

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ
Факультет кібербезпеки, комп'ютерної та програмної інженерії
Кафедра комп'ютерних систем та мереж

ДОПУСТИТИ ДО ЗАХИСТУ
Завідувач кафедри

_____ Жуков І.А.

« ____ » _____ 2020 р.

ДИПЛОМНА РОБОТА

(ПОЯСНЮВАЛЬНА ЗАПИСКА)

випускника освітнього ступеня "МАГІСТР"

спеціальності 123 «Комп'ютерна інженерія»

освітньо-професійної програми «Комп'ютерні системи та мережі»

на тему: "Обробка повідомлень про доставку екіпажів в інформаційній мережі
авіакомпанії"

Виконавець: _____ Сидорчук М.Ю.

Керівник: _____ Сураєв В.Ф.

Нормоконтролер: _____ Надточій В.І.

Засвідчую, що у дипломній роботі
немає запозичень з праць інших авторів
без відповідних посилань

Сидорчук М.Ю.

Київ 2020

MINISTRY OF EDUCATION AND SCIENCE OF UKRAINE
NATIONAL AVIATION UNIVERSITY
Faculty of Cybersecurity, Computer and Software Engineering
Computer Systems and Networks Department

“PERMISSION TO DEFEND
GRANTED”

The Head of the Department

_____ Zhukov I.A.

“ _____ ” _____ 2020

MASTER’S DEGREE THESIS

(EXPLANATORY NOTE)

Specialty: 123 Computer Engineering

Educational-Professional Program: Computer Systems and Networks

Topic: “Processing of crew delivery notifications in the airline's information network”

Completed by: _____ Sydorчук М.У.

Supervisor: _____ Сураев В.Ф.

Standards Inspector: _____ Надточій В.І.

Kyiv 2020

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет кібербезпеки, комп'ютерної та програмної інженерії

Кафедра комп'ютерних систем та мереж

Освітній ступінь: «Магістр»

Спеціальність: 123 «Комп'ютерна інженерія»

Освітньо-професійна програма: «Комп'ютерні системи та мережі»

“ЗАТВЕРДЖУЮ”

Завідувач кафедри

Жуков І.А.

“ _____ ” _____ 2020 р.

ЗАВДАННЯ

на виконання дипломної роботи

Сидорчука Максима Юрійовича

(прізвище, ім'я та по-батькові випускника в родовому відмінку)

1. Тема дипломної роботи: “ Обробка повідомлень про доставку екіпажів в інформаційній мережі авіакомпанії ” затверджена наказом ректора від 25.09.2020 р. № 1793/ст
2. Термін виконання роботи (проекту): з 1 жовтня 2020 р. до 25 грудня 2020 р.
3. Вихідні дані до роботи (проекту): *Інформаційна система авіакомпанії та емулятор польотів на мові програмування java. Обробка повідомлень про реєстрацію екіпажу під конкретний рейс. Час обробки даних – не більше ніж 3 секунди.*
4. Зміст пояснювальної записки: Вступ. Концепція екіпажу літака та їх доставка на борт. Аналіз вимог до мікросервісів. Структура веб-сервісів. Реалізація мікросервісів.
5. Перелік обов'язкового графічного (ілюстративного) матеріалу: Графічні матеріали результатів дослідження надати у вигляді презентації у форматах .ppt, .pdf.

NATIONAL AVIATION UNIVERSITY

Faculty of Cybersecurity, Computer and Software Engineering

Department: Computer Systems and Networks

Educational Degree: “Master”

Specialty: 123 “Computer Engineering”

Educational-Professional Program: “Computer Systems and Networks”

“APPROVED BY”

The Head of the Department

_____ Zhukov I.A.
“ _____ ” _____ 2020 p.

Graduate Student’s Degree Thesis Assignment

_____ Sydorchuk Maksym Yuriyovych _____

1. Thesis topic: “Processing of crew delivery notifications in the airline's information network” approved by the Rector’s order of 25.09.2020 p. № 1793/CT
2. Thesis to be completed between з 1 жовтня 2020 p. до 25 грудня 2020 p.
3. Initial data for the project (thesis): *Airline information system and flight emulator in java programming language. Processing of crew registration notifications for a specific flight. Database queries - no more than 3 seconds.*
4. The content of the explanatory note (the list of problems to be considered): *Introduction. The concept of the aircraft crew and their delivery on board. Analysis of requirements for microservices. The structure of web services. Implementation of microservices. Data processing time - no more than 3 seconds.*
5. The list of mandatory graphic materials: *Graphic materials are given in MS Power Point presentation.*

6. Календарний план-графік

№ пор.	Завдання	Термін Виконання	Підпис керівника
1	Узгодити технічне завдання з керівником дипломної роботи	1.10.20- 8.10.20	
2	Виконати пошук та вивчення науково-технічної літератури за темою роботи	9.10.20- 15.10.20	
3	Опрацювати теоретичний матеріал щодо принципів роботи екіпажів під час рейсу	16.10.20- 18.10.20	
4	Проаналізувати відомі підходи та методи розв'язку проблеми, розробити систему для обробки повідомлень в інформаційній мережі авіакомпанії	19.10.20- 03.11.20	
5	Розробити інформаційну систему авіарейсів та обробку повідомлень про доставку екіпажів	04.11.20- 15.12.20	
6	Оформити графічну частину записки та подати матеріали роботи на антиплагіатну перевірку матеріалів	13.12.20- 14.12.20	
7	Отримати рецензію та відгук керівника. Надати матеріали роботи на кафедру.	15.12.20 18.12.20	

7. Дата видачі завдання: “1” жовтня 2020 р.

Керівник дипломної роботи _____ Сураєв В.Ф.
(підпис керівника)

Завдання прийняв до виконання _____ Сидорчук М.Ю.
(підпис випускника)

6. Timetable

#	Completion Stages of Degree Project	Stage Completion Dates	Signature of the supervisor
1	Agree on the terms of reference with the thesis supervisor	1.10.20-8.10.20	
2	Perform a search and study of scientific and technical literature on the topic of work	9.10.20-15.10.20	
3	Develop theoretical material on the principles of crews during the flight	16.10.20-18.10.20	
4	Analyze known approaches and methods of problem solving, develop a system for processing messages in the information network of the airline	19.10.20-03.11.20	
5	Develop a flight information system and processing of crew delivery notifications	04.11.20-15.12.20	
6	Make a graphic part of the note and submit the materials of the work for anti-plagiarism	13.12.20-14.12.20	
7	Get a review and feedback from the manager. Provide work materials for the department.	15.12.20 18.12.20	

7. Assignment issue date: 01.10.2020

Diploma Thesis Supervisor _____ Suraev V.F.
(Signature)

Assignment accepted for completion _____ Sydorчук M.Y.
(Signature)

ABSTRACT

The Explanatory Note to the Master's Degree Thesis "Mail robot for agreements monitoring in the airline's information network": 70 pages, 24 figures, 24 references.

MODEL, NETWORK, SOFTWARE, MODELING, WEB-SERVICE, DESIGN, INTERFACE, DESIGN, SITE DEVELOPMENT, BROWSER, FLIGHT MONITORING, AIRCRAFT TRACKING.

The Goal of the Master's Degree Thesis – create the mail robot for agreements monitoring in the airline information network.

Main Tasks create the mail robot for agreements monitoring in the airline information network.

The Designing Object of Project – conception creating of Web system of monitoring in the airline information network.

The Subject of Project – the Web system of monitoring in the airline information network.

Practical Usage. Designed the Web system of accounting for for agreements monitoring in the airline information network.

CONTENT

LIST OF SYMBOLS, ABBREVEATIONS, TERMS	9
INTRODUCTION	10
PART 1 THE CONCEPT OF AN AIRCRAFT CREW AND THEIR DELIVERY ON BOARD	12
1.1. The concept and composition of the crew of an aircraft.....	12
1.2. The flight safety	22
1.3. Crew resource management.....	25
Conclusion on the first part	32
PART 2 ANALYSIS OF REQUIREMENTS FOR MICROSERVICES	33
2.1. Web service design requirements	33
2.2. Functional requirements	34
2.3. Non-functional requirements	36
Conclusion on the second part.....	41
PART 3 STRUCTURE OF WEB SERVICES	42
3.1. Structure of the crew registration microservice.....	42
3.2. Component diagram for MVC pattern	48
3.3 Advantages of java	48
3.4. Class structure diagram	61
Conclusion on the third part	61
PART 4 IMPLEMENTATION OF MICROSERVICES	63
4.1. Configuring spring boot application	63
4.2. Running of the flight system	65
Conclusion on the fourth part	67
CONCLUSIONS.....	68
REFERENCES	69

LIST OF SYMBOLS, ABBREVEATIONS, TERMS

MVC	Model View Controller
DI	Dependency Injection
ISO	International Standardization Organization
SF	Spring Framework
UML	Unified Modeling Language
IP	Internet Protocol
OSPF	Open Shortest Path First
SP	Spring Core
FTP	File Transfer Protocol
DB	Data Base
OOP	Object Oriented Programming
LLC	Logical Link Control
DHCP	Dynamic Host Configuration Protocol
OS	Operating System
VLAN	Virtual Local Area Network
MAC	Media Access Control
PG	Postgres
IOC	Inversion of Control
SS	Spring Security

INTRODUCTION

In the modern world and Ukraine, aviation ranks first in the transportation of passengers and cargo and is undoubtedly the most convenient and fastest mode of transport.

The main task of aviation is to ensure the safety of passengers. Flight safety is affected by several factors: the reliability of aircraft and ground equipment, the quality of flight training, climate, the quality of the control system. Analyzing plane crashes, scientists say that the main factor is the mistakes of pilots or controllers.

In order for an ATS worker to be able to make decisions quickly in stressful conditions caused by a lack of time or information, he must have deep practical skills.

Practice shows that after long breaks in work caused by various circumstances (illness, vacation), the operator loses their skills. That is why the urgent task is to develop a simulator for recovery and skills training for ATC controllers.

Aviation achieves the famous record safety of flights - almost four cases per million flights around the world. However, in the last ten years, the relative number of aviation occurs in the world, which remains virtually constant, with the category of occurrences associated with runway operations consistently representing the most numerous group. This can be used in the safety of the runway to reduce the relative number of aviation phenomena in the world, as well as the number of related deaths, despite the projected growth of the total air volume in the future. As a result, the International Aviation Communication called on ICAO to be a leader in reducing the number of events and incidents associated with the runway operation. Through its Runway Security Program, ICAO aims to coordinate ongoing global efforts to enhance the security of runway operations.

These three programs are just a few examples of how ICAO and its partners are working to identify and address threats, improve safety and enable all partners to collaborate and connect seamlessly.

The goal of the degree thesis – create the mail robot for agreements monitoring in the airline information network.

Main tasks : create the mail robot for agreements monitoring in the airline information network.

The designing object of oject – conception creating of Web system of monitoring in the airline information network.

The subject of project – the Web system of monitoring in the airline information network.

Practical usage - designed the Web system of accounting for for agreements monitoring in the airline information network.

PART 1

THE CONCEPT OF AN AIRCRAFT CREW AND THEIR DELIVERY ON BOARD

1.1. The concept and composition of the crew of an aircraft

The specificity of the legal status of aviation personnel who are part of the aircraft crew predetermines the existence of norms regulating the activities of the crew and constituting one of the most important institutions of air law, which requires separate consideration.

The crew of a civil aircraft are persons who, in accordance with the established procedure, are entrusted with performing certain duties in the management and maintenance of an aircraft in flight.

The aircraft crew consists of the commander, other flight personnel and service personnel.

The division of the crew into flight and maintenance personnel is of fundamental importance. The flight crew performs duties related to the control of the aircraft in flight. The flight crew includes persons who have special training and a certificate for the right to operate aircraft and their equipment: pilots, navigators, flight engineers, flight mechanics, radio operators, observer pilots, as well as flight operators performing special work. The operating personnel do not perform functions related to the control of the aircraft in flight. The crew attendants include flight attendants, flight operators of transport aircraft and other specialists, the list of which is established by the MGA. Thus, during test flights and flights for research purposes, engineers and other specialists may be included in the crew.

The crew may include inspectors and trainees, the number of which is limited. As a general rule, the crew is allowed to include no more than one inspector and one trainee from among the flight personnel. NPP GA determines the procedure for performing flights with an inspector in the crew [1].

The composition of the crew is determined by order of the Minister of Civil Aviation, depending on the type, class and purpose of the aircraft, as well as the goals and conditions of its operation. The flight of an aircraft with an incomplete crew is not allowed.

The procedure for the formation of aircraft crews is determined by the Civil Aviation Enterprise, the Manual for the Organization of Flight Operations in Civil Aviation.

The crew may include inspectors and trainees, the number of which is limited. As a general rule, the crew is allowed to include no more than one inspector and one trainee from among the flight personnel. NPP GA determines the procedure for performing flights with an inspector in the crew.

In addition to the key requirements for the crew, the national characteristics of the flight personnel play an important role. For many passengers, this issue is very significant and determines the choice of those who will form their environment during the flight. It should be noted that this factor has both objective and, mainly, subjective grounds. As the first, we can cite an example when passengers traveling with business partners prefer that the crew do not speak Russian in order to avoid leakage of important or personal information.

As for the subjective grounds, they are associated, first of all, with the personal stereotypes and prejudices of passengers and relate to the issue of their personal psychological comfort. It is safer and more comfortable for a passenger to fly with crews of certain nationalities. However, it should be noted that regardless of gender, place of birth, skin color, religion or language, all specialists have the same level of training, education and experience in flying an aircraft. It is these moments that are still dominant in assessing the quality of their work. For those for whom the national characteristics of the team play an important role, we will describe several features of the different crews.

1. The crew of an aircraft consists of the flight crew (commander, other flight personnel) and the cabin crew (flight operators and flight attendants). The flight of a civil aircraft is not allowed if the number of the flight crew is less than the minimum established number.

2. The composition of the crew of an aircraft of a certain type is established in accordance with the requirements for the flight operation of an aircraft of this type.

3. For the period of testing an experimental aircraft, the composition of its crew is determined by the designer of this aircraft.

4. The flight crew of a civil aircraft may include only citizens of the Russian Federation, unless otherwise provided by federal law.

Aircraft crew members for health reasons must meet the established requirements. Fitness for flight work for health reasons is determined by medical and flight expert commissions of civil aviation. The aircraft crew must be trained to fly this type of vessel.

An aircraft commander can only be a person who has the specialty of a pilot (pilot), as well as the preparedness and experience necessary for independent piloting and control of an aircraft of this type [2].

The commander of the aircraft manages all the activities of the crew, and in the event of an emergency landing - and the actions of all persons on board the aircraft, before transferring their powers to the competent authorities, ensures strict discipline and order on the ship, compliance with the rules of flight and operation of the ship. The orders of the aircraft commander must be complied with by all persons on board without question.

Depending on the specialty, the level of preparedness and work experience, flight personnel and other aviation personnel are assigned a class and a corresponding certificate is issued. Crew members on duty must be dressed in the prescribed uniform, carry a valid license and present it at the request of authorized officials.

The aircraft commander is subordinate to the commander of his unit and superior commanders (chiefs), and at airports outside the base also to the commander of the airline (airport chief) or a person replacing him.

The second pilot is subordinate to the aircraft commander and superior direct commanders (chiefs) and is obliged to:

- possess the technique of piloting and air navigation to such an extent as to ensure the safe performance of the flight in the event that the pilot-in-command for health reasons or other reasons cannot fulfill his duties in flight;

- observe pre-flight rest;

- be able to analyze and correctly assess the meteorological and aeronautical situation in preparation for flights and in flight;

- refuse to complete the flight task if he considers it too much for himself or is not sure of the safety of its execution;

- fully prepare for the flight;

- to control the condition and readiness of the aircraft, the correctness of its loading in accordance with the flight manual and the technology of the crew;

- check before departure the closure of emergency and cargo hatches, fuel fillers, close the fuselage doors on aircraft, where it is the co-pilot's duty to do so;

- know and follow the rules of discretion, phraseology of radio exchange and the rules of conducting radio communications;

- timely report in flight to the pilot-in-command of the aircraft on all deviations and malfunctions in the operation of aviation technology and aircraft equipment and make proposals for their elimination;

- take care of passengers, take, at the direction (permission) of the aircraft commander, measures to ensure their safety, safety of the aircraft and the cargo on board, special equipment and flight documentation;

- make a decision and act in accordance with the prevailing in-flight situation, if the pilot-in-command cannot fulfill his duties for health reasons or other reasons;

- perform a missed approach from a decision-making altitude in accordance with the Airplane Flight Manual, if by this moment the aircraft commander has not made or informed the crew about the landing or go-around;

- inspect the aircraft in accordance with the flight manual after landing and taxiing into the parking lot and report to the pilot-in-command of the aircraft [3].

The co-pilot has the right to:

- to operate the aircraft at all stages of the flight after passing the appropriate training and with the permission of the aircraft commander;

- to apply in flight the rights of the aircraft commander in the event that the commander, for health reasons or for other reasons, cannot perform his duties and the co-pilot has taken over these duties.

The co-pilot is responsible for:

- compliance with legal requirements and guidance materials;

- placement and fastening of the load in compliance with the established balance of the aircraft and the flight weight (in accordance with its duties determined by the flight manual and the technology of the crew);

- prudence during taxiing and in flight;

- the timeliness and correctness of their actions at altitude, making decisions on a par with the aircraft commander;

- maintaining the flight parameters set by the aircraft commander;

- safe flight outcome when piloting an aircraft with the permission of the aircraft commander and in the case when the commander for some reason cannot fulfill his duties in flight.

The navigator reports to the aircraft commander and superior direct commanders (chiefs). The navigator is obliged:

- know the instrumentation and navigation equipment of the aircraft, methods of application

The aircraft crew consists of the commander, other flight personnel and service personnel. The composition of the crew is determined depending on the type, class and purpose of the aircraft, as well as the goals and conditions of its operation. The flight of an aircraft with an incomplete crew is prohibited.

The crew of an aircraft, which is controlled in flight by one pilot and does not require other crew members on board, consists of the pilot-in-command. The minimum number of flight personnel is specified in the aircraft flight manual. The flight crew includes persons who have a valid flight certificate, as well as the training and

experience necessary to operate an aircraft of this type or its equipment: pilots, navigators, flight engineers, flight mechanics, radio operators, observer pilots, as well as flight operators performing special work. Crew attendants include flight attendants, transport aircraft operators and other professionals. The crew of civil aircraft of the Russian Federation may include only citizens of the Russian Federation. Exceptions to this rule can be established in the manner determined by the Government. The crew members, in accordance with their position, must:

- navigation aids and provide air navigation on established routes and flight patterns;
- know and follow the rules for storing and handling aeronautical information documents and flight topographic maps;
- select the necessary reference documentation, personally prepare flight cards;
- observe pre-flight rest;
- be able to analyze and correctly assess the meteorological and aeronautical situation in preparation for flight and in flight;
- refuse to complete the flight task if he considers it too much for himself or is not sure of the safety of its execution;
- to fully carry out navigational training for the flight;
- monitor the condition and readiness of the instrument and navigation equipment of the aircraft and operate it in accordance with the Airplane Flight Manual;
- know and follow the rules of discretion, phraseology of radio exchange and the rules of conducting radio communications;
- timely report in flight to the pilot-in-command of the aircraft about all deviations and malfunctions and give proposals for their elimination;
- to take care of passengers, to take, at the direction (permission) of the aircraft commander, measures to ensure their safety, safety of the vessel and the cargo on board, special equipment and flight documentation;
- inspect the aircraft in accordance with the flight manual after landing and taxiing into the parking lot and report to the pilot-in-command of the aircraftp[4].

- draw up documentation, write down comments on the operation of aviation equipment and the results of the inspection, hand over the aircraft in the prescribed manner.

The flight engineer (flight mechanic) has the right to require specialists from the aviation engineering service to eliminate the detected malfunctions.

The flight engineer (flight mechanic) is responsible for:

- reception of the aircraft in good condition and prepared for the flight;
- compliance with the rules of aircraft operation on the ground and in flight;
- closing of fuel fillers, emergency and cargo hatches, fuselage doors;
- the presence on board of the established ship documentation, rescue equipment, the amount of fuel, oil, liquids and gases required for the flight;
- timely information to the aircraft commander about malfunctions of aviation equipment.

The radio operator reports to the aircraft commander and superior direct commanders (chiefs).

The radio operator must:

- know and be able to operate electrical, radio and lighting equipment of an aircraft, ensure the operation of onboard radio facilities and two-way radio communication;
- observe pre-flight rest;
- refuse to complete the flight task if he considers it too much for himself or is not sure of the safety of its execution;
- fully prepare for the flight;
- control the condition and readiness of the electrical, radio and lighting equipment of the aircraft;
- know and observe the phraseology of radio exchange and the rules of radio communication, timely transmit to the pilot-in-command of the air traffic control units and meteorological information;
- timely report to the pilot-in-command of the aircraft about all deviations and malfunctions and give proposals for their elimination;

- to take care of passengers, to take, at the direction (permission) of the aircraft commander, measures to ensure their safety, safety of the vessel and the cargo on board, special equipment and flight documentation;

- inspect the aircraft after landing and taxiing into the parking lot and report to the pilot-in-command of the aircraft your comments.

The radio operator has the right to demand that the specialists of the aviation engineering service eliminate the detected faults in the electrical, radio and lighting equipment of the aircraft. The radio operator is responsible for:

- ensuring reliable operation of airborne radio facilities in flight and maintaining stable two-way communication;

- the accuracy and timeliness of the adopted dispatch instructions, meteorological and other information and messages transmitted from the aircraft.

The flight attendant of the aircraft is subordinate to the pilot-in-command during the flight mission. The flight attendant must:

- to know and control the condition of the aircraft rescue and household equipment and properly operate it;

- observe pre-flight rest;

- control the amount and placement of the load on the aircraft;

- control the sanitary condition of the aircraft, maintain cleanliness and order in the passenger cabins and at the workplace;

- promptly report to the aircraft commander on malfunctions of household equipment;

- ensure that passengers comply with the rules of conduct on board the aircraft, promptly inform the aircraft commander about all violations of these rules;

- take care of passengers, take measures to ensure their safety at the direction (permission) of the aircraft commander;

- at the end of the flight, report your comments to the pilot-in-command and receive an assessment of your work.

The flight attendant has the right:

- demand from workers of ground services to eliminate the discovered deficiencies;
- require passengers on board the aircraft to punctually follow the rules of conduct on board.

The flight attendant is responsible for:

- compliance with the requirements of legislation and other regulatory documents in the part concerning it;
- taking measures to ensure that passengers comply with the rules of conduct on board the aircraft, prevent smoking in flight;
- timely information of the aircraft commander about violation of the rules of conduct by passengers.

The flight operator is subordinate to the aircraft commander when performing a flight mission. The flight operator must:

- know the equipment of the cargo compartment (baggage compartments) of the aircraft and operate it in accordance with the requirements, know the rules for loading, stowing and securing cargo, the procedure for preparing the necessary documentation;
- observe pre-flight rest;
- control, in accordance with the shipping documents and the alignment schedule, the presence, placement and fastening of goods and the serviceability of their packaging;
- timely report to the pilot-in-command on equipment malfunctions;
- take measures to ensure the safety of the aircraft and the cargo on board;
- to report at the end of the flight to the pilot-in-command of the aircraft your comments and get an assessment of your work.

The flight operator has the right:

- require specialists from the aviation engineering service to eliminate malfunctions of the equipment of the cargo compartment (baggage compartments) of the aircraft;
- demand from the employees of the transportation organization the placement and fastening of goods in accordance with the alignment schedule, elimination of violations in the packaging of goods

The flight operator is responsible for:

- compliance with the requirements of the legislation in the part related to it;

- availability, placement and fastening of goods in accordance with shipping documents and alignment schedule.

Observer pilots, flight operators performing special work, as well as other specialists included in the crew, are subordinate to the aircraft commander and perform their duties in accordance with job descriptions [5].

The crew may include trainees, but not more than one person, from among the flight personnel. The trainee reports to the aircraft commander, superior direct commanders (chiefs) and the crew member directly involved in his internship. During preparation for the flight and in flight, the trainee performs the functional duties of a crew member, in whose position he is on probation and whose job he occupies.

A trainee, in agreement with the aircraft commander, enjoys the rights of a crew member in whose position he is training, and within the limits provided to him by the person directly involved in his training. Responsibility for the timeliness, completeness and correctness of the performance of functional duties and decisions made by the trainee rests with the crew member directly involved in his internship.

The hospitality and tourism industry plays an essential role in global growth around the world. One of the biggest key players is an airline industry. An airline industry contains both hospitality and tourism. One of the key players to this industry is the cabin crew. However, “with recent economic decline the airline industry is not in good health in terms of operation and customer service organization” said (Laszlo, 1999). It is therefore, important to understand that the crews must perform function as key player in an airline industry by providing various types of customer service, safety and security threats, sales and promotion.

As cabin crews are playing an essential role in the industry, they are to be selected carefully by the human resources. The desired corporate skills sought for the crews are; a motivation to deliver high standard customer service, team skills and commercial awareness. In addition to these skills, it is necessary that an individual is interviewed and upon successful application, and to satisfy all other needs for its industry.

An airline crews has been playing an essential role worldwide for more than decade by comforting dozens of passengers around the world. The meaning of providing

comforting guests is to also make a profit but as well as making customer feel satisfied just like staying at a hotel. Perhaps this might be the reason why an airline attracts many people around the world. As a result of the popularity of an airline, this is obvious that there are more than thousands of applicants every year.

Once a hired employee has gone through with all above required training, there is a need for she or he to complete on the job training also known as OJT.

This training provides new crew to taste and experience a real valuable experience by boarding on an actual plane. During this training, crews are no longer taught by the instructor but by their seniors whom already have experienced in customer services. Upon successful feedbacks and real job experiences, crews are able to incorporate fully to an aviation industry.

1.2. The flight safety

Throughout the history of aviation, accidents have and will continue to occur. With the introduction of larger and more complex aircraft, the number of humans required to operate these complex machines has increased as well as, some say, the probability of human error. There are studies upon studies of aircraft accidents and incidents resulting from breakdowns in crew coordination and, more specifically, crew communication. These topics are the driving force behind crew resource management.

One of the two key elements of CRM is situational awareness, or, "SA". Simply put, it is the understanding of the conditions surrounding your flight. Knowing what is happening, what has happened in the past and how that may affect your flight in the future. Situational awareness is probably best described as a conditioned state of mind while flying. It comes from experience and knowledge and can be blocked by being unfit to fly do to fatigue, for example. This concept is obviously a major consideration in flying all aircraft, but can be considered to be somewhat easier maintained in a crew aircraft than in a single pilot one. Skills of CRM, that provides flight are shown in fig. 1.1. [6].

	Communications <ul style="list-style-type: none"> • Cultural influence • Role (age, crew position, etc.) • Assertiveness • Participation • Listening • Feedback 	
Situational Awareness <ul style="list-style-type: none"> • Total awareness of surrounding environment • Reality versus perception of reality • Fixation • Monitoring • Incapacitation (partial/total, physical/psychological) 	Problem Solving/ Decision Making/ Judgment <ul style="list-style-type: none"> • Conflict resolution • Review (time-constrained) 	Leadership/ Followership <ul style="list-style-type: none"> • Team building • Managerial and supervisory skills • Authority • Assertiveness • Barriers • Cultural influence • Roles • Professionalism • Credibility • Team responsibility
Stress Management <ul style="list-style-type: none"> • Fitness to fly • Fatigue • Mental state 	Critique (three basic types) <ul style="list-style-type: none"> • Preflight analysis and planning • Ongoing review • Post-flight 	Interpersonal Skills <ul style="list-style-type: none"> • Listening • Conflict resolution • Mediating

Fig 1.1. Skills of CRM

Increase of a level of a flight safety in conditions of broad use of aircraft in a national economy, more and broader use of airplanes and heavy-lift helicopters and passenger capacity gets special importance and urgency.

The flight safety is a central problem in activity of a civil aviation. From a position of the theory of systems the problem of a flight safety acts, first of all, as a problem of system integrity preservation “crew - aircraft - environment”. Besides the problem of flight safety actuates psycho physiological problems of interaction of crew members, controllers and other aviation specialists, and also a problem of improvement of all normative documents spectrum.

The flight safety is one of the main properties of an air transport system which determines a capability to execute flights without threat for life and health of people. The air transport system is a big economic complex, which includes aircrafts, which execute passenger and freight traffic and air activities, all aeronautical engineering, experts, which provide operation of aircrafts (crew, air traffic control service, engineering service and maintenance service of aircraft), the airports, various technical equipment.

The high level of flight safety of modern aircrafts depends both on the aircraft, and on qualification and training of crew which controls the aircraft. Functional efficiency of

an aircraft is determined by its structurally - industrial perfection, stability, controllability, maneuverability, and also high maintainability of a airframe, the engine and the equipment. Functional efficiency of crew depends on its theoretical preparation, knowledge of an aeronautical engineering, rules of its operation in all a range of expected operation conditions, including special situations, and habits of application of this knowledge, and also on discipline and diligence of pilot- in-command of aircrafts and crew members. The management efficiency air traffic, the organizations of flight activity and all kinds of maintenance of aircraft in a maximum degree is determined by the efficiency of the activity organization in the industrial enterprises, consciousness of initiators and the personal responsibility of heads of all ranks for activity concerning safety control of flights.

Many people travel by airplane all around the world. For some people it is the only way they can get to where they are going. On a daily basis, averages of 28 to 30,000 seats are filled on airplanes. At each airport, there are hundreds of arrivals and departures worldwide. Even though airline officials say flying is safe, accidents kill many people because airlines neglect to prevent human error or repair faulty equipment.

Sometimes I think the only reason an airplane could crash is if something on the plane were to break. However, most of the time that is not the case. A survey conducted by Boeing found that flight crews were responsible for at least seventy-three percent of all fatal airplane accidents. (Gray 17). Forty-one percent of these accidents occurred during landing because of unstable approaches. Also an investigation by the National Aeronautics and Space Administration on the causes of airline accidents revealed that more than eighty percent of all airline accidents involved some degree of human error (Helmreich 62). This is very alarming when people are putting their lives in the hands of flight crews. Forty-four passengers died aboard a new British Midland 737 after its crew shut down the wrong engine after the other one malfunctioned (Greenwald 40). Do you really think that flying on an airplane, over which you have absolutely no control is very safe[7]?

Reasons for flight crew error can be explained by the conditions under which they are flying. Flight crew fatigue is a largely increasing problem on many of the jumbo jet flights today. Although there are laws that prohibit cockpit crews from sleeping in flight,

there have been many weary pilots that have been known to nod off on occasion during some of their seventeen hour, non-stop flights (Urquhart 15). Perhaps laws should regulate the number of hours a flight crew is in the air instead of prohibiting sleep in flight. Another condition, alcohol abuse, has been found to inhibit the abilities of some flight crews. A northwest crew flying from North Dakota to Minnesota was found to be intoxicated on the job (Air Safety 61). Some people refuse to drive at night because of the number of drunk drivers on the road. Would passengers want a drunken pilot to be responsible for their lives while 20,000 feet up in the air?

Another reason for flight crew error is pressure to meet flight time schedules. Some of these flights take place during hazardous weather conditions. When I was

younger, I saw an airplane crash at the St. Louis Airport after the pilot was ordered to take off even though the plane had ice on its wings. The airplane skidded off the runway because the pilot could not control the steering mechanisms on the icy runway. Other incidents have occurred solely because of bad weather and an urgency to stay on schedule. When flight 803 came in to land at Tripoli, the pilot decided to land the plane even though a dense fog covered the runway. One hour earlier a Soviet jet scheduled to land at the same airport detoured to another to avoid the fog. There were no mechanical malfunctions of the plane; however, it missed the runway by more than a mile, cartwheeled, and slammed into two farmhouses (New Qualms 20). Another accident, which killed most of the passengers on board, occurred when an Air Florida Boeing 737 crashed into the Potomac River near Washington, D.C. in 1982. The speed gage for take off was covered with ice which caused the takeoff acceleration to be miscalculated (Helmreich 62).

1.3. Crew resource management

In the preceding chapter, the outcomes of several approaches to studying error were considered. Analyses of accident and incident data coupled with data from NASA's structured-interview program provided evidence for the need to focus more on crew-level issues rather than on training of individual skills. Research in simulators like the classic Ruffell Smith (1979) study were responsible for the introduction of the concept of management of resources on the flight deck. NASA's sponsoring of the first workshop on

flight deck resource management in 1979 formalized this notion of Cockpit Resource Management. Helmreich and Foushee (1993) write, "...with recognition of the applicability of the approach to other members of the aviation community including cabin crews, flight dispatchers, and maintenance personnel..." (p. 3), the term Cockpit Resource Management is being replaced by Crew Resource Management. The following sections provide a synopsis of the evolution of the CRM concept.

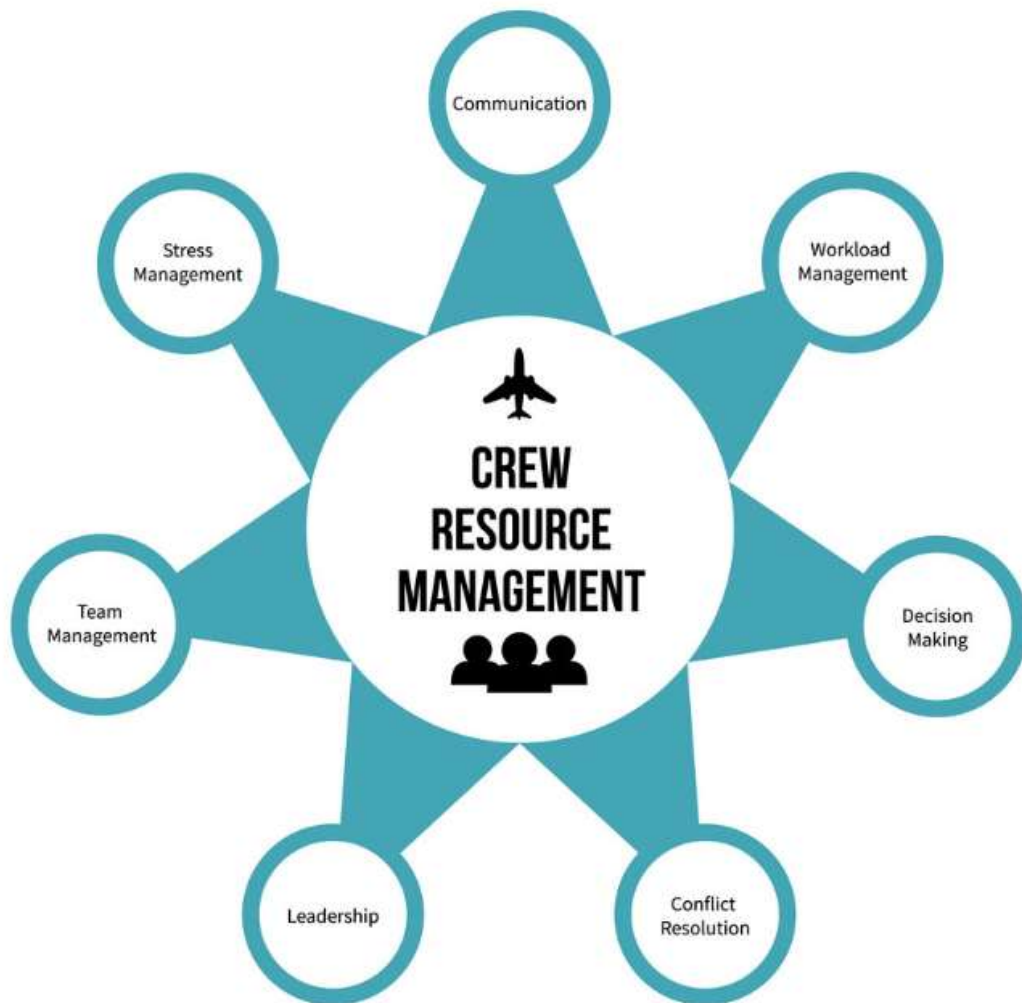


Fig. 1.2. Crew resource management

Most major airlines today incorporate some kind of CRM training program. Such "buy-in" from the operational community would not have been possible were it not for the persistent efforts of the NTSB and the cooperation of the FAA. As Kayten (1993) writes, "...the NTSB has played a major part in fostering the wide acceptance CRM concepts now

enjoy in the regulatory, airline, and military environments". Several accidents prior to 1979 also had what are today referred to as 'CRM problems'. The NTSB in their reports used to refer to these as problems centering around 'teamwork', 'inadequate flight management', 'task-sharing', and 'delegation of authority'. According to Kayten, earlier treatment of these issues was in the framework of operational procedures and airmanship. It was only after the NASA sponsored workshop in 1979 (see Cockpit Resource Management Chapter 2) that "...these problems were thought to have a unique training solution" (p. 289). Incidentally, the NTSB's first explicit mention of 'poor CRM' was in its report after the crash of a United Airlines flight, in 1978, in Portland, Oregon (Kayten, 1993). According to NTSB (1979), this accident "...exemplifies a recurring problem a breakdown in cockpit management and teamwork during a situation involving malfunctions 18 of aircraft systems in flight" (p. 26). Because of its significance, it is elaborated upon below. On December 12, 1978, the three person flight crew of United Airlines' Flight 173 prepared to land the four-engine DC-8-61 at Portland International Airport.

Upon lowering the landing gear on final approach, they heard a dull thump which seemed to be in the main gear area. Uncertain about what might have caused the thump or its implications, the crew decided to abort the landing and put the aircraft in a holding pattern. The aircraft stayed in this holding pattern for almost an hour during which the captain directed the crew to ascertain the cause of the noise. During this time, both the First-officer and Second-officer warned the Captain, on four separate occasions that they were running low on fuel. The Captain apparently ignored these warnings and insisted they stay in the holding pattern. Low on fuel, one of the engines filially flamed out. The Captain, bringing the plane near the field, demanded an explanation from the Second-officer for the cause of the engine failure. Meanwhile, the other three engines, with fuel tanks now dry, began to fail in sequence and the DC-8 nosed downward. The aircraft crashed into a wooded area near the airfield killing 8 passengers, the Second-officer, and a flight attendant. Among those seriously injured were 21 passengers, the Captain and the First-officer. The NTSB determined the probable cause of the accident to be the Captain's failure "...to monitor properly the aircraft's fuel state and to properly respond to the low

fuel state and the crew members' advisories regarding fuel state". The NTSB later made recommendations to the FAA to ensure that flightcrews were indoctrinated in principles of flightdeck resource management. By now, CRM was a familiar concept in the aviation community and the NTSB frequently referred to it in making subsequent recommendations to the FAA following similar accidents.

NASA and the NTSB have played important roles in the recognition of the concept of CRM, but it is the FAA in its dual role of enforcer and enabler (Birnbach & Longridge, 1993) which has been largely responsible for ensuring the indoctrination of the concept in airline operations. Following the 1978 Portland crash and the recommendations by the NTSB, the FAA played an active role in ensuring the implementation of CRM principles by air carriers. One of its first actions, as Kayten (1993) reports, was to issue "...Air Carrier Operations Bulletin 8430.17 Change 11, which included instructions regarding resource management and interpersonal communications training for air carrier flightcrew." (pp. 19 294-295). Subsequent responses to NTSB recommendations included the initiation of specific projects like the one to optimize Line-Oriented Flight Training (LOFT) to enhance CRM within the FAA's Aviation Behavioral Technology Program (Kayten, 1993), and the publication of its Advanced Simulator Plan as Appendix H to Part 121 in 1981 "...to encourage the use of advanced flight simulators and to specify their permissible use for training and checking..." (Birnbach & Longridge, 1993, p. 267). In 1987, the FAA formed the Joint Government-Industry Task Force to review the FAA's airline crew standards. Perhaps the most significant of FAA responses to NTSB recommendations to urge the fostering of CRM concepts among airlines was by the Joint Task Force in the form of the Advisory Circular (AC) 120-51 on CRM training dated December 1989, and the issuance of the final rule for an Advanced Qualification Program (AQP) Special Federal Aviation Regulation (SFAR-58) dated October 1990. Of these, the AQP promises to be the most prominent catalyst for the acceptance of CRM by many major airlines.

The AQP is a voluntary, alternative training program aimed at integrating a number of training features and factors aimed at providing airman performance when compared to traditional programs. As defined in SFAR-58 (FAA, 1990), the AQP

"...provides for a voluntary, alternative method for meeting the training, evaluation, certification, and qualification requirements...the principle factor of the AQP is true proficiency-based qualification and training". More detail on the role of AQP in CRM will be provided in a later section.

Since its conception, there has been a lot of research in CRM being conducted by NASA's Aerospace Human Factors Division in conjunction with airlines. In 1990, the FAA outlined the National Plan for Aviation Human Factors. This provided the blueprint for addressing the needs of the aviation community. But perhaps the best guideline for the development of CRM was provided by the FAA's (1991) Advisory Circular (AC) on CRM mentioned earlier. Many of the findings in this circular are based upon research conducted by researchers at NASA in collaboration with Robert Helmreich and his colleagues at the University of Texas, Austin. This advisory circular made the initial stab at providing a "...contemporary explanation of aircrew behaviors..." as well as provided "...guidelines for developing training and evaluation programs...".

In attempting to provide an explanation of aircrew behaviors, the AC identified some "new" cognitive and interpersonal skills and attitudes in addition to the traditional individual technical skills and knowledge. This new "family" of skills was further classified under three "clusters" (a) communications process and decision behavior, (b) team building and maintenance, and (c) workload management and situational awareness. These three clusters are further subdivided into behavioral categories. These categories illustrate the kinds of behavior and associated skills that the concept of CRM encompasses. For instance, the 'communications process and decision behavior' cluster is subdivided into categories that include briefings/debriefings, inquiry/assertion, and conflict resolution, while the 'team building and maintenance' cluster includes categories of behavior such as leadership, concern for operation, and interpersonal climate. Once these behavioral categories included under CRM had been identified, the issue was one of being able to observe and identify behavior as classified under the

clusters. For this purpose, the AC further identifies several 'behavioral markers' associated with the behavioral categories of each cluster. As Helmreich, Wilhelm, Kello, Taggart, and Butler (1991) write, "Behavioral markers are specific behaviors that serve as

indicators of how effectively resource management is being practiced" (p. 7). Further, they point out that these markers are not intended to be "...exhaustive lists of behaviors that should be seen, but rather as exemplars of behaviors associated with more and less effective CRM" (p. 7). While CRM emphasizes the importance of these Cognitive and Interpersonal (C&I) skills, it does not cease to recognize the importance of so-called, traditional, 'technical skills' However, as the AC emphasizes, "it is now accepted that these two behavioral sets are in fact mutually supportive, provide an effect greater than the sum of their parts and must be taught and evaluated together." By describing crew performance in terms of integrated sets of technical, and C&I behavior, the AC essentially outlined a model of CRM. Research into CRM thus focused on several interrelations between these issues. In most cases, such research had to be conducted in conjunction with developing training programs.

In a study originally aimed at assessing the effects of fatigue on crew performance, Foushee, Lauber, Baetge, and Acomb (1986) found that fatigued crews, surprisingly, made fewer errors than did crews composed of rested pilots who had not yet flown together. In another study, Foushee and Manos (1981) analyzed the cockpit voice data obtained from the Ruffell Smith (1979) study. The approach involved classifying speech acts as to type and the primary conclusion was that crews who communicated more overall, performed better. Kanki, Lozito, and Foushee (1989) further refined the methodology and analyzed earlier gathered data.

The AC also sets the stage for conducting training in CRM and provides broad guidelines for airlines. The AC stresses certain concepts seen as basic or essential to any form of CRM training. For example, it emphasizes the focus of the program to be the "...functioning of crews as intact teams, not simply as a collection of technically competent individuals..." (p. 10). It also stresses the need for training to be continuously reinforced (i.e., recurrent training should be a feature of any CRM training program). The AC further outlines three phases of CRM training. The first

phase is the indoctrination or awareness phase which typically constitutes an introduction to CRM concepts in a classroom or seminar, involving role-playing, and group discussions. The second phase is the practice or feedback phase. This phase typically

includes Line-Oriented Flight Training or LOFT which makes use of simulators accompanied with video feedback during debriefing. The 23 full-mission scenarios in LOFT are oriented towards emphasizing critical CRM concepts such as decision making, team participation and leadership sharing. The third phase recommended in the AC is the operational reinforcement phase. A short, one-time CRM training course, cannot be effective in influencing behavior that has been developed over a crewmember's life span. The AC recognizes the fact that attitudes revert to their pretraining status in the absence of continual reinforcement, and therefore stresses the importance of this phase. It stresses the need for CRM training to include refresher curriculum, integrating feedback exercises such as LOFT with video feedback. The AC also stresses the important role of check airmen or CRM instructors, since these are the people who serve as role models for all crews in any airline.

Fundamentally related to the issue of performance is that of measurement or evaluation. In the earlier section on CRM research and training, it was noted that survey data obtained by Helmreich and his colleagues clearly indicate the acceptance of CRM concepts. If this is true, why then does it become necessary to assess performance in CRM related skills? Indeed, as the AC on CRM notes, there are concerns expressed by industry as to the necessity of evaluating CRM, particularly since CRM training has been indicative of being effective in changing behavior when it is not evaluated. While there are some philosophical and psychological issues in this debate, from the regulatory standpoint, there is no question that performance in CRM needs to be evaluated. Thus, from a regulatory perspective such as the FAA's, ultimately the issue of crew performance and training boils down to one of qualification and certification. Typically reflective of an industry whose regulatory structure as well as physical characteristics of operations focus on the individual, pilot training and evaluation, to this day, is done on an individual basis. With the advent of CRM and particularly under the auspices of AQP, this is being re-examined and is slowly changing. As Birnbach

and Longridge (1993) note, "the changing role of the pilot warrants a complete reexamination of the standards which should be applied to a determination of competency...AQP provides a systematic methodology for such a reexamination"[8].

Conclusion on the first part

In the first part was analyzed the concept of an aircraft crew and their delivery on board. It also shows the relevance of flight monitoring tools, which solve the main security problem. The principles of crew delivery on board and existing solutions are considered. These tools are analogs of those used within one airline.

The work of the crew during the flight and their duties during the flight were also studied.

PART 2

ANALYSIS OF REQUIREMENTS FOR MICROSERVICES

2.1. Web service design requirements

Service Oriented Architecture

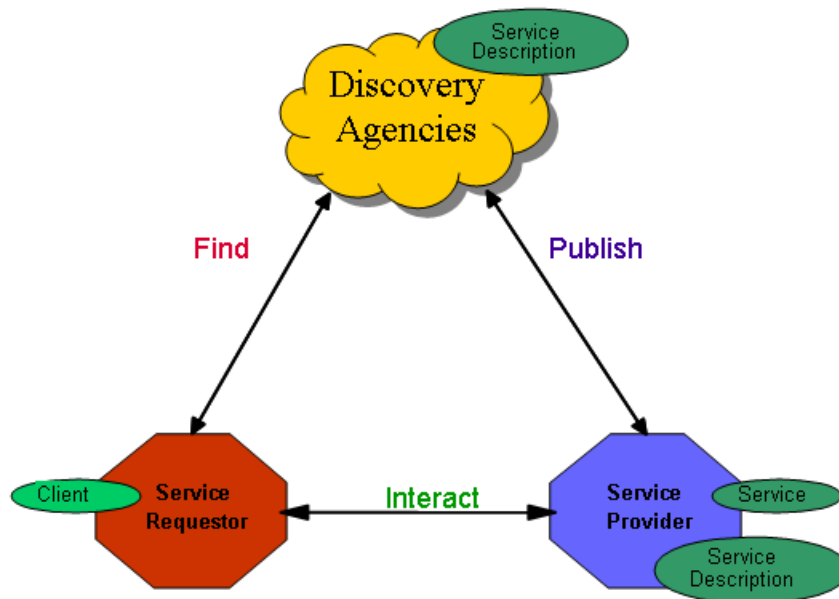


Fig. 2.1. Architecture of web-services

However, the essential part of Web services is the Interact relationship between a Service provider and Service requestor. This is the Web Service. Discovery agencies need not be used - they will in some cases but not in others. The discovery agencies are well represented as a cloud, rather than being a well-defined module in the web services architecture. They will become interface to a huge world of data and query services which provide data about web services as well as many other things [12].

Interfaces between software modules have well-defined functions, which in turn have well-defined and typed input and output parameters.

2.2. Functional requirements

Many requirement specifications and functional specifications differentiate between functional requirements and non-functional requirements. Functional requirements make a statement about a property to be fulfilled or the performance of a product, system or process. The non-functional requirements describe how well the system should perform and are often understood as boundary conditions and quality properties.

This distinction between functional and non-functional requirements dates back to the days of ISO 9126 (which has since been replaced by the ISO 25010 standard).



Fig. 2.2. Functional requirements diagram

Often not all quality requirements can be met equally in a software product. Rather, they compete with each other. In this case, the development team has to weigh them against each other and prioritize them. Here are two classic examples of this:

Security versus performance: To increase the security of the application, we use a more expensive encryption algorithm that increases the response time of the services.

Usability versus security: Entering a 30-digit password increases security, but affects the user experience. In software engineering and systems engineering, a Functional Requirement can range from the high-level abstract statement of the sender's necessity to detailed mathematical functional requirement specifications. Functional software requirements help you to capture the intended [13].

The software team has powerful tools and concepts available for all of the aspects mentioned. Here are some examples:

Maintainability / portability: SonarQube, assessment procedures (ATAM, audits)

Security: OWASP CVE Scanner, Ness Security Scanner, SonarQube

Reliability: stress tests, regression tests

Usability: A / B tests, WAI / AAA tests, usability inspection (guideline-based review), surveys

Performance: load tests, performance tests.

Quality takes time and resources. Therefore quality is not an end in itself, but is based on the economic weighing of benefits and costs. Such quality costs can be divided as follows.

Error prevention costs: quality-increasing measures, controls, planning, audits, etc.

Failure costs: loss of customers, reputation, contractual penalties, deviations

Ultimately, this consideration means that a certain amount of errors can be accepted due to economic constraints. (It is well known that this rule of thumb is sometimes overstimulated in the software area, see Banana Software. The error costs are often correspondingly high.)

Exceptions are highly critical systems such as those in the medical field or in air traffic control, in which errors cannot be tolerated for obvious reasons.

The developed flight monitoring emulator must meet the following functional requirements:

- 1) the user must log in to the system;
- 2) login details must be e-mail and password;
- 3) when the field is empty, the user must see what needs to be filled;
- 4) in case of incorrect e-mail or incorrect password, the user should see the corresponding message;
- 5) while waiting for authorization, the user must see the download animation;
- 6) the user after authorization must get to the main page;
- 7) on the main page the user must see the name;
- 8) after successful login, the user must see the logout button on top of each selected page;
- 9) the user should see his e-mail at the top of the page;
- 10) the user can view flights that arrived no more than 10 hours ago;
- 11) the user can view flights that will depart in the next 10 hours;

- 12) the user must see in the tables the flight number, the name of the aircraft, the airport of departure and arrival, the time of departure and arrival, the date of departure;
- 13) the user should not be able to change the information in the flight tables;
- 14) the user must be able to find the flight by his number;
- 15) the user must see a map of all flights through airport;
- 16) buttons for selecting the table of departures and arrivals, flight search, and real-time flight maps should be located at the top of the page in the selected menu, and be present on each page;
- 17) the user must be able to log out;
- 18) after logging out the user must see the login page;
- 19) there should be an admin page for crew registration;
- 20) there must be a form on the crew registration page;
- 21) crew registration form must have a field "Pilot1 name";
- 22) crew registration form must have a field "Pilot2 name";
- 23) crew registration form must have a field "Stewardess name";
- 24) crew registration form must have a field "Engineer name";
- 25) crew registration form must have a field "Flight";
- 26) crew registration form must have a field "Pilot1 name";

2.3. Non-functional requirements

The aim of every software development is to meet the requirements of the client or the market for the software product. A distinction is made between functional and non-functional requirements. Functional requirements (functionalFRs) specify which functionality or behavior the Own or should fulfill software product under specified conditions. Non-functional requirements (non-functional briefly NFRs), also called technical requirements Aspects that typically affect or overlap several or all functional requirements (cross-cut). You have in the Usually an influence on the entire software architecture. Furthermore non-functional requirements influence each other. Security often conflicts with usability, example Storage efficiency is usually the opposite of runtime

efficiency. For some non-functional requirements, a Quality of Service (QoS) - are defined, z. B. related to performance and efficiency. Non-functional requirements can affect different Get block types. The efficiency of the software system results from the complete example xen interaction of different components. The reliability and partly also the changeability are determined by the individual Components determined. Put simply, define functional requirements, what the software product is supposed to do while the non-functional Requirements specify how it should work. The demarcation between functional and non-functional demarcation Requirements is often not easy. For example, time requirements can be viewed as behavior of the software product or can also be viewed as a non-functional requirement [14]. When writing requirements, make sure they are complete and accurate and avoid vagueness. At the same time, avoid including extraneous information that may confuse people. Use “must” instead of “should” as you write the requirements document. Be consistent in the use of terminology and units, and be consistent in the format and language used.

In terms of the competitiveness of an IT system, the performance-related requirements are THE requirements to which the most attention should be paid. Because these form i.a. the basis for the functional requirements and influence the performance of an IT system.

If standards are missing in the organization, it can be difficult to define the right quantity for an IT system. Benchmarks can help here. An IT system in operation that is similar to the target system can be used as a benchmark. Nevertheless, the basis for determining the quantity is largely derived from the stakeholders' expectations of the IT system.

If the performance requirements are known, these include to be clearly documented in relation to the system success / failure. “Understandable” means that every performance requirement is enriched with values that create an unambiguously interpretable requirement from an unclearly formulated requirement. This is achieved among other things with the quantification of values.

The question arises as to how detailed a performance requirement should be described. This level of detail depends, among other things, on the size and criticality of

the risk to be controlled. In practice, this aspect in particular is often a tightrope walk for the requirements engineer. However, the requirements must reflect the reality to be formed with today's knowledge and be able to be adapted to changes. The INVEST approach known from agile methods can support this: the effort per requirement is estimated (E), the requirement fits into the sprint (S) and can be tested (T)[15].

Documenting performance requirements with Planguage enables solid communication between the stakeholder groups and increases the added value for the customer. This is also thanks to the standardized procedure, which prevents misunderstandings and increases transparency.

If a system doesn't do its job, we have a serious problem: What would a music player be that doesn't play music? But there are also many requirements that have nothing to do with the functioning of the system. And they are becoming more and more important.

First about the terminology: The functional requirements are, well, about the function of the system. What should the system do? The music player should play music (among other things). There is a long tradition and many approaches for working with functional requirements.

The word “non-functional requirement” is often criticized, among other things because the name suggests that these are not so important. That is why the term “quality property” is often used.

Non-functional requirements come in many shades. Some are directly relevant to the user, such as appearance or performance. Others are indirectly relevant, such as security or operating costs.

Patience, practice and discipline are required to ensure that Planguage is used correctly and efficiently in everyday life. The method offers a large number of parameters and icons for documentation. The parameters and icons can vary depending on the type of requirement. Since Planguage is not self-explanatory, the stakeholders who come into contact with Planguage documented requirements should be familiarized with the terminology in advance.

I have based my blog post on Tom Gilb's book (Competitive Engineering, 1st edition, 2005) and supplemented it with my practical experience. The blog post gives a rough

overview of Planguage. For a more comprehensive insight, I recommend reading the aforementioned reading and / or consulting the documentation of the concept.

The guiding principle in design, "Form Follows Function", has been around for a long time - function comes first, and form (a non-functional property) must not stand in the way and, in the best case, supports it.

This way of thinking is still, without a doubt, important and correct today. However, we are currently experiencing a decoupling of form and function, partly driven by software. This is particularly easy to see with products that have "stabilized": four wheels have proven their worth in cars, and smartphones with the flat, rectangular shape with the display on one side. Even if there are variants, it is difficult to differentiate here. This often comes from non-functional properties.

This is one of the reasons why functions take a back seat, as these can often be upgraded using software. During development, it is sometimes sufficient if a function can potentially be supported. For example, the music player mentioned at the beginning could be retrofitted with the ability to play other file formats. This defuses a missing function for the end customer.

At the same time, the non-functional properties are often those with which the user comes into contact. Who doesn't remember ticket machines with horrific user guidance? The function is available, but the non-functional user guidance makes it frustrating to use.

In software development in particular, it is quite easy to specify and take into account non-functional requirements right from the start. Working with user stories, for example, often has a positive impact on usability. Automated load tests ensure early on that the system is performing, etc[16].

Even in the modeling, non-functional requirements can be taken into account. This can be done using languages geared towards this, but also through the quantification of aspects (e.g. performance), or through the systematic recording of cross-sectional issues.

Features are easy to spot and can be copied by competitors. Non-functional requirements are an important way to set yourself apart from the competition. This was not the case in the past, which is why non-functional requirements are more important than ever.

System should be presented as follows:

- 1) a small number of images, so that the user feels comfortable while working;
- 2) the web page should be minimized to get the smallest size in order for it to load as quickly as possible;
- 3) spelling and punctuation for all page content;
- 4) a clear separation of the header, the main content and the bottom of the page;
- 5) menu and content should be in the middle of the page;
- 6) unique logo;
- 7) completeness of design;
- 8) adaptability of design;
- 9) transparent to the user navigation and target orientation in the program;
- 10) clarity and clarity of the user's understanding of texts, icons and buttons;
- 11) the speed of learning when working with the tool, for which it is necessary to use mainly standard elements of interaction, their traditional or conventional location;

Software requirements for the interface. These requirements must be implemented at the level of the software components that make up the entire flight monitoring tool. We are talking about support for platforms, databases and other software structures. All of them are described in more detail below;

- 1) the database management system must be relational PostgreSQL;
- 2) query language - SQL;
- 3) cross-platform - support for all browsers;

Productivity. The tool must support up to 50 users at a time. It follows that database queries must be made from a connection pool.

Accessibility. Responses to user requests should be as fast as possible, 2-3 seconds.

Reliability. The database must have 2 backup servers in case the main one fails.

Transfer. The system should be easily portable, in case of future creation of a mobile application.

Conclusion on the second part

This section clearly sets out the functional and non-functional requirements for web-services.

The list of functional requirements was grouped into a single list, which includes items on authorization, home page, and content.

Non-functional requirements are divided into separate blocks, thanks to which the future monitoring tool will be safe, accessible, portable, reliable and monitored.

PART 3

STRUCTURE OF WEB SERVICES

3.1. Structure of the crew registration microservice

Microservices are designed with the help of the Model-View-Controller manual pattern.

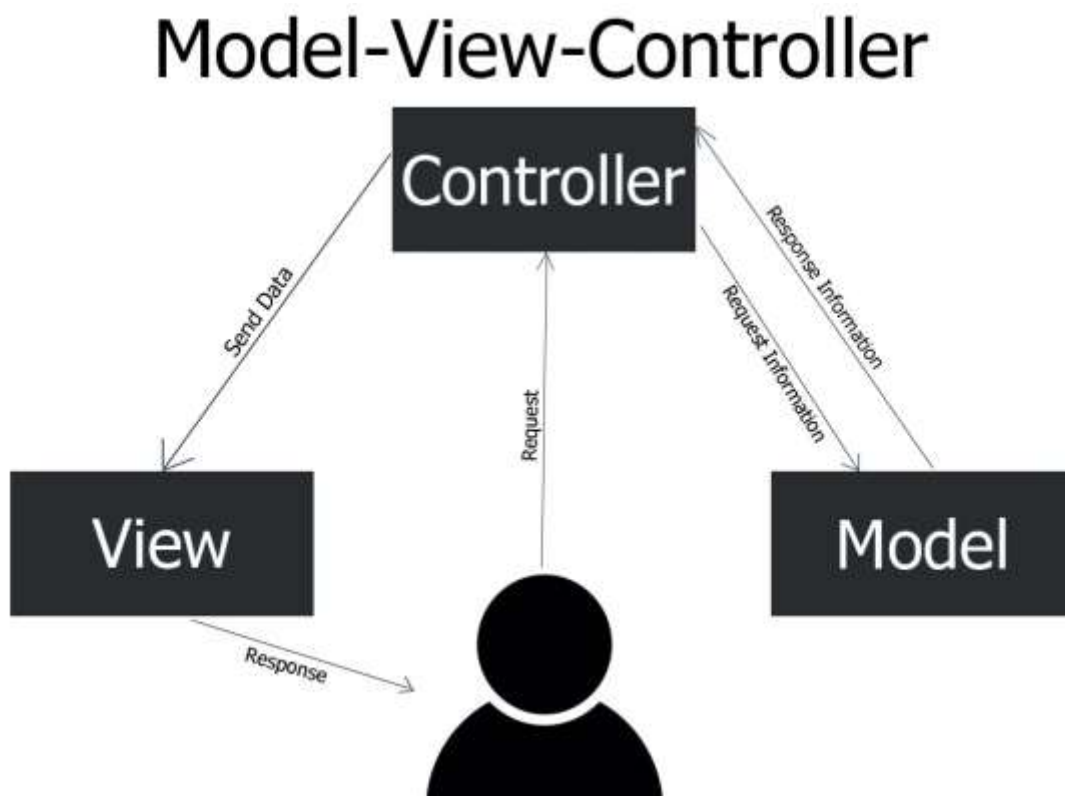


Fig. 3.1. MVC

The template will distribute the robot to three different functional roles: model of data (model), attributes for the operator, interface (view), and keruyuch logic (controller). In such a rank, the change, which is introduced into one of the components, will produce the least possible inflow on the other components.

Today most of the applications follow this pattern. It helps in the reusing of code and parallel development. This makes working easier and simpler. The components created through the MVC design pattern are independent of each other in nature [16].

MVC design pattern is also known as Model-View-Controller. It is a common architectural pattern which is used to design and create interfaces and the structure of an application.

This pattern divides the application into three parts that are dependent and connected to each other. These designs are used to distinguish the presentation of data from the way the data is accepted from the user to the data that is being shown. These design patterns have become common in the use of web applications and for developing GUIs.

Understanding these Design patterns is easy and simple. The theory stands for Model-View-Controller Pattern. The functions of the three parts are:

1. Model

This part of the design pattern is the primary part and contains purely application information. It doesn't contain any information on how to show the data to the user. It is independent of the user interface. It controls the logic and rules of application.

2. View

This part helps the user to see the model's data. The main concern of this part is to access the model's data. The view section uses a chart, table or diagrams to represent the information. It can also show similar data and use bar graphs and tables for different purposes. It is a visualization of information that the application contains.

3. Controller

Most of the work is done by the controller. It provides the support for input and converts the input to commands for the application. It is used between the model and view part. The model and the view are interconnected, so the execution is reflected in the view part.

Today most of the applications follow this pattern. It helps in the reusing of code and parallel development. This makes working easier and simpler. The components created through the MVC design pattern are independent of each other in nature. This feature helps the developers to reuse the components and codes easily and quickly in other multiple applications.

MVC is often used in web applications. The view in these applications is the HTML or XHTML files created by the application.

The controller receives the input in the form of getting input and it then manages and handles the input to the model. The model contains the data and the rules on the process of carrying a specific task [17].

The duplication of the model code which is not of the higher level is removed across the different User Interface implementations. The MVC pattern provides the core of the solution to any issues and helps in adapting these solutions to each machine.

Some of the major pros of using MVC Design Pattern are:

- multiple views can be made to models;
- the partition of duties helps the developer in future developments and upgradations;
- the MVC theory works have low coupling behaviour among the models, views, and controllers;
- multiple developers can work on models, views, and controllers at the same time;
- the views for a required model are grouped together.

It is an architectural pattern used in web applications. A prior understanding of programming and web applications will be an advantage to the user. The practice of coding and scripting and basic knowledge languages such as Python, Java or C# will give a boost to the learners and developers. MVC is not a complete application and it usually requires service layer, data access layer or logic layer.

The most important use of it is to segregate the views from the model and controllers. It helps in separating the display and the data and allow modification in each data without affecting the others. It is mostly used for developing Graphical User Interface.

MVC has been widely used for web applications in major programs. Some frameworks such as JavaScript MVC, Ember JS, and Backbone support the process of MVC partly on the client.

The scope of this is bright and demanding. Almost all the top companies and industries based on websites use MVC design patterns for developing User Interfaces and models.

It has so much in store for its developers. The users learn many skills and methods by using the MVC theory. There are a lot of skills associated with these design patterns and learning this technology will help the learner to boost not only their skills but also their future prospectus. These skills will help the learners in the long run and provide influential growth in their careers.

Understanding it is an important technique. This technology allows creating reusable and separate models that can be easily upgraded. Time taken to develop applications become less and the developers create an efficient application. The MVC theory is a basic concept of computer programming and helps in giving several web development services and projects.

Lastly, it is important for the developer to grasp the techniques and methods of the MVC model and learn how to apply them to their own projects.

Implementation of the pattern in our service:

Flight controller receives requests from the client and forwards them further to the service:

```
@RestController
public class FlightController {

    @Autowired
    private FlightsService service;

    public FlightController(FlightsService service) {
        this.service = service;
    }
}
```

Airplane service stores all the business logic of working with airplane objects:

```
@Service
public class AirplaneService {
```

```

    @Autowired
    private AirplaneRepository airplaneRepository;

    public AirplaneService(AirplaneRepository airplaneRepository) {
        this.airplaneRepository = airplaneRepository;
    }
}

```

Flights service stores all the business logic of working with flights objects:

```

@Service
public class FlightsService {

    @Autowired
    private FlightRepository flightRepository;

    public FlightsService(FlightRepository flightRepository) {
        this.flightRepository = flightRepository;
    }

    public List<Flight> getAll() {
        return flightRepository.getAll();
    }
}

```

Pilot service stores all the business logic of working with pilot objects:

```

@Service
public class PilotService {

    private PilotRepository pilotRepository;
}

```

The Java Persistence API (JPA) is the Java standard for mapping Java objects to a relational database.

JPA is one possible approach to ORM. Via JPA, the developer can map, store, update, and retrieve data from relational databases to Java objects and vice versa.

JPA can be used in Java-EE and Java-SE applications. JPA is a specification and several implementations are available [18].

The Object Relational Mapping is the base of JPA, which is all about representing and accessing data in the form of plain Java Objects – called Entities.

Hibernate, EclipseLink, and Apache OpenJPA are some of the JPA implementations, out of which, Hibernate is more popular and widely used. To reduce the burden of writing codes for relational object management of an enterprise application, a programmer follows the JPA Provider framework (an implementation of JPA), which allows easy interaction with the database instance. In a relational database, the relationships between the two tables are defined by foreign keys. Typically, one table has a column that contains the primary key of another table's row.[19]

In JPA, we deal with entity objects that are Java representations of database tables. So we need a different way of establishing a relationship between two entities. JPA entity relationships define how these entities refer to each other[20].

Repositories are implemented for direct connection to the database using the JPA library.

Airplane repository is inherited from the repository and thus has all the methods for getting and saving the airplane object:

```
@Repository
public interface AirplaneRepository extends CrudRepository<Integer, Airplane> {
    List<Airplane> getAll();
}
```

Flight repository is inherited from the repository and thus has all the methods for getting and saving the flight object:

```
@Repository
public interface FlightRepository extends CrudRepository<Integer, Flight> {
    List<Flight> getAll();
}
```

Pilot repository is inherited from the repository and thus has all the methods for getting and saving the pilot object.

3.2. Component diagram for MVC pattern

The Component Diagram shows the structural components of the information system and the relationship between them. Components are considered as information objects: files, modules, libraries, packages, etc [20].

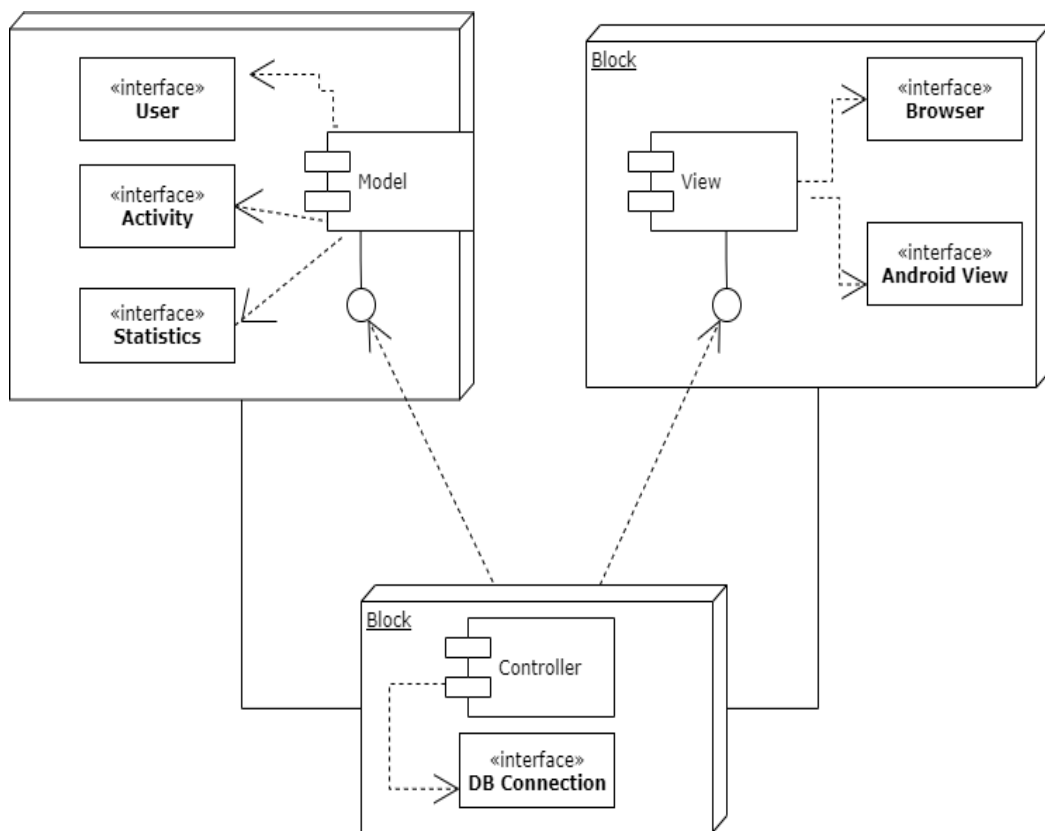


Fig. 3.2. Diagram of the components of the tool according to the pattern Model-View-Controller

3.3 Advantages of java

The JavaFX framework offers the possibility to develop Java applications with graphical user interfaces. It has been developed from scratch since 2008 and is intended to replace the older program libraries AWT (Abstract Window Toolkit) and Swing. JavaFX

can also be used to create two-dimensional drawings and animations (2D) as well as the 2D representation of three-dimensional drawings and animations (3D). It can directly access the corresponding interfaces of modern graphics cards.

In this section follows an example of a simple 2D animation using JavaFX. First, an object is created in the upper left corner of the screen: a yellow square with a black border. A "film" then runs, during which the object is animated one after the other in different ways, each lasting two seconds:

The object is moved to the right, then to the left and right again, the object is moved downwards, the fill color of the object is changed to green the color of the edge of the object is changed to red, the object is moved to the top left and simultaneously rotated by 45 degrees and the object becomes completely transparent, so it disappears.

The entire process then runs back to the start state. If you already have knowledge of another programming language, you will find many familiar elements in the code and the short comments, but also many elements that are specific to Java. Of course, this demo example is not yet suitable for a thorough learning of Java. At this point I refer to my book at Rheinwerk-Verlag.

The portable object-oriented programming language of the Internet age was published on March 23, 1995 by sun microsystems.

As early as the 1980s, the developer Gosling, who came to Sun from IBM, experimented with extensions for C ++, which should make it easier to transfer programs from one type of computer to another. Since C ++ appeared to be not robust enough for the intended application areas, the programming language Oak was created in 1991, which was used for devices such as remote controls and set-top boxes for televisions, but it was never really successful or popular. In the fall of 1994, a web browser, WebRunner, was written in Oak, and was first published in December 1994 in a small group. In 1995, Oak was renamed to Java and WebRunner to HotJava. HotJava was introduced to a wider public in May 1995 at SunWorld '95. The HotJava display itself does not know any file formats or protocols; all files are represented by programs that can be copied from external computers if necessary. These "slots" are implemented in Java and can therefore be run on

many types of computers. This means that the display can be expanded flexibly and can "learn" new file types and protocols if necessary.

At that time it was feared that Microsoft would soon react with a competing product Blackbird, which felt a certain time pressure. This explains why some parts of the language and library did not appear to be fully developed at first, but this is no different with most other products.

Java then became popular in 1995 and especially in 1996 because Java allowed programs called applets to be embedded in web pages. Back then it was said: "Sun's Java is the hottest thing on the Web since Netscape." Java was also used for programs outside of a web browser.

Java programs are relatively portable, so they run on a variety of different devices (Windows, Macintosh, Linux, sometimes Android).

Because Java programs do not run directly on a specific computer, but only on the simulated virtual machine, they can run on any computer for which there is a Java virtual machine. This means that Java programs can usually run on more different types of computers (with different processors and operating systems) without further revision than programs written in other languages specifically for a particular operating system and / or processor type.

Applications that are written in Java today have the chance of being usable in a variety of environments tomorrow.

Most new computers can also run applications in Java - the emphasis in this statement is on the fact that this can be done independently of the operating system used - this is even possible without an additional operating system: Then the Java Runtime Environment (jre) take on the role of an operating system. Because the same Java virtual machine (jre) is emulated on every computer architecture and under every operating system, Java classes (byte codes) are portable.

Since Java programs run in a simulated computer, they can be largely sealed off from the real computer, whereby the real computer can be protected from malicious Java programs. This protection is not perfect, however, that programs can exploit security holes to circumvent it. This issue has sometimes been reported in the press in connection with

Java security. However, it would be wrong to believe that Java programs are now more insecure than executable programs such as exe files, because if the additional security of Java fails, then this only means that Java programs are reduced to the low security level of exe. Files fall behind. It is also not always a security gain in favor of JavaScript and Flash programs on websites to do without Java programs, since security gaps have also been reported repeatedly with regard to JavaScript and Flash.

Java contains an integrated security concept to allow third-party programs to run on a machine with restricted rights (e.g. with regard to file access or communication).

System programming includes the programming of programs for the basic software of an operating system, for infrastructure such as device drivers, communication stacks, virtual machines, operating systems, development environments or foundation libraries (libraries with universally usable components); these are often programs that communicate directly with devices or who have to get by with few resources or who interact directly with such programs. Such programs rarely have a user interface.

Application programming includes the development of application programs, that is, programs for a specific task, for example for accounting or a specific computer game. An application program often has a user interface.

System programming is generally more difficult than application programming because it requires more knowledge, more abstraction, and more training. Java is more intended and suitable for application programming.

Java has a relatively large standard library (collection of pre-defined program parts), so that many tasks can be done with the standard output of the language.

When programming Java, the programmer makes use of various class libraries, such as JavaFX for creating a graphical user interface or jdbc for connecting to a database with sql. The Java beans are components that can be used in different environments by adhering to a certain interface standard.

There are a number of additional libraries and auxiliary programs for Java that make programming easier and some of them are free of charge. At runtime, a Java program can reload classes from the network.

The contemporary paradigm of object-oriented programming with classes and interfaces is supported by Java, so that programmers have access to many published methods of object-oriented programming.

The memory of Java programs is managed automatically, which makes programming easier and safer. Java supports the clear structuring of larger programs using classes and packages.

The lecture deals with the basics of the programming language (program structure, data, expressions, instructions) and describes the concept of object-oriented programming (basics, classes and objects, methods, constructors, inheritance, use of APIs). Furthermore, the processing and presentation of events / results using a graphical user interface (objects, triggering and handling of events) is taught. For data storage within Java programs, options for connecting to databases are presented (JDBC). The contents are available as online materials and are clarified and deepened within the internship using exercises (programming tasks).

Now we roughly know what programming is all about and should finally turn to Java. Compared to C and C ++, Java is a relatively new high-level language and

is also very popular. By the way, Java has absolutely nothing to do with JavaScript to do. The two languages have completely different areas of application and are very different.

Java has a special feature in contrast to classic high-level languages: A program written in Java can be run on dozens of platforms in its finished state,

That means a Java program can be used unchanged on different operating systems and

Processor architectures are executed. A program in C ++ can do this, for example not because the compiler translates it directly into the machine code of the target platform

translated. Java achieves this platform independence by introducing an intermediate level. The compiler in Java does not generate machine code from the source text (file extension .java), but generates so-called bytecode (file extension .class). Bytecode can be thought of as a kind of machine code that a specific processor is independent. However,

bytecode cannot be executed directly by a processor. Instead, the bytecode is read in by the Java Runtime Environment (JRE) and executed and this runtime environment is in turn for a specific platform has been created but the functionality is identical on all platforms. That means, that everyone who wants to run Java programs needs such a runtime environment. We will clarify where the runtime environment and the Java compiler are obtained from at the end of the section.

Another characteristic of Java is that the inventors of Java have many working. Have adopted concepts from C ++ and other languages. Java is an object-oriented programming language. However, we will only learn what that means in one of the clarify later chapter. However, the developers of Java deliberately did not adopt all elements from C ++. For example, Java intentionally does not support so-called pointer arithmetic, that is the ability to manipulate the memory address of data directly. Instead, the Java Runtime Environment takes care of memory management completely independent. The programmer does not have to clean up the memory either. The so-called garbage collector searches for those that are no longer used data and returns the memory areas of this data to the operating system. Due to such properties of Java, however, one is generally not at the programming restricted; Java just does the work for you. Java comes with a very extensive class library. A class library is a collection of small subroutines which you can use can be used to do many frequently recurring tasks and thus do not have to program yourself. Java is already able to deal with networks and especially with the Internet. Java also includes components for graphical user interfaces. This class library is part of the Java Runtime.

Environment and thus automatically available on every computer, the Java programs can perform.

But where do you get the Java Compiler and the Runtime Environment from? For Windows users: Both are available at <http://java.sun.com>. To do this, select “Java SE” in the download area. SE stands for Standard Edition. The Compiler is included in the Java SE Development Kit, or JDK for short. That’s already there suitable runtime environment, JRE for short, included. This means that you only have to download the Runtime Environment for computers on which you only want to run Java programs, but not write them. You can also go here download the documentation for the class library.

The simplest way to generate Java source code is to use a simple text editor such as Notepad. However, it is important that it is an editor which saves the data as pure text (file extension under Windows often .txt). Open Office Writer or Microsoft Word, however, are not suitable, because the files produced with them contain additional information such as font, color, size and anything else amount of information about the document itself. With this additional information can the Java Compiler, however, does nothing at all and the source file as such not seen. When saving, it is also important that the file has the file extension .java must have for the compiler to work with it. Besides, there are several essentials more suitable text editors than notepad. A text editor suitable for programming should namely can display line numbers. This is very useful when debugging, too but later. It is also very helpful if the text editor highlights the commands in color. This is called syntax highlighting.

In addition to simple text editors, there are also so-called IDEs. IDE stands for Integrated Development Environment, in German: development environment; So a program in which tools for programming are already built in. Such IDEs help you to keep track of several open source texts and can often also automate smaller tasks for the programmer. It is particularly pleasant that IDEs generally have a button that displays the automatically compiles and executes the current file. They can also often point to errors in Programming. We have a list of editors and IDEs for you in the appendix collected.

The source text consists of five lines. The first thing you notice is that some lines are indented. This indentation is for readability only and is for Java

Compiler irrelevant. More about indenting is discussed at the end of the introduction. Basically, however, is the following: In the first line something is written and

behind it a curly bracket opens. The counterpart to this bracket is on line five. Here the so-called block is closed again. All lines

in between are indented by at least one level to make it clear that these lines are inside this block from lines one to five. The same applies to line

three, which is inside the block from the end of line two to the fourth line.

For this reason line three is indented again. Let's get to the content:

Line one starts the class called Hello. The class is public. This means that other program parts may also use this class. More details will follow in the chapter on object orientation. It is important that it is described in the source text.

The public class Hello, also the file that contains this source code is called Hello.java otherwise the compiler reports an error. Everything else takes place inside this class. A so-called method with the name main is specified in the second line. Methods are functional units that can be used again. For example would be Make coffee a functional unit that can be reused many times over for mine personal butler. More about methods is dealt with in the Methods chapter. This method is also public here, so it can be used from other parts of the program will. In particular, this method is carried out by the Java Runtime Environment, when the Hello.class file compiled in bytecode is to be executed. The main method is not only public but also static. What this means, however, can only be explained in the chapter on methods. The third word on the line is called void and stands for the return value. This is also covered in the chapter on methods. But so much in advance: If I write myself a method that will give me, for example calculates something so I also expect her to give me the result of where I've used this method from. The main method, on the other hand, is not an auxiliary function but the starting point of the entire program.

However, the main method can certainly receive information from the program start. This information is called parameters. So when you start a program with a file - for example, double-click on a video file - the program - in this case the video player - with this parameter communicated with which file it was started, so that the program will then also process this file. These start parameters are converted into a list of character strings by the runtime environment written. Strings are referred to as strings in Java and are well known to us meet again in the first chapter. The square brackets behind the string mean that it can be more than just a string. This is discussed in the chapter on arrays treated. We call this list of strings args. That is any chosen one name that stands for Arguments. Any parameters passed to our program are available as a list of character strings under the name args. There but we don't do anything with this args in the following, we absolutely

don't care, whether and which parameters we get passed. After all, the program is only supposed to output "Hello World" and that is exactly what the next line does:

In line three we are already using the first method from the Java class library provided. We pass the string "Hello World" to the method `println`, which actually outputs the transferred character string in the text console. However, since the `println` method was not programmed in the `Hello` class, but entirely is different, at the beginning of the third line we first have to tell the compiler where it is can find this method. In fact, this method is in the `System` class in the class library and there again within out, hence the longer name. It is also interesting that the line ends with a semicolon has been. This is because all commands in Java must end with a semicolon and line three is the command "Execute the method `println` with the transfer value `Hello World` off ". Lines one and two do not have a semicolon because they do not contain one immediate Java command, but define a Java class and a method. For what you will learn these terms in a later chapter. As stated at the beginning of the section, it is not necessary at all to do all of these knowing or understanding the details right away. The underlying issues are dealt with in detail in the chapters of this script and it may be interesting after a few chapters to come back here and understand in detail what is actually there. The most important thing about this section is line one and two plus the closing brackets on lines four and five. Because every Java program is started in the `main` method and therefore everything takes place in the area between lines two and four at the beginning. In other words, every Java program contains these first two lines. At the beginning, however, it is completely sufficient to accept that it is so. The exact meaning of the first two lines can only be found in the first chapters once irrelevant.

First of all, a few words about the command line: In the command line you enter the command to be executed by the computer directly from the keyboard. The command line is completely text-based and was there long before graphical user interfaces. Under Windows, the command line is a relic from the old days, keyword MS-DOS. The command line is still very popular under Unix-based operating systems powerful tool. Before we get into translating and executing, considerwe the most basic functions of the command line under Windows as well as Linux and

MacOS:

Under Windows you start the command line with Start -> Run -> cmd. Im now. When the window that appears, you are always in a folder on the hard drive whose. Name and location is displayed to the left of the>. For example, enter notepad here and press Enter, Notepad will open. With the command cd you change directories, for example you change with cd C: \ Programs in the program folder and with cd forty-two into the subfolder named forty-two of the current folder, if such a subfolder exists. With cd .. you switch to the higher-level folder. You can use dir to see which files are in the current folder. If these commands are new to you, it is advisable to experiment with them a little. You can also use the tab key to enter file and folder names complete what saves typing. Exit the command line with exit. Under Unix-based operating systems, the command line is usually called Terminal or Bash.

When you start this, you are also always in a folder on the hard drive. Here, too, there is the cd command to switch between folders. To display the files in a folder, use the command ls. The tab key is even more used here, because it can complete commands as well as file and folder names and quickly double-press the tab key shows you the possible alternatives if there is more than one file or folder name to which you could complete what you typed. Here, too, one ends command line with exit.

It happens to you every now and then at the beginning that you make a change in your makes source code, but then determines that it has not arrived in the program. In this case you should first check whether you have saved the new version of the source text and whether the change is not just in the open editor. In addition, you have to of course, translate the file each time with the compiler. Please note that the compiler only creates a new .class file if the new version the source code has no errors. Otherwise only error messages are output and if you now run the .class file, you start the previous one version. Especially with the JCreator and the "Compile and Execute" function this happens to you often. Should the program not show changed, one should look for compiler errors. The more important use of comments, however, is leaving notes, how the source code works and why something was made the way it was in one place it just stands there. At the beginning, one completely underestimates the importance of comments.

But you shouldn't forget that you are not only commenting for others but mainly for yourself. After a few days you can completely forget how a complicated method works and relies on your own comments. At the commenting, it is important not to comment on the what but on the why. So it's not at all useful if you add an addition as a comment writes "here two variables are added", because you can tell by yourself. It is much better if you write why the variables are added, for example "the sum." is the number of all entries in the list, there is no further reading in ". Only by Such comments can later be understood why the addition was done at all power. It is also very helpful if the variables used are not a or b but for example line number or max_number_users. By these so-called "speaking variables" become much more comprehensible in the source text. The additional typing becomes princely due to less frustration when looking for errors rewarded. Another measure to provide an overview is indenting and formatting the source text. Already at Hello World we got to know that you can do everything within a blocks happened to indent one level deeper. An indentation of four spaces is common. The consistent indentation ensures that you have a better overview of the source text has, since the indentation already makes it clear which source text lines of which depend. In addition, it often happens at the beginning that you forget parentheses. This then becomes visible much more quickly through the consistent indentation. One can search for errors for minutes, which can be identified by properly indented source text within.

Seconds would discover. In addition, you should always format everything you write in source text according to the same rules, for example that a closing curly bracket always on a new line and it has the same indentation as the element to which the opening bracket belongs. Whether you stick to our examples with these formatting rules or doing it differently is irrelevant. The important thing is to do it consistently. This discipline is rewarded with making typing errors and forgotten characters immediately apparent jump once you've got used to a type of formatting.

In computer science, in addition to the primitive data types that we have already discussed in the first chapters and, for example, how to save individual numerical values serve, still so-called abstract data types for the representation of more complex data or quantities thereof. These abstract data types are extended data types that are used for many

algorithms and problem solutions are required or only enable efficient problem solving. Basically, abstract data types are initially descriptions, such as these data structures are built, what they are for and how they work. However, in many programming languages as well as in Java there are already pre-implemented extended data types. But in a computer science course you should at least once yourself simple abstract have implemented data types to develop a deeper understanding of them. A simple abstract data type is the so-called list, which is similar to an array several can accommodate elements of a certain type. Unlike an array, a list can contain a variable number of elements and thus the length the list can be changed dynamically, which is not possible with an array. Other important data types include stacks, queues, trees, heaps or associative arrays (hash maps). The theme of the abstract data types is discussed later in the course in the lecture Algorithms and data structures still deepened.

A temporary variable current element of the type ListenElement is created, which initially points to the head of the list. In the while loop, the respective element to be worked on. At the end of the loop body, the reference of the current element is now pushed onto its successor. Not until the last element the value of this variable then becomes zero and the loop is exited, since the repetition condition is no longer fulfilled.

For many other operations, such as inserting, overwriting or deleting items in the list, there are two important things to consider. It must be ensured that the rest of the list is still intact after each operation. So should for example, after deleting an element, not all element after the deleted one element is still in the list and cannot be removed. Also have to exceptional cases are carefully observed. These are operations on the head as well as on the end the list for which the procedure may have to be adjusted. As with many abstract data types, there are also various variations in the lists. So there are also lists with list elements that link their successors and their predecessors, or lists in which the last element points to the list header again, and thus creates a ring.

While loops have been offered to iterate over list elements, since one thus could go through the individual elements one after the other, this concept now works for trees no longer, since every node in a binary tree is not just one, but also may have two or possibly no successors at all. The concept is used here of recursion to completely traverse a tree.

There are different possibilities of how trees can be traversed recursively, however, all have in common that a node is evaluated using a method and the method is also called recursively with its successor node. This concept works because every node in a tree is also a root of a subtree, namely the tree consisting of this node and all of its subsequent nodes. Such recursive methods are always called on the root node of a tree. Recursive the successor nodes of the root are then called, and so on, until finally the method calls reach the leaves and end the recursion. The following code example enables all values in an integer binary tree to be summed up. In real use, such methods should also be added to the corresponding tree class.

Trees become particularly powerful as a data structure when they are also combined with a order relation are provided. For example, for a binary tree for integer values specify that each right successor node of a node must always contain a larger numerical value than the current node and the left successor node a number less than or equal to the current node. This property must also be used for all subsequent nodes also apply. Such a tree is called a search tree, as it is suitable for efficient searches on amounts of data. The efficiency of this search is based on the fact that starting from the root node the recursive search method always choose exactly one of the two possible subtrees must to continue the search there as long as the value has not yet been found, there in the other subtree all values are too large or too small. That means at during the search, the algorithm only moves along a branch and does not visit that branch complete tree with all elements. This is comparable to searching for a name in one

Phone book. If you open the phone book in the middle when searching for “Wagner” and if you find names beginning with the letter "M", it is clear that you only have in the back half of the book has to look further and the front half no longer must take a closer look. The search begins at 23, since the value you are looking for is smaller, it must be in the left subtree lie. The successor node 17 is still too big, so the search is in the left subtree continued from 17. The node 14 is too small, so its right subtree must be contain wanted value. One more step later, the destination node is reached. However, the introduction of an order relation on a binary tree requires the Adjust tree operations to keep order.

3.4. Class structure diagram

A class diagram is used to describe the structure of classes, attributes, methods, and relationships between them. It should be remembered that a class in UML is a template that creates many objects, not a set of existing objects. In other words, a class is primary in the sense that objects are created on its basis, not a class is formed on the basis of existing objects.

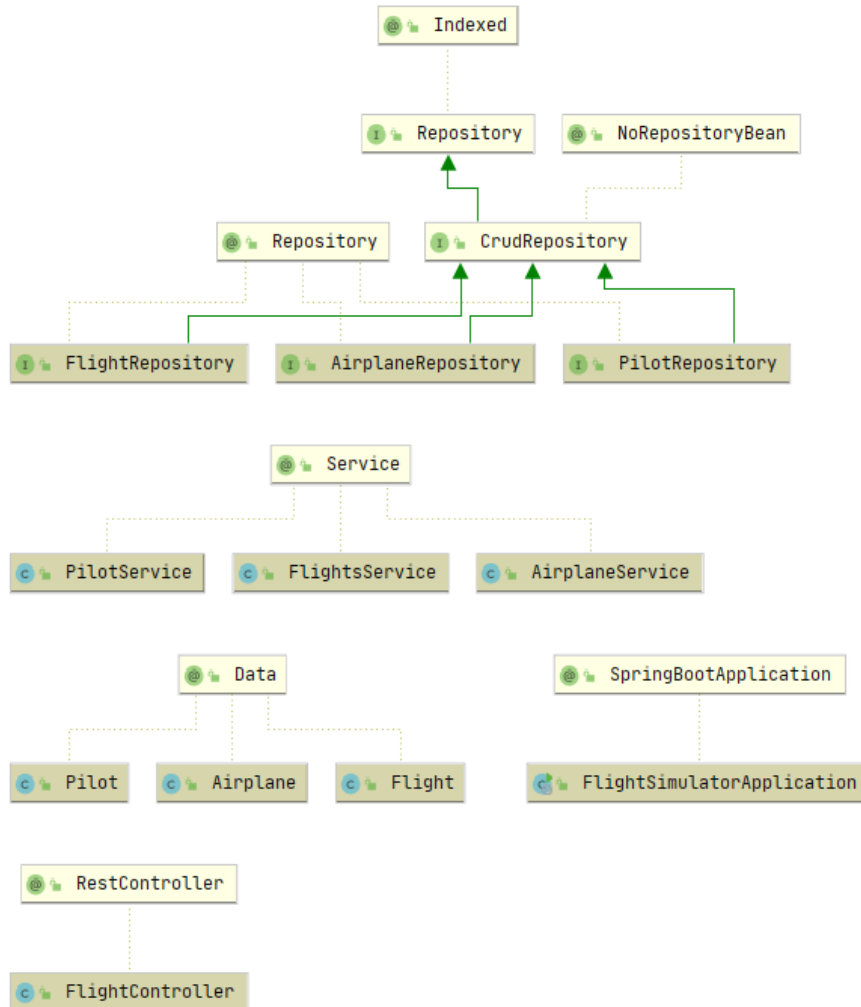


Fig. 3.3. Class diagram

Conclusion on the third part

This section presents three diagrams that describe the system, both hardware and software.

The deployment diagram shows how the entire flight system will work.

The component diagram demonstrated the structural content of the Model-View-Controller template, which is the core of the entire tool.

The class diagram shows all software packages, namely controllers, services and repositories.

PART 4

IMPLEMENTATION OF MICROSERVICES

4.1. Configuring spring boot application

Check configuration of spring boot application. If necessary, specify a directory that will be used by the running application. This directory is the starting point for all relative input and output paths. The project directory is specified by default. By default, the newest JDK from the module dependencies is used to run applications. You can specify an alternative JDK or JRE.

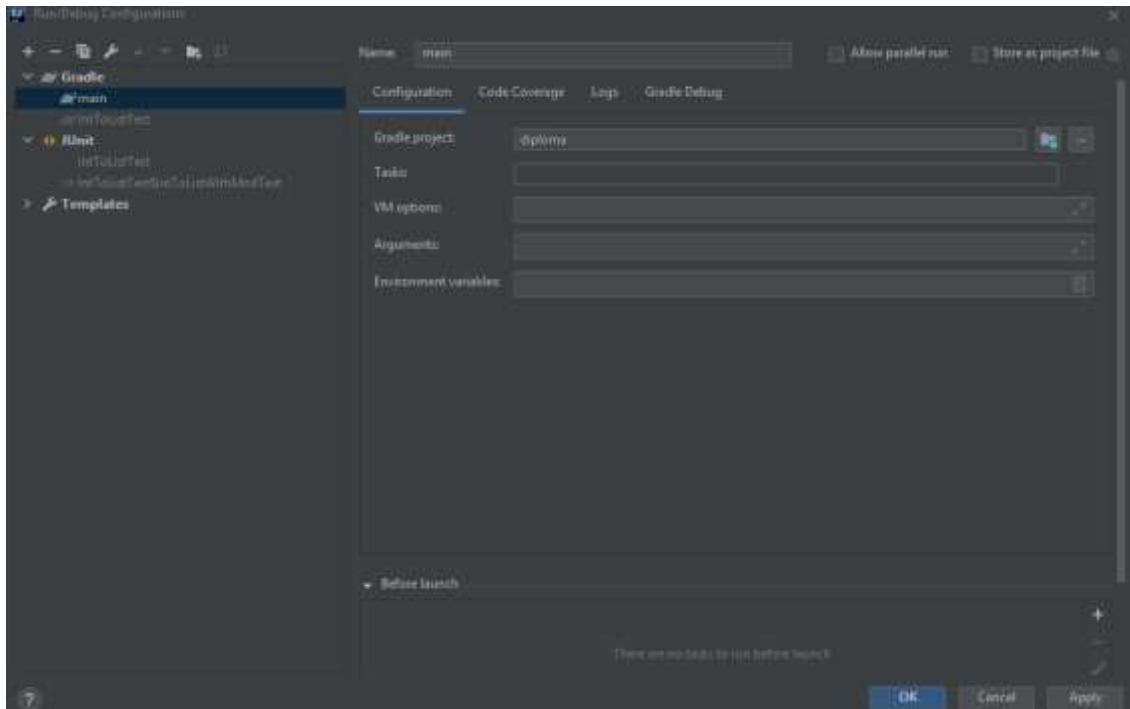


Fig. 4.1 Configuring the project

Clean the latest jar. Uses my project gradle wrapper to execute my project's clean task. Usually, this just means the deletion of the build directory.

```
grow\diploma>gradlew clean
```

Fig. 4.2 Command line

Build the project. It removes build folder, as well configure my modules and then build my project.

```
\diploma>gradlew build
```

Fig. 4.3 Command line

Run spring boot application. To run the application was used the following command in a terminal window (in the complete) directory:

```
diploma>gradlew start
```

Fig. 4.4 Command line

Main page of application. Here we can observe the name of the user who registered, also next to the logout button, and the main flight simulator menu, where you can select all flights, find a flight by number. All functional requirements for the system are implemented.

4.2. Running of the flight system

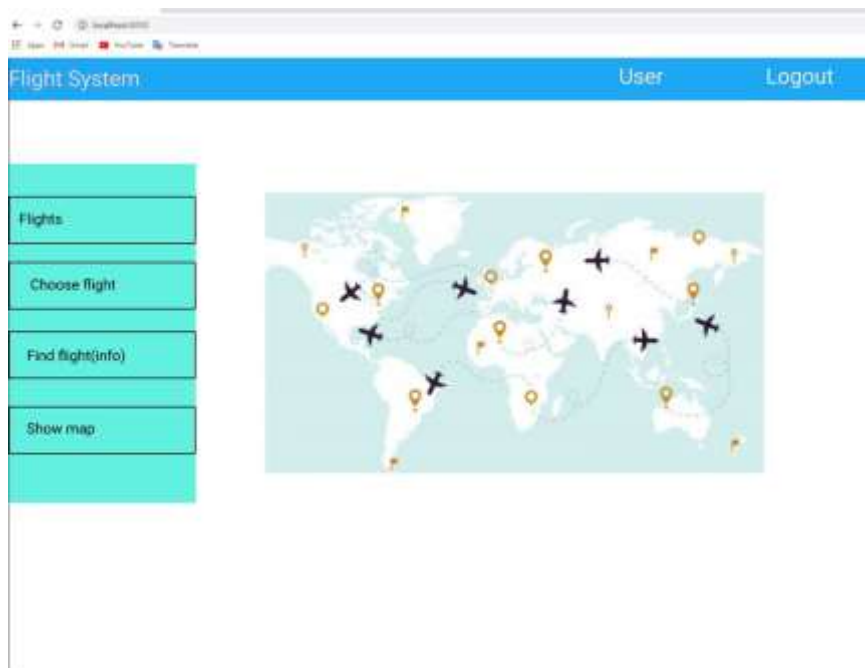
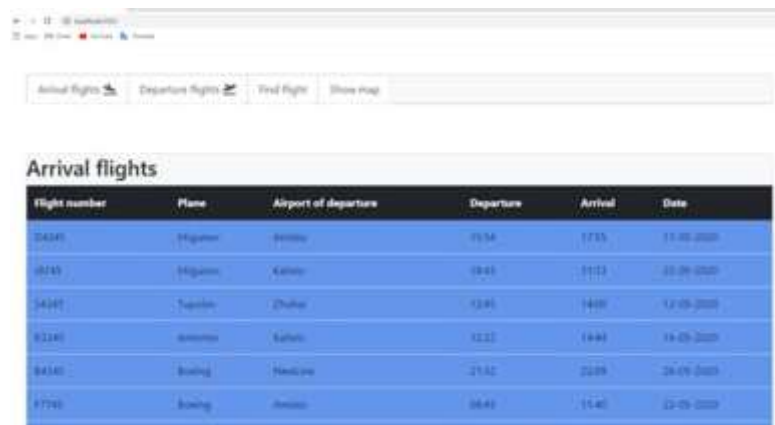


Fig. 4.5 Main page

1. Test table of flights where is the information about arriving and departing flights, as well as their places of departure and destination



Flight number	Plane	Airport of departure	Departure	Arrival	Date
04325	Hughes	Winnipeg	15:54	17:35	11-05-2020
02745	Hughes	Kelowna	08:45	11:13	20-05-2020
34387	Turbin	Dubai	12:45	14:00	17-05-2020
82280	Boeing	Salt Lake	12:22	14:44	18-05-2020
84340	Boeing	Medford	21:42	22:09	26-05-2020
87740	Boeing	Ames	08:44	11:40	22-05-2020

Fig. 4.6 Test table of flights

2. Form for filling in crew details

Flight System User Logout

Set crew for flights

Choose pilot 1

Choose pilot 2

Choose stewardess 1

Choose stewardess 2

Choose number of flight

Submit

Fig 4.7. Page for setting crew

3. Here we fill in the fields of each member of the flight crew where we indicate the name of the first and second pilots, we also choose two flight attendants and be sure to specify which flight the crew will be attached to for future delivery.

Flight System User Logout

Set crew for flights

Choose pilot 1 Antony Doil

Choose pilot 2 John Brew

Choose stewardess 1 Gabriella Sally

Choose stewardess 2 Stefany Foble

Choose number of flight D4345

Submit

Fig.4.8 Form crew for flights

4. In the completed form, we selected first pilot Anthony Doyle, who was assigned to flight D4345. We go into the system under his name and check that the flight was assigned to a specific crew member.



The screenshot shows a web interface for a flight system. At the top, there is a blue header bar with the text "Flight System" on the left, "Antony Doil" in the center, and "Logout" on the right. Below the header, the text "My flights" is displayed. Underneath, there is a table with a light blue background and a white header. The table has four columns: "Flight number", "Departure date", "Place of transer", and "Date of transer". The first row of data contains the following information: "D4345" for the flight number, "10-12-2020 / 17:33" for the departure date, "st. Kalinova 5/12" for the place of transer, and "10-12-2020 / 09:00" for the date of transer.

Flight number	Departure date	Place of transer	Date of transer
D4345	10-12-2020 / 17:33	st. Kalinova 5/12	10-12-2020 / 09:00

Fig. 4.9 Table of pilot flights

Conclusion on the fourth part

Before the hour of the implementation of the fourth part of the project, a prototype was implemented for monitoring air flights to airport and setting crew for this flights. The software product allows the work personnel to be amazed when they arrive and send a flight. In the tables, all the necessary information about the flight is required for any flight, the date is indicated, the hour by the hour, the flight number, the model of the letter. When registering a crew for a specific flight, information is sent to the page of a crew member for which flight he is registered.

The authorization side is shown, if you can log in to the system. This package also has a listing of the main packages of programs. All the necessary classes for displaying the Model-View-Controller pattern are presented in the new one.

CONCLUSIONS

In the course of this work, Java technologies, Spring framework, MVC 4, work with REST-services, approach to development through testing, basics of functional programming were studied.

In the first part was analyzed the concept of an aircraft crew and their delivery on board. It also shows the relevance of flight monitoring tools, which solve the main security problem. The principles of crew delivery on board and existing solutions are considered. These tools are analogs of those used within one airline.

Second section clearly sets out the functional and non-functional requirements for web-services. The list of functional requirements was grouped into a single list, which includes items on authorization, home page, and content. Non-functional requirements are divided into separate blocks, thanks to which the future monitoring tool will be safe, accessible, portable, reliable and monitored.

The work of the crew during the flight and their duties during the flight were also studied. It can be stated that the main purpose of this work was fulfilled - a means of monitoring flights and setting crew for this flights, which has rich functionality, was developed.

Thanks to the presented tool, the airport staff will be able to track flights in a convenient way, as well as observe flights in real time. A very important point is the multilingual nature of the program, because it fully allows foreign employees to work.

The software product allows the work personnel to be amazed when they arrive and send a flight. In the tables, all the necessary information about the flight is required for any flight, the date is indicated, the hour by the hour, the flight number, the model of the letter. When registering a crew for a specific flight, information is sent to the page of a crew member for which flight he is registered.

REFERENCES

1. Typical ship organization [Internet resource] / Website: <https://fas.org/>; Access mode: <https://fas.org/irp/doddir/navy/rfs/part04.htm>, free.
2. Web Services [Internet resource] / Website: <https://www.w3.org/>; Access mode: <https://www.w3.org/DesignIssues/WebServices.html> , free
3. What is a Functional Requirement? [Internet resource] / Website: <https://www.guru99.com/>; Access mode: <https://www.guru99.com/functional-requirement-specification-example.html>, free.
4. Basic concepts of network technologies [Internet resource] / Website: mobiz.com.ua; Access mode: <https://mobiz.com.ua/bazovi-poniattia-merezhevykh-tekhnologij.html>, free.
5. Basic concepts of network technologies [Internet resource] / Website: mobiz.com.ua; Access mode: <https://mobiz.com.ua/bazovi-poniattia-merezhevykh-tekhnologij.html>, free.
6. Vince Barnes Why Create A Web App?/ Vince Barnes // Non-Technical Introduction. – 2009. – #1. – 1
7. Vince Barnes Where Do I Put My Web App?/ Vince Barnes // Non-Technical Introduction. – 2009. – #2. – 1
8. Midphase What is Web Hosting? [Web resource]. – Hosting Services, Inc. Midphase. – Access mode: <https://www.midphase.com/website-hosting/what-is-web-hosting.php>
9. Pete Zaborszky The 5 Best Ways to Build a Website in 2014 [Web resource]. – Access mode: <http://www.make-a-web-site.com/5-best-ways-build-website-2014/>
10. Joe Burns To Use or Not to Use a Database? / Joe Burns // Database Basics. – 2009. – #1. – 4
11. Curtis Dicken Top 5 Databases for Web Developers / Curtis Dicken // Database/SQL Primer. – 2011. – 1

12. Glenn Fowler A Flat File Database Query Language / Glenn Fowler // AT&T Labs Research. – 1994. – 10
13. SQL Server [Web resource]. – Microsoft, 2014. – Access mode: <http://www.microsoft.com/en-us/server-cloud/products/sql-server/>
14. PostgreSQL 5.7 Reference Manual [Web resource]. – Postgres Corporation, 2015. – Access mode: <http://dev.postgresql.com/doc/refman/5.7/en/>
15. Content management system [Web resource]. – DocForge, 2010. – Access mode: http://docforge.com/wiki/Content_management_system
16. Guy W. Lecky-Thompson Just Enough Web Programming with XHTML, Java and MySQL / Guy W. Lecky-Thompson. – USA. Boston, MA: Course Technology, 2008. – 449
17. Pete Zaborszky How to Make a Web App [Web resource]. – Access mode: <http://www.make-a-web-site.com/how-to-make-wordpress-website/>
18. Pete Zaborszky Why Spring Boot is the Best Choice for a WebApp in 2014 [Web resource]. – Access mode: <http://www.make-a-web-site.com/spring-boot-best-choice-website-2014/>
19. Pete Zaborszky How to Make a Spring Boot WebApp [Web resource]. – Access mode: <http://www.make-a-web-site.com/make-joomla-website/>
20. Dan Rahmel Beginning Spring!: From Novice to Professional / Dan Rahmel. – USA. NY: apress, 2007. – 494