figure 23 [72]. The first one being used at the top of the class and before every test method and the second one being used in the first method to create mock data that will be tested in the following test methods. This data is not stored anywhere in the sandbox and, at the end of each of the test methods, if the data was modified, it will be rollbacked in order to be used for any remaining test methods with the values used in the test setup method.

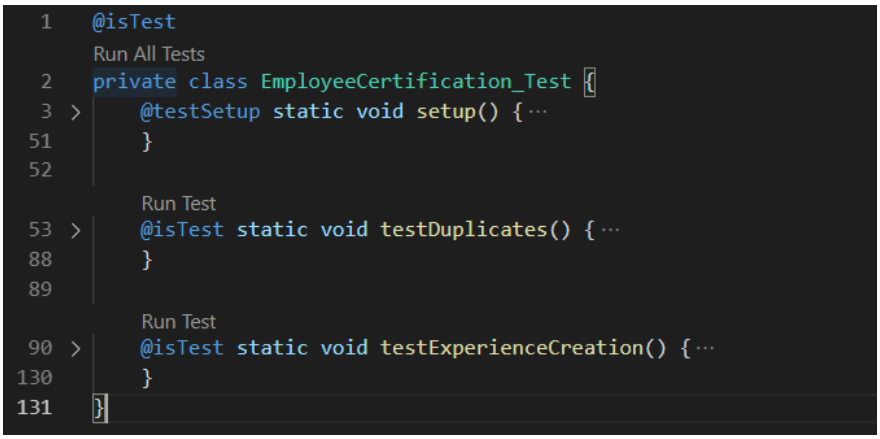


Figure 23 - Test Class Annotations

Some best practice while creating test classes include having one test class per class, making it easier to find the test class associated with a specific class and to have mock data stored in a static resource file, avoiding to change data in the test setup methods across all test classes to become outdated with new validation rules or new fields.

However, for this project, since it would be redundant to create a test class for every class without logic like the selector classes and domain classes, instead a test class per object and a test class per each controller class was created. This way, the test classes for the objects will cover the logic in all the SOC related classes and the test classes for the controllers would test the logic that they implement, thus, testing all the functionalities of the applications.

After all the unit tests were done, the sandbox had a 96% coverage on all the code.

6.2. User Acceptance Testing

Near the end of the project development, some collaborators of the company were selected to try the application and to evaluate it based on a form that is attached to Appendix 2. These collaborators included the Salesforce Platform Practice Manager, the person that specified all the requirements for the project, other practice managers from Isobar and collaborators from the Salesforce Platform practice and developers/consultants from other practices as well, in order to have more diversity on the insights given. The total of collaborators testing the application and answering the form was 17.

The form is composed of various tasks in which the user is asked to complete and to rate according to the difficulty when completing the task. The form was designed to understand if the users could use it easily or if they would struggle while completing the

tasks. It is also possible to write a comment regarding the collection of tasks for each of the two sections with tasks. Table 17 contains the average rating for each different question and time that took answering each section. (The first section of the form only contains instructions).

Table 17 - User Acceptance Testing Results Table

Question	Average Rating	Average Time
Section 1 – Question 1	2.31	Not Important
Section 2 - Question 1	1.31	6 minutes and 40 seconds
Section 2 - Question 2	1.85	
Section 2 - Question 3	1	
Section 2 - Question 4	1.08	
Section 2 - Question 5	1.08	
Section 2 - Question 6	1.38	
Section 3 - Question 1	1.46	2 minutes and 42 seconds
Section 3 - Question 2	1	
Section 3 - Question 3	1.07	

6.2.1. Results Analysis

While having the users completing the tasks and rating them on the form by their difficulty is possible to notice the different types of users that exist in the company. The ones who have never seen a Salesforce application, the ones who have never programmed one but know what it is and had previous experience with it, the ones who program applications in Salesforce daily but do not use the Lightning visuals and the ones who do use them.

However, the results are not much different from user to user as expected. Obviously, the ones that had never seen something like this had a slow start while completing the tasks but caught up very easily while claiming that once they understand the concept and paradigm, the application was easy to use and intuitive. These users were the ones that were most interesting to observe since their difficulties created space for changes. One of the changes was prior to the evaluations and was already mentioned: the URL fields. But the most impactful change that reduced the times for everyone that did the tasks after a first short analysis, including the collaborators that are not used to use the Lightning visuals in Salesforce, is the change of having the related list tab disappearing and instead, see all the related lists for the record directly in the layout (the related lists being all the links to related records of the one the user is seeing). This change can be seen in the following two figures where Figure 24 and Figure 25 contain the previous layout and Figure 26 contains the new layout.

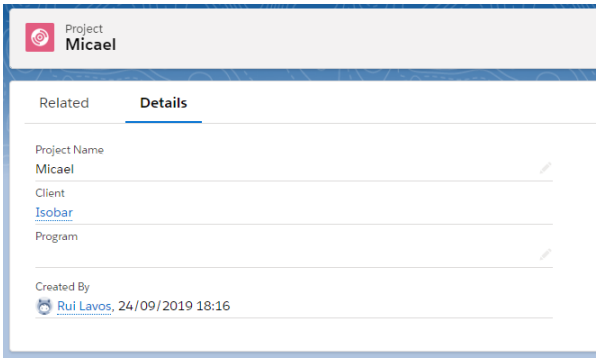


Figure 25 - Details Tab

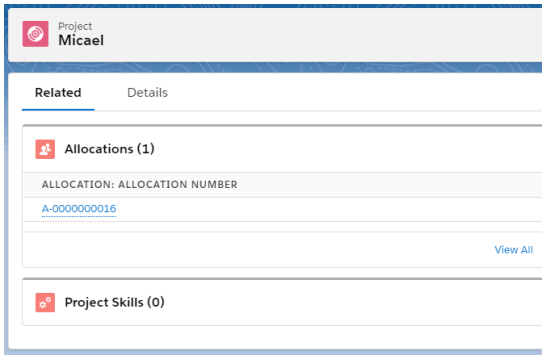


Figure 24 - Related Lists Tab

Figure 26 - New Layout with Related Lists not being in a Tab

Before writing any comments on the first section, the collaborators were invited to look at the existing Excel file that is being used to do the same operations that they just did. Every collaborator preferred the new way to organize the data against the old one and the general comment was that information was much more readable and easier to look for.

7. Conclusion

The objective of this project was to transform the existing Excel file into a working Salesforce application containing some features that could enhance the user experience while using it. This was accomplished in the sense that the application was built successfully, with a data model that follows the existing data in the Excel file and exceeds it, allowing for more different types of information to be stored and consulted while having the possibility to keep escalating the data model and features inside the sandbox.

While working on this project, some difficulties were found, mostly because Lightning Web Components are a recent framework of Salesforce that is still being improved. Many of Lightning Web Components problems are not documented yet. Other difficulties had to do with Salesforce's known limits on some features, but those are already well discussed online between the Salesforce community. Some of these limitations even have ideas opened in order for Salesforce to fix them or create solutions [73].

Completing this project was the opportunity that was missing to gather experience with the new Lightning Web Components technology, that is supposed to be the next greatest thing after the Aura Components. It was very interesting to work and learn with something that is every day becoming more utilized and improved.

7.1. Future Work

Developing a project in a platform like Salesforce that has so many features and possibilities compels to keep on creating new things and have more and more functionalities. Obviously, given the scope of the project, many of these functionalities could never be implemented in the given time frame but should always be welcome to be created in future iterations of the project. Some of these ideas to improve the project include the usage of the Salesforce Einstein product [74] with the Skill Search application to enhance the search with the aid of artificial intelligence.

Recalling the Skills Base software, one big change on this project could be the creation of a Salesforce Community [75] where the collaborators could log in and have the option to request directly their absences, manage their certificates, their skills, their personal data, etc. This would be the next step in terms of management for an application of this kind.

It was never a requirement since the company has its own tools to do some tasks like this, but it should always be preferable to have every functionality centralized in the same system.

Another improvement that could be made in the future is to make use of the FullCalendar capabilities to create the original desired Lightning application with the capabilities of the Float software. This would also include being able to drag and drop the events on the Employee record view and updating the Allocation and Absence records with these events.

Bibliographic References

[1] J. L. Spiegel, "(12) Patent Application Publication (10) Pub. No.: US 2003/0177027 A1," 2003.

[2] "Isobar Switzerland | Global Digital Agency." [Online]. Available: https://www.isobar.com/ch/en/. [Accessed: 01-Sep-2019].

[3] "Leading Digital Transformation Player blue-infinity becomes Isobar | Isobar News." [Online]. Available: https://www.isobar.com/ch/en/news/leading-digital-transformation-player-blue-infinity-becomes-isobar/. [Accessed: 01-Sep-2019].

[4] "Dentsu Aegis Network." [Online]. Available: https://www.dentsuaegisnetwork.com/#top. [Accessed: 28-Aug-2019].

[5] "Skills Base - Skills management software." [Online]. Available: https://www.skills-base.com/. [Accessed: 26-Sep-2019].

[6] "Skills Base - Testimonials." [Online]. Available: https://www.skills-base.com/testimonials. [Accessed: 26-Sep-2019].

[7] "Employee Relationship Management (Next level Human Resources Management) - mployee - AppExchange." [Online]. Available: https://appexchange.salesforce.com/appxListingDetail?listingId=a0N30000005utCwEAI. [Accessed: 30-Aug-2019].

[8] "AI Learning Platform: Learning Management System (LMS) and Social Learning ecosystem." [Online]. Available: https://www.docebo.com/. [Accessed: 26-Sep-2019].

[9] "How to Create and Manage Skills for the Docebo Perform Module." [Online]. Available: https://www.docebo.com/knowledge-base/skills/. [Accessed: 26-Sep-2019].

[10] "Spreadsheet Software - Excel Free Trial - Microsoft Excel." [Online]. Available: https://products.office.com/en/excel. [Accessed: 02-Sep-2019].

[11] "Float - Resource Scheduling App and Team Planning Software." [Online].

Available: https://www.float.com/. [Accessed: 29-Aug-2019].

[12] “Okta | The Identity Standard.” [Online]. Available: https://www.okta.com/. [Accessed: 02-Sep-2019].

[13] “OneLogin: Workforce/Customer Identity & Access Management (IAM).” [Online]. Available: https://www.onelogin.com/. [Accessed: 02-Sep-2019].

[14] “Scaling agile in large organizations | Atlassian.” [Online]. Available: https://www.atlassian.com/agile/agile-at-scale. [Accessed: 11-Sep-2019].

[15] “Scrum - what it is, how it works, and why it’s awesome.” [Online]. Available: https://www.atlassian.com/agile/scrum. [Accessed: 11-Sep-2019].

[16] “What is a Scrum Master? | Atlassian.” [Online]. Available: https://www.atlassian.com/agile/scrum/scrum-master. [Accessed: 11-Sep-2019].

[17] “Agile Scrum Roles | Atlassian.” [Online]. Available: https://www.atlassian.com/agile/scrum/roles. [Accessed: 11-Sep-2019].

[18] “Sprints | Atlassian.” [Online]. Available: https://www.atlassian.com/agile/scrum/sprints. [Accessed: 11-Sep-2019].

[19] “The product backlog: your ultimate to-do list | Atlassian.” [Online]. Available: https://www.atlassian.com/agile/scrum/backlogs. [Accessed: 11-Sep-2019].

[20] “Standups for agile teams | Atlassian.” [Online]. Available: https://www.atlassian.com/agile/scrum/standups. [Accessed: 11-Sep-2019].

[21] “Three steps to better sprint reviews | Atlassian.” [Online]. Available: https://www.atlassian.com/agile/scrum/sprint-reviews. [Accessed: 11-Sep-2019].

[22] “Sprint Planning | Atlassian.” [Online]. Available: https://www.atlassian.com/agile/scrum/sprint-planning. [Accessed: 11-Sep-2019].

[23] “Trello.” [Online]. Available: https://trello.com/en. [Accessed: 26-Aug-2019].

[24] “Jira | Issue & Project Tracking Software | Atlassian.” [Online]. Available: https://www.atlassian.com/software/jira. [Accessed: 26-Aug-2019].

[25] “What is a Kanban Board? | Atlassian.” [Online]. Available:

https://www.atlassian.com/agile/kanban/boards. [Accessed: 25-Sep-2019].

[26] “Deploy a Community from Sandbox to Production | Lightning Communities Developer Guide | Salesforce Developers.” [Online]. Available: https://developer.salesforce.com/docs/atlas.en-us.communities_dev.meta/communities_dev/networks_migrate_overview.htm. [Accessed: 11-Sep-2019].

[27] “Introduction to SOQL and SOSL | SOQL and SOSL Reference | Salesforce Developers.” [Online]. Available: https://developer.salesforce.com/docs/atlas.en-us.soql_sosl.meta/soql_sosl/sforce_api_calls_soql_sosl_intro.htm. [Accessed: 26-Aug-2019].

[28] “Understand Separation of Concerns Unit | Salesforce Trailhead.” [Online]. Available: https://trailhead.salesforce.com/en/content/learn/modules/apex_patterns_sl/apex_patterns_sl_soc. [Accessed: 26-Aug-2019].

[29] “Trailhead | The fun way to learn.” [Online]. Available: https://trailhead.salesforce.com/en. [Accessed: 27-Sep-2019].

[30] “Apex Enterprise Patterns: Domain & Selector Layers | Salesforce.” [Online]. Available: https://trailhead.salesforce.com/en/content/learn/modules/apex_patterns_dsl. [Accessed: 27-Sep-2019].

[31] “Apex Enterprise Patterns: Service Layer | Salesforce Trailhead.” [Online]. Available: https://trailhead.salesforce.com/en/content/learn/modules/apex_patterns_sl. [Accessed: 27-Sep-2019].

[32] “Learn Service Layer Principles Unit | Salesforce Trailhead.” [Online]. Available: https://trailhead.salesforce.com/content/learn/modules/apex_patterns_sl/apex_patterns_sl_learn_sl_principles. [Accessed: 29-Sep-2019].

[33] “Learn Selector Layer Principles Unit | Salesforce Trailhead.” [Online]. Available: https://trailhead.salesforce.com/en/content/learn/modules/apex_patterns_dsl/apex_patterns_dsl_learn_selector_l_principles. [Accessed: 29-Sep-2019].

[34] “Learn Domain Layer Principles Unit | Salesforce Trailhead.” [Online]. Available: https://trailhead.salesforce.com/en/content/learn/modules/apex_patterns_dsl/apex_patterns_dsl_learn_dl_principles. [Accessed: 29-Sep-2019].

[35] “Define Sharing Rules Unit | Salesforce Trailhead.” [Online]. Available: https://trailhead.salesforce.com/en/content/learn/modules/data_security/data_security_sharing_rules. [Accessed: 27-Sep-2019].

[36] “Understand Custom & Standard Objects Unit | Salesforce Trailhead.” [Online]. Available: https://trailhead.salesforce.com/en/content/learn/modules/data_modeling/objects_intro. [Accessed: 02-Sep-2019].

[37] “Tailor Business Processes to Different Users.”

[38] “Considerations for Creating and Updating Record Types and Picklists.”

[39] “Crow’s Foot Notation.” [Online]. Available: http://www2.cs.uregina.ca/~bernatja/crowsfoot.html. [Accessed: 02-Sep-2019].

[40] “Dependent Picklists.”

[41] “Git.” [Online]. Available: https://git-scm.com/. [Accessed: 26-Aug-2019].

[42] “Sourcetree | Free Git GUI for Mac and Windows.” [Online]. Available: https://www.sourcetreeapp.com/. [Accessed: 26-Aug-2019].

[43] “Bitbucket | The Git solution for professional teams.” [Online]. Available: https://bitbucket.org/product. [Accessed: 26-Aug-2019].

[44] “GitHub - joeferraro/MavensMate-Desktop: Packaged desktop app for MavensMate server.” [Online]. Available: https://github.com/joeferraro/MavensMate-Desktop. [Accessed: 30-Aug-2019].

[45] “Salesforce DX | Build Together and Deliver Continuously | Salesforce Developers.” [Online]. Available: https://developer.salesforce.com/platform/dx. [Accessed: 30-Aug-2019].

[46] “Set Up Salesforce DX Unit | Salesforce Trailhead.” [Online]. Available: https://trailhead.salesforce.com/en/content/learn/modules/sfdx_app_dev/sfdx_app_d

ev_setup_dx. [Accessed: 30-Aug-2019].

[47] S. Bobrowski, “The Force.com Multitenant Architecture,” *Dly. Cloud Feed*, p. 16, 2008.

[48] “Why Salesforce is the #1 CRM for growing businesses - Salesforce.com.” [Online]. Available: https://www.salesforce.com/crm/is-salesforce-a-crm/. [Accessed: 26-Sep-2019].

[49] “Introducing Lightning Web Components | Developer Force Blog.” [Online]. Available: https://developer.salesforce.com/blogs/2018/12/introducing-lightning-web-components.html. [Accessed: 25-Aug-2019].

[50] “Introducing Lightning Web Components | Developer Force Blog.” [Online]. Available: https://developer.salesforce.com/blogs/2018/12/introducing-lightning-web-components.html. [Accessed: 28-Sep-2019].

[51] “Create a Developer or Sandbox Org | Apex Developer Guide | Salesforce Developers.” [Online]. Available: https://developer.salesforce.com/docs/atlas.en-us.apexcode.meta/apexcode/apex_intro_get_dev_account.htm. [Accessed: 25-Aug-2019].

[52] “Salesforce AppExchange | Leading Enterprise Cloud Marketplace.” [Online]. Available: https://appexchange.salesforce.com/. [Accessed: 26-Aug-2019].

[53] “Apex Design Patterns - Singleton - developer.force.com.” [Online]. Available: https://developer.salesforce.com/page/Apex_Design_Patterns_-_Singleton. [Accessed: 25-Aug-2019].

[54] “HTML Templates.” [Online]. Available: https://lwc.dev/guide/html_templates. [Accessed: 25-Aug-2019].

[55] “Components - Salesforce Lightning Component Library.” [Online]. Available: https://developer.salesforce.com/docs/component-library/overview/components. [Accessed: 25-Aug-2019].

[56] “lightning-dual-listbox - example - Salesforce Lightning Component Library.” [Online]. Available: https://developer.salesforce.com/docs/component-library/bundle/lightning-dual-listbox/example. [Accessed: 25-Aug-2019].

[57] “lightning-radio-group - example - Salesforce Lightning Component Library.” [Online]. Available: https://developer.salesforce.com/docs/component-library/bundle/lightning-radio-group/example. [Accessed: 25-Aug-2019].

[58] “lightning-combobox - example - Salesforce Lightning Component Library.” [Online]. Available: https://developer.salesforce.com/docs/component-library/bundle/lightning-combobox/example. [Accessed: 25-Aug-2019].

[59] “lightning-slider - example - Salesforce Lightning Component Library.” [Online]. Available: https://developer.salesforce.com/docs/component-library/bundle/lightning-slider/example. [Accessed: 02-Sep-2019].

[60] “lightning-button - example - Salesforce Lightning Component Library.” [Online]. Available: https://developer.salesforce.com/docs/component-library/bundle/lightning-button/example. [Accessed: 02-Sep-2019].

[61] “lightning-datatable - example - Salesforce Lightning Component Library.” [Online]. Available: https://developer.salesforce.com/docs/component-library/bundle/lightning-datatable/example. [Accessed: 25-Aug-2019].

[62] C. Tricks, “A Complete Guide to Flexbox | CSS-Tricks,” 2018. [Online]. Available: https://css-tricks.com/snippets/css/a-guide-to-flexbox/. [Accessed: 11-Sep-2019].

[63] “Salesforce on the App Store.” [Online]. Available: https://apps.apple.com/us/app/salesforce/id404249815. [Accessed: 02-Sep-2019].

[64] “FullCalendar - JavaScript Event Calendar.” [Online]. Available: https://fullcalendar.io/. [Accessed: 26-Aug-2019].

[65] “Static Resources.” [Online]. Available: https://help.salesforce.com/articleView?id=pages_static_resources.htm&type=5. [Accessed: 26-Aug-2019].

[66] “Platform Events Developer Guide | Platform Events Developer Guide | Salesforce Developers.” [Online]. Available: https://developer.salesforce.com/docs/atlas.en-us.platform_events.meta/platform_events/platform_events_intro.htm. [Accessed: 27-Aug-2019].

[67] “Reports and Dashboards.”

[68] “Sass: Syntactically Awesome Style Sheets.” [Online]. Available: https://sass-lang.com/. [Accessed: 02-Sep-2019].

[69] “Live Sass Compiler - Visual Studio Marketplace.” [Online]. Available: https://marketplace.visualstudio.com/items?itemName=ritwickdey.live-sass. [Accessed: 28-Sep-2019].

[70] “Node.js.” [Online]. Available: https://nodejs.org/en/. [Accessed: 02-Sep-2019].

[71] “GitHub - sass/node-sass: Node.js bindings to libsass.” [Online]. Available: https://github.com/sass/node-sass/. [Accessed: 02-Sep-2019].

[72] “Annotations | Apex Developer Guide | Salesforce Developers.” [Online]. Available: https://developer.salesforce.com/docs/atlas.en-us.apexcode.meta/apexcode/apex_classes_annotation.htm. [Accessed: 11-Sep-2019].

[73] “IdeaExchange Status definitions.” [Online]. Available: https://help.salesforce.com/articleView?id=000333882&type=1&mode=1. [Accessed: 03-Sep-2019].

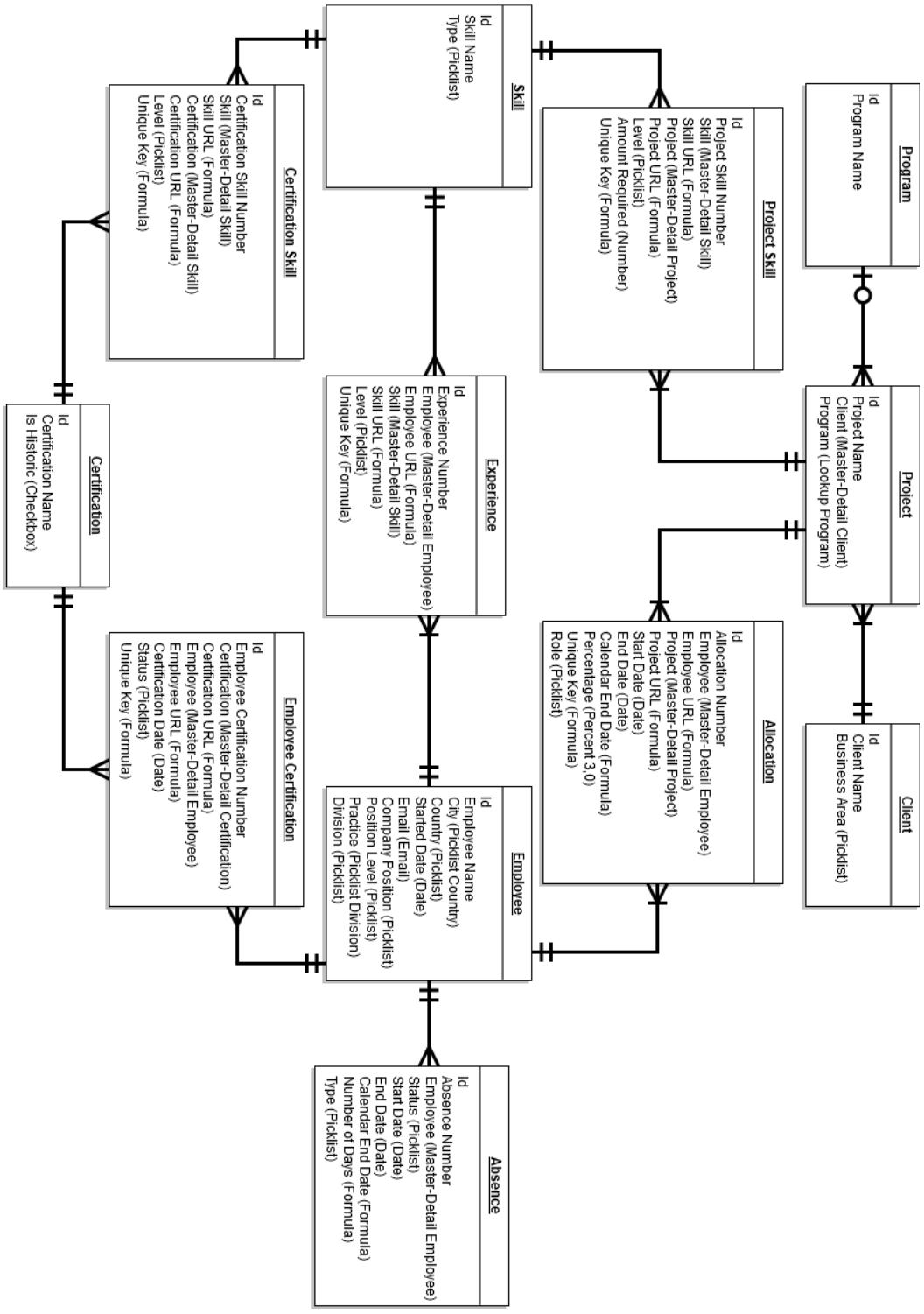
[74] “Artificial Intelligence Technology and Resources: Salesforce Einstein - Salesforce.com.” [Online]. Available: https://www.salesforce.com/products/einstein/overview/#. [Accessed: 03-Sep-2019].

[75] “Salesforce Communities Overview.” [Online]. Available: https://help.salesforce.com/articleView?id=networks_resources.htm&type=5. [Accessed: 25-Sep-2019].

Appendices

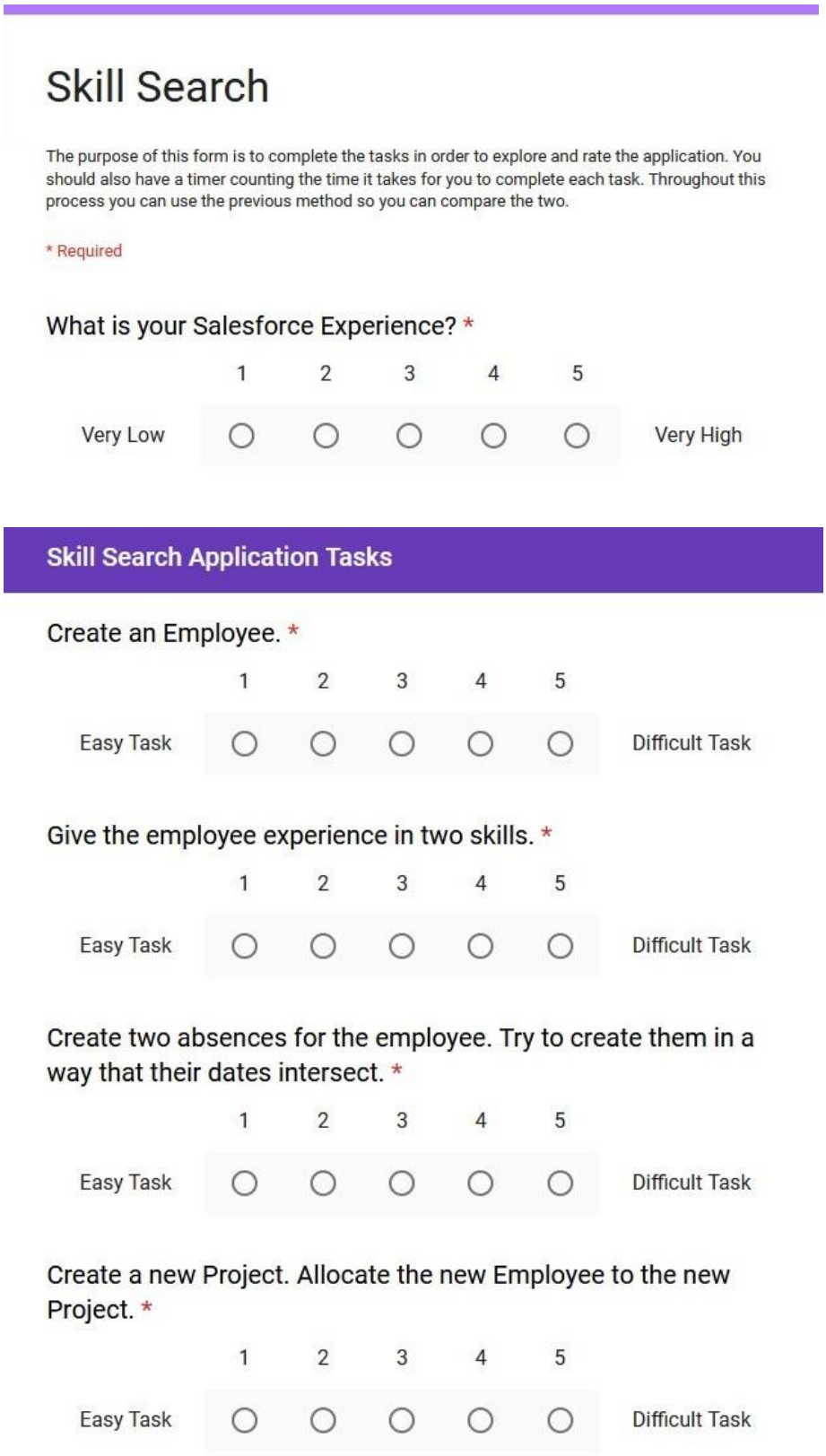
Appendix A

Entity Relationship Diagram in a larger scale



Appendix B

User Acceptance Testing form



Skill Search

The purpose of this form is to complete the tasks in order to explore and rate the application. You should also have a timer counting the time it takes for you to complete each task. Throughout this process you can use the previous method so you can compare the two.

* Required

What is your Salesforce Experience? *

	1	2	3	4	5	
Very Low	○	○	○	○	○	Very High

Skill Search Application Tasks

Create an Employee. *

	1	2	3	4	5	
Easy Task	○	○	○	○	○	Difficult Task

Give the employee experience in two skills. *

	1	2	3	4	5	
Easy Task	○	○	○	○	○	Difficult Task

Create two absences for the employee. Try to create them in a way that their dates intersect. *

	1	2	3	4	5	
Easy Task	○	○	○	○	○	Difficult Task

Create a new Project. Allocate the new Employee to the new Project. *

	1	2	3	4	5	
Easy Task	○	○	○	○	○	Difficult Task

Visualize the allocations and absences of the new employee on the calendar. *

	1	2	3	4	5	
Easy Task	○	○	○	○	○	Difficult Task

Use a report to visualize a chart with a count of all the skills in the company. *

	1	2	3	4	5	
Easy Task	○	○	○	○	○	Difficult Task

Comments:

Your answer

Skill Search Page Tasks

Use the Skill Search page to search for all the employees that know Apex or English. *

	1	2	3	4	5	
Easy Task	○	○	○	○	○	Difficult Task

Filter the search results by adding a certificate into the search parameters. *

	1	2	3	4	5	
Easy Task	○	○	○	○	○	Difficult Task

Change the priority of each topic to change the results. *

	1	2	3	4	5	
Easy Task	○	○	○	○	○	Difficult Task

Comment your evaluation:

Your answer