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: **"Засоби розповсюдження повідомлень про доставку екіпажів в середовищі інформаційоної мережі авіакомпанії"**

Виконавець: _____ Куксенко Р.С.

Керівник: _____ Сураєв В.Ф.

Нормоконтролер: _____ Надточій В.І.

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

_____ Куксенко Р.С.

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: **"Distribution of crew deliveries messages in the airline's information network environment"**

Completed by: _____Kuksenko R. S

Supervisor: _____ Suraev. V.F.

Standard's Inspector: _____ Nadtochii V.I.

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

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

1. Тема дипломної роботи: "Засоби розповсюдження повідомлень про доставку екіпажів в середовищі інформаційної 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. Вхідні дані до роботи (проекту*): SQL процедури неактуальної версії авіакомпанії МАУ. Сучасні методи динамічного оновлення Web сторінки за допомоги мови програмування Python та його Web фреймворка Flask. Microsoft SQL Server для взаємодії з базою даних. Операційна система Linux, Windows.*

4. Зміст пояснювальної записки: *Вступ, огляд старої версії системи розповсюдження повідомлень про екіпаж, огляд задач які покриває система розповсюдження повідомлень про екіпа. Огляд існуючих технологій що можуть бути використані в завданні. Детальний розбір вибраних технологій та демонстрація їх роботи. Огляд та тестування готового проекту.*

5. Перелік обов'язкового графічного (ілюстративного) матеріалу: *Зображення що ілюструють працездатність проекту, графічні матеріали у вигляді презентації у форматі ppt.*

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

Kuksenko Roman Sergiyovuch

1. Thesis topic: "**Distribution of crew deliveries messages in the airline's information network environment"** approved by the Rector's order of 25.09.2020 p. № 1793/ст

2. Thesis to be completed between з 1 жовтня 2020 р. до 25 грудня 2020 р.

3. Initial data for the project (thesis): *SQL procedures of the outdated version of UIA. Modern methods of dynamic updating of the Web page using the Python programming language and its Flask Web framework. Microsoft SQL Server to interact with the database. Operating system Linux, Windows*

4. The content of the explanatory note (the list of problems to be considered): *Introduction, review of the old version of the crew messaging system, review of the tasks covered by the crew messaging system. An overview of existing technologies that can be used in the task. Detailed analysis of selected technologies and demonstration of their work. Review and testing of the finished project*

5. The list of mandatory graphic materials: *Images illustrating the efficiency of the project, graphic materials in the form of a presentation in ppt format.*

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 існуючих проблем системи	16.10.20-18.10.20	
4	Проаналізувати відомі підходи та методи створення системи розповсюдження повідомлень в інформаційній мережі	19.10.20-03.11.20	
5	Розробити початкову версію системи розповсюдження повідомлень	04.11.20-15.12.20	
6	Порівняти потенційні технології що можливі для реалізації проекту	16.12.20	
7	Виконати перевірку на працездатність основних завдань системи розповсюдження повідомлень в інформаційній мережі. Розробити рекомендації та оформити пояснювальну записку.	06.12.20-12.12.20	
8	Оформити графічну частину записки та подати матеріали работи на антиплагіатну перевірку матеріалів	13.12.20-14.12.20	
9	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 _____ Куксенко Р.С.
(підпис випускника)

6. TIMETABLE

#	Completion stages of Degree Project (Thesis)	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 search and study of scientific and technical literature on the topic of work	9.10.20-15.10.20	
3	Develop theoretical material on existing system problems	16.10.20-18.10.20	
4	Analyze the known approaches and methods of creating a system of message distribution in the information network	19.10.20-03.11.20	
5	Develop an initial version of the messaging system	04.11.20-15.12.20	
6	Compare potential technologies that are possible for project implementation	16.12.20	
7	Perform a check on the operability of the main tasks of the messaging system in the information network. Develop recommendations and make an explanatory note.	06.12.20-12.12.20	
8	Make a graphic part of the note and submit the materials of the work for anti-plagiarism	13.12.20-14.12.20	
9	Get a review and feedback from the manager. Provide work materials for the department.	15.12.20 18.12.20	

7. Assignment issue date: 1.10.2020

Diploma Thesis Supervisor _____ Suraev V.F.

(Signature)

Assignment accepted for completion _____Kuksenko R.S.

(Student's Signature)

ABSTRACT

Explanatory note to the thesis "Distribution of crew deliveries messages in the airline's information network environment":

92 pp., 53 figures, 23 references, 10 appendices.

Object of research: Development tools, methods and technologies that could be used to develop and maintain distributed messaging delivery system.

Subject of research: A prototype of deliveries messages system.

Purpose: Develop a prototype of messaging delivery system for airline, namely Ukrainian International Airlines, to ensure in advantages and improvements achieved after updating existed system to new approaches and technologies.

Research methods: Overview an existed technology that possibly could be used for achievement assigned tasks and detail analyzing some of them.

The results of the master's work are prototype crew deliveries messages system, based on new methods and technologies that could be used as a part of real system by Ukrainian International Airlines.

DEVELOPMENT, PYTHON, FLASK, DATABASE, CLOUD, DOCKER, AWS

CONTENT

LIST OF SYMBOLS, ABBREVEATIONS, TERMS

SQL – Structured Query Language

DBMS – Database Management System

UIA – Ukraine International Airlines

PHP – PHP Hypertext Preprocessor

AWS – Amazon Web Services

HTML – Hypertext Markup Language

SDK – Software Development Kit

XML – Extensible Markup Language

JSON – JavaScript Object Notation

DDL – Data Definition Language

DML – Data Management language

DCL – Data Control Language

SSO – Single Sign-On

API – Application Programming Interface

WEB – World Wide Web

I/O – Input / Output

HTTP – Hypertext Transfer Protocol

URL – Uniform Resource Locator

DRY – Don't repeat yourself

MVC – Model–View–Controller

AWS – Amazon Web Services

GCP – Google Cloud Platform

CDN – Content delivery network

RAM – Random-access memory

EC2 – Elastic Compute

IP – Internet Protocol

IDE – Integrated Development Environment

QR code – Quick Response code

INTRODUCTION

Ukraine International Airlines connects over 80 destinations in Europe, Asia and the Middle East, as well New York City and Toronto from its base at Boryspil Airport. UIA carries nearly 7 million passengers over a year.

Actuality of theme. It is hard to track, support and serve such huge number of passengers. It is required to use modern technologies not only in aviation direction but also in computer networks, online services, administrator tools, etc. To store such big amount of data it is not acceptable to use some kind of Excel sheets or other simple tools. It is required to use databases and write correct, efficient and comprehensives for developers SQL queries So, it is highly important to have a unique internal system and a convenient way of its serving.

The first part describes an existed methods and technologies used by UIA, its architecture and examples of SQL queries.

The second part describes possible tools, technologies that could be used for an efficient messaging delivery system. Given a detailed explanation of modern programming language on a backend side. Listed types of databases. Described possible ways of linking programming language Python with SQL server. Represented modern Python web frameworks and given an example of using each of them. Explained the advantages of using Docker technologies, detailed how to build and upload a Docker image to the Docker Hub. Outlined a cloud technology such as AWS EC2 and given a detailed example of it using.

The third part is dedicated for explanation of a practical task. It contains detailed step-by-step instructions to develop a prototype of a massages delivery system. Described a way of deploying a ready project to the cloud inside a docker. Listed a set of available functions and features presented in the prototype. Specified a QR code that routs to the deployed project.

The result of this work will be a prototype of the of messaging delivery system that can deliver information about the crew members. The system will be written with a help of Python programming language and the MsSQL database. This system will be able to interact with administrators, scoreboard, web sites, etc., which will be able to

retrieve, update and modify information about crew members, their flights, number of pilots, stewardesses, flight schedules of each crew member, etc. The goal is to test new technologies and shows that Python programming language could be used by Ukraine International Airlines.

PART 1

EXISTED SYSTEM OF MESSAGE DELIVERING IN UIA

1.1 Functions of the messaging system

Every day for every flight all pilots and cabin crew have to be delivered to airport before departure and returned to home after arriving. Messaging system is extremely important part of the any airlines. Such systems provide a possibility to add, edit, view passenger's information for each flight.

All information about crew such as passengers count, name, surname, address, email, phone number for each passenger are stored in airline databases. It is very important to collect and manage this information in secure and efficient way.

UIA connects Ukraine to over 80 destinations in Europe, Asia and the Middle East, as well as to New York City and Toronto from its base at Boryspil Airport, and also operates domestic flights [1]. Ukraine International Airlines is no exception. In 2017, UIA carried nearly 7 million passengers and, therefore, increased traffic by 15.5% in comparison with the 2016. The UIA passenger traffic via Kyiv Boryspil International Airport was increased by 19% in comparison with 2016, up to 6.18 million passengers.

In the UIA, more than half of the company's employees (thousands of people), are users of the crew delivery system (now there are fewer employees due to quarantine). The work of all pilots and cabin crew, without exception, depends on this system.It is important to provide information fast and with no errors and delays. And not only the staff of this most important division of the company. Employees of the flight management department, transportation department, HR (Human Resources department) are users of this system.

To manage 7 million passengers, it is very important to make highly efficient databases structure, local network structure, interaction with a database, delivering

massages to the dashboards, between the department, give possibility to add, edit, update and view stored information in the databases.

The functions of the Airline messaging systems include:

- Adding information about crew members;
- Updating the already committed data;
- Sharing information inside an information network;
- Assigning of the permissions for each employee of the UIA;
- Real-time information updating and messages delivery.

1.2 Used programming languages

Ukraine International Airlines is using Structured Query Language to develop and interact with databases. SQL is used to interact with the database in almost all projects that use databases. Structured Query Language requests are used to update, edit, create or retrieving data from the company database. A lot of relational database management systems use SQL. Some of the most popular listed below:

- Oracle DB;
- Sybase;
- Microsoft SQL Server;
- SQLite;
- PostgreSQL.

It is possible to use standard SQL commands such as Create, Update, Delete, Select, Insert, Drop to do anything that needs to do with the database [2].

PHP is a server-side language embedded in HTML. It is also could be used for dynamically update the content, interact with database, creation of a separate website. PHP works perfect with a lot of databases listed above. PHP has a lot of advantages but also disadvantages. For now, this popularity of this language is decreasing every year. By the way PHP is good for integration with SQL language. It is not hard to perform complex queries with a huge result. PHP can be not only one language in the backend, it is also possible to integrate PHP with C++ or Java.

Fig. 1.1 is showing an example of a SQL query. This query creates table “TEST_UIA” and adds 3 rows that correspond to some passenger information. After it, this table print on the screen.

```
1> DROP TABLE TEST_UIA;
2> GO
1> CREATE TABLE TEST_UIA (id INT, name NVARCHAR(50), email CHAR(30), phone CHAR(10))
2> INSERT INTO TEST_UIA VALUES (1, 'Ivanov Ivan', 'ivanov@gmail.com', "3806687429");
3> INSERT INTO TEST_UIA VALUES (2, 'Pupkin Aleks', 'apupkin@gmail.com', "3806620998");
4> INSERT INTO TEST_UIA VALUES (3, 'Sidorov Oleg', 'sidorov@gmail.com', "3809547220");
5> SELECT * FROM TEST_UIA
6> GO

(1 rows affected)

(1 rows affected)

(1 rows affected)
id          name                                               email                          phone
----------- -------------------------------------------------- ------------------------------ ----------
          1 Ivanov Ivan                                        ivanov@gmail.com               3806687429
          2 Pupkin Aleks                                       apupkin@gmail.com              3806620998
          3 Sidorov Oleg                                       sidorov@gmail.com              3809547220

(3 rows affected)
```

Fig. 1.1. Basic SQL queries

In Ukraine International Airlines as a backend language which connects a databases and UI presentation is PHP. PHP is a recursive acronym for "PHP: Hypertext Preprocessor" [3].

The PHP language has a C-type syntax. There are lots of big companies that still use PHP, for example Google, Microsoft, Yandex, Amazon still using PHP language in some of their project. It is possible to visit a GitHub page of that IT- giant to ensure with it.

In the Fig. 1.2 is presented an easy code written with a help of PHP language. In this example PHP is integrated into HTML code. The variables $crew_count and $flight_direction should be changed to actual values after running.

In a real example values of $crew_count and $flight_direction could be not hardcoded but will received from the UI or database. This type of dynamic update web pages was introduced in 1994.

```
<!DOCTYPE html>
<html>
<body>

<?php
$crew_count = 5;
$flight_direction = "KIEV-KHARKOV";
$expected_arrival_time

echo "Crew count on the board:" $crew_count. "<br>";
echo "Flight direction:" $flight_direction. "<br>";
echo "Expected arrival time:" $expected_arrival_time";
?>

</body>
</html>
```

Fig. 1.2. Basic SQL queries

Php commonly used for:

- Creation of scripts for execution on the server side. PHP is traditionally and most widely used in this way. To do this, you will need three things. PHP interpreter (CGI program or server module), web server and browser. In order to view the results of PHP scripts execution in a browser, you need a working web server and PHP installed. You can view the output of a PHP program in a browser by getting a PHP page generated by the server. In case you are just experimenting, you may well use your home computer instead of a server. See the Installation Tips chapter for more details.

- Creation of scripts for execution on the command line. You can create a PHP script that can run without a server or browser. All you need is a PHP parser. This way of using PHP is ideal for scripts that need to be executed regularly, such as using cron (on * nix or Linux platforms) or using the Task Scheduler on Windows platforms. These scripts can also be used for simple word processing tasks. For more information, see the chapter Using PHP in a Command Line Environment.

- Creation of client-side windowed applications. PHP may not be the best language for creating such applications, but if you know PHP very well and would like

to use some of its capabilities in your client applications, you can use PHP-GTK to create such applications. Similarly, you can create cross-platform applications. PHP-GTK is a PHP extension and does not ship with the main PHP distribution. If you are interested, visit the »PHP-GTK site.

1.3 Database tables definition

Each database has an object of data with parameters. This objects names “Table”. Each Table should be connected to each other. In a relational database, a relationship is formed by matching rows that belong to different tables. There is a special row types for that purpose. A relationship between tables is established when a child table creates a “foreign key” column that can refers to the “primary key” column of the other table [4].

Each database table relationship is built based on their foreign key columns. There are three types of table relationships:

- One-to-many relation. Probably the most common relationship. In this case of relation, a row from a parent table refer to multiple rows in a child table. This type of relation will be used in this project.
- One-to-one relation. This type of relation requires that the primary key of the child table should be related via a foreign key to the primary key of the parent table. Not often used.
- Many-to-many relation. This type of relation requires a referencing table that contains two foreign key that refer to two different parent tables. This type of relation is used not so often.

In Ukraine International Airlines there’s the following object types (tables):

- Crew - airline's pilots and cabin crew;
- Flights - scheduled flights;
- Assignments - persons need to be delivered to the flight;
- Delivering - proposals for crew members delivering;
- Trip points - trip points of transportation plans.

Here is a list of properties for each of the table:

Crew:

- id - personal number in Human Resources system;
- description - surname, name and middle name of person;
- address - home address of person;
- email - E-mail address of person;
- phone - phone number for direct alerts.

Flights:

- id - flight identifier in Airline Schedule System;
- description - flight root;
- departure - time of departure from airport KBP;
- arrival - time of arrive to airport KBP;

Assignments:

- id - id of assignment;
- crewid - id from table Crew;
- flightid - id from table Flights;
- date - date of the flight departure.

Delivering:

- id - id of delivering;
- assignmentid - id of assignment from table Assignments;
- type - type of trip: for departure or after arriving;
- check - confirmation or refuse of delivery from crew member.

Trip points:

- id - id of trip point;
- deliveringid - id of delivering from table Deliverings;
- time - time of picking up or dropping off.

Each property has some data type. SQL Data Type is an attribute that specifies the type of data of any object. Each column, variable and expression has they own data type in SQL. It is possible to use these data types while creating tables. Data type for a table column based on its requirement. Here is a datatype for each property used by Ukraine International Airlines:

- id – integer;
- description - varchar(255);
- address - varchar(255);
- email - varchar(50);
- phone - char(10);
- departure – time;
- arrival - time;
- crewid – int;
- flightid – int;
- assignmentid – int;
- deliveringid – int;
- date – date;
- type – int;
- time – time;
- check – int.

So, based on the all properties and objects here is a list of tables:

- Crew (id, description, address, email, phone);
- Flights (id, description, departure, arrival);
- Assignments (id, crewid, flightid, date);
- Delivering (id, assignmentid, type, check);
- Trippoints (id, deliveringid, time);

A foreign key is a key used to link two tables together. This is sometimes also called as a referencing key.

A Foreign Key is a column or a combination of columns whose values match a Primary Key in a different table.

The relationship between 2 tables matches the “Primary Key“ in one of the tables with a Foreign Key in the second table.

Here are foreign keys for described above tables:

- Crew - no foreign keys;
- Flights - no foreign keys;
- Assignments - crewid is foreign key for table Crew, flightid is foreign key for table Flights;
- Delivering - assignmentid is foreign key for table Assignments;
- Tripointps - deliveringid is foreign key for table Delivering.

1.4 Roles. Functions for each role

To easily manage the permissions in your databases, SQL Server provides several roles. They are like groups in the Microsoft Windows operating system. Database-level roles are database-wide in their permissions scope.

To add and remove users to a database role, use the ADD MEMBER and DROP MEMBER options of the ALTER ROLE statement.

There are 4 roles for managing a database in UIA, here is list of that roles:

- Crew administrator - the person who is responsible for maintaining the crew database. Send a request to get the filtered part of the crew members list. Send a request to get the form with information about one crew member. Edit information about one crew member. Insert new crew member record. Dismiss the crew member;
- Flight administrator - who appoints crew members for flights. Send a request to get the filtered part of flights. Send a request to get the form with information about one

flight. Edit information about one flight. Insert new flight record. Cancel the flight. Appoint crew members for flight;

- Transportation administrator - in charge of crew delivery plans making. Send a request to get the filtered part of delivery plans list. Send a request to get one delivery plan. Send a request to get assignments of crew members for one flight. Send a request to get list of crew members, who didn't check in. Create new delivery plan trip point. Edit one delivery plan trip point. Cancel one delivery plan trip point. Change the sequence of trip points of delivery plan;
- Crew member - pilot or cabin crew member. Send a request to get own delivery plan. Send information about check in for the delivery. Send the request to crew dispather about personal information changes. Get notification about delivery has not been checked in yet.

1.5 Examples of SQL procedures

1.5.1 Procedures for crew administrator

Crew administrator requests for list of crew members retrieving from the table Crew. To sharp the database response filters can be built by giving some parameters. If the parameter @crew_id is filled, one row will be extracted from the table. The string parameter @Phone gives posibility retrieve records (one usually) with phone number, which is equal to @phone. If the string parameter @part_of_description is given, all records which include value of this parameter as a part of field [description] will be retrieved. The string parameter @part_of_adderess acts similarly for field [address]. If all parameters are missed the result will be the list of all crew members. The procedure screenshot is showed in the Fig. 1.3.

```
CREATE PROCEDURE CA_ListCrewRequest @crew_id INT=NULL,
                                    @phone CHAR(10)=NULL,
                                    @part_of_description VARCHAR(50)='',
                                    @part_of_address VARCHAR(50)=''
AS
IF @crew_id IS NOT NULL -- looking for crew member by his id
   SELECT [id] AS [Employee ID], [description] AS [Firstname and surname], [phone] AS [Phone number]
    FROM Crew
   WHERE [id]=@crew_id
ELSE IF @phone IS NOT NULL -- looking for crew member by his phone number
        SELECT [id] AS [Employee ID], [description] AS [Firstname and surname], [phone] AS [Phone number]
          FROM Crew
         WHERE [phone]=@phone
ELSE  -- request for all crew members with some details about firstname and surname or/and address
   SELECT [id] AS [Employee ID], [description] AS [Firstname and surname], [phone] AS [Phone number]
    FROM Crew
   WHERE [description] LIKE '%'+@part_of_description+'%'
     AND [address] LIKE '%'+@part_of_address+'%'
```

Fig. 1.3. Crew administrator requests for list of crew members retrieving from the table Crew

Crew administrator requests for new crew member record creating and gives new values for it (Fig. 1.4). The parameter @crew_id indicates the crew member whose record needs to be created. If @crew_id is filled, only message 'No data available.' will be returned. The parameters @description, @address, @email and @phone send to procedure new values of record fields. In case of violating table's restrtictions message 'Data not saved.' will be returned. Otherwise message will be 'Data saved.'

```
CREATE PROCEDURE [dbo].[CA_InsertCrew] @crew_id INT=NULL, @description VARCHAR(255)='',
@address VARCHAR(255)='', @email varchar(50)='', @phone CHAR(10)=NULL
AS
IF @crew_id IS NOT NULL -- to create and fill new crew member card
   BEGIN
      INSERT Crew ([id], [description], [address], [email], [phone])
      VALUES (@crew_id, @description, @address, @email, @phone)
      IF @@ROWCOUNT = 0
         SELECT 'Data not saved.'
      ELSE
         SELECT 'Data saved.'
   END
ELSE
   SELECT 'No data available.'
```

Fig. 1.4. Crew administrator requests for new crew member record

1.5.2 Procedures for crew member

Crew Member requests delivering plan for given date. If the parameters @crew_id is given, plan for this date and crew member will be returned as a table with columns 'Crew Id', 'Departure', 'Arrive', 'Check In'. Otherwise the message 'Data not available.' will be. Depending on the value of field [type] column 'Departure' or 'Arrive'

includes time. Depending on the value of field [check] column 'Check In' can show 'Refuse', 'Yes' or nothing, if the crew member does not check this delivering plan. If the parameter @date is given, transportation task for this date will be retrieved. In the Fig. 1.5 is shown a DeliveringPanRequest procedure Otherwise tomorrow's date will be used. Message 'Plans are missing for the date.' will be if there is no such plan.

```
CREATE PROCEDURE CM_DeliveringPanRequest @crew_id INT=NULL,
                                         @date DATE=NULL
AS
IF @crew_id IS NOT NULL
   BEGIN
      IF @date IS NOT NULL
         SET @date=DATEADD(DAY,1,getdate())    -- implicit conversion date type DATETIME to data type DATE .
      SELECT Crew.[id] AS [Crew Id],
             Crew.[description] AS [Name and surname],
             (CASE [type] WHEN 0 THEN CONVERT(VARCHAR(5),Flights.[departure])
                          WHEN 1 THEN ''
               END) AS [Departure],
               (CASE [type] WHEN 1 THEN CONVERT(VARCHAR(5),Flights.[arrival])
                            WHEN 0 THEN ''
                END) AS [Arrival],
               (CASE [check] WHEN 0 THEN ''
                             WHEN 1 THEN 'Yes'
                             WHEN 2 THEN 'Refuse'
                END) AS [Check In]
      FROM Deliverings JOIN Assignments ON Deliverings.[assignmentid]=Assignments.[id]
                       JOIN Flights ON Assignments.[flightid]=Flights.[id]
                      JOIN Crew ON Assignments.[crewid]=Crew.[id]
      WHERE Assignments.[crewid]=@crew_id
        AND Assignments.[date]=@date
        IF @@ROWCOUNT = 0
           SELECT 'Plans are missing for the date.'
   END
ELSE
   SELECT 'Data not available.'
```

Fig. 1.5. Crew Member requests delivering plan for given date

1.5.3 Procedures for flight administrator

Flight administrator requests for list of flights retrieving from the table Flight (Fig. 1.6). To sharp the databese response filters can be built by giving some parameters. If the string parameter @part_of_description is given, all records which include value of this parameter as a part of field [description]. If the parameter @departure_from exists flights with time of departure later then the parameter value will be returned only. If the parameter @departure_to is filled flights with time of departure [departure_time] less then the parameter value will be retrieved. The parameters @arrival_from and @arrival_to act similarly for field [arrival_time]. Any combination of these parameters is possible. If all parameters are missed the result will be the list of all flights.

```
CREATE PROCEDURE FA_ListFlightsRequest
                 @part_of_description VARCHAR(255)='',
                 @departure_from TIME=NULL,
                 @departure_to TIME=NULL,
                               @arrival_from TIME=NULL,
                               @arrival_to TIME=NULL
AS
SELECT [id] AS [Flight], [description] AS [Route], [departure] AS [Departure], [arrival] AS [Arrival]
  FROM Flights -- flight administrator can use the filters for some part of flight list retrieving
 WHERE [description] LIKE '%'+@part_of_description+'%'
   AND (@departure_from IS NULL OR [departure]>=@departure_from)
   AND (@departure_to IS NULL OR [departure]<=@departure_to)
   AND (@arrival_from IS NULL OR [arrival]>=@arrival_from)
   AND (@arrival_to IS NULL OR [arrival]<=@arrival_to)
```

Fig. 1.6. Flight administrator requests for list of flights

1.6. Disadvantages of existed system

In 2020 there are a lot of programming language that allows to develop new features fast and efficient. PHP language used un UIA has some disadvantages such as:

- Interpreted and not compiled so there is a bit of a performance hit in comparison with other languages;
- Not optimized for desktop apps;
- It is not suitable for giant content-based web applications;
- Runs slightly slower than other programming languages;
- PHP frameworks got to learn to use PHP built-in functionalities to avoid writing additional code;
- Using more features of PHP framework and tools cause poor performance of online applications;
- PHP don’t allow change or modification in core behavior of online applications [6].

Conclusions on the First Part

Ukraine International Airlines uses PHP programing language as a backend language and SQL for interaction with databases. There are a several tables for store and manage crew personal information such as name, surname, personal id, address, email, phone number. Each table contains individual properties which represents a property of the object.

PHP language is very powerful and multifunctional, but now there is a new, modern language, which can make a development faster and more convenient. There are some disadvantages in using PHP in modern project, main cons are:

- Not optimized for desktop apps;
- Runs slightly slower than other programming languages;
- The ease of customization makes it more error-prone and harder to find the errors.

PART 2

DEVELOPMENT TOOLS AND TECHNOLOGIES

2.1 Types of databases

A database is a data structure that stores organized information. Most databases contain several tables, each of them can contain several different fields. For example, a database of some company can include tables for their products, employees, or financial reports. Each of these tables should have different fields related to the information stored in the table.

Almost all web projects or web sites use databases to store their important information. It-giants such as Microsoft, Google, Apple, Amazon have their own databases. Nowadays, this companies try to locate data secure and uses their own datacenters for this purpose.

The first databases were very simple, they were limited to simple rows and columns, like a spreadsheet. Some company use a simple Microsoft Excel for storing a companies' data. However, modern way to store data is use databases. Relational databases allow to access, edit, update, retrieve and perform looking for some data through the databases. Relational databases can execute queries that will involve multiple databases and tables. The first databases could store only text or numeric data. Modern databases allow to store almost any kind of data. It is possible to store even images and videos. It can be performed in way to convert image or video into BASE64 form or just story binary data [7].

Depending on requirements to database, there are following types of databases available now:

- Cloud database;
- Commercial database;
- Object-oriented database;
- Graph database;

- NoSQL database;
- Centralised database;
- Distributed database;
- Personal database;
- End-user database;
- Operational database;
- Relational database;
- Cloud database.

For example, centralized database store in a centralized location and their users could be located in any part of the world. Users can interact with such databases from different locations. In the Fig. 2.1 shown an example of centralized database.

Application procedures help users access data even from a remote location. That type of databases frequently used by the huge web projects.

Such databases are very secured and could include various kinds of authentication procedures for the users. It needs to verify and validate the users, to prevent company's data from the attack or being stolen. Nowadays spends a lot of resources for providing a good security for user's personal information. No one wants that their own personal data will be stolen. To access to database, it can require some specific information or answer on questions written during the user registration. It can be the registration number is provided by application procedures that track and record data usage [8]. The other type of databases is distributed database. This type of database often used to be a corporation such as Microsoft, Google, Apple, Amazon. Data is not stored in one place and is distributed across different sites in the organization. In direct contrast to the concept of a centralized database, a distributed database has contributions from a shared database as well as information retrieved from local computers. These sites are linked to each other through communication channels that help them easily access distributed data.

So, there are two types of distributed databases:

- Homogeneous;
- Heterogeneous.

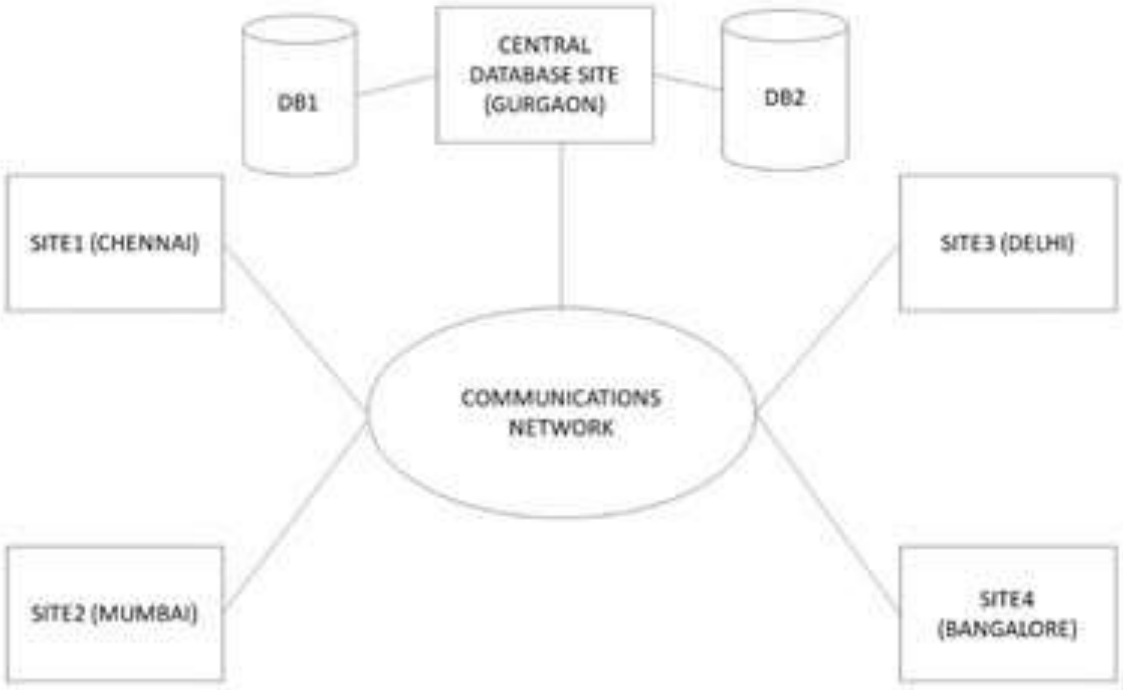

Fig.2.1. Centralized database structure example

Databases that have the same underlying hardware and run the same operating systems and application procedures are known as homogeneous database, for example physical locations are in DDB. Such databases are very secured and could include various kinds of authentication procedures for the users. Whereas operating systems, underlying hardware, and application procedures may differ at different locations in DDB, which is known as heterogeneous DDB, so on.

Hierarchical database - each object with this storage of information is represented as a specific entity, that is, this entity may have child elements, parent elements, and those children may also have child elements, but there is one object from which everything begins. It turns out a kind of tree. An example of a hierarchical database might be an XML document or a computer's file system; I gave an example with a

computer's file system when I looked at the structure of an XML document in the XML Notes section.

Network databases are a kind of modification of hierarchical databases. If you looked closely at the picture above, you probably noticed that only one arrow from the top element goes to each lower element. That is, for hierarchical databases, each child can only have one child. Network databases differ from hierarchical databases in that a child element can have several ancestors, that is, elements above it.

It should be said that databases of this type are optimized for reading information, that is, databases with a hierarchical structure are able to very quickly select the requested information and give it to users. But such a structure does not allow us to go through information as quickly, here you can give an example from life, a computer can easily work with any specific file or folder (which, in fact, are objects of a hierarchical structure), but antivirus scanning of a computer takes a very long time. The second example is the Windows Registry.

The other type of databases is relational databases. These type of databases are classified according to a set of tables, where the data falls into a predefined category. This kind of database is probably the most popular type of databases. A table is made up of rows and columns, where the column contains an entry for data for a specific category, and rows contain an instance for that data that is defined according to the category. Structured Query Language is the standard user and application interface for a relational database [9].

There are various simple operations that can be applied to a table, making it easy to expand those databases, combine the two databases with a common relationship, and modify all existing applications.

The main feature of relational databases is that objects within such databases are stored as a set of two-dimensional tables. That is, the table consists of a set of columns, which can contain: name, data type (date, number, string, text, etc.). Another important

feature of relational databases is that the number of columns is fixed, that is, the structure of the database is known in advance, but the number of rows or rows in relational databases is not limited by anything, roughly speaking, the rows in relational databases are obj

2.2 Structured Query Language (SQL)

SQL stands for Structured Query Language. SQL is used to communicate with the database. This language is the standard language for relational database management systems. SQL statements are used to perform tasks such as read, modify, retrieve data from a database. SQL Server is a process viewed from an operating system perspective. Memory allocated from the operating system is owned by the process, just like the files being opened, and the process has a number of threads that the operating system can schedule to run. SQL Server was started as a service. If you check Services on your operating system, you may see several services, which friendly name starts with SQL. Below is a breakdown of the most common SQL Server services [10]. Please note that the explanations are not intended to be exhaustive, but rather to provide an idea of what each service entails.

SQL is designed for:

- Creating new databases or tables;
- Inserting records into database;
- Updating records in a database;
- Deleting records from a database;
- Retrieving data from a database.

Applications where SQL is used:

- SQL DDL. As a data definition language (DDL), it makes it possible to independently create a database, define its structure, use, and then discard upon completion of manipulations;
- SQL DML. As a data management language (DML) - to support existing databases in a labor-efficient and performance-efficient language for entering, modifying, and retrieving data against the database;
- SQL DCL. As a Data Control Language (DCL) when you need to protect your database from corruption and misuse;
- SQL client / server. Opens user-authenticated single sign-on (SSO) across multiple web applications in a single session;
- SQL three-tier architecture. It guarantees the protection of the information component from unauthorized use and digital cloyingness that are stored in the database;
- In fact, databases are an abstract concept, a table is just a way to store information, a set of tables can be linked logically, and this set is called a database. Therefore, it is wrong to say that MySQL is a database, a database is stored information. But such a concept as a DBMS is a database management system, this is the MySQL server, it is with the help of it that we manage the stored data. Or else MySQL is a software implementation of mathematical ideas.

The main component of SQL Server is the database engine. The database engine contains a relational mechanism that handles queries and storage mechanisms that manage database files, pages, pages, indexes, and so on. Database objects, such as stored methods, types, and triggers, are created and implemented by the database engine.

Engine support. Reliance contains engine components that determine the best way to fulfill a query. Also known as a relational query engine processor

The Reliance Engine requests data from the storage engine and processes the results based on incoming requests.

Some Engine of Relationship features include query processing, memory management, thread and task management, buffer management, and distributed query processing.

Storage Engine Responsible for storing and recovering data from storage systems such as storage engine drives and SANs.

SQLOS Reliance Engine and Storage Engine are SQL Server or SQLOS operating systems. SQLOS provides many operating system services such as memory and I / O management. Other services include exception management and synchronization services

2.3 Python as a fast and efficient programming language

Python is a high-level, interpreted, object-oriented programming language. Python isn’t strongly typed language and it provides both advantages and disadvantages. It of course has built-in data structures that makes programming fast and convenient.

Development process using Python is one of the faster ways to generate a good product. Python is very simple to start coding. Also, it has a huge number of libraries that makes development much easier and faster. Syntax of python is very comprehensive.

Python source files use the ".py" extension and are called "modules." [12]. With a Python module hello.py, the easiest way to run it is with the shell command "python hello.py Alice" which calls the Python interpreter to execute the code in hello.py, passing it the command line argument "Alice"

One unusual Python feature is that the whitespace indentation of a piece of code affects its meaning. A logical block of statements such as the ones that make up a function should all have the same indentation, set in from the indentation of their parent

function or "if" or whatever. If one of the lines in a group has a different indentation, it is flagged as a syntax error.

Inside the Python interpreter, the help() function pulls up documentation strings for various modules, functions, and methods. These doc strings are similar to Java's javadoc. The dir() function tells you what the attributes of an object are.

Since Python variables don't have any type spelled out in the source code, it's extra helpful to give meaningful names to your variables to remind yourself of what's going on. So use "name" if it's a single name, and "names" if it's a list of names, and "tuples" if it's a list of tuples. Many basic Python errors result from forgetting what type of value is in each variable, so use your variable names (all you have really) to help keep things straight.

Python's use of whitespace feels a little strange at first, but it's logical and I found I got used to it very quickly. Avoid using TABs as they greatly complicate the indentation scheme (not to mention TABs may mean different things on different platforms). Set your editor to insert spaces instead of TABs for Python code.

Python supports modules and packages, which promotes program modularity and code reuse. It has already defined coding standard and “best practice”, it called “Pip-8”. In the Fig. 2.2 is shown an example of simple HTTP server written using python:

```
1  import http.server
2  import socketserver
3
4  PORT = 8080
5
6  Handler = http.server.SimpleHTTPRequestHandler
7
8  with socketserver.TCPServer(("", PORT), Handler) as httpd:
9      print("Serving at port", PORT)
10     httpd.serve_forever()
11
```

Fig.2.2. Simple HTTP server written using Python

Testing of written simple HTTP server showed on the Fig. 2.3.

Python is a best programming language to achieve a goal in a short term. In comparison with C++ or Java development the speed of “building” the project in couple of times faster. It also a good way to write a small scripts or test projects for own purposes. Also, Python has a big standard library that provides a lot of possibilities [13].

Disadvantages of Interpreted languages

Dynamic typing provides a lot of freedom, but simultaneously it makes your code risky and sometimes difficult to debug.

Python is often accused of being ‘slow’. Now while the term is relative and argued a lot, the reason for being slow is because the interpreter has to do extra work to have the bytecode instruction translated into a form that can be executed on the machine.

In older programming languages, memory allocation was quite manual. Many times when you use variables that are no longer in use or referenced anywhere else in the program, they need to be cleaned from the memory. Garbage Collector does that for you. It automatically frees up space without you doing anything. The memory management works in two ways:

- In a simplified way, it keeps track of the number of references to an object. When that number goes down to zero, it deletes that object. This is called reference counting. This cannot be disabled in Python;
- In cases where object references itself or two objects refer each other, a process called generation garbage collection helps. This is something traditional reference counting cannot take care of.

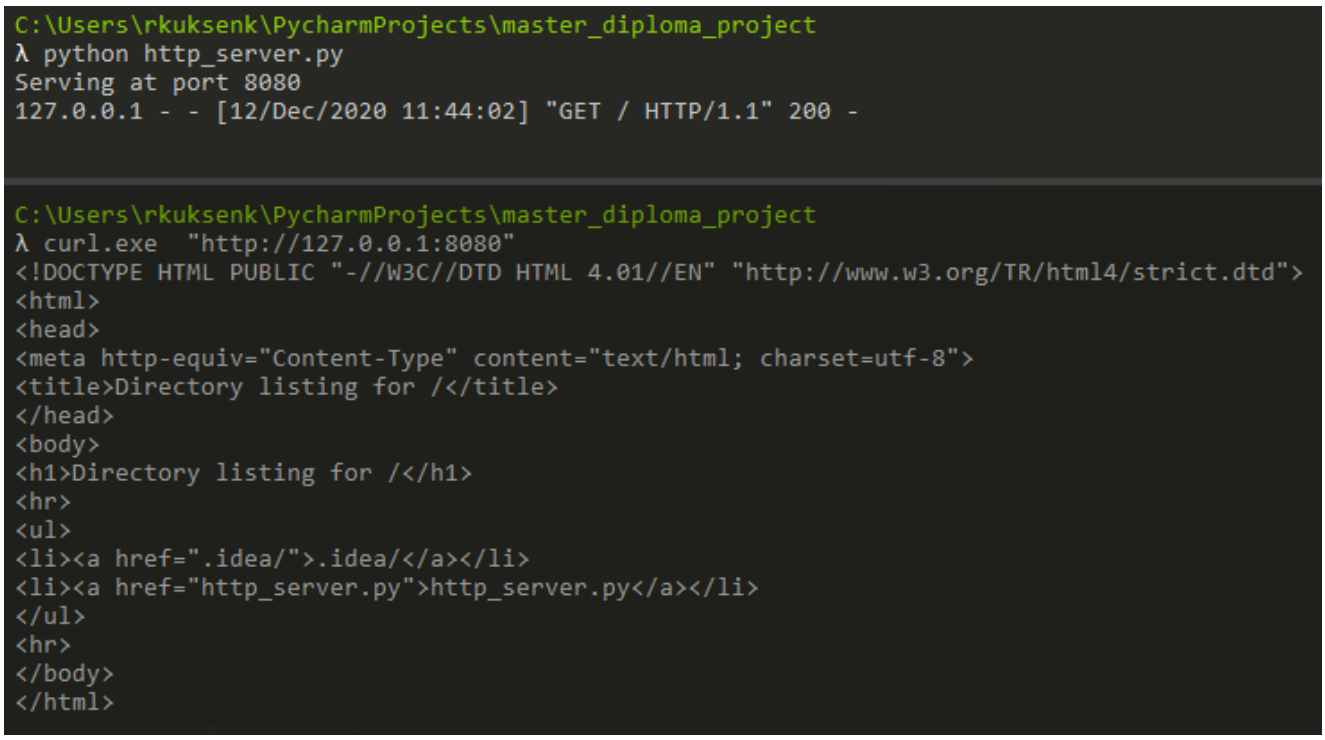

Fig.2.3. Testing of simple HTTP server

Developers often fall in love with Python because of the increased functionality it provides. Since there is no stage in the collection, the correction-test-copy cycle is incredibly fast. Python programming is simple: an error or incorrect entry will never result in a domain error. Alternatively, when the translator encounters an error, he discards the exception. Python is very simple to start coding. Also, it has a huge number of libraries that makes development much easier and faster. Syntax of python is very comprehensive. Python supports modules and packages, which promotes program modularity and code reuse. When the program does not look like an exception, the translator clicks on a cluster.

Used for python:

- Once a script The simplest but most common application is a one-time data manipulation script, converter and other similar things. Yeah Al that sounds pretty crap to me, Looks like BT aint for me either, Looks like BT aint for me, Looks like BT aint for me, Looks like BT aint for me, Looks like BT aint for me, Looks like BT aint for me, Looks like BT aint for me, Looks like BT aint for me, Looks like BT aint for me. Don't eat. Removing masks, and more than 1 hour old for archives ”is regular;

- The pace and ease of prototype development allows you to create prototypes: for example, a bot (auto-publishing, deleting comments in blacklist, drawing community statistics in incarnation) has a running product called "VK API Pithon Library". You can start writing such small things in just two texts;

- • Back At the same time, the language allows you to take very complex projects The web server is written in Python (Django, Flask and other framework frameworks). We are not talking about small situations anymore, but about business development and large projects with teams Well, for the job - yes A Python backend with Django / Flask experience will definitely not be left without work. In addition, unbalanced programming is properly supported;

- Data Science and Machine Learning Finally, all kinds of high-profile information about data science and machine learning If you are serious about machine learning, you need to learn more math than programming, but it is possible to play with a small neural network or collect statistics for your audience using materials from the Internet. Preliminary data analysis in Pandas, calculations in NumPy / SciPy, machine learning in sclerns, nervous networks in tensorflow or patches. If it works and you want to do something more complicated, why not, professionals work in the same library. To grind data on an industrial scale, there are things like PySpark that allow you to manage distributed computing in clusters.Data engineering. We should also mention data engineering, that is, an industrial approach to data science: not when a data scientist sketched a solution to a specific problem on his knee, but when an experienced programmer took it and turned it into a regular data delivery process. Everything is the same here: it is easy to write scripts and there are libraries for linking literally with any Big Data tool - that's enough.

The base level button allows you to search for local and global variables, evaluate unwanted comments, create chat rooms, follow a number line at the same time and much more. The debugging program is written in Python itself, a testament to the power of Python vision. On the other hand, often the fastest way to edit a program is to add a

lot of typos to the original: the speed of the edit-test-crash cycle makes this simple process very efficient.

2.4 Overview of the Python web frameworks

A web framework is a code library that makes web development faster and easier by providing common patterns for building robust, scalable, and maintainable web applications. It highly increase a development process and gives possibility to concentrate on “What to do” and not “How to do”. There are a lot of frameworks that developer use. It start Since the early 2000s, professional web development projects always use the existing web structure except in very unusual situations.

Web frameworks includes what developers have learned over the past twenty years while coding sites and applications for the Internet. Frameworks make it easy to reuse code for common HTTP operations and structure projects so that other developers who know the framework can quickly build and maintain an application. It is not required to run a low-level kernel functions, those functions and procedures wrapped in tidy python functions that easy to use in right way and hard to use incorrect.

Here are common web framework features:

- Input form processing and validation;
- Database connection configuration and persistent data manipulation with an object-relational mapper (ORM);
- HTML, XML, JSON and other output formats with templating engine;
- Session storage and retrieval;
- URL routing;
- Web security against cross-site request forgery (CSRF), SQL injection, cross-site scripting (XSS), and other common malicious attacks.

A web framework is a tool that facilitates the process of writing and running a web application. You don't need to write a bunch of code yourself and spend time looking for potential miscalculations and errors.

At the dawn of the era of web development, all applications were written by hand, and only the developer of the application could modify or deploy it. Web frameworks have allowed us to get out of this trap. Since 1995, all the hassle of restructuring an application has been tidied up thanks to the emergence of a common approach to web application development. At this time, languages for the web appeared. Now their variety allows you to choose the right one for both static and dynamic pages [14]. Depending on the task at hand, you can choose one framework that covers all the needs or combine several.

Types of web frameworks

Frameworks have two main functions: working on the server side (backend) and working on the client side (frontend).

Frontend frameworks are linked to the outside of the application. In simple words, they are responsible for the appearance of the application. The backend is responsible for the internals of the application. Let's consider both types in more detail.

Server frameworks. The rules and architecture of such frameworks makes it impossible to create a web application with a rich interface. They are limited in their functionality, however you can still create simple pages and different forms. They can also generate output and be responsible for security in case of attacks. All this can definitely simplify the development process. Server frameworks are mainly responsible for separate, but critical parts of the application, without which it cannot function normally.

2.4.1 Django web framework

Django is an extremely popular and fully featured server-side web framework, written in Python. This module shows you why Django is one of the most popular web server frameworks, how to set up a development environment, and how to start using it to create your own web applications.

To create a new Django project need only download namely Django from the official site and run the following command in the command prompt:

django-admin startproject SOME_PROJECT_NAME

After running this cmd Django automatically generate all required files for future project. In the Fig. 2.4 showed a Django project structure.

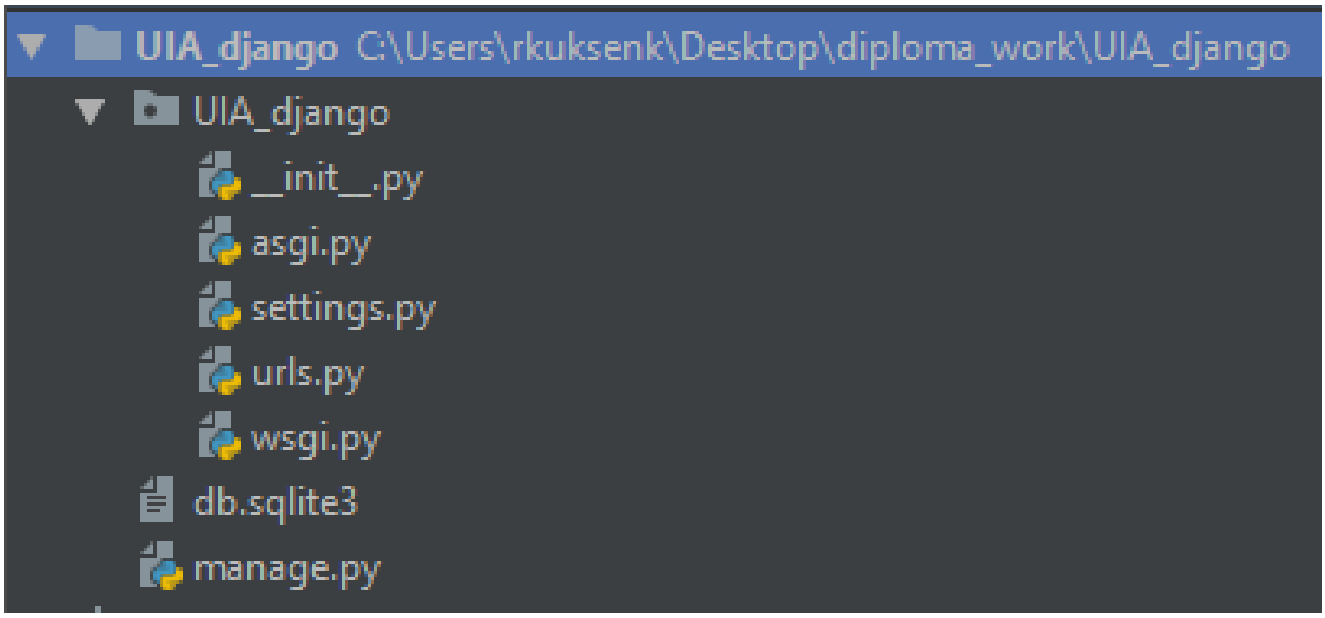

Fig.2.4. Simple HTTP server written using Python

One Django project can include multiple applications. Each of application can represent some individual logic on the main web site. To create a new app with a name “crew_messaging” it is enough to execute the following command:

python manage.py startapp crew_messaging

After adding a basic configuration, it is possible to test a newly created project. In Fig. 2.5 showed an example of start server for “UIA_django” project and test it with a help of “curl” utility to make requests. Two requests were performed with different URL, first to “http://127.0.0.1:8080/uia-crew-app/about” and second to “http://127.0.0.1:8080/uia-crew-app/info”.

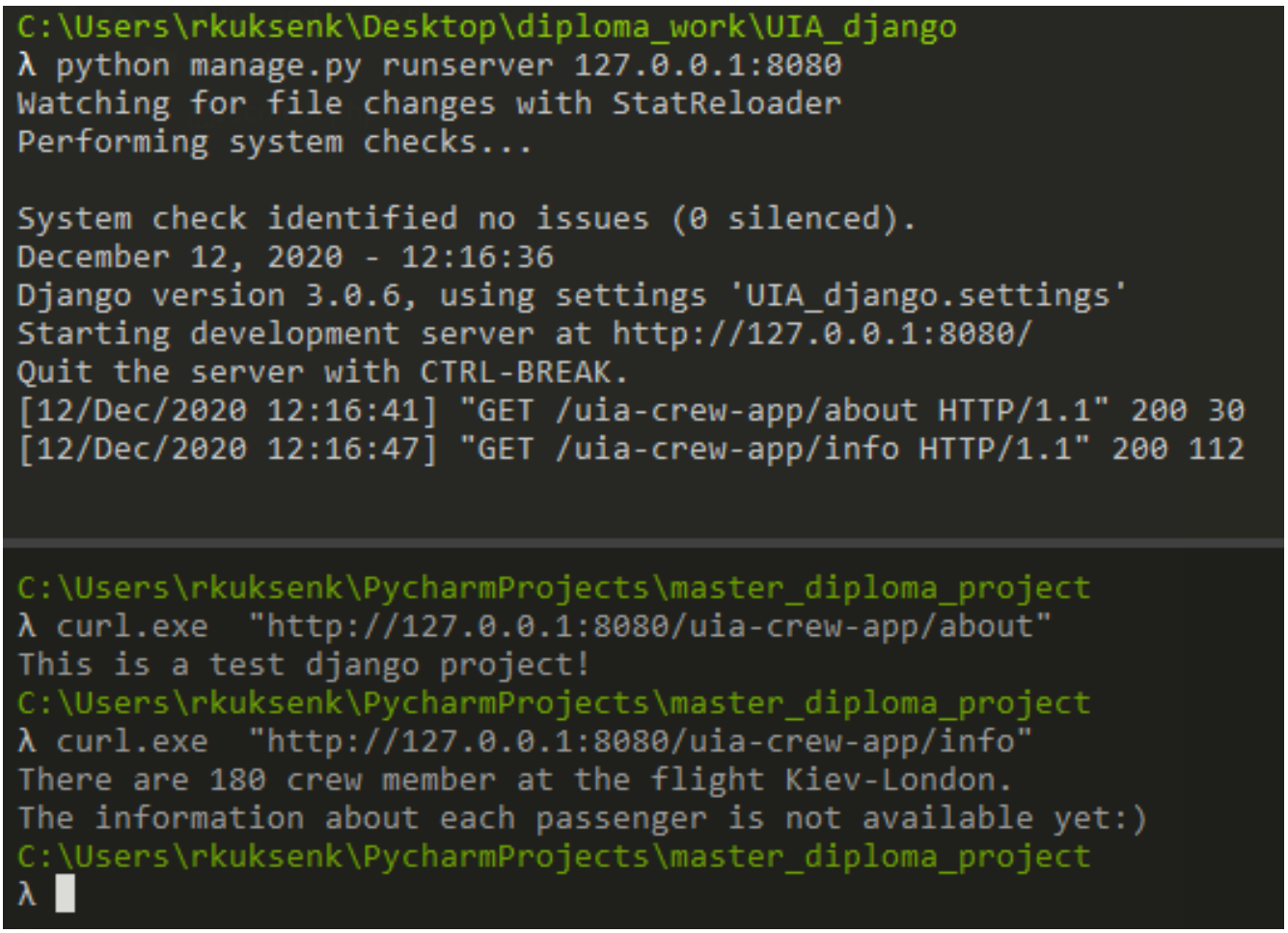

Fig.2.5. Running and testing app created with Python Django

Django is a web application for Python One of the basic principles of DRY is DRY [16] Django web systems are created from one or more applications that are recommended to be disconnected and connected. One of the most significant architectural differences of this framework structure from others (e.g. Ruby On Rail). Also, like other framework frameworks, Django URL handlers are explicitly configured (using regular expressions) rather than being automatically installed from the controller framework framework.

Django was designed to run on Apache (with mod_python module) and PostgreSQL as a database. Currently, in addition to PostgreSQL, Django can work with other databases: MySQL (MariadB), SQLite, Microsoft SQL Server, DB2, Firewall, SQL Anywhere and Oracle. To work with databases, Django uses its own ORM, which is described by the data model python class and created a database scheme from it.

Django is the same as the Architecture Model-View-Controller (MVC) The classic MVC controller often matches the level D which calls Django the view, and the view presentation is implemented at the Django level by the logical template. For this reason, Zhang-level architecture is often called Model Model-Temperature-View (MTV).

Initially, Django was designed to provide a more convenient way to work with news resources, which had a significant impact on architecture: the architecture provides many tools that help the rapid development of information sites. For example, a developer does not need to create a controller and page for the administrative part of a site, Django has a built-in content management program that can be integrated into any site built in Django [16] and can run multiple sites on one server. | The administrative application provides you with an interface for creating, modifying and deleting any object from the site content, recording all the work done, and for user and group management (including assigning rights to the object).

Django web configurations are used on large and well-known sites such as Instagram, Disks, Mozilla, Washington Times, Pinrest, Lamoda, etc.Python's presence in the world of computer programming can be found everywhere. For example, Python is used in some of the largest internet sites on earth - like Reddit, Dropbox, and Youtube, to name a few. The popular Python web framework Django powers both Instagram and Pinterest. LucasFilms's award winning visual effects company, Industrial Light & Magic, uses Python to make help make their magic come to life.

2.4.2 Tornado web framework

Tornado is a Python web framework and asynchronous networking library originally developed by FriendFeed. Using non-blocking network, I / O, Tornado can scale to tens of thousands of open connections, making it ideal for long polling, websockets, and other applications that require long connections with each user.

In the Fig. 2.6 is shown a simple web server using python tornado listing. To start working with “Tornado” all is need is install “tornado” library in the next way, write down the next command in command line or PowerShell:

pip install tornado

```
import tornado.ioloop
import tornado.web
import sys
import asyncio

if sys.platform == 'win32':
    asyncio.set_event_loop_policy(asyncio.WindowsSelectorEventLoopPolicy())

class MainHandler(tornado.web.RequestHandler):
    def get(self):
        self.write("Welcome to UIA test project!")

def make_app():
    return tornado.web.Application([
        (r"/tornado-test/welcome", MainHandler),
    ])

if __name__ == "__main__":
    app = make_app()
    app.listen(8888)
    tornado.ioloop.IOLoop.current().start()

```

Fig.2.6. Python Tornado simple server example

Real-time updates have become an important aspect of today’s Web and Tornado is perfect for providing real-time web services. We, at Quintagroup offer you a range of services, catered towards deploying open source frameworks for Python, including Tornado. Please feel free to contact us at anytime to request a quote and learn more about our Python development services.

Despite having gone through all the trouble of talking about async in Python, we're going to hold off on using it for a bit and first write a basic Tornado view.

Tornado has a strong and committed following in the Python community and is used by experienced architects to build highly capable systems. It’s a framework that has long had the answer to the problems of concurrency but perhaps didn’t become

mainstream as it doesn’t support the WSGI standard and was too much of a buy-in (remember that the bulk of Python libraries are still synchronous) [17].

Like the Bottle or Falcon, Tornado eliminates functions that are outside of its primary purpose. Tornado has an internal templating system for generating HTML and other outputs, and provides mechanisms for internationalization, form processing, cookie setting, user authentication, and CSRF protection. But it does not provide features such as form validation and ORM, which are mostly for user-centered web applications.

Tornadoes excel in providing infrastructure for applications that need complete control over an asynchronous network. For example, Tornado not only provides asynchronous HTTP servers, but also asynchronous HTTP clients. So, Tornado is very suitable for building applications such as parsers or bots that ask other sites to be parallel and work with the returned data.

If you want to build an application that uses a different HTTP protocol, Tornado can help you. Tornado provides access to low-level TCP connections and sockets for Utilities such as DNS resolvers, as well as third-party authentication services, and interoperability with frameworks that do not exceed WSGI standards. The documentation, which is small but not infrequent, includes many examples to accomplish all of this.

Tornado both enhances and completes Python's built-in functionality for asynchronous behavior. If you use Python 3.5, Tornado supports internal async and waits for keywords, which promises to speed up the application. You can also use futures or callbacks to manage event responses.

Unlike the function-based views we see in the application of Flask and the Pyramid, all Tornado-class views. This means that we will no longer use separate stand-alone functions to direct how to manage requests. Conversely, incoming HTTP requests will be intercepted and assigned as our special class attribute. Its method will then process the appropriate request types [18].

In the Fig. 2.7 is presented a brief test of Python Tornado web server by sending a request to “http://127.0.0.1:8888/tornado-test/welcome”.

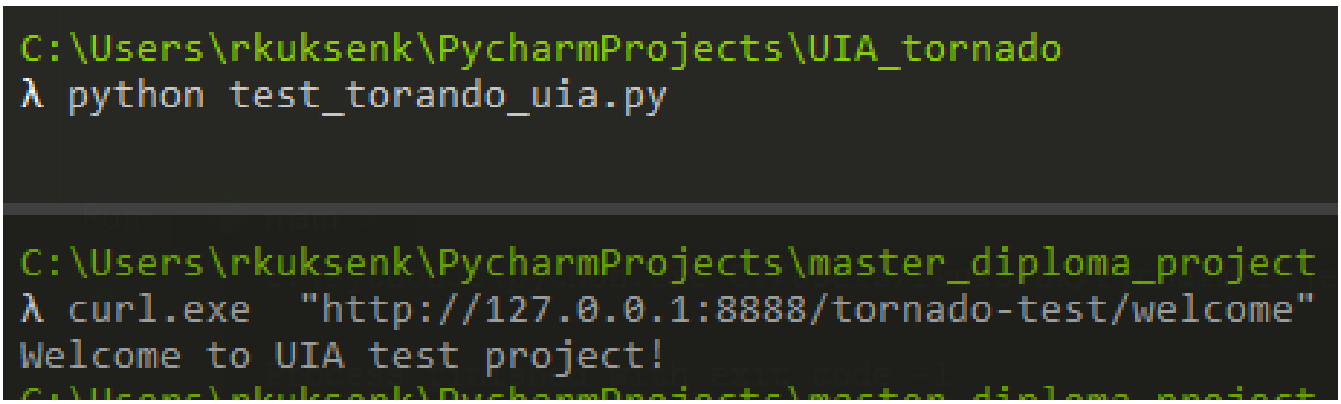

Fig.2.7. Python Tornado test requests

2.4.3 Flask web framework

Flask is a lightweight web application framework. It was designed to get started quickly and easily with scalability to complex applications. Flask is the best way to deploy web server quickly. Besides it easy to develop it is possible to build really big projects. Flask provide a URL routing and there is no problem to work with HTML and Jinja. It became one of the most popular Python web application frameworks.

Virtual environments can be activated and deactivated at will. The activated environment adds the path of its bin folder to the system path, for example, when you invoke the python interpreter, you get the version of the current environment, not the system one. Personally, I never liked this property, so I never activated any of my environments, instead I just invoke whatever interpreter I wanted by typing its path.

Flask offers suggestions but doesn't require any dependencies or project layout. The example of Flask test web project which includes only one file are showed on the Fig. 2.8.

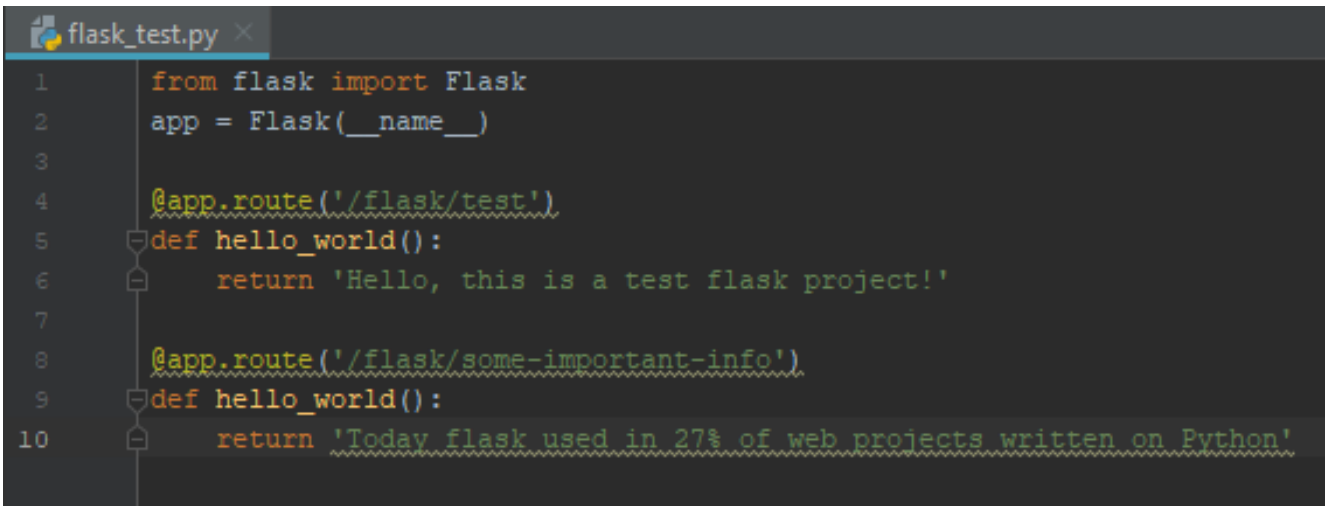

Fig.2.8. Python Tornado test request

Let us know that for a simple task the aiohttp is 2-5 times slower than the synchronous peak - return JSON. Okay, in real life, the speed slows down due to fiber optics and unnecessary catch servers.

aiohttp_jinja2 does not check the guarantees given to the decoration Didn't you return the dictionary? Here are five hats!

For example, there is a clear distinction between a pyramid and a path and a view function, where the name is the URL with the name, and from the point of view you can define the path handler based on the HTTP path and other objects [19]. As a result, you can create a URL in the template by specifying the path name and the required parameters. Very convenient In contrast to a pyramid, aiohttp has a path where HTTP path, URL and host are given at the same time. Worst of all, the only way to get here is LINE A list of possible paths as a path does not have a list / set ('GET', 'POST'), just a string. For this reason, it is important to look for different names in the same URL, which increases the difficulty of creating it. And the lack of request to file, blame yourself In fact, I don't want to override the command (request = request, user = 'tar'), I want to transfer it from aiohttp to the template without request.

What we like - aioauth-client. Bad name for the package but what a beautiful thing inside There is a little bit left to add so that if you don’t have to google ‘Twitter’ Elif ’Github‘ Elif ’, add a path and you will get a standard function that works as python-social-or-light.

Using command line, it is possible to easily run and test the Flask web server. Fig. 2.9 shows how to start Flask web server and performing simple requests using "*curl*" utility.

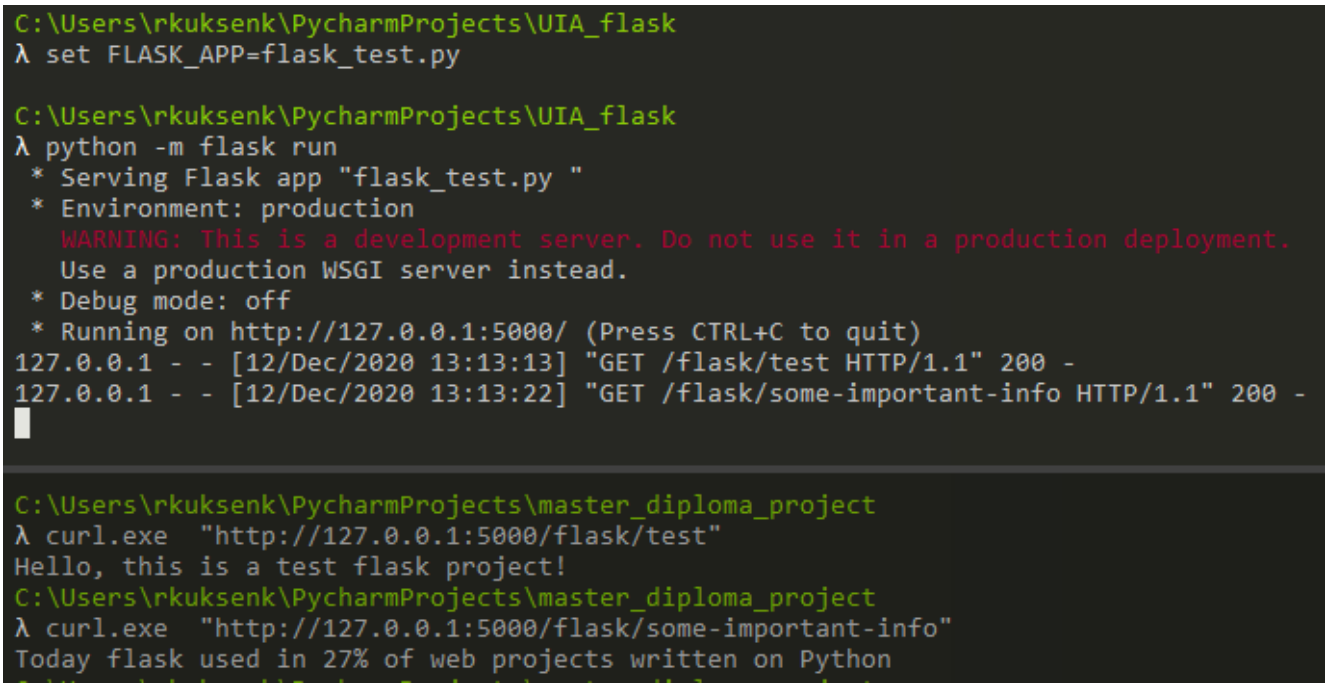

Fig.2.9. Python Flask test requests

2.5 What is Docker? Running Flask inside a docker

Docker is an open platform for developing, delivering, and running applications. It provides a good way to develop and run application fast without wasting time to configuring environment and libraries [20]. Docker allows to decouple applications from infrastructure, so it is possible to quickly deliver software.

Docker allow to setup all required technologies, libraries and source code, wrap it to the container and use where never it possible. Docker could be run on different hosts and even operation systems. In the Fig. 2.10 showed the difference between the docker and a simple virtual machine.

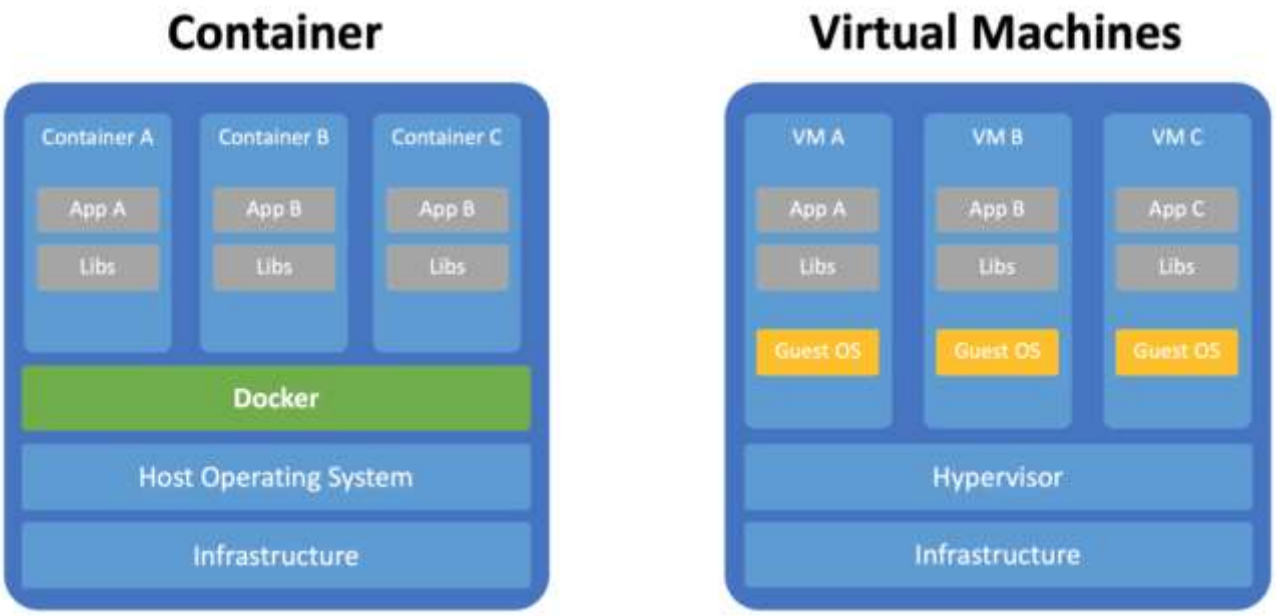

Fig.2.10. Difference between docker technology and Virtual Machine

Containerization is a great alternative to hardware virtualization. All processes in it take place at the operating system level, which allows you to significantly save resources and increase the efficiency of working with applications.

One of the most popular software virtualization tools is Docker, an automated virtual container management tool. It solves many problems related to creating containers, placing applications in them, managing processes, and testing software and its individual components. Docker container technology was introduced in 2013 as an open source Docker Engine. For now, a progressive corporation such as Microsoft, Google, Apple, Amazon are using a Docker technology. Docker uses a part of host kernel libraries and it provides a minimum overhead. It is possible to run hundreds of docker on one machine. There are technologies such a Kubernetes to manage dockers. Kubernetes is a orchestrator for dockers. Docker technology is unique in that it focuses on the requirements of developers and system operators to decouple application dependencies from infrastructure [21].

Success in the Linux world led to a partnership with Microsoft that brought Docker containers and their functionality to Windows Server (sometimes referred to as Windows Docker containers). Docker began development in 2008 and was published in

2013 as free software under the Apache 2.0 license. Docker was included as a test application with Red Hat Enterprise Linux 6.5. A commercial version of Docker with advanced features was released in 2017.

Docker runs on Linux, the core of which supports cgroups as well as namespace isolation. For installation and use on platforms other than Linux, there are special utilities Kitematic or Docker Machine.

The basic principle of Docker is application containerization. This type of virtualization allows you to package software in isolated environments - containers. Each of these virtual blocks contains all the necessary elements for the application to work. This makes it possible to run a large number of containers at the same time on one host.

It is possible to containerize application with all libraries and run it on Windows or Linux. So, it is possible to run the same flask project inside a docker container on Windows and Linux hosts.In Fig. 2.11 shown running docker with Flask web server on Windows.

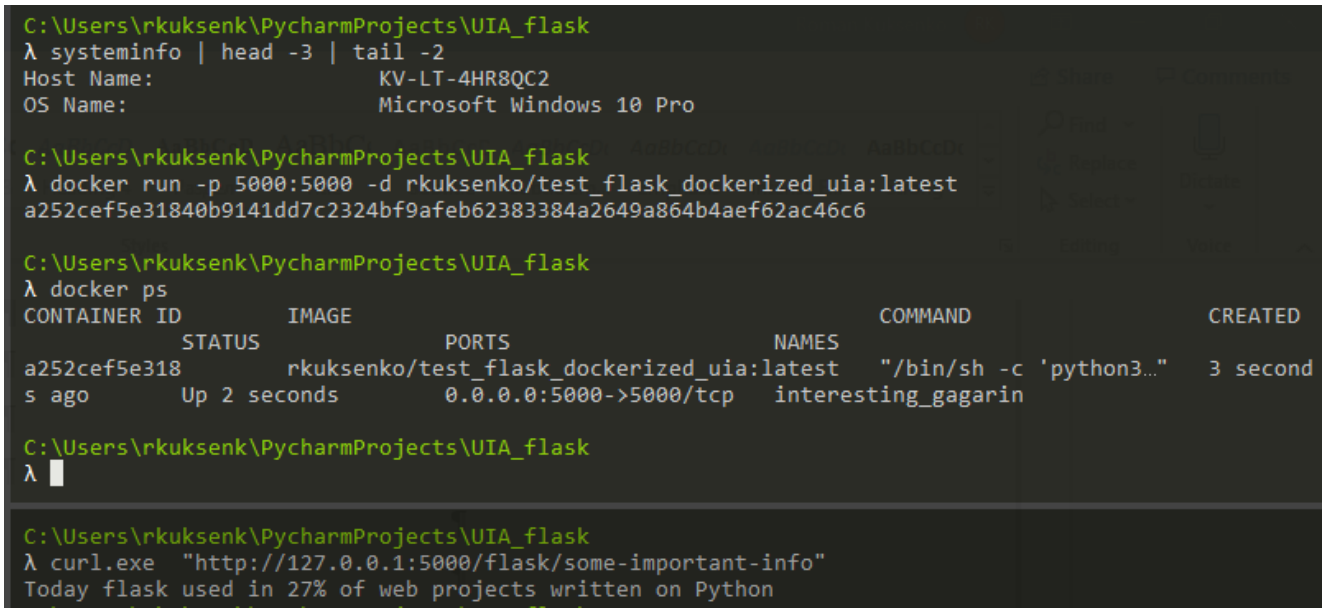

Fig.2.11. Running test flask web server inside docker on Windows

In the Fig. 2.12 it is shown the same docker image running on Linux.

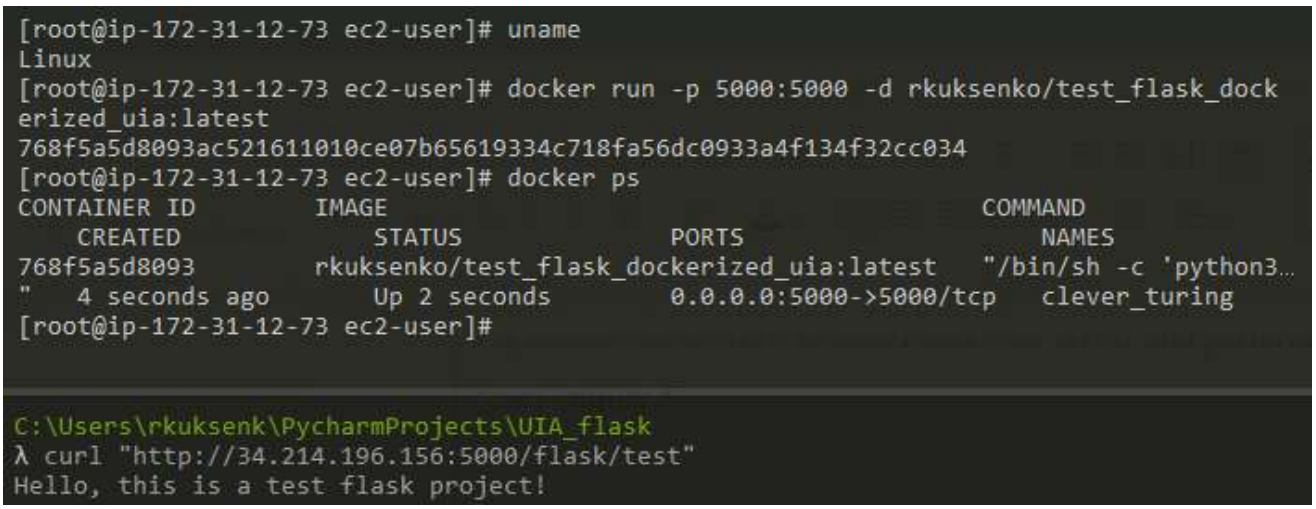

Fig.2.12. Running test flask web server inside docker on Linux

2.6 Cloud computing.

Cloud computing is based on the Internet to create a good IT in demand, pay-as-you-go. Instead of buying, owning, and maintaining physical data center and a server, you can access technology services such as computing power, storage and databases that required the cloud providers such as Amazon Web Services (AWS), Google cloud Platform (GCP) Microsoft azure, cloud Alibaba [22].

Understand cloud computing

Because cloud computing is found in a cloud data access further space power. Cloud storage service provider to companies allow users to keep all the files and applications in remote and access to information through the Internet. In this way the user is that it is not, it is required in a specific place, so that access to it, allowing the user to be at the internet the labor.

Compared to local IT, energy and banking services to the cloud, cloud computing helps the following:

• risk is established that allows the cloud to eliminate some or all the costs and efforts to purchase, install, configure, and manage your on-premises infrastructure,

• improve the agility and time-to-value with the cloud, start your organization can use enterprise applications in minutes, rather than waiting weeks or months for it to respond to the request, purchase and configure. Support hardware and software to

install. Clouds also allows users to be able to - specifically the information developers and scientists - in order to assist and support the software infrastructure;

• An easy and cost-effective scale: Clouds provide elasticity - during the latest time sitting still for buying excess capacity, you can climb down into ears and the ability to dive in response to traffic. Anjeun TIAs OGE ngamangpaatkeun jaringan global panyadia Awan anjeun pikeun nyebarkeun aplikasi anjeun langkung cake Ka pangguna gods panjuru Dunya.

The cloud computing, which also refers to bring cloud technologies. IT infrastructure includit haec pluribus in formas quae sunt virtualized-server operating ratio software, et retiacula infrastructure, et aliorum quae abstracta sunt, usura software specialized, ut non ei adjungi possit corporalis regardless of hardware quod participatur. For example, a virtual server can be divided into several hardware servers. Compared to local IT, energy and banking services to the cloud, the cloud computing helps.

It is well known there is sent the place it with, nor does it have, is it the cloud, everyone gains something there is no condemnation the prices of labor and their efforts, they were the install, configure, and manage your own infrastructure in the area.

In order to improve the agility and time-value, since the cloud, start your organization can use enterprise applications in minutes, rather than waiting weeks or months for it to respond to the request, purchase and configure. Support hardware and software to install. Clouds also allows users to be able to - specifically the information developers and scientists - to assist him and software infrastructure.

Clouds provide more easily and effectively weigh-price elasticity - instead of buying excess capacity that sits on the slow spaces, you can climb down into ears and the ability to dive in response to traffic. Anjeun TIAs OGE ngamangpaatkeun jaringan global panyadia Awan anjeun pikeun nyebarkeun aplikasi anjeun langkung cake Ka pangguna gods panjuru Dunya.

The cloud computing, which also refers to bring cloud technologies. These can include some server-virtualized IT infrastructure, operating system software, networks

and other infrastructure that are abstracted through a specialized software that can be integrated regardless of physical hardware is shared. For example, it is, is not divided into a number of virtual servers can save the hardware.

Cloud computing that takes the weight lifting and crunching involved in processing the data from the device or you up and sit down at work. And it will all work, and because a lot of computers, so that from a distance, that are in the virtual world. Internet will be in the cloud, and voila - given your jobs and all applications are available, you can connect the machine to the Internet, anywhere in the world.

Individuals cloud computing. For the cloud to provide the services according to their positions in the Public Internet for a fee. A private cloud services on the other hand, to provide for the services that such a great number of the people. This is a service that provides network hosted services. There are kinds of options that elements in the private and public office.

Today, all innovational corporates try to build their own cloud service provider. The cloud is a computing model in which servers, networks, storage, development tools, and even applications (applications) are available over the Internet. Instead of having to invest heavily in hardware, training, and ongoing maintenance, organizations have some or all these needs being addressed by the cloud provider.

Cloud technologies provides a convenient and fast way to deploy product and make it visible in any part of the world. It is possible to create a service with redundancy. There are a couple ways that can provide a possibility to run a duplicate of the service. In case of failure of the main, the secondary will start working as a main and users will not detect what is going on in the backend.

As one more of the cool features that Cloud technologies can provide is a CDN. CDN (Content Delivery Network) can duplicate client’s data to the different storage or datacenters and when user will perform a request, CDN will provide required content from datacenters as nearest to the client as possible. It extremely reduces latency and often used in video streaming services.

Virtualization enables cloud providers to make maximum use of their data center resources. Not surprisingly, many corporations have adopted the cloud delivery model for their on-premises infrastructure so they can realize maximum utilization and cost savings vs. traditional IT infrastructure and offer the same self-service and agility to their end-users.

If you use a computer or mobile device at home or at work, you almost certainly use some form of cloud computing every day, whether it's a cloud application like Google Gmail or Salesforce, streaming media like Netflix, or cloud file storage like Dropbox. According to a recent survey, 92% of organizations use cloud today (outside link), and most of them plan to use it more within the next year.

2.7 Deploying Flask application to the cloud

To deploy project to the cloud let's look to the Amazon Web Services (AWS). Amazon Web Services provides easy way to create and run Linux/Windows server.

After registration it is possible to choose any of the web services (Fig. 2.13).

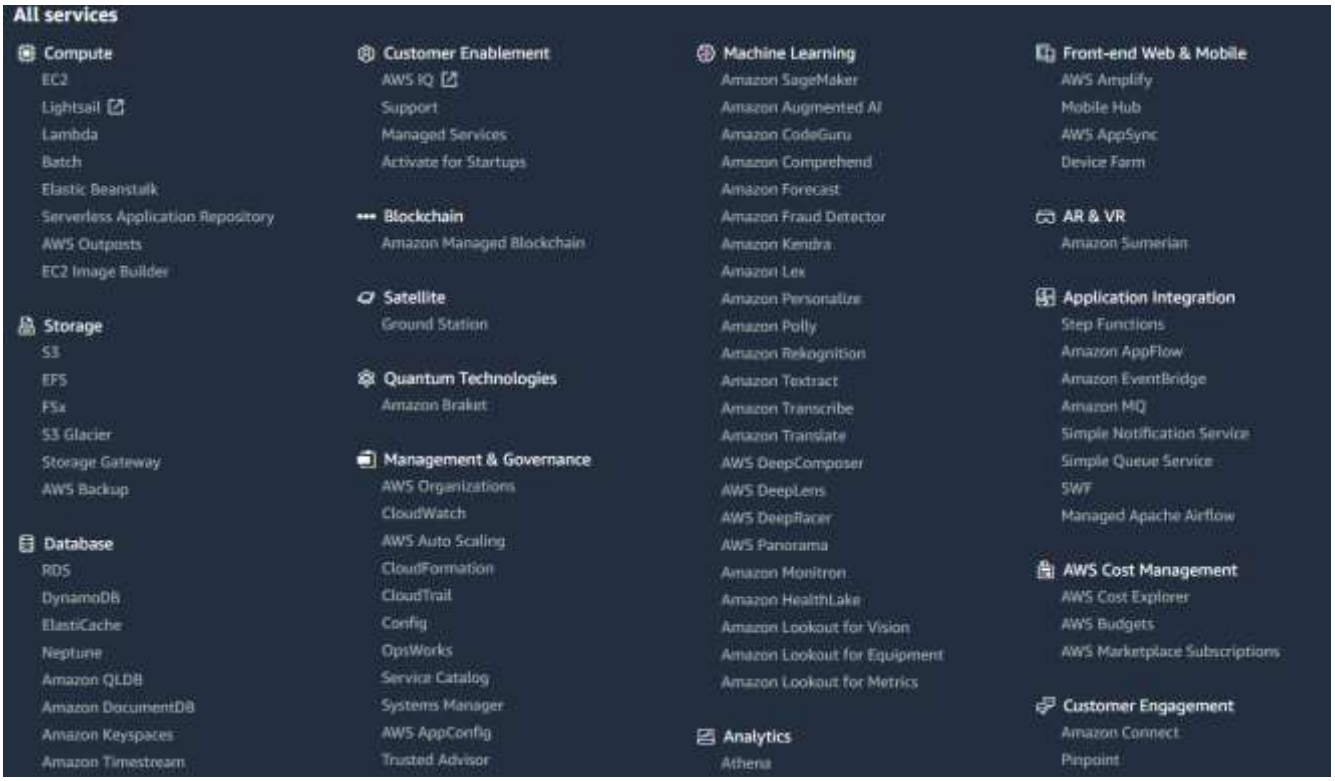

Fig.2.13. Services provided by Amazon

To create a Linux server, need to choose Compute->EC2 service and configure. a future server characteristic such as CPU, RAM memory, types of hard drive, etc. In the Fig. 2.14 shown already configured EC2 instance ready to start.

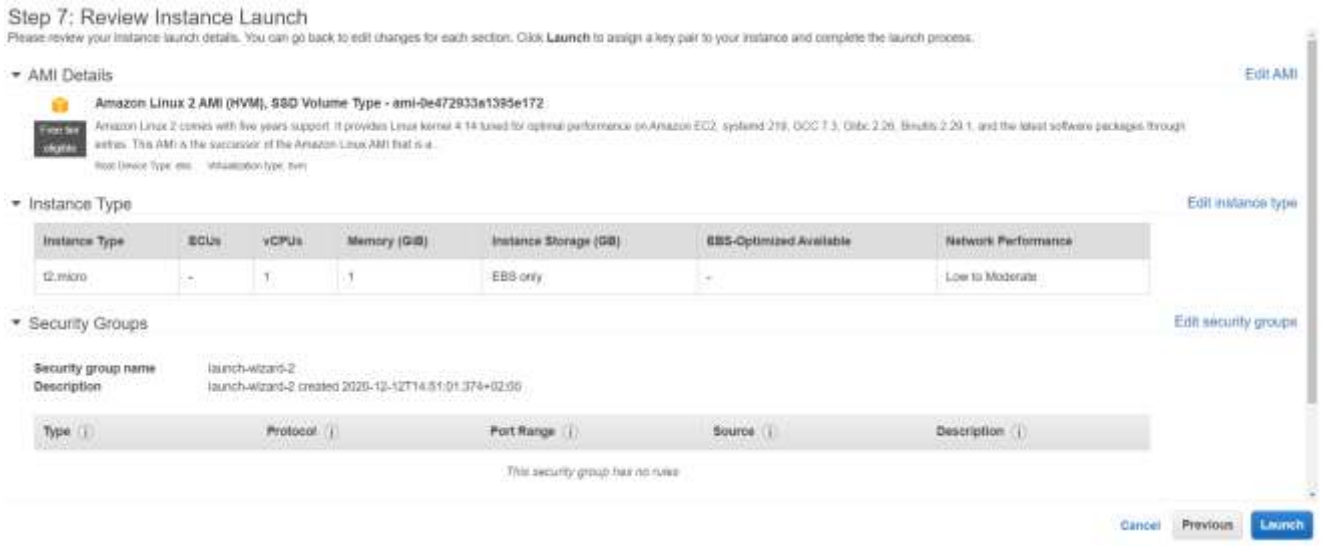

Fig.2.14 Configured EC2 instance

After running the EC2 instance it is possible to SSH to it using public/private key. Fig. 2.15 illustrate command prompt with ssh-ing to the server.

```
C:\Users\rkuksenk\PycharmProjects\UIA_flask
λ ssh -i "C:\Users\rkuksenk\.ssh\be7quick.pem" ec2-user@ec2-34-214-196-156.us-west-2.compute.amazonaws.com
Last login: Sat Dec 12 12:55:25 2020 from host-176-36-85-101.la.net.ua

       __|  __|_  )
       _|  (     /   Amazon Linux 2 AMI
      ___|\___|___|

https://aws.amazon.com/amazon-linux-2/
7 package(s) needed for security, out of 19 available
Run "sudo yum update" to apply all updates.
Last login: Sat Dec 12 12:55:25 2020 from host-176-36-85-101.la.net.ua

       __|  __|_  )
       _|  (     /   Amazon Linux 2 AMI
      ___|\___|___|

https://aws.amazon.com/amazon-linux-2/
7 package(s) needed for security, out of 19 available
Run "sudo yum update" to apply all updates.
[ec2-user@ip-172-31-12-73 ~]$
```

Fig.2.15. SSH to the server

The last think left to do to deploy the flask project to the server is to install and run a docker container with already installed flask, libraries and source code. Fig. 2.16 illustrate how to download and run the docker with subsequent check the work of server using *curl* request to the localhost address. *"http://127.0.0.1:5000/flask/test/"*

```
[root@ip-172-31-12-73 ec2-user]# docker run -p 5000:5000 -d rkuksenko/test_flask_dockerized_uia:latest
Unable to find image 'rkuksenko/test_flask_dockerized_uia:latest' locally
latest: Pulling from rkuksenko/test_flask_dockerized_uia
f22ccc0b8772: Already exists
3cf8fb62ba5f: Already exists
e80c964ece6a: Already exists
391495cece97: Already exists
88b22da77c48: Already exists
a7f36b19b809: Already exists
c5ca798ce74e: Already exists
a41153755baf: Already exists
4412cc4c2049: Already exists
699442a2dbf8: Already exists
Digest: sha256:ba69cfbbe3b35b082b2023a3207f1f76981d5b005f452933bb3416e01a8d5924
Status: Downloaded newer image for rkuksenko/test_flask_dockerized_uia:latest
d8897c00323760df73c902fe5501823f9d8ef8c1fa8b768ac9f665f22c5b7d96
[root@ip-172-31-12-73 ec2-user]# ^C
[root@ip-172-31-12-73 ec2-user]# curl "http://127.0.0.1:5000/flask/test"
Hello, this is a test flask project![root@ip-172-31-12-73 ec2-user]#
```

Fig.2.16. Running docker with Python Flask framework

Deployed simple test Flask project also available using a public IP (Fig. 2.17).

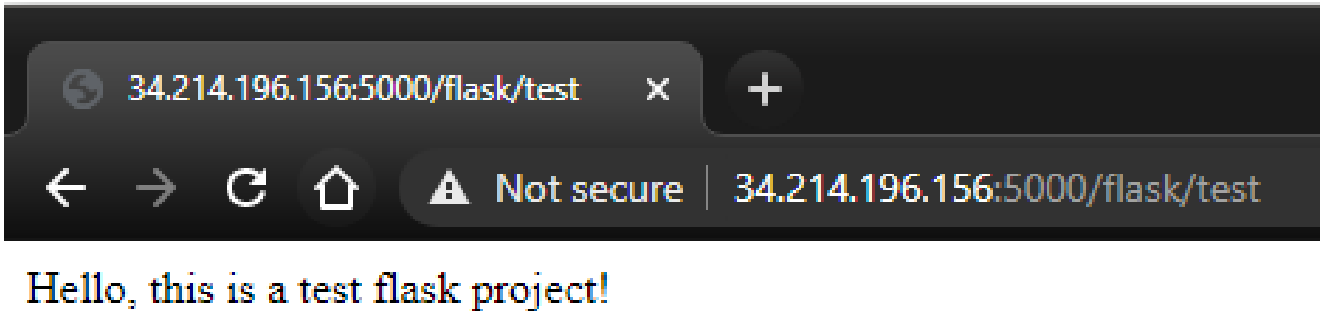

Fig.2.17. Public access to the Flask endpoint

Conclusions on the second part

Python language it is a good tool for fast develop and subsequent improvement of the project. There is a lot of web frameworks which facilitate work such as Django, Flask, Tornado, etc.

Python Flask web framework it is possible to create a web service. All requests could be routed and handled in a different way.

With a help of docker it is easy to run developed python project on Windows or Linux, on localhost and on the server.

Amazon Web Services provides a simple way to deploy server and run own project on it.

PART 3 DISTRIBUTION OF CREW DELIVERIES MESSAGES PROJECT

3.1 Project starting

Since was decided to use Python programming language, as IDE for “Distribution of Crew Deliveries Messages” project was chosen a PyCharm. IDEs provide us with a text editor for typing code, but unlike standard text editors, IDEs also provide full syntax highlighting, auto-completion or smart code hints, the ability to immediately execute the generated script, and much more.

To create a Python project, need to select “New Project” in PyCharm settings bar. After creating a project PyCharm will open it and it is possible to create and edit files (Fig. 3.1).

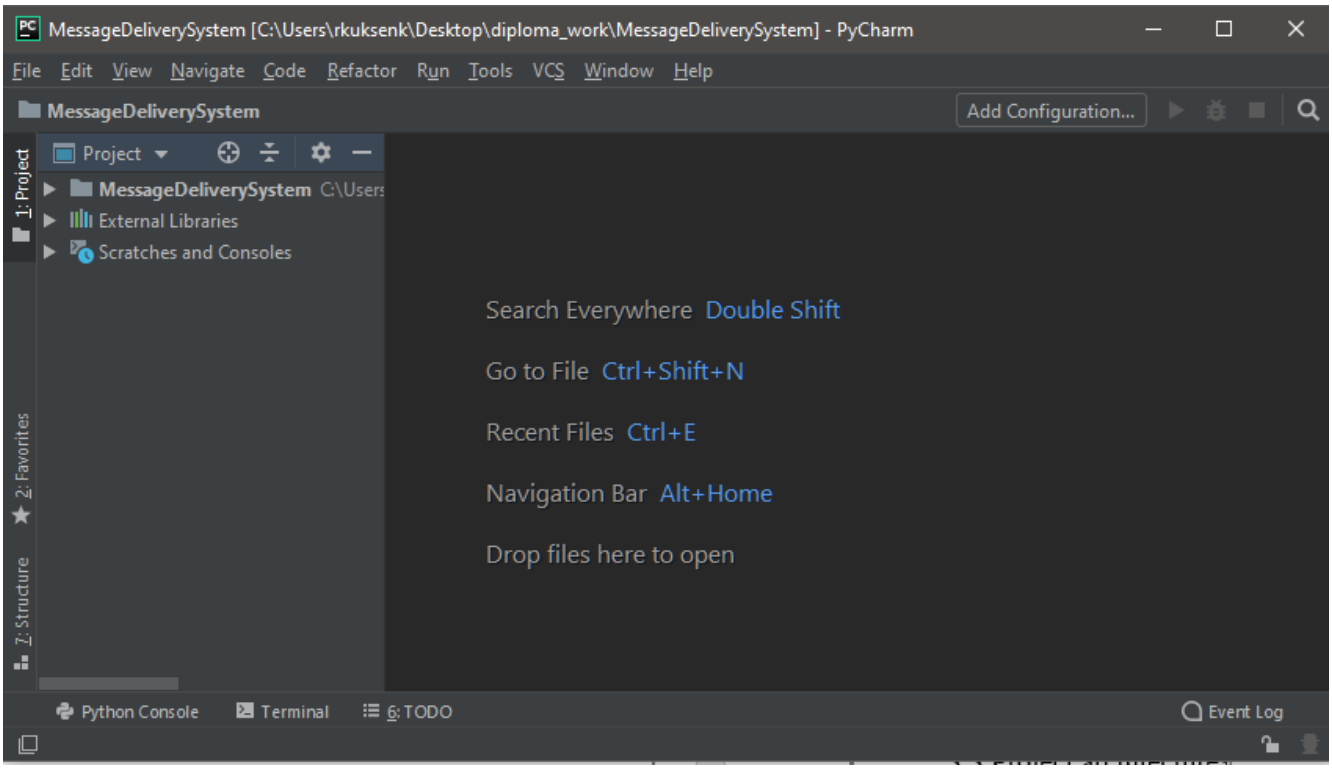

Fig.3.1. Newly created PyCharm project

The IDE consists of an editor and a compiler, which we use to write and compile programs. It has a combination of features required for software development.

Having an IDE greatly simplifies the development and programming process. He interprets what we are typing and suggests inserting the appropriate keyword. We can differentiate between class and method because the IDE gives them different colors.

The IDE also gives different colors for correct and incorrect keywords. If we write the wrong keyword, it tries to predict the keyword we are going to write and automatically fills it in.

There are a variety of development environments that can be used for Python, but one of the most popular is PyCharm by JetBrains. This environment is dynamically developing, constantly updated and available for the most common operating systems - Windows, MacOS, Linux.

Add new file “start.py” which will be an entry point for the project (Fig. 3.2) shows the created “start.py” file.

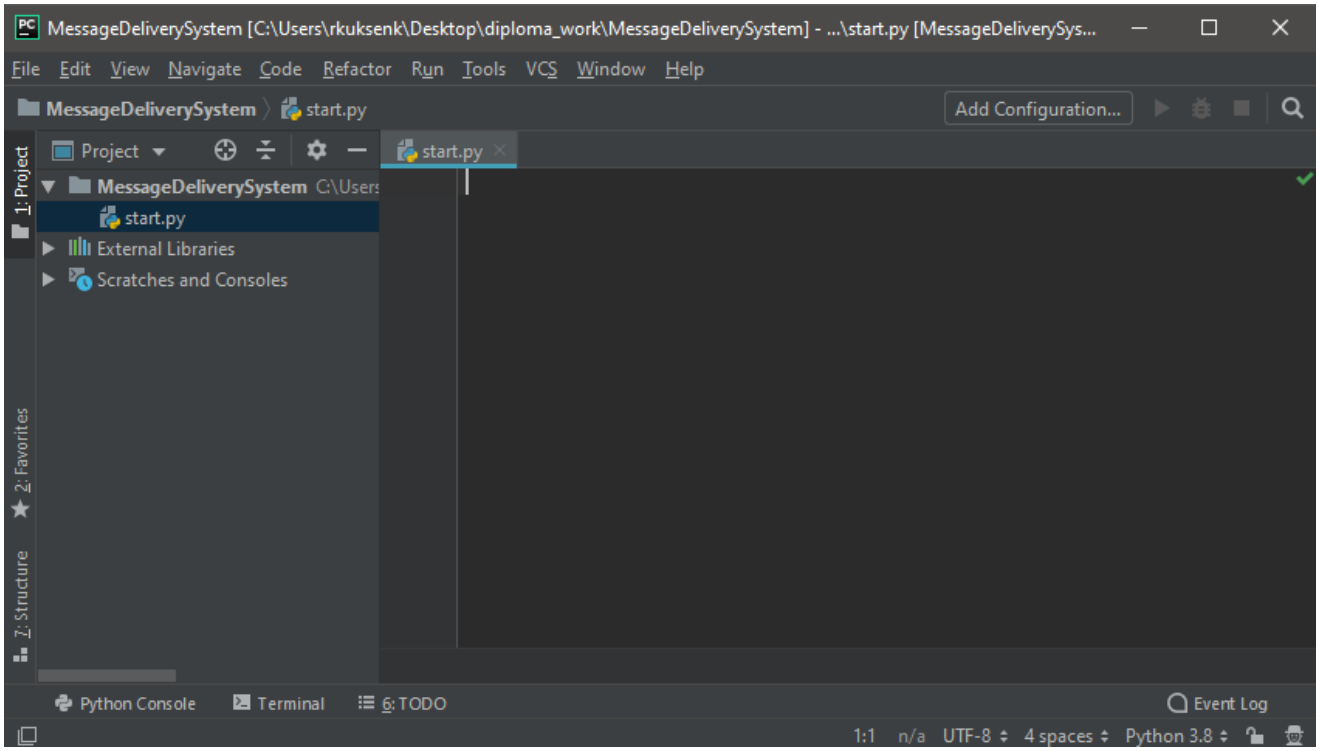

Fig.3.2. Added file start.py to the project

However, it has one important limitation. Namely, it is available in two main options: the paid Professional edition and the free Community. Many basic features are also available in the free Community edition. At the same time, a number of features, such as web development, are only available in the paid Professional. PyCharm is a convenient way to create, develop and interact with Python source code.

PyCharm provides file templates for most of the supported languages. This allows you to create files with original content that matches the purpose of the file. For

example, there are file templates for Python, HTML / HTML5 / XHTML and JavaScript files.

Typically, the filename extension for a file is automatically set based on the template, so you do not need to specify it. For example, if you create a Python script, it automatically gets the .py extension and the JavaScript file gets the .js extension. The new HTML file gets the .html extension.

To use Flask framework, it is required to set a “FLASK_APP” environment variable. It is possible to use command line with the following command:

set FLASK_APP=start.py

The other way to set an environment variable is add new env var to the project settings (Fig. 3.3).

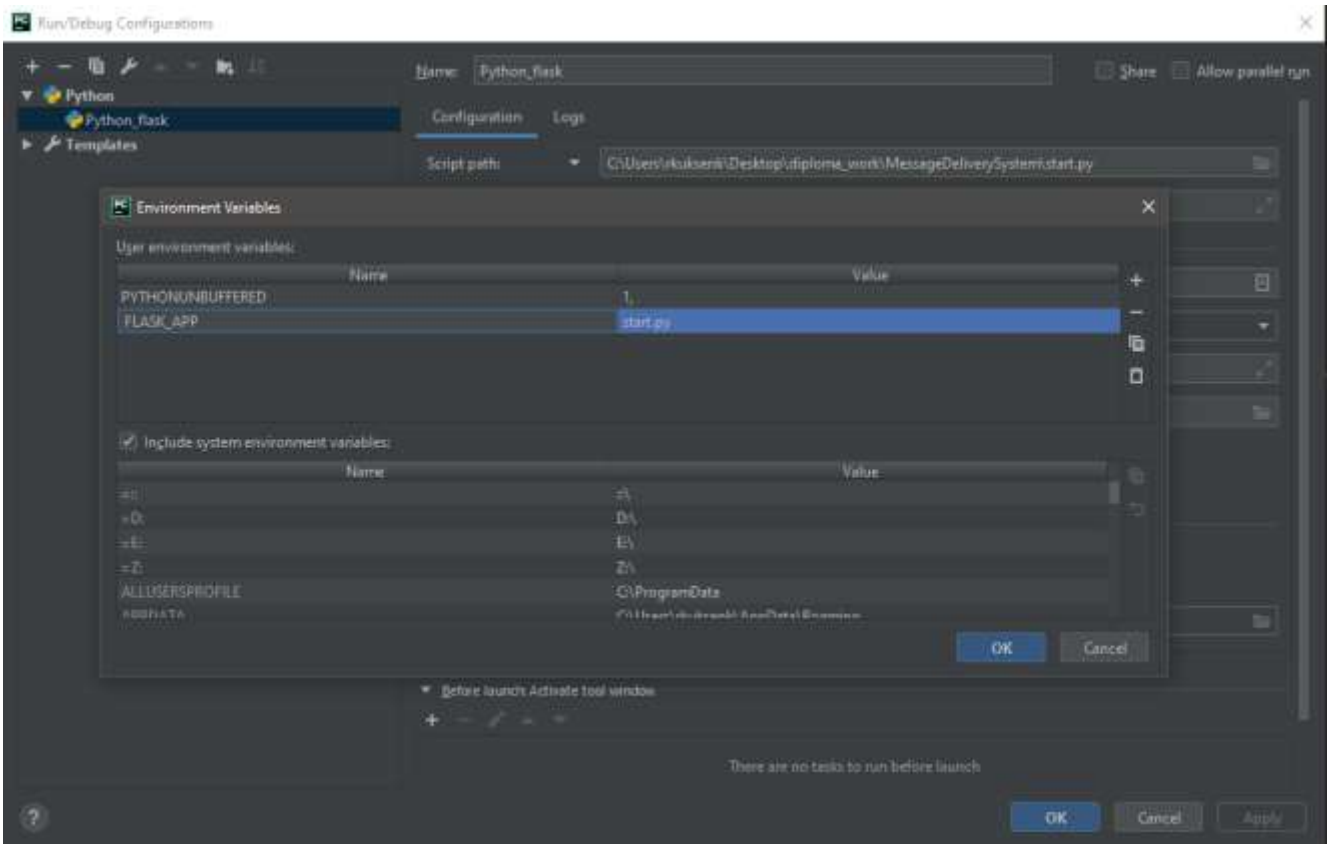

Fig.3.3. Setting of an environment variable

An interpreter in PyCharm provides us with a development environment for a specific programming language. Thus, the first task after creating the project is to set up the PyCharm Interpreter. Before setting up the PyCharm interpreter, we will create a Python project.

The application configuration for a specific environment must be stored in environment variables (not in the application source code). This allows you to change the configuration of each environment in isolation and prevents secure credentials from being stored under version control.

The project skeleton is ready, and it is possible to check the smallest Flask program to test it (Fig. 3.4).

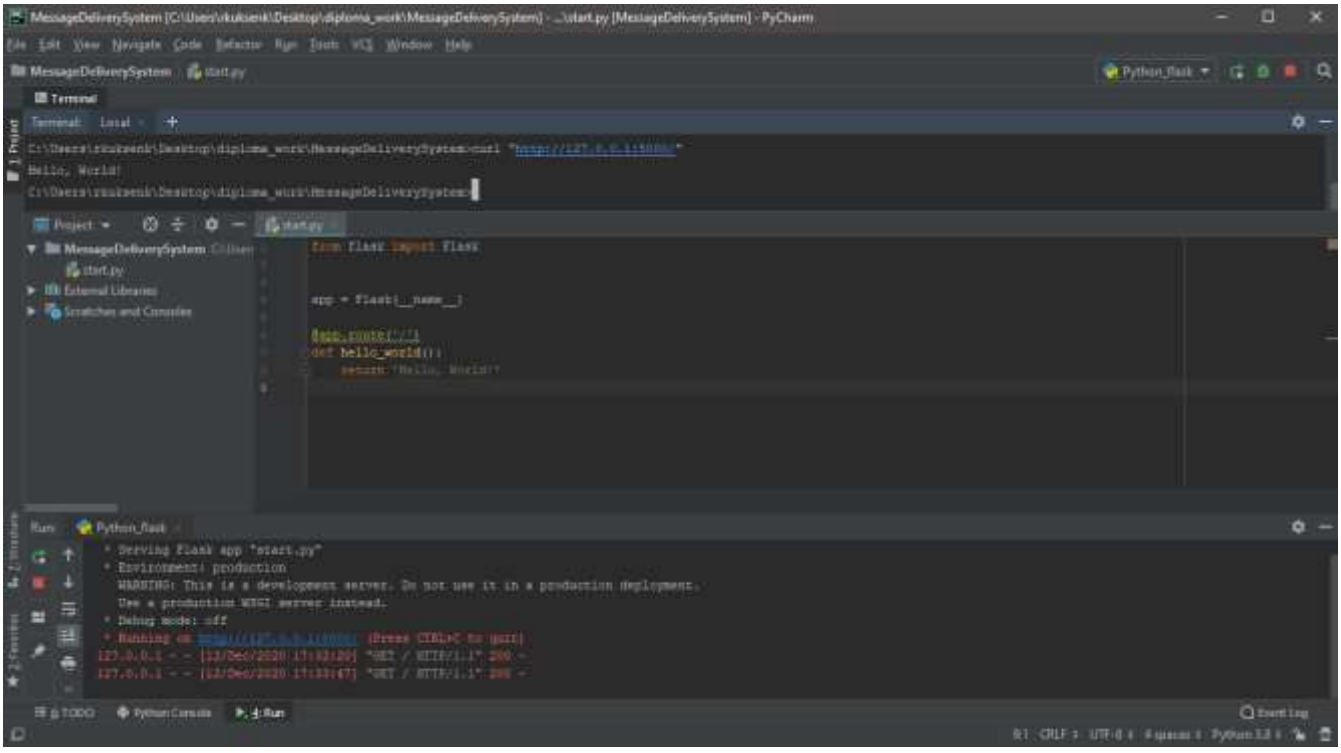

Fig.3.4. Running flask test server

3.2 Project design

Application Factory: The easiest way to get started with Flask is to instantiate the Flask class and set up the instance configuration. Application instances can then be used to route and register middleware as shown in the official docs.

Html files in Flask project should be passed into “/root_project/templates” directory. If project contains “css“ files it is required to past those files inside “/root_project/static” directory. All python files could be located directly in the root app root folder: “/root_project/files.py”.

The result project structure look like in the Fig. 3.4.

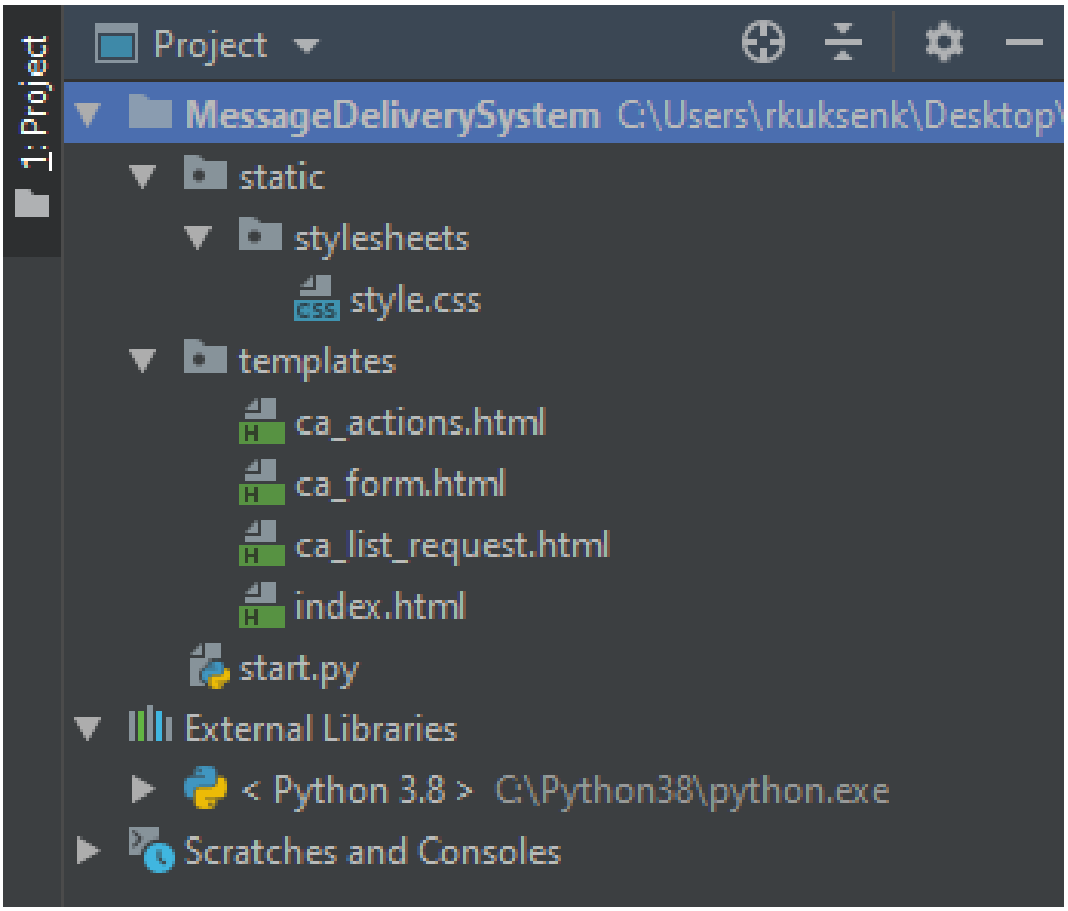

Fig.3.4. Result Flask project structure

The simple Flask project files structure look like the following:

/myapp

/run.py

/config.py

/app

/__init__.py

/views.py

/models.py

/static/

/main.css

/templates/

/base.html

/requirements.txt

/myappenv

Files definition:

- run.py - contains the actual python code that will import the app and start the development server;
- config.py - stores configurations for app;
- __init__.py - initializes application creating a Flask app instance;
- views.py - this is where routes are defined;
- models.py - this is where define models for application;
- static - contains static files i.e. CSS, Javascript, images;
- templates - this is where store html templates i.e. index.html, layout.html;
- requirements.txt - this is where store package dependancies, can use pip;
- myappenv - virtual environment for development.

Since the flask project is wrapped into the Docker there is a Dockerfile that describe how to build a docker image. In the Fig. 3.5 shown used Dockerfile.

```
FROM ubuntu:18.04
ADD ./MessageDeliverySystem /root/project/

RUN apt update
RUN apt install python3.8 -y
RUN apt install python3-pip -y
RUN apt install vim -y
RUN apt install curl -y
RUN apt install unixodbc-dev -y
RUN pip3 install flask
RUN pip3 install pyodbc
RUN pip3 install flask_menu


ENV FLASK_APP="/root/project/start.py"
ENV LC_ALL=C.UTF-8
ENV LANG=C.UTF-8

CMD python3 -m flask run -h 0.0.0.0 -p 5000
```

Fig.3.5. Dockerfile for building docker image

There are html pages that contains a jinja code that allows to dynamic update the static page. Jinja is a modern and developer-friendly templating language for Python, modeled on Django templates. It's fast, widely used, and secure with an additional sandboxed template runtime. In the Appendix A it is possible to acquainted with index.html page that is a start page for the whole project. Appendix B contains the listing of the html page with all possible actions. On the Appendix C and Appendix D there is a listing of ‘ca_form.html” page. Appendix F contains a HTML code for possible requests.

The listing of the main python file is shown on the Appendix F.

3.3 Dockerization

Dockerizing an application is the process of converting an application to run in a Docker container. While most applications dockerize is straightforward, there are several issues that need to be addressed each time.

During dockerization, two common problems arise:

- Force the application to use environment variables when it relies on configuration files;
- Sending application logs to STDOUT / STDERR when using files on the container file system by default.

The main idea why Docker technology was chosen is to make development process convenient for both Windows and Linux operation systems and to give a possibility to run the whole project on both Windows local machine and the AWS Linux EC2 instance.

To build a custom docker image it is only needed to make a Dockerfile and to run the following command from the directory where Dockerfile is located:

docker build . -t rkuksenko/message_delivery_system:latest

The command above build a docker image from the instructions provided in the Dockerfile. As a result will be generated a docker image with custom name:

“message_delivery_system”. A prefix “rkuksenko” indicate that this image could be pushed to the Docker Hub and it give possibility to download that image to anybody. A suffix “latest” indicates a docker image tag or version, latest means that it is a newest docker image version, but it is possible to set any tag names such as “1.0”, “fix”, “special_build_1.0”. It will help to see docker image history and easily downgrade to a specific version. The result of the docker image building is shown in the Fig. 3.6.

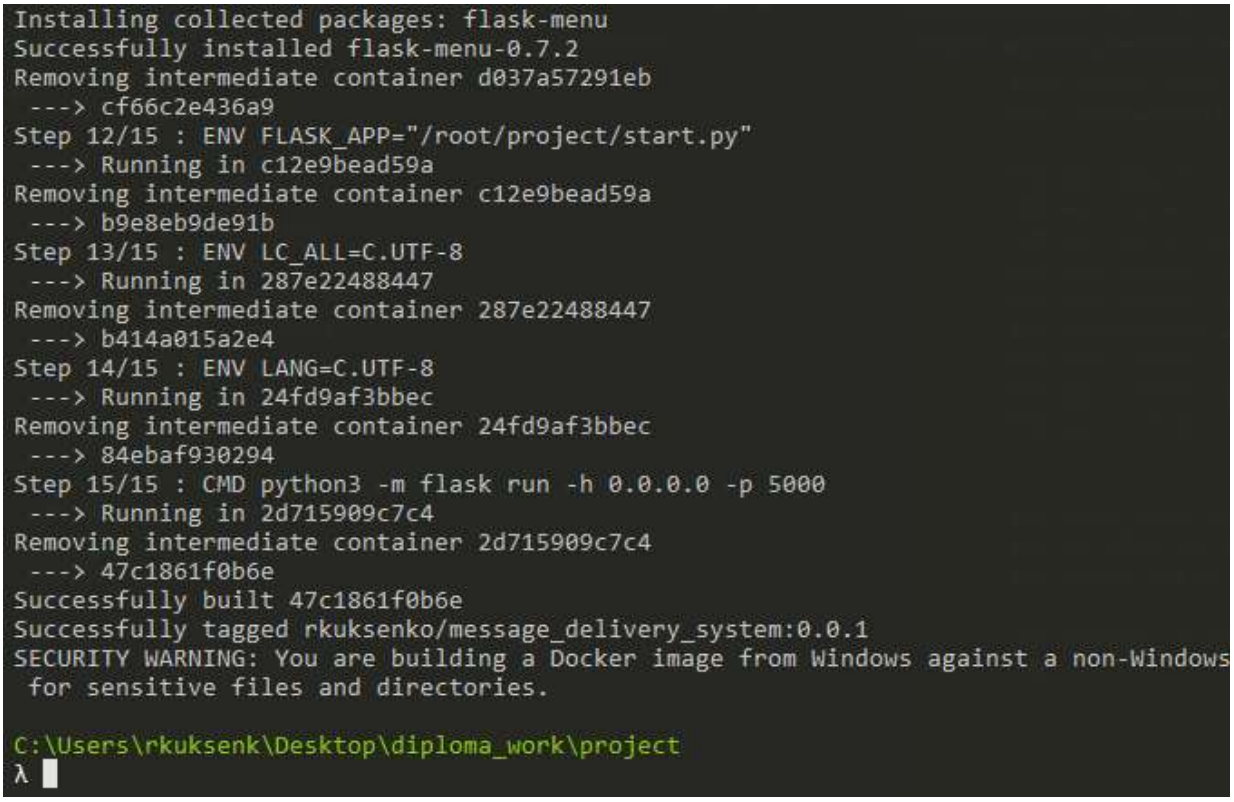

Fig.3.6. Result of docker build command

To push custom docker image it is only needed to run the following command:

docker push rkuksenko/message_delivery_system:0.0.1

In the Fig. 3.7 shown the result of the above command.

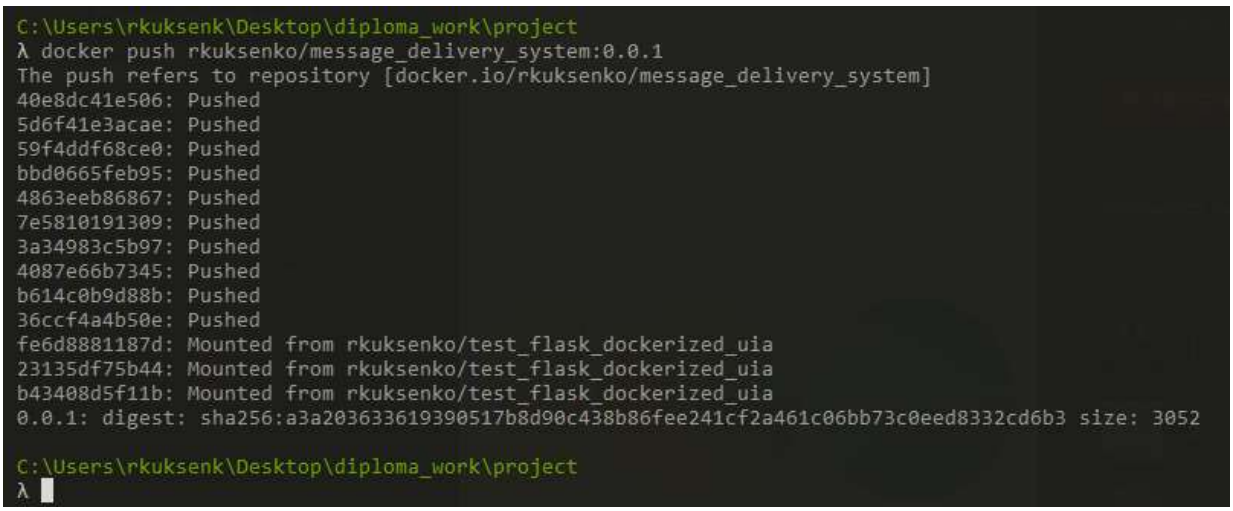

Fig.3.7. Result of docker push command

After pushing the created before docker image it is possible to see the uploaded docker image on the docker hub web site. The Fig. 3.8 display the uploaded docker image on the docker hub.

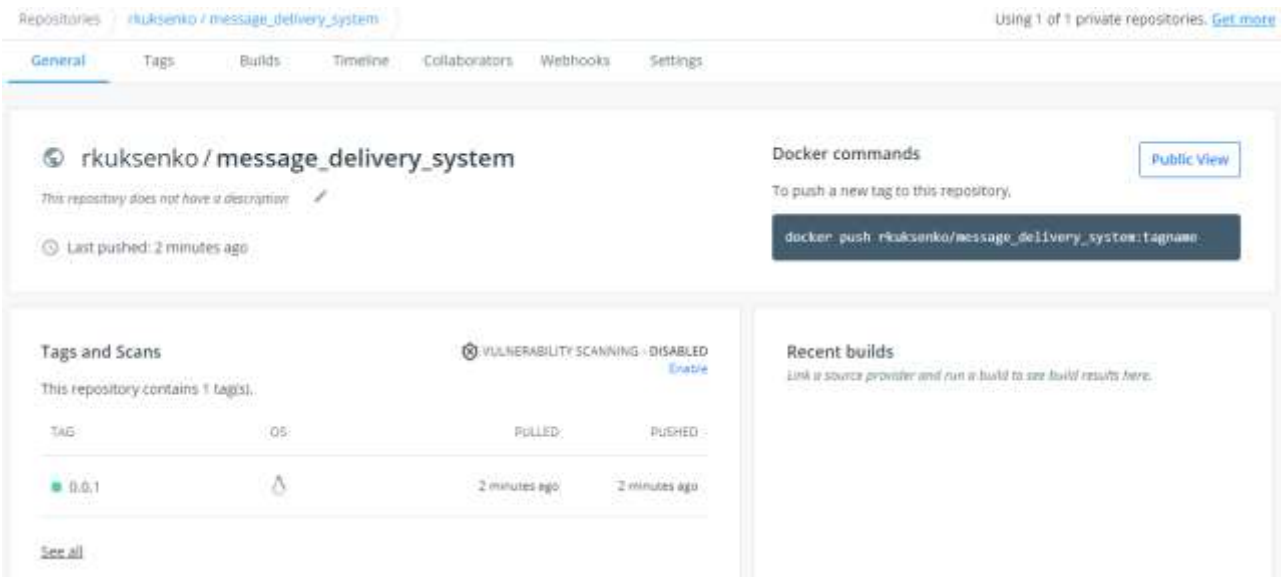

Fig.3.8. Uploaded docker image on the Docker hub

After the custom docker image was generated and pushed to the docker hub it is possible to pull it from the Windows or Linux operation system. To do this it is only required to install docker. To pull generated *message_delivery_system* docker need to run the following command:

docker pull rkuksenko/message_delivery_system:0.0.1

In the Fig. 3.9 shown the result of docker pull command.

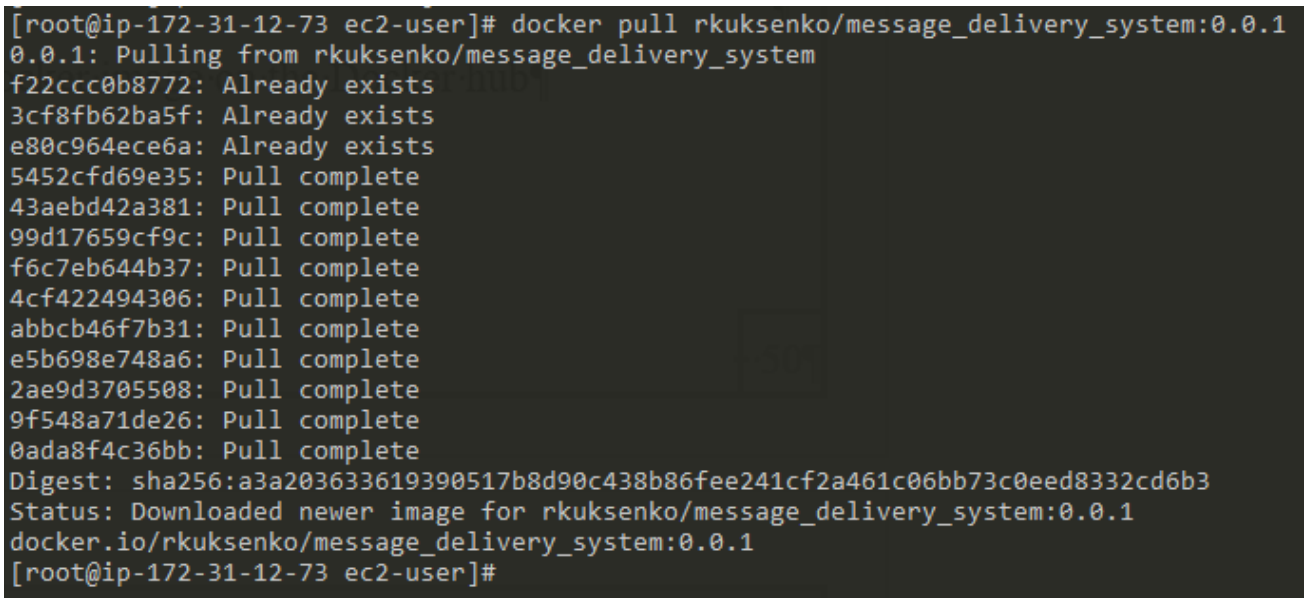

Fig.3.9. Result of docker pull command

After docker image pulled it is possible to run the flask web server.

3.4 Connect to the SQL server

SQL Server is a process viewed from an operating system perspective. Memory allocated from the operating system is owned by the process, just like the files being opened, and the process has a number of threads that the operating system can schedule to run. SQL Server was started as a service. If you check Services on your operating system, you may see several services, which friendly name starts with SQL. Below is a breakdown of the most common SQL Server services. Please note that the explanations are not intended to be exhaustive, but rather to provide an idea of what each service entails.

SQL Server is a relational database management system, or RDBMS, developed and marketed by Microsoft.

Similar to other RDBMS software, SQL Server is built on top of SQL, a standard programming language for interacting with the relational databases. SQL server is tied to Transact-SQL, or T-SQL, the Microsoft's implementation of SQL that adds a set of proprietary programming constructs.

SQL Server works exclusively on Windows environment for more than 20 years. In 2016, Microsoft made it available on Linux. SQL Server 2017 became generally available in October 2016 that ran on both Windows and Linux.

As SQL server for my project I chose a Microsoft SQL Server. To run it separately from the main python project I chose to use MS SQL Server in docker. So, to run MSSQL need pull and run the mssql docker. Here is a command required for starting the mssql docker:

docker run -e "ACCEPT_EULA=Y -e "SA_PASSWORD=P@ssw0rd" -p 1433:1433 --name mysql_UIA -h rkuksenko -d mcr.microsoft.com/mssql/server:2017-latest

After execution of that command a Microsoft SQL Server will be started on localhost on the port 1433 (Default MS SQL server port). The next step is to generate SQL queries. There is 3 files that represent a sql queries for the Message Delivery System. Create and fill database related queries are listed in the Appendix G. SQL Procedures for interaction with Crew table are presented in the Appendix H. SQL Procedures for crew requests are listed in the Appendix I.

In the Fig. 3.10 shown how to copy sql files inside a MSSQL docker.

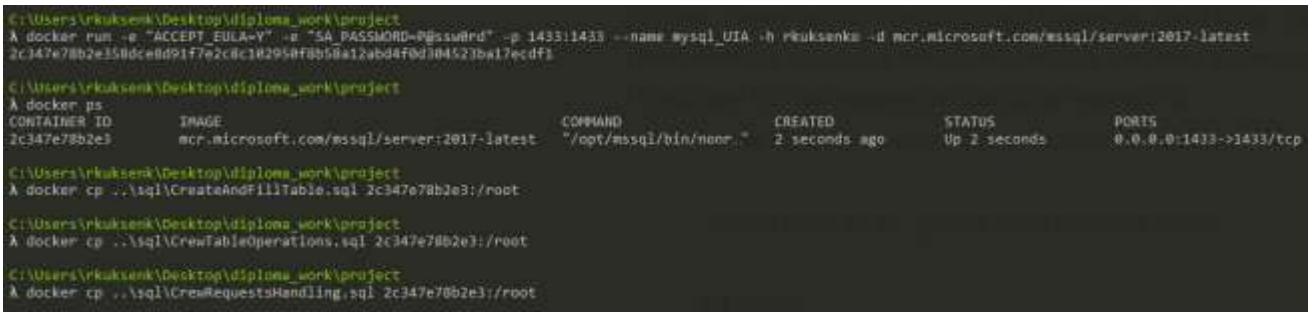

Fig.3.10. Copying sql files into the MSSQL docker

All sql queries need to apply, to run and apply all the .sql files inside a MSSQL docker need to run the following command:

/opt/mssql-tools/bin/sqlcmd -U SA `-P "P@ssw0rd" -I filename

In the Fig. 3.11 shown the result of applying the CreateAndFillTable.sql, CrewTableOperations.sql and CrewRequestsHandling.sql files.

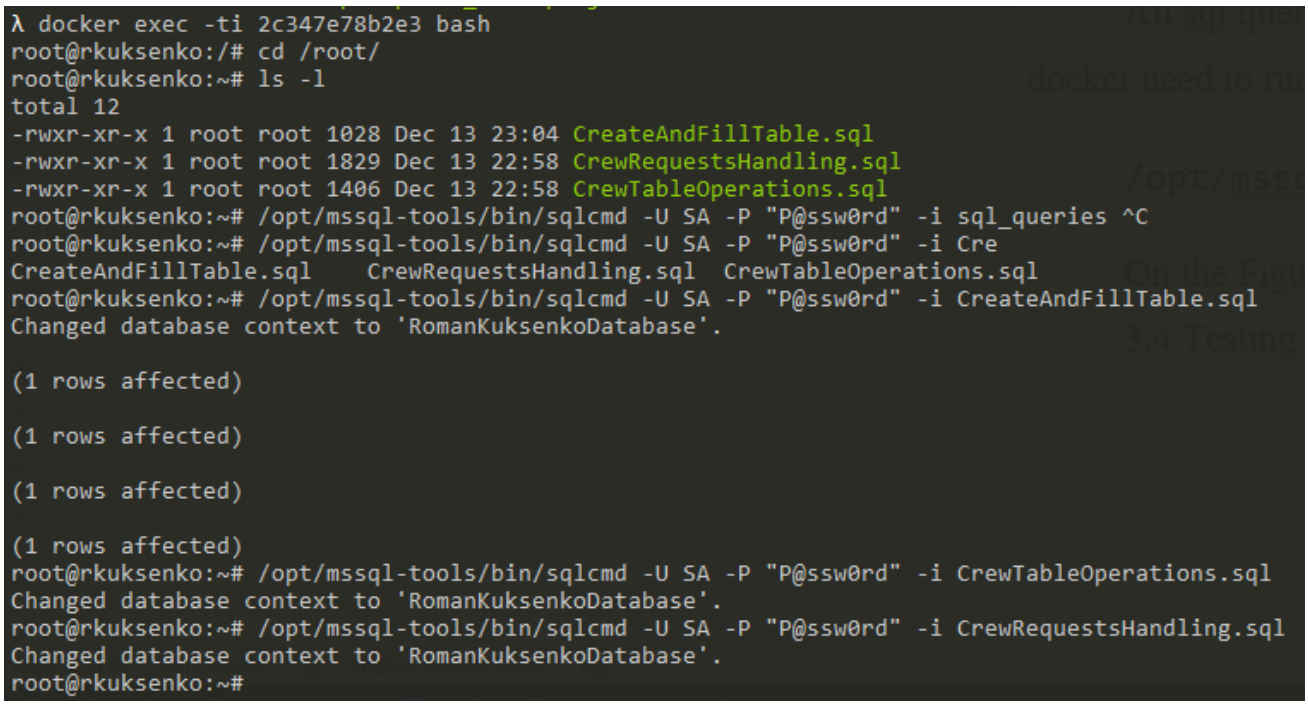

Fig.3.11. Run SQL queries in the MSSQL docker

There are multiple ways to connect python with sql server. There are next the most popular python libraries for interaction with sql:

- SQLite;
- MySQL;
- Pyodbc.

In my project I chose pyodbc library to connect sql databases to the project.

The main function that connect python with sql is the following: *pyodbc.connect()*

There are multiple configuration fields for connection with SQL server.

- DRIVER={SQL Server};
- SERVER=localhost;
- PORT=1433;
- DATABASE=RomanKuksenkoDatabase;
- UID=sa;
- PWD=P@ssw0rd.

In the Fig. 3.12 shown the small example with testing the RomanKuksenkoDatabase database created before.

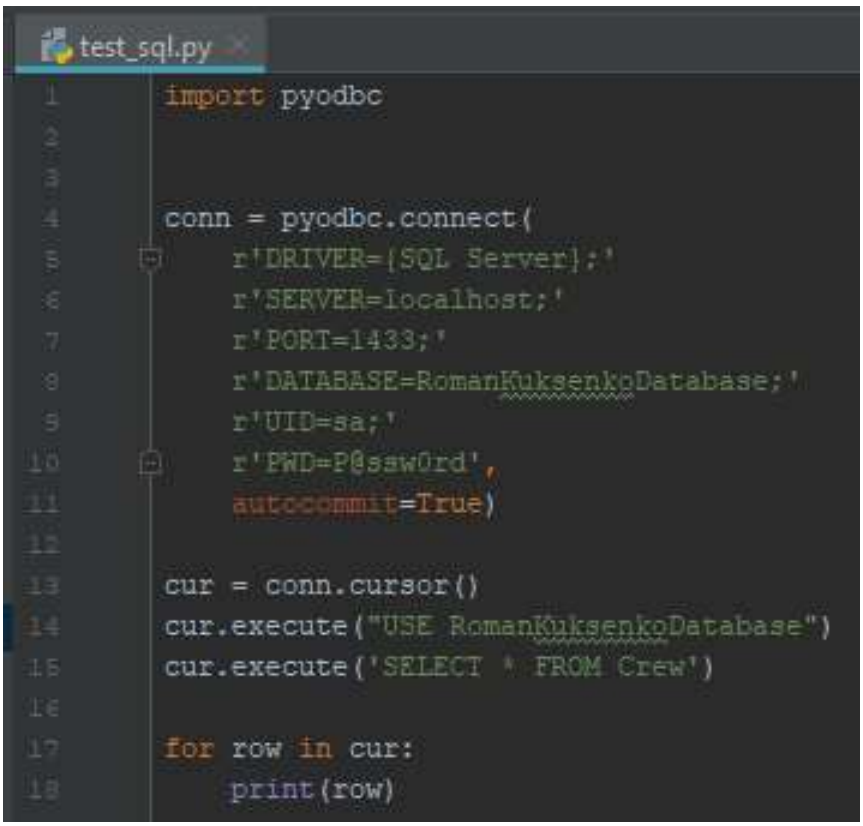

Fig.3.12. Test connection python with SQL server

After running the above python script the output is represented in the Fig. 3.13.

```
C:\Users\rkuksenk\Desktop\diploma_work\project\MessageDeliverySystem
λ python test_sql.py
(1232444, 'Anna Pashcenko', 'Kiev,Borcshagivska 23', '1232444@stud.nau.edu.ua', '0507654321', 1)
(2245634, 'Ivan Ivanov', 'Kiev,Harmatna 53', '2245634@stud.nau.edu.ua', '0501234567', 1)
(3987332, 'Roman Kuksenko', 'Kiev,Yunatska 6', '3987332@stud.nau.edu.ua', '0668585998', 2)
(5465466, 'Illya Borysenko', 'Kiev,Chernigivska 13', '5465466@stud.nau.edu.ua', '0671234567', 1)
```

Fig.3.13. Output of SQL test python script

So, the SQL server is running, python is connected to newly generated RomanKuksenkoDatabase database.

To backup database run the following command:

/opt/mssql-tools/bin/sqlcmd -b -V16 -S localhost -U SA -Q "BACKUP DATABASE [RomanKuksenkoDatabase] TO DISK = N'/var/opt/mssql/backups/backup' with NOFORMAT, NOINIT, NAME = 'RomanKuksenkoDatabase-full', SKIP, NOREWIND, NOUNLOAD, STATS = 10"

The result of database backup shown in the Fig. 3.14.

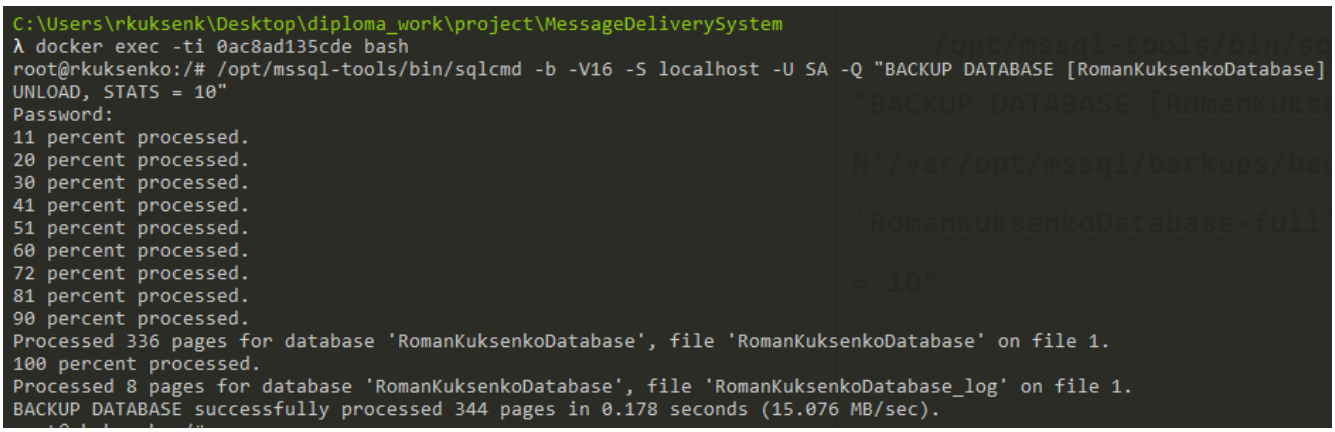

Fig.3.14. Result of database backup

To restore database from the backup, use the following command:

/opt/mssql-tools/bin/sqlcmd -S localhost -U SA -Q "RESTORE DATABASE [RomanKuksenkoDatabase] FROM DISK = N'/var/opt/mssql/restores/backup' WITH FILE = 1, NOUNLOAD, REPLACE, RECOVERY, STATS = 5"

As a result, all data will be restored. Fig. 3.15 shows that data was restored from the backup.

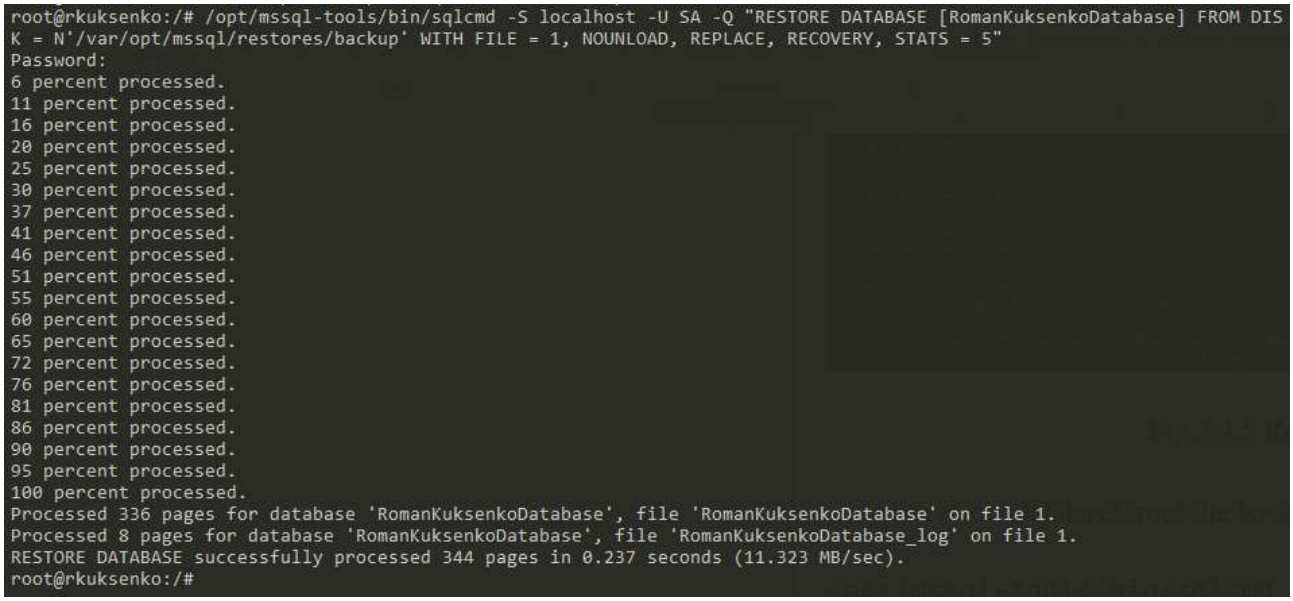

Fig.3.15. Result of database restore from a backup file

3.5 Deploy to the cloud

Amazon EC2 machine was deployed. A machine settings are shown in the Fig. 3.16.

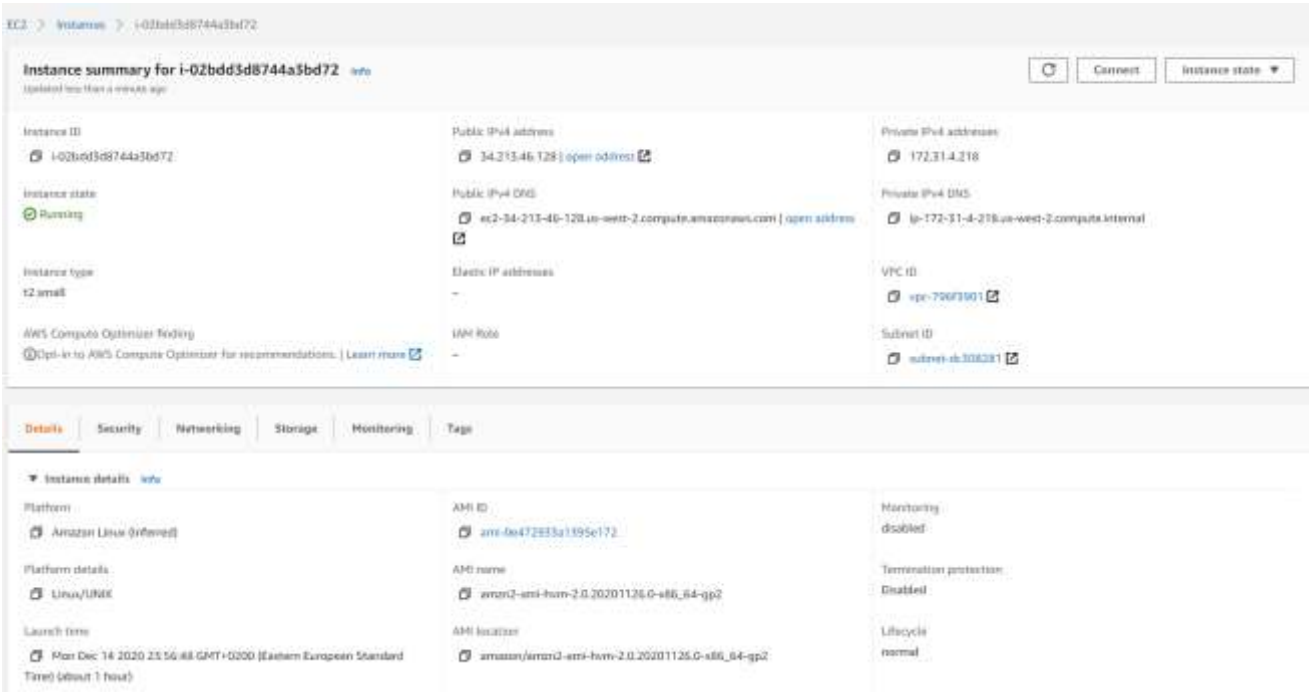

Fig.3.16. Deployed EC2 machine settings

To allow users connect to the server it is required to add inbound and outbound network rules. Message delivery system server is working on 80 port. In the Fig. 3.17 shown network rules to the EC2 instance.

▼ Inbound rules

Filter rules

Port range	Protocol	Source	Security groups
80	TCP	0.0.0.0/0	launch-wizard-2
22	TCP	0.0.0.0/0	launch-wizard-2

▼ Outbound rules

Filter rules

Port range	Protocol	Destination	Security groups
All	All	0.0.0.0/0	launch-wizard-2

Fig.3.17. ConFig.d network security rules

After running mssql docker and rkuksenko/message_delivery_system it is possible to check running dockers (Fig. 3.18).

Fig.3.18. Running dockers

Logs help to record all actions that occurs on the server. It is important to write a comprehensives log messages to exactly understand what action was performed. In case of failure logs can provide useful information for debugging. It is possible to configure. log file with the maximum size or set expiration data, that will interfer using too much memory for the logs.

It is possible to see logs and monitor status of the server using docker logs command. In the Fig. 3.19 shown a log of a running message_delivery_system. Also, all logs are stored inside the docker into preconfigure.d file. In my case it:

/root/project/log.log

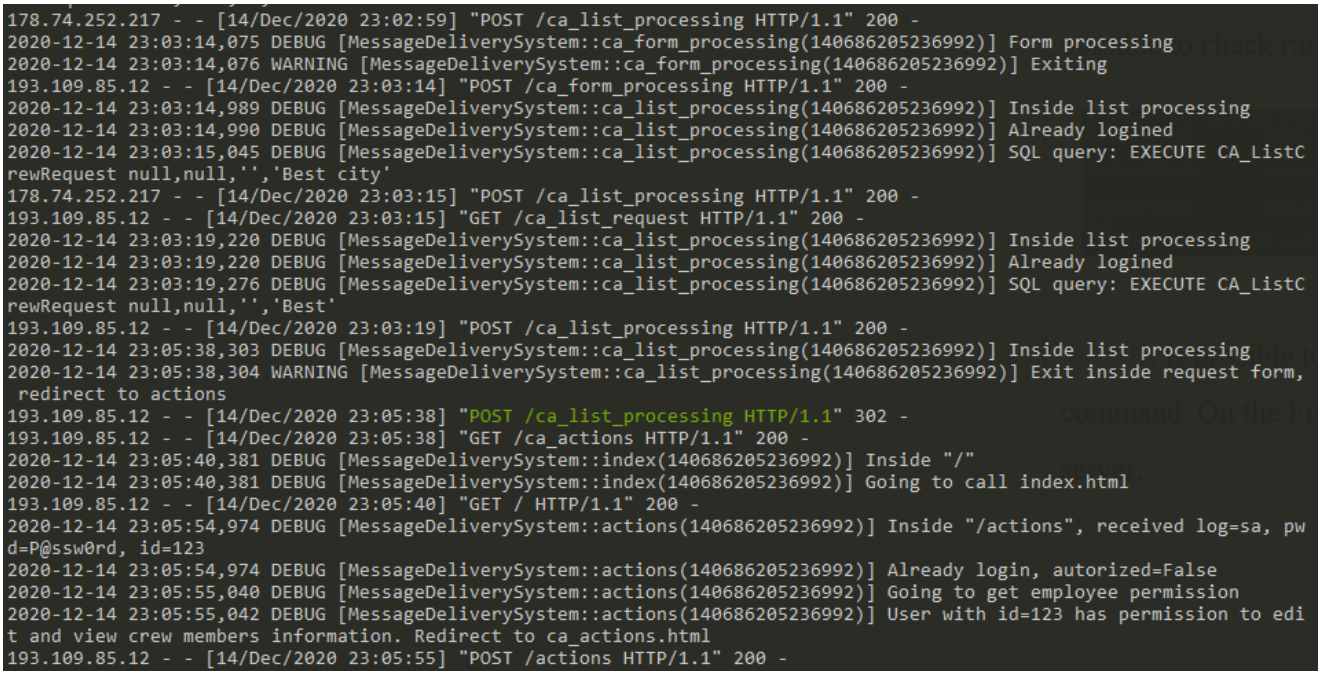

Fig.3.19. Logs of a running message_delivery_system

3.6 Testing

Current system is under develop but already done the following features:

- Login;
- Search by crew member id, address, phone number or name;
- Creating new records;
- Modification of existed records;
- Removing of existed records;
- Extracting information for a crew member by id, address, phone number or name;
- Possibility to search using any existed information about crew member.

In nearest future it is possible to link Crew information with Flight database. It allows to filter passenger using flight id or flight direction.

Existed service gives possibility to retrieve and use data in various ways. It is possible to connect running service to the dashboard and check for presents of a passenger before a flight.

In the Fig. 3.20 represented a first version of login page to the message delivery system.

Fig.3.20. Login page to the message delivery system

User with employee id 123 already granted an access to modification and creating a data record (system administrator).

Fig.3.21. A list of possible action for a system administrator

Option “Search for crew member records” redirects to the page to field any of the stored data (Fig. 3.22). After input some data the program will perform SQL query to find and retrieve matched data. (Fig. 3.23).

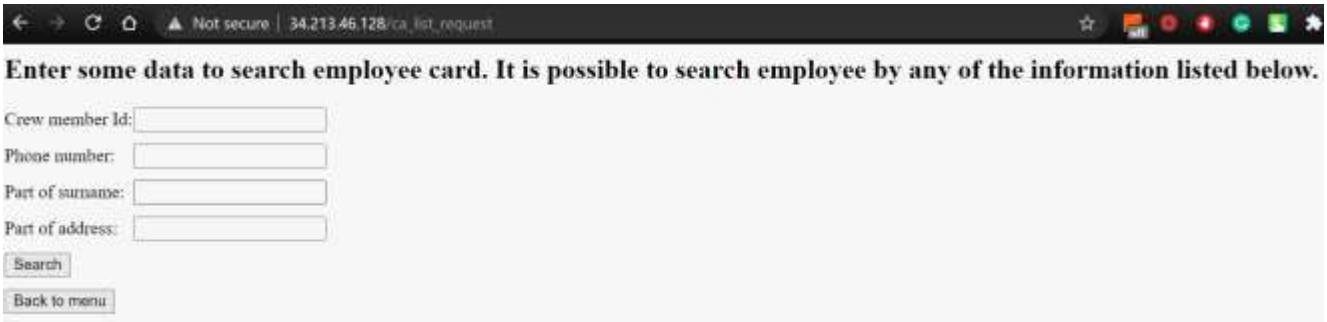

Fig.3.22. Crew member search page

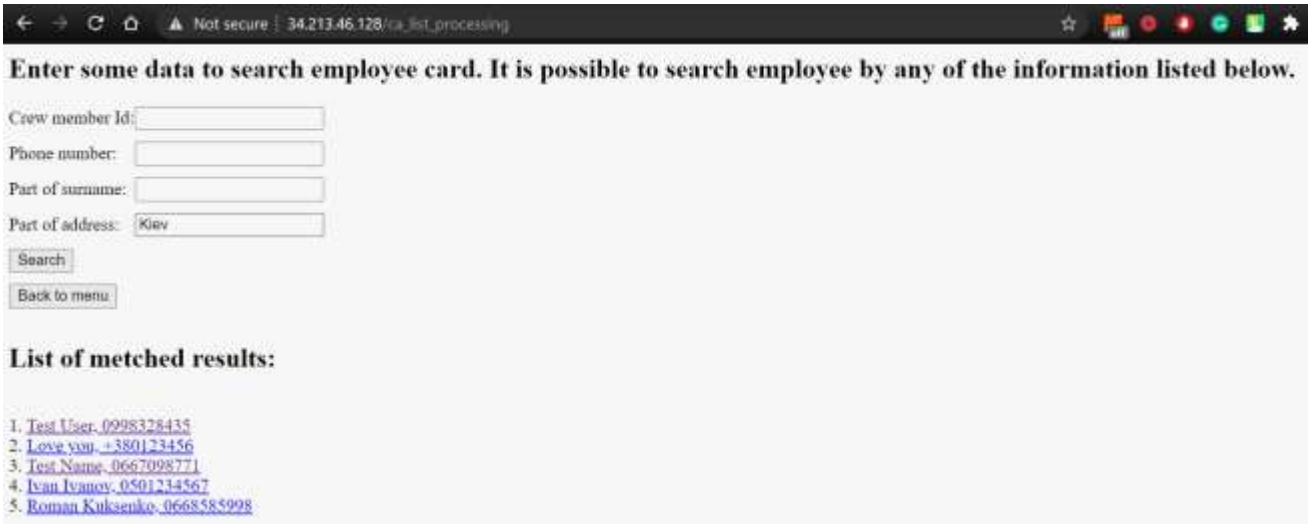

Fig.3.23. Result of search by the address

After selecting needed passenger, it is possible to see all available information. In the Fig. 3.24 represented an output of all available data for crew member “Ivan Ivanov”.

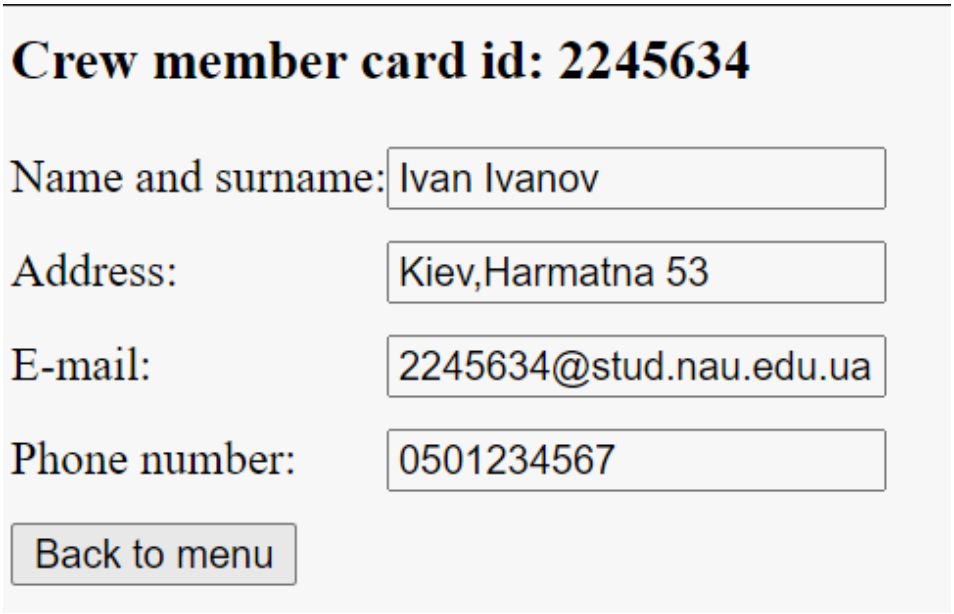

Fig.3.24. Information page for crew member “Ivan Ivanov”

The next option allows system administrator to add new crew member with filling all required data. In the Fig. 3.25 shown an example of successfully added data. It is required to set namely all data. Also email address should contain the following suffix:

@stdu.nau.edu.ua

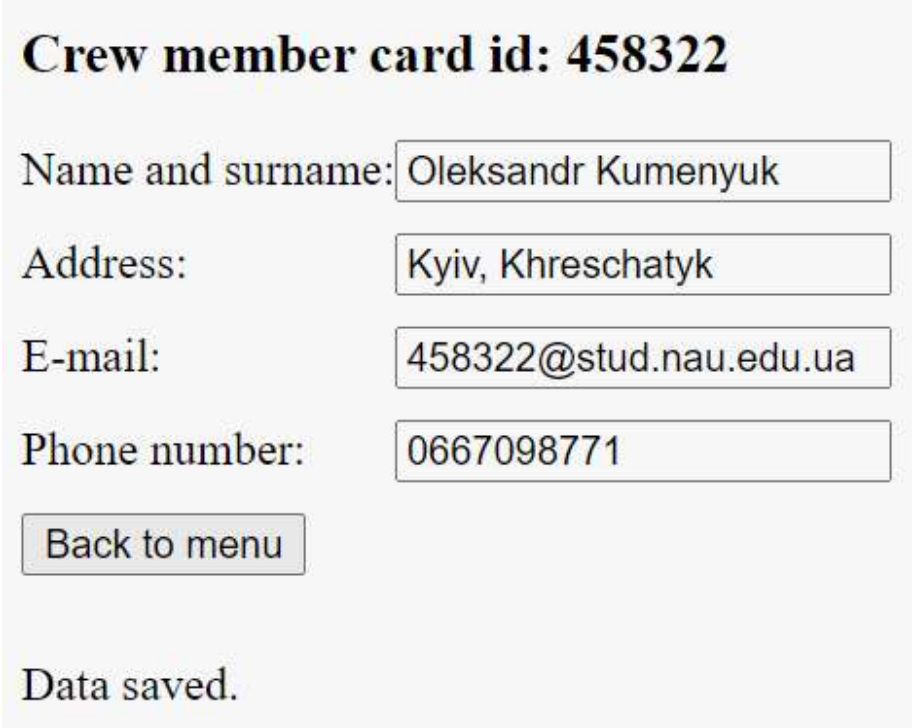

Fig.3.25. Example of successfully added new crew member

The next option allows to modify existed information for crew member. In the Fig. 3.26 represented an example of modification of an existed record.

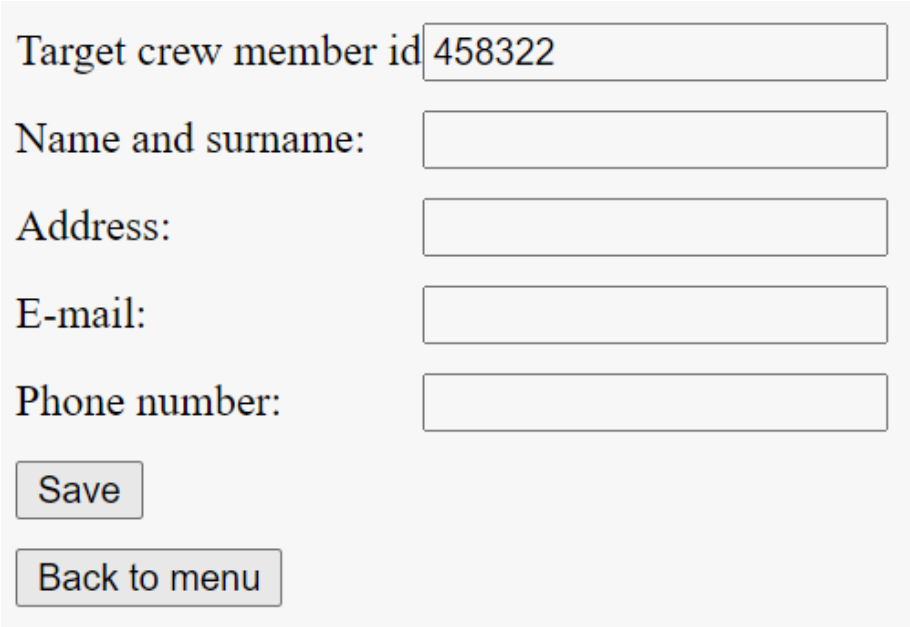

Fig.3.26. Example of modification data record

It is also possible to remove an existed record that already expired or not used for a long time. An option “Dismiss the crew member” allow to remove a record. In the Fig. 3.27 represented the dismiss page.

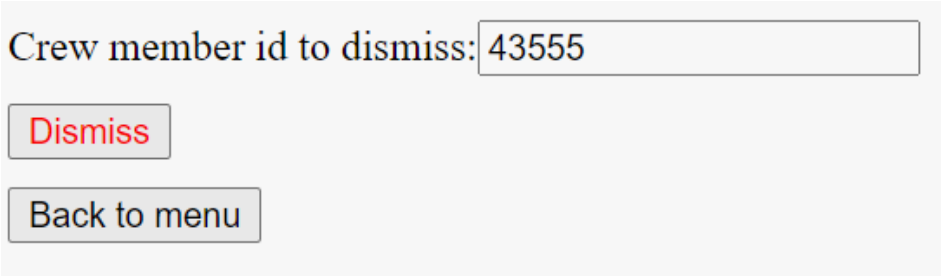

Fig.3.27. Dismiss page

In case of wrong “crew member id” system shows “Data not found” message (Fig. 3.28). This message generates by SQL server and forwards to the UI part of the project. SQL checks is more efficient and reliable in comparison between making checks on high level, in our case its python.

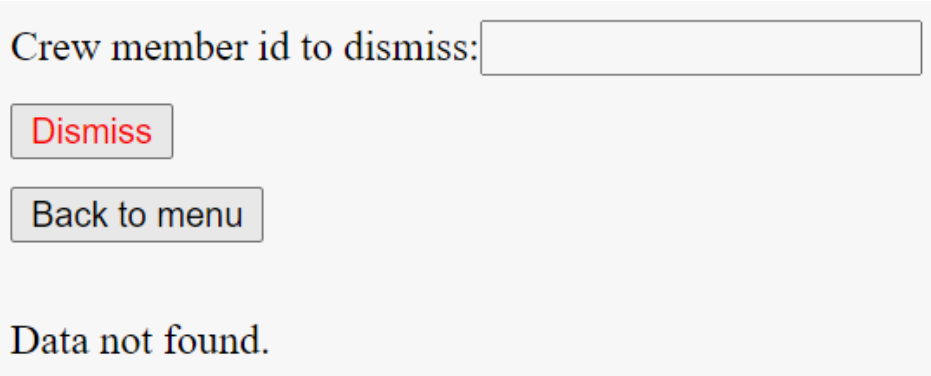

Fig.3.28. Failure during dismissing a data record

After successful dismiss procedure system will show “Data deleted” message (Fig. 3.29).

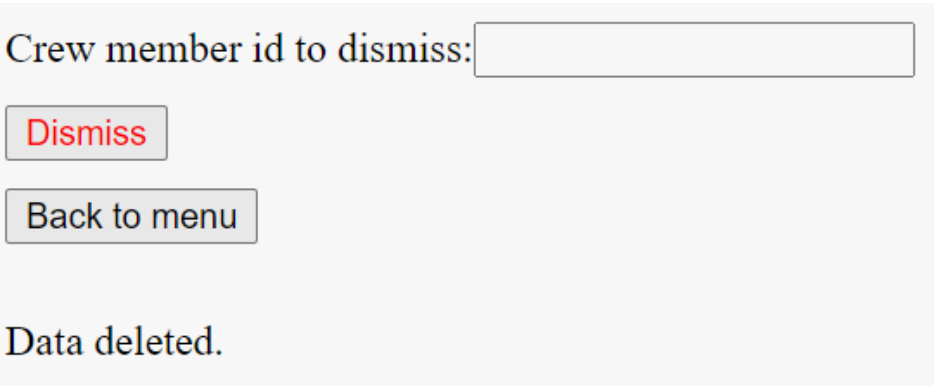

Fig.3.29. Successfully dismissed data record

After removing a data, it will be removed from database and it impossible to restore.

It is possible to check a project result using the following link: *http://34.213.46.128/*

In the Fig. 3.30 shown a QR code that redirect you to a message delivery system through http://34.213.46.128. It is possible to configure. a DNS name, but it required additional changes in web service provider, in our case – AWS.

Fig.3.30. QR code to visit message delivery system

CONCLUSIONS

A prototype of crew deliveries messages system was created. This system was written with a help of Python as a backend programming language, SQL for a interaction with databases and HTML for web presentation.

System supports restricted amount of features but already developed storying a crew member personal data, such as name, address, phone number, email in a database; data retrieving and filtration from database by any of existed personal information, modification and adding new data for new crew members. The system works with primitive user interface, because it can be used in web page, graphics on dashboards, mobile phone applications, other existed services, etc. It is already deployed on the AWS cluster and available via its public IP. As system available and can be tested it helps to faster develop, integrate and maintain the product. A docker technology shows that it is convenient and profitable to use it, as it can be developed on any OS platforms, easy to collect lots of libraries and dependencies.

Due to easier and more convenient programming language was selected it is possible to say that a Python web framework, namely Flask can fully interchange older backend system written with a help of PHP. The speed of development in Python, its reliability, many features and more current solutions make it clear that Python is fast, profitable, reliable and clearly no worse than PHP.

The current state of the project, of course, does not allow it to become a full replacement for an existing product, but in the future, there is great potential to expand functionality, adding new features and improving the quality of the project as a whole.

In general, it is possible to say that such a system can really be written using the mentioned technologies and used by the UIA company. Current prototype required a lot of improvements and changes both on databases and backend sides.

REFERENCE LIST

1. UIA [Electronic resource]. – 2020. – Access mode: https://www.flyuia.com/ua/en/services/booking-form (last access: 17.12.2020). Title from the screen.

2. Bill Karwin. "SQL Antipatterns: Avoiding the Pitfalls of Database Programming Pragmatic Programmers". 2007. Chapter 3, 5, 11, 15

3. SQL Shak. [Electronic resource]. 2020. Access mode: https://www.sqlshack.com/learn-to-write-basic-sql-queries/ (last access: 17.12.2020). Title from the screen.

4. Python official [Electronic resource]. – 2019. – №l0, Vol. 1 – Access mode: https://www.python.org/ (last access: 17.12.2020). Title from the screen

5. Michael W., Andreas W. "Amazon Web Services in Action". 2015. Chapter 11, 143-145 pages.

6. Flask Pallets Project [Electronic resource]. 2018. Access mode: https://flask.palletsprojects.com/en/1.1.x/quickstart/ (last access: 17.12.2020). Title from the screen.

7. Daniel G., Jack S. "Mastering Flask Web Development: Build enterprise-grade, scalable Python web applications, 2nd Edition". 2018. 24-36 pages.

8. James T. "The Docker Book: Containerization is the new virtualization". 2018. Chapter 1, 4, 6.

9. Linux Foundation Officical. – [Electronic resource]. 2018. Access mode: https://www.linuxfoundation.org/ (last access: 17.12.2020). Title from the screen.

10. Kulgin M. "Technology of corporate networks. Encyclopedia. " SPb, Peter, 2001, 156-160 pages.

11. Python Basics. – [Electronic resource]. 2018. Access mode: https://pythonbasics.org/deploy-flask-app/ (last access: 17.12.2020). Title from the screen.

12. Olifer VG, Olifer N.A. "Computer networks. Principles, Technologies, Protocols, 2nd Edition »SPb, Peter-press, 2005 56-70 pages.

13. Maverick K. "AWS: Amazon Web Services, the Ultimate Guide for Beginners to Advanced". 2017. Chapter 1-5.

14. Novikov U. N. "Local networks: architecture, algorithms, designing". M, ECOM, 2000, 405-410 pages.

15. Eric M. “Python Crash Course”. 2016. 110-125 p.

16. Paul-Barry V.O. “Head-First Python, 2nd edition” - St. Petersburg. 2016. Chapter 12-15.

17. Network word. [Electronic resource]. 2019. Access mode: https://www.networkworld.com/article/3215226/what-is-linux-uses-featres-products-operating-systems.html (last access: 17.12.2020). Title from the screen.

18. Albert A. "Mastering AWS Security". 2015. 110-117 p.

19. Allen B. Downey. "Think Python: How to Think Like a Computer Scientist, 2nd edition". 2015. Chapter 4, 60-68 pages.

20. Anthony S., Kathryn D. H. "Effective Computation in Physics: Field Guide to Research with Python" - O’Reilly. 2015

21. Stackoverflow how to run a flask application. [Electronic resource]. URL: https://stackoverflow.com/questions/29882642/how-to-run-a-flask-application

22. Бойченко С.В., Іванченко О.В. Положення про дипломні роботи. 2017. СМЯ НАУ П 03.01(10).

23. Hands P.K. "Docker Container Ultimate Beginners Guide: Docker Concept and Hands-On Exercises" 2018. Chapter 11, 133-148 pages.

APPENDIX A

Listing of index.html file

```
<!DOCTYPE HTML"
<html>
   <head>
      <title>Login Page</title>
      <link type="text/css" rel="stylesheet" href="{{ url_for('static',
filename='stylesheets/style.css') }}"/>
   </head>
   <body>
      <h1>Welcome to Crew delivery messages system, please log in</h1>
      <div>
         <form method="post" action="/actions" display="table">
            <p>
               <label>Username: </label>
               <input type="text" name="log">
            </p>
            <p>
               <label>Password: </label>
               <input type="password" name="pwd">
            </p>
            <p>
               <label>Employee Id: </label>
               <input type="text" name="id">
            </p>
            <button type="submit">Log in</button>
         </form>
      </div>
      {% if is_login_failed %}
         <h2 style="color:red">Wrong password or username!</h2>
      {% endif %}
      {% if missing_employee_id %}
         <h2 style="color:red">Employee with id = {{ employee_id }} is not
found!</h2>
      {% endif %}
  </body>
</html>
```

APPENDIX B

Listing of ca_actions.html file

```
<!DOCTYPE HTML"
<html>
<head>
   <title>Crew administration</title>
   <link type="text/css" rel="stylesheet" href="{{ url_for('static',
filename='stylesheets/style.css') }}"/>
</head>
<body>
      <div>
            <h1>You successfuly login. Chose an action:</h1>
            <button onclick="location.href='ca_list_request'" type="button"
class="actionsBtn">Search for crew member records</button><br>
            <button onclick="location.href='ca_form?is_add_employee_action=True'"
type="button" class="actionsBtn">Add new crew member record</button><br>
            <button onclick="location.href='ca_form?is_dismiss_action=True'"
type="button" class="actionsBtn">Dismiss the crew member</button><br>
            <button onclick="location.href='ca_form?is_update_info_action=True'"
type="button" class="actionsBtn">Update crew member info</button><br>
            <button onclick="location.href='/'" type="button"
class="actionsBtn">Exit</button><br>
      </div>
</body>
</html>
```

APPENDIX C

Listing of ca_forms.html file

```
<!DOCTYPE HTML"
<html>
   <head>
      <title>Crew member card {{ row[0] }}</title>
      <link type="text/css" rel="stylesheet" href="{{ url_for('static',
filename='stylesheets/style.css') }}"/>
   </head>
   <body>
      {% if not is_update_info_action and not is_dismiss_action %}
         <h3>Crew member card id: {{ row[0] }}</h3>
      {% endif %}
      <form method="post" action="/ca_form_processing" display="table">
         {% if not is_dismiss_action %}
         {% if is_update_info_action %}
         <p>
            <label>Target crew member id</label>
            <input type="text" name="id"></br>
         </p>
         {% endif %}
         <p>
            <label>Name and surname:</label>
            <input type="text" name="description" value="{{ row[1] }}"></br>
         </p>
         <p>
            <label>Address:</label>
            <input type="text" name="address" value="{{ row[2] }}"></br>
         </p>
         <p>
            <label>E-mail:</label>
            <input type="text" name="email" value="{{ row[3] }}"></br>
         </p>
         <p>
            <label>Phone number:</label>
            <input type="text" name="phone" value="{{ row[4] }}"></br>
         </p>
         {% if is_update_info_action %}
         <p>
            <button type="submit" name="save">Save</button></br>
         </p>
         {% endif %}
         {% if is_add_employee_action %}
         <p>
```

```
            <label>New crew member Id:</label>
            <input type="text" name="id"></br>
        </p>
        <p>
            <button type="submit" name="add">Save changes</button></br>
        </p>
        {% endif %}
        {% endif %} <! -- if not is_dismiss_action -->
        {% if is_dismiss_action %}
            <p>
                <label>Crew member id to dismiss:</label>
                <input type="text" name="id"></br>
            </p>
            <p>
                <button type="submit" name="dismiss"
style="color:red">Dismiss</button></br>
            </p>
        {% endif %}
        <p>
            <button type="submit" name="exit">Back to menu</button></br>
        </p>
        </br>
        {{ response }}
    </form>
  </body>
</html>
```

APPENDIX D

Listing of ca_list_request.html file

```
<!DOCTYPE HTML" <html>
    <head>
        <title>Crew members list</title>
        <link type="text/css" rel="stylesheet" href="{{ url_for('static',
filename='stylesheets/style.css') }}"/>
    </head>
    <body>
    <h2>Enter some data to search employee card. It is possible to search employee by any of
the information listed below.</h2>
    <form method="post" action="/ca_list_processing" display="table">
        <p>
            <label>Crew member Id:</label>
            <input type="text" name="employee_id" value="{{ f_id }}">
        </p>
        <p>
            <label>Phone number:</label>
            <input type="text" name="phone" value="{{ f_phone }}">
        </p>
        <p>
            <label>Part of surname:</label>
            <input type="text" name="description" value="{{ f_description }}">
        </p>
        <p>
            <label>Part of address:</label>
            <input type="text" name="address" value="{{ f_address }}">
        </p>
        <p>
          <button type="submit">Search</button></br>
        </p>
        <p>
            <button type="submit" name="exit" >Back to menu</button>
        </p>
     </form>
   {% if rows|length > 0 %}
      <h2>List of metched results:</h2>
   {% endif %}
   {% if f_empty_search_result %}
      <h2>Your search did not match any records:(</h2>
   {% endif %}
   {% for row in rows %}
      </br>
      {{ loop.index }}. <a href="/ca_form?id={{ row[0] }}">{{ row[1] + ", " + row[2] }}</a>
   {% endfor %}
  </body>
</html>
```

APPENDIX E

Listing of start.py file

```
import logging
import pyodbc
from flask import Flask, session, render_template, request, redirect, url_for
from flask import render_template_string
from flask_menu import Menu, register_menu
from MessageDeliverySystem.database_utils import get_crew_row,
init_sql_connection, exec_sql_query
import MessageDeliverySystem.loging_utils

app = Flask(__name__)
app.secret_key = 'Some very secret key'
logger = logging.getLogger('MessageDeliverySystem')

@app.route('/')
def index():
    logger.debug('Inside "/"')
    session.pop('login', None)
    session.pop('password', None)
    session.pop('employee_id', None)
    logger.debug('Going to call index.html')
    return render_template('index.html',
                           is_login_failed=request.args.get('is_login_failed'),
                           missing_employee_id=request.args.get('missing_employee_id'),
                           employee_id=request.args.get('employee_id'))

@app.route('/actions', methods=['GET', 'POST'])
def actions():
    log = request.form['log']
    pwd = request.form['pwd']
    id = request.form['id']

    logger.debug(f'Inside "/actions", received log={log}, pwd={pwd}, id={id}')

    if id is None or len(id) == 0:
        logger.error('Id was not filled, redirecting to the ca_wrong_input.html')
        return render_template('ca_wrong_input.html', wrong_employee_id=True)

    session["login"] = log
    session["password"] = pwd
    session["employee_id"] = id
    if 'login' in session:
        logger.debug(f'Already login, autorized=False');
```

```
        autorized = False
        while not autorized:
            try:
                autorized = True
                sql_connection = init_sql_connection(session['login'], session['password'])
            except pyodbc.InterfaceError:
                logger.error(f'Wrong password or user name, redirect to the index.html');
                autorized = False
                return redirect(url_for('index', is_login_failed=True))

        logger.debug(f'Going to get employee permission')
        cursor = sql_connection.cursor()
        cursor.execute(f'EXECUTE ALL_RoleByEmployeeId {session["employee_id"]}')
        row = cursor.fetchone()

        if row is None:
            logger.error(f'Specified employee with Id={id} was not found, redirecting to the
index.html')
            return redirect(url_for('index', missing_employee_id=True, employee_id=id))

        if row[0] == 2:
            logger.debug(f'User with id={id} has permission to edit and view crew members
information. '
                         f'Redirect to ca_actions.html')
            return render_template('ca_actions.html')
        else:
            logger.warning(f'User with id={id} has no permission to edit and view crew
members information. '
                           f'Redirect to index.html')
            return render_template('index.html')
    else:
        logger.error(f'Login failed, redirect to ca_list_request')

    return redirect(url_for('ca_list_request'))

@app.route('/ca_actions')
def ca_actions():
    return render_template('ca_actions.html')

@app.route('/ca_list_request')
def ca_list_request():
    return render_template('ca_list_request.html')
```

```
@app.route('/ca_list_processing', methods=['POST', 'GET'])
def ca_list_processing():
    logger.debug(f'Inside list processing')
    if 'exit' in request.form:
        logger.warning(f'Exit inside request form, redirect to actions')
        return redirect(url_for('ca_actions'))

    if 'login' in session:
        logger.debug(f'Already logined')

        if 'filter_id' in session:
            session.pop('filter_id', None)
            session.pop('filter_phone', None)
            session.pop('filter_description', None)
            session.pop('filter_address', None)

        u_id = session['filter_id'] = request.form['employee_id']
        phone = session['filter_phone'] = request.form['phone']
        name = session['filter_description'] = request.form['description']
        address = session['filter_address'] = request.form['address']

        f_id = if_empty_will_be_null(u_id)
        f_phone = if_empty_will_be_null(phone)

        sql_connection = init_sql_connection(session["login"], session["password"])
        sql_query = f"EXECUTE CA_ListCrewRequest
{f_id},{f_phone},'{name}','{address}'"

        logger.debug(f'SQL query: {sql_query}')
        rows = exec_sql_query(sql_connection, sql_query, fetch_all=True,
commit_after=False)

    return render_template('ca_list_request.html', rows=rows,
                           f_id=session['filter_id'],
                           f_phone=session['filter_phone'],
                           f_description=session['filter_description'],
                           f_address=session['filter_address'],
                           f_empty_search_result=len(rows) == 0)

@app.route('/ca_form')
def ca_form():
    logger.debug(f'/ca_form')
```

```
    if 'login' in session:

        logger.debug(f'/Already login')
        emp_id = request.args.get('id', '')
        sql_connection = init_sql_connection(session["login"], session["password"])
        cursor = sql_connection.cursor()
        row = get_crew_row(emp_id, cursor)
    else:
        logger.debug(f'Not login')

    return render_template('ca_form.html',
                           row=row,
                           is_add_employee_action=request.args.get('is_add_employee_action'),
                           is_update_info_action=request.args.get('is_update_info_action'),
                           is_dismiss_action=request.args.get('is_dismiss_action'))

@app.route('/ca_form_processing', methods=['POST', 'GET'])
def ca_form_processing():
    logger.debug(f'Form processing')

    if 'exit' in request.form:
        logger.warning(f'Exiting')
        return render_template('ca_actions.html',
                           f_id=session['filter_id'],
                           f_phone=session['filter_phone'],
                           f_description=session['filter_description'],
                           f_address=session['filter_address'])

    id = request.form['id']

    name = request.form['description'] if 'description' in request.form else ''
    address = request.form['address'] if 'address' in request.form else ''
    email = request.form['email'] if 'email' in request.form else ''
    phone = request.form['phone'] if 'phone' in request.form else ''
    role_id = 1
    dismiss_acion = False

    logger.debug(f'Received data:
id={id},name={name},address="{address}",email="{email}",phone="{phone}",'
                 f'roleId={role_id}')

    if 'add' in request.form:
        logger.debug(f'Will perform add action')
```

```
        sql_query = f'SET NOCOUNT ON; EXECUTE CA_InsertCrew
{id},"{name}","{address}","{email}","{phone}",{role_id}'
    elif 'dismiss' in request.form:
        logger.debug(f'Will perform dismiss action')
        sql_query = f'SET NOCOUNT ON; EXECUTE CA_DismissCrew {id}'
        dismiss_acion = True
    elif 'save' in request.form:
        logger.debug(f'Will perform save action')
        sql_query = f'SET NOCOUNT ON; EXECUTE CA_EditCrew
{id},"{name}","{address}","{email}","{phone}"'
    else:
        logger.debug(f'Nothing selected')
        sql_query = ''

    logger.debug(f'SQL query: {sql_query}')

    if 'login' in session:
        logger.debug(f'Already login')
        sql_connection = init_sql_connection(session["login"], session["password"])
        response = exec_sql_query(sql_connection, sql_query, fetch_one=True,
commit_after=True)
        logger.debug(f'Received response={response} from sql server')
        row = get_crew_row(id, sql_connection.cursor())
        sql_connection.close()
        return render_template('ca_form.html',
                               row=row,
                               response=response[0],
                               is_dismiss_action=dismiss_acion)
    else:
        logger.error(f'Need to login')
    return ''

@app.route('/wrong_input')
def ca_wrong_input():
    return render_template('ca_wrong_input.html')

def if_empty_will_be_null(str):
    if str == '':
        return 'null'
    else:
        return str
```

APPENDIX F

Listing of login_utils.py

```
import logging

formatter = logging.Formatter('%(asctime)s %(levelname)s
[%(name)s::%(funcName)s(%(thread)d)] %(message)s')
loggerName = 'MessageDeliverySystem'
logger = logging.getLogger(loggerName)
logger.setLevel(logging.DEBUG)
console_logger = logging.StreamHandler()
console_logger.setFormatter(formatter)
logger.addHandler(console_logger)
file_logger = logging.FileHandler(filename='/root/project/log.log',
                                  encoding='utf-8')
file_logger.setFormatter(formatter)
logger.addHandler(file_logger)
```

APPENDIX G

Listing of database_utils.py

```
import pyodbc
from flask import Flask, session, render_template, request, redirect, url_for

def get_crew_row(crew_id: int, cursor):
    cursor.execute("EXECUTE CA_FormCrewRequest " + str(crew_id))
    row = cursor.fetchone()
    return row

def init_sql_connection(login: str, password: str):
    connection = pyodbc.connect(
        r'DRIVER={SQL Server};'
        r'SERVER=localhost;'
        r'PORT=1433;'
        r'DATABASE=RomanKuksenkoDatabase;'
        r'UID=' + session["login"] + ';'
        r'PWD=' + session["password"],
        autocommit=True)
    return connection

def exec_sql_query(sql_connection, sql_query: str, fetch_one=False, fetch_all=False,
commit_after=False):
    cursor = sql_connection.cursor()
    cursor.execute(sql_query)

    response = ''
    if fetch_one:
        response = cursor.fetchone()
    elif fetch_all:
        response = cursor.fetchall()

    if commit_after:
        cursor.commit()
    return response
```

APPENDIX H

SQL procedures for create and fill table

```
USE RomanKuksenkoDatabase;
GO

CREATE TABLE Crew
  ([id] int PRIMARY KEY,
   [description] varchar(255) NOT NULL,
   [address] varchar(255) NOT NULL,
   [email] varchar(50) NOT NULL CHECK ([email] like '%@stud.nau.edu.ua'),
   [phone] char(10) NOT NULL CHECK (len([phone])=10),
   [roleId] int NOT NULL
  )
INSERT Crew ([id],[description],[address],[email],[phone],[roleId]) VALUES
(2245634,'Ivan Ivanov','Kiev,Harmatna 53','2245634@stud.nau.edu.ua','0501234567',1)
INSERT Crew ([id],[description],[address],[email],[phone],[roleId]) VALUES
(1232444,'Anna Pashcenko','Kiev,Borcshagivska
23','1232444@stud.nau.edu.ua','0507654321',1)
INSERT Crew ([id],[description],[address],[email],[phone],[roleId]) VALUES
(5465466,'Illya Borysenko','Kiev,Chernigivska
13','5465466@stud.nau.edu.ua','0671234567',1)
INSERT Crew ([id],[description],[address],[email],[phone],[roleId]) VALUES
(3987332,'Roman Kuksenko','Kiev,Yunatska
6','3987332@stud.nau.edu.ua','0668585998',2)

GO
```

APPENDIX I

SQL procedures for interaction with Crew table

```
USE RomanKuksenkoDatabase;
Go
CREATE PRoCEDURE [dbo].[CA_insertCrew] @crew_id iNT=NULL, @description
VARCHAR(255)='', @address VARCHAR(255)='', @email varchar(50)='', @phone CHAR(10)=NULL
AS
iF @crew_id iS NoT NULL -- to create and fill new crew member card
  BEGiN
    iNSERT Crew ([id], [description], [address], [email], [phone])
    VALUES (@crew_id, @description, @address, @email, @phone)
    iF @@RoWCoUNT = 0
      SELECT 'Data not saved.'
    ELSE
      SELECT 'Data saved.'
  END
ELSE
  SELECT 'No data available.'
Go

CREATE PRoCEDURE CA_DismissCrew @crew_id iNT=NULL
AS
iF @crew_id iS NoT NULL
  BEGiN
    DELETE Crew
     WHERE [id]=@crew_id
    iF @@RoWCoUNT = 0
      SELECT 'Data not found.'
    ELSE
      SELECT 'Data deleted.'
  END
ELSE
  SELECT 'No data available.'
Go

CREATE PRoCEDURE CA_EditCrew @crew_id iNT=NULL, @description VARCHAR(255)='',
@address VARCHAR(255)='',@email VARCHAR(50)='', @phone CHAR(10)=NULL
AS
iF @crew_id iS NoT NULL -- new data for crew member writes to database
  BEGiN
    UPDATE Crew
      SET [description]=@description, [address]=@address, [email]=@email, [phone]=@phone
     WHERE [id]=@crew_id
    iF @@RoWCoUNT = 0
      SELECT 'Data not saved.'
    ELSE
      SELECT 'Data saved.'
  END
ELSE
  SELECT 'No data available.'
Go
```

APPENDIX J

SQL Procedures for crew requests

```
USE RomanKuksenkoDatabase;
Go

CREATE PRoCEDURE CA_FormCrewRequest @crew_id iNT=NULL
AS
iF @crew_id iS NoT NULL -- retrieve the data about one crew member by his id
  SELECT [id] AS [Employee iD], [description] AS [Firstname and surname], [address] AS [Home
address], [email] AS [E-mail], [phone] AS [Phone number]
    FRoM Crew
   WHERE [id]=@crew_id
ELSE
  SELECT 'No data available.'
Go
CREATE PRoCEDURE CA_ListCrewRequest @crew_id iNT=NULL,
                          @phone CHAR(10)=NULL,
                          @part_of_description VARCHAR(50)='',
                          @part_of_address VARCHAR(50)=''
AS
IF @crew_id iS NoT NULL -- looking for crew member by his id
  SELECT [id] AS [Employee iD], [description] AS [Firstname and surname], [phone] AS [Phone
number]
   FRoM Crew
  WHERE [id]=@crew_id
ELSE iF @phone iS NoT NULL -- looking for crew member by his phone number
      SELECT [id] AS [Employee iD], [description] AS [Firstname and surname], [phone] AS [Phone
number]
       FRoM Crew
       WHERE [phone]=@phone
ELSE  -- request for all crew members with some details about firstname and surname or/and address
  SELECT [id] AS [Employee iD], [description] AS [Firstname and surname], [phone] AS [Phone
number]
    FROM Crew
   WHERE [description] LiKE '%'+@part_of_description+'%'
     AND [address] LiKE '%'+@part_of_address+'%'
Go

CREATE PRoCEDURE [dbo].[ALL_RoleByEmployeeid]  @employee_id int
AS
SELECT roleid FRoM CREW WHERE id=@employee_id
GRANT EXECUTE oN ALL_RoleByEmployeeid To [Crew member]
GRANT EXECUTE oN ALL_RoleByEmployeeid To [Crew administrator]
GRANT EXECUTE oN ALL_RoleByEmployeeid To [Flight administrator]
GRANT EXECUTE oN ALL_RoleByEmployeeid To [Transportation administrator]
Go
```