

Ministry of Education and Science of Ukraine
Sumy State University
Educational and Scientific Institute of Business, Economics and Management
Department of Economic Cybernetics

BACHELOR'S QUALIFICATION WORK

on the topic “Automation of the retail trade at the enterprise”

Completed student of 4th course, group AB-71a.an
(course number) (group code)

Specialties 051 “Economics” (Business analytics)

Yu.V. Lebedeva

(student's last name, initials)

Supervisor Dr. Sc. in Economics, professor,

O.V. Kuzmenko

(position, degree, last name, initials)

Ministry of Education and Science of Ukraine
Sumy State University
Educational and Scientific Institute of Business, Economics and
Management
Department of Economic Cybernetics

APPROVE
Head of the Department
Dr. Econ. Sciences, Professor
_____ Kuzmenko O.V.
“ ” _____ 2021

TASK
FOR THE BACHELOR'S QUALIFICATION WORK
in the direction of training 051 Economic (Business analytics)
student 4th year of the group AB-71a.an

Lebedeva Yulia Vitaliivna

1. Topic of the work: Automation of the retail trade at the enterprise approved by order of the university 0382-III from 15.03.2021.
2. The deadline for the student to submit the completed work " " _____ 2021.
3. The purpose of the work is to develop a prototype of a postage management module at a retail enterprise “Oxygen3000”.
4. The object of the study is the "Oxygen3000" enterprise.
5. The subject of research is the process of postage automation at the "Oxygen3000" enterprise.
6. Thesis is performed on materials of "Oxygen3000" enterprise.
7. Indicative plan of qualification work, terms of submission of sections to the head and the maintenance of tasks for performance of the set purpose

Section 1 Analysis of the current state of automation of business processes of the enterprise

In section 1 to reveal the characteristics of the object of automation and analysis of the current state of automation of business processes, the characteristics of existing postal services and solutions for automation of postal items, definition and formulation of requirements for future prototype.

Section 2 Implementation of the prototype of the postal management module

In section 2 make designing the architecture of the prototype of the postal management module, select the most useful technology for realization, develop the prototype of the postal management module, create reference example and instructions for use.

8. Consultations on work:

Chapter	Consultant	Signature, data	
		Task issued by	Task accepted by
1		Kuzmenko O.V.	Lebedeva Yu. V.
2		Kuzmenko O.V.	Lebedeva Yu. V.

9. Date of issue of the task“ ___ ”_____20__ p.

Supervisor _____ O.V. Kuzmenko
 Signature Initials, surname

Received the task to perform _____ Yu.V. Lebedeva
 Signature Initials, surname

ABSTRACT
of the qualifying work
for obtaining the educational and qualification level “bachelor”

Lebedeva Yulia Vitaliivna
(surname, name, patronymic of the student)

Automation of the retail trade at the enterprise

The relevance of qualification work is that in today's competitive environment, firms need to have significant competitive advantages to retain their own and attract new customers. One of the competitive advantages is the speed of postal items. The less the customer waits for his product – the greater his satisfaction. In addition, the current quarantine conditions for the COVID-19 were a significant challenge for retailers – outlets closed and firms could only sell their goods by mail. Automation of postal items saves the work of the company's manager, frees up man-hours, improves the company's service. The purpose of this work is to develop a prototype of a postage management module at a retail enterprise.

The object of the research is the “Oxygen3000” enterprise.

The subject of the research is the process of postage automation at the “Oxygen3000” enterprise.

Methods of research – analysis, algorithmization, synthesis.

Information base – LLC «Oxygen3000», python documentation, API Nova Poshta.

The main contribution of the work is the practical implementation of the prototype of the mail management module.

The work was implemented at LLC “Oxygen3000”, reference number 2021/2 from 09.06.2021.

Keywords: automation, mailings, New Mail API, programming, Django framework.

The content of the qualification work is presented on 35 pages. The references consist from 40 names, placed on 4 pages. The work contains 6 tables, 14 figures, 2 appendices.

Year of performance of qualification work – 2021.

Year of protection of work – 2021.

CONTENTS

INTRODUCTION.....	7
SECTION 1 ANALYSIS OF THE CURRENT STATE OF AUTOMATION OF BUSINESS PROCESSES OF THE ENTERPRISE	9
1.1 Characteristics of the object of automation and analysis of the current state of automation of business processes.....	9
1.2 Characteristics of existing solutions for automation of postal items	10
1.3. Formation of system requirements.....	17
SECTION 2 IMPLEMENTATION OF THE PROTOTYPE OF THE POSTAL MANAGEMENT MODULE.....	19
2.1 Designing the architecture of the prototype of the postal management module 19	
2.2 Selecting of implementation technology.....	23
2.3 Developing the prototype of the postal management module	25
2.4 Reference example and instructions for use	33
CONCLUSIONS.....	36
REFERENCES.....	37
APPENDICES.....	41

INTRODUCTION

In the 21st century, almost all processes have become automated. Automation of business processes gives us the opportunity to conduct and control our business without making a lot of effort, because one program can do in a couple of minutes what a person will do for several hours. With the increase in productivity, the income of the company also increases, and all this is due to information systems and technologies.

To date, to open your own business, it is not enough to have only a sales outlet and a good supplier, automation is most relevant for managing business processes.

The main advantages of automation are:

- the emergence of an automated management system (CRM, ERM, PM, etc.), where it will be possible to see what tasks are and when they need to be done;
- minimization of the human factor in business processes. This includes forgotten data, procrastination with simple tasks, clients that have not been entered into the database, etc.;
- saving data from cyber attacks or illegal access. With an automated control system, you will enter all the actions in the system and, in which case, you will be able to track them. It is clear that automation does not solve all security problems, but it makes life much easier.

The purpose of this work is to develop a prototype of a postage management module at a retail enterprise.

The object of the research is the «Oxygen3000» enterprise.

The subject of the research is the process of postage automation at the “Oxygen3000” enterprise.

The main tasks of qualifying research work are:

- to characterize the object of research and analyze the state of automation of business processes;

- to analyze the mail services existing on the market and their solutions for delivery automation;
- to formulate requirements for a web-oriented system;
- to carry out the design of the prototype for the module of postal mailing;
- to reverse the technology corks to the prototype;
- to develop a prototype of an automated postage system;
- develop a test case and instructions for use.

SECTION 1 ANALYSIS OF THE CURRENT STATE OF AUTOMATION OF BUSINESS PROCESSES OF THE ENTERPRISE

1.1 Characteristics of the object of automation and analysis of the current state of automation of business processes

The object of the study is LLC “Oxygen3000”. Today there are many options for starting your own business. And in this case, the company focused on a limited liability company – one of the most popular organizational forms of business in Ukraine. LLC is a business company, the authorized capital of which is divided into shares, the size of which is determined by the constituent documents.

This enterprise is engaged in realization and sale of oxygen in cylinders of the Ukrainian production on various trading platforms, such as the socket, prom.yua and also through the website. Now this company wants to increase turnover and automate the registration and shipment of its goods through the site <https://oxygen3000.com/> [1]

The peculiarity of the product is its use not only for rehabilitation after coronavirus disease. Oxygen “OXYGEN 3000” is used

- for to relieve the symptoms of hypoxia,
- for the prevention of respiratory diseases.
- helps to relieve emotional stress,
- stimulates mental activity,
- increases concentration,
- enhances brain activity,
- increases the overall tone of the body;
- helps to overcome stress, insomnia, weather dependence,
- significantly improves the condition of the skin.

Retail trade is a type of activity for the sale of goods and services directly to end users for their personal use.

The main goal of each organization is to meet the needs of the consumer market. As competition grows, so do consumers' demands for services and products. This is all motivation for the company to make its product and services better. Based on this, it is understood that meeting the needs of consumers is paramount, so you should also pay due attention to the business processes of the enterprise, paying attention not only to production but also to product sales. The existing system operates at the basic level of business processes.

The client leaves a request for an order on the site or trading platform, the manager calls back, clarifies the details of the order and is engaged in the registration of the parcel for sending by new mail.

The manager works with the mail service manually, and spends a lot of time entering information into the information system of the delivery service.

The current system loads the manager, and as a result, delays sending parcels to customers. The development of automation of the postal process will increase the speed of processing applications, which will lead to increased turnover, increased loyalty to the company and image growth.

For a higher level, you need to automate the registration of postage; to do this, you need to use the integration with the Nova Poshta API.

API (Application Programming Interface) is a set of tools for automating work with Nova Poshta. The API functionality allows you to quickly integrate logistics processes into any business and is the only entry point for all customers and services [2].

1.2 Characteristics of existing solutions for automation of postal items

Various transport organizations of international and local importance operate on the territory of Ukraine. Local carriers include private companies and the state service Ukrposhta [3]. International companies mainly cooperate with national operators, but they can also operate through their own representative branches.

In the segment of Ukrainian e-commerce, where logistics issues are of particular importance, the following carriers are popular:

1. Nova Poshta – 97%.
2. Ukrposhta – 33%.
3. Intime – 24% [4].
4. Delivery – 18% [5].
5. Mist Express – 8% [6].
6. Autolux – 4% [7].
7. Zruchna – 3% [8].
8. Delfast – 0,8% [8].

The percentage next to the name of the organization is the approximate number of online entrepreneurs that are working with it.

Up to 40% of online stores offer their own courier services, providing express delivery of goods around the city. Maintaining your own courier service is quite expensive. But it allows you to keep under full control the efficiency and quality of delivery, which affect customer satisfaction and their further desire to buy in the store. In other cases, it all depends on the characteristics of the company delivering the parcel.

Delivery services have specific billing features. For all economic entities, the first and one of the highest priority issues when choosing a delivery company is pricing. Each organization has its own prices. They are calculated individually depending on the distance and route of dispatch, weight, dimensions, declared value of the parcel, type of packaging.

Initial price tags of services (box up to 1 kilogram):

- Nova Poshta – from 45 hryvnia;
- Ukrposhta – from 18 hryvnia;
- Intime – from 30 hryvnia;
- Delivery – from 20 hryvnia;
- Mist Express – from 25 hryvnia;
- Autolux – from 25 hryvnia.

In direct mailing services, the pricing system is different. In Zruchna, rates start at 60 hryvnia. Delfast operates at a single fixed rate, which currently amounts to 170 hryvnia.

The second criterion is the speed of sending. On average, services deliver orders across Ukraine in such terms:

- Nova Poshta – 1-2 days;
- Ukrposhta – 4-6 days;
- Intime – 1-2 days;
- Delivery – 1-3 days;
- Mist Express – 1-3 days;
- Autolux – 1-3 days.

Express service Zruchna delivers goods within 1 day, Delfast – in a few hours.

Parcels from Ukrposhta arrives the slowest. Although this company also has an express delivery service, which is invested in terms of 2-3 days. The most efficient work is the Nova Poshta.

The coverage area of the logistics company determines the settlements available for service. The wider it is, the more efficiently Internet sales are made.

Online stores cooperate with shipping organizations with the largest number of branches in the country. The leader by this criterion is the national operator Ukrposhta. In second place – Nova Poshta, then – Mist Express, Deliveri and Intime. The smallest coverage area is at Autolux. Targeted express delivery services are represented by several branches in Kiev.

Analyzing all the advantages and disadvantages of postal services, it is necessary to take into account the wishes of customers. NovaPoshta is the most popular among “Oxygen3000” customers.

The considered operators can boast of a decent level of manufacturability. Each of them has an API and a modern mobile application (with the exception of the Zruchna service).

The API integrates the capabilities of the service into the website of the online store. For example, such:

- registration and receipt of a TTN number;
- tracking cargo tracking according to the receipt;
- calculation of the cost of shipments;
- displaying background information on the coverage area of the service.

The software product provided by NovaPoshta (Figure 1.1 and 1.2) is easy to use and allows you to use the service to the fullest. The forms of the program are shown in the figures. However, given the peculiarity of the company, all the functionality provided by Nova Poshta is unnecessary. The company sells one type of product with certain dimensions. The sender in the company is always the same. In addition, to create an invoice, you must manually enter this data each time the manager.

[Створити ЕН](#) [ЕН](#) [Реєстри](#) [gedBOX](#) [Фрахт](#) [Міжнародна доставка](#) [Документація](#) [Налаштування](#) [Ваші дані](#) [Доступна знижка: 0.00 грн](#)

[NOVAPOSHTA.UA](#) / [Головна](#) / [Список ЕН](#) / [Створити ЕН](#)

Натисніть для вибору ВІДПРАВНИКА

Місто:
 Назва (або П. І. Б) Ві...
 Адреса:
 Контактна особа:
 Телефон:

Натисніть для вибору ОДЕРЖУВАЧА

Місто:
 Назва (або П. І. Б) О...
 Адреса:
 Контактна особа:
 Телефон:

Параметри відправлення

Посилка

Параметри кожного місця відправлення

Штрих-код gedBOX:

Загальна вага * кг.

Вага об'ємна (Ширина x Довжина x Висота/4000) кг.

Загальний об'єм відправлення* м³.

Кількість місць * шт.

Оголошена вартість, грн* грн.

Опис відправлення *

Номер пакування

Внутрішній номер замовлення Клієнта

Figure 1.1 – View of the invoice creation window in the Nova Poshta service [10]

Параметри кожного місця відправлення ×

№ н/п	Об'єм	Ширина	Довжина	Висота	Вага фактична	Вага об'ємна	Ручна обробка
1	<input type="text"/> м³ або <input type="text"/>	<input type="text"/> см.	<input type="text"/> см.	<input type="text"/> см.	<input type="text"/> кг.	<input type="text"/> кг.	<input type="checkbox"/> <input type="button" value="☰"/> <input type="button" value="✕"/> <input type="button" value="🗑"/>
+	<input type="text"/> м³ або <input type="text"/>	<input type="text"/> см.	<input type="text"/> см.	<input type="text"/> см.	<input type="text"/> кг.	<input type="text"/> кг.	<input type="checkbox"/> <input type="button" value="☰"/> <input type="button" value="✕"/> <input type="button" value="🗑"/>

Figure 1.2 – View of the window describing the characteristics of the shipped goods [10]

Given the above, it is advisable to integrate on your own website a system for dialogue with the mail service through the API. Implementation of such a system will allow you to flexibly adjust the operation of the program.

Nova Poshta provides access to the following services via the API [11]:

Working with addresses:

- Online search in the directory of settlements
- Online street search in the directory of settlements
- Create counterparty address (sender / recipient)
- Edit counterparty address (sender / recipient)
- Delete counterparty address (sender / recipient)
- Company Cities Directory
- Directory of settlements of Ukraine
- Directory of geographic regions of Ukraine
- Directory of departments and types of departments
- Company Directory
- Working with Counterparty data

- Create Counterparty
- Create a contact person of the Counterparty
- Create a Counterparty with the type (legal entity) organization
- Create a third-party counterparty
- Download the list of addresses
- Load Counterparty parameters
- Download the list of contact persons of the Counterparty
- Download the list of Consignors / consignees / third parties
- Update Counterparty details
- Update the details of the contact person of the Counterparty
- Delete the recipient's business partner
- Delete Contact face of the Counterparty

API Print Forms:

- Markings – printed form
- Registers – printable
- Express waybill – printed forms
- Working with express waybill registries:
 - Add express waybills
 - Download information on one registry
 - Download the list of registries
 - Delete (disband) registries
 - Delete express invoices from the register

Working with directories:

- Types of time intervals
- Types of cargo
- Types of return shipping
- Pallet types

- Payer types
- Types of return shipping payers
- Types of packaging
- Types of tires and disks
- Description of cargo
- List of errors
- Delivery technologies
- Types of counterparties
- Forms of payment
- Forms of ownership
- API Service return shipment. Implements the Possibility of the Client's self-registration of the “Return of consignment” service when using the API:
 - Check if you can create a return request
 - Getting a list of reasons for return
 - Getting a list of subtypes and reasons for a return
 - Create a return request
 - Retrieving a list of return requests
 - Deleting a return request

API Service Change data.

Implements the possibility of the Client's self-registration of the “Change data” service when using the API:

- Checking the possibility of creating a request for data change
- Creation of a data change request
- Deleting an order
- Receiving a list of applications

API Service forwarding shipment. Implements the possibility of the Client's self-registration of the Call Forwarding service when using the API:

- Checking the possibility of creating a request for forwarding a dispatch

- Create a request forwarding a dispatch (branch / address)
- Deleting an order
- Receiving a list of applications

Working with express invoices:

- Calculate the cost of services
- Delivery date forecast
- Create Express Invoice
- Create an express invoice to the address
- Create an express invoice for a branch
- Create an express waybill for the “Nova Poshta” parcel machine
- Create Express Return Shipping Invoice
- Edit express invoice
- Tracking
- Get EN List
- Delete express waybill
- Formation of a request to receive a full report on invoices
- Formation of requests for the creation of EN with additional services
- Formation of requests for the creation of electronic devices with various types of cargo

1.3. Formation of system requirements

Placing an order, namely delivery, is the main business process associated with the management of postal items.

The system is must:

1. Collect the information from the customer needed to send the parcel:
 - Full name;
 - Telephone number;
 - your city of residence;

- choice of payment (online / cash on delivery);
- choice of delivery (self-pickup from the branch of the NP / courier of the NP);
- comment (optional, optional).

This will form the customer's card.

By creating a customer card, the process of creating a consignment note will be automated.

2. Form a consignment note
3. Keep track of how much time the customer has left to pick up the item.
4. Make an auto-return of delivery, if the customer did not pick up the parcel or refused it.

The mail module must be integrated into the company's general website. The module must be integrated with the Nova Poshta API. To use this function, you need to generate an APIkey in your personal account, which will be used when generating requests.

With the help of the API, the creation of the TTN will be automated, the registration of auto returns after the expiration of the specified period.

SECTION 2 IMPLEMENTATION OF THE PROTOTYPE OF THE POSTAL MANAGEMENT MODULE

2.1 Designing the architecture of the prototype of the postal management module

The basis of the program is a dialogue with the API Nova Poshta. The main process that implements the management of postal items is the automatic formation of the express invoice. To create an online express invoice document, you need to perform the following steps:

- Specify sender data
- Specify recipient details
- Select branch and delivery address
- Determine the weight and size of the shipment
- Print / save the document

The conceptual scheme of the module is shown in Figure 2.1

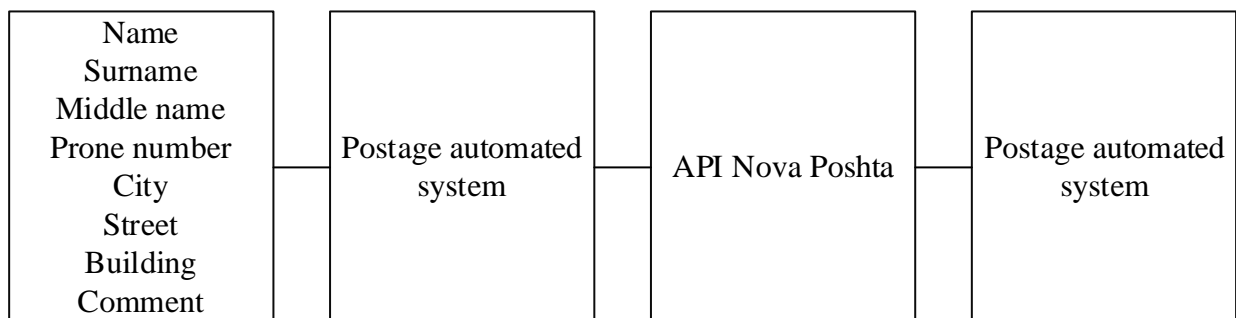


Figure 2.1 – Conceptual scheme of operation of the mail management module

After entering the information, the client forms a table with orders, with which the company's manager works in the future.

The manager needs to call the client, clarify the order. In case of confirmation – the manager creates an invoice, prints it, forms a parcel and takes the appropriate order to the post office.

Consider in more detail the algorithm for creating an Internet document of the consignment note (Figure 2.2).

At the input, the program module receives input data. Then the sender object is created in the Nova Poshta system via the API. The result is an object reference identifier. The next step is to obtain a list of sender's contacts. The result of the stage is an identifier-link to the contact person, his name and phone number.

After creating and obtaining the necessary information about the sender, you need to determine whether the client is registered in the Nova Poshta system. To do this, a search is made in the list of contractors, and if such a person exists, then we get a link ID from the list. If there is no such person – then we create a new recipient. In the next step we get a list of contact persons of the recipient, or register a contact person, in case of creating a new recipient.

The following three blocks are sent to search in the lists of addresses: cities, streets, houses, apartments of the recipient.

Then there is the creation of an Internet document of the consignment note and its printing [12].

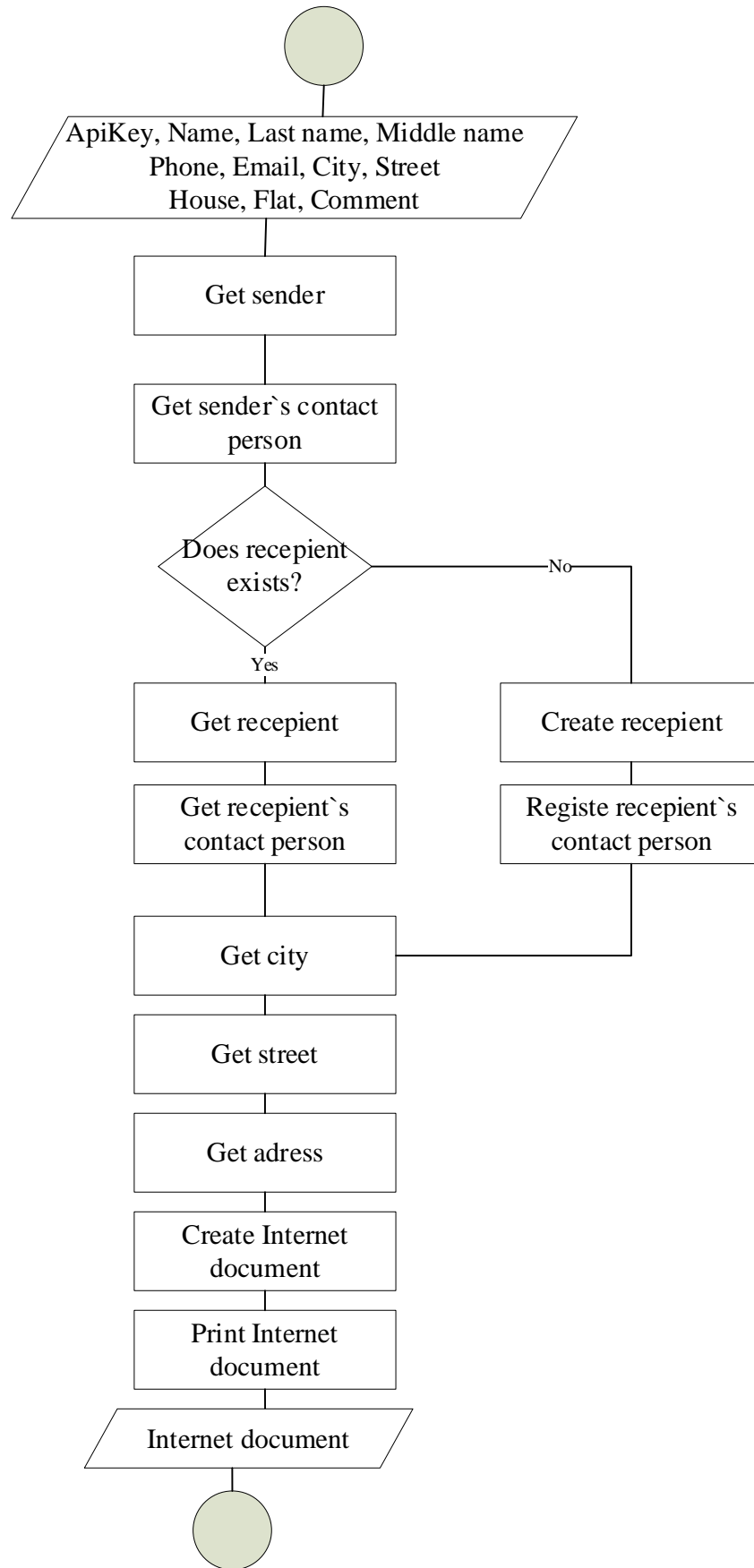


Figure 2.2 – Block diagram of the algorithm for creating an Internet document of the consignment note

The next part of the software module is tracking the status of the shipment.

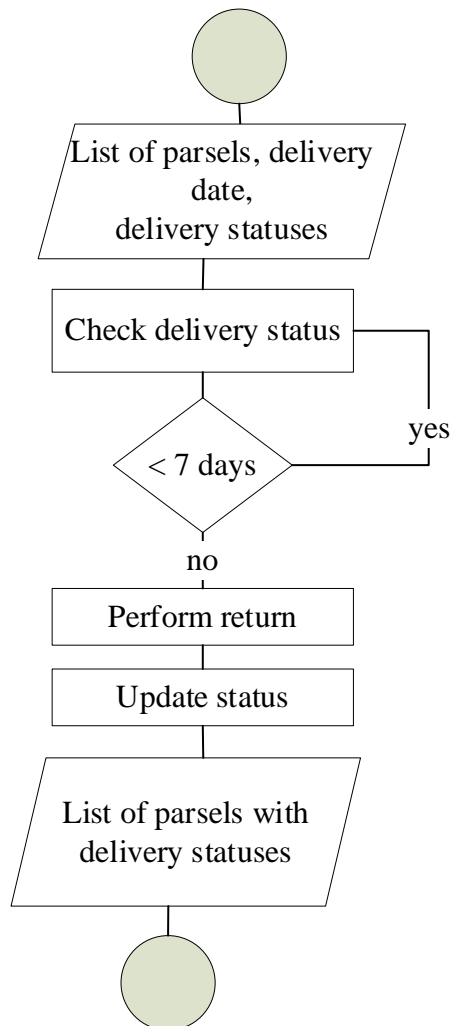


Figure 2.3 – Block diagram of the algorithm for tracking the status of the parcel and registration of auto-return of the parcel

Considering the algorithm for tracking the status of parcels, we note that the login includes a list of items marked with the date of departure, date of arrival at the branch, the date when the parcel must be picked up by the customer and the status of the parcel. Statuses can contain information about the location of the parcel and the approximate date of delivery, a mark of arrival at the branch, a mark of receipt of the shipment, a change of address, the date of receipt of the shipment, the termination of free storage [13].

The program must determine whether the term of free storage of the parcel has been exceeded (up to 7 days), and in case of exceeding this term – registration of auto-return.

To ensure the storage of order and delivery information, you need to create a database table that will store all the information. The structure of the table is described in table 2.1.

Table 2.1 – The structure of the database table [14]

№	Name	datatype	Null	Additional	Purpose
1	id	int(10)	not null	auto increment	primary key
2	name	char(100)	not null	–	Client`s name
3	surname	char(100)	not null	–	Client`s surname
4	middleName	char(100)	not null	–	Client`s middleName
5	phone	char(100)	not null	–	Client`s phone
6	eMail	char(100)	null	–	Client`s eMail
7	city	char(100)	Null	–	Client`s city
8	street	char(100)	null	–	Client`s street
9	building	char(100)	null	–	Client`s building
10	apartment	char(100)	null	–	Client`s apartment
11	purchaseId	char(100)	null	–	Number of purchase
12	status	char(100)	null	–	Order status
13	ttn	char(100)	null	–	Order invoice reference
14	dateSent	DateTime	null	–	Date mail sent
15	deliveryDate	DateTime	null	–	Date mail delivered
16	deliveryStatus	char(100)	null	–	Delivery status

2.2 Selecting of implementation technology

The prototype is a module of an existing web-based information system. Based on the analysis of existing solutions for automation of mail management, it would be wise to choose the client-server architecture used by web applications. This architecture will allow you to process operations directly on the server, which will ensure the efficiency and security of the application.

The most common and advanced client-server architecture is three-tier. The architecture model is divided into three parts: client, server, and database server.

The first level (thin client) – the client which is presented by a graphic component and provides dialogue of the user with system. Users in the designed system are employees of the financial monitoring department of the bank.

Important for the first level is the functionality of the user when interacting with the system. It is necessary to clearly regulate the operations that can be performed by the user, what data can be entered and what the user sees as a result of the system.

The second level is represented by the application server (application server) – at this level is stored in compliance with business rules. Here is all the logic of the program. The server provides data processing from the client, generating queries to the database, processing the received data and presenting them to the user.

The third level is a database server that provides data storage, including their consistent conversion and protection against unauthorized adjustments. Also at this level is the function of data protection and backup [19].

Among the technologies that provide a three-tier architecture, we will focus on the Django framework of the Python programming language.

The advantages of development on Django include the principle of “All Inclusive” (“Batteries included”). The phrase “all-inclusive” means that most of the tools for creating a program are part of the framework, and do not come as separate libraries.

Django contains a huge amount of functionality to solve most web development tasks.

Django supports:

- Object Related Mapping;
- Database migrations;
- User authentication;
- Admin panel;
- Forms;
- Standardized structure.

Django as a framework sets the structure of the project. It helps developers understand where and how to add new functionality [15].

Thanks to the same structure for all projects, it is much easier to find ready-made solutions or get help from the community. A huge number of enthusiastic developers will help to cope with any task much faster [16].

Django applications allow you to divide a project into several parts. Add-ons are installed by adding to `settings.INSTALLED_APPS`. This approach makes it easy to integrate ready-made solutions [17].

Hundreds of universal modules and applications will greatly accelerate development.

For coding is better to use Visual Studio Code software [20]. The basic version of Python – 3.9.5 [21]

2.3 Developing the prototype of the postal management module

Prototype development begins with setting up the environment, creating a project and related files.

Figure 2.3 shows the structure of the Django project. The migrations folder stores a description of the changes made to the database. The “templates” folder contains a list of templates used by the module. Files “forms.py”, “models.py”, “urls.py”, “views.py” contain classes and functions that describe forms, database table, list of pages and binding handlers to them, handler functions pages respectively [18].

The “settings.py” file includes the general settings of the project. File “db.sqlite3” – saves the database. The file “manage.py” is responsible for starting the project [22].

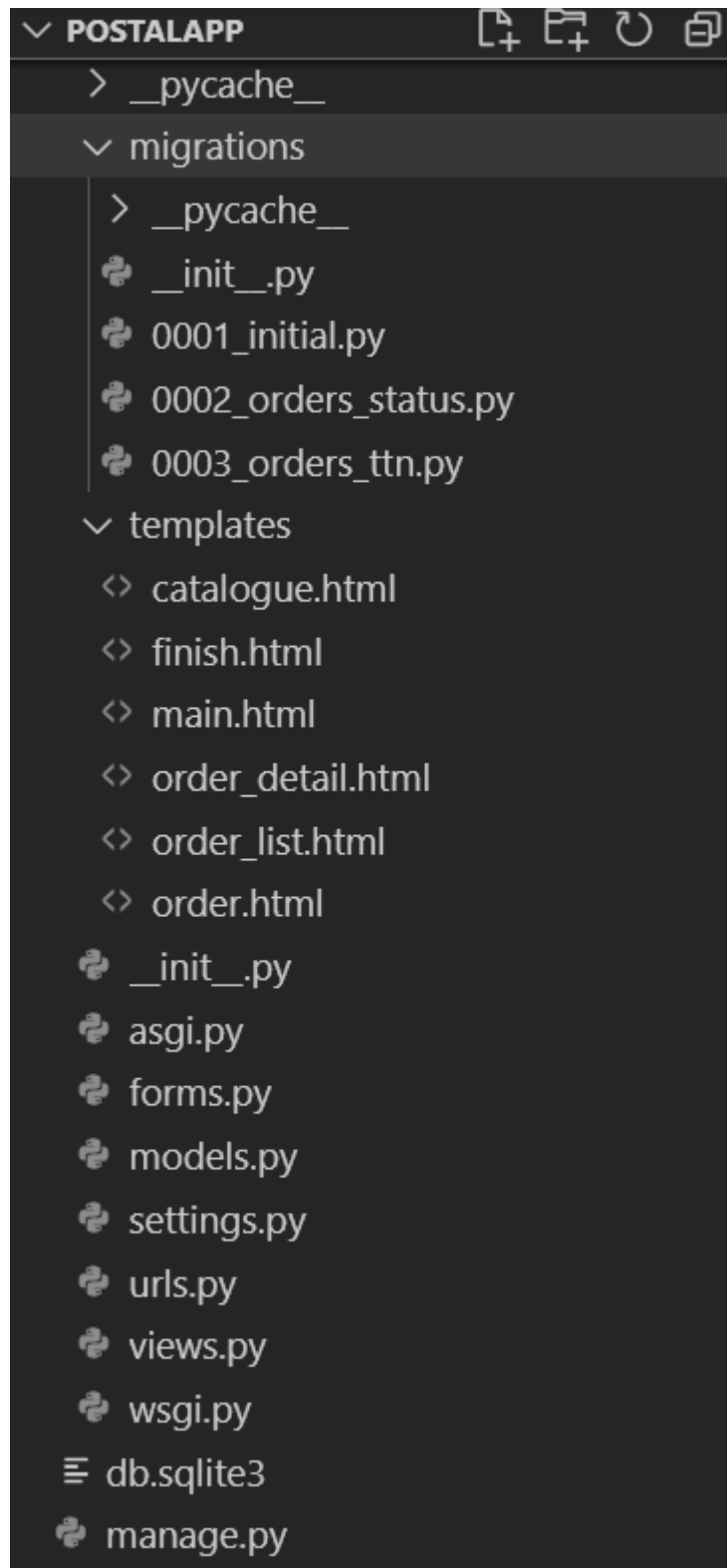


Figure 2.4 – Project structure

First we configure the pages we need. Figure 2.5 shows the pages involved in the project.

```

6  urlpatterns = [
7      path('admin/', admin.site.urls),
8      url(r'^$', postalApp.views.index, name='index'),
9      url(r'^order$', postalApp.views.order, name='order'),
10     url(r'^catalogue$', postalApp.views.catalogue, name='catalogue'),
11     url(r'^finish$', postalApp.views.finish, name='finish'),
12     url(r'^manager$', postalApp.views.manager, name='manager'),
13 ]

```

Figure 2.5 – Project pages

Pages “index”, “order”, “catalog”, “finish” – service pages, which in the prototype of the system are responsible for integration with the general website. Page “manager” – contains a list of orders and all information about orders and shipments.

The next step is to create a database. In the file “models.py” we define with the help of the Orders class the corresponding structure of the database [24]. The appearance of the file “models.py” is shown in Figure 2.6.

```

postalApp > models.py > ...
1  from django.db import models
2  class Orders(models.Model):
3      name = models.CharField(max_length=100)
4      surname = models.CharField(max_length=100)
5      middleName = models.CharField(max_length=100)
6      phone = models.CharField(max_length=100)
7      eMail = models.CharField(max_length=100)
8      city = models.CharField(max_length=100)
9      street = models.CharField(max_length=100, null=True)
10     building = models.CharField(max_length=100, null=True)
11     apartment = models.CharField(max_length=100, null=True)
12     comment = models.CharField(max_length=100, null=True)
13     purchaseId = models.CharField(max_length=100, null=True)
14     status = models.CharField(max_length=10, null=True)
15     ttn = models.CharField(max_length=100, null=True)
16     dateSent = models.DateTimeField(null=True)
17     deliveryDate = models.DateTimeField(null=True)
18     deliveryStatus = models.CharField(max_length=100, null=True)

```

Figure 2.6 – Database structure

In Django, the database is implemented by creating classes that are descendants of the ancestor class “models.Model”. Table columns are attributes of a class. Class attributes are defined using appropriate methods, according to data types: CharField () – for text fields, DateTimeField () – for fields that contain time and date information. The “id” field, which acts as the primary key and has the “autoincrement” property, is created automatically by the framework. You can use method attributes to control the validity of empty values, set the maximum field length, or set default values [25].

The interface with customers is supported by interactive forms. In Django the forms are placed in the file “forms.py” in the form of classes. Classes are created as descendants of the “forms.Form” class, in which attributes, by analogy with models, are methods that return objects. Method attributes specify the maximum field length, field name, interactive widget (such as dropdownlist), or hidden fields. An example of a class for creating a shape is shown in Figure 2.7.

```

postalApp > forms.py > Order
1  from django import forms
2  PAYMENT_CHOICES= [
3      ('online', 'Online'),
4      ('cash', 'Cash')
5  ]
6  DELIVERY_CHOICES= [
7      ('self', 'Self'),
8      ('courier', 'Courier')
9  ]
10 class Order(forms.Form):
11     name = forms.CharField(label='name', max_length=100)
12     surname = forms.CharField(label='surname', max_length=100)
13     phone = forms.CharField(label='phone', max_length=100)
14     payment = forms.CharField(label='payment', widget=forms.Select(choices=PAYMENT_CHOICES))
15     delivery = forms.CharField(label='delivery', widget=forms.Select(choices=DELIVERY_CHOICES))
16     city = forms.CharField(label='city', max_length=100)
17     postbr = forms.CharField(label='postbr', max_length=100)
18     street = forms.CharField(label='street', max_length=100)
19     building = forms.CharField(label='building', max_length=100)
20     apartment = forms.CharField(label='apartment', max_length=100)
21     comment = forms.CharField(label='Add comment', max_length=100)
22     purchaseId = forms.CharField(widget=forms.HiddenInput())
23

```

Figure 2.7 – Form for creating an order

The form is integrated into the html template using the Order class variable and the csrf_token certificate, which is responsible for the security of the transmitted data. Integration into the html-template is shown in Figure 2.8.

```
<form method="post" action="/finish">
  {% csrf_token %}
  {{ form.non_field_errors }}
  <table>
    <tr>
      <td>
        {{ form.name.errors }}
        <label for="{ { form.name.id_for_label } }">Name:</label>
      </td>
      <td>
        {{ form.name }}
      </td>
    </tr>
  </table>
</form>
```

Figure 2.8 – Integration of the form into the html-template

The views.py file contains site page handlers. To implement the required functionality, you must use the following modules of the Django framework and the actual Python language [35, 36, 37, 38, 39, 40]:

- from django.shortcuts import render
- from django.http import HttpResponse
- from django.contrib.auth import authenticate
- from django.http import HttpResponseRedirect
- from django.conf import settings
- from datetime import datetime, timedelta
- import http.client
- import requests
- import json
- from .forms import DeleteOrder, DeclineOrder, Order
- from .forms import changeStatus

- from .forms import Product
- from .models import Orders

Modules “render”, “HttpResponse”, “HttpResponseRedirect”, “authenticate”, “settings”, “http.client”, “requests” – are responsible for the operation of the project, displaying information on the interface, data transfer through variables in the template and receiving http requests . Modules “datetime”, “timedelta” – necessary for working with timers and date. Json module – implements the ability to transfer data to the API and receive data from the API in json format. Modules “DeleteOrder”, “DeclineOrder”, “Order”, “ChangeStatus”, “Product”, “Orders” – connect the appropriate forms and models to the file “views.py” [23, 32].

Processing of the order form is implemented by the finish (request) method. This method takes as an argument the request received from the html-form [33, 34].

The method has the following structure:

```
def finish(request):
    if request.method == 'POST':
        form = Order(request.POST)
        if form.is_valid():
            new_order = Orders(name = form.cleaned_data['name'],
                               surname = form.cleaned_data['surname'],
                               phone = form.cleaned_data['phone'],
                               payment = form.cleaned_data['payment'],
                               delivery = form.cleaned_data['delivery'],
                               city = form.cleaned_data['city'],
                               postbr = form.cleaned_data['postbr'],
                               street = form.cleaned_data['street'],
                               building = form.cleaned_data['building'],
                               apartment = form.cleaned_data['apartment'],
                               comment = form.cleaned_data['comment'],
                               purchaseId = form.cleaned_data['purchaseId'],
                               status = '1' )
            new_order.save()
            return render(request, 'finish.html', {'Thank': "Thanks \n our manager will contact you" })
        else:
            form = Order()
    return render(request, 'order.html', {'form': form, 'id': 'vvv'})
```

Code Listing 2.1 – Code method for processing the form and writing order information to the database

In addition to filling in the fields of the database table obtained from the form, the status is set to – 1, which corresponds to the new order [31].

The dialogue with the API is implemented using the `post()` methods of the “requests” module and `loads()` [26] of the “json” module.

Figure 2.9 shows an example of a Counterparty query to create a new sender object.

```

272     countr = {
273         "apiKey": "aa4e0992ea5446acbef2a12f8a4328b4",
274         "modelName": "Counterparty",
275         "calledMethod": "getCounterparties",
276         "methodProperties": {
277             "CounterpartyProperty": "Sender",
278             "Page": "1"
279         }
280     }
281     response = requests.post('https://api.novaposhta.ua/v2.0/json/', json=countr)
282     content = response.content
283     countr_full = json.loads(content)
284     countr_ref = countr_full['data'][0]['Ref']

```

Figure 2.9 – Example of a query to the Counterparty model

Lines 272-288 form the request body in json format. Line 273 contains the `apiKey` generated for the company by the Nova Poshta system and identifies it. Line 281 generates a request to the API method `post()`. Line 282 receives the content of the request. Then the content is transformed into a dictionary from the json format. The last step is to get the Ref ID of the entire response array [27, 30].

After making a number of such requests (Appendix B), the following request is generated (Listing 2.2).

```

params1={
    "apiKey": "aa4e0992ea5446acbef2a12f8a4328b4",
    "modelName": "InternetDocument",
    "calledMethod": "save",
    "methodProperties": {
        "PayerType": "Sender",
        "PaymentMethod": "Cash",
        "DateTime": date,
        "CargoType": "Cargo",

```

```

"VolumeGeneral": "0.1",
"Weight": "10",
"ServiceType": "WarehouseDoors",
"SeatsAmount": "1",
"Description": "Oxygen3000",
"Cost": "500",
"CitySender": "8d5a980d-391c-11dd-90d9-001a92567626",
"Sender": countr_ref,
"SenderAddress": "01ae2635-e1c2-11e3-8c4a-0050568002cf",
"ContactSender": countr_cont_full_ref,
"SendersPhone": countr_cont_full_phones,
"CityRecipient": get_city_full_ref,
"Recipient": new_receipient_full_ref,
"RecipientAddress": get_adress_ref,
"ContactRecipient": get_cont_receipient_full_ref,
"RecipientsPhone": get_cont_receipient_full_phones
}
}

```

Code Listing 2.2 – Request for the formation of the consignment note [28]

This request indicates the date of creation, type of delivery, type of goods, identifiers of the sender, recipient, contact persons of the sender and recipient, delivery address, telephone numbers of contact persons.

The result of the request – created identifier express invoice. Next, the specified ID is added to the string [https://my.novaposhta.ua/orders/printDocument/orders\[/a01e801c-cef0-11eb-8513-b88303659df5/orders\[/a01e801c-cef0-11eb-8513-b88303659df5/type/pdf/apiKey/aa4e0992ea5446acbef2a12f8a4328b4](https://my.novaposhta.ua/orders/printDocument/orders[/a01e801c-cef0-11eb-8513-b88303659df5/orders[/a01e801c-cef0-11eb-8513-b88303659df5/type/pdf/apiKey/aa4e0992ea5446acbef2a12f8a4328b4), after which it can be printed.

Then all the information is recorded in the database and transmitted to the manager interface using function manager (response). The program code is given in appendix B [29].

2.4 Reference example and instructions for use

On the company's website, the customer adds the product to the cart, fills out the form with their contact information. View of the form in the Figure 2.10.

The screenshot shows the Oxygen 3000 website interface. At the top, there is a navigation menu with links for 'Головна', 'Корисні властивості', 'Новини', and 'Магазин'. The 'oxygen 3000' logo is prominently displayed, along with a shopping cart icon labeled 'Кошик (1)'. Below the navigation, there is a 'Назад' button. The main content area is titled 'Платіжна інформація' and contains several input fields for contact information: 'Електронна пошта', 'Номер телефону', 'Ім'я' and 'Прізвище' (split into two fields), 'По-батькові', 'Вулиця', 'Будинок', 'Місто' and 'Індекс' (split into two fields). A green button labeled 'В наявності!' is visible. Below the form, there is a section for 'Інформація' with a checked checkbox 'Так само, як' and a 'Коментарі замовлення' field. At the bottom, a small text line reads: 'Кисень газоподібний для дихання «OXYGEN 3000», 8 літрів.'

Figure 2.10 – Form for filling in delivery information

After filling out the form, the client displays the message “Thanks, our manager will contact you”.

In his personal account, the manager sees a list of orders and detailed information about the shipment, shown in Figure 2.11

As can be seen in Figure 2.11, columns 1-5 are filled in by the customer when ordering. The manager receives a list of orders, new orders are displayed

marked “New” in the column “status”. Order information with its quantity is displayed in the “purchase Id” column.

name	surname	middleName	phone	Adress	purchase Id	status	Approve	Decline	Delete	TTN	DateSent	DateReceived	Delivery Status
Yulia	Lebedeva	Vitaliivna	0505656569	Харків, Сумська, 7 кв. 2	1	Declined	<input type="button" value="Approve"/>	<input type="button" value="Decline"/>	<input type="button" value="Delete"/>				
Юлія	Лебедева	Віталіївна	0505656569	Харків, Сумська, 7 кв. 2	2	Approved	<input type="button" value="Approve"/>	<input type="button" value="Decline"/>	<input type="button" value="Delete"/>	open ttn			
Юлія	Лебедева	Віталіївна	0505656569	Харків, Сумська, 7 кв. 2	3	Approved	<input type="button" value="Approve"/>	<input type="button" value="Decline"/>	<input type="button" value="Delete"/>	open ttn			
Юлія	Лебедева	Віталіївна	0505656569	Харків, Сумська, 7 кв. 2	4	New	<input type="button" value="Approve"/>	<input type="button" value="Decline"/>	<input type="button" value="Delete"/>				

Figure 2.11 – View of the List of orders on the manager page

After the manager confirms the order from the customer, he clicks on the Approve button and changes the status of the order to “Approved”. In case of order cancellation – the manager can click “Decline” and cancel it, the status changes to “Declined”. If necessary, the “Delete” button deletes the order from the list.





When you click the “Approved” button, an Internet document is automatically generated – a consignment note, which becomes available for printing by following the link in the “TTN” column.

The view of the consignment note is shown in Figure 2.12.

After creating an Internet document of the consignment note, the manager prints it out and forms a parcel with the goods. After that, all he has to do is take the parcels to the Nova Poshta branch and send them.

After sending parcels using a request to the API, the program will update the date of departure (12th column of Figure 2.11) and change the delivery status (14th column of Figure 2.11).

The program tracks the storage time of 7 days from the date of delivery, and if such time has expired, the status in the 14th column changes to “Return”. In the implemented prototype, the data cells are empty, because the APIN New Mail works in real time, and the prototype needs to run and send real parcels to demonstrate the appropriate functionality.

ЕКСПРЕС-НАКЛАДНА № 20450402626632 <small>17.06.2021 01:17:23</small>					
Інформація про Відправника Місто відправник: <input type="text" value="Харків"/> Назва фірми або П.І.Б. приватної особи: <input type="text" value="Приватна особа"/> Адреса (звідки забирається відправлення): <input type="text"/> Телефон: <input type="text"/> П.І.Б. контактної особи (повністю): <input type="text"/>		Інформація про відправлення Кількість місць: <input type="text" value="1"/> шт. Фактична вага: <input type="text" value="10"/> кг. Об'ємна вага: <input type="text" value="25"/> кг. Контрольна вага: <input type="text"/> Оголошена вартість: <input type="text" value="500"/> грн. Повний опис відправлення: <input type="text" value="Абажур"/> Документи, що супроводжують відправлення: <input type="text"/>		НОВА ПОШТА www.novaposhta.ua 0 - 8 0 0 - 5 0 0 - 6 0 9	
Регіон-одержувач Харків Посилковий		№ району X02/301		Додаткові послуги Номер замовлення: <input type="text"/> Додаткова інформація про відправлення: <input type="text"/>	
Інформація про Одержувача Місто одержувач: <input type="text" value="Харків"/> Назва фірми або П.І.Б. приватної особи: <input type="text" value="Приватна особа"/> Адреса (куди доставляється відправлення): <input type="text" value="Сумська вул. 7 кв. 2 Коментарий"/> Телефон: <input type="text" value="380997979789"/> П.І.Б. контактної особи (повністю): <input type="text" value="Лебедева Юлія Віталіївна"/>		Сторона, що сплачує послуги <input checked="" type="checkbox"/> Відправник <input checked="" type="checkbox"/> Готівковий розрахунок Вартість доставки, грн.: <input type="text" value="138"/> (Сто тридцять вісім гривень 00 копійок) Підпис платника: <input type="text"/>		Представник Одержувача Дата: <input type="text"/> Час: <input type="text"/> Посада: <input type="text"/> П.І.Б. (повністю): <input type="text"/> Паспортні дані (для фізичних осіб): <input type="text"/> Підпис, печатка Одержувача: <input type="text"/>	
		Представник ТОВ "Нова Пошта"*** П.І.Б., Дата, підпис 20-06-2021 <input type="text"/>			
		Підпис Відправника, печатка <input type="text"/>			

З умовами Публічного договору про надання послуг з організації перевезення відправлень, які розміщено на сайті www.novaposhta.ua, ознайомлений

Figure 2.12 – View of the Internet document of the consignment note

CONCLUSIONS

The importance of process automation in our time is difficult to overestimate. The use of computer technology simplifies the functioning of both individual processes of the organization and the entire work of the company as a whole.

During the qualifying bachelor's thesis, the peculiarities of automation of the process of managing postal items at the enterprise were investigated. The analysis of existing solutions of automation of postal items is carried out. Based on the analysis, the basic requirements for the mail management system at the enterprise were identified and formed.

As a result, a prototype of the web-oriented automated mail management system module at Oxygen 3000 LLC was designed and implemented.

According to the tasks, there were:

- Oxygen 3000 LLC was characterized as an object of research and the state of business process automation was analyzed;
- the analysis of existing on the market of postal services and their decision of automation of registration of postal deliveries is carried out;
- formed requirements for web-based system;
- design of the architecture of the prototype module of postal subdivisions;
- selected technology prototype development: Django framework, Python programming language;
- a prototype of an automated system of postal corrections was developed;
- developed a control example with instructions for use.

Prospects for further automation activities of LLC “Oxygen 3000” are to expand the ability to pay online for the ordered goods. In addition, it is possible to upgrade an existing web system to meet modern requirements.

REFERENCES

1. Кисень для дихання: веб-сайт. URL: <https://oxygen3000.com/uk/> (дата звернення 10.06.2021)
2. Портал для розробників API Нова Пошта: веб-сайт. URL: <https://devcenter.novaposhta.ua/> (дата звернення 10.06.2021)
3. Укрпошта: веб-сайт. URL: <https://ukrposhta.ua/ua> (дата звернення 10.06.2021)
4. Логістична компанія Інтайм: веб-сайт. URL: <https://intime.check-track.com/ua/> (дата звернення 10.06.2021)
5. Група компаній «Делівері»: веб-сайт. URL: <https://www.delivery-auto.com/uk-UA/Home/Index> (дата звернення 10.06.2021)
6. Група компаній «Meest»: веб-сайт. URL: <https://ua.meest.com/> (дата звернення 10.06.2021)
7. Автолюкс Глобал Пост: веб-сайт. URL: <https://autolux-post.com.ua/> (дата звернення 10.06.2021)
8. ТОВ «Зручна Доставка»: веб-сайт. URL: <http://zruchnadostavka.com/> (дата звернення 10.06.2021)
9. Delfast: website. URL: <https://kiev.delfast.co/en/> (date accessed 10.06.2021)
10. Бізнес-кабінет Нова Пошта: веб-сайт. URL: <https://new.novaposhta.ua/create> (дата звернення 10.06.2021)
11. Документація API Нова Пошта: веб-сайт. URL: <https://devcenter.novaposhta.ua/docs/services/> (дата звернення 10.06.2021)
12. Zingaro D. Algorithmic thinking: a problem-based introduction. San Francisco: No starch press, 2021. 410 p. URL: <https://dokumen.pub/algorithmic-thinking-a-problem-based-introduction-1nbsped-9781718500808-9781718500815-1718500807.html> (date accessed 10.06.2021)

13. Erickson J. Algorithms, 2019. 472 p. URL: <https://jeffe.cs.illinois.edu/teaching/algorithms/book/Algorithms-JeffE.pdf> (date accessed 10.06.2021)
14. Kreibich J.A. Using SQLite. O`Reily press, 2010. 503 p. URL: <https://allitbooks.net/programming/5373-using-sqlite.html> (date accessed 10.06.2021)
15. Django documentation: website. URL: <https://docs.djangoproject.com/en/3.2/> (date accessed 10.06.2021)
16. Sedhain S. Web framework for Python: Django, 2006. 190 p. URL: <https://www.programmer-books.com/wp-content/uploads/2018/08/Django-Book-Web-framework-for-Python.pdf> (date accessed 10.06.2021)
17. Forcier J., Bissex P., Chun W. Python Web Development with Django. Pearson Education inc, 2009. 405 p. URL: <https://freepdf-books.com/download/?file=4536> (date accessed 10.06.2021)
18. Dauzon S., Bendoraitis A., Ravindran A. Django: Web Development with Python. Birmingham: Packt Publishing Ltd, 2016. 717 p. URL: [http://englishonlineclub.com/pdf/Django%20-%20Web%20Development%20with%20Python%20\(Learning%20Path\)%20\[EnglishOnlineClub.com\].pdf](http://englishonlineclub.com/pdf/Django%20-%20Web%20Development%20with%20Python%20(Learning%20Path)%20[EnglishOnlineClub.com].pdf) (date accessed 10.06.2021)
19. Three Level Architecture of Database: tuorialspoint website. URL: <https://www.tutorialspoint.com/Three-Level-Architecture-of-Database> (date accessed 10.06.2021)
20. Visual Studio Code: website. URL: <https://code.visualstudio.com/> (date accessed 10.06.2021)
21. Python: website. URL: <https://www.python.org/> (date accessed 10.06.2021)
22. Django: The web framework for perfections with deadlines: website. URL: <https://www.djangoproject.com/> (date accessed 10.06.2021)

23. Writing views: website. URL: <https://docs.djangoproject.com/en/3.2/topics/http/views/> (date accessed 10.06.2021)
24. Stratton J. Django: Beyond the SQL: website. URL: <https://www.pdf-tutorial.com/database/749-django-beyond-the-sql> (date accessed 10.06.2021)
25. Django Web Development in Python: website. URL: <https://www.datacamp.com/community/tutorials/web-development-django> (date accessed 10.06.2021)
26. McKinney W. Python for Data Analysis. O`Reilly, 2013. 470 p. URL: <https://bedford-computing.co.uk/learning/wp-content/uploads/2015/10/Python-for-Data-Analysis.pdf> (date accessed 10.06.2021)
27. Python Notes for Professionals: website. URL: <https://books.goalkicker.com/PythonBook/> (date accessed 10.06.2021)
28. Shaw Z. Learn Python the hard way: a very simple introduction to the terrifyingly beautiful world of computers and code. Crawfordsville: Addison-Wesley, 2014. 306 p. URL: <https://learntocodetogether.com/learn-python-the-hard-way-free-ebook-download/> (date accessed 10.06.2021)
29. McKinney W. Pandas: powerful Python data analysis toolkit, 2021. 3325 p. URL: <https://pandas.pydata.org/docs/pandas.pdf> (date accessed 10.06.2021)
30. Learning Django: free unaffiliated eBook created from Stack Overflow contributors. 228 p. URL: <https://riptutorial.com/Download/django.pdf> (date accessed 10.06.2021)
31. Документация Django: веб-сайт. URL: <https://djbook.ru/rel3.0/> (date accessed 10.06.2021)
32. Gorelick M., Ozsvald I. (2014) High Performance Python. O`Reilly Media, 370 p. ISBN: 978-1-449-36159-4
33. Lutz M. (2013) Learning Python, Fifth Edition. O`Reilly Media, 1594 p. ISBN: 978-1-449-35573-9

34. Лутц М. (2019) Изучаем Python, том 1, 5-е изд.: Пер. с англ. СПб.: ООО “Диалектика”, 2019. 832 с.
35. Downey A. (2016) Think Complexity. URL: <https://github.com/AllenDowney/ThinkComplexity2> (date accessed 10.06.2021)
36. Златопольский Д.М. Основы программирования на языке Python. – М.: ДМК Пресс, 2017. 284 с.
37. Лутц М. Программирование на Python, том I, 4-е издание. – Пер. с англ. СПб.: Символ-Плюс, 2011. 992 с.
38. Лутц М. Программирование на Python, том II, 4-е издание. – Пер. с англ. СПб.: Символ-Плюс, 2011. 992 с.
39. Requests: HTTP for Humans: website. URL: <https://docs.python-requests.org/en/master/> (date accessed 10.06.2021)
40. Basic date and time types: website. URL: <https://docs.python.org/3/library/datetime.html> (date accessed 10.06.2021)

APPENDICES

Appendix A

SUMMARY

Lebedeva Yu. V. Automation of the retail trade at the enterprise. Qualifying work of the bachelor. Sumy State University, Sumy, 2021.

The process of creating a prototype of an automated module for managing postal items at the enterprise is investigated. An analysis of postal services that provide their services in Ukraine. The prototype database was designed, algorithmic software was determined. A prototype of an automated mail management system at the enterprise has been implemented.

Keywords: automation, mailings, Nova Poshta API, programming, Django framework.

АНОТАЦІЯ

Лебедева Ю. В. Автоматизація поштових відправлень на підприємстві роздрібної торгівлі. Кваліфікаційна робота бакалавра. Сумський державний університет, Суми, 2021 р.

У роботі досліджено процес створення прототипу автоматизованого модулю управління поштовими відправленнями на підприємстві. Проведено аналіз поштових сервісів, які надають свої послуги в Україні. Було спроектовано базу даних прототипу, визначено алгоритмічне забезпечення. Реалізовано прототип автоматизованої системи управління поштовими відправленнями на підприємстві.

Ключові слова: автоматизація, поштові відправління, API Нова Пошта, програмування, Django фреймворк.

Appendix B

Program code

models.py

```

from django.db import models
class Orders(models.Model):
    name = models.CharField(max_length=100)
    surname = models.CharField(max_length=100)
    middleName = models.CharField(max_length=100)
    phone = models.CharField(max_length=100)
    eMail = models.CharField(max_length=100)
    city = models.CharField(max_length=100)
    street = models.CharField(max_length=100, null=True)
    building = models.CharField(max_length=100, null=True)
    apartment = models.CharField(max_length=100, null=True)
    comment = models.CharField(max_length=100, null=True)
    purchaseId = models.CharField(max_length=100, null=True)
    status = models.CharField(max_length=10, null=True)
    ttn = models.CharField(max_length=100, null=True)
    dateSent = models.DateTimeField(null=True)
    deliveryDate = models.DateTimeField(null=True)
    deliveryStatus = models.CharField(max_length=100, null=True)

```

forms.py

```

from django import forms
PAYMENT_CHOICES= [
    ('online', 'Online'),
    ('cash', 'Cash')
]
DELIVERY_CHOICES= [
    ('self', 'Self'),
    ('courier', 'Courier')
]
class Order(forms.Form):
    name = forms.CharField(label='name', max_length=100)
    surname = forms.CharField(label='surname', max_length=100)
    phone = forms.CharField(label='phone', max_length=100)
    payment = forms.CharField(label='payment', widget=forms.Select(choices=PAYMENT_CHOICES))
    delivery = forms.CharField(label='delivery', widget=forms.Select(choices=DELIVERY_CHOICES))
    city = forms.CharField(label='city', max_length=100)
    postbr = forms.CharField(label='postbr', max_length=100)
    street = forms.CharField(label='street', max_length=100)
    building = forms.CharField(label='building', max_length=100)
    apartment = forms.CharField(label='apartment', max_length=100)

```

```

    comment = forms.CharField(label='Add comment', max_length=100)
    purchaseId = forms.CharField(widget=forms.HiddenInput())
class Product(forms.Form):
    pr_id = forms.CharField(label='Product', max_length=100)
class changeStatus(forms.Form):
    approve = forms.CharField(widget=forms.HiddenInput())
class DeclineOrder(forms.Form):
    decline = forms.CharField(widget=forms.HiddenInput())
class DeleteOrder(forms.Form):
    delete = forms.CharField(widget=forms.HiddenInput())

```

urls.py

```

from django.contrib import admin
from django.urls import path
from django.conf.urls import include, url
import postalApp.views
urlpatterns = [
    path('admin/', admin.site.urls),
    url(r'^$', postalApp.views.index, name='index'),
    url(r'^order$', postalApp.views.order, name='order'),
    url(r'^catalogue$', postalApp.views.catalogue, name='catalogue'),
    url(r'^finish$', postalApp.views.finish, name='finish'),
    url(r'^manager$', postalApp.views.manager, name='manager'),
]

```

Settings.py

```

from pathlib import Path
# Build paths inside the project like this: BASE_DIR / 'subdir'.
BASE_DIR = Path(__file__).resolve().parent.parent
# Quick-start development settings - unsuitable for production
# See https://docs.djangoproject.com/en/3.2/howto/deployment/checklist/
# SECURITY WARNING: keep the secret key used in production secret!
SECRET_KEY = 'django-insecure-@+lu_ydf!z0k_$&0&ibmpioy-
&@u*$&@2#vp9#8_f&q&=8g&sw'
# SECURITY WARNING: don't run with debug turned on in production!
DEBUG = True
ALLOWED_HOSTS = []
# Application definition
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'postalApp',
]
MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',

```

```

'django.middleware.common.CommonMiddleware',
'django.middleware.csrf.CsrfViewMiddleware',
'django.contrib.auth.middleware.AuthenticationMiddleware',
'django.contrib.messages.middleware.MessageMiddleware',
'django.middleware.clickjacking.XFrameOptionsMiddleware',
]
ROOT_URLCONF = 'postalApp.urls'
TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
            ],
        },
    },
]
WSGI_APPLICATION = 'postalApp.wsgi.application'
# Database
# https://docs.djangoproject.com/en/3.2/ref/settings/#databases
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': BASE_DIR / 'db.sqlite3',
    }
}
# Password validation
# https://docs.djangoproject.com/en/3.2/ref/settings/#auth-password-validators
AUTH_PASSWORD_VALIDATORS = [
    {
        'NAME': 'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.MinimumLengthValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.CommonPasswordValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.NumericPasswordValidator',
    },
]
# Internationalization
# https://docs.djangoproject.com/en/3.2/topics/i18n/
LANGUAGE_CODE = 'en-us'
TIME_ZONE = 'UTC'

```

```

USE_I18N = True
USE_L10N = True
USE_TZ = True
# Static files (CSS, JavaScript, Images)
# https://docs.djangoproject.com/en/3.2/howto/static-files/
STATIC_URL = '/static/'
# Default primary key field type
# https://docs.djangoproject.com/en/3.2/ref/settings/#default-auto-field
DEFAULT_AUTO_FIELD = 'django.db.models.BigAutoField'

```

migrations

```

from django.db import migrations, models
class Migration(migrations.Migration):
    initial = True
    dependencies = [
    ]
    operations = [
        migrations.CreateModel(
            name='Orders',
            fields=[
                ('id', models.BigAutoField(auto_created=True, primary_key=True, serialize=False, verbose_name='ID')),
                ('name', models.CharField(max_length=100)),
                ('surname', models.CharField(max_length=100)),
                ('phone', models.CharField(max_length=100)),
                ('payment', models.CharField(max_length=100)),
                ('delivery', models.CharField(max_length=100)),
                ('city', models.CharField(max_length=100)),
                ('postbr', models.CharField(max_length=100, null=True)),
                ('street', models.CharField(max_length=100, null=True)),
                ('building', models.CharField(max_length=100, null=True)),
                ('apartment', models.CharField(max_length=100, null=True)),
                ('comment', models.CharField(max_length=100, null=True)),
                ('purchaseId', models.CharField(max_length=100, null=True)),
            ],
        ),
    ]
# Generated by Django 3.2 on 2021-06-13 22:01
from django.db import migrations, models
class Migration(migrations.Migration):
    dependencies = [
        ('postalApp', '0001_initial'),
    ]
    operations = [
        migrations.AddField(
            model_name='orders',
            name='status',
            field=models.CharField(max_length=10, null=True),
        ),
    ]
from django.db import migrations, models

```

```

class Migration(migrations.Migration):
    dependencies = [
        ('postalApp', '0002_orders_status'),
    ]
    operations = [
        migrations.AddField(
            model_name='orders',
            name='ttn',
            field=models.CharField(max_length=100, null=True),
        ),
    ]

```

order_list.html

```

<html>
<head><title></title></head>
<body>
<style>
    table {
        width: 100%; /* Ширина таблицы */
        border: 4px double black; /* Рамка вокруг таблицы */
        border-collapse: collapse; /* Отображать только одинарные линии */
    }
    th {
        text-align: left; /* Выравнивание по левому краю */
        background: #ccc; /* Цвет фона ячеек */
        padding: 5px; /* Поля вокруг содержимого ячеек */
        border: 1px solid black; /* Граница вокруг ячеек */
    }
    td {
        padding: 5px; /* Поля вокруг содержимого ячеек */
        border: 1px solid black; /* Граница вокруг ячеек */
        width:50px;
    }
</style>
<table>
    <tr >
        <td>name</td>
        <td>surname</td>
        <td>middleName</td>
        <td>phone</td>
        <td>Adress</td>
        <td>purchaseId</td>
        <td>status</td>
        <td>Approve</td>
        <td>Decline</td>
        <td>Delete</td>
        <td>TTN</td>
        <td>DateSent</td>
        <td>DateReceived</td>
        <td>Delivery Status</td>

```

```

</tr>
{% for order in content %}
<tr>
  <td>{{ order.name }}</td>
  <td>{{ order.surname }}</td>
  <td>{{ order.phone }}</td>
  <td>{{ order.city }}</td>
  <td>{{ order.purchaseId }}</td>
  <td>{% if order.status == "1"%}
    <div>New</div>
    {% elif order.status == "2"%}
    <div>Approved</div>
    {% elif order.status == "3"%}
    <div>Declined</div>
    {%endif%}

  </td>
  <td>
    <form method="post" action="/manager">
      {% csrf_token %}
      <input type="hidden" name="approve" value="{{ order.id }}">
      <input type="submit" value="Approve">
    </form>
  </td>
  <td>
    <form method="post" action="/manager">
      {% csrf_token %}
      <input type="hidden" name="decline" value="{{ order.id }}">
      <input type="submit" value="Decline">
    </form>
  </td>
  <td>
    <form method="post" action="/manager">
      {% csrf_token %}
      <input type="hidden" name="delete" value="{{ order.id }}">
      <input type="submit" value="Delete">
    </form>
  </td>
  <td>
    {% if order.ttn != None %}
    <a href="https://my.novaposhta.ua/orders/printDocument/orders[{{ order.ttn }}/orders[{{ order.ttn }}/type/pdf/apiKey/aa4e0992ea5446acbef2a12f8a4328b4" target="_blank">open ttn</a>
    {%endif%}
  </td>
</tr>
{% endfor %}
</table>
</body>
</html>

```

Views.py

```

from django.shortcuts import render
from django.http import HttpResponse
from django.shortcuts import render
from django.contrib.auth import authenticate
from django.http import HttpResponseRedirect
from django.conf import settings
from datetime import datetime, timedelta
import http.client
import requests
import json
from .forms import DeleteOrder, DeclineOrder, Order
from .forms import changeStatus
from .forms import Product
from .models import Orders
def index(request):
    pass
def order(request):
    if request.method == 'POST':
        form = Product(request.POST)
        if form.is_valid():
            pr_id = form.cleaned_data['pr_id']
            form1 = Order()
            form1.fields['purchaseId'].initial = pr_id
            return render(request, 'order.html', {'form': form1})
    else:
        form = Order()
        return render(request, 'order.html', {'form': form})
def catalogue(request):
    form = Product()
    return render(request, 'catalogue.html', {'form': form})
def finish(request):
    if request.method == 'POST':
        form = Order(request.POST)
        if form.is_valid():
            new_order = Orders(name = form.cleaned_data['name'],
                surname = form.cleaned_data['surname'],
                phone = form.cleaned_data['phone'],
                payment = form.cleaned_data['payment'],
                delivery = form.cleaned_data['delivery'],
                city = form.cleaned_data['city'],
                postbr = form.cleaned_data['postbr'],
                street = form.cleaned_data['street'],
                building = form.cleaned_data['building'],
                apartment = form.cleaned_data['apartment'],
                comment = form.cleaned_data['comment'],
                purchaseId = form.cleaned_data['purchaseId'],
                status = '1' )
            new_order.save()
            return render(request, 'finish.html', {'Thank': "Thanks \n our manager will contact you"})
    else:

```



```

    form = Order()
    return render(request, 'order.html', {'form': form, 'id': 'vvv'})
def manager(request):
    appr = changeStatus()
    orders = Orders.objects.all()
    decline = DeclineOrder()
    delete = DeleteOrder()
    if request.method == 'POST':
        appr = changeStatus(request.POST)
        declinef = DeclineOrder(request.POST)
        deletef = DeleteOrder(request.POST)
        if appr.is_valid():
            new_appr = appr.cleaned_data['approve']
            Orders.objects.filter(id=new_appr).update(status="2")
            client = Orders.objects.filter(id=new_appr).values()[0]
            gg = client['surname']
            ttn = create_ttn(client['name'], "Лебедева")
            Orders.objects.filter(id=new_appr).update(ttn=ttn)
            return render(request, "order_list.html", {'content': orders, "gg": gg})
        elif declinef.is_valid():
            decline = declinef.cleaned_data['decline']
            Orders.objects.filter(id=decline).update(status="3")
            return render(request, "order_list.html", {'content': orders})
        elif deletef.is_valid():
            delete = deletef.cleaned_data['delete']
            Orders.objects.filter(id=delete).delete()
            return render(request, "order_list.html", {'content': orders})
    else:
        return render(request, "order_list.html", {'content': orders})
def create_ttn(name, lastname):
    countr = {
        "apiKey": "aa4e0992ea5446acbef2a12f8a4328b4",
        "modelName": "Counterparty",
        "calledMethod": "getCounterparties",
        "methodProperties": {
            "CounterpartyProperty": "Sender",
            "Page": "1"
        }
    }
    response = requests.post('https://api.novaposhta.ua/v2.0/json/', json=countr)
    content = response.content
    countr_full = json.loads(content)
    countr_ref = countr_full['data'][0]['Ref']

    countr_contact_pers = {
        "apiKey": "aa4e0992ea5446acbef2a12f8a4328b4",
        "modelName": "Counterparty",
        "calledMethod": "getCounterpartyContactPersons",
        "methodProperties": {
            "Ref": countr_ref,
            "Page": "1"
        }
    }

```

```

}
response = requests.post('https://api.novaposhta.ua/v2.0/json/', json=countr_contact_pers)
content = response.content
countr_cont_full = json.loads(content)
countr_cont_full_ref = countr_cont_full['data'][0]['Ref']
countr_cont_full_phones = countr_cont_full['data'][0]['Phones']
create_receipient = {
    "apiKey": "aa4e0992ea5446acbef2a12f8a4328b4",
    "modelName": "Counterparty",
    "calledMethod": "save",
    "methodProperties": {
        "FirstName": name,
        "MiddleName": "Віталіївна",
        "LastName": lastname,
        "Phone": "0999203721",
        "Email": "",
        "CounterpartyType": "PrivatePerson",
        "CounterpartyProperty": "Recipient"
    }
}
}
response = requests.post('https://api.novaposhta.ua/v2.0/json/', json=create_receipient)
content = response.content
new_receipient_full = json.loads(content)
new_receipient_full_ref = new_receipient_full['data'][0]['Ref']
#new_receipient_full_phones = new_receipient_full['data'][0]['Phones']
get_cont_receipient = {
    "apiKey": "aa4e0992ea5446acbef2a12f8a4328b4",
    "modelName": "Counterparty",
    "calledMethod": "getCounterpartyContactPersons",
    "methodProperties": {
        "Ref": new_receipient_full_ref,
        "Page": "1"
    }
}
}
response = requests.post('https://api.novaposhta.ua/v2.0/json/', json=get_cont_receipient)
content = response.content
get_cont_receipient_full = json.loads(content)
get_cont_receipient_full_ref = get_cont_receipient_full['data'][0]['Ref']
get_cont_receipient_full_phones = get_cont_receipient_full['data'][0]['Phones']
get_city = {
    "modelName": "Address",
    "calledMethod": "getCities",
    "methodProperties": {
        "FindByString": "Харьков"
    },
    "apiKey": "aa4e0992ea5446acbef2a12f8a4328b4"
}
}
response = requests.post('https://api.novaposhta.ua/v2.0/json/', json=get_city)
content = response.content
get_city_full = json.loads(content)
get_city_full_ref = get_city_full['data'][0]['Ref']

```

```

get_street = {
  "modelName": "Address",
  "calledMethod": "getStreet",
  "methodProperties": {
    "CityRef": get_city_full_ref,
    "FindByString": "Сумська"
  },
  "apiKey": "aa4e0992ea5446acbef2a12f8a4328b4"
}
response = requests.post('https://api.novaposhta.ua/v2.0/json/', json=get_street)
content = response.content
get_street_full = json.loads(content)
get_street_full_ref = get_street_full['data'][0]['Ref']
get_adress = {
  "modelName": "Address",
  "calledMethod": "save",
  "methodProperties": {
    "CounterpartyRef": new_receipient_full_ref,
    "StreetRef": get_street_full_ref,
    "BuildingNumber": "7",
    "Flat": "2",
    "Note": "Коментарий"
  },
  "apiKey": "aa4e0992ea5446acbef2a12f8a4328b4"
}
response = requests.post('https://api.novaposhta.ua/v2.0/json/', json=get_adress)
content = response.content
get_adress_full = json.loads(content)
get_adress_ref = get_adress_full['data'][0]['Ref']
date="20.06.2021"
params1 = {
  "apiKey": "aa4e0992ea5446acbef2a12f8a4328b4",
  "modelName": "InternetDocument",
  "calledMethod": "save",
  "methodProperties": {
    "PayerType": "Sender",
    "PaymentMethod": "Cash",
    "DateTime": date,
    "CargoType": "Cargo",
    "VolumeGeneral": "0.1",
    "Weight": "10",
    "ServiceType": "WarehouseDoors",
    "SeatsAmount": "1",
    "Description": "Oxygen3000",
    "Cost": "500",
    "CitySender": "8d5a980d-391c-11dd-90d9-001a92567626",
    "Sender": countr_ref,
    "SenderAddress": "01ae2635-e1c2-11e3-8c4a-0050568002cf",
    "ContactSender": countr_cont_full_ref,
    "SendersPhone": countr_cont_full_phones,
    "CityRecipient": get_city_full_ref,
    "Recipient": new_receipient_full_ref,

```

```
    "RecipientAddress": get_adress_ref,  
    "ContactRecipient": get_cont_receipient_full_ref,  
    "RecipientsPhone": get_cont_receipient_full_phones  
  }  
}  
response = requests.post('https://api.novaposhta.ua/v2.0/json/', json=params1)  
content = response.content  
rezult = json.loads(content)  
get_t = rezult['data'][0]['Ref']  
return get_t
```