UKRAINIAN CATHOLIC UNIVERSITY

BACHELOR THESIS

Automation of service management for residents based on Salesforce Platform

Author:
Olena MATRUNYCH

Supervisor:
Viktoriya TSYTSAK

A thesis submitted in fulfillment of the requirements for the degree of Bachelor of Science

in the

Department of Computer Sciences
Faculty of Applied Sciences

Lviv 2022

Declaration of Authorship

I, Olena MATRUNYCH, declare that this thesis titled, "Automation of service management for residents based on Salesforce Platform" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed:

Date:

"The whole of science is nothing more than a refinement of everyday thinking. "

Albert Einstein

UKRAINIAN CATHOLIC UNIVERSITY

Faculty of Applied Sciences

Bachelor of Science

Automation of service management for residents based on Salesforce Platform

by Olena MATRUNYCH

Abstract

At a time when the world is experiencing an era of digital transformation, huge industries such as manufacturing, healthcare, education, retail and real estate are becoming digital. To stay competitive, businesses need to modernize and automate their processes and services.
In particular, property management companies need to transform their processes such as lease and account management, resident service and maintenance management. The first two areas are already primarily digitized, while there are still services with a need to order them via phone call or personal request.
When something breaks in the house, quite a lengthy process of choosing a repairman is expected, including searching for a suitable candidate, feedback overview, and then calling to order the service. And if there is a breakdown in a rented apartment in a new city, where all the repair services are unknown, it causes great difficulties and results in significant efforts.
The purpose of this work is to solve this business problem and improve the experience of residents using the services of a property management company.
To achieve this goal, a solution for resident management was established and a prototype of an automated service management system for residents based on the Salesforce platform was proposed.

Acknowledgements

First of all, I want to express gratitude to my supervisor Viktoriya Tsytsak for providing guidance and feedback throughout this work.
I am very grateful to my family for constant support and blind belief in me even when I was in despair.
I could not have undertaken this journey without three precious girls I have met in university. Thank you for the opportunity to share my worries, fears and victories. I would not have been able to overcome all the challenges without their support and soothing words.
Finally, I am deeply indebted to the Armed Forces of Ukraine for every day I had the opportunity to write this work being at home.

Contents

List of Figures

List of Abbreviations

CRM	**C**ustomer **R**elatioship **M**anagement
PaaS	**P**latform **as** **a** **S**ervice
SaaS	**S**oftware **as** **a** **S**ervice
LWC	**L**ightning **W**eb **C**omponent
FSL	**F**ield **S**ervice **L**ightning
SA	**S**ervice **A**ppointment

To my parents

Chapter 1

Introduction

1.1 Motivation

Sooner or later, every adult is faced with the question of living separately from their parents, and then immediately there is another dilemma: whether to buy their own apartment or rent a property.

The main advantage of property purchase is obvious — it will be your own home, for which you do not need to pay rent permanently. When you are young, just after graduating from university, you cannot afford to buy a home. This is a case when renting a home becomes your way of living separately and independently. Although renting is not only a student's choice, renting has many advantages such as no large investments, no need to be bound to the house for life, you can always move and try to live in another type of housing, another district or city.

The increase in the number of rent over the housing purchase is also influenced by global changes. Since the COVID-19 pandemic and quarantine, people have begun to work remotely and become more flexible. Now they do not want to settle down in one place, in one home, when there is an opportunity to work from almost anywhere in the world. Therefore, for such people, the choice of renting rather than buying a home is obvious. In the digital age, a person who rents a home needs more than just a room, he/she also needs quality service and quick solutions to household problems.

That is why I decided to develop a solution that can improve the resident's experience while living in a rented house, as well as improve the management of residents, properties and their problems for landlords and real estate agencies. The purpose of this work is to simplify these inconveniences and to develop a solution for resident management. As a result, the prototype of a service system for residents is developed.

1.2 Problem

Assume you rented a home in a foreign city and one of your home appliances broke down. To repair this item, you need to find a good technician and review his feedback. It is not so easy in a new city. If this is your landlord's equipment, then you probably want to inform him about the breakdown and ask for help. There are too many steps, do you agree?

Now imagine yourself as a landlord who has 15 apartments for rent. Today you have to remind everyone about the rent payment and extend the lease agreement with 3 tenants. In one apartment, the refrigerator does not work; in the other two, there are leaking faucets. Tenants have been asking you for help for a week. There are a lot of things to do for a day, even if you have a dream vacation.

Each role has its problems. It is difficult for a landlord to track residents' problems and monitor contracts and payments. The resident has difficulties in repairing breakdowns and making rent payments.

1.3 Goal

The purpose of my work is to improve the resident experience of receiving service, as well as simplify field service management for landlords and real estate agents.

To fulfill my goals, real estate rental market research will be done. I will focus on the US market, as there is a lot of research on this region. I also need to analyze related solutions that already exist, feedback, and determine which functionality works well, and which could be improved or added.

The result will be a prototype of a system that is developed on the Salesforce platform. In this system, landlords and real estate agents will be able to track the life of a resident in their rented accommodation. They will also have an automated service process for their tenants. For residents, this system will look like a portal with the opportunity to track their payments, leases, and most importantly the possibility to submit their issues related to household malfunction.

Chapter 2

Real Estate Rent Market Overview

2.1 Overview

The size of the US real estate rental market is large. About 34%, that is 44 million out of total 122 million households in all states are renter-occupied. One-third of US households have rented their homes for the last 50 years.[16] Let's look at the recent changes that are continuing in the market. 2020 was a difficult year of challenges for small and large businesses. How has the coronavirus pandemic affected the real estate rental market?

2.2 Pandemic impact

At the beginning of the pandemic rental housing markets went through a slight decline. Housing Studies of Harvard University conducts exhaustive research about the rental market in 2021.[1] On the basis of this work the market analysis is carried out.

Despite negative changes, the number of tenant households increased by the end of 2020, numbered 44 million, which is 870,000 more households than in the first quarter of 2020.

There have been noticeable changes in the market for professionally managed apartments. At the beginning of 2020, the highest decline in prices was experienced by apartments of the highest class, which pulled down the total price of rental real estate. However, the next year's rental prices increased and already in the third quarter of 2021 exceeded the 2019 values. Rents for apartments of medium and higher quality even soared by almost 9 and 15 percent respectively.

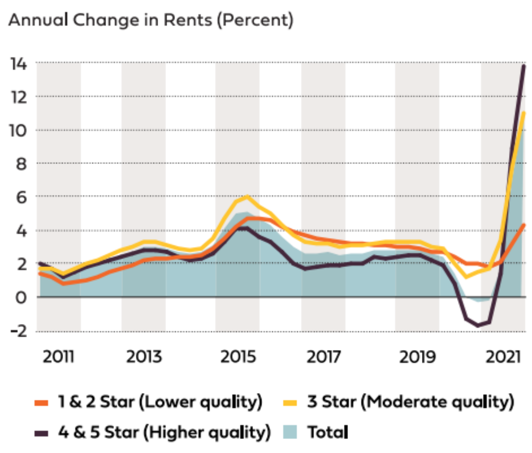


FIGURE 2.1: Annual Change in Rents by Harvard University Joint Center [1]

The economy began to recover in 2021, so the number of rents began to grow. US renters are people of different wealth, but sometimes rent is their only way out, as buying a home is very expensive with limited availability.

Let us consider the factors in restoring the growth of rents. First of all, the number of young people entering the rental market for the first time increased in late 2020 and early 2021. However, older people also continued to rent housing. Representatives of the millennial generation reach the age of 20-30. This is the age when rental housing is most common. In the United States, the number of rental housing among adults under 35 is growing.

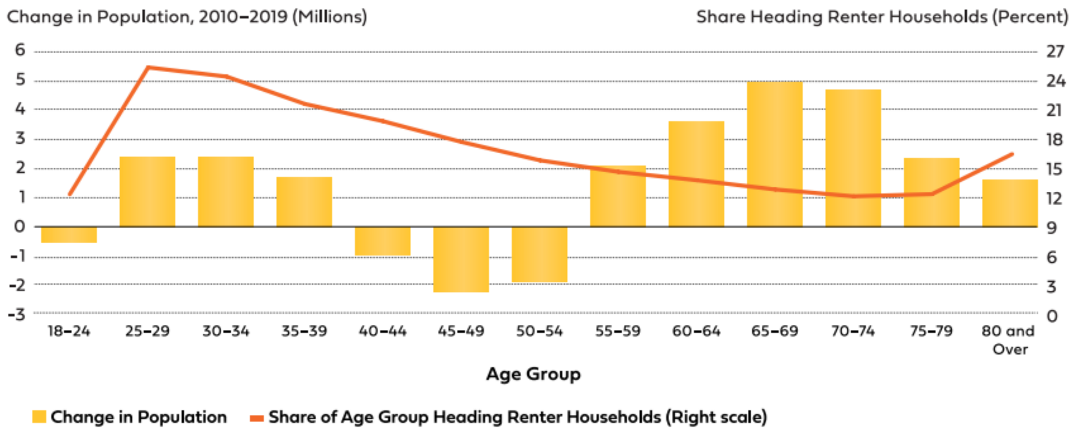


FIGURE 2.2: Distribution of rented households among age groups by Harvard University Joint Center [1]

But during the pandemic, the number of relocations to live on their own fell slightly, creating a delayed demand for 2021. Millennials, which peaked at 29 in 2020, postponed their demand for the next year too.

The second is a change in people's attitudes to buying and renting housing. According to the survey, the number of people who believe that now is a good time to buy a home has fallen from 61% in 2020 to 24% in 2022. The number of those who believe that now is a bad time to buy a home has grown from 27% in 2020 to 73% in 2022.[5] As a result, the number of renewed leases reaches record highs.

According to a study of Entrata in 2021, which examined the behaviour of renters, let's highlight the following causes of the predominance of rent over the purchase of their own homes. One of the main reasons is the high cost of housing. 39% of 1,000 respondents cannot afford even an advance payment for their own home, and 33% are unable to keep homeownership. Because of the pandemic, many Americans have started and continue to work from home. It has also increased their mobility, with almost 66% of tenants saying renting is more in line with their current lifestyle than owning a home. People like the fact that they can move closer to relatives or friends, try to live in different parts of the city and not settle down in one place for a long time. Another fact is that nearly half of the respondents increased their interest in monthly payments more than annual ones and are even willing to pay more for rent to be able to make month-to-month payments.[4]

In conclusion, the COVID-19 pandemic has had a negative impact on the US rental market. However, in 2021 demand sharply recovered. As demand has risen, housing prices have risen, making it impossible for many people to become homeowners and they are forced to rent. In the coming years, the demand for rental housing will grow even more. The ageing of millennials will not reduce the number of rented apartments, as they will continue to rent at higher prices to be mobile, to avoid the risks of buying a home due to the uncertainty of world situations.

After the pandemic in 2020, the demand for rental housing in the United States began to grow with renewed vigour. As demand grows, homeowners need to offer tenants more than just space to stay competitive.

2.3 Real Estate Digitalization

One major takeaway from the global shutdown during the COVID-19 pandemic is that the future is digital. Digitalization captures all spheres of life to simplify routine human activities. Of course, the digital concept is becoming more common in the real estate industry. The real estate industry is undergoing renovations from product and space to systems and services. Services improve the user experience. Their value focuses on providing services to people based on their needs and ways to cover them. Deloitte researched what innovations are expected in the real estate market after the coronavirus pandemic. And here is an interesting statement about the transformation of the main value: "Real estate is therefore transformed from ″fixed product″ (space and infrastructure) to ″connected system″ (information and services)." Digital services include property, leases, and payments management. They can be monetized by additional rent charges or individual subscriptions, which will bring additional income to the landlords.[2]

Digitalization is already here and is developing in the real estate market. Therefore, homeowners need to adopt it to stay on the same level as competitors. The real estate market has a large turnover of about $ 88.4 billion in annual revenue. Therefore, many software companies are ready to help investors and homeowners stay competitive and organized in the market. Property management software will be able to attract and manage tenants, respond to their maintenance requests, and account for all related costs and revenues. The main advantage of real estate management software is its ability to automate various tasks, which saves time and resources. Another advantage is that digital software means that all your data is securely stored in the cloud, giving you access to it from anywhere at any time, as well as ensuring its security and backup.

One of the areas of digitalization of property management which improves the tenant experience is the service management of residents. How is it now implemented in the software market? What solutions already exist?

Chapter 3

Related works

3.1 Buildium

Buildium by RealPage company is a cloud-based property management solution. This software allows tenants to rent online, renew contracts and make payments. Also provides complete general ledger accounting. Residents and property owners can apply for maintenance through the site and mobile application. Property managers can convert requests to work orders and assign them to service and personnel managers, then change the status of requests or confirm completion. Buildium has a very large variety of features, however, for each of them, you have to pay. There are three plans: Essential - starting at 50$ per month, Growth - starting at 160$ per month and Premium - 460$, each of them includes Accounting, Maintenance, Tasks, Violations, Resident Board Member Communications, and Online Portals.

Reviews of users who use this system have shown that users are mostly satisfied. The system is well suited for keeping records of real estate, accounting, and leases. Provides access for communication between managers and tenants. However, there is some problem with maintenance requests service. This feature is not fully automated. Dealing with workflow from maintenance requests to active repair and to complete repair is confusing for managers. Technicians and vendors are not in this system with managers and residents, so managers have to check maintenance requests and transfer them to another system.

3.2 MRI software

MRI software includes a wide range of solutions such as property management, accounting, lease management, asset management and more. Consider Cloud residential property management software.

The main advantages are that cloud support provides unlimited access to data, a simple and intuitive process of attracting new tenants and concluding agreements, an online portal for residents, which expands communication between residents and managers and online rent payments.

After analyzing users' feedback, the following disadvantages of the service are a difficult training process and no free trial. There are also problems with access to technical support. The MRI portal for residents is not provided with a maintenance request function.

3.3 Yardi Breeze

Yardi Breeze is a cloud-based property management software. It provides users with features such as accounting software, searching for property and rentals online, portals for renters and residents, online payments, leases and maintenance requests management. Residents can also upload photos to their work order requests. The main advantages are a variety of different features, free training for administrators and customer support.

After analyzing users' reviews about the Yardi, users determine the easy intuitive use of software and access to the help centre. Some users consider a lot of functionality a disadvantage because it is difficult for tenants to deal with it. Another problem is the automation of processes, such as charging or automatic delivery - it does not always work properly. Yardi also lacks the ability to integrate with and synchronize with third-party resources, no mobile version. Yardi's main drawback is that technicians or vendors do not have access to the portals to track repair requests assigned to them. This requires another separate application where data for repairmen will be entered by the manager.

3.4 Summary

These three systems are decent solutions. They have many features that are useful for tenants and homeowners. They improve the experience for residents insanely, but there are still many unresolved issues that could simplify the work of landlords and real estate company managers - such as service management of residents. Buildium and Yardi collect maintenance requests, but the processing of these requests by the repairmen themselves is in a third-party application, where data is entered by managers. MRI does not have such a function at all.

On-site service and processing of maintenance requests of residents have to be fast and automated. Therefore in this work, the prototype of a system for service management of residents is developed, where there are the most necessary features for the account of residents and also field service management.

Chapter 4

Background

4.1 Salesforce overview

Salesforce is a Customer Relationship Management (CRM) platform that helps to automate and to simplify sales, service, marketing, analysis, and customer relationships. It is fully hosted on the Salesforce company's cloud servers. Based on cloud computing technology, Salesforce uses two models at once: Software as a Service - SaaS and Platform as a Service - PaaS. First, let us consider what CRM and cloud computing are.

CRM is a way to manage customer relationships and optimize business processes. The main component of business management in the CRM is a special single software for organizing work with leads - prospects, tracking customer actions, and automation of communications. It should also provide team collaboration and the opportunity to communicate through various channels with customers and between employees. Salesforce CRM includes the best properties of CRM systems:

- Convenience and ease of use. Salesforce has many guides on how to use the system and documentation. Therefore, it is easy to teach employees to use the system, so that they start working as quickly as possible.
- Flexibility. Salesforce offers many out-of-the-box solutions out of the box, however, and a wide variety of settings to customize the system to your needs.
- Mobile version. This is a very necessary option in the era of remote work. The mobile version will provide convenient 24/7 access to the system.

All information managed by the CRM system in Salesforce is stored in the cloud. The word "cloud" in this context itself represents a large black box that hides an infrastructure, a lot of data and technical details, which are invisible to the user. Cloud computing is a method of storing, processing, and accessing data that is stored on remote servers. Due to this, the software is provided to the user as an online service, with no need to install any additional programs.

As mentioned above, Salesforce uses two models of cloud computing services that differ in the type of services provided:

- SaaS is on-demand access to ready-to-use, cloud-hosted application software. Users pay a monthly or annual fee to use a complete application from within a web browser, desktop client, or mobile app. The application and all of the infrastructure required to deliver it - servers, storage, networking, middleware, application software, and data storage - are hosted and managed by the SaaS vendor. [3]
  The Salesforce CRM system is provided exclusively by the SaaS model.

- PaaS is on-demand access to a complete, ready-to-use, cloud-hosted platform for developing, running, maintaining, and managing applications. The cloud services provider hosts manage and maintain all the hardware and software included in the platform - servers (for development, testing and deployment), operating system software, storage, networking, databases, middleware, runtimes, frameworks, and development tools - as well as related services for security, operating system and software upgrades, backups and more. [3]
  PaaS Salesforce product is Force.com, which is used for the self-development of applications and websites. Using cloud IDE developers can create apps and deploy them to multi-tenant servers.

Salesforce has various cloud service products that are called "clouds". Such as Sales Cloud, Service Cloud, Marketing Cloud, Experience Cloud, and others. For my solution, I use two tools of two clouds: Service and Experience.

4.2 Backend development

For server development, Apex language is used. It is a strictly typed object-oriented programming language that has a syntax that looks like Java. [7] The language is case insensitive, but sticking to registers is considered a good practice. Apex code is stored in the format of classes and triggers, this is how Salesforce defines them[7]:

- A **class** is a template for creating Apex objects.
- A **trigger** is the code that executes in response to specific data manipulation language (DML) event.

For working with databases there are DML, Salesforce Object Query Language (SOQL) and Salesforce Object Search Language(SOSL).

- **SOQL** is a query language, used for reading information stored in your organization's database. It is similar to SQL (Structured query language), the main difference is that there is no Join clause, but it has numerous identical features although with its limitations.[9]
- **SOSL** is a query language designed purely for search. It is used for global search, less often than SOQL.[9]

Apex code execution is restricted to prevent monopolization of distributed resources, as Salesforce is built on a multitenancy architecture. There are governor limits on the platform, most often they relate to restrictions in the context of one transaction. For example, the total number of records retrieved by SOQL queries, the number of SOSL queries issued or the number of DML statements issued. [10]

Another important thing to remember while developing for Salesforce is test coverage. Before running Apex code in production, it is needed to cover it with 75% unit tests.

4.3 Frontend development

Salesforce has two types of user interfaces - Salesforce Classic, which is no longer updated and the new modern - Lightning Experience, which uses the following technologies:

- **Lightning Component Framework** is a framework for creating user interfaces, used for the development of one-page Salesforce applications for desktops and mobile devices. The framework supports two programming models: Aura Components and Lightning Web Components (LWC), which allow developers to create custom UI elements using HTML and JavaScript.
- **Visualforce** is a framework that includes an HTML-like markup language and a set of server-side controllers. Visualforce maintains an extensive library of components that make it easier for developers to create web pages. You can use the framework to modify the standard Salesforce user interface or create a completely new one. Visualforce is the oldest of the available technologies. The page or component is generated on the server-side, but the Apex code and page markup are written separately. There are some features that are still difficult to replace. For example, advanced email template, generation of PDF files, or development of SObject List page.

Separately it is worth mentioning the built-in declarative programming tools. In Salesforce there is a lot of things to do using the user interface: creating new object, fields or record. Let us consider declarative automation tools such as Workflow Rules, Process Builder, Flows, Visual Flow, and Approvals.

- **Workflow Rules** is the oldest automation tool. It is attached to a specific object and is fired after creating a new object record or when the defined condition is met. It can perform the following actions: create Task, send an Email Alert, send Outbound Message to an external web resource, and update the triggered record or fields of the master record if there is a master-detail relationship.
- **Process Builder** allows you to build logic that contains many if-else conditions. It can update the attached record and all related ones. Performs such actions: create records, update related entries, publish messages in Chatter, run Flow, call Invocable Apex code, submit an entry to Approval, and call another process.
- **Flows** can automate complex processes and can be triggered in different ways. There are several types of flow:
  - Screen flows are used to create a user interface for collecting data.
  - Scheduled-triggered flows can be scheduled for a specific time.
  - Auto launched flows can be triggered from Process builder, Apex or Platform Events.
  - Record-triggered flows are run in the background when a record of a specific object is created, edited or deleted.
  - Platform event-triggered flows are run each time when receiving a Platform Event message from subscribed to Platform Events.
- **Approval Proccess** allows automating the decision-making process.

4.4 Field Service Management Solutions by Salesforce

4.4.1 Field service management

Before describing this Salesforce product, first, let us define what field service management is.

SAP company describes field service management as the system of managing a company's field resources within the service fulfilment process. [15] Field service refers to on-site service, which is referred to as in the "field" when the worker performs a job outside the company.

Field workers are often technicians who provide qualified services in places which the client needs, where they are dispatched to install, repair or maintain equipment. Thinking about on-site service we imagine the work of electricians or internet providers, but field service management is now useful for many other industries such as healthcare, property maintenance, and construction. Regardless of the type of scope, it needs effective management, and an automated system in order to accelerate customer service, increase client satisfaction and most importantly stay competitive.

In fact, the hard reality is that half of the service companies use manual methodologies to serve customers.[6] Therefore, they cannot fully analyze their customers, workers, and the work process.

Due to the lack of a good management system companies face a number of problems. Here are the main gaps defined in the Softserve company's report:[17]

- low customer engagement due to non-support of new technologies
- no clearly defined channels of communication with customers
- no real-time communications with customers
- no notification system for late or missed appointments
- customer accounting for understanding the potential upcoming maintenance and the possibility of additional sales of goods and services
- low-efficiency rate due to lack of correct as-maintained documentation, and poor routing
- complicating the work of field technicians due to customer dissatisfaction with ordering services

Therefore, in order to improve the customer experiences, create tracking of issues and jobs, to establish communication between customers, employees and managers, it is necessary to have strong field service management software.

Salesforce Field Service is exactly the product that can cover all the problems that exist in field service management.

4.4.2 Salesforce Field Service

Salesforce Field Service, formerly known as Field Service Lightning (FSL), is a Salesforce product for the enhancement of field service management. FSL involves four key roles of users:[12]

- **Administrator** - installs FSL application and sets up features according to the business requirements, provides different access rights to the app according to the user role.
- **Agent** - in other companies is often referred to as a support group. Receives requests from customers by phone or email and forms service appointments (SA).

- **Dispatcher** - assigns SAs to field technicians and schedules them using the Dispatcher Console UI.
- **Field Technicians** - is a specialist who performs field service work. Manages assigned SAs.

FSL is a tool that provides agents, dispatchers, and technicians with the resources to optimize their working process.

The main features that considerably simplify field service management are:

- Service appointments management: the creation of appointments of a certain work order (type of work needed to perform) with the territory where it is located; assignment of technicians with the necessary skills.
- Scheduling through the Dispatcher Console: the ability to set the criteria by which the best candidates for the work order will be automatically assigned.
- Administration App: managing objects, optimization and customizing Dispatcher Console.
- Technicians and jobs monitoring: through the console, you can track employees in real-time, and monitor their performance.
- Mobile version: access to SA and other FSL features through a mobile application for technicians, constant communication with the team.

Salesforce Field Service is paid and has several types of licenses and pricing plans. For example, dispatcher and technician plans start at 150$. There is also a plan for contractors - external non-employees, with access to work orders, cases, contracts, and accounts, starting at 50$.[11]

4.5 Experience cloud

Salesforce Experience Cloud – is a digital experience platform for connecting and managing communication among employees, partners, and customers. With this product, you can quickly create forums, portals, support communities, web applications, and sites. Until 2021, it was called the Community Cloud, because its main idea was to create an online community for customers and partners. But the cloud provides much more opportunities than just building a community and has been renamed.

Digital experience is now a crucial aspect of the business. It is an interaction between customers, employees, partners, and an organization achieved through digital technologies. It can attract large numbers of customers to companies by interacting with them through portals or websites. The usage of the Experience Cloud is most concentrated in the following areas: customer and partner portals, where customers can access the resources to which they are entitled; get support for issues, and communicate.

A significant advantage of Experience Cloud is its security. It is built on the trusted Salesforce Platform with high-level security. You can define any level of user access to data and actions: determine who can log in or what data can be read or edited by granting different permissions.

The second advantage of the product is flexibility and speed of implementation. It has many templates for creating digital sites, which saves time and resources for their development, but they can also be customized.

Experience Cloud is paid. The price depends on the type of experience, and the number of users, while each user must have the appropriate paid license, which may provide different opportunities and has a different price.

4.6 Salesforce benefits

According to the results of 2020, the company Salesforce is a leading CRM provider holding a 19.8% market share.[14]

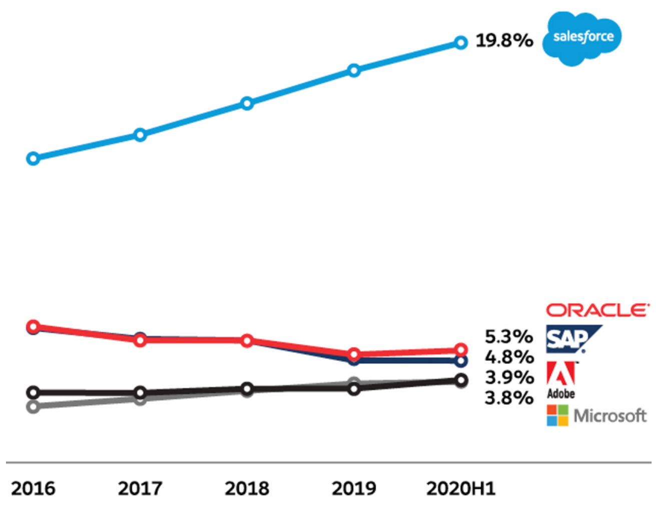


FIGURE 4.1: CRM market share change [14]

Salesforce platform provides many opportunities for development. It is possible to program using declarative tools, develop own backend and frontend and design a database according to own requirements.

Let's consider what other benefits can be gained using this platform:

- **Customisation and scalability.** As mentioned above, it is possible to customize various tools and processes on the platform. An org scales well if the number of users or data increases.
- **Multitenant Platform.** Every customer shares the same infrastructure and one platform. This allows automatic and simultaneous updates for all users without any requirement for customer intervention. Automatic updates occur three times a year.
- **Cloud computing.** One of the most significant factors and benefits of Salesforce technology. There is no need to install any programs or purchase hardware. Cloud-based technology provides 24/7 access to the platform. Internet is the one thing needed to work with Salesforce.
- **Integrations.** Salesforce has many built-in integrations and can be easily integrated into other online platforms.
- **Out-of-the-box solutions.** Provides a variety of ready-made modules such as sales, marketing, analytics and more. Users can easily configure and customise them.

- **Easy to get started.** System has an intuitive and clear navigation bar and Setup menu.
- **Fast learning.** Salesforce has a great online platform for learning - Salesforce Trailhead. There are a lot of tutorials for learning different tools.
- **Salesforce Community.** Salesforce holds regular meetings for professionals to share experiences.

Chapter 5

Solution

5.1 User roles and their functions

The prototype of the developed system is designed for three types of users and consists of two parts: the Resident Portal and the administrative Property Management App.
The proposed portal is created for the needs of the resident user, who lives on rented property.
Furthermore, the administrative app is developed for administrators to manage all residents and their maintenance requests.

The **administrator** is the user with the widest rights to records and the greatest opportunities to operate data. In the property management system, there are such use cases for administrator:

- create resident users in the system;
- create and manage leases;
- receive notification of the lease expiration;
- create and manage resident invoices;
- manage resident's maintenance requests;
- create field technicians;
- create work types;
- monitor the technicians' schedule;

As mentioned above, the FSL package is used for this solution, so the administrator is endowed with special licenses and permissions for this package: Field Service Admin License, Field Service Admin Permissions, Field Service Dispatcher License and Field Service Dispatcher Permissions. These permissions give the administrator access to the FSL functionality required to perform some of the above functions.

The main user who receives services from a real estate agency is a **resident**. A separate portal has been developed for the resident and the following use cases are covered:

- view all his leases;
- view invoices and in the future - make payments;
- view a list of his cases;
- submit maintenance requests;

As the portal is developed on the Experience Cloud, the resident needs a special license for access - Customer Community license. In order to expand access to custom objects, a separate profile has been created, which is a clone of the standard license profile but has extended access rights.

The last type of user of the system is the **field technician**. This is a person who directly performs the work to fulfil maintenance requests. He has access to FSL objects and has Field Service Resource License, Field Service Resource Permissions, FSL Mobile License. Field technician can view and manage service appointments assigned to him.

5.2 Data model

Let's consider how the data model was designed for this system and what objects are contained. Object in Salesforce is similar to a database table - a structure organized into rows and columns that is responsible for storing data. Database columns are represented as object fields in Salesforce and rows as records.

There are two types of objects in Salesforce. **Standard objects** are created and provided by Salesforce. **Custom objects** are created by a user according to their requirements. They are identified by a '__c' suffix added to the object name. Same as object fields, custom field names have '__c' ending.

There are two types of relationships in Salesforce. **Master-detail** relationship is set between two objects, one of them is a child "detail", and the other controls it - the "master". **Lookup** relationship connects two objects, like a master-detail, but it does not affect the deletion and access rights of the detail record.

For a better understanding of the objects of my solution, the data model is divided into two parts.
The first part consists of custom objects, which are created to store data on resident, real estate, leases and invoices.

The second part is the objects from the FSL package, which represent the process of work order management. In this model there are standard objects which are modified for the needs of this solution. Several standard objects have custom fields to satisfy the logic of the system. Also, some objects have special custom rules that prevent editing or inserting records, as well as there are implemented triggers.

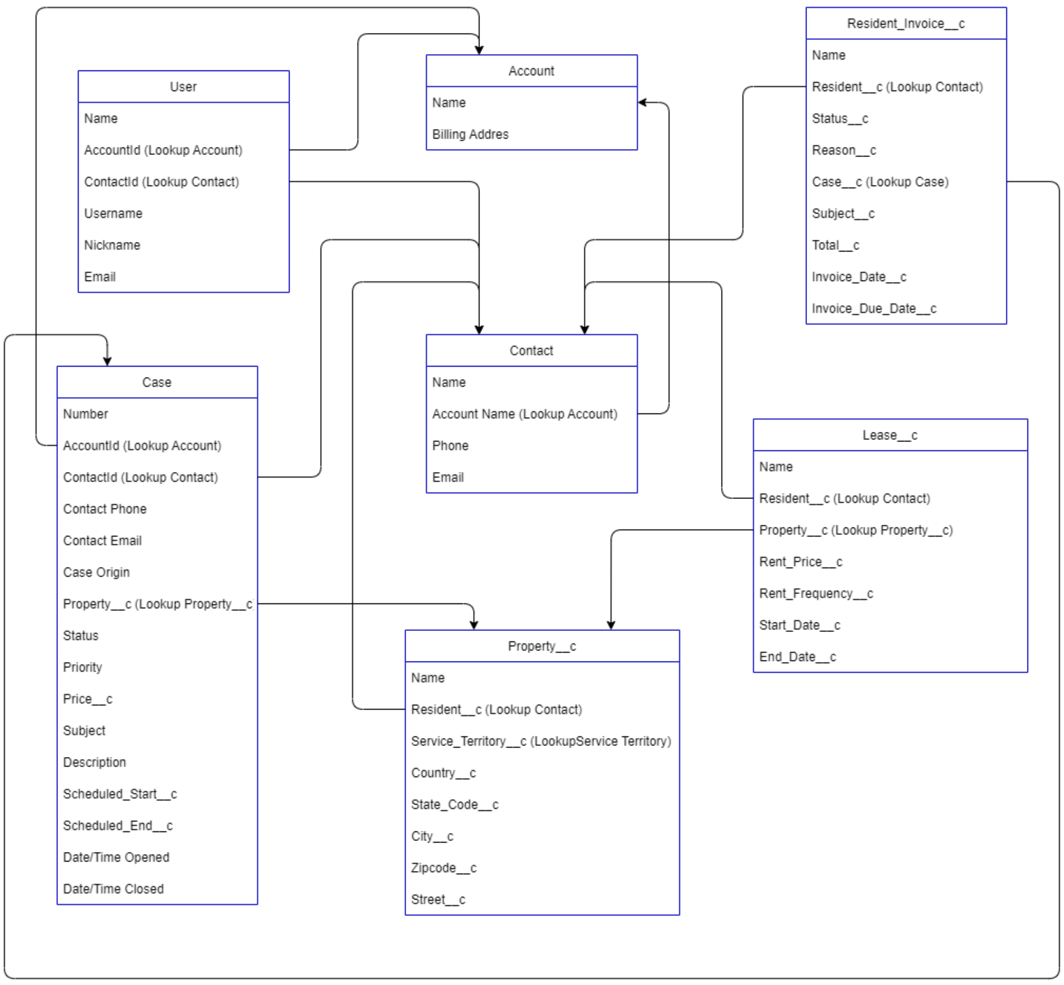


FIGURE 5.1: Data model part 1

- **User** - standard object, that represents a user of organization.
  Fields:
  - FisrtName (Type: Text) - first name of user.
  - LastName (Type: Text) - last name of user.
  - AccountId (Type: Lookup to Account) - represent an account associated with a Customer Portal user.
  - ContactId (Type: Lookup to Contact) - represents a contact linked to account
  - Username (Type: Text) - username used for log in to the org.
  - Email (Type: Text) - user's email
  - ProfileId (Type: Lookup to Profile) - reference to profile assigned to user.
- **Account** - standard object, that represents a customer account or organization involved with your business. For this solution, Account represents a real estate agency to which all real estate, residents and employees are tied.
  Fields:
  - Name (Type: Text) - the title of account.
  - BillingAddress (Type: Address) - billing address of the account.

- **Contact** - standard object, represents a contact, which is a person associated with an account. In this system, contact represents a resident.
  Fields:
  - Name (Type: Text) - compound field, that consists of FirstName, LastName, and Salutation.
  - AccountId (Type: Lookup to Account) - reference to associated account.
  - Phone (Type: Phone) - contact's phone number.
  - Email (Type: Email) - contact's email.
- **Property__c** - custom object, representing a property, which can be rented.
  Fields:
  - Name (Type: Autonumber) - auto-generated property number displayed in format "PR-0000".
  - Resident__c (Type: Lookup to Contact) - represents a resident that rents in property.
  - Service_Territory__c (Type: Lookup to Service Territory) - a reference to the service territory to which property is connected.
  - Country__c , City__c, Street__c (Type: Text), Zipcode__c (Type: Text) - fields, that represent address where property is located.
- **Lease__c** - custom object, that represents a contract by which the landlord conveys property to the resident.
  Fields:
  - Name (Type: Autonumber) - auto-generated lease number displayed in format "L-0000".
  - Resident__c (Type: Lookup to Contact) - a reference to the resident with whom the contract is concluded.
  - Property__c (Type: Lookup to Property__c) - a reference to property that is conveyed by a lease.
  - Start_Date__c (Type: Date) - the date from which the lease becomes valid
  - End_Date__c (Type: Date) - the date until which the contract is valid.
  - Rent_Price__c (Type: Currency) - the rental price presented in decimal number.
  - Rent_Frequency__c (Type: Picklist) - determines the frequency with which the rent will be paid. Picklist consists of values 1, 3, and 6, which means that the fee will be charged every 1, 3 or 6 months

  There is a validation rule on the Lease__c object. A validation rule is a special rule that verifies whether specific fields, entered by a user in a record while creating or editing, correspond to the indicated criteria. Due to this rule, End_Date__c value cannot be less than Start_Date__c value.
- **Resident_Invoice__c** - represents a financial document describing the amount that a resident has to pay for rent or services.
  Fields:
  - Name (Type: Autonumber) - auto-generated invoice number displayed in format "I-0000".

- Resident__c (Type: Lookup to Contact) - a reference to the resident to whom the invoice is created.
- Invoice_Date__c (Type: Date) - the date when the invoice was posted
- Invoice_Due_Date__c (Type: Formula (Date)) - formula fields that define the date by which the resident must pay the invoice. The formula is calculated as Invoice_Date__c + 10.
- Reason__c (Type: Picklist) - reason for creating the invoice. The picklist contains two values: Rent Payment, and Service Appointment Payment.
- Subject__c (Type: Text) - subject of the invoice.
- Case__c (Type: Lookup to Case) - a reference to the case for which the invoice was created.
- Total__c (Type: Currency) - the amount that the resident must pay.

• **Case** - standard object that represents a customer issue or problem.
Fields:

- CaseNumber (Type: Autonumber) - assigned automatically when each case is inserted. It can't be set directly, and it can't be modified after the case is created.
- ContactId (Type: Reference to Contact) - reference to contact. In our case contact is the resident.
- AccountId (Type: Reference to Account) - a reference to the resident's account.
- Contact Phone (Type: Phone) - resident's phone number.
- Contact Email (Type: Email) - resident's email.
- Case Origin (Type: Picklist) - the source of the case, such as Email, Phone, or Web. In my solution, the case origin will be Web by default, as the cases will be created by a resident from the portal
- Property__c (Type: Lookup to Property__c) - reference to resident's property.
- Status (Type: Picklist) - status of the case, can be New, In Progress, Completed, Closed. The default value is New.
- Priority (Type: Picklist) - the urgency of the case, can be High, Medium, or Low. The default value is Medium.
- Price__c (Type: Currency) - the price of the work that was done to solve the case. This field is a duplicate of the Price__c field from the Service Appointment record, which will be connected to the case record.
- Subject (Type: Text) - the subject of the case.
- Description (Type: Text) - description of the case.
- Scheduled_Start__c (Type: Datetime) - the date and time when SA is scheduled to start.
- Scheduled_End__c (Type: Datetime) - the date and time when SA is scheduled to end.
- Date/Time Opened (Type: Datetime) - the date when the case was created.
- Date/Time Closed (Type: DateTime) - the date when the case was closed.

After considering the custom object, let us move on to **FSL data model**.

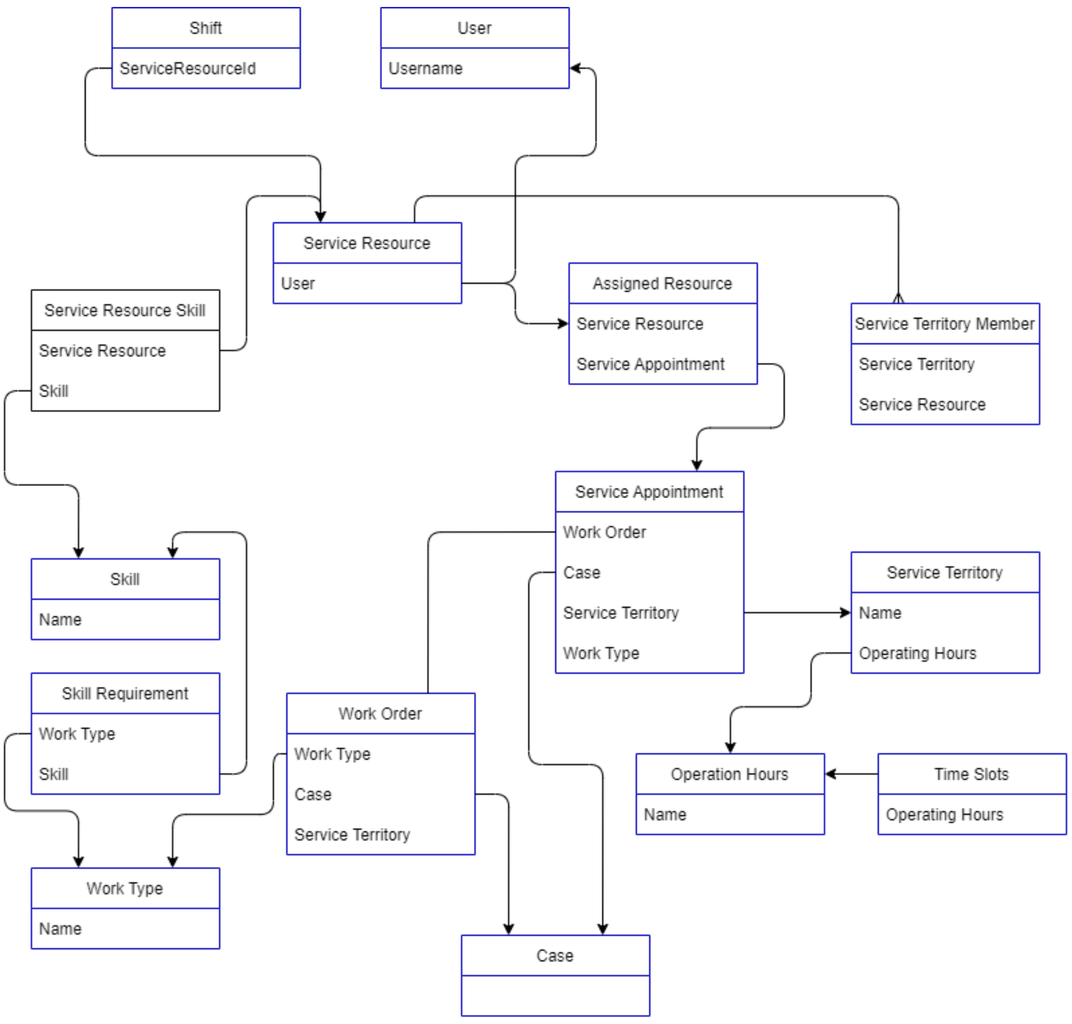


FIGURE 5.2: Data model part 2

- **Work Order** - standard object, that represents field service work to be performed for a customer.
  Fields:
  - WorkOrderNumber (Type: Autonumber) - auto-generated number that identifies the work order.
  - WorkTypeId (Type: Lookup to WorkType) - reference to the work type.
  - Status (Type: Picklist) - status of the work order. Picklist includes the following values: New, In Progress, On Hold, Completed, Closed, Cannot Complete, Canceled. The default value is New.
  - CaseId (Type: Lookup to Case) - a reference to a case connected to a work order.
  - Property__c (Type: Lookup to Property__c) - reference to property.
  - Duration (Type: Number) - time needed to complete the work order. When the WorkTypeId field is defined Duration is duplicated field from the Work Type object.

- ServiceTerritoryId (Type: Lookup to Service Territory) - a reference to Service Territory object.
- ContactId (Type: Lookup to Contact) - a reference to the resident, in whose property work order needs to be completed.
- Subject (Type: Text) - subject, the aim of the work order.
- Description (Type: Text) - description of the work order.
- FSL__Scheduling_Priority__c (Type: Text) - represents the scheduling priority of the work order.

• **Work Type** - standard object, that defines the type of work; a template that can be applied to work order.
Fields:

- Name (Type: Text) - the name of work type.
- Issue Section (Type: Picklist) - section to which work type belongs.
- Description (Type: Text) - description of the work type, and steps to complete the job.
- EstimatedDuration (Type: Number) - duration of work type. It is measured in minutes or hours - defined in the DurationType field.
- DurationType (Type: Picklist) - the value in which the duration is measured. The picklist includes two types: Minutes and Hours. The default value is Hours.
- ShouldAutoCreateSvcAppt (Type: Checkbox) - defines if a service appointment is automatically created on work orders with such work type.

• **Service Appointment** - standard object, representing an appointment to complete work for a customer. It is actually the visits of technician to a resident to perform the work.
Fields:

- AppointmentNumber (Type: Autonumber) - auto-generated number that identifies the service appointment.
- ContactId (Type: Lookup to Contact) - reference to resident who need work ordre to be done in his/her property.
- ParentRecordType (Type: Text) - name of the parent record type.
- ParentRecordId (Type: Lookup) - reference to parent record. In this solution parent record is work order record.
- WorkTypeId (Type: Lookup to WorkType) - reference to the work type. Subject (Type: Text) - the subject of the SA.
- Description (Type: Text) - description of the SA.
- Property__c (Type: Lookup to Property__c) - reference to property.
- ServiceTerritoryId (Type: Lookup to Service Territory) - a reference to Service Territory object.
- Status (Type: Picklist) - status of the work order. Picklist includes the following values: None, Scheduled, In Progress, Completed, Cannot Complete, Canceled. The default value is None.
- Price__c (Type: Number) - the cost of work.

  - EarliestStartTime(Type: Datetime) - the date after which the appointment must be completed.
  - DueDate (Type: Datetime) - the date after which by appointment must be completed.
  - SchedEndTime (Type: Datetime) - scheduled time to start SA.
  - SchedEndTime (Type: Datetime) - scheduled time to end SA
  - ActualDuration (Type: Number) - the amount of time the service resource used to work.
  - ActualStartTime (Type: Datetime) - the date and time when the service resource starts the job.
  - ActualEndTime (Type: Datetime) - the date and time when service resource ends the job.

  There is a validation rule on the Service Appointment object, that restricts editing the Price__c field when the status is "Complete".

- **Service Territory** - standard object, representing a geographic location where work order need be performed.
  Fields:
  - Name (Type: Text) - territory name.
  - Address (Type: Address) - compound fields representing territory address.
  - Operating Hours (Type: Lookup to Operating Hours) - a reference to Operating Hours, a time when service appointment can be performed in the service territory.
- **Operating Hours** - standard object, representing working hours due to timezone.
  Fields:
  - Name (Type: Text) - the name of operating hours.
  - Description (Type: Text) - the description of operating hours.
  - Timezone (Type: Picklist) - the timezone of operating hours.
- **Time Slot** - standard object, representing working hours of a weekday.
  Fields:
  - DayOfWeek (Type: Picklist) - the day of the week of time slot.
  - Type (Type: Picklist) - the type of time slot: Normal or Extended.
  - StartTime (Type: Time) - start time of time slot.
  - EndTime (Type: Time) - end time of time slot.
  - OperatingHoursId (Type: Lookup to Operating Hours) - a reference to Operating Hours to which time slot is linked.
- **Service Resource** - standard object, representing a field technician.
  Fields:
  - Name (Type: Text) - the resource's name.

- RelatedRecordId (Type: Lookup to User) - reference to user representing service resource.
- ResourceType (Type: Picklist) - a type of service resource. Picklist consists of Technician, Dispatcher, Crew, Asset, Agent, and Planner values. In this solution all service resources are technicians.
- IsActive (Type: Checkbox) - defines if resource is active.

- **Service Territory Member** - standard junction object containing many-to-many relationships between Service Resource and Service Territory.
  Fields:
  - MemberNumber (Type: Autonumber) - an auto-generated number that identifies the service territory member.
  - ServiceResourceId (Lookup to Service Resource) - the reference to service resource, that is linked to service territory.
  - ServiceTerritoryId (Master-Detail to Service Territory) - the reference to service territory linked to service resource.
  - EffectiveStartDate (Type: Date) - date of joining the service resource to the service territory.
  - EffectiveEndDate (Type: Date) - the date until when the resource operates on the territory.
- **Assigned Resource** - standard junction object containing many-to-many relationships between Service Resource and Service Appointment. Represents service resource assigned to service appointment.
  Fields:
  - ServiceResourceId (Lookup to Service Resource) - the reference to the service resource, that is assigned to the service appointment.
  - ServiceAppointmentId (Type: Master-Detail to Service Appointment) - the reference to service appointment.
- **Shift** - standard object, representing the time when the service resource can perform SA.
  Fields:
  - StartDate (Type: Date) - the day from which service resource can do the job.
  - EndDate (Type: Date) - the day when service resource stops doing the job.
  - TotalDurationInHours (Type: Formula(Number)) - the amount of worked hours done in the timesheet.
  - ServiceResourceId (Type: Lookup to Service Resource) - reference to service resource, to which time sheet is assigned.
- **Skill** - standard object, representing skill.
  Fields:
  - Name (Type: Text) - the name of skill.
- **Service Resource Skill** - standard junction object, containing many-to-many relationships between Service Resource and Skill. Represent resource's skill.
  Fields:

- ServiceResourceId (Type: Master-Detail to Service Resource) - a reference to service resource, to whom skill is linked
- SkillId (Type: Lookup to Skill) - a reference to a skill linked to the resource.
- SkillLevel (Type: Number) - skill level of resource. Measured from 0 to 99.99.
- EffectiveStartDate (Type: Datetime) - date, when service resource gains skill.
- EffectiveEndDate (Type: Datetime) - date of skill expiration.

• **Skill Requirement** - standard junction object, between Work Type and Skill. Represent resource's skill.
Fields:

- RelatedRecordId (Type: Master-Detail to WorkType) - reference to the work type.
- SkillId (Type: Lookup to Skill) - reference to skill linked to resource.
- SkillLevel (Type: Number) - service resource's skill level.

5.3 Resident portal

The resident accesses his data through a Resident Portal made with Experience Cloud. Before creating any Salesforce product, it is necessary to choose a license for the users who will use it. There are several licenses, they differ in price, levels of access to facilities and functions. Basic licenses for registered users to access the portal are Customer Community, Customer Community Plus and Partner Community.

Customer Community - is created for B2C portals, where customers can create and view cases, and knowledge articles.

Customer Community Plus - includes all the features of the previous one, and in addition, you get access to Reports, Dashboards, and Files.

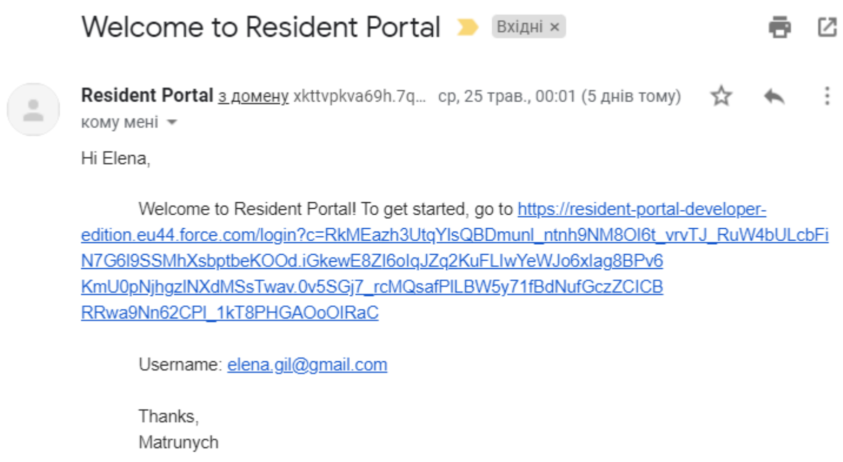
Partner Community - has the maximum access to your data, it allows to work with all objects that are involved in the sales process Account, Contact, Lead, and Opportunity. To meet our requirements Customer Community license is enough for users.

The creation of users will be handled by the administrator. Firstly, the administrator needs to create a contact of a user. The next step is to Enable Customer User for creating a user record. Then administrator needs to edit the user record, and assign an appropriate license, profile and permission set if needed. After saving the record, the user will receive a welcome email with his username and link for resetting his password.

First step when the developer creates a portal to which created residents will have access is a choice of the theme for the portal. Salesforce provides several out-of-the-box themes ready to install. For our case, Customer Account Portal theme is chosen. It is a portal which provides access to knowledge articles, viewing and editing accounts, cases and other Salesforce records. Then it is necessary to create the site name and URL address of this site.

After that, the developer needs to create the pages for the site and to fill the pages with components. The most important thing is to add access to the site for community users. This option is available in settings and it is needed to add a user profile to the portal members.

Let us consider what the resident's journey through the portal looks like. Once the administrator creates and saves the user record, the resident receives a welcome email. By following the link in the mail, the user has to create a password. Then our user can access Resident Portal.



Resident Portal з домену xkttvpkva69h.7q... ср, 25 трав., 00:01 (5 днів тому)
кому мені

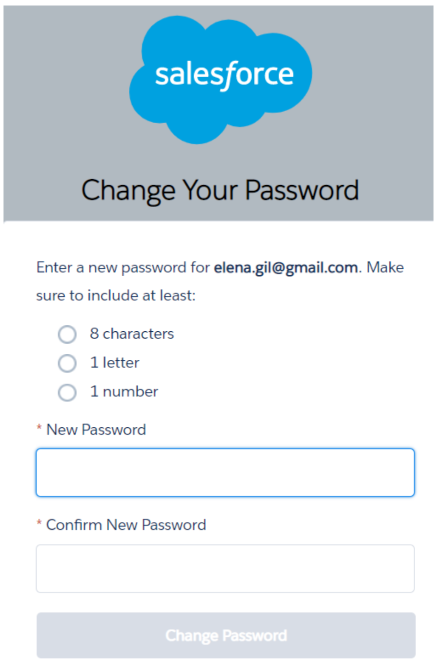
Hi Elena,

Welcome to Resident Portal! To get started, go to https://resident-portal-developer-edition.eu44.force.com/login?c=RkMEazh3UtqYlsQBDmunl_ntnh9NM8Ol6t_vrvTJ_RuW4bULcbFiN7G6l9SSMhXsbptbeKOOd.iGkewE8Zl6olqJZq2KuFLIwYeWJo6xlag8BPv6KmU0pNjhgzlNXdMSsTwav.0v5SGj7_rcMQsafPILBW5y71fBdNufGczZCICBRRwa9Nn62CPl_1kT8PHGAOoOIRaC

Username: elena.gil@gmail.com

Thanks,
Matrunych

FIGURE 5.3: Welcome email to resident



Enter a new password for **elena.gil@gmail.com**. Make sure to include at least:

- ○ 8 characters
- ○ 1 letter
- ○ 1 number

FIGURE 5.4: Change password screen

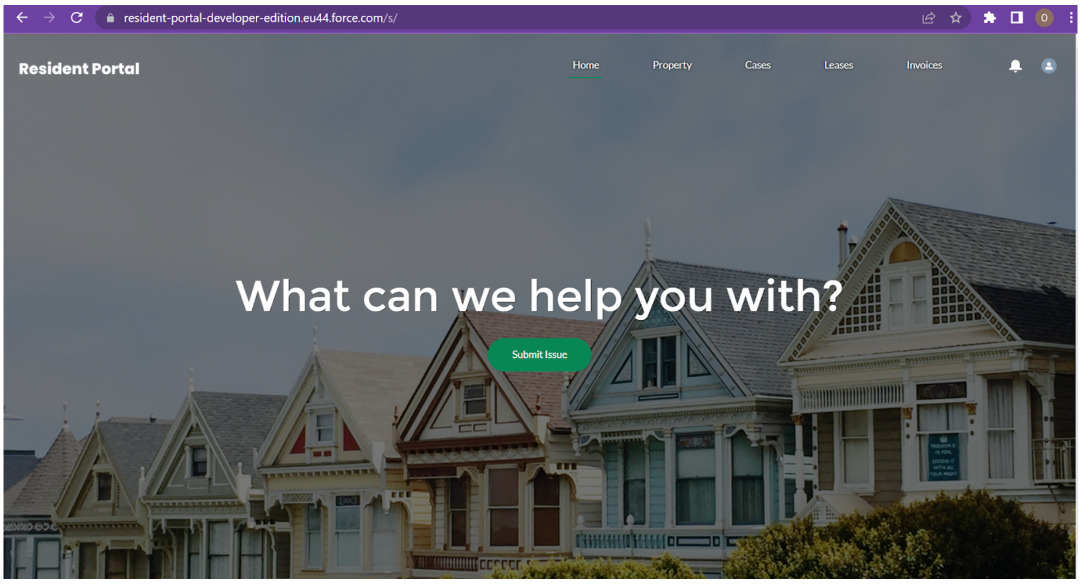
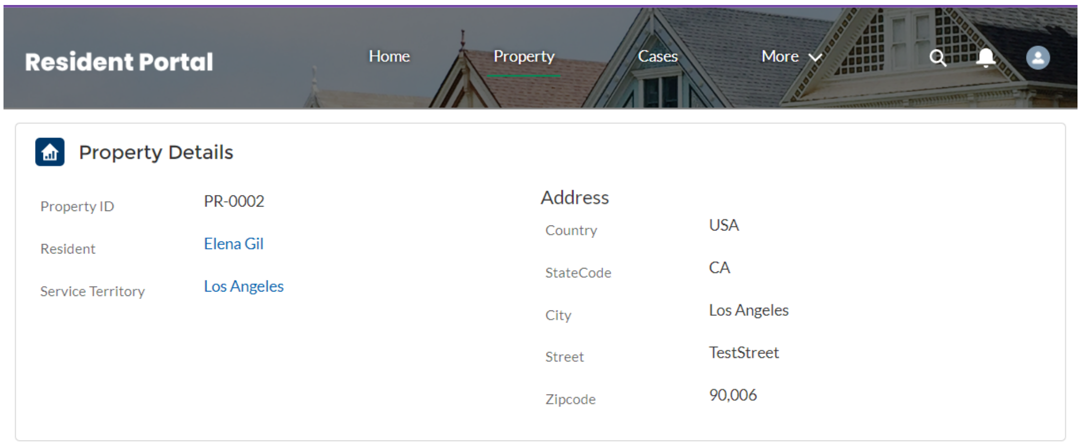

resident-portal-developer-edition.eu44.force.com/s/
Resident Portal
Home
Property
Cases
Leases
Invoices
What can we help you with?
Submit Issue

FIGURE 5.5: Resident portal Home page

On the Home page there is a button "Submit Issue" - it performs the main function of this portal, but it will be considered later in this chapter. On the Navigation bar, there are five menu items.
Pressing the "Property" tab the resident goes to the page with the information about his apartment.



Property Details

Property ID	PR-0002
Resident	Elena Gil
Service Territory	Los Angeles

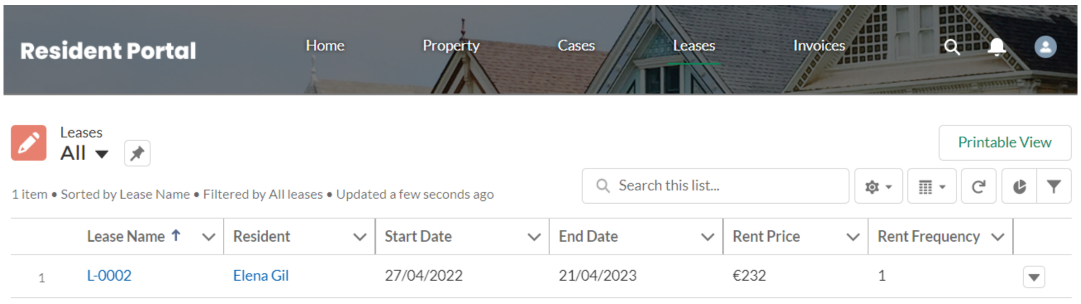
Address

Country	USA
StateCode	CA
City	Los Angeles
Street	TestStreet
Zipcode	90,006

FIGURE 5.6: Resident portal Property page

"Property Details" is a custom developed Lightning Web Component, which displays details of the property that has reference to the current resident, or rather has a lookup field to ContactId, which is linked to our resident's user. Current user Id is imported into js LWC controller and is passed to server-side Apex controller, which takes it and gets Property__c record, which has reference to the resident.

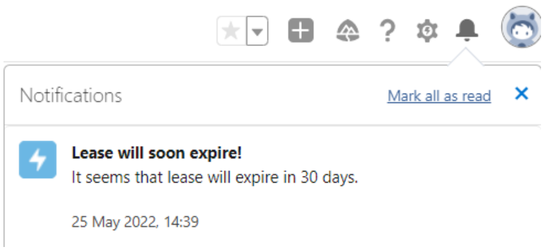
By clicking on the tab Leases on the navigation bar, the user will go to the page with a list of all his leases. The resident can also review the details of each lease. Leases are created by the administrator. 30 days before the date of lease expiration, the administrator receives a reminder that the leases will end soon. It is implemented by Apex scheduled job, that is run every day and checks if lease due date is coming.



1 item • Sorted by Lease Name • Filtered by All leases • Updated a few seconds ago

	Lease Name	Resident	Start Date	End Date	Rent Price	Rent Frequency
1	L-0002	Elena Gil	27/04/2022	21/04/2023	€232	1

FIGURE 5.7: Resident portal Leases page

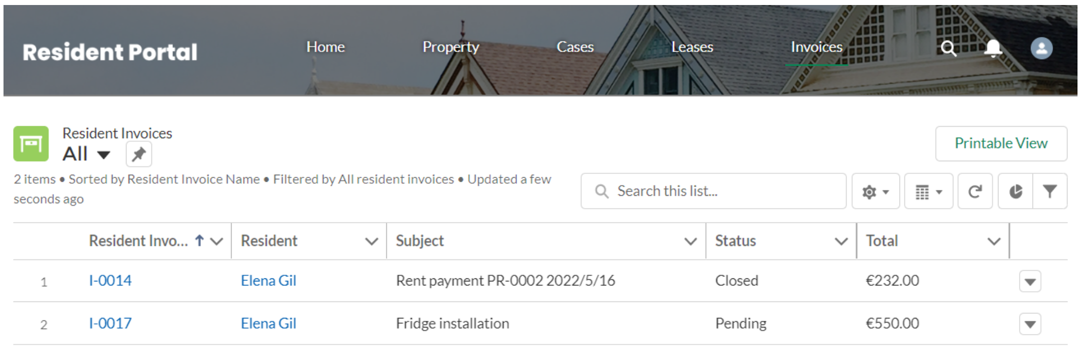


Lease will soon expire!
It seems that lease will expire in 30 days.

25 May 2022, 14:39

FIGURE 5.8: Leases expiration notification

The menu item Invoices redirects user to a page, where list of user's invoices are displayed.

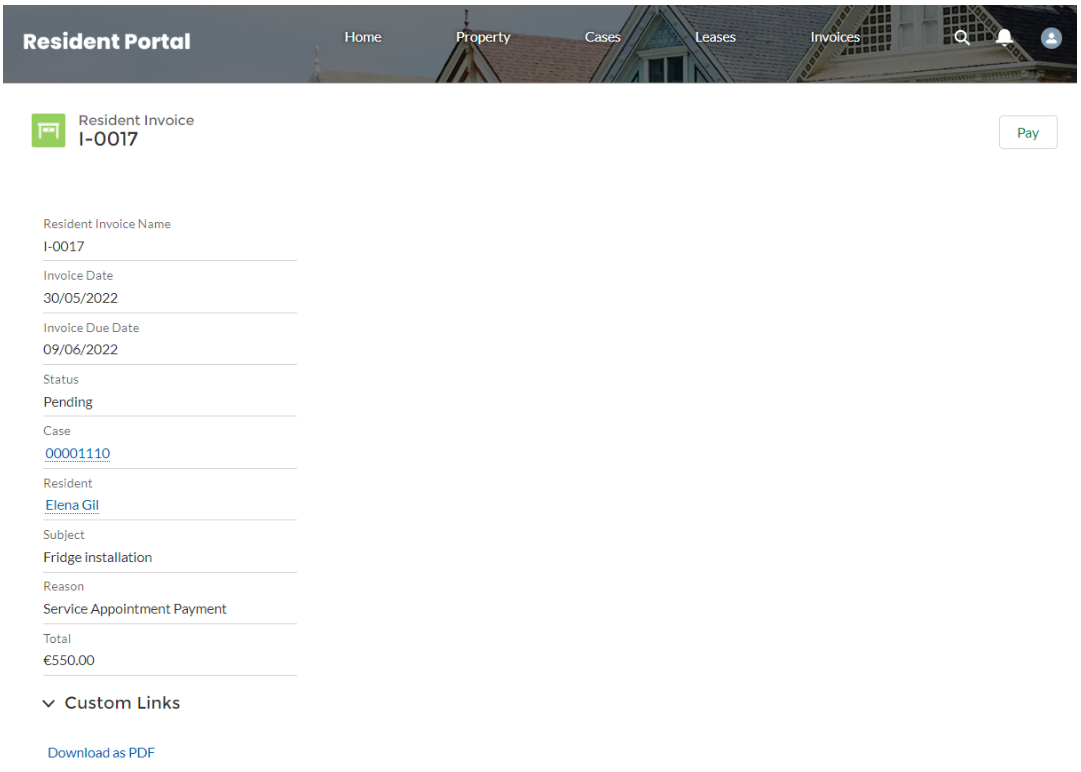


2 items • Sorted by Resident Invoice Name • Filtered by All resident invoices • Updated a few seconds ago

	Resident Invo...	Resident	Subject	Status	Total
1	I-0014	Elena Gil	Rent payment PR-0002 2022/5/16	Closed	€232.00
2	I-0017	Elena Gil	Fridge installation	Pending	€550.00

FIGURE 5.9: Resident portal Invoices page

Resident can see details of each of his invoice.



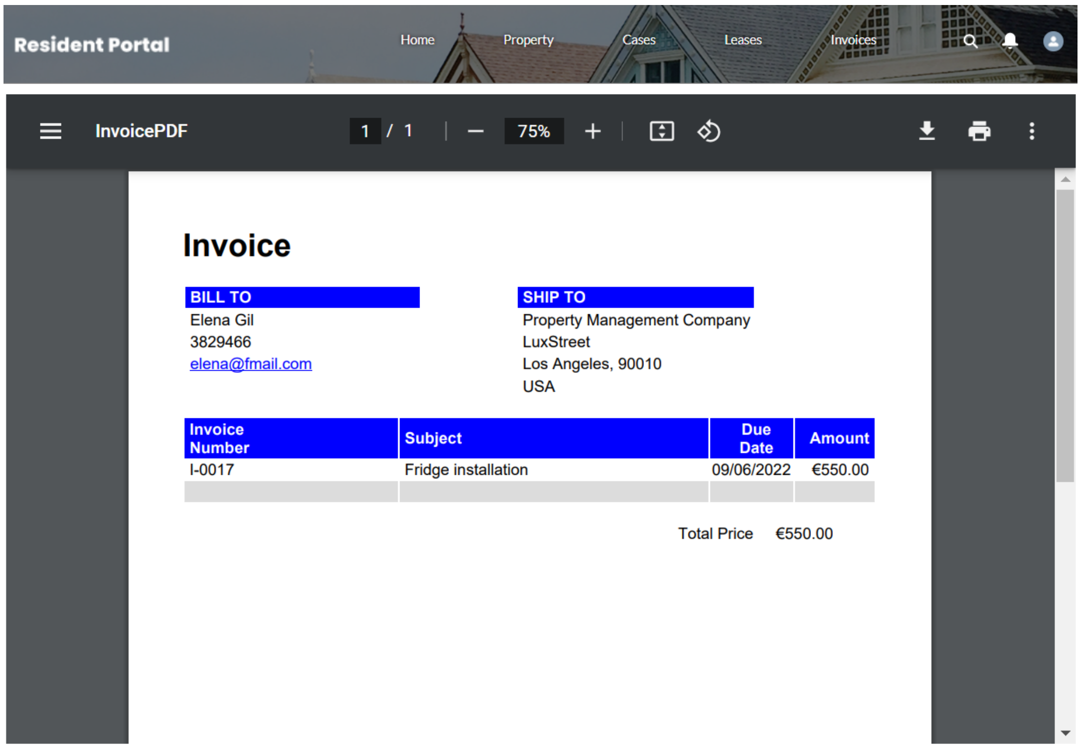
Resident Invoice
I-0017

Resident Invoice Name	I-0017
Invoice Date	30/05/2022
Invoice Due Date	09/06/2022
Status	Pending
Case	00001110
Resident	Elena Gil
Subject	Fridge installation
Reason	Service Appointment Payment
Total	€550.00

Download as PDF

FIGURE 5.10: Resident portal Invoice details page

In the Custom Links section, there is a link named "Download as PDF". By clicking on this link invoice details in PDF format will be rendered and the user can save them to his device. It is implemented by creating a Visualforce Page.



Invoice

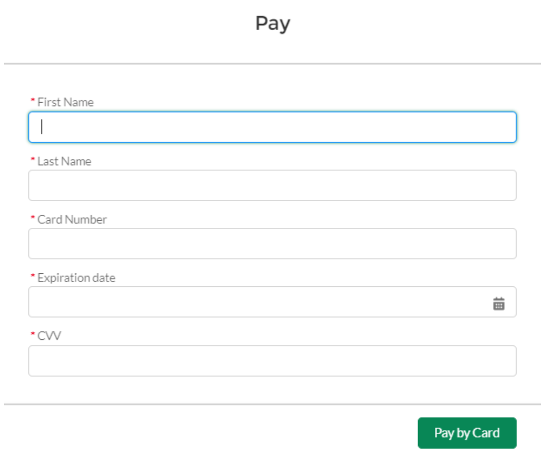
BILL TO	SHIP TO
Elena Gil	Property Management Company
3829466	LuxStreet
elena@fmail.com	Los Angeles, 90010
	USA

Invoice Number	Subject	Due Date	Amount
I-0017	Fridge installation	09/06/2022	€550.00

Total Price €550.00

FIGURE 5.11: Invoice in PDF format

On the top left, there is a "Pay" button. By clicking on it a modal box will be opened. It displays Screen Flow with fields that need to be filled to make payment. At this stage of the work, the payment functionality is missing. However, it will be a good decision to add integration with online banking in the future.


Pay

*First Name

*Last Name

*Card Number

*Expiration date

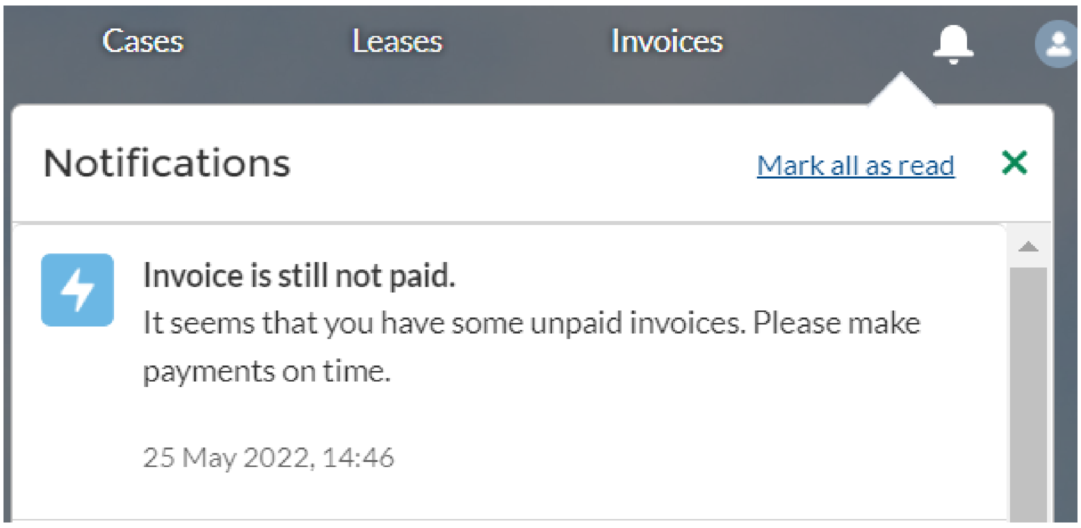
*CVV

FIGURE 5.12: Payment modal box

The resident Invoice creation is automated. Invoices for rent payments are created automatically every first day of the month. It is implemented by the Apex class that implements a special Schedulable interface needed to schedule an Apex class to run at regular intervals. Then this class is scheduled programmatically or via the user interface. Invoices for SA are created when SA status is updated to “Completed”. It is implemented by Apex Trigger. In our case, when SA is updated, the trigger is fired.

There is also a trigger on the Resident_Invoice__c object, that prevents the insertion of duplicated invoices.

Three days before the invoice due date, the resident receives a notification that he has to pay the invoice. It is implemented by Apex scheduled job.



Notifications

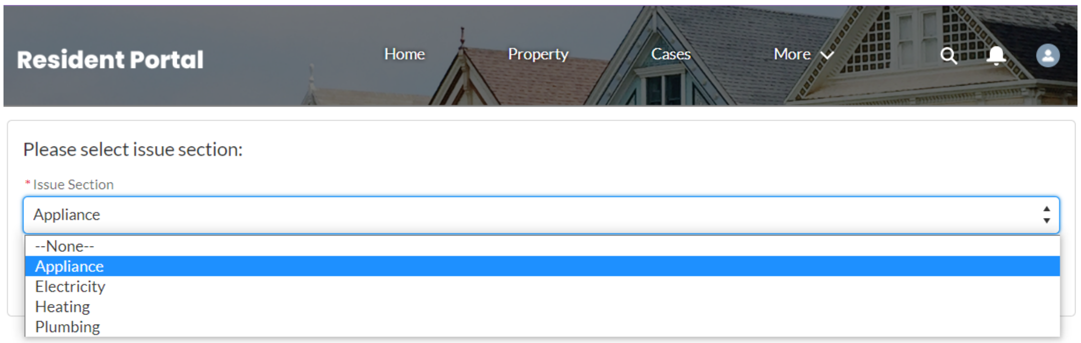
Invoice is still not paid.

It seems that you have some unpaid invoices. Please make payments on time.

25 May 2022, 14:46

FIGURE 5.13: Invoice due date coming notification

Now we turn to the “Submit Issue” button on the Home page, which was mentioned previously. By pressing this button our resident will be redirected to a page, where he can submit his issue. There is a designed Screen Flow component. Firstly, the resident should choose the issue section from a dropdown list and click "Next". After that, another dropdown list will appear. The resident can also add some description of his issue and finally submit.



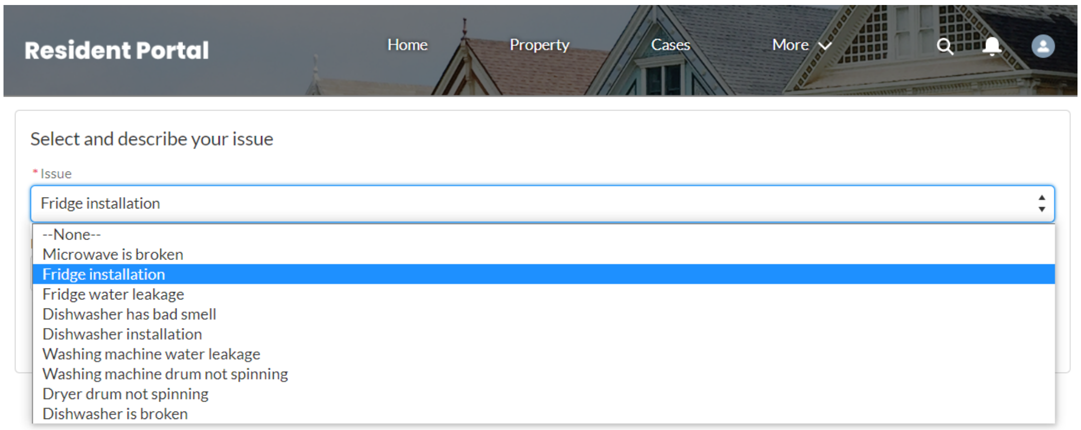
Please select issue section:

* Issue Section

Appliance

--None--
Appliance
Electricity
Heating
Plumbing

FIGURE 5.14: Submitting issue flow section selection



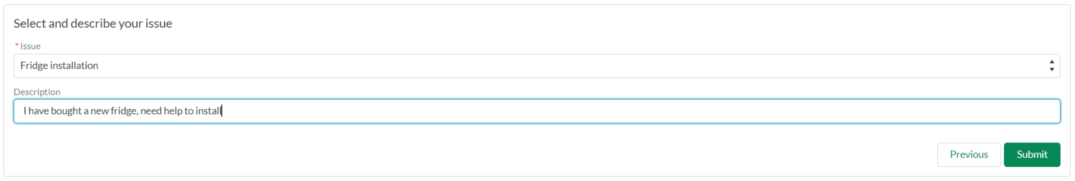
Select and describe your issue

* Issue

Fridge installation

--None--
Microwave is broken
Fridge installation
Fridge water leakage
Dishwasher has bad smell
Dishwasher installation
Washing machine water leakage
Washing machine drum not spinning
Dryer drum not spinning
Dishwasher is broken

FIGURE 5.15: Submitting issue flow issue selection



Select and describe your issue

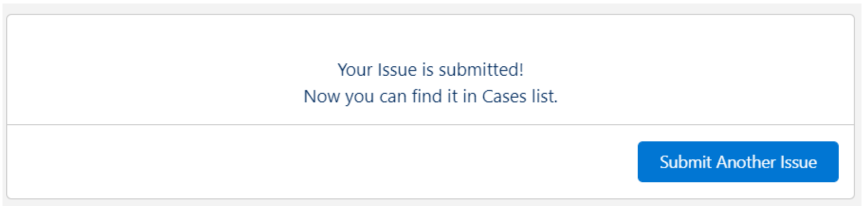
* Issue

Fridge installation

Description

I have bought a new fridge, need help to install

FIGURE 5.16: Submitting issue flow description adding



Your Issue is submitted!
Now you can find it in Cases list.

FIGURE 5.17: Submitting issue flow final screen

After submission the resident is redirected to the beginning of the flow. When the issue is submitted, he find a new case in the cases list. This is a record where the user will be able to view the status of his issue, and after the specialised work, the price will appear on the case, and then a new invoice will be created.

Such a procedure can be observed by the resident himself, but in reality, it is a bit more complicated. The fact is when the resident submits an issue, three records of different objects are created at once. Let's look at the Screen Flow implementation.

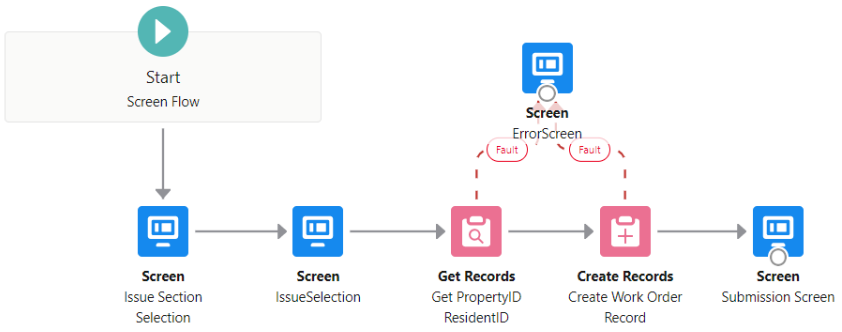


FIGURE 5.18: Work Order Screen Flow

The first record that is created after submission is the Work Order record. **Issue Section Selection** screen component contains input picklist field. Picklist values are gotten from Work Type Issue_Section__c picklist values.

After selecting the "Issue Section", the **Issue Selection** screen appears. It has two input fields: "Issue" picklist, whose choice set consists of Work Type names filtered by issue section saved from the previous screen and text field "Description" for entering some description. When the user selects issue, Work Type Id and Name are saved to variables, they are needed in the next steps.

The next flow item is the **Get Records** element. It gets Property__c records, which meet the condition that Resident__c field on Property__c object equals to user's ContactId. When receiving records, we save Id, Resident__c and Service_Territory__c fields to variables.

Create Records element creates Work Order records and set values of previously saved variables to the appropriate fields.

Submission Screen displays text confirming the submitting the issue.

Error Screen displays an error message if any exception occurs.

Once the Work Order record is created, another custom flow is run. It is a record-triggered flow, that is fired when the Work Order record is created.

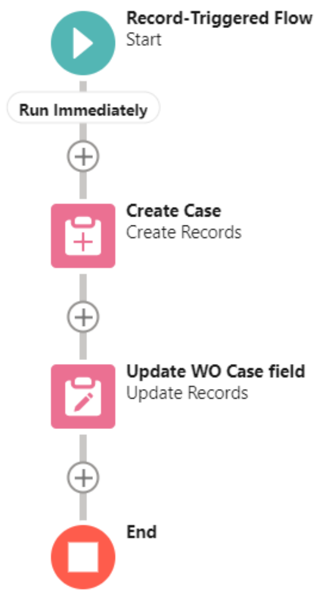
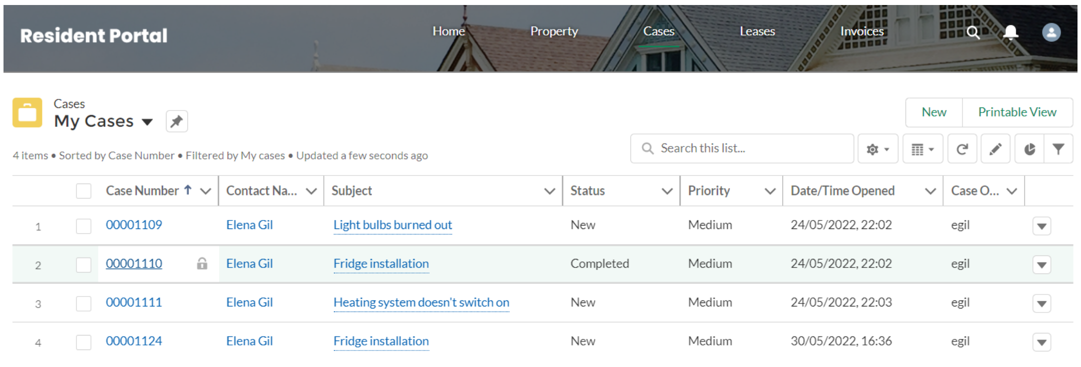


FIGURE 5.19: Case creation Flow

It has **Create Records** element, which creates a Case record, exactly the one that will appear in the resident's case list. Fields ContactId, Property__c and Description will be duplicates of the fields of the same name from the Work Order. When Case is inserted into the database, its Id will be assigned to the Work Order record that triggered the flow. To make this there is an **Update Records** element, that identifies records for updating and set the field values.

Also after Work Order creation, an SA record is created. It is implemented by FSL automatization. We need to duplicate custom fields Property__c and Case__c from Work Order to SA - it is implemented by after insert/update Apex Trigger.

When the user has submitted the issue, he can find it in the list of his cases by going to the tab "Case" in the navigation menu.

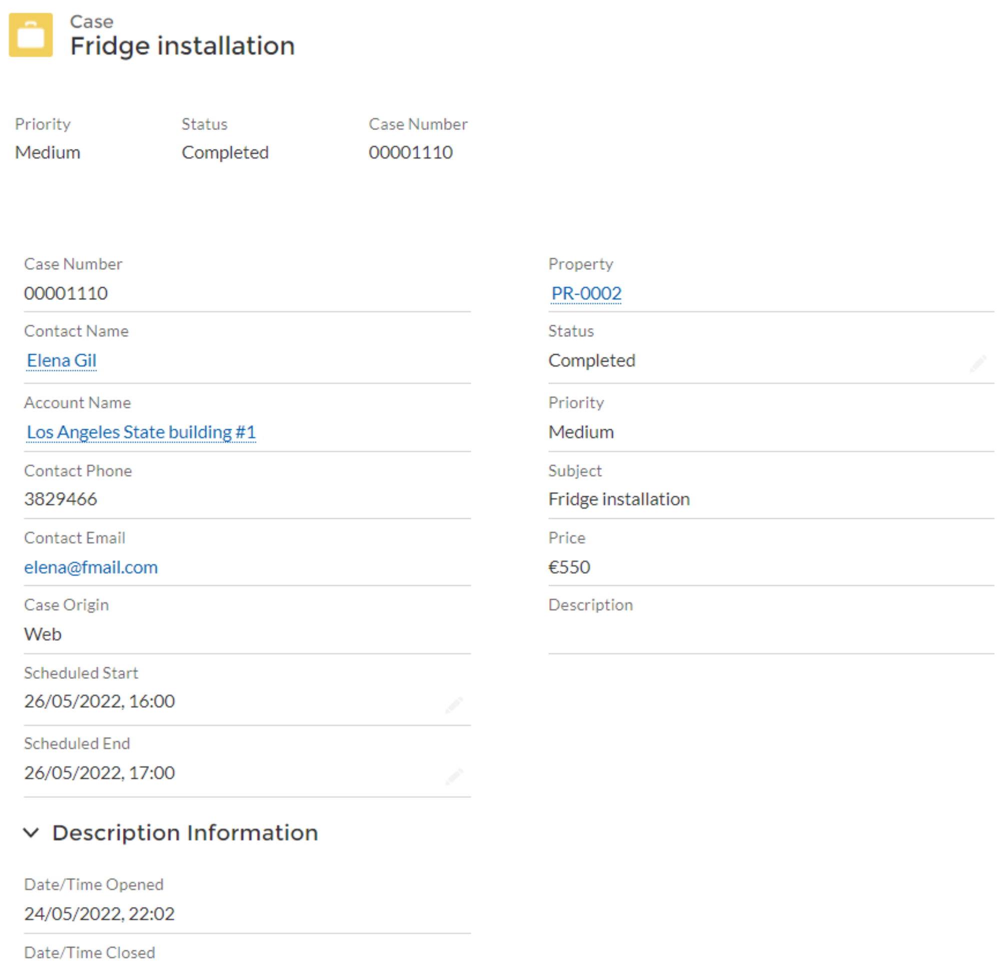


4 items • Sorted by Case Number • Filtered by My cases • Updated a few seconds ago

	Case Number	Contact Na...	Subject	Status	Priority	Date/Time Opened	Case O...
1	00001109	Elena Gil	Light bulbs burned out	New	Medium	24/05/2022, 22:02	egil
2	00001110	Elena Gil	Fridge installation	Completed	Medium	24/05/2022, 22:02	egil
3	00001111	Elena Gil	Heating system doesn't switch on	New	Medium	24/05/2022, 22:03	egil
4	00001124	Elena Gil	Fridge installation	New	Medium	30/05/2022, 16:36	egil

FIGURE 5.20: Resident portal Cases page

On the Case detail page, the resident can see the price of the work and the scheduled time, when the technician comes.


Case
Fridge installation

Priority	Status	Case Number
Medium	Completed	00001110

Case Number: 00001110	Property: PR-0002
Contact Name: Elena Gil	Status: Completed
Account Name: Los Angeles State building #1	Priority: Medium
Contact Phone: 3829466	Subject: Fridge installation
Contact Email: elena@fmail.com	Price: €550
Case Origin: Web	Description:
Scheduled Start: 26/05/2022, 16:00	
Scheduled End: 26/05/2022, 17:00	

Description Information

Date/Time Opened	24/05/2022, 22:02
Date/Time Closed	

FIGURE 5.21: Resident portal Case detail page

5.4 Property Management app

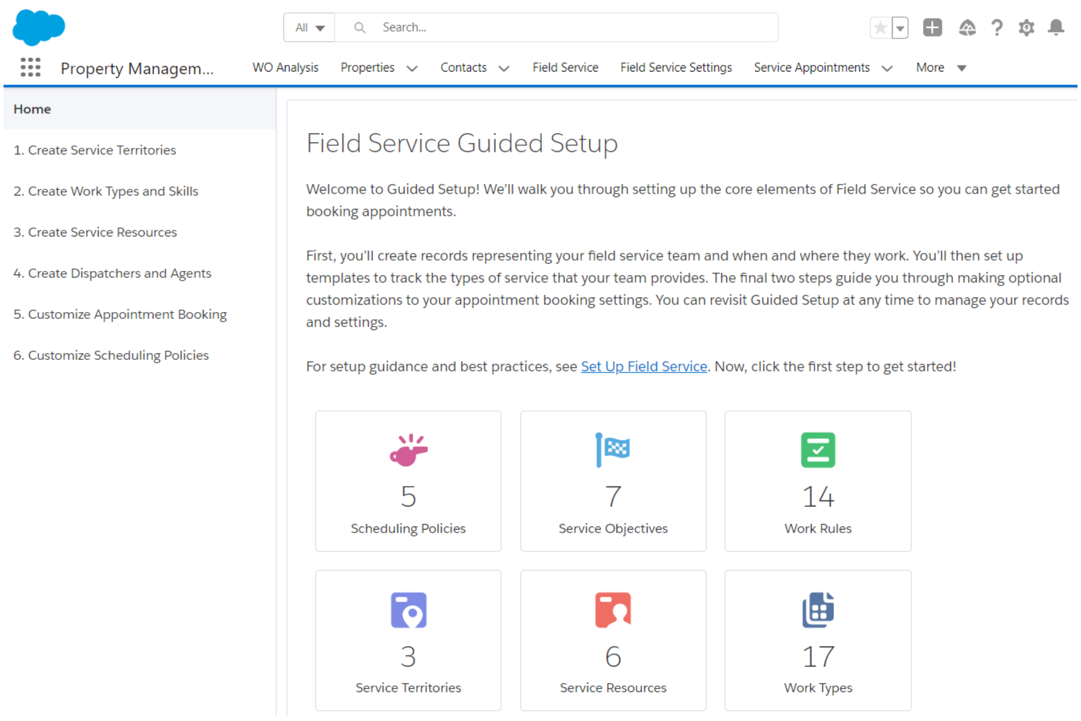
Once the service appointment has already been created, it is necessary to assign a service resource - field technician who will come to the resident's house and do the job. This is the next part of our solution related to the functionality of the FSL.
This part of proposed system is Property Management app - a special app for the administrator. So let's look at how everything is set up.

5.4.1 Data Configuration

To start working with FSL, it is required to enable Field Service Lightning on the org and install FSL Managed Package. After that, the administrator can start his configuration work. There is Field Service Settings tab in the Field Service Admin app. This is where all the necessary tools for the administrator are.

Initially, the administrator assigns required permission sets to users according to their role in a system: administrator or field technician. A permission set is a set of settings and permissions that grant users access to different instruments and functions.

After granting permissions, including for himself, the administrator can proceed to data configuration. There are two ways he can do this: using the Guided Setup, which is one of the features get after installing the managed package or the classic way through the standard Salesforce user interface.


All
Search...
Property Managem...
WO Analysis
Properties
Contacts
Field Service
Field Service Settings
Service Appointments
More
Home
1. Create Service Territories
2. Create Work Types and Skills
3. Create Service Resources
4. Create Dispatchers and Agents
5. Customize Appointment Booking
6. Customize Scheduling Policies

Field Service Guided Setup

Welcome to Guided Setup! We'll walk you through setting up the core elements of Field Service so you can get started booking appointments.

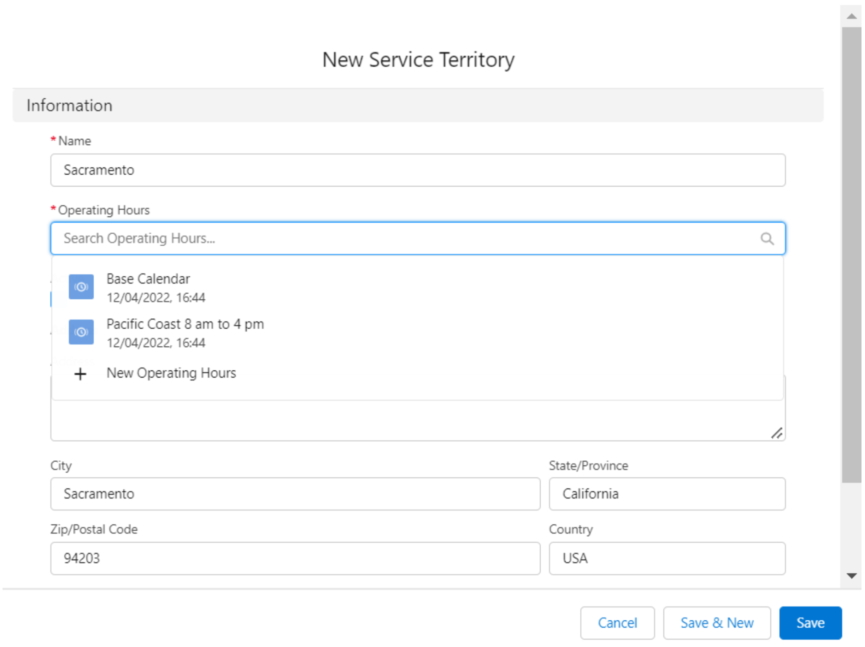
First, you'll create records representing your field service team and when and where they work. You'll then set up templates to track the types of service that your team provides. The final two steps guide you through making optional customizations to your appointment booking settings. You can revisit Guided Setup at any time to manage your records and settings.

For setup guidance and best practices, see Set Up Field Service. Now, click the first step to get started!

5 Scheduling Policies
7 Service Objectives
14 Work Rules
3 Service Territories
6 Service Resources
17 Work Types

FIGURE 5.22: Field Service Guided Setup in Property Management app

We start configuration with the second approach. Firstly, we will create Service Territories and Operating Hours. When filling fields of Service Territory, it is needed to choose Operating Hours, we will select Base Calendar.


New Service Territory

Information

*Name	Sacramento
*Operating Hours	Search Operating Hours...

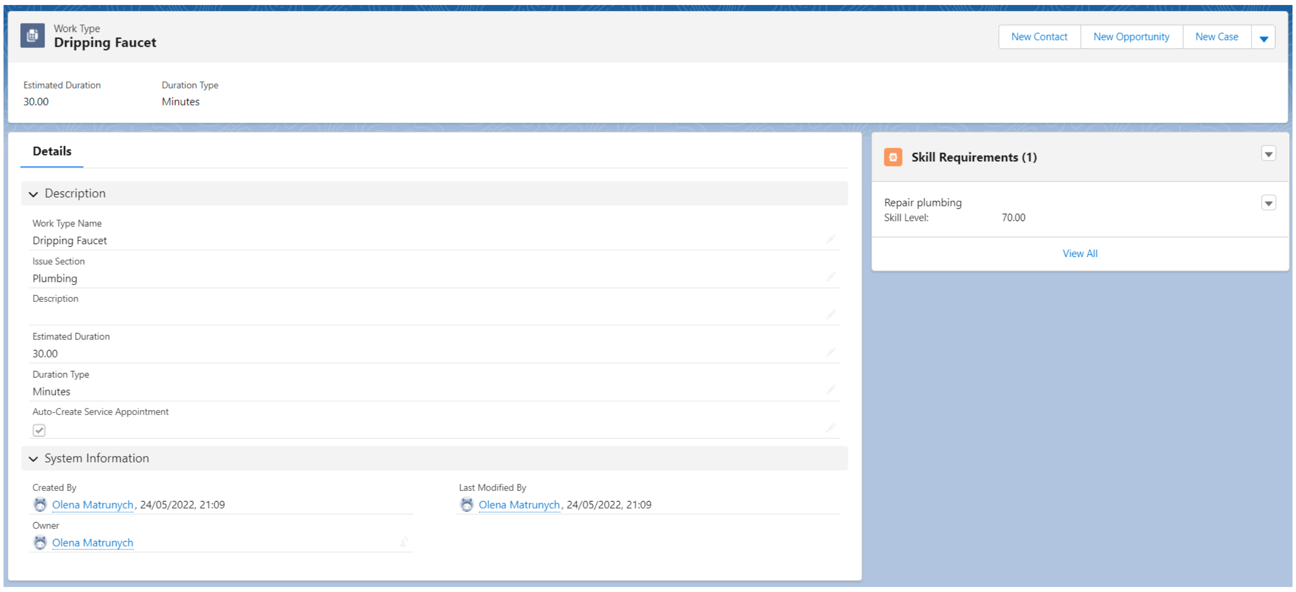
Base Calendar
12/04/2022, 16:44
Pacific Coast 8 am to 4 pm
12/04/2022, 16:44
+ New Operating Hours

City	Sacramento
State/Province	California
Zip/Postal Code	94203
Country	USA

FIGURE 5.23: Service Territory creation screen

Base Calendar is Operating Hours that is exactly necessary for Sacramento territory, because its timezone corresponds to city's one.

Now let's move on to Work Types. On the record details page of Work Type there is a related list Skill Requirement that display records from objects of the same name - required skill for performing this job.


Work Type
Dripping Faucet

Estimated Duration
30.00
Duration Type
Minutes

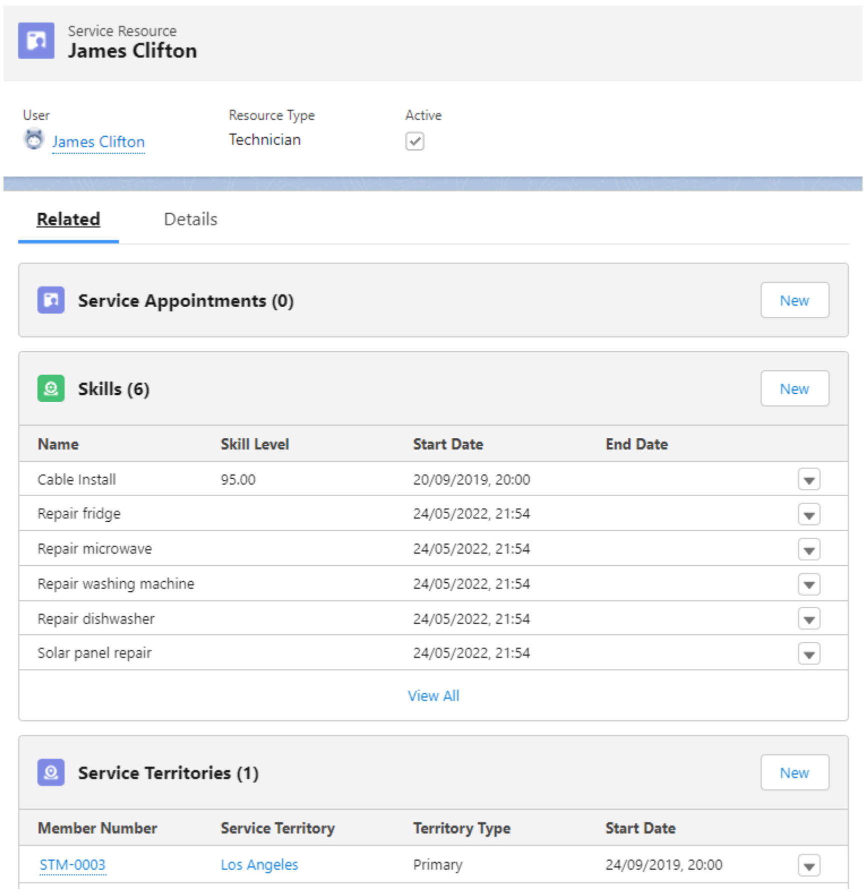
Description
Work Type Name
Dripping Faucet
Issue Section
Plumbing
Description
Estimated Duration
30.00
Duration Type
Minutes
Auto-Create Service Appointment
✓

System Information
Created By
Olena Matrunych, 24/05/2022, 21:09
Last Modified By
Olena Matrunych, 24/05/2022, 21:09
Owner
Olena Matrunych

Skill Requirements (1)
Repair plumbing
Skill Level: 70.00

FIGURE 5.24: Work Type record page

The next step is to create service resources. Firstly, it is needed to create users for our technicians and then service resource record. Going to the Related tab on Service Resource record page, we can see and add different records that are linked to technician, for example skills or service territories.


Service Resource
James Clifton

User	Resource Type	Active
James Clifton	Technician	✓

Related Details

Service Appointments (0) New

Skills (6) New

Name	Skill Level	Start Date	End Date
Cable Install	95.00	20/09/2019, 20:00	
Repair fridge		24/05/2022, 21:54	
Repair microwave		24/05/2022, 21:54	
Repair washing machine		24/05/2022, 21:54	
Repair dishwasher		24/05/2022, 21:54	
Solar panel repair		24/05/2022, 21:54	

View All

Service Territories (1) New

Member Number	Service Territory	Territory Type	Start Date
STM-0003	Los Angeles	Primary	24/09/2019, 20:00

FIGURE 5.25: Service Resource detail page

5.4.2 Scheduling policy

Scheduling policy is a set of rules and objectives that guides the schedule optimizer in its decisions. These policies define the principles on which SA will be planned.[8]

Scheduling policy consist of Work Rules and Service objectives. FSL has four standard scheduling policies, but we will create our own called “Custom Policy”.

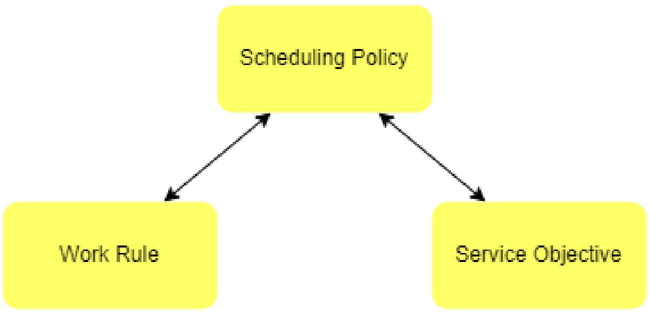


FIGURE 5.26: Scheduling policy diagram

After creating policy, it is needed to add existing or create new work rules. Work rules define service resources for SA, who comply with the rules.

FSL includes numerous standard work rules and different work rule types. We will operate with standard rules and here are the ones we will add to the policy:

- Active Resources: considers only active resources.

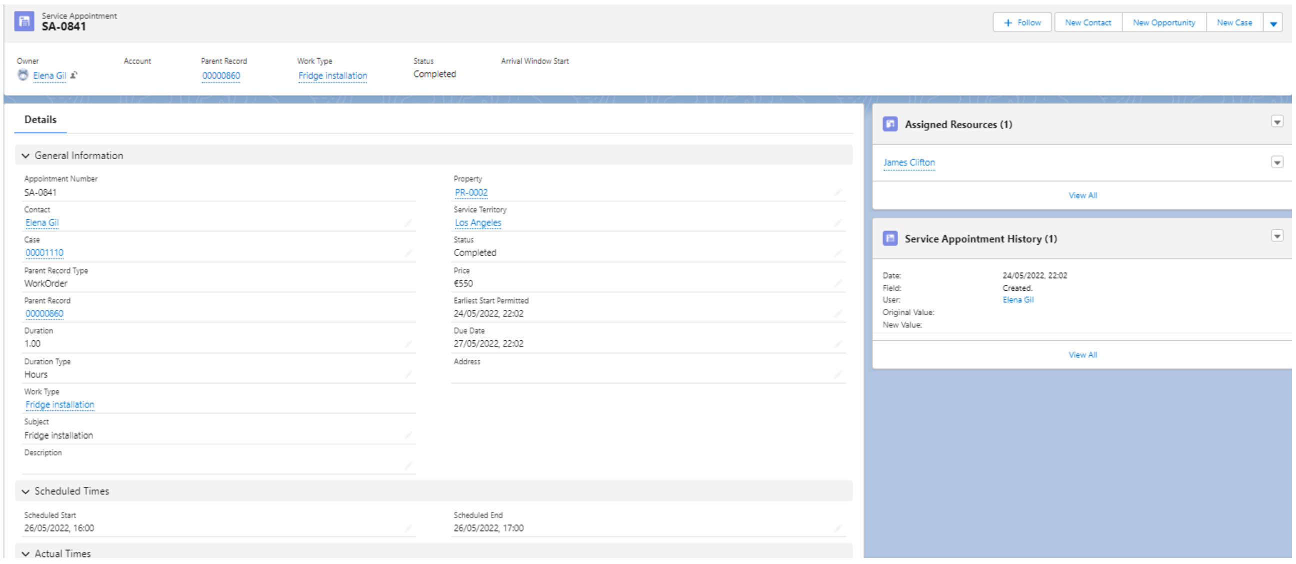
- Earliest Start Permitted: guarantees that SA is scheduled no earlier Earliest Start Permitted.
- Due Date: guarantees that SA is done by the Due Date.
- Scheduled Start: guarantees that service resource starts work after the Arrival Window Start.
- Scheduled End: guarantees that service resource starts work before the Arrival Window End.
- Match Skills: guarantees that service resource has needed skills with appropriate skill level.
- Match Territory: considers service resources assigned to service territory.

Now let's consider which service objectives assign to policy. Service objectives represent the policy's purpose. FSL also has standard service objectives, for our case two are used:

- **ASAP** - Measures the ability to schedule an appointment as soon as possible.[8]
- **Skill Level** - Measures assigned resources' adherence to a work order's skill requirements.[8]

5.4.3 Service appointment management

When the resident submits the issue, a case, work order and service appointment are created. After that the service appointment are managed by the administrator and the field technician.



Service Appointment
SA-0841

Owner	Account	Parent Record	Work Type	Status	Arrival Window Start
Elena Gil		00000860	Fridge installation	Completed	

Details

General Information

Appointment Number: SA-0841	Property: PR-0002
Contact: Elena Gil	Service Territory: Los Angeles
Case: 00001110	Status: Completed
Parent Record Type: WorkOrder	Price: €550
Parent Record: 00000860	Earliest Start Permitted: 24/05/2022, 22:02
Duration: 1.00	Due Date: 27/05/2022, 22:02
Duration Type: Hours	Address:
Work Type: Fridge installation	
Subject: Fridge installation	
Description:	

Scheduled Times

Scheduled Start: 26/05/2022, 16:00	Scheduled End: 26/05/2022, 17:00

Actual Times

Assigned Resources (1)

James Clifton

View All

Service Appointment History (1)

Date:	24/05/2022, 22:02
Field:	Created.
User:	Elena Gil
Original Value:	
New Value:	

View All

FIGURE 5.27: Service Appointment detail page

There is a custom developed Apex Trigger on the Service Appointment object. This trigger controls the fields of the same name on the case, which is tied to the SA, precisely Price, Scheduled Start, Scheduled End. There is another custom automation on SA - auto-scheduling. There is a Scheduled Apex job running every 2 hours and it assigns service resource to SA due to scheduling policy.

FSL package provides another feature Dispatcher Console - workspace for the dispatcher, and in our case for the administrator. There is an opportunity to overview the calendar with SA, and to schedule and reschedule SA.

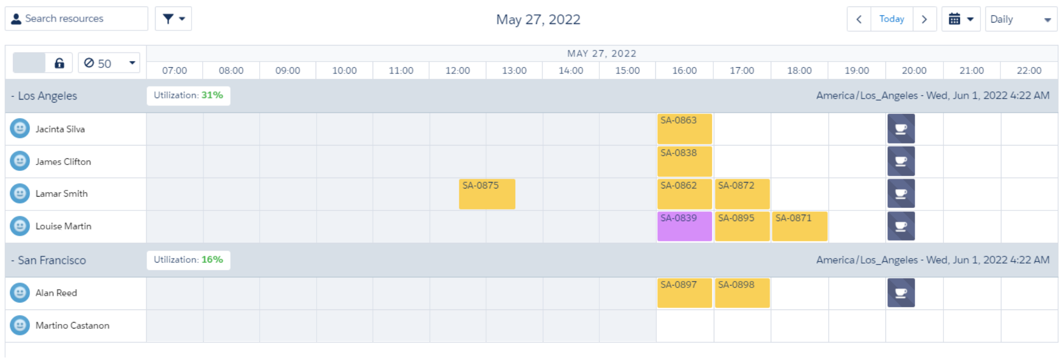


FIGURE 5.28: Dispatcher Console UI

When the support service is created and the service resource is automatically detected, the **field technician** goes to resident's property and starts working. When he starts work, he changes the status of SA to "In progress". At the end he sets the price for the work and changes the status to "Completed" or "Cannot complete".

5.4.4 Work Order Analysis

Another feature for the administrator has been developed. It provides an opportunity to analyze work orders for various aspects. On the "WO Analysis" tab, there is a dashboard having two reports with charts.

The chart of the first report shows the distribution of work orders by service territories. It shows in which area how many problems the residents had.

The chart of the second report shows the distribution of work orders by the section to which the problem belongs. For example, how many breakdowns with appliances or electricity occurred.

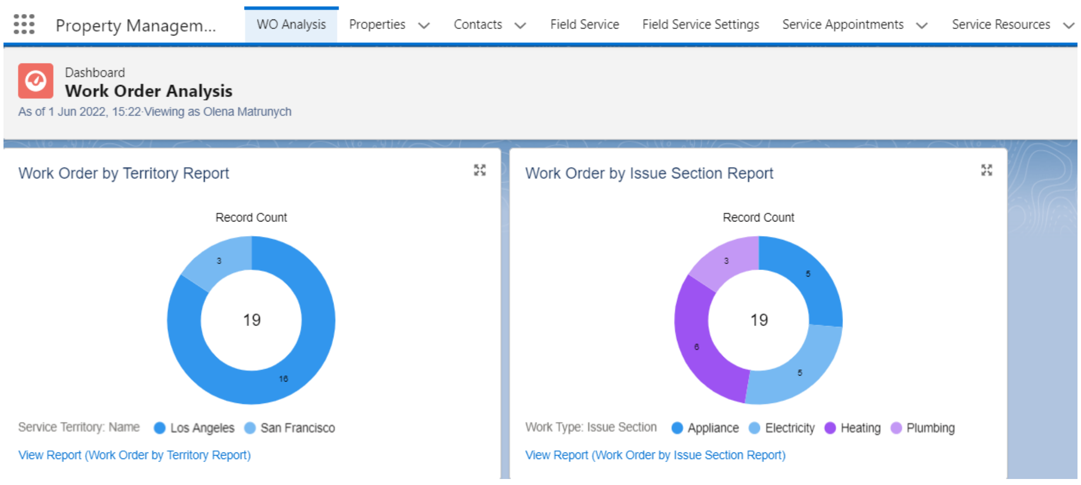


FIGURE 5.29: Work Order Analysis Dashboard

Dashboards in Salesforce can be created with declarative tools. However, there is a certain algorithm of actions behind it. Before creating and dashboard and adding it to the page and assign to the app, the developer needs to create report types and

reports.
The report type determines which records and fields appear in the report.[13] They are of two types: standard and custom. For our dashboards two report types are necessary: standard for "Work Order by Service Territory Report", because it needs fields only from Work Order object, and custom for "Work Order by Issue Section Report", because we need to take field value from the parent object.

The second step is to create the reports. A report is a list of records that meet the defined criteria.[13] In the report it is defined by which columns records are grouped, and then according to these configurations the chart is created.
Then you need to create a dashboard and add reports. The last thing is to add a dashboard to the page, as well as assign access to it to the users.

Chapter 6

Conclusion

6.1 Summary

The purpose of my work was to develop a prototype of a service system for residents. To achieve this, an analysis of the real estate rental market was conducted to determine the extent to which such a system may be needed. As well as an analysis of similar existing solutions to determine which functions should be in the system and which are missing. The Salesforce platform was chosen for development. Although the platform offers many out-of-the-box products, they need customization to meet the requirements of functionality or custom development from scratch.

The goal of this work was accomplished. The proposed solution consists of a resident portal, where the user can find information about their activities in rented accommodation and an administrative app, for the management of residents and their service requests.

The introduction of such a system for property management companies will bring numerous business benefits such as:

- The resident portal enables efficient case resolutions. As soon as the resident submits an issue, it is immediately processed by the system.
- Automated service appointment scheduling increases the field efficiency of technicians.
- Significant savings of resources and time. Due to the fact that the submission of maintenance requests will be automated, the need for call centre managers will disappear.
- Personalized customer journey results in more happy customers.

At a time when the world is experiencing an era of digital transformation, real estate industry has to be modernized and provide the best automated resident services.
It is time to forget about phone calls to order services and digitalize the service sector.

6.2 Future work

There are multiple additional functions that can be added to the proposed solution. For example:

- The ability to pay for rent and CA in the system;
- Tracking the resident's expenses;

- Wide variety of reports and dashboards types for comprehensive analysis of work orders. Therefore, the real estate agent will know what potential prospective residents may face.

Bibliography

[1] Harvard University Joint Center. *America's Rental Housing 2022*. 2022. URL: https://www.jchs.harvard.edu/sites/default/files/reports/files/Harvard_JCHS_Americas_Rental_Housing_2022.pdf.

[2] Deloitte. *Real Estate Predictions 2021*. URL: https://www2.deloitte.com/content/dam/Deloitte/global/Documents/Infrastructure\%20\%20Capital\%20Projects/gx-real-estate-predictions-2021.pdf.

[3] IBM Cloud Education. *IaaS versus PaaS versus SaaS*. 2021. URL: https://www.ibm.com/cloud/learn/iaas-paas-saas.

[4] Entrata. *Renters on the Move:* URL: https://info.entrata.com/newsletters/whitepaper/2021/Renters_on_the_Move.pdf.

[5] Fannie Mae. *Consumer Pessimism Regarding Direction of Mortgage Rates Hits New Survey High*. URL: https://www.fanniemae.com/newsroom/fannie-mae-news/consumer-pessimism-regarding-direction-mortgage-rates-hits-new-survey-high.

[6] Finances Online. *36 Vital Field Service Management Statistics: 2021 Analysis of Data Market Share*. URL: https://financesonline.com/field-service-management-statistics/.

[7] Salesforce. *Apex Code Overview*. URL: https://help.salesforce.com/s/articleView?id=sf.code_about.htm&type=5.

[8] Salesforce. *Complete Guide to Field Service*. URL: https://resources.docs.salesforce.com/latest/latest/en-us/sfdc/pdf/salesforce_field_service_implementation_guide.pdf.

[9] Salesforce. *Execute SOQL and SOSL Queries*. URL: https://trailhead.salesforce.com/content/learn/modules/developer_console/developer_console_queries.

[10] Salesforce. *Execution Governors and Limits*. URL: https://developer.salesforce.com/docs/atlas.en-us.apexcode.meta/apexcode/apex_gov_limits.htm.

[11] Salesforce. *Field Service Pricing*. URL: https://www.salesforce.com/products/field-service/pricing/s.

[12] Salesforce. *Get Started with Field Service for Salesforce*. URL: https://trailhead.salesforce.com/en/content/learn/modules/field_service_basics/field_service_basics_intro?trail_id=field_service&trailmix_creator_id=egreenwald1&trailmix_slug=fsl-modules.

[13] Salesforce. *Introduction to Reports and Dashboards*. URL: https://trailhead.salesforce.com/content/learn/modules/reports_dashboards/reports_dashboards_overview.

[14] Salesforce. *Salesforce is 1 in Global CRM Market Share*. URL: https://www.salesforce.com/news/stories/salesforce-is-1-in-global-crm-market-share/.

[15] SAP. *What is field service management?* URL: https://www.sap.com/insights/what-is-field-service-management.html.

[16] Jenny Schuetz Sarah Crump. *US rental housing markets are diverse, decentralized, and financially stressed*. URL: https://www.brookings.edu/essay/us-rental-housing-markets/.

[17] Eugene Klymenko Viktoriya Tsytsak. *Field Service Lightning for field operations modernization and automation*. URL: https://info.softserveinc.com/hubfs/2020/ad-hoc/field-operations-modernization-salesforce.pdf?utm_campaign=OFR\%20NRTR\%20WW\%20AUG2020\%20HZ\%20Salesforce\%20Campaign&utm_medium=email&_hsmi=93519984&_hsenc=p2ANqtz-99RbgseR-etn26TzkEdPyATdyJLIk2_KmCFuwuySEGYSYk8lYCSCvTjtTGkCMAiuvkj-4mUzhud9koRpycMDnn4UefuetcNVSfD6hk2jhdXQk7oKA&utm_content=93519984&utm_source=hs_automation.