

**ПЛОВДИВСКИ УНИВЕРСИТЕТ „ПАИСИЙ ХИЛЕНДАРСКИ“  
ФАКУЛТЕТ ПО МАТЕМАТИКА И ИНФОРМАТИКА**

**КАТЕДРА „КОМПЮТЪРНИ СИСТЕМИ“**

# **ДИПЛОМНА РАБОТА**

на тема

## **Облачна система за управление на авиокомпаниии**

Дипломант:

**Лилия Дочкова Теодошева**

Специалност: Софтуерни технологии и дизайн

Факултетен номер: 2001681041

Научен ръководител: **доц. д-р Владимир Вълканов**

Пловдив, 2024 г.

**PLOVDIV UNIVERSITY “PAISHI HILENDARSKI”  
FACULTY OF MATHEMATICS AND INFORMATICS**

**DEPARTMENT OF COMPUTER SYSTEMS**

# **THESIS**

on the topic of

## **Cloud-Based Management System for Commercial Airlines**

by:

**Lilia Dochkova Teodosheva**

Major: Software Technologies and Design

Faculty Number: 2001681041

**Scientific Advisor: Assoc. Prof. Vladimir Valkanov**

Plovdiv, 2024 г.

## Резюме

В днешния бързо развиващ се и технологично ориентиран свят, компаниите се нуждаят от напреднали технологии, за да бъдат конкурентноспособни. Бизнесите се стремят към това да автоматизират всеки възможен процес в компанията си, за да може техните служители да се фокусират върху работата, която не може да бъде изпълнена от един софтуер.

Без използването на каквито и да е технологии, управлението на един бизнес би било истинско предизвикателство. А, когато става въпрос за комерсиални авиокомпаниии, които има комплексни операции, засягащи голямо количество данни, това е близо до невъзможно.

С желанието на компаниите да дигитализират повече от своите бизнес процеси, те стигат до заключението, че може би ще им бъдат необходими различни софтуерни продукти, за да постигнат това. Колкото повече различни продукти, толкова по-трудно става поддържането на здрава и сигурна комуникация помежду им.

По тази причина, настоящата дипломна работа разглежда възможността за изграждане на система за управление на авиокомпаниии, включваща автоматизациите и технологиите, от които се нуждае тя, реализирани единствено и само на една платформа – **Salesforce**.

В първа глава ще бъдат разгледани по-детайлно темата и целта на проекта. Като се обърне внимание на развитието на авиационната индустрия и от какво се нуждае тя в момента.

Във втора глава ще бъдат представени използваните технологии за реализацията на проекта заедно с техните положителни и отрицателни страни и каква роля в имплементацията заемат те.

Трета глава включва аргументи за избора на технологията, използвана за проекта и описание на реализираните автоматизации като за всяка от тях е описан подробно процесът на изграждането ѝ.

В четвърта глава е изложена сравнителна характеристика като процесите, включени в настоящия проект, биват сравнени с процесите на други две компании, използващи същата технология, избрана и за текущата разработка.

Приносите на автора се изразяват в дизайн на модела на базата данни, създаването на персонализирани компоненти и автоматизации, със стремежа крайният продукт да бъде максимално гъвкав и удобен за потребителите.

## Abstract

In today's fast-paced and technology-driven world, companies need advanced technology to be competitive. Businesses are looking to automate every possible process in their company so that their employees can focus on the work that cannot be done by any piece of software.

Without the use of any technology, running a business would be a real challenge. And when it comes to commercial airlines, which have complex operations involving a large amount of data, this is close to impossible.

As companies desire to digitize more of their business processes, they conclude that they may need different software products to achieve this. The more different products, the more difficult it becomes to maintain robust and secure communication between them.

For this reason, the current thesis explores the possibility of building a management system for airlines that includes the necessary automations and technologies, all implemented on a single platform – **Salesforce**.

The first chapter will explore the topic and goal of the project in more detail, focusing on how the aviation industry has evolved over the years and its current needs.

The second chapter will present the technologies used in the process of making this project, discussing their pros and cons and their roles in the implementation.

The third chapter includes arguments for the choice of the technology used and a description of the implemented automations, detailing the process of building each one.

The fourth chapter provides a comparative analysis, comparing the processes included in the current project with those of two other companies that use the same technology as the current project.

The contributions of the author include designing the data model, creating custom components and automations, with the aim of making the final product as flexible and user-friendly as possible.

# Съдържание

Резюме.....	3
Abstract .....	4
Списък с фигури .....	7
Списък с фрагменти от код .....	8
Списък с таблици .....	8
Речник с използвани термини.....	9
1    Увод .....	10
1.1    Актуалност на темата .....	10
1.2    Цел.....	11
1.3    Структура и обем на дипломната работа.....	11
2    Използвани технологии .....	13
2.1    Salesforce .....	13
2.1.1    Experience Cloud.....	13
2.1.2    Service Cloud.....	14
2.1.3    Salesforce CMS .....	14
2.2    Apex .....	15
2.3    SOQL .....	16
2.4    LWC.....	16
2.5    SLDS.....	17
2.6    Visualforce Pages.....	17
3    Реализация.....	18
3.1    Избор на технология за разработката .....	18
3.1.1    Защо облачно решение? .....	18
3.1.2    Защо Salesforce? .....	19
3.2    Технически изисквания .....	21
3.2.1    Платформа .....	21
3.2.2    Портал .....	23
3.3    Модел на базата данни .....	24
3.3.1    Стандартни обекти.....	26

3.3.2	Персонализирани обекти.....	27
3.4	Процеси, реализирани на платформата .....	30
3.4.1	Персонализирано приложение.....	30
3.4.2	Създаване на продукти при добавяне на нов полет .....	31
3.4.3	Актуализация на продукти при редакция на полет .....	33
3.4.4	Изпращане на имейл при добавяне на член в екипажа .....	34
3.4.5	Чат в реално време.....	36
3.4.6	Изтриване на съобщения след определено време.....	41
3.4.7	Календар с полети.....	43
3.5	Процеси, реализирани в портала .....	47
3.5.1	Локализация .....	48
3.5.2	Система за управление на съдържание.....	49
3.5.3	Дизайн .....	50
3.5.4	Компонент за основна навигация.....	51
3.5.5	Страници за Вход и Регистрация .....	52
3.5.6	Компонент за закупуване на самолетни билети .....	52
3.5.7	Картови разплащания .....	57
3.5.8	Други компоненти .....	58
4	Сравнителна характеристика.....	60
4.1	Salesforce портал .....	60
4.2	Визуализация и анализ на данни (Business Intelligence) .....	60
4.3	Чат роботи .....	61
	Авторска справка за приносите .....	62
	Перспективи за бъдещо развитие.....	64
	Използвана литература.....	65

## Списък с фигури

Фигура 1. Сравнение между „софтуер като услуга“, „платформа като услуга“ и традиционна разработка .....	19
Фигура 2. Use Case диаграма за търговски представител, администратор и агент .....	22
Фигура 3. Use Case диаграма за екипажа .....	23
Фигура 4. Use Case диаграма за потребител на портала .....	24
Фигура 5. Модел на базата данни.....	25
Фигура 6. Лого, заглавие и навигация на персонализирано приложение .....	31
Фигура 7. Имейл, получен от член на екипа при добавянето му към полет .....	35
Фигура 8. Компонент за чат в реално време .....	37
Фигура 9. Известия в платформата и от мобилното приложение .....	41
Фигура 10. Панел със Scheduled Jobs .....	43
Фигура 11. Календар функционалност за средни и големи екрани .....	44
Фигура 12. Календар функционалност за малки екрани.....	45
Фигура 13. Генериран PDF файл .....	47
Фигура 14. Панел с параметри на компонент .....	50
Фигура 15. Основна навигация.....	51
Фигура 16. Страница за вход в портала.....	52
Фигура 17. Търсене на полети .....	53
Фигура 18. Избор на полети.....	54
Фигура 19. Попълване на информация за пътници .....	55
Фигура 20. Избор на места.....	56
Фигура 21. Избор на багаж .....	56

## Списък с фрагменти от код

Код 1. Формула за изчисляване на възрастта на самолет в години .....	28
Код 2. Формула за визуализиране на продължителността на полет в четим вид.....	29
Код 3. Тригер на обект "Полет" .....	31
Код 4. Разпределение на логиката.....	32
Код 5. Метод за създаване на продукти.....	32
Код 6. Създаване на колекция за цените на местата .....	32
Код 7. Записи за места спрямо вида самолет .....	33
Код 8. Генериране на връзка за точна локация с координати .....	35
Код 9. Създаване на имейл инстанция и задаване на параметри .....	36
Код 10. Визуализация на съобщения в хронологията .....	38
Код 11. Абониране за канал.....	40
Код 12. Създаване и изпращане на известие за ново съобщение.....	40
Код 13. Batch Job за изтриване на съобщения.....	42
Код 14. Scheduled Job за изтриване на съобщения .....	42
Код 15. Стартиране на Scheduled Job.....	42
Код 16. Генериране на PDF файл .....	46

## Списък с таблици

Таблица 1. Сравнителна таблица.....	60
-------------------------------------	----



## Речник с използвани термини

- CRM (Customer Relationship Management) – система за управление на връзките с клиенти;
- Тригер – автоматизация, която се изпълнява при промяна на запис в базата данни;
- Фреймуърк - софтуерен модул, при използването на който контролът е обърнат и вече не е изцяло в ръцете на приложния код;
- Обект – таблица в базата данни на *Salesforce*;
- CMS (Content Management System) – система за управление на съдържанието;
- Experience Cloud – приложение на *Salesforce*, позволяващо създаването на уеб портали;
- Experience Builder – инструмент, чрез който се изгражда един сайт на платформата;

# 1 Увод

## 1.1 Актуалност на темата

Историята за дигиталната трансформация в авиационния сектор е една от най-невероятните истории, доказваща как технологиите са били най-доброто нещо, което е можело да бъде измислено. Всичко започва през далечната 1903 година с първия полет, осъществен от братята Райт. Малко по-късно, през 1914 година е извършен и първият комерсиален полет, превозващ пътници, които са заплатили за тази услуга. От тогава се появява възможността за закупуване на самолетни билети. Процесът, който се е извършвал е доста по различен от това, с което хората са свикнали днес.

Авиокомпаниите наемали множество оператори, които да обработват резервациите. Те работили с индексни карти за всеки полет. За да се резервира място операторът трябвало да вземе съответната карта, да отбележи мястото, избрано от човека и да напише самолетния билет на ръка. Този процес отнемал около 90 минути за една резервация [1]. Дъгълс Колман, работил в резервационния център на *National Airlines* през 1964 година споделя: „Полет 26 на 18-ти Януари - JAX до MSY имаше от 30 до 40 джоба, по един за всеки ден, в който се изпълнява. Служителите се качваха на стълби, за да видят броя на пътнически записи във всеки джоб. Ако изглежда почти пълен, се поставя оранжев етикет в лицето на джоба, за да се вижда от всички агенти, приемащи обаждания. Ако полета е запълнен, се поставя червен етикет и повече места не се потвърждаваха.“ [2]

С популяризирането на пътуването със самолет тази система на работа била крайно неефективна. Затова през 1952 година *American Airlines* създава компютъризирана система, която позволявала едновременно достъп до информацията за полетите. Въпреки това, процесът по закупуване на билети остава в по-голямата си част – ръчен.

Съществената крачка към технологиите дошла 7 години и 40 милиона долара по-късно (~ 430 милиона долара днес [3]) след случайна среща между директора на *American Airlines* и търговски представител на *IBM*. Появява се системата *SABRE (Semi-Automated Business Research Environment)* – първата голяма *e-commerce* система, обработваща над милиони долара на ден още преди съществуването на Интернет. Благодарение на тази система времето за обслужването на клиенти от 90 минути станало няколко секунди [4].

През годините тази индустрия се развива с много бързи темпове. Така остава и до ден днешен. Въпреки многобройните проблеми, с които се е сблъскала и разрешавала, днес авиокомпаниите се изправени пред нови такива, които изискват интегрирани и рационални решения. Традиционния подход за използване на различни софтуерни продукти, които да покрият различни аспекти от операциите на компанията, може да бъде тромаво и да доведе до несъответствия. Чрез реализирането на комплексна система за управление, базирана на

*Salesforce*, всички необходими процеси от закупуване на билети до управление на екипажа са безпроблемно интегрирани на една единствена платформа. Това не само опростява операциите, но също и подобрява достоверността на данните, потребителското преживяване и намалява разходите за поддръжка.

Настоящият проект използва платформата на *Salesforce*, за да реализира процесите, от които една авиокомпания се нуждае, като ги обединява в един краен продукт.

## **1.2 Цел**

Целта на проекта е да се реализира обширна система за управление на авиокомпания, използвайки платформата *Salesforce*. Системата има за цел да рационализира различни оперативни процеси, да подобри управлението на данни и да подобри цялостната ефективност. Чрез събирането на всички възможни бизнес процеси в една платформа, разработката цели да отстрани нуждата от наличието на няколко софтуерни продукта, което само по себе си означава намаляването на сложността на системата, разходи за управление и човешки ресурси, които да управляват тези системи. Крайната цел е изграждането на безпроблемно интегрирано решение, което да поддържа управление на полети и екипаж, обслужване на клиенти и други важни операции, присъщи за този вид бизнес, гарантирайки, че авиокомпанията ще работи по-ефективно и ще може да предостави на своите потребители и клиенти незабравимо преживяване.

## **1.3 Структура и обем на дипломната работа**

Настоящата дипломна работа е изградена от 4 глави, авторска справка за приносите, перспективи за бъдещо развитие и използвана литература.

Обемът на основната част е 64 страници, а на използваната литература – 1. Литературните източници са 9 на брой и представляват адреси на уеб страници. Всеки източник включва информация за датата на последното посещение.

В първа глава бяха разгледани по-детайлно темата и целта на проекта. Като се обръща внимание на развитието на авиационната индустрия и от какво се нуждае тя в момента.

Във втора глава ще бъдат представени използваните технологии за реализацията на проекта заедно с техните положителни и отрицателни страни и каква роля в имплементацията заемат те.

Трета глава включва аргументи за избора на технологията, използвана за проекта и описание на реализираните автоматизации като за всяка от тях е описан подробно процесът на изграждането ѝ.

В четвърта глава е изложена сравнителна характеристика като процесите, включени в настоящия проект, биват сравнени с процесите на други две компании, използващи същата технология, избрана и за текущата разработка.

Приносите на автора се изразяват в дизайн на модела на базата данни, създаването на персонализирани компоненти и автоматизации, със стремежа крайният продукт да бъде максимално гъвкав и удобен за потребителите.

## 2 Използвани технологии

В тази глава ще бъдат представени технологиите, използвани за реализиране на проекта, описан в тази дипломна работа. Тъй като той се разполага върху готова облачна услуга – платформата на *Salesforce*, всички технологии са подбрани, така че да се поддържат от системата.

### 2.1 Salesforce

*Salesforce* е облачно базирана платформа, създадена през 1999 година от Марк Беньоф, Паркър Харис, Франк Домингез и Дейв Моленхоф. Нейната цел е да помага на бизнесите да управляват връзките с клиентите си и да рационализира техните операции. *Salesforce* предлага обширен набор от инструменти и апликации за различни аспекти на бизнеса като – продажби, услуги, маркетинг и други. Една от основните причини компаниите да избират тази система е възможността за обединяване на различни функционалности в една платформа. По този начин се опростява управлението на данните, позволявайки на потребителите да изготвят различни отчети, на базата на които да взимат информирани решения.

*Salesforce* оперира на облачна инфраструктура, което означава, че бизнесите имат достъп до своите данни и приложения без значение къде са, стига да имат интернет връзка. Тази гъвкавост е в полза на компании, които искат да се разрастнат бързо без да инвестират много средства в хардуер и неговата поддръжка. Освен това, *Salesforce* много се придържа към сигурността, осигурявайки сензитивните данни да са защитени чрез сложно криптиране, автентикация и регулации за защита на данните.

#### 2.1.1 Experience Cloud

*Experience Cloud* е един от „облаците“ на *Salesforce*, който предоставя възможността на бизнесите да изграждат и управляват брандирани уеб сайтове за своите клиенти, служители и партньори. Предоставя набор от готови шаблони, изготвени за различни бизнес сценарии, като: партньорски портал, клиентски портал или „направи си сам“. Целта на тези шаблони е да улеснят разработката като предоставят готово разположение на съдържанието и компоненти, които могат лесно да бъдат персонализирани, за да пасват на бизнес изискванията, които трябва да бъдат реализирани.

Инструментът, чрез който се изграждат тези портали се нарича *Experience Builder*. Той позволява конфигурирането на сайта чрез принципа на *drag-and-drop*. Това позволява на хора, които нямат опит с програмирането, да създават професионални и функционални уеб сайтове. В допълнение, този инструмент поддържа голям набор от готови компоненти

като: текст блокове, колони, изображения, банери, статии, форми, повтарящи се компоненти и други; позволявайки още по-комплексно персонализиране на сайта.

*Experience Cloud* също така предоставя използването на функционалности като регистрация и вход на потребители в порталите като се грижи за сигурността на техните данни, отстранявайки нуждата от създаване на персонализирани процеси, които да се грижат за това. Освен това, поддържа безпроблемна интеграция с останалите продукти от палитрата на *Salesforce*, позволявайки използването на инструментите за визуализация и анализ на данни на момента, без да е нужна допълнителна интеграция за това.

### 2.1.2 Service Cloud

*Service Cloud* е един от ключовите компоненти на *Salesforce*, който е фокусиран върху обслужването на клиенти. Предоставя съвкупност от инструменти за управляването на комуникацията с клиентите по различни канали като: телефон, имейл, чат и социални мрежи. Той помага на отделите за обслужване на клиенти да предоставят по-бързи и ефикасни услуги на своите клиенти като събира всички потребителски запитвания и проблеми на едно място.

Една от основните функционалности на *Service Cloud* е неговата чат функционалност, която позволява на клиентите да общуват с агенти в реално време. Наличието на такъв чат може много да подобри удовлетвореността на клиентите. Освен това, този чат може да се вгради директно в *Experience Cloud* портал, което за пореден път доказва колко е гъвкава *Salesforce* платформата.

Освен агентите, които са човешки същества, *Service Cloud* включва в себе си и *Einstein Bot* – чат бот, който може да бъде вграден във вече създаден чат. Този бот е изграден от предварително създадена поредица от действия с множество разклонения и възможност за включване на различни автоматизации. При добре изграден алгоритъм, може да успее да отговори на проблемите на потребителите без дори да стигат до разговор с истински агент.

### 2.1.3 Salesforce CMS

*Salesforce CMS* е система за управление на съдържание, която се интегрира безпроблемно с порталите, изградени чрез *Experience Cloud*. Тя позволява на бизнесите да създават, управляват и използват това съдържание по различни канали. Системата е проектирана да бъде удобна за потребителите, които я използват, като им позволява да създават богато съдържание без да имат нужда от добре развити технически умения.

Един от най-големите плюсове на тази система е интеграцията ѝ с *Experience Builder*, която позволява потребителите на плъзгат и поставят съдържание директно в сайтовете. Това позволява управлението на съдържанието да биде ефективно и интуитивно. Освен това

*Salesforce CMS* предоставя възможността за поддържането на многоезично съдържание, позволявайки на бизнесите да достигнат глобалната публика като превеждат съдържанието на различни езици.

## 2.2 Apex

*Apex* е език за програмиране, създаден от *Salesforce*. Използва се за създаването на персонализирани свойства и автоматизиране на процеси в платформата. Той е строго типизиран и обектно-ориентиран. Създаден е да бъде лесен за ползване от тези, запознати с Java или C#, което го прави достъпен за голям брой програмисти.

*Apex* в ключов в създаването на персонализирана бизнес логика и автоматизирането на комплексни бизнес процеси в *Salesforce*. Позволява на програмистите да пишат персонализирани тригери, класове и контролери, които да се използват за разширяването на функционалностите, предоставени от платформата. Той е силно интегриран със *Salesforce*, позволявайки безпроблемния достъп и манипулация на данните от базата.

Основни свойства на *Apex*:

- Персонализирана бизнес логика – *Apex* позволява създаването на персонализирани бизнес процеси и правила. Това означава, че платформата може да бъде настроена точно както съответният бизнес работи;
- *Triggers* – тригерите позволяват да се изпълни определена автоматизация, когато настъпи промяна в базата данни. Спрямо това кога се изпълняват, съществуват няколко вида: преди и след създаване на запис, преди и след обновяване на запис и преди изтриване на запис;
- *Batch Apex* – функционалност, която се използва при работа с голямо количество данни. Разбива данните на парчета с максимална големина 200 записа, което помага на платформата да ги обработи по-лесно и ефикасно. Освен това, е една от техниките за спазване на лимитите на платформата. Често се използва за обновяване на голям брой записи или за чистене на данни;
- *Scheduled Apex* – позволява настройването на дадена автоматизация да се изпълни в точно определен момент като може да се повтаря или изпълни само веднъж. Подходящ за задачи, които се случват регулярно, като изпращането на дневни справки;
- *Web Services* и *Callouts* – *Apex* може да комуникира с други системи като използва уеб услуги, което означава, че платформата може както да изпраща, така и да получава информация от други приложения. Поддържа *RESTful* и *SOAP*, което прави платформата лесна за интегриране с други платформи.

- *Unit Tests* – тези тестове имат за цел да валидират поведението на платформата след реализирането на персонализирани процеси. Те се пишат в тестовия фреймуърк на *Apex*, което позволява на програмистите да създават тестови методи, за да се уверят, че техният код работи както се предполага и по не влияе негативно на платформата. Идеята е по възможност да се засекат потенциални проблеми в по-ранен етап. Освен това, за да се качи дадена *Apex* автоматизация в продукционна среда е задължително тези класове да имат поне 75% покритие, при създаването на тестови класове.

## 2.3 SOQL

*SOQL* или *Salesforce Object Query Language* е инструмент, който се използва за извличането на данни от *Salesforce*. Работи като добре познатия *SQL*, но е пригуден специално за данните от платформат. Чрез него може да се вземат необходимите записи като позволява филтриране и сортиране. Ако в даден обект има връзка към друг такъв, чрез заявка могат да бъдат достигнати и данните на записа свързан към текущия. Позволява използването на заявка в заявката, когато има нужда от по-комплексно филтриране. Освен това, когато се използва в *Apex*, има възможност за използването на динамични параметри, които са променливи, константи или колекции, дефинирани в кода.

## 2.4 LWC

*LWC* или *Lightning Web Components* е сравнително нов фреймуърк за изграждане на графични потребителски интерфейси. Създадени през 2019 година, това са компоненти, които използват стандартни уеб технологии като: *HTML*, *CSS* и *JavaScript*. Всеки компонент представлява пакет от файлове от видовете, изброени в предното изречение. Освен тях, в пакета се включва и *XML* файл, който описва свойствата на компонента като: дали може да се използва на платформата, име, етикет, места, на които може да бъде използван и параметри, които могат да се различават спрямо мястото, на което е поставен компонента.

Тези компоненти са тясно интегрирани с *Apex*. Методите, написани в *Apex*, могат да бъдат внедрени в *JavaScript* частта на компонента, което позволява изпълнението на сървърни операции да бъде лесно достъпно за клиентската страна. Единственото условие е съответните методи да носят анотацията *@AuraEnabled*

*LWC* предлага компонентно-базирана архитектура като позволява на програмистите да изграждат капсулирани и преизползваеми компоненти. Едно от ключовите свойства на компонентите е използването на декоратори като:

- *@api* – поставен пред променливи или методи, той ги прави достъпни като позволява да бъдат използвани от външен за компонента източник;



- *@track* – прави променливите реактивни, тоест ако се използва един списък за визуализиране на данни в *HTML* файла на компонента и настъпи промяна по списъка, тя няма да бъде отразена автоматично в интерфейса, ако липсва този декоратор пред съответния списък;
- *@wire* – един от начините за извикване на метод от *Apex*. По този начин методът се извиква автоматично при зареждането на компонента. В този случай е задължително метода в *Apex* класа да бъде отбелязан като *Cacheable = true*, тъй като данните, които се връщат биват запазени в кеш паметта на платформата, за по бързо повторно зареждане, ако е необходимо.

Другият начин за изпълнението на *Apex* методи през клиентската част на приложението е те да бъдат извикани императивно. Това позволява повече контрол върху това кога точно се изпълнява метода и как е обработен резултата.

Използвайки такива компоненти позволява разработването на сложни и адаптивни потребителски интерфейси. Чрез тях могат да се постигнат автоматизации, които се присъщи за бизнеса и няма готово решение за тях.

## 2.5 SLDS

*Salesforce Lightning Design System (SLDS)* е обхватен *CSS* фреймуърк и пакет от насоки за дизайна, изградени от *Salesforce* за да се осигури консистентност в изгледа във всички *Salesforce* апликации. *SLDS* предоставя голям набор от преизползваеми компоненти, стилове и дизайн токени, които отговарят на интерфейса на платформата.

Системата включва базови компоненти като: бутони, форми, карти, матрици; които лесно могат да се вградят в различните апликации на платформата, за да се постигне единен изглед. Освен това, включва и дизайн токени, които представляват предефинирани променливи за цветове, шрифт, разстояние и други. В допълнение, предоставя и готови *CSS* класове, за прилагането на често срещани стилове като: вътрешно и външно отстояние, подравняване и видимост. Друго, което се включва е голям брой икони, разделени в три категории, спрямо стила, който представят.

## 2.6 Visualforce Pages

*Visualforce* е мощен фреймуърк в *Salesforce*, позволяващ на програмистите да създават персонализиран потребителски интерфейс като използват технологии като: *HTML*, *CSS* и *JavaScript*. Едни от основните му положителни страни са възможностите за генериране на *PDF* файл и изграждането на имейл шаблони. И двете функционалности са описани подробно в следващата глава от дипломната работа.

## 3 Реализация

За всяка една компания в днешно време е изключително важно да стои на гребена на вълната с всяка нова тенденция, която се появява на пазара. Авиационният бизнес и по-точно комерсиалните авиокомпаниите е изграден от множество служители, клиенти и процеси, с който да се управлява всичко това. Именно това е причината за използването на комбинация от различни софтуерни продукти от всякакво естество, покриващи нуждите на бизнеса. Проблемът се появява, когато трябва да се осъществи безпроблемна комуникация между тези системи. Затова настоящият проект има за цел да представи как е възможно всички бизнес процеси на една авиокомпания да бъдат реализирани върху една единствена платформа – **Salesforce**. За да бъде постигнато това се преминава през задълбочен анализ на бизнес изискванията, изграждането на кореспондиращ модел на данни и самата реализация.

### 3.1 Избор на технология за разработката

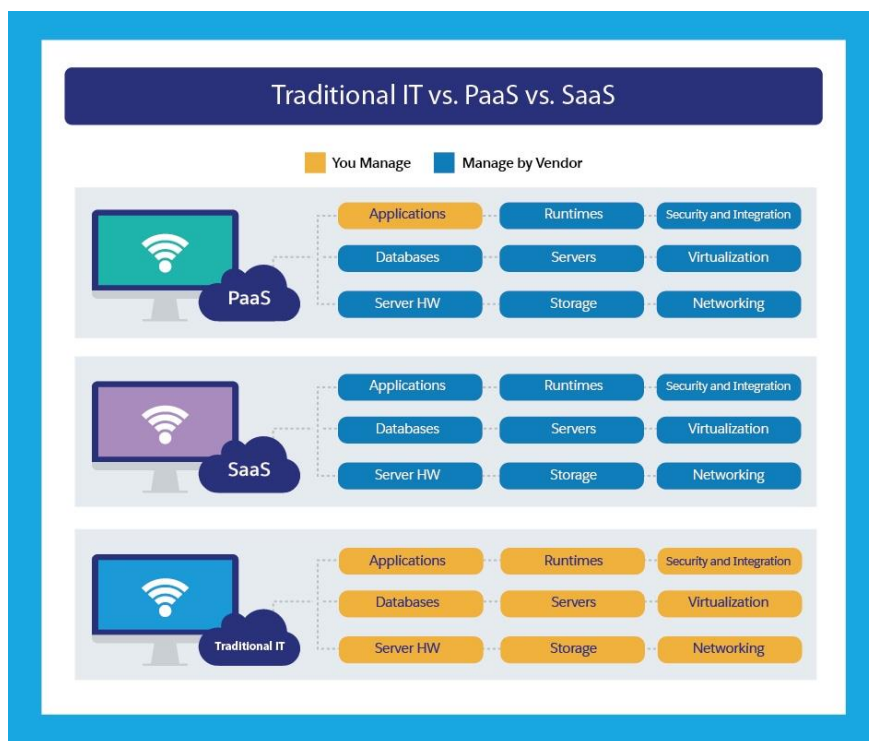
Част от началния етап на разработването на едно приложение е изборът на технологии, които ще бъдат използвани за неговото реализиране. Този избор трябва да бъде съобразен не само с това дали даденото нещо отговаря на нуждите на бизнеса, но също така и колко време би отнела реализацията, колко е надеждно, колко е сигурно и най-накрая колко средства ще се вложат до получаването на завършения продукт. Друг аспект при избора на технологии е и колко лесна би била за поддръжка системата след като всички процеси са реализирани.

#### 3.1.1 Защо облачно решение?

Създаването на едно приложение не е никак лесна задача. То трябва да бъде съобразено с принципите на потребителския интерфейс, потребителското изживяване, графичния дизайн и самата разработка. Освен това всеки от екипа трябва да бъде запознат с предназначението на приложението и целта на компанията, за която бива разработвано то. Всичко изброено на куп звучи сложно, но най-трудната част за изграждане е всъщност инфраструктурата на самото приложение и по-точно как ще бъде осъществена комуникацията между софтуера и хардуера.

Използването на облачните решения като „платформа като услуга“ (**PaaS**) и „софтуер като услуга“ (**SaaS**), спестяват усилията, които биха били необходими за изграждане на тази връзка. Услуги от този тип предоставят възможността за разработването, пускането в употреба и поддръжката на едно приложение без да се мисли за инфраструктурата, която се изисква при повечето приложения налични на пазара, защото за нея се грижи самата платформа.

Всеки бизнес има нужда от инфраструктура – сървъри, терминали, интернет свързаност, помещения за наличния хардуер, софтуер, служители. Поддръжката на всичко това представлява немалка част от финансите на една компания. Следователно, използването на облачна услуга би намалило драстично тези разходи, тъй като с тях се избягва инвестицията в скъп хардуер, необходим за бизнес процесите. Също така освен парични средства се спестява и времето, което се използва за поддръжка и актуализация при нужда.



**Фигура 1. Сравнение между „софтуер като услуга“, „платформа като услуга“ и традиционна разработка**

На гореприложената фигура може да се види каква голяма част от процесите (бази данни, сървър, хранилища, сигурност и др.) се поемат от облачната услуга, което допълнително доказва положителните страни на тази технология.

Всички изброени качества досега водят до избора на облачно решение за реализацията на проекта.

### 3.1.2 Защо Salesforce?

Salesforce е номер едно CRM платформа за бизнеси от всякакъв мащаб. Този резултат е постигнат с постоянното развитие на продукта и предлаганите от тях разнообразни готови решения, които помагат на различните бизнеси да имплементират едни от най-модерните

технологии. Наблюдава се средно 38% по-бързо взимане на решения, 25% ръст в печалбите и 35% увеличение в удовлетвореността на клиентите при компаниите, които използват *Salesforce CRM* [5].

Един от най-големите плюсове на платформата е екосистемата, която е изградена на нея. Комбинацията от различни „облаци“, които предоставят разнообразни услуги, предоставя 360-градусов изглед за клиентите на една компания. От продажби (Sales Cloud), обслужване (Service Cloud), преживяване (Experience Cloud), маркетинг (Marketing Cloud), търговия (Commerce Cloud), до аналитика (Analytics Cloud), всичко е обединено на едно място. Това прави платформата подходяща за използване от всеки един бизнес, без значение от сферата, в който е той. Salesforce предоставя много възможности за персонализация, за да може да пасне на уникалните нужди на бизнеса.

През годините Salesforce успява да се докаже като едно от най-добрите решения на пазара. С пускането на нови версии, които включват много нови функционалности, три пъти годишно, компанията показва желанието си за непрестанно развитие. Поддръжката на платформата се случва автоматично и ако се появи проблем, технически екип реагира незабавно и се отстранява в много кратки срокове.

Това, че е платформа, която се използва наготово, по никакъв начин не ограничава нейните функционалности. Предоставена е възможност за безпроблемна интеграция с външни системи като ERP системи, имейл платформи, платформи за автоматизиран маркетинг и други, което допълнително обогатява приложението ѝ.

Наличието на онлайн магазин (AppExchange) с хиляди готови приложения и компоненти също би помогнал при разширяването на възможностите на платформата, тъй като предлага разнообразни решения за всякакви нужди.

Друга положителна страна е наличието на мобилно приложение, което позволява на хората, използващи платформата да имат достъп до нея от всяка точка. В днешно време голяма част от хората водят забързан начин на живот и винаги са в движение, без възможност за използване на компютър или лаптоп, което прави лесния достъп до платформата от мобилно устройство изключително важно.

Последно, но не на последно място е изградената покрай Salesforce общност. Организирането на стотици събития всяка година, на които се представят актуални теми свързани с бизнеса, изнасянето на презентации за различни функционалности на платформата, поддържане на иновативна онлайн обучителна платформа с интерактивни модули, покриващи както основни теми така и такива за напреднали, допринася до събирането на милиони хора от всички краища на света, с различен опит и различни квалификации, заедно. Което само по себе си означава, че ако някой има нужда от помощ, винаги има кой да откликне отсреща.

Всичко това аргументира защо изборът на Salesforce за настоящия проект, е правилен.

## 3.2 Технически изисквания

Както е описано по-горе, целта на проекта е да реализира бизнес процесите, от които се нуждае една авиокомпания, само върху платформата Salesforce, което да предотврати нуждата за осъществяване на комуникация между различни системи. Управлението на бизнес от такъв мащаб обхваща голямо количество потребители. Основната идея е разработката да служи както за управление на служителите на компанията, така и за клиентите, които се явяват купувачи на самолетни билети.

По тази причина, внедрените функционалности, могат да бъдат разделени на две основни групи – такива за вътрешни служители и такива за потребителите на авиокомпанията. Всички служители използват самата Salesforce платформа и по-точно персонализираното за тях приложение, което съдържа различни раздели и функционалности спрямо длъжността на човека. Докато потребителите използват уеб сайт (портал), който отново е изграден върху Salesforce, но нямат директен достъп до самата платформа.

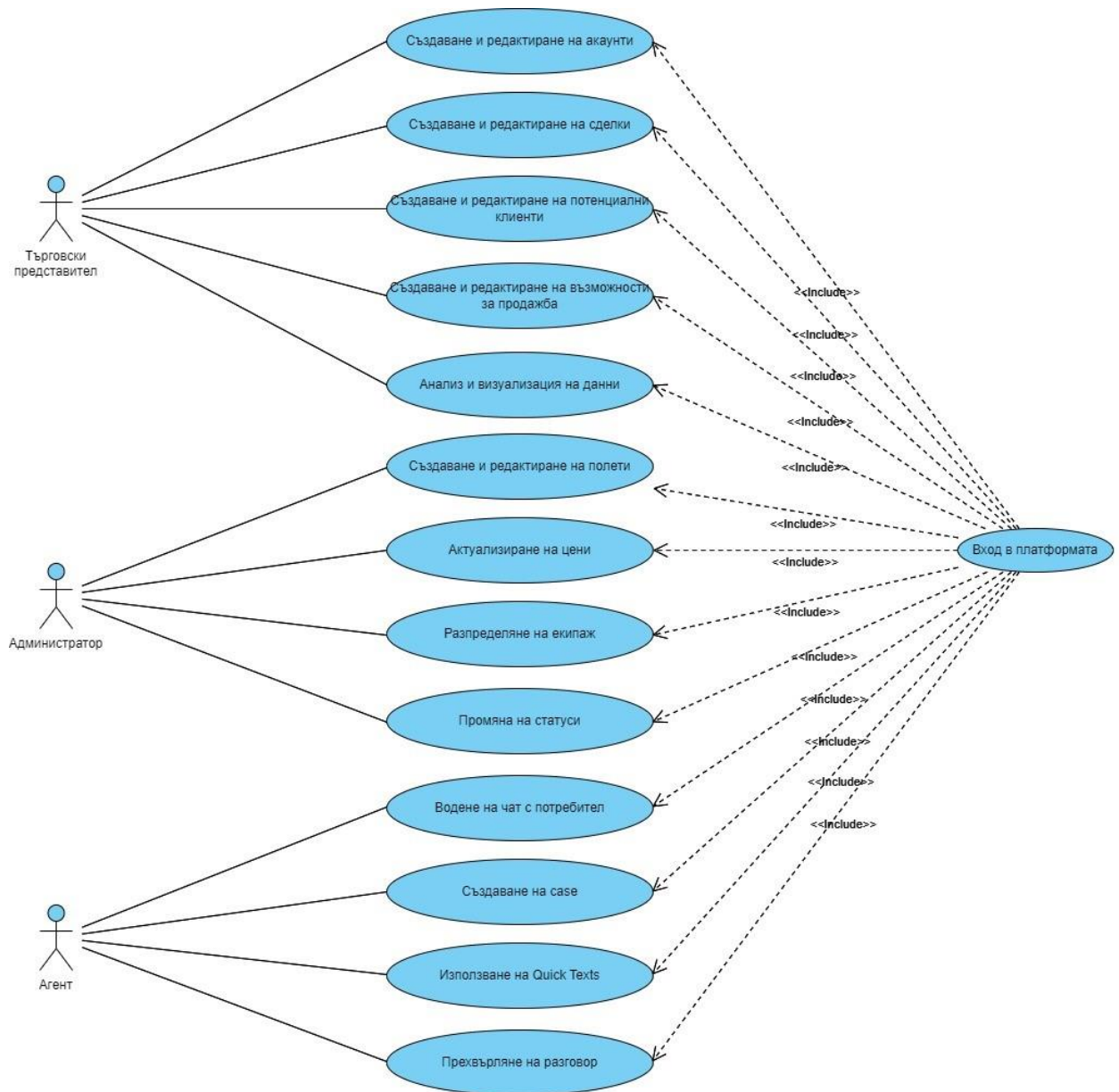
### 3.2.1 Платформа

Всичко свързано с вътрешните процеси на авиокомпанията заема място върху платформата на Salesforce, където има изградени множество автоматизации, за по-лесна реализация. Всеки от служителите на компанията може да влезе в своя профил, използвайки уеб браузър или приложението на Salesforce, което могат да инсталират на своите лични устройства. Различните отдели имат достъп до различни функционалности като например:

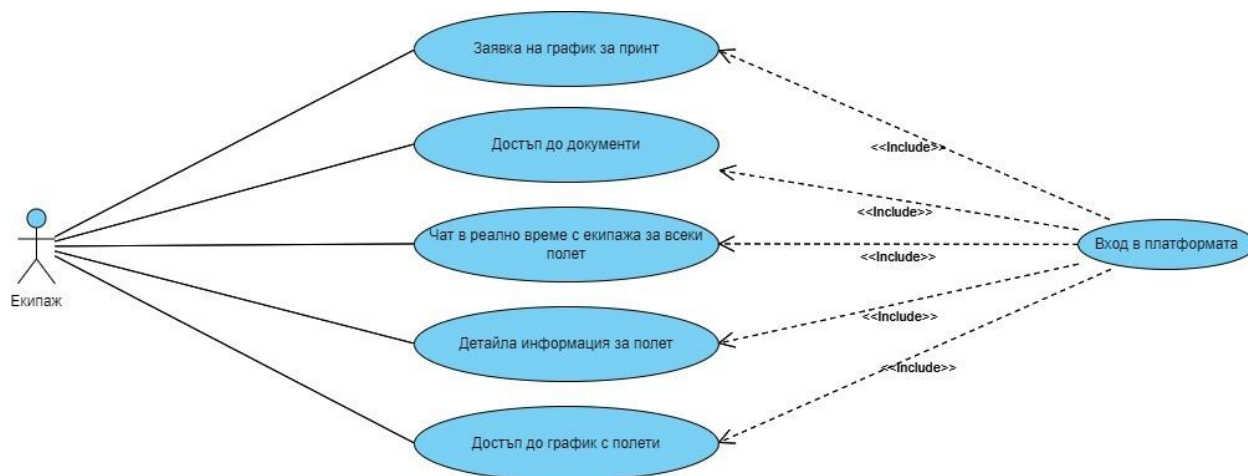
- **Екипаж (капитани и стюардеси)** – възможност да виждат график с полетите, към които са добавени, детайлна информация за всеки полет, достъп до чат стая с възможност за чат в реално време, състояща се с останалите хора от екипажа спрямо всеки полет, достъп до всички необходими документи за достъп до летище и престой в хотели, получаване на нотификации в платформата и имейл нотификации при разпределяне в нов полет и при промяна на статуса, получаване на PDF документ по имейл с график за текущия месец;
- **Отдел продажби** – достъп до всички налични акаунти, включващи производители на самолети, летища и потребителите на компанията, възможност за създаване на сделки с тези акаунти, запис на нови потенциални клиенти и възможности за продажба, анализ и визуализация на данни, 360-градусов изглед на всеки клиент;
- **Отдел обслужване на клиенти** – достъп до Service конзола, откъдето могат да отговарят на клиенти, инициирали чат от портала, 360-градусов изглед на всеки

потребител, създаване на „случаи“, съдържащи цялата кореспонденция и потенциални следващи стъпки за разрешаване на проблема, достъп до библиотека с „бързи текстове“ при разговор, възможност за прехвърляне на разговор към друг агент;

- **Администратор** – създаване полети, актуализиране на цени, разпределяне на екипаж, промяна на статуси.



**Фигура 2. Use Case диаграма за търговски представител, администратор и агент**



**Фигура 3. Use Case диаграма за екипажа**

### 3.2.2 Портал

Порталът представлява уеб сайт, изграден върху Experience Cloud на Salesforce, който се разпространява като всеки обикновен сайт в Интернет. Ето и какви функционалности са достъпни там:

- Вход и регистрация;
- Управление на профил;
- Проследяване на полет;
- Закупуване на самолетни билети;
- Избор на места;
- Избор на багаж;
- Плащане с карта;
- Управление на резервация;
- Достъп до документи/файлове;
- Допълнителна информация за авиокомпанията и важни съобщения;
- Известия при промяна на статус на полет;
- Провеждане на текстов разговор с вътрешен служител.



**Фигура 4. Use Case диаграма за потребител на портала**

### 3.3 Модел на базата данни

В изграждането на каквото и да е приложение едно от най-важните неща е добре структурираният модел на данни, за да се осигури правилното записване и след това управление на данните. Настоящият модел е изграден от комбинация между стандартни и персонализирани обекти (таблици), за да се напасне приложението към нуждите на една комерсиална авиокомпания. Стандартните обекти се използват за управлението на CRM насочените функционалности като – акаунти, контакти, сделки, бизнес аспекти и други. Докато персонализираните такива или по-точно новосъздадените са основно насочени към специфичната за индустрията информация като – самолетни модели, полети, маршрути и други. С множеството връзки, изградени между тези обекти, се постига безпроблемната интеграция на данните през различните звена на проекта. В следващите точки ще бъдат разгледани по-детайлно основните обекти в базата с техните атрибути и как фигурират в цялостната идея на приложението.





### 3.3.1 Стандартни обекти

Платформата Salesforce в най-основния си вид представлява една CRM система, помагаща на компаниите да развиват своя бизнес по-бързо и по-лесно, като се възползват от структурирани и свързани наготово обекти, които съхраняват цялата информация, от която биха се нуждаели. Точно затова и в този проект се използват и някои от стандартните обекти, особено за отдел търговски представители, на който цялата дейност се върти около типичните за CRM функционалности.

На първо място е обект „Акаунт“ (Accounts), който в съществото си представлява фирми, с които бизнеса има връзка по някакъв начин. В текущата разработка обаче обикновеният обект няма да бъде достатъчен, за да покрие нуждите на бизнеса, затова бива разделен на няколко разновидности. Salesforce предоставя тази възможност с така наречените „record types“, чрез които може да има различни видове записи от един тип, като се различават по полета, описващи информация за тях, и различни зависимости помежду им. Първият вид акаунти са тези за производителите на самолети. Това се налага, за да се приложи филтрация при създаването на записи за самолетни полети, за да се предотврати коректност на данните. Другият вид акаунти са тези за летищата. Те съдържат в себе си полета, съответстващи на този вид бизнес като: брой терминали, локация, най-близък град до тях и уникален трибуквен код, служещ за идентифициране на летището, възложен от Международната асоциация за въздушен транспорт (IATA). Този вид записи се използва при създаването на записи за маршрути, които обслужва авиокомпанията. Двата вида акаунти включват в себе си и няколко припокриващи се полета като: брой служители, телефон за контакт, уебсайт и други. На пръв поглед, тази схема би могла да се постигне и чрез създаването на два отделни персонализирани обекта, но това ще доведе до дублиране на полета, което не е за предпочитане в един модел на база данни. Освен това, тези записи ще се използват и при създаването на бизнес аспекти и сделки, когато предстои покупка на нови летателни машини или внедряването на нови маршрути от летища, от които досега не е имало. Обектите, пазеци информация за тези неща (Lead и Opportunity), имат директна връзка с Акаунт обекта, което е другата причина да се избегне създаването на нови обекти. Така би се усложнила допълнително структурата на данните затова е избран подхода с разделянето на вече съществуващия стандартен обект. Последният вид акаунти е малко по-различен от стандартния акаунт, тъй като в себе си той обединява и акаунт и контакт. Нарича се „person account“ и се използва от потребителите на портала, тъй като те са явяват като еднолични търговци и не принадлежат към някоя фирма, за да бъдат просто контакти. Този вид акаунти липсва по подразбиране и трябва да се отключи като функционалност в средата през настройките. Като разпределение на полета те съдържат в себе си комбинация от стандартните такива от обектите „Акаунт“ и „Контакт“ като записите от този вид могат да бъдат достъпвани и създавани от раздела, отговарящ за акаунти.

Следващия стандартен обект, който влиза в реализацията, е „Контакт“. Всеки запис от там представлява служител на вече съществуващ акаунт в системата. Съответно, има пряка връзка с обекта Акаунт, което позволява бързо намиране, лесна визуализация и филтриране на записите спрямо този аспект. Полетата му описват информация като: име, телефон за връзка, имейл, статус, отдел, индикация дали може да се осъществи контакт с него и други.

Обектът за бизнес аспекти (Lead) и този за сделки (Opportunity) са неизменна част от една CRM система. Записите на потенциални клиенти помага за проследяването и евентуалното им конвертиране във важни бизнес възможности. Докато самите сделки предоставят ясен изглед за продажби, прогнози за приходи и като цяло как се развива бизнеса.

За ефективно управление на обслужването на клиенти се използва обектът „Случай“ (Case). С всеки запис от него могат да се следят всички клиентски запитвания и проблеми, улеснявайки бързото им разрешаване, което само по себе си повишава удовлетворителността на клиентите. Този обект се използва основно от отдел обслужване на клиенти и по-точно агентите, които отговарят на чат запитванията, идвайки от портала. При инициране на чат и неговото приемане от агент, автоматично се създава „случай“, който в себе си съдържа цялата проведена дискусия, бележки от агента и друга информация.

Последният стандартен обект, който се използва е „Продукт“ (Product). Той влиза в употреба при създаването на запис за полет. Реализирана е автоматизация, която при създаване на запис, автоматично се създава продукт за базов самолетен билет, местата в самолета и наличните опции за багаж. Тези записи след това се използват за създаването на резервация за определен полет като държат в себе си цената за съответния продукт, вида закупено място или вида багаж, който е избран. Споменатата автоматизация ще бъде описана в следваща точка от документа.

### **3.3.2 Персонализирани обекти**

Всеки бизнес има своите особености, следователно и данни, от които се изгражда са различни. Именно затова в допълнение на стандартните обекти, предоставени от Salesforce, са изградени и още няколко персонализирани такива, описващи специфични за индустрията данни, които се използват в разработката на всички реализирани процеси.

Всичко започва от моделите на самолети. В този обект (Aircraft Model) се пази информация за всеки наличен модел самолет, който се притежава от компанията. Съдържа техническа информация за машината като: скорост на летене (км/ч), вид на двигателя (списък с опции), височина, дължина, размах на крилата (метри), максимален обхват (километри), капацитет на горивото (литри), брой редове и места, минимален размер на

екипажа и други. Освен това има информация и за производителя, която се явява под формата на връзка към обект „Акаунт“, и по-специфично – акаунт от тип „Производител“.

Всеки модел самолет има различно разпределение на местата в него, както различен брой редове, така и разположението на самите седалки – било то по три броя в две колони или пък осем броя, разпределени в три колони. По тази причина е създаден обект (Aircraft Seat), който да пази информация за всяко място в даден самолет. С негова помощ се реализира схемата за избор на места при закупуване на билети в портала, която е описана по-долу. За да се идентифицира една седалка освен връзката към модела самолет, тя включва и информация за това на кой ред се намира и кое точно място е, например – ред: 1, място: А. В допълнение, за всеки запис трябва да се избере и позицията на седалката – до прозорец, по средата или до пътешката; ако се избере опция „до пътешката“, задължително трябва да се попълни и дали пътешката се намира отляво или отдясно на мястото. Друга отличителна характеристика, която трябва да се попълни е каква класа е това място – първа класа, бизнес класа, икономична класа или премиум икономична класа. Изборът се извършва чрез списък със стойности. Тези данни след това се използват при създаването на продуктите и разпределението на цените за местата. Накрая, има и поле, в което могат да се оставят бележки за съответното място като например – „Икономична класа с липсващ прозорец“. Каквото бъде въведено, ще се визуализира на потребителя при избор на места.

След като са налични данни за наличните модели самолети, вече могат да бъдат създавани и записи в обект „Самолет“ (Aircraft), който съдържа информация за всяка единична машина, закупена от компанията. Решението за разделянето на двата обекта – „Самолет“ и „Модел Самолет“ и изграждане на връзка между тях, е взето поради факта, че данни като техническите спецификации ще бъдат дублирани, също така и за по-лесно създаване на записи. Всеки самолет носи уникален регистрационен номер, по който се отличава. Попълват се и полета като – дата на закупуване и цената, на която е закупен. Спрямо датата на закупуване, е създадено поле, представляващо формула, която да пресмята възрастта на машината в години. Формулата е настроена да взема годината от днешната дата и годината от датата на закупуване и да пресметне разликата:

**YEAR(TODAY()) - YEAR(Purchase\_Date\_\_c)**

#### ***Код 1. Формула за изчисляване на възрастта на самолет в години***

Този вид полета се променят динамично, без да се изисква някаква промяна от потребител. Също така, за всеки самолет се задава статус от списък с опции като примерни състояния са: активен, в ремонт, пенсиониран, даден под наем и други.

Следва обект „Маршрут“ (Route), съдържащ данни за линиите, по които се осъществяват полети от авиокомпанията. Конвенцията за името на маршрута е то да съдържа кодовете на летището, от което се излита, и това, на което се каца, разделени с тире, например: SOF-CGN (от летище София до летище Кьолн-Бонн). Освен името,

информация за летищата се попълва е в двете полета, които са връзки към обект „Акаунт“ от тип „Летище“. Също така, трябва да се добавят и данни за това колко е директната дистанция по въздух и колко е очакваната продължителност на полета в минути. За да е лесно за разчитане, времетраенето е преобразувано в четим формат, например: 2h 15m; с помощта на формула, изглеждаща по този начин:

```
TEXT(FLOOR( Estimated_Duration_In_Minutes__c / 60)) & "h " &  
TEXT(MOD( Estimated_Duration_In_Minutes__c , 60)) & "m"
```

#### ***Код 2. Формула за визуализиране на продължителността на полет в четим вид***

Обект „Полет“ (Flight) съхранява в себе си всички полети, които се извършват. Свързан е с обект „Самолет“ и обект „Маршрут“. Въвежда се час и дата на излитане и пристигане, както и номерът на самия полет. Има информация и за статус, терминал и изход, която се визуализира на потребителите в портала. По-специфичното за този обект е, че за всеки запис трябва да се зададе освен цената за базовият билет, и цени за място от всяка класа и за различните видове багаж (ръчен и чекиран). Тези цени след това се използват при създаването на продуктите, свързани с полета.

За разпределението на екипажа се използва обектът „Член на екипаж“ (Flight Crew), който служи установяването на връзка от тип „много към много“ между обекта за полети и този за потребители на платформата (User). Освен връзки към двата обекта, има и поле, което описва ролята на съответния човек в екипажа (стюардеса, капитан, втори капитан). Спрямо записите в този обект се управляват правата, които определят кой потребител има достъп до данните на кой полет.

Друг обект, който е свързан с полетите, е „Текстово съобщение за полет“ (Flight Text Message), който се използва за реализацията на функционалността „Чат в реално време“, която ще бъде представена подробно по-късно. В него освен връзката към полет, се съдържа и информация за подателя на съобщението и самото съобщение.

Подобно на обекта, който съдържа информация за всяка седалка, е изграден и такъв за различните видове багаж (Baggage). За да се избегне повторение на данни, съществуват записи за всеки един от трите основни вида багаж, които биват реферирани по време на автоматизацията за създаване на продукти за всеки полет.

При завършване на процеса за закупуване на билети през портала се създава запис в поле „Резервация“ (Reservation), което също има връзка към обект „Полет“. Всяка резервация има статус, показващ дали тя е активна, отменена или чекирана, и обща сума, която се изчислява автоматично като събира стойността на всеки запис от тип „Артикул“ (Reservation Item), свързан към нея. От своя страна, всеки артикул освен връзка към дадена резервация, има връзка и към продукт, за да се знае какво точно е закупено. Съдържа също така и поле за стойност, защото не винаги цената, която се заплаща за определен продукт, съвпада с цената, посочена в продукта. Например, когато човек се чекира двадесет и четири

часа преди полета, може да използва, разпределеното му на случаен принцип, място безплатно. В допълнение, този обект има връзка и към обект „Пътник“ (Passenger), което осигурява правилното генериране на документи. Налага се използването на допълнителен билет, тъй като един потребител може да закупи билети за други хора, които не са регистрирани потребители в портала.

След подробното описание на всеки обект в следващите точки ще бъдат описани как влизат в употреба те при реализирането на различните функционалности на системата.

### **3.4 Процеси, реализирани на платформата**

Облачната услуга Salesforce предоставя разнообразни методи за реализацията на различни автоматизации, които да улеснят управлението на един бизнес. Създаването на тези процеси включва използването както на комбинацията между стандартни и персонализирани обекти, така и на технологии като: уеб компоненти, потоци, страници и други. Следващите точки имат за цел да представят детайлно тяхното осъществяване, така и как точно биха били полезни за компанията.

#### **3.4.1 Персонализирано приложение**

Платформата е изградена от няколко на брой стандартни приложения, всяко от които представлява различна функционалност на Salesforce. Тези приложения представляват всеки един от „облаците“, които са достъпни за платформата като например: Sales Cloud, Service Cloud и други. Невинаги това, което се съдържа в тях отговаря на нуждите на бизнеса, използващ платформата – или липсват раздели, които са необходими, или има ненужни такива. Затова, първата стъпка от разработката е създаването на персонализирано приложение, което да помогне на потребителите лесно да управляват и достъпват информацията, от която се нуждаят. При изграждането му има пълна свобода в това какво да включва и как да изглежда то. В допълнение, наличието на такова приложение улеснява задаването на правата на достъп измежду потребителите.

Персонализираната визуализация включва изборът на име, лого и основни цветове. За изображение за лого се поддържат форматите: PNG, JPG, BMP и GIF; като максималният размер не трябва да надвишава 5MB. Препоръчителният размер е 128x128 пиксела. Основният цвят се появява в навигационното меню на приложението. За текущата разработка е създадено уникално лого с помощта на продукта за обработка на векторни изображения – Adobe Illustrator. То представлява абривиатурата „FMI“, която е образувана от началните букви на думите, участващи в мотото, на проекта: „Fly me instantly“.

За всяко персонализирано приложение могат да бъдат добавяни раздели, които да участват в главната навигация. Съществуват няколко разновидности раздели, но в проекта

се използват само два от тях – стандартните, които са базирани на обекти, (Object Tabs) и такива, които в себе си съдържат избран уеб компонент, (Lightning Component Tabs).



**Фигура 6. Лого, заглавие и навигация на персонализирано приложение**

### 3.4.2 Създаване на продукти при добавяне на нов полет

За продажбата на билети, места и багаж през портала се използва информацията в записите от обекта „Продукт“ (Product). За всеки запис има поле „Тип“, което може да приема една от стойностите: базов билет, място или багаж. Още при създаването на полет се изисква информация за цените на всяко от тези неща, за да могат да бъдат правилно разпределени цените. За местата има четири вида цени спрямо класата, в която се намират. Докато при багажа, са три – спрямо неговия вид.

Реализирана е автоматизация с помощта на тригер, която при създаване на полет взима цените попълнени в записа и създава множество продукти, които представляват: един продукт за базовия билет, три продукта от тип „багаж“ и толкова продукти от тип „място“, колкото са на брой местата в самолета, избран да изпълнява полета, като се съобразява от каква класа е всяко място, за да разпредели правилната цена.

Това е постигнато с един тригер на обекта „Полет“ (Flight) и два помощни класа, в които е разпределена логиката за изпълнение. При създаване на тригера се задава за кой обект е той и в какви ситуации да се изпълнява. В настоящия обект се използва след създаване и след промяна на запис. Следвайки добрите практики в тялото на тригера не следва да има логика. По тази причина директно се извиква метод от класа *FlightTriggerHandler*, който е отговорен за разпределението на логиката спрямо това от какво действие е предизвикан той.

```
trigger FlightTrigger on Flight__c ( after insert, after update ) {  
  
    FlightTriggerHandler.handleTrigger(Trigger.new, Trigger.old, Trigger.newMap,  
    Trigger.oldMap, Trigger.operationType);  
  
}
```

#### **Код 3. Тригер на обект "Полет"**

Параметърът *Trigger.operationType* съдържа от какъв вид е тригерът. Тази стойност се използва за определяне на методите, които трябва да бъдат изпълнени. Цялата помощна логика се намира в класа *FlightTriggerHelper*. Това разделение е приложено, за да бъде добре структуриран, изчистен и лесен за четене написаният код.

```
switch on triggerEvent {  
  
    when AFTER_INSERT {
```



```

        FlightTriggerHelper.actionAfterInsert(newFlights);
    }

    when AFTER_UPDATE {
        FlightTriggerHelper.actionAfterUpdate(oldFlights, newFlightsMap);
    }
}

```

#### Код 4. Разпределение на логиката

Тъй като един тригер може да бъде предизвикан от няколко записа наведнъж, логиката трябва да бъде приложена за всеки един от тях. Следователно следва листът с полети да бъде обходен и за всеки полет да се извиква един и същ метод, който да създаде продуктите за него. Този метод носи името *createSeatProducts* и като параметър приема стойност от тип „Полет“. В него се създава лист за всички продукти, след което се добавят елементи от три отделни метода – за създаване на продукт за базовия билет, за продуктите от тип „място“ и за тези от тип „багаж“. След като са добавени в списъка те се създават и в базата данни.

```

private static void createFlightProducts(Flight__c flight){
    List<Product2> flightProducts = new List<Product2>();
    flightProducts.add(createBaseTicketProduct(flight));
    flightProducts.addAll(createSeatProducts(flight));
    flightProducts.addAll(createBaggageProducts(flight));

    insert flightProducts;
}

```

#### Код 5. Метод за създаване на продукти

Всеки продукт съдържа име, тип, цена, връзка към полета, за който се създава, и се вдига флаг, че е активен. За продуктите от тип „място“ и „багаж“ се създава и връзка към съответните записи за място и багаж от персонализираните обекти за тях.

При създаването на продукти за местата първоначално се създава колекция от типа „ключ - стойност“, като ключове са имената на класите, а стойностите са цените от съответното поле, взети от записа на полета.

```

Map<String, Decimal> seatPrices = new Map<String, Decimal>();
seatPrices.put('Economy', flight.Economy_Class_Seat_Price__c);
seatPrices.put('Premium Economy', flight.Premium_Economy_Class_Seat_Price__c);
seatPrices.put('Business', flight.Business_Class_Seat_Price__c);
seatPrices.put('First Class', flight.First_Class_Seat_Price__c);

```

#### Код 6. Създаване на колекция за цените на местата

За взимането на записите, отговарящи на местата, първо се взима уникалният идентификатор на модела на самолет спрямо това кой самолет е избран за съответния полет и след това с него се стига до записите на местата.



```

Aircraft__c aircraft = [
    SELECT Id, Aircraft_Model__c
    FROM Aircraft__c
    WHERE Id =: flight.Aircraft__c
];
List<Aircraft_Seat__c> seats = new List<Aircraft_Seat__c>([
    SELECT Id, Name, Class__c
    FROM Aircraft_Seat__c
    WHERE Aircraft_Model__c =: aircraft.Aircraft_Model__c
]);

```

#### *Код 7. Извличане на записи за места спрямо вида самолет*

След като вече са налични записите започва итерация по тях и създаване на продукт за всяко място като цената се взема от колекцията за цени чрез нейния ключ, съответстващ на стойността в полето *Class\_\_c* на мястото.

За продуктите от тип „багаж“ се повтаря почти същата процедура, с разликата, че записите, по които се създават продуктите са от обекта „Багаж“ (Baggage).

### **3.4.3 Актуализация на продукти при редакция на полет**

Авиокомпаниите често променят цените на билетите на своите полети. Няма как да се разчита на това, че няма да настъпи промяна. Затова се налага създаването на автоматизация, която да поддържа продуктите свързани с полета актуални.

Изводката, представена в *Код 4*, разкрива, че е реализирана и логика, която да бъде изпълнена, ако тригерът е от тип „след промяна“.

Логиката покрива случаите, когато има промяна в някое от полетата за цени. Тази проверка се извършва като се сравняват старият и новият запис. Главният метод приема лист от старите записи и колекция от тип „ключ - стойност“. Итерирайки по листа, на метода за проверка за редакция се подава текущия запис и новата му версия, взета от колекцията, използвайки уникалния му идентификатор като ключ.

Имайки двата записа, започват проверки за всяко поле, представляващо цена, затова дали има разлика в стойностите между двата записа. Спрямо вида на продукта се изпълнява различна логика, ако има промяна. Предварително са създадени два списъка – един с всички съществуващи продукти за съответния полет и един празен, който ще пази вече актуализираните продукти. Методите за промяна на цената на продуктите си приличат - и трите приемат като параметри списъка от всички продукти и цена. Тези за място и багаж приемат и съответно вид класа и вид багаж. В същината си представляват минаване на всеки продукт от списъка и който има съвпадение с входните параметри бива редактиран и добавян в списък, който в края на цикъла се връща към главния метод.

След като са минати всички проверки и актуализации, ако е имало такива, списъкът с всички променени продукти се използва за да се актуализират всички съответни записи в базата данни.

Освен при редакция на цените, автоматизация се прилага и при смяната на самолета, който ще изпълнява полета, ако той е с различен модел от предишният, тъй като това означава смяна в разпределението на местата. В такава ситуация се изтриват всички продукти от тип „място“ и биват създадени на ново, използвайки метода, който осъществява това и при създаването на нов полет.

#### **3.4.4 Изпращане на имейл при добавяне на член в екипажа**

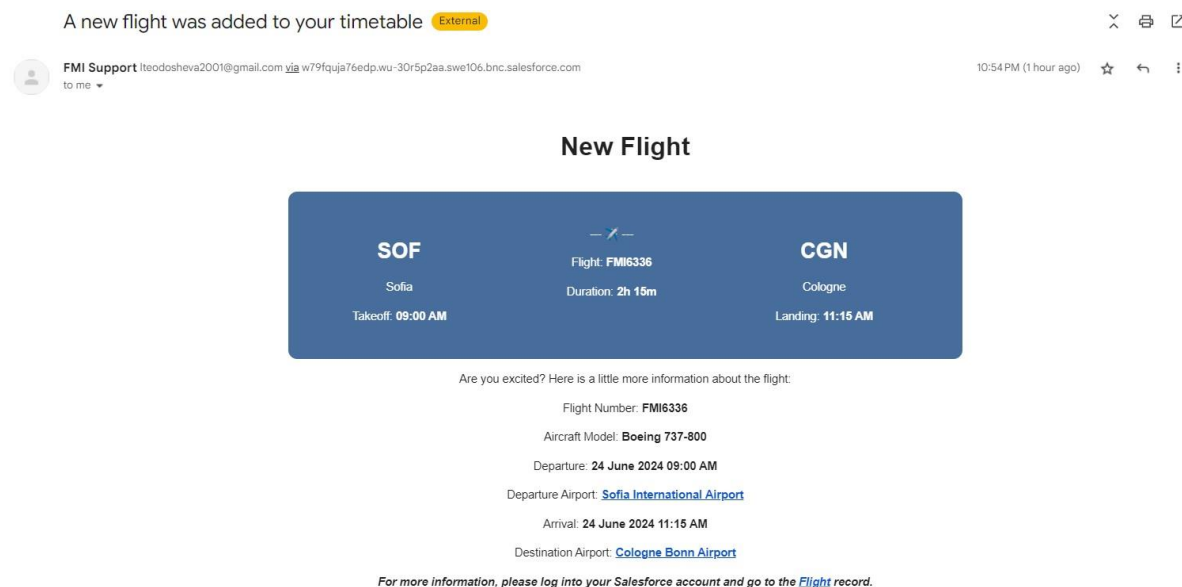
Една от основните реализирани функционалности са автоматизирани имейл нотификации при разпределянето на екипажа по различните полети. Когато потребител е добавен в екипажа на някой полет, той веднага получава имейл, съдържащ цялата информация за полета. Тази функционалност е изключително важна, тъй като помага за организацията на екипажа. По този начин всеки служител е наясно с предстоящите си ангажименти и ще може да се подготви за тях. В допълнение, улеснява потребителите като събира информацията, от която се нуждаят, на едно място, което я прави лесно достъпна.

Това е реализирано чрез предварително подготвен имейл шаблон и логика, която се изпълнява при създаването на запис в обекта *Flight Crew*, отговарящ за разпределението на екипаж.

Шаблонът е изграден чрез *Classic Email Template*, зад който стои *HTML* структура, описваща елементите на имейла и техния стил. Целият шаблон е ограден с тага `<messaging:emailTemplate>`. Той приема параметри параметрите: тема, вид на получател (потребител, контакт и други), вид на записа, с който е свързан, тоест от който ще черпи информация и *reply to* имейл. Следващият таг е - `<messaging:htmlEmailBody>`, който съдържа в себе си структурата на имейла – какви секции да има, кои полета да използва, за показване на информацията, как да са подредени и как да изглеждат те.

Стилът се задава в *style*, като е съобразено какви точно стилове могат да бъдат използвани, тъй като имейл клиентите се различават по това кои стилове поддържат. Например, един от сравнително новите и гъвкави *CSS* модули – *Flexbox*, който се поддържа на почти всеки браузър [6], не се препоръчва за използване при работа с имейл клиента *Outlook*, поради факта, че той филтрира всички свойства на модула, като оставя само неговото разположение, прилагайки стойностите по подразбиране [7]. Във връзка с това, при изграждането на шаблона за хоризонталната подредба на секции се използва таблица, като всяка колона от нея представлява и колона в разположението на данните.

Тялото на имейла е изградено от информация за полета, към който потребителят е бил добавен, като например: номер на полет, от кое и до кое летище е полета, час и дата на тръгване и пристигане и модел на самолета.



**Фигура 7. Имейл, получен от член на екипа при добавянето му към полет**

Освен статична информация, електронното съобщение включва в себе си и връзка към точната локация на летищата, за която са използвани стойностите на географската дължина и ширина, зададени в записите на всяко летище. В допълнение има и връзка към записа на самия полет в платформата, за да има опция за бързо извличане на допълнителна информация за него.

```
<a href="https://www.google.com/maps/place/{!relatedTo.Route__r.Departure_Airport__r.Location__Latitude__s},{!relatedTo.Route__r.Departure_Airport__r.Location__Longitude__s}">{!relatedTo.Route__r.Departure_Airport__r.Name}</a>
```

**Код 8. Генериране на връзка за точна локация с координати**

Изпращането на имейл нотификациите е реализирано чрез тригер, който се активира при създаването на запис в обекта, свързващ потребител с полет. За разлика от тригера, представен в точка 3.4.2, този се използва само при условието – „след запис“. След получаването на списък от създадени записи, се извиква метод, който създава имейл за всеки и ги изпраща наведнъж. Създаването на имейл се осъществява чрез създаване на нова инстанция от класа *Messaging.SingleEmailMessage*. Използвайки методите, които предоставя класът, се задават параметри като – шаблон, визуализация на името на изпращача, получател, запис, който да се използва за извличане на информация в шаблона и флаг дали да се създаде запис за дейност, което да пази история на имейлите. За намирането на съответният шаблон се прави заявка, която го намира спрямо неговото име. За получател се задава уникалният идентификатор на потребителя, който се съдържа в

полето *Member* на текущия запис, а за извличане на информацията се подава това на полета от *Flight*. Тъй като не се изисква проследяване на историята на имейлите, флагът за създаване на запис при всеки изпратен имейл не е вдигнат.

```
Messaging.SingleEmailMessage emailNotification = new Messaging.SingleEmailMessage();
emailNotification.setTemplateId(emailTemplate.Id);
emailNotification.setSenderDisplayName('FMI Support');
emailNotification.setTargetObjectId(fc.Member__c);
emailNotification.setWhatId(fc.Flight__c);
emailNotification.saveAsActivity = false;
notificationsToSend.add(emailNotification);
```

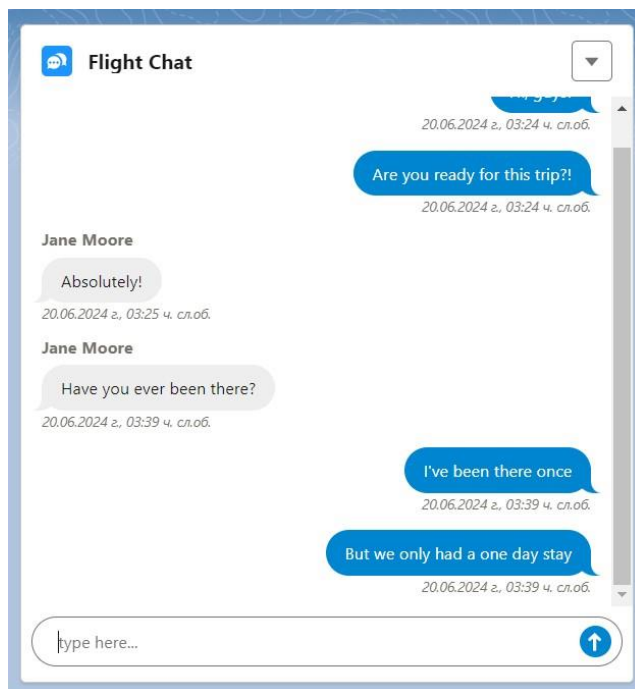
#### **Код 9. Създаване на имейл инстанция и задаване на параметри**

При изпращането на всички съобщения, методът връща резултати от изпращането под вида на списък с елементи от тип *Messaging.SendEmailResult*. Итерирайки по този списък може да се проследи, ако някой имейл не е бил изпратен и каква е причината за неговия провал.

### **3.4.5 Чат в реално време**

За да се подобри комуникацията между членовете на всеки екипаж, е разработена персонализирана чат функционалност с помощта на *Lightning Web Component*. Този компонент е стратегически разположен на детайлната страница за всеки запис, представляващ полет. Това позволява на потребителите, които са разпределени към този полет да комуникират безпроблемно като същевременно виждат и всичката налична информация за полета. В допълнение, когато каналът за комуникация е директно поставен на същата страница, всички дискусии са събрани на едно място и могат да бъдат достъпвани по-лесно. Тази функционалност не само подобрява съвместната работа в реално време, а и помага със запазване на история на комуникацията, свързана със съответен полет.

Аспектът за комуникация в реално време е постигнат чрез използването на *Platform Events*, където има елемент на публикуване на съобщение по даден канал, което бива прихванато от компонента, разработен за чата, и по този начин информацията бива презаредена автоматично.



**Фигура 8. Компонент за чат в реално време**

Самият компонент съдържа в себе си изображение, което се асоциира с чат дискусия, заглавие, бутон, който при натискане визуализира членовете на чата, секция със съобщения, поле за писане на съобщение и бутон за изпращане.

За извличане на информация за останалите потребители в чата се прави заявка, която да вземе записите от обекта *Flight Crew*, като се филтрират по уникалния идентификатор за запис на съответния полет и по това потребителят да не е този, който извършва самата заявка. От тези записи се използва полето *Member*, което съдържа информацията за всеки потребител. В случаят се използва само тяхното име.

Всички съобщения се пазят в обект „Текстово съобщение“ (Text Message), където се държи информация за полета, за който са изпратени, потребителят, който ги е изпратил и самото съдържание на съобщението. Тъй като компонента е поставен на детайлна страница, използвайки декоратора `@api`, автоматично може да се вземе уникалният идентификатор на записа, който е отворен, като се дефинира променлива, носеща името `recordId`. Чрез тази стойност се взимат всички записи със съобщения, филтрирани по полет, и се визуализират в хронологията на чата при началното зареждане на компонента. Специфичното е, че списъкът трябва да е в низходящ ред спрямо датата на създаване, тоест последно записаното съобщение да бъде на първо място. Това се използва, за да се постигне ефектът на истински чат, където винаги контейнерът, съдържащ съобщенията е превъртян до дъното си, за да показва най-новите съобщения, и за проследяване на по-ранна дискусията, трябва да се плъзга нагоре. Ако съдържанието на един контейнер е по-дълго, отколкото височината, която му е зададена, то по подразбиране винаги показва информацията, която е на върха, но

ако се използва свойството **flex-direction** със стойност **column-reverse**, това поведение бива променено и вече стартираща точка за визуализация е дъното на съдържанието. Тази стойност също така обръща реда на елементите в един контейнер, затова първите стойности, в случаят най-актуалните съобщения, ще бъдат показвани най-отдолу. Преди самата визуализация, списъкът със съобщения бива обработен като за всеки запис се запазва уникален идентификатор на съобщението, изпращач, час и дата на изпращане в точно определен формат, описан с параметри на елемента, който ги визуализира, и флаг за това дали съответното съобщение е изпратено от човека, който в момента гледа компонента или от друг. Тази стойност се задава чрез сравнението на уникалния идентификатор на изпращача и този на текущия потребител. Тя се използва за определяне на това как да се визуализира съобщението в хронологията, като спрямо нея се зарежда различна част от *HTML* кода, използвайки параметрите за условно зараждане на информация – **lwc:if** и **lwc:else**; като на първото се подава променлива, на която да се провери стойността.

```
<template for:each={messages} for:item="message">
  <div lwc:if={message.IsMine} key={message.Id} class="textMessageContainer myMessageContainer">
    <div class="message myMessage">
      {message.TextMessage}
    </div>
    <div class="timestamp">
      <lightning-formatted-date-time
        value={message.Timestamp}
        year="numeric" month="2-digit" day="2-digit"
        hour="2-digit" minute="2-digit" hour12="false">
      </lightning-formatted-date-time>
    </div>
  </div>
  <div lwc:else key={message.Id} class="textMessageContainer">
    <p class="senderName">
      {message.SenderName}
    </p>
    <div class="message yourMessage">
      {message.TextMessage}
    </div>
    <div class="timestamp">
      <lightning-formatted-date-time
        value={message.Timestamp}
        year="numeric" month="2-digit" day="2-digit"
        hour="2-digit" minute="2-digit" hour12="false">
      </lightning-formatted-date-time>
    </div>
  </div>
</template>
```

#### **Код 10. Визуализация на съобщения в хронологията**

При изпращане на съобщение от компонента се извиква метод, който създава запис в обекта *Flight Text Message* . Реализирана е автоматизация, която се изпълнява, когато се

това се случи. Тя е изградена от две дейности – публикуване на събития и изпращане на известия.

Създадено е персонализирано събитие за платформата, за което са създадени и набор от персонализирани полета, които пазят информация като – уникален идентификатор на полет и изпращач, текстово съобщение и дата и час на създаването му. Когато то е налично, може да се реализира логика базирана на събития, като има, в текущия случай, един изпращач и един получател. Това, което прави автоматизацията по-горе, е да създаде по едно събитие от създадения вече вид за всеки запис в обекта за съобщения като го попълни със съответната информация за него. След като са създадени събитията, те биват публикувани на така наречения *event bus*, който се явява като опашка, в която всяко събитие се подрежда в строго хронологичен ред спрямо неговото публикуване. За публикуването се използва - **EventBus.publish(events)**. Като параметър, който е подаден представлява списък от събитията, които са създадени преди това.

Получаването на тези събития се осъществява в компонента за чата, като се използва модулът *empApi*, за абониране за съответния канал. Името на канала се състои от името на платформеното събитие, което е създадено предварително. В проекта това име е */event/New\_Flight\_Message\_\_e*. Освен името, трябва да се подаде позицията на събитие, от което да започне да слуша и функция, която се изпълнява при получаване на съобщение. При успешно абониране се връща *promise*, който може да се използва за прекратяване на абонамента към този канал. Самото абониране се извършва още при стартиране на компонента във функцията *ConnectedCallback*. При получаване на съобщение се проверява дали този запис е за текущия полет, ако е – нов запис бива добавен в списъка със съобщенията в чата на първа позиция.

```
handleSubscribe() {
  const messageCallback = (response) => {
    if (response.data.payload.Flight_Id__c === this.recordId){
      this.messages.unshift(
        {
          Id: response.data.event.replayId,
          SenderName: response.data.payload.Sender_Name__c,
          TextMessage: response.data.payload.Text_Message__c,
          Timestamp: response.data.payload.Timestamp__c,
          IsMine: this.userId===response.data.payload.Sender_Id__c
        }
      );
    }
  };
};
```

```

    subscribe(this.channelName, -1, messageCallback).then(response => {
        this.subscription = response;
    });
}

```

#### Код 11. Абониране за канал

За да са винаги информирани потребителите, когато ново съобщение е добавено в дискусия за полет, на който те са част от екипажа, е реализирана и система за известяване. Подобно на събитията в платформата, така и за нотификации могат да бъдат създавани персонализирани видове. За всяка такава може да се избере до кои устройства да бъде изпращана тя – десктоп или мобилно, тоест дали да са само в платформата или да стигат и до мобилното приложение. При избор за получаване и в мобилното приложение, може да се избере и за определената операционна система – *IOS* и *Android*. В настоящия проект всички опции са отбелязани, за да може всеки, каквото и устройство да използва, да получава известия при ново съобщение в чата.

След логиката за публикуване на събитие се изпълнява тази за изпращане на известия. За събиране на потребителите, които трябва да получат известие се използва същата логика, която е използвана за визуализиране на останалите членове на чата. Освен списък с уникалните идентификатори на тези потребители, за изграждането на нотификация са нужни и заглавие, тяло, вид на нотификацията (обяснено по-горе) и опционално може да се подаде запис, към който да бъде отведен потребителят при натискането и. В този случай това ще бъде записа на полета, за който е изпратен съобщение, тъй като на тази страница се намира и чата с всички съобщения.

```

Set<String> recipientsIds = FlightChatController.getCrewMembers(message.Flight__c,
message.Sender__c);

CustomNotificationType notificationType =
[SELECT Id, DeveloperName
FROM CustomNotificationType
WHERE DeveloperName='New_Flight_Chat_Message'];

Messaging.CustomNotification notification = new Messaging.CustomNotification();
notification.setTitle('New crew message: ' + message.Flight_Name__c);
notification.setBody(message.Sender_Name__c + ': ' + message.Text_Message__c);

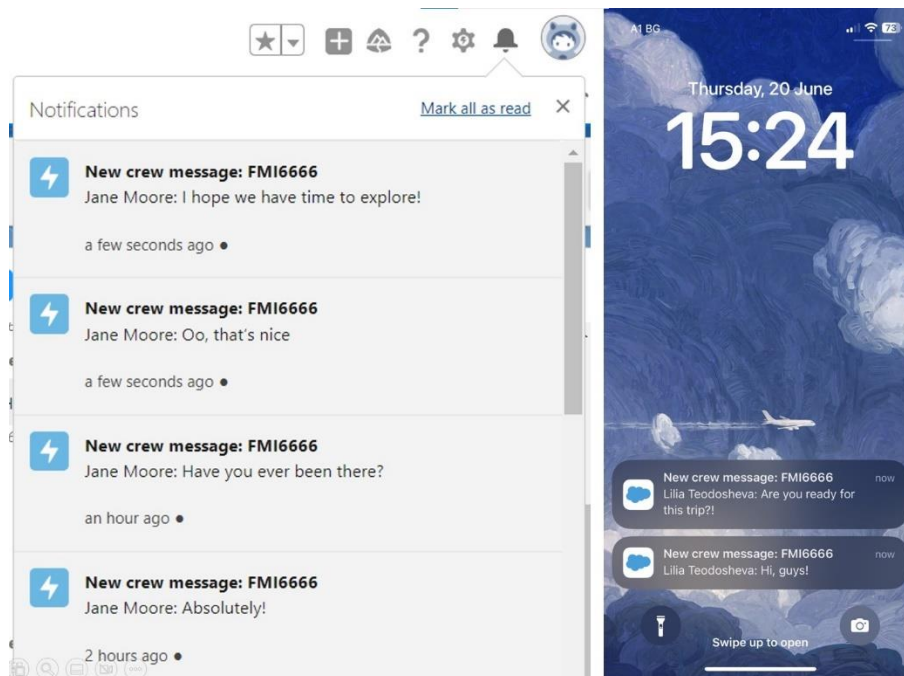
notification.setNotificationTypeId(notificationType.Id);
notification.setTargetId(message.Flight__c);

try {
    notification.send(recipientsIds);
}
catch (Exception e) {
    System.debug('Problem sending notification: ' + e.getMessage());
}

```

#### Код 12. Създаване и изпращане на известие за ново съобщение





Фигура 9. Известия в платформата и от мобилното приложение

### 3.4.6 Изтриване на съобщения след определено време

Ако компанията реши, че за нея е приоритет да предостави възможност за бърза и безпроблемна комуникация между членовете на екипажа, без да е нужно да поддържа история затова, може да се възползва от автоматизация, която се задейства на всеки час, за да трие съобщения по-стари от определен брой часове, например – двадесет и четири часа. Информацията за тези часове е изнесена в персонализирани метаданни, за да бъде променяна от администратора на системата без намесата на програмист. Този подход осигурява фокус върху актуалните дискусии, поддържайки оптимална производителност като се избягва работата с прекалено много записи.

Реализираната автоматизация е изградена от комбинация между *Batch Job* и *Scheduled Job*. И двете представляват разновидности на *Apex*. Нуждата от *Batch Job* е породена от потенциалния брой записи, които могат да бъдат натрупани с времето. Тази функционалност позволява работата с голям обем от данни като ги разпределя на парчета, за да не натоварва системата. В допълнение, този метод помага с придържането към лимитите, които са приложени от Salesforce, за брой обработени записи в една транзакция. За извличането на записите, които трябва да бъдат изтрети, се използва броят часове, запазени в персонализираните метаданни. Тези часове се вадят от сегашния момент и получената стойност бива използвана за сравнение с датата на създаване на всеки запис. Списъкът с филтрирани записи се подава на метода *execute*, където те биват изтрети от базата с помощта на *DML* операция. Ако се получи грешка, тя се записва в лога.

```

private static Integer getRetentionHours() {
    FlightTextMessageSettings__mdt settings = [SELECT RetentionHours__c FROM
    FlightTextMessageSettings__mdt LIMIT 1];
    return Integer.valueOf(settings.RetentionHours__c);
}

global Database.QueryLocator start(Database.BatchableContext BC) {
    Integer retentionHours = getRetentionHours();
    Datetime cutoffDate = System.now().addHours(-retentionHours);
    String query = 'SELECT Id FROM Flight_Text_Message__c WHERE CreatedDate <
: cutoffDate';
    return Database.getQueryLocator(query);
}

global void execute(Database.BatchableContext BC, List<Flight_Text_Message__c> scope)
{
    try {
        delete scope;
    } catch (DmlException e) {
        System.debug('Error deleting records: ' + e.getMessage());
    }
}

```

#### Код 13. Batch Job за изтриване на съобщения

Причината за използване на *Scheduled Job* е, за да се отстрани нуждата от ръчна намеса при изтриването на записи. В тялото си той съдържа метод *execute*, в който се изпълнява стартирането на съответния *Batch Job* чрез негова инстанция.

```

global void execute(SchedulableContext SC) {
    DeleteOldFlightTextMessagesBatch batch = new DeleteOldFlightTextMessagesBatch();
    Database.executeBatch(batch);
}

```

#### Код 14. Scheduled Job за изтриване на съобщения

Регулярното разчистване на тези данни помага за по-добро управление на мястото в хранилището и поддържането на бързината на базата данни. По тази причина, след като вече има съществуващ *Scheduled Job*, той може да бъде стартиран като се зададе негова инстанция, име и *Cron Expression*, задаващ на каква честота да се изпълнява логиката. В текущия случай, тази функционалност е настроена да се изпълнява на всеки час.

```

ScheduleDeleteOldFlightTextMessages scheduleJob = new
ScheduleDeleteOldFlightTextMessages();
String sch = '0 0 * * * ?';
System.schedule('Delete Old Flight Text Messages', sch, scheduleJob);

```

#### Код 15. Стартиране на Scheduled Job

В случай, че компанията реши, че вече не желае да се изпълнява тази автоматизация, администратор лесно може да я прекрати използвайки панела за *Scheduled Job* в *Setup*. Тя става налична в панела още при нейното стартирането. Визуализира се с името, което е било зададено при настройването. От панела може да се редактира, изтрие или постави на пауза.

Също така има информация под формата дата и час за това кога е създадена, последното ѝ изпълнение и следващото и такова. Наличието на такъв панел е практично, тъй като помага за лесно управление на всякакви такива автоматизации, на които не им е нужна човешка намеса.

**SETUP Scheduled Jobs**

**All Scheduled Jobs** [Help for this Page](#)

The All Scheduled Jobs page lists all of the jobs scheduled by your users. Multiple job types may display on this page. You can delete scheduled jobs if you have the permission to do so.

**Percentage of Scheduled Jobs Used: 1%**  
You have currently used 1 scheduled Apex jobs out of an allowed organization limit of 100 active or scheduled jobs. To learn about how this limit is calculated and what contributes to it see the [Lightning Platform Apex Limits](#) topic.

View: **All Scheduled Jobs** [Create New View](#)

A B В Г Д Е Ж З И Й К Л М Н О П Р С Т У Ф Х Ц Ч Ш Щ Ю Я **All**

Action	Job Name	Submitted By	Submitted	Started	Next Scheduled Run	Type	Cron Trigger ID
Del	CommIncrementalSitemapJob-00DWU0000030r5p-0DMWU0000002wXm	Teodosheva, Lilia	7.05.2024 г., 12:53 ч.	21.06.2024 г., 3:30 ч.	22.06.2024 г., 3:30 ч.	Sitemap SEO Incremental Job	08eWU000001sM7t
Del	CommSitemapJob-00DWU0000030r5p-0DMWU0000002wXm	Teodosheva, Lilia	7.05.2024 г., 12:53 ч.	16.06.2024 г., 3:47 ч.	23.06.2024 г., 3:47 ч.	Sitemap SEO Generation Job	08eWU000001sM7s
Manage   Del   Pause Job	Delete Old Flight Text Messages	Teodosheva, Lilia	21.06.2024 г., 23:26 ч.	22.06.2024 г., 0:00 ч.	22.06.2024 г., 1:00 ч.	Scheduled Apex	08eWU0000036sf7
Del	Metalytics Data Loader Job for Org : 00DWU0000030r5p	User Integration	17.03.2024 г., 1:42 ч.	21.06.2024 г., 15:37 ч.	22.06.2024 г., 15:37 ч.	Autonomous Data Loader Job	08eWU000000hwGr

**Фигура 10. Панел със Scheduled Jobs**

### 3.4.7 Календар с полети

За всеки служител е важно да има достъп до графика си по всяко време. Една такава функционалност предоставя обобщен поглед над полетите, към които съответният потребител е разпределен. Представянето на информацията за планираните полети под формата на календар подобрява координацията между екипа. Чрез избор на определен месец потребителя лесно може да навигира задълженията си като има достъп до вече осъществили се такива и тези, които предстоят. Освен това, всеки изобразен полет, крие зад себе си директна връзка към запис на полета, където потребителят може да види детайлна информация за него. В допълнение към това, е включена и функционалност, която позволява на екипажа да получи като *PDF* файл цялостното разпределение на полетите за текущия месец, само с едно натискане на бутон. Тази функционалност е полезна, когато потребителят иска да види своя график без да има връзка към системата.

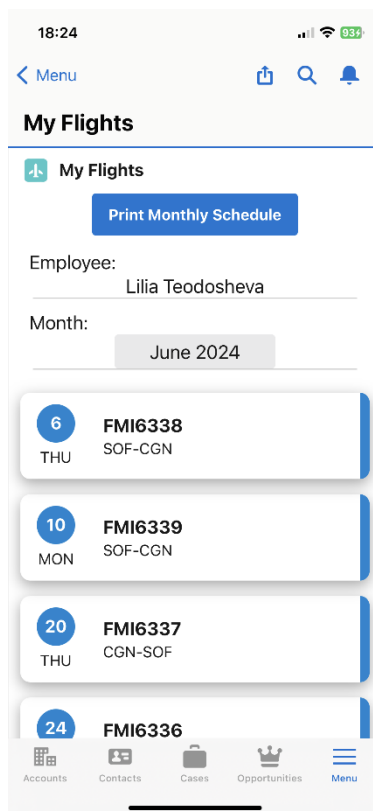
Целият компонент е изкаран в отделен раздел, на който е разположен само той. В себе си той включва заглавна част и тяло. В заглавната част се визуализира името на текущия потребител, което се взема динамично спрямо човека, който е влязъл в системата и е отворил компонента. Също така има поставен елемент за входни данни от тип „месец“, зареден по подразбиране с настоящия месец, като потребителят може да избира друг и по този начин да вижда графика си за избрания месец. В тази част се намира и бутонът, който генерира справка с полетите, към които служителят е разпределен, и изпраща файл на неговата електронна поща.

Тялото на компонента има две разновидности спрямо това на какво устройство се гледа компонента. На таблет и компютър, тоест устройства с по-голяма от 768 пиксела ширина, разписанието се зарежда под формата на календар, с клетка за всеки ден, като в дните, в които полет се появява оцветена ивица с номера на полета. При натискане потребителя бива пренасочен към записа на полета. Записите за дните и полетите от съответният месец се взимат чрез метод в *Apex* контролер, който се извиква през *JavaScript* частта на компонента като се използва *@wire* декоратор, за да може да се изпълни още при зареждането на компонента. На метода се подава като параметър избраният месец и тъй като стойността на параметърът е зададена като реактивна, този метод ще се изпълнява автоматично при всяка промяна на стойността на променливата, което отстранява нуждата от допълнителна логика за негово извикване. Друг плюс на този вид методи е, че веднъж изпълнен с определен параметър, данните се запазват в кеш паметта, позволявайки по-бързо зареждане при връщане към вече избрана стойност. За конструирането на списъка от данни се използва колекция от тип „ключ - стойност“, като за ключ се използва число на ден от месеца, а стойността е лист от новосъздаден клас-обект, за да пасне на информация, която трябва да се събере. Първата стъпка е чрез определяне на първата дата от месеца и броят дни в него да се напълни колекция със запис за всеки ден, за да се визуализира правилно календарът. След това след взимането на всички записи за полети, отговарящи на параметрите – месец и член на екипажа, този списък да се обходи като добавя записите към списъкът за съответния ден, криещ се зад ключа, отговарящ на датата, от вече създадената колекция. За всеки полет се връща информация като: уникален идентификатор, име/номер на полета, дата на излитане, ден от седмицата с три букви и името на маршрута. При наличието на първият и последният ден от месеца се определя кой ден от седмица се пада тази дата и чрез тази стойност се пресмята отстоянието за календара и накрая допълването на клетки, за да се получи пълен правоъгълник.

MONDAY	TUESDAY	WEDNESDAY	THURSDAY	FRIDAY	SATURDAY	SUNDAY
					1	2
3	4	5	6 FM6338	7	8	9
10 FM6339	11	12	13	14	15	16
17	18	19	20 FM6337	21	22	23

**Фигура 11. Календар функционалност за средни и големи екрани**

На мобилни устройства с по-малки екрани тази функционалност изглежда по малко по-различен начин. Вместо като календар, информацията се показва в „плочки“, подредени в колона. Във всяка една се визуализира денят като число и трибуквен текст, номер на полета и името на маршрута, който ще изпълнява той. Друга разлика е, че за разлика от календара, тук се визуализират само дните, в които има разпределен полет, за да не се „утежнява“ изгледа и да затруднява потребителя да намери информацията, която търси. При натискане на някоя от тях, потребителят отново бива отвеждан към записа на съответният полет. Това се извършва с функцията **NavigationMixin** от модула **navigation**, който е един от главните подходи за навигиране през различните страници на *Salesforce*. На функцията се подава видът страница, към който трябва да се препрати потребителят, в случая – детайлна страница; и атрибути, които в себе си съдържат – уникален идентификатор на записа, името на обекта и вид дейност (изглед, редактиране). Уникалният идентификатор на полета се задава в *HTML* като **data-id** параметър на елемента и при натискането му се достъпва, използвайки **event.currentTarget.dataset.id**.



**Фигура 12. Календар функционалност за малки екрани**

Смяната между различните изгледи е реализирана с помощта на *CSS* стилове и *Media Queries*. Като по-точно има два контейнера – един за големия календар и един за списъка с полети. Първоначално първия се вижда, а вторият е скрит, и когато се засече ширина на екрана до 768 пиксела, двете секции променят свойствата си, като втората става видима, а

първата се скрива. Това не забавя зареждането на елементите, тъй като е даден момент само една от секциите е видима.

Функционалността с изпращане на *PDF* файл с графика за текущия месец до електронната поща на потребителят заявил тази справка, е постигната чрез използването на *Visualforce* страница, която се използва както за шаблон, така и за генерирането на файла. Страницата е изградена от лого на компанията, което е вградено като изображение, качено като статичен ресурс в *Salesforce*, заглавие и таблица, която е реализирана чрез компонент, предствалващ таблица, който може да итерира през лист с до десет хиляди записа [8]. На компонента се подава лист, чиито стойности трябва да визуализира, като предварително се посочват колоните на таблицата с тяхното заглавие и каква стойност трябва да съдържат те. Възможността на една *Visualforce* страница да бъде превърната в *PDF* файл се определя от параметъра **renderAs**, който трябва да съдържа стойността **pdf**. Самото извличане на данни, които да се използват в таблицата, се осъществява в контролера, който е свързан към страницата. При нейното извикването, той се изпълнява автоматично и по този начин шаблонът бива попълнен.

За да бъде прикачен *PDF* файлът към имейлът, който ще бъде изпратен до потребителят, първо се създава **PageReference** обект, който сочи към съответната страница. След това създава *Blob* променлива, към която бива присвоена бинарната стойност на генерирания *PDF*.

```
PageReference pdfPage = Page.FlightPDF;  
pdfPage.getParameters().put('id', currentUser.Id);  
Blob pdfBlob;  
try {  
    pdfBlob = pdfPage.getContentAsPDF();  
} catch (VisualforceException e) {  
    System.debug('Error rendering PDF: ' + e.getMessage());  
    return;  
}
```

#### *Код 16. Генериране на PDF файл*

За изпращането на имейла първо се създава променлива от тип **Messaging.EmailFileAttachment**, която представлява прикачен файл. На този файл се задава име, тип на файла и съдържанието му, което представлява *Blob* променливата, генерирана по-горе. Създаването на имейла включва задаване на адрес на получатели, в текущия случай само адресът на текущия потребител, задаване на тема и текст, който да се визуализира в тялото на имейлът, и задаване на прикачените файлове, където се добавя горесъздадения такъв. Цялата логика за генериране и изпращане на имейл е събрана в един метод, който се вика при натискането на бутон в компонента с календар функционалността. При успешно изпращане се появява съобщение, което посочва, че имейлът е изпратен и потребителят може да провери своята електронна поща, за да го достъпи.



Flight Assignments

Flight Name	Route	Departure	Departure City	Departure Country	Arrival	Destination City	Destination Country
FMI6338	SOF-CGN	6.06.2024 ., 20.00 .	Sofia	Bulgaria	6.06.2024 ., 22:15 .	Cologne	Germany
FMI6339	SOF-CGN	10.06.2024 ., 12:00 .	Sofia	Bulgaria	10.06.2024 ., 14:15 .	Cologne	Germany
FMI6337	CGN-SOF	20.06.2024 ., 15:30 .	Cologne	Germany	20.06.2024 ., 17:45 .	Sofia	Bulgaria
FMI6336	SOF-CGN	24.06.2024 ., 12:00 .	Sofia	Bulgaria	24.06.2024 ., 14:15 .	Cologne	Germany

Фигура 13. Генериран PDF файл

### 3.5 Процеси, реализирани в портала

За потребителите, които се явяват клиенти на авиокомпанията, е изграден портал с помощта на *Experience Cloud* директно върху вече съществуващата и настроена спрямо бизнес изискванията платформа. Целта на този портал е потребителите да могат да купуват самолетни билети и да управляват своите резервации. Освен това, те ще могат да виждат важни съобщения, касаещи авиокомпанията или пътната обстановка, като ще има и опция за проследяване на статуса на избрани полети. В допълнение, всеки потребител ще има профил, откъдето ще може да проследява цялата си история.

Изграждането на един такъв сайт става чрез използването на инструмента *Experience Builder*. Той работи на принципа на *drag-and-drop*, като компонентите, които могат да се поставят са както готови такива, предоставени от *Salesforce*, така и новосъздадени, от екипа, разработващ цялостната имплементация. Първото нещо, което трябва да се направи при създаване на такъв сайт, е да се избере някой от готовите шаблони, описани по-горе. В текущия случай е избран шаблона *Build Your Own*, тъй като порталът, който трябва да се изгради за клиента, включва основно персонализирани компоненти. Освен това, тъй като този шаблон се разполага на новата технология *Lightning Web Runtime*, което осигурява по-голямо бързодействие, и като цяло по-добри характеристики, отговарящи на „модерните“ уеб приложения. В допълнение, този вид шаблон предлага по-голям набор от готови компоненти, които ще помогнат с цялостното разпределение на компонентите по страниците и представянето на информация.

Построяването на един портал директно върху *CRM* системата позволява да възползването от всички нейни отличителни качества като – добре развитата им сигурност на данните, обновления в реално време и възможността за различни интеграции. Освен това



има пряко установена връзка с базата данни и не се налага цялостно изграждане и поддържане на подобна връзка. Всяка една връзка с чужда система се явява като слабо място, от където може да се „пробие“ и да се навреди на системата. Като говорим за обмен на данни и в двете посоки, една директна връзка към базата, намираща се на същото място, може само да подобри бързодействието на извличане и запис на информация. Така всяко действие на клиента се отразява директно в базата и благодарение на екосистемата, която *Salesforce* са успели да изградят, тази информация ще може да бъде използвана от всеки отдел, работещ на платформата, безпроблемно.

*Experience Builder* предоставя възможността за създаване на нови страници, на които могат да се задават различни теми, като например – страница без горен и долен колонтитул за такива като Регистрация, Влизане в профил и страници, в които присъстват тези секции. Освен това има различни видове страници спрямо това какво ще се разполага на тях – обикновена страница, обект страница и страница за *CMS* съдържание.

### 3.5.1 Локализация

Локализацията е от решаващо значение за постигането на удобно за потребителя и строго персонализирано изживяване във всяко глобално приложение. Осигурява възможността потребителят да може да работят със сайта на език, предпочитан от тях, което подобрява удовлетворението на клиентите. Използвайки функционалността на *Salesforce* за създаването на персонализирани етикети (*Custom Labels*), е реализирано цялостния превод на портала. Тези етикети позволяват запазването на текстови стойности, към които могат да се добавят преводи на избрани езици. Платформата позволява превеждането на стойностите към над 30 езика. Когато се използват такива етикети, те автоматично показват превода на стойността, съответстваща с езика, който е избран от потребителя. Ако липсва превод на даден език, се визуализира стойността, въведена по подразбиране. Този подход улеснява целия процес по локализация като позволява на портала да бъде лесно адаптивен към различни езици и по този начин да достигне повече хора.

Персонализираните етикети могат да се използват както в директно в компонентите *Experience builder*, така и в персонализираните такива. За тази цел се използва строго определен синтаксис. Например, за използването им в инструмента за изграждане на сайтове, етикетът трябва да изглежда по този начин: `{!Label.namespace.name}`. Докато, когато се използват в персонализираните компоненти има няколко стъпки. Първо трябва да бъдат внедрени в *JavaScript* файла на компонента по следния начин: `import NAME from '@salesforce/label/namespace.Name'`; . Това трябва да се направи за всеки етикет, който ще бъде използван в съответния компонент. След това чрез функцията `get labels()` се връща обект, на който за всеки етикет се създава атрибут, носещ неговата стойност. Накрая в HTML частта на компонента тези стойности могат да се използват по този начин: `{labels.Name}`.



Използвайки тази функционалност, се предотвратява нуждата от извършване на проверки по настройките на потребителя, за да се покаже съдържанието на съответния език. Изборът на език през портала се извършва чрез готов компонент, представляващ падащ списък с наличните езици. Опциите за езици, които се визуализират, кореспондират езиците, които са отключени на платформата. По този начин компанията може да избере кои езици да поддържа и да ги активира и деактивира по всяко време.

### 3.5.2 Система за управление на съдържание

*Salesforce CMS* е мощен инструмент, позволяващ лесно създаване, управление и разпределяне на съдържание по множество разнообразни канали, като един от възможните такива е *Experience Cloud* сайтове. Използването на тази система през *Experience Builder* е сравнително интуитивно като позволява безпроблемното внедряване на различно съдържание като текст, изображения, видеа и цели статии, изградени от предните три вида съдържание. Този продукт позволява дори хора, които не са експерти в сферата на технологиите, успешно да управляват всички ресурси, които ще бъдат използвани в портала. В допълнение, *Salesforce CMS* поддържа и добавянето на преводи към съдържанието. С тази функционалност администраторите могат да създават и управляват многоезично съдържание, осигурявайки консистентност и персонализирано преживяване на потребители от глобално ниво.

Подобно на персонализираните етикети, съдържанието, записано в *CMS* системата, може да бъде използвано както директно в *Experience Builder*, така и в персонализирани компоненти. Всеки запис в системата има уникален ключ, по който може да бъде различаван. Използвайки него в относителен път към ресурс, е единият начин, по който може да се вгради съдържание е персонализиран компонент. Например, ако трябва да се използва точно определено изображение, се създава променлива, която съдържа относителния път към ресурса, който се подава в параметъра **src** на *HTML* тага за изображение. Този път съдържа името на *Salesforce* организацията и ключ и изглежда по следния начин:  `'/FMI/sfsites/c/cms/delivery/media/MC40XXWLGfQ5BDBIAIAXPLRQYK5A'` .

Друг начин за използването на *CMS* в персонализирани компоненти е да се създаде параметър на компонент от тип **ContentReference**, който позволява на администратор да избере съдържание от системата, което динамично да се промени в компонента без да има нужда да се променя неговият код. Това позволява лесното персонализиране на компоненти без намесата на програмист. Тази техника е описана в компонентите, описани по-долу.

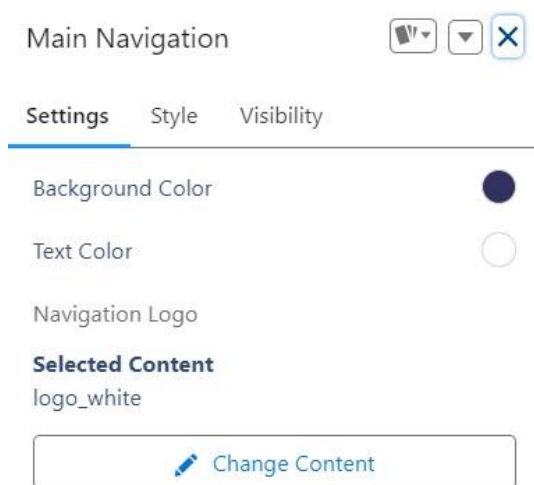
Последния вариант за използване на съдържание от системата в персонализирани компоненти е тяхното извличане с помощта на *Apex* код. Когато съдържанието е разпределено в колекции, те могат да бъдат взимани с помощта на различни методи от

*Connect REST API*. След това съдържанието може да бъде представено в компонента по разнообразни начини.

### 3.5.3 Дизайн

Дизайнът на един онлайн портал е от изключителна важност, тъй като от него зависи клиентското преживяване, ангажираността и цялостното удовлетворение. Портал с добър дизайн осигурява възможността на потребителите да навигират през неговите страници лесно, да намират информацията, която търсят, бързо и да работят с него безпроблемно. Също така, дизайнът пряко представлява същността и ценностите на една компания, което изгражда доверието на потребителите към нея. Ако един сайт изглежда естетически добре, той не само привлича повече хора, но предизвиква разговори за него между хората, а както са казали хората: „Доволният клиент е най-добрата реклама“.

Реализирането на този проект винаги е бил с мисълта да се направи максимално гъвкав като се има предвид, че един администратор ще може да прави промени по персонализираните компоненти без да трябва да търси програмист, който да оправя кода. Например, управлението на периода, в който да се пазят чат съобщенията, е изнесено в персонализирани метаданни и може да бъде променяно от там (описано в 3.4.6). По същия начин са изградени и компонентите за портала. Където е възможно, са създадени параметри на компонентите, които могат да бъдат управлявани през *Experience Builder*. Тези параметри могат да бъдат от тип: текст, дата, цвят, *CMS* съдържание и други. Компонента, изграден за навигацията, има три параметъра, които задават цвят на фон и текст и изображение за логото на компанията. Първоначално те идват със зададени стойности (не е задължително) и при промяна автоматично се отразява на компонента и е видима в рамките на секунди.



**Фигура 14. Панел с параметри на компонент**

Правейки компонентите гъвкави, откъм дизайн, позволява този продукт да бъде предлаган на различни компании без да трябва да се предприемат драстични промени по структурата. Освен това, дори да се използва от една компания, този подход е подходящ, когато се наложи малка и бърза промяна в случай на ребрандиране или нещо подобно.

### 3.5.4 Компонент за основна навигация

Избраният шаблон няма готов компонент за навигация, затова тя трябва да се реализира с персонализиран компонент. В текущия случай тя е изградена от две основни части – лого на компанията и списък с елементи, които се явяват връзки към други страници от сайта. Както е описано в предишната точка, изгледът на този компонент се управлява с помощта на параметри. Те се задават в *XML* файла, който е част от пакета на компонента. За всеки параметър се задава име, етикет, тип и стойност по подразбиране (пожелание). За да може да бъде използвана тяхната стойност в компонента, трябва да се създаде променлива, носеща името на параметъра и да бъде аотиран с декоратора - `@api`. Със стойностите за цвят на фон и текст се създават *CSS* променливи, които после да се преизползват за стилизирането на други елементи от компонента, а стойността от параметъра за *CMS* съдържание, която представлява уникалния ключ на ресурса, се създава относителен път, чрез който да се визуализира съответното изображение.

Контейнерът съдържащ елементите на навигационното меню има две разновидности спрямо това дали потребителят е влязъл в своя профил или е гост. Разликата е в последните две опции за страници като, когато потребителят е гост, той вижда опциите за влизане в профил и регистрация, а когато вече е вътре в системата, ще вижда избор за отваряне на неговия профил и излизане от него. Тази промяна се извършва спрямо детекция на *Salesforce* профила на потребителят. При зареждане на компонента автоматично се взема името на този профил. Ако името носи стойността и на *Guest User* профила, отреден за този сайт, означава потребителят е гост. Ако името му съответства с изборния за потребители на портала профил, например – *Customer Community Login Profile*, това е индикация, че потребителят е влязъл в своя профил.



Фигура 15. Основна навигация

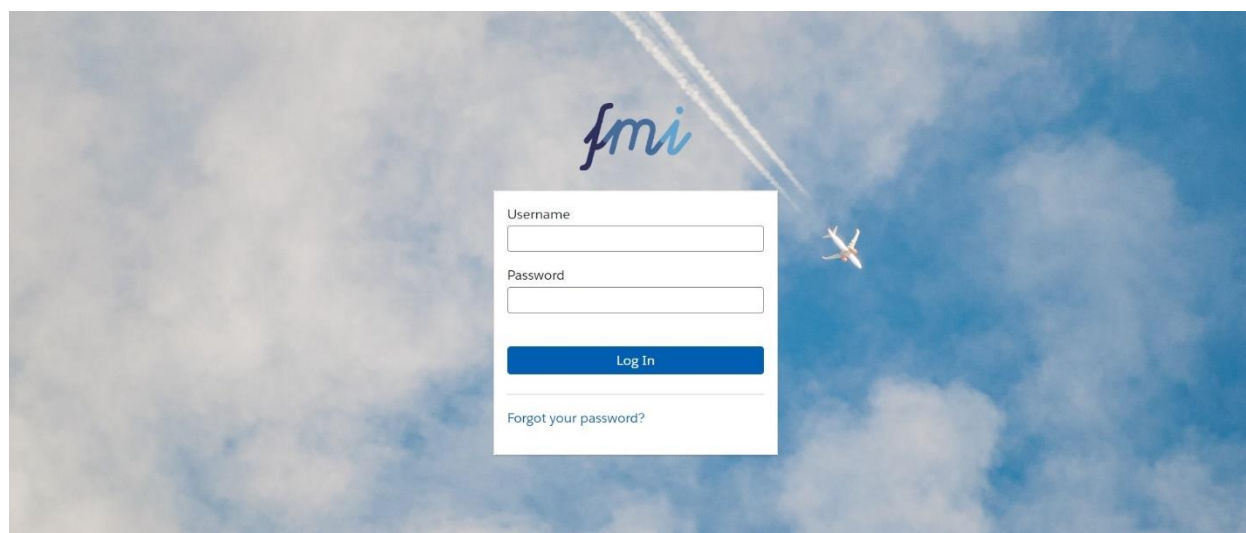
Мобилната версия на навигацията е изградена по един от най-популярните модели – „хамбургер“ меню. Частта с елементите на менюто се заменя от емотикона, репрезентираща „сгънато“ меню, която при натискане визуализира елементите в секция, заемаща цялата страница, с основния цвят за фон като елементите са подредени в колона и центрирани по хоризонтала.

### 3.5.5 Страници за Вход и Регистрация

Функционалностите за регистрация и влизане в портала са изцяло поети от *Salesforce*. В портала са предоставени страници и за двете действия, като могат да бъдат персонализирани по всякакъв начин. По подразбиране на тях е поставена форма за съответната дейност и компонент, където да бъде поставено логото на компанията.

Използвайки готовите функционалности, се избягва нуждата от създаване на сигурна интеграция за изпращането на данните на потребителите до сървърите на *Salesforce*. Тази работа се поема от тях и тя включва: събиране на данните, изпращането им по сигурна (*HTTPS*) връзка, хеширане на паролите преди да бъдат запазени в базата данни на *Salesforce*, създаване на потребител в платформата, към който се запазва хешираната парола и други данни, които се били въведени при регистрацията и изпращане на имейл за потвърждение. Освен това *Salesforce* спазва различни разпоредби за защита на данните като *GDPR*, гарантирайки, че данните се обработват в съответствие със законовите стандарти [9].

Имейлите, които се получават от потребителят при регистрация или забравена парола, могат да бъдат персонализирани, използвайки *Email Template* функционалността. За всеки вид служебен имейл може да бъде конфигуриран горен и долен колонтитул, съдържащ логото и цветовете на компанията, връзки към различни страници, информация за контакт и други.

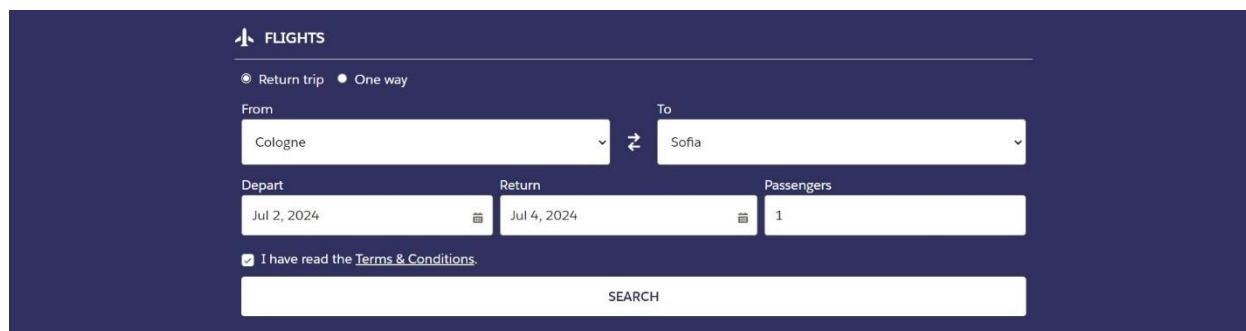


**Фигура 16. Страница за вход в портала**

### 3.5.6 Компонент за закупуване на самолетни билети

Първото нещо, което вижда потребителят, когато отвори портала е компонент, в който да въведе информация за полет, който търси. Първият избор, който трябва да направи е дали ще търси билети за едностранен или двустранен полет чрез радио бутон. Спрямо тази селекция се променя избора на дата като, ако полета е двустранен, има две полета, в

които да се избере дата – едното за тръгване, другото за пристигане. Ако се избере еднопосочен полет, полето за дата става само едно – за тръгване. Освен това потребителят трябва да избере от кой град ще лети и каква е неговата дестинация. Това е реализирано чрез групиран падащ списък като градовете са групирани по държави, например Дортмунд и Кьолн ще се визуализират под етикета Германия. Списъкът е подреден по азбучен ред – първо за държавите и после за градовете. Извличането на данните става от обекта „Акаунт“ като записите са филтрирани по това техният вид да е „летище“. Стойността, която се гледа за град е в полето *Destination City*, а тази за държава е от *Billing Country*. Причината градът да не се взема от *Billing City* и наличието на допълнително поле за град е, че *Billing Address* съдържа точния адрес на самото летище, но понякога той не съвпада с точната дестинация, на което то отговаря, например летището в град Бергамо отговаря за дестинацията Милано, друг пример е главното летище за град Букурещ е всъщност разположено в град Отопени. След избора на градове следва поле за броя пътници и потвърждение за прочетени правила на компанията.



FLIGHTS

☒ Return trip ☐ One way

From: Cologne To: Sofia

Depart: Jul 2, 2024 Return: Jul 4, 2024 Passengers: 1

☒ I have read the [Terms & Conditions](#).

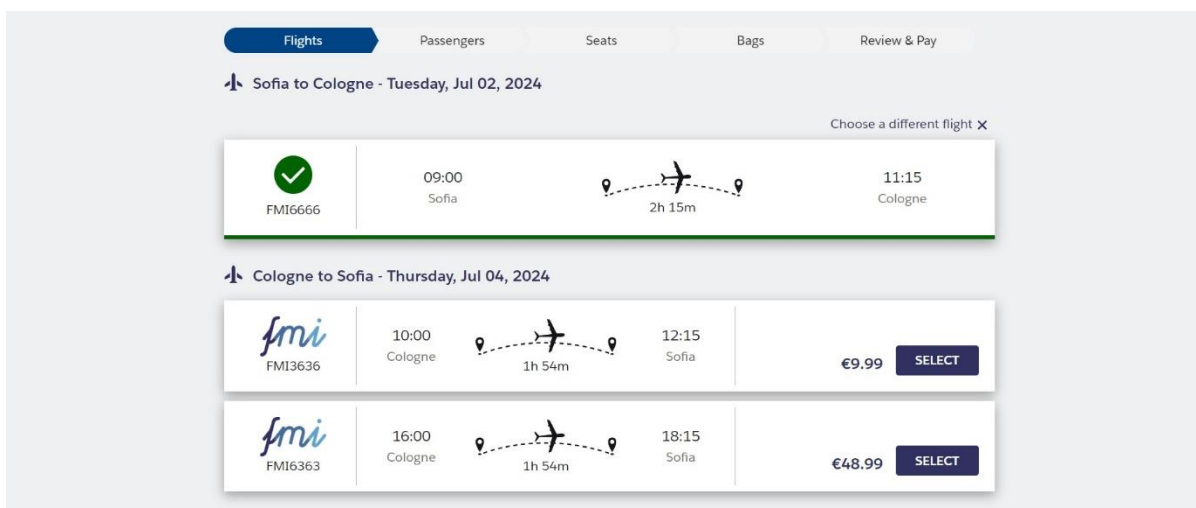
SEARCH

**Фигура 17. Търсене на полети**

При натискане на бутона „търсене“, потребителят бива отведен в нова страница, където се зарежда същият компонент с предварително избраната от потребителя информация и под него вече се визуализира секция с полети, отговарящи на избраните параметри. Запазването на информацията през смяната на страници е реализирана чрез изпращане на параметри заедно с пренасочването на страницата, които компонента при зареждане достъпва и попълва в съответните променливи. Компонента разположен под този за търсенето на полети е един родителски такъв, който в себе си съдържа няколко компонента, отговарящи за всяка стъпка от пътя, който потребителят трябва да измине за да закупи своите билети. Стъпките са: избор на полети, попълване на данни за пътници, избор на места, избор на багаж, визуализация на избраните продукти и плащане. През целият процес се извършва комуникация между компонентите от вида дете-родител и родител-дете. Причината за това разделение на отделни компоненти е спазването на добри практики, гласящи, че всяка логическа единица трябва да е отделно обособена, стига това да е възможно. Комуникацията родител-дете се осъществява чрез откриване на променливи,

намиращи се в дъщерния компонент, като се постави декораторът @api пред тях. По този начин родителски компонент има директен достъп до тях и може да променя стойността им като предава актуални данни. Обратната комуникация се случва чрез персонализирани събития, които биват изпращани от дъщерния компонент. В родителския компонент се задава какво да се случи, когато едно такова събитие бъде засечено.

Компонентът, който визуализира наличните полети е разделен на две секции – за отиване и за връщане. Ако потребителят е избрал еднопосочен полет, втората секция не се показва. Всеки полет се визуализира на собствена „плочка“, която съдържа информация за номера на полета, час на излитане и кацане, продължителност на полета и цената, зададена в полето *Base Ticket Price*. При натискане на бутона „Избери“, останалите полети за съответната дата, ако има такива, биват скрити и избраният променя изгледа си, за да се отличи, че вече е избран. След като е избран един полет до него се появява опцията да се отмени избора, която при избиране ще отмени селекцията и отново ще визуализира всички налични полети за съответната дата. При избор на полет за избраната дата/дати се появява опцията за продължаване към въвеждане на информация за пътниците.



Фигура 18. Избор на полети

Спрямо броя на пътниците, въведен от потребителят при търсенето, толкова форми за въвеждане на информация се появяват. Тази стойност се предава от родителския компонент към този, който отговаря за данните на пътниците. Формата включва падащо меню за избор на обръщение и двете имена на човека. Още при зареждането на компонента се създава списък с брой обекти колкото е броят на пътниците и при продължаване към следващата стъпка всеки обект се попълва със съответната информация за всеки пасажер. След което списъкът се изпраща към родителския компонент чрез персонализирано събитие. От своя страна, родителския компонент обновява този списък в себе си, скрива текущия компонент и показва следващия, който е за избор на места в самолета.

The screenshot shows a flight booking interface. At the top, there are four tabs: 'Passengers' (active, highlighted in blue), 'Seats', 'Bags', and 'Review & Pay'. Below the tabs, there are two identical forms for 'Passenger 1' and 'Passenger 2'. Each form has three input fields: 'Title' (a dropdown menu with '--None--' selected), 'First Name' (a text input), and 'Last Name' (a text input). At the bottom right of the form area, there is a blue button labeled 'CONTINUE'.

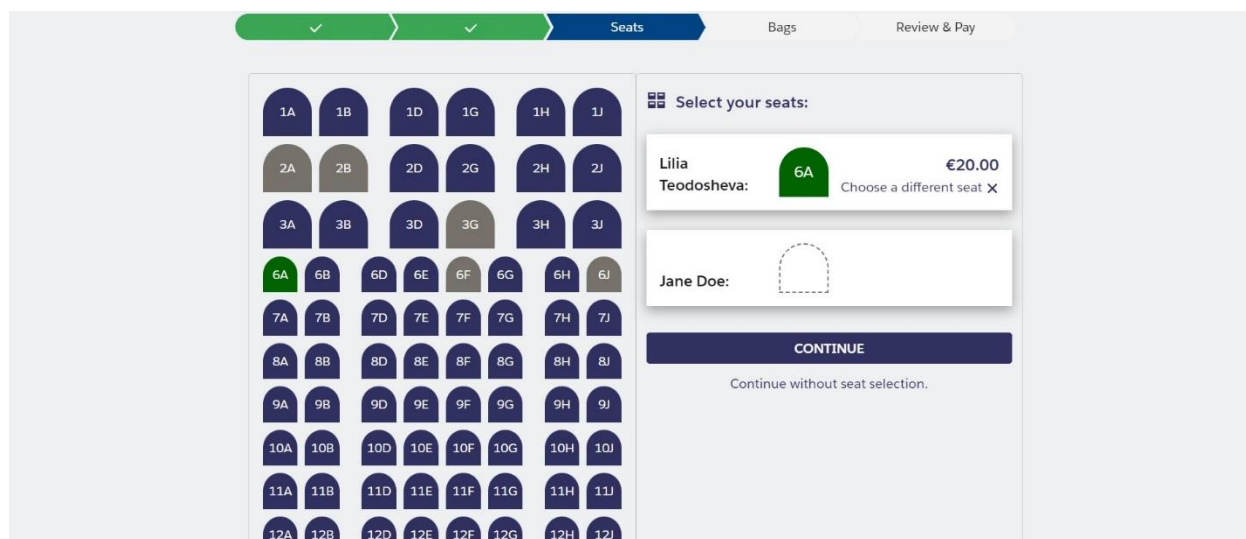
**Фигура 19. Попълване на информация за пътници**

Компонентът за избор на места е изграден основно от две секции. Първата съдържа схемата на местата в самолета. Тази схема се зарежда динамично спрямо полета, който е избран. Всяко място представлява запис в обекта *Aircraft Seat*. Взимат се всички записи като се подреждат в колекция от тип „ключ-стойност“ като ключът представлява номера на реда, а стойността списък от тип *SeatWrapper*, създаден за да запази информацията, която е необходима за всяко място. В този списък са подредени местата, намиращи се на съответния ред. За да се изгради схемата се използва информацията за позицията на всяка седалка и точно, когато тя е до пътеката. Спрямо това дали пътеката се намира отляво или отдясно се добавя *CSS* клас, който да сложи външно отстояние от елемента от съответната страна. Подобна проверка се извършва и за това дали седалката е заета или не. Ако заета, към променливата, държаща класовете, които ще бъдат приложени на съответния елемент, се добавя и такъв, който ще отрази на схемата, че това място не може да бъде заето. Друга информация, която се записва за мястото е бележките за него, които се показват в „балонче“ при заставане върху него на схемата. Записва се и цената за съответното място. За визуализацията на местата се използва повтарящ е *HTML*, който итерира по ключовете на колекцията и за всеки ключ се итерира по записите от списъка, който е добавен като негова стойност. Всеки ред представлява контейнер, на който е зададен *CSS* свойството **flex**, със стойности: **1 1 0**; чрез който се задава елементите, намиращи се в контейнера да растат и намаляват ширината си респективно спрямо мястото, в което са разположени. Добавянето на външните отстояние за всяко място, което се намира до пътека, създава илюзията за наличието на такава в схемата. По този начин се постига ефектът схемата на местата да се доближава максимално до това, което е в реалността.

Другата секция от този компонент представлява изобразяване на информация за пътниците и кое място е избрано за тях. Преди да е избрано такова се вижда само рамка, сякаш мястото е празно. При избор тази рамка се запълва като в нея пише кое е избраното

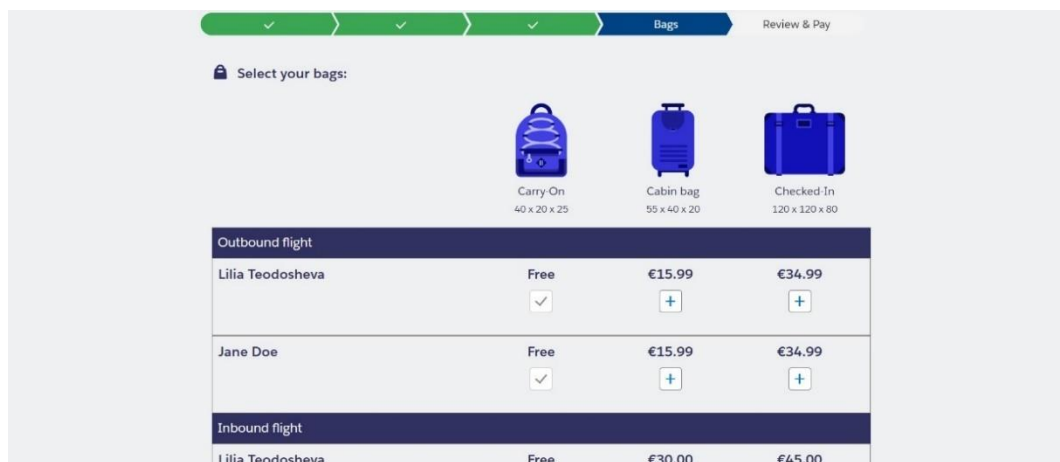


място. Освен това до нея се появява и сумата, която трябва да се заплати за нея, и възможност да се премахне селекцията. В допълнение, щом бъде избрано място, на схемата то се оцветява в зелен цвят. На схемата също така има и места в сив цвят, което индикира, че те вече са заети и не могат да бъдат избирани. Секцията, показваща пътниците, винаги остава видима, дори и при плъзгане надолу, за да се види цялата схема с местата. По този начин се осигурява потребителят винаги да е информиран за своя избор. При натискане на бутона „Продължи“, ако потребителят е избрал двупосочен полет, същият екран бива зареден със схемата, която отговаря на самолета за втория полет, и процедурата по избор на места се повтаря.



**Фигура 20. Избор на места**

Следващата част от потока е изборът на багаж. Отново компонента е разделен на две секции – една за полета на отиване и една за този на връщане. За всеки пътник може да се избере какъв вид багаж да се добави към неговия билет. Визуализират се трите вида багаж с техните размери и цени. За всеки пътник може да се избере един или повече вида багаж.



**Фигура 21. Избор на багаж**



След продължаване се появява и последната стъпка. На този екран потребителят вижда информация за избраните продукти и крайната сумата, която трябва да се заплати. При потвърждаване той бива препратен към плащането, което може да бъде реализирано по различни начини, които се разгледани в следващата точка. След успешно заплащане се създава запис в обекта *Reservation*, към който се свързва всеки продукт, закупен от потребителя, под формата на *Reservation Item*.

### 3.5.7 Картови разплащания

Когато става въпрос за една комерсиална компания, плащането с карта е с голяма тежест в цялостната имплементация. С развитието на технологиите плащането по начин, различен от този с карта, е просто недопустимо. Освен това, потребителите, които искат да си закупят билети, няма как да се намират в една държава, което също подтиква към решението заплащането да бъде дигитално. Както с повечето неща, които трябва да се реализират в една система, и тук стои избора дали да се използва готово решение или да се изгради от нулата. Един от вариантите за реализиране на плащане с карта е да се използва готовият продукт Chargent, който може да бъде свален от магазина за приложения на Salesforce. Другият вариант е да се напише интеграция, която директно се свързва с банката и след инициентирано плащане обработва съответния отговор от системата.

Chargent е продукт, изграден изцяло върху Salesforce, което прави неговото внедряване сравнително лесно. За свързването на предпочитан payment gateway, не е необходима никаква интеграция, тъй като създателите на продукта са изградили форма, която прави всичко това. Тя поддържа предварително създадена интеграция с повече от 30 разплащателни системи като например - PayU, PayPal, Stripe и други. Единственото, което трябва да се направи, е да се въведат уникалните входни данни, предоставени от системата, която е избрана за payment gateway. След като те са вкарани е нужно да се направи тестова транзакция и ако тя бъде успешна, връзката между двете системи е установена. Цялостният продукт включва в себе си освен настройката за свързване с gateway, и множество от персонализирани обекти, които след успешна транзакция запазват информация, необходима за бизнес, също така предоставят и готов компонент, който е може да бъде използван в Experience портали, какъвто е разработен за този проект. Един от недостатъците, както и при всеки продукт, който може да се използва наготово, е това, че тази услуга се заплаща на месечна база. Друго нещо е, че дори да поддържа връзка с над 30 различни payment gateway системи, тази, която е възложена от бизнеса може да не е част от тях.

Реализирането на плащане с карта от нулата не е толкова лесно. Трябва да се изгради цялостна интеграция с банката като преди това трябва да се определят правилата, по които двете системи ще си комуникират. Тъй като основно Salesforce ще се обръща към външната система, следва да се получат документи от вида на:

- ръководство за интегриране на платежен портал/комуникационен интерфейс;
- ръководство за платежен протокол XML;
- фактуриране;
- верификация на сметка;
- токенизация;
- ръководство за работа с Административен интерфейс/портал;
- списък с възможни отговори/кодове при инициране на платежна операция;
- списък с тестови карти.

Базирайки се на информацията, предоставена от банката, трябва да се изгради сложна архитектура за построяването на различните заявки и обработването на техните отговори. Освен това, трябва да бъде изграден и персонализиран графичен потребителски интерфейс, от който потребителят за иницира плащането. В следващия абзац ще бъде представено примерна комуникация между двете системи, за да се осъществи плащане.

Когато бъде продължено към плащане, системата трябва да изпрати формуляр с необходимите данни за транзакцията към сайта на банката и потребителят да бъде пренасочен към форма за въвеждане на данни за банкова карта, която се намира на сървърите на банката, тоест потребителят напуска Salesforce портала. След това доставчика на услугата извършва проверка за изпълнимост на плащането. Банката иницира server-to-server REST заявка към Salesforce като изпраща статус за плащането - успешно или не. SFDC (Salesforce dot com) прави проверка дали е изпратена такава заявка и дали не е изтекло времето за плащане, записва статуса и връща отговор на другата система. Банката чака положителен отговор за да финализира транзакцията. Ако SFDC откаже или има провал в комуникацията, транзакцията се отменя.

И двата варианта имат своите положителни и отрицателни страни, затова решението трябва да се вземе от компанията, за която се разработва проекта. Много зависи от това, как самия бизнес управлява парите си и с какви банкови структури работи.

### 3.5.8 Други компоненти

Останалите елементи, които има на портала са реализирани чрез използването на готовите компоненти, предоставени от *Experience Builder*. Например, визуализацията на важни съобщения и новини е постигната чрез използването на компонента *List*, който е повтарящ се компонент, тоест предварително се изгражда структурата на елемент, който трябва да се визуализира, и тази структура се прилага за всеки елемент от колекцията, която

се подава като ресурс с данни. В този случай това е *CMS* колекция от новини. Това позволява да се използва вече създадено и преведено съдържание.

За визуализацията на резервациите за всеки потребител се използва компонента *Grid*, който по същество е същият като гореспоменатия, с разликата, че тук елементите могат да се визуализират под формата на матрица, като ръчно може да се определи по колко елемента да има на всеки ред. Другата разлика е източникът на данни, който в този случай е *Record List* със записи от обекта *Reservation*, които са филтрирани спрямо притежателят на резервацията.

Внедряването на чат, чрез който потребителите да се свързват с агентите от отдел обслужване на клиенти, също се осъществява чрез готовия компонент *Embedded Messaging*. Този компонент се поставя в зоната за долен колонтитул, след което е необходимо да се посочи вече създадената и настроена инстанция на чат.

Използването на готови компоненти улеснява поддръжката на системата в бъдеще, тъй като един администратор ще има възможността да променя параметрите на всеки компонент и неговата визуализация. Също така, тъй като тези компоненти са разработени от *Salesforce*, при възникване на проблем може да се поиска директна помощ от техния екип. В допълнение, по този начин е сигурно, че компонентите са адаптирани към новите версии на системата.

## 4 Сравнителна характеристика

Това как една компания управлява бизнеса си обикновено не е нещо, което се споделя публично. Въпреки това, благодарение на факта, че *Salesforce* споделя истории за успешни клиенти, бяха намерени две авиокомпаниии, които използват системата под някаква форма. И двете са големи американски компании като едната от тях е нискотарифна авиокомпания. Това са **Southwest Airlines** и **Delta Airlines**. Тъй като всяка от тях използва различна част от *Salesforce*, са избрани 3 критерия, по които да се сравнят продуктите. В следващите точки е описано защо точно тези критерии са избрани.

	Fly Me Instantly	Southwest Airlines	Delta Airlines
Salesforce Portal	За всички потребители	За туристически агенти и агенции	-
Business Intelligence	Salesforce	Tableau	Salesforce
Chatbots	Да	Да	-

Таблица 1. Сравнителна таблица

### 4.1 Salesforce портал

**Southwest Airlines** са реализирали портал с помощта на *Experience Cloud*, но той е насочен към бизнес клиенти, желаещи да правят групови резервации за по над 25 човека. Докато при текущата разработка се наблюдава портал, насочен към всеки човек, който иска да закупи билети за себе си или за повече хора. **Delta Airlines** имат портал, отворен към клиенти, но не използва технологиите, предоставени от *Salesforce*.

Този критерий е важен, защото не е нужна интеграция между две платформи, за да се синхронизира информацията. Когато порталът е разположен директно върху *Salesforce*, всички данни се записват директно в базата на платформата.

### 4.2 Визуализация и анализ на данни (Business Intelligence)

Това е изключително важна част от всеки един бизнес. *Tableau* е отделно приложение, предоставящо възможността за най-различни динамични отчети с разнообразни визуализации. Позволява извличане на данни не само от *Salesforce*, но и от други източници. Въпреки това, това е допълнителен разход за един бизнес и изисква допълнителна интеграция, за да се внедрят създадените отчети и табла в *Salesforce*. Вградената функционалност на системата за създаване на такива визуализации би била

напълно достатъчна за една компания, която съхранява нужната за тях информация върху платформата. Въпреки че изборът на видове отчети и визуализации не е толкова голям, все пак е едно добро решение без допълнителни разходи.

### 4.3 Чат роботи

Една от най-оценяваните функционалности за един потребител е възможността за връзка в реално време с представител, който би могъл да помогне за едно по-гладко преживяване. *Service Cloud* предоставя възможността за създаване на чат и управлението на агенти, които да бъдат ангажирани с клиентите. Допълнително към чата може да се вгради *Einstein Bot*, който представлява предварително разписана поредица от избори и техните пътища. Той може да се оптимизира достатъчно добре, така че клиента да се задоволи само с отговорите получени от него, без дори да достига до разговор с реален човек. Въпреки това, след края на всички сценарии следва да се предостави опция клиента да се свърже с агент, ако желае. Цялата функционалност е напълно произволна и позволява много различни опции.

## Авторска справка за приносите

В настоящата дипломна работа е разработена облачна система за управление на авиокомпаниите – както вътрешно за служителите, така и външно за клиентите на компанията.

Освен технологии, присъщи за платформата **Salesforce** – *Apex*, *Lightning Web Components*, *Platform Event* и други; за реализирането на проекта са използвани и типичните за едно приложение такива – *HTML*, *CSS*, *JavaScript*. Полученият краен резултат е осъществен чрез:

- Модел на базата данни и неговата имплементация – изграждане на цялостен модел на базата, включващ персонализирани обекти и връзки, отговарящи на изискванията към проекта; интегриране на стандартните обекти с персонализираните такива, за да се подобри функционалността и интегритета на данните;
- Автоматизации и подобрения в работния процес – реализиране на няколко автоматизации с помощта на *Apex*, за да се рационализира процеса на работа и да се подобри ефикасността на екипа; създаване на *Batch Job* за периодично изтриване на записи, за да не се претоварва базата;
- Персонализирани компоненти – изграждане на персонализирани компоненти за предоставяне на интерактивни и удобни за потребителя интерфейси; добавяне на параметри, позволяващи компонентите да бъдат лесно адаптирани, спрямо променящите се нужди на бизнеса;
- Известия – създаване на автоматизирани нотификации по електронна поща при промяна, касаеща екипажа на полета; известия в платформата и от мобилното приложение при ново съобщение в чата на екипажа;
- Чат функционалност – реализиране на чат в реално време чрез *LWC*, позволяващ на екипажа да общува и обменя информация директно в запис на съответния полет, подобрявайки комуникацията и координацията помежду им;
- Генериране на PDF документи – интегриране на *Visualforce* страница, използвана за генерирането на файл, съдържащ график с полети, и изпращането му по имейл на потребителят, осигурявайки възможност за офлайн достъп до него;
- *Experience Cloud* портал – създаване на уеб портал за клиентите на авиокомпанията, от където могат да закупуват билети и управляват резервациите си; използване на *Salesforce CMS* за лесно управление и локализация на съдържание, позволявайки еднакво преживяване на потребителите, избрали различни езици;

- Локализация – използване на *Custom Labels*, преведени на няколко езика, за да може порталът да бъде достъпен за всички;
- Календар функционалност – изграден календар, позволяващ на екипажа достъп до графика им с полети, включващ както осъществени, така и бъдещи такива, с възможност за директна връзка към детайлна информация за полета с едно натискане; възможност за моментална заявка за получаване на принт копие на графика на електронната си поща;
- Сигурност на данните – използване на изградената от *Salesforce* функционалност за работа с пароли и сензитивни данни при процеса на регистрация.

Изброените точки включват главните аспекти, които са част от разработката на проекта. Всички те водят до създаването на мощна и лесна за ползване система за една авиокомпания. Чрез създаването на гъвкав модел на данните, добавяне на полезни автоматизации и създаването на различни компоненти са подобрили ефикасността и потребителското преживяване. Порталът и персонализираните компоненти в него осигуряват гладко и приятно преживяване за клиентите на компанията. Този проект демонстрира как чрез **Salesforce** могат ефективно да се управляват комплексни бизнес процеси.

## Перспективи за бъдещо развитие

Гледайки в бъдещето, се появяват няколко интересни функционалности, които могат да бъдат включени към текущата разработка, за да я направят по-ефикасна и комплексна с цел да бъдат обхванати и други отдели на компанията. Крайната цел е всички бизнес дейности да бъдат изнесени в платформата на **Salesforce**. Перспективите за бъдещето включват:

- **Мобилно приложение:** Всеки един сайт, изграден чрез *Experience Cloud*, може да бъде превърнат в мобилно приложение с помощта на *Mobile Publisher*. Този инструмент предоставя възможността за изграждане на мобилно приложение без да се пише никакъв код. Единственото, което трябва да се направи е да се предоставят ресурси като: изображение за зареждащ екран и лого на приложението в няколко размера. Освен това, *Salesforce* се грижи за процеса по публикуване в магазините за приложения на двете най-популярни операционни системи – *iOS* и *Android*. Наличието на мобилно приложение би улеснило допълнително клиентите на компанията като предоставя по-лесен достъп, нотификации в реално време и други;
- **Marketing Cloud:** Модул от платформата *Salesforce*, позволяващ създаването на персонализирани маркетинг кампании, проследяване на реакцията на потребителите и анализ на маркетинговото представяне. Това позволява по-добра комуникация с клиентите, използвайки различни маркетингови похвати, следователно и привличането на нови такива;
- **Certinia:** Внедряването на *ERP* система в текущия проект ще рационализира финансовите операции като ги интегрира с останалата част от платформата. В допълнение, наличието на такава система ще осигури единен изглед на финансовите данни, ще подобри възможностите за отчитане и ще автоматизира много счетоводни процеси. *Certinia* се внедрява в една *Salesforce* организация чрез инсталирането на *managed* пакет, съдържащ в себе си всички необходими обекти и връзки помежду им.

Всички изброени елементи са по-някакъв начин част от *Salesforce*, което осигурява придържане към целта на този проект – реализиране на всички бизнес процеси на една компания върху единствена платформа, което допринася опростена и улеснена комуникация между различните части, нужни за управлението на един голям бизнес.



## Използвана литература

- [1] The Technology that Changed Air Travel - <https://retool.com/blog/air-travel-software>  
(последно посетен: юни 2024 г.)
- [2] Цитат от Дъглъс Колман - <https://www.quora.com/How-were-airline-reservations-made-before-the-internet> (последно посетен: юни 2024 г.)
- [3] Калкулатор за инфлация - <https://www.in2013dollars.com/us/inflation/> (последно посетен: юни 2024 г.)
- [4] The Technology that Changed Air Travel - <https://retool.com/blog/air-travel-software>  
(последно посетен: юни 2024 г.)
- [5] Here's why Salesforce is the #1 CRM for growing businesses - <https://www.salesforce.com/crm/what-is-crm/is-salesforce-a-crm/> (последно посетен: юни 2024 г.)
- [6] A Complete Guide to Flexbox - <https://css-tricks.com/snippets/css/a-guide-to-flexbox/#aa-browser-support> (последно посетен: юни 2024 г.)
- [7] Using Flexbox in an Email - <https://medium.com/emails-hteumeuleu/using-flexbox-in-an-email-4b1aa7a69886> (последно посетен: юни 2024 г.)
- [8] Salesforce Documentation - [https://developer.salesforce.com/docs/atlas.en-us.pages.meta/pages/pages\\_compref\\_pageBlockTable.htm](https://developer.salesforce.com/docs/atlas.en-us.pages.meta/pages/pages_compref_pageBlockTable.htm) (последно посетен: юни 2024 г.)
- [9] Compliance - <https://compliance.salesforce.com/en> (последно посетен: юни 2024 г.)