

МГТУ им. Н.Э. Баумана

Отчёт по домашнему заданию
по курсу «Программирование и конструкции
языков программирования»

Руководитель
Гапанюк Ю. Е.

Студент группы ИУ5-33Б
Ковалев А. В.

2023 г.

Задание

Создать веб-сервис реализующий функционал создания, хранения и корректировки списков покупок пользователей. В дополнение реализовать Telegram бота, выдающего содержимое списка покупок.

Данный веб-сервис должен осуществлять:

- 1) Хранение данных пользователей, причем должно применяться хэширование паролей.
- 2) Авторизацию и регистрацию пользователей.
- 3) Добавление в и удаление из корзины (списка) позиций каталога.
- 4) Демонстрацию содержимого корзины (списка).

Telegram бот должен по введенным логину и паролю отправлять сообщение перечисляющее позиции списка.

Код веб-составляющей

Web_test.py

```
from flask import *
from werkzeug.security import generate_password_hash, check_password_hash
import json
from Class_User import *
from Class_Card import *

app = Flask(__name__)

Users = Users_data()

Cards = Cards_data()

Users.load('users.json')

Cards.load('cards.json')

Users.show()

@app.route('/exit')
def exit():
    Users.set_non_active(request.remote_addr)
    return entrance()

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

```

@app.route('/sign_up/', methods=['post', 'get'])
def sign_up():
    if request.method == 'POST':
        username = request.form.get('username')
        password = request.form.get('password')
        conf_pas = request.form.get('conf_pas')
        if (Users.find_user(username) == True):
            return render_template('Sign_up_form.html', message='There is user
with such name')
        elif (password != conf_pas):
            return render_template('Sign_up_form.html', message='Passwords don\'t
match')
        Users.add(username, password)
        Users.set_last_ip(username, request.remote_addr)
        Cards.add_card(username, items=[])
        Users.save('users.json')
        return render_template('Shopping_cart.html', message=username + '\s
card')
    return render_template('Sign_up_form.html', message='Registration')

@app.route('/log_in/', methods=['post', 'get'])
def log_in():
    if request.method == 'POST':
        username = request.form.get('username') # запрос к данным формы
        password = request.form.get('password')
        if (Users.find_user(username) == True):
            if (Users.enter(username, password) == False):
                return render_template('Log_in_form.html', message='Wrong
password')
            else:
                Users.set_last_ip(username, request.remote_addr)
                Users.save('users.json')
                return render_template('Shopping_cart.html', message=username +
'\s card')
        else:
            return render_template('Log_in_form.html', message='There is no user
with such name')
    return render_template('Log_in_form.html', message='Authorization')

@app.route('/add_item', methods=['post', 'get'])
def add_item():
    item = ''
    owner = Users.active_ip_finding(request.remote_addr)
    if request.method == 'POST':
        content = request.json
        item = content['item']
    if (owner == False):
        return entrance()
    else:

```

```

        Cards.add_to_user_card(owner, item)
        Cards.save('cards.json')
        return make_response("", 200)

@app.route('/del_item', methods=['post', 'get'])
def del_item():
    item = ''
    owner = Users.active_ip_finding(request.remote_addr)
    if request.method == 'POST':
        content = request.json
        item = content['item']
    if (owner == False):
        return entrance()
    else:
        Cards.del_from_user_card(owner, item)
        Cards.save('cards.json')
        return make_response("", 200)

@app.route('/renew', methods=['post', 'get'])
def renew():
    owner = Users.active_ip_finding(request.remote_addr)
    if (owner == False):
        return entrance()
    else:
        return make_response(Cards.no_user_items(owner), 200)

@app.route('/catalog', methods=['post', 'get'])
def catalog():
    if (Users.active_ip_finding(request.remote_addr) == False):
        return entrance()
    return render_template('Catalog.html', message='Authorization')

@app.route('/shopping_cart')
def Shopping_cart():
    owner = Users.active_ip_finding(request.remote_addr)
    if (owner == False):
        return entrance()
    return render_template('Shopping_cart.html', message=owner + '\'s card')

if __name__ == "__main__":
    app.run(host='0.0.0.0', port=4567)

```

Class_User.py

```

from werkzeug.security import generate_password_hash, check_password_hash

```

```

import json

class User():

    def __init__(self, name, password_hash):
        self.password_hash = password_hash
        self.name = name
        self.date_time = 0
        self.last_ip = 0
        self.activity = False

    def compare(self, name):
        if (self.name == name):
            return True
        return False

    def enter(self, password):
        return check_password_hash(self.password_hash, password)

class Users_data(User):
    users = []

    def find_user(self, name):
        for i in self.users:
            if (i.compare(name) == True):
                return True
        return False

    def enter(self, name, password):
        for i in self.users:
            if (i.compare(name) == True and check_password_hash(i.password_hash,
password) == True):
                return True
        return False

    def add(self, name, password):
        if (self.find_user(name) == True):
            return False
        self.users.append(User(name, generate_password_hash(password)))
        return True

    def save(self, file):
        dicts_list = list()
        for x in self.users:
            result = x.__dict__
            result["className"] = x.__class__.__name__
            dicts_list.append(result)
        with open(file, "w") as data:
            json.dump(dicts_list, data)

```

```

def load(self, file):
    with open(file, "r") as data:
        json_data = json.load(data)
    for x in json_data:
        if (x['className'] == 'User'):
            self.users.append(User(x['name'], x['password_hash']))

def __init__(self):
    pass

def show(self):
    for x in self.users:
        print(x.name, x.password_hash, x.activity, x.last_ip)

def set_last_ip(self, name, ip):
    for i in range(0, len(self.users)):
        if (self.users[i].name == name):
            self.users[i].last_ip = ip
            self.users[i].activity = True
        if (self.users[i].last_ip == ip and self.users[i].name != name):
            self.users[i].activity = False

def set_non_active(self, ip):
    for i in range(0, len(self.users)):
        if (self.users[i].last_ip == ip):
            self.users[i].activity = False

def active_ip_finding(self, ip):
    for i in range(0, len(self.users)):
        if (self.users[i].last_ip == ip and self.users[i].activity == True):
            return self.users[i].name
    return False

```

Class_Card.py

```

import json
catalog = ['carrot', 'banana', 'orange', 'apple', 'blueberry']

class Card():

    def is_item(self, item):
        for x in self.items:
            if (x == item):
                return True
        return False

    def __init__(self, owner, items=[]):

```

```

        self.owner = owner
        self.items = items

class Cards_data(Card):

    cards = []

    def find_user_card(self, owner):
        for i in range(0, len(self.cards)):
            if (self.cards[i].owner == owner):
                return i
        return -1

    def add_card(self, owner, items=[]):
        if (self.find_user_card(owner) == -1):
            self.cards.append(Card(owner, items))

    def add_to_user_card(self, owner, item):
        n = self.find_user_card(owner)
        if (n == -1):
            print('add-')
            self.add_card(owner)
        n1 = self.find_user_card(owner)
        if not (item in self.cards[n].items):
            self.cards[n].items.append(item)

    def del_from_user_card(self, owner, item):
        n = self.find_user_card(owner)
        if (n == -1):
            print('del-' + owner)
            self.add_card(owner)
        n1 = self.find_user_card(owner)
        if (item in self.cards[n].items):
            self.cards[n].items.remove(item)

    def save(self, file):
        dicts_list = list()
        for x in self.cards:
            result = x.__dict__
            result["className"] = x.__class__.__name__
            dicts_list.append(result)
        with open(file, "w") as data:
            json.dump(dicts_list, data)

    def load(self, file):
        with open(file, "r") as data:
            json_data = json.load(data)
        for x in json_data:
            if (x['className'] == 'Card'):
                self.cards.append(Card(x['owner'], x['items']))

```

```

def no_user_items(self, owner):
    n = self.find_user_card(owner)
    res = ''
    if (n != -1):
        for i in catalog:
            if not (i in self.cards[n].items):
                res += i + ', '
    return res

def user_items(self, owner):
    n = self.find_user_card(owner)
    res = ''
    if (n != -1):
        for i in self.cards[n].items:
            res += i + ', '
        if len(self.cards[n].items) != 0:
            res = res[:-2]
    return res

def __init__(self):
    pass

def show(self):
    for x in self.cards:
        print(x.owner, x.items)

```

Shopping_cart.html

```

<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <title>Shopping cart</title>
</head>
<style>
    html {
        font-family: Times New Roman;
        display: inline-block;
        text-align: left;
    }

    h2 {
        font-size: 2.0rem;
        color: #ccc;
        padding: 0em 0em 0em;
        text-align: center;
    }

```



```
h1 {
  font-size: 1.6rem;
  color: #ccc;
  padding-left: 10px;
  padding-right: 10px;
  text-align: center;
  margin-top: 10;
}

p {
  font-size: 1.6rem;
  color: #ccc;
  padding: 0em 0em 0em;
  text-align: center;
}

body {
  background: #000000;
  max-width: 600px;
  margin: 0px auto;
  padding-bottom: 25px;
}

.button3 {
  display: inline-block;
  background-color: #008CBA;
  border: none;
  border-radius: 4px;
  color: #ccc;
  padding-top: 6px;
  margin: 6px;
  text-decoration: none;
  font-size: 25px;
}

.button4 {
  display: inline-block;
  background-color: #008CBA;
  border: none;
  border-radius: 4px;
  color: #ccc;
  padding: 5px 20px;
  margin: 6px;
  text-decoration: none;
  font-size: 25px;
}
</style>

<body>
  {% if message %}
```

```

<h2>{{ message }}</h2>
{% endif %}

<h1><button class="button3" id="banana" onclick="Del_item(this)"></button>
    <button class="button3" id="apple" onclick="Del_item(this)"></button>
    <button class="button3" id="carrot" onclick="Del_item(this)"></button>
    <button class="button3" id="blueberry" onclick="Del_item(this)"></button>
    <button class="button3" id="orange" onclick="Del_item(this)"></button>
</h1>
<h1><a href='/catalog'><button class="button4" id="banana">To
catalog</button></a></h1>
<h1><a href='/exit'><button class="button4" id="banana" style="background-
color: grey;">Exit</button></a></h1>

</body>
<script>
    document.addEventListener("DOMContentLoaded", ButtonStyle);
    function ButtonStyle(event) {
        var req = new XMLHttpRequest();
        req.open("GET", "/renew", false);
        req.addEventListener("load", function () {
            if (req.status < 400) {
                var words = req.responseText.split(",");
                words.forEach(element => {
                    var elem = document.getElementById(element);
                    elem.remove();
                });
            }
        });
        req.send(null);
    }

    function Del_item(_this) {
        const item = {
            "item": _this.id,
        };
        const Http = new XMLHttpRequest();
        const url = '/del_item';
        Http.open("POST", url);
        Http.setRequestHeader("Content-Type", "application/json");
        Http.send(JSON.stringify(item));
        _this.remove()
    }
</script>

```

```
</html>
```

Catalog.html

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <title>Catalog</title>
</head>
<style>
  html {
    font-family: Times New Roman;
    display: inline-block;
    text-align: left;
  }

  h2 {
    font-size: 2.0rem;
    color: #ccc;
    padding: 0em 0em 0em;
    text-align: center;
  }

  h1 {
    font-size: 1.6rem;
    color: #ccc;
    padding-left: 10px;
    padding-right: 10px;
    text-align: center;
    margin-top: 10;
  }

  p {
    font-size: 1.6rem;
    color: #ccc;
    padding: 0em 0em 0em;
    text-align: center;
  }

  body {
    background: #000000;
    max-width: 600px;
    margin: 0px auto;
    padding-bottom: 25px;
  }

  .button3 {
    display: inline-block;
```

```

        background-color: #008CBA;
        border: none;
        border-radius: 4px;
        color: #ccc;
        padding-top: 6px;
        margin: 6px;
        text-decoration: none;
        font-size: 25px;
    }

    .button4 {
        display: inline-block;
        background-color: #008CBA;
        border: none;
        border-radius: 4px;
        color: #ccc;
        padding: 5px 20px;
        margin: 6px;
        text-decoration: none;
        font-size: 25px;
    }
</style>

<body>
    <h2>Catalog</h2>
    <h1><button class="button3" onclick="ButtonStyle(this)" id="banana"></button>
        <button class="button3" onclick="ButtonStyle(this)" id="apple"></button>
        <button class="button3" onclick="ButtonStyle(this)" id="carrot"></button>
        <button class="button3" onclick="ButtonStyle(this)" id="blueberry"></button>
        <button class="button3" onclick="ButtonStyle(this)" id="orange"></button>
    </h1>
    <h1><a href='/shopping_cart'><button class="button4" id="banana">To shopping
cart</button></a></h1>
    <h1><a href='/exit'><button class="button4" id="banana" style="background-
color: grey;">Exit</button></a></h1>
</body>
<script>
    function ButtonStyle(_this) {
        const item = {
            "item": _this.id,
        };
        const Http = new XMLHttpRequest();
        var url = ''

        if (_this.style.backgroundColor == "red") {

```

```

        _this.style.backgroundColor = "#008CBA";
        url = '/del_item';
    }
    else {
        _this.style.backgroundColor = "red";
        url = '/add_item';
    }

    Http.open("POST", url);
    Http.setRequestHeader("Content-Type", "application/json");
    Http.send(JSON.stringify(item));
}

</script>

</html>

```

Log_in_form.html

```

<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <title>Log in</title>
</head>
<style>
    html {
        font-family: Times New Roman;
        display: inline-block;
        text-align: left;
    }

    h2 {
        font-size: 2.0rem;
        color: #ccc;
        padding: 0em 0em 0em;
        text-align: center;
    }

    h1 {
        font-size: 1.6rem;
        color: #ccc;
        padding-left: 10px;
        padding-right: 10px;
        text-align: center;
        margin-top: 10;
    }

    p {
        font-size: 1.6rem;
        color: #ccc;
    }

```

```

        padding: 0em 0em 0em;
        text-align: center;
    }

    body {
        background: #000000;
        max-width: 600px;
        margin: 0px auto;
        padding-bottom: 25px;
    }

    .button3 {
        display: inline-block;
        background-color: #008CBA;
        border: none;
        border-radius: 4px;
        color: #ccc;
        padding: 5px 20px;
        margin: 6px;
        text-decoration: none;
        font-size: 25px;
    }
</style>

<body>

    {% if message %}
    <h1>{{ message }}</h1>
    {% endif %}

    <form action="" method="post">
        <h1>
            <input type="text" name="username" placeholder="Username" value=""
required minlength="4" maxlength="16"
            style="border: none; font-size:18pt; height:30px; width:300px;
border-radius: 4px">
        </h1>
        <h1>
            <input type="password" name="password" placeholder="Password"
value="" required minlength="4" maxlength="16"
            style="border: none; font-size:18pt; height:30px; width:300px;
border-radius: 4px">
        </h1>
        <h1><input type="submit" value="Sign up" class=button3></h1>
    </form>

</body>

</html>

```

Sign_up_form.html

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <title>Sign up</title>
</head>
<style>
  html {
    font-family: Times New Roman;
    display: inline-block;
    text-align: left;
  }

  h2 {
    font-size: 2.0rem;
    color: #ccc;
    padding: 0em 0em 0em;
    text-align: center;
  }

  h1 {
    font-size: 1.6rem;
    color: #ccc;
    padding-left: 10px;
    padding-right: 10px;
    text-align: center;
    margin-top: 10;
  }

  p {
    font-size: 1.6rem;
    color: #ccc;
    padding: 0em 0em 0em;
    text-align: center;
  }

  body {
    background: #000000;
    max-width: 600px;
    margin: 0px auto;
    padding-bottom: 25px;
  }

  .button3 {
    display: inline-block;
    background-color: #008CBA;
    border: none;
    border-radius: 4px;
```

```

        color: #ccc;
        padding: 5px 20px;
        margin: 6px;
        text-decoration: none;
        font-size: 25px;
    }
</style>

<body>

    {% if message %}
    <h1>{{ message }}</h1>
    {% endif %}

    <form action="" method="post">
        <h1>
            <input type="text" name="username" placeholder="Username" value="" required
minlength="4" maxlength="16"
            style="border: none; font-size:18pt; height:30px; width:300px; border-
radius: 4px">
        </h1>
        <h1>
            <input type="password" name="password" placeholder="Password" value=""
required minlength="4" maxlength="16"
            style="border: none; font-size:18pt; height:30px; width:300px; border-
radius: 4px">
        </h1>

        <h1>
            <input type="password" name="conf_pas" placeholder="Password again"
value="" required minlength="4" maxlength="16"
            style="border: none; font-size:18pt; height:30px; width:300px; border-
radius: 4px">
        </h1>
        <h1><input type="submit" value="Register" class=button3</h1>
    </form>

</body>

</html>

```

Entrance.html

```

<html>

<head>
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>Entrance</title>
    <style>
        html {

```



```
        font-family: Times New Roman;
        display: inline-block;
        text-align: left;
    }

    h2 {
        font-size: 2.0rem;
        color: #ccc;
        padding: 0em 0em 0em;
        text-align: center;
    }

    h1 {
        font-size: 1.6rem;
        color: #ccc;
        padding-left: 10px;
        padding-right: 10px;
        text-align: left;
        margin-top: 10;
        margin-down: 10;
    }

    p {
        font-size: 1.6rem;
        color: #ccc;
        padding: 0em 0em 0em;
        text-align: center;
    }

    body {
        background: #000000;
        max-width: 600px;
        margin: 0px auto;
        padding-bottom: 25px;
    }

    .button3 {
        display: inline-block;
        background-color: #008CBA;
        border: none;
        border-radius: 4px;
        color: #ccc;
        padding: 5px 20px;
        margin: 6px;
        text-decoration: none;
        font-size: 25px;
    }
</style>
</head>

<body>
```

```

    <h2>Entrance</h2>
    <p><a href="/log_in"><button class="button3">Log in</button></a><a
href="/sign_up"><button class="button3">Sign
        up</button></a></p>
</body>

</html>

```

Код Telegram бота

bot_test.py

```

from Class_Card import *
from Class_User import *
from telebot import *
bot = telebot.TeleBot('6893332938:AAEvs3DyR1-nL2b_Xzxvgf821oA_CAc1T6k')

dictionary = {}

Users = Users_data()

Cards = Cards_data()

Users.load('users.json')

Cards.load('cards.json')

@bot.message_handler(content_types=['text'])
def get_text_messages(message):
    keyboard = types.InlineKeyboardMarkup() # наша клавиатура
    key_start = types.InlineKeyboardButton(
        text='Получить список', callback_data='list')
    keyboard.add(key_start)
    key_end = types.InlineKeyboardButton(
        text='Ничего', callback_data='end')
    keyboard.add(key_end)
    bot.send_message(message.from_user.id, text="Что вы хотите?",
        reply_markup=keyboard)

@bot.callback_query_handler(func=lambda call: True)
def callback_worker(call):
    if call.data == "list" or call.data == "re_username":
        bot.send_message(call.message.chat.id, "Введите логин")
        bot.register_next_step_handler(call.message, get_username)
    elif call.data == "end":
        bot.send_message(call.message.chat.id, "Хорошего дня")
    elif call.data == "re_password":
        bot.send_message(call.message.chat.id, "Введите пароль")

```

```

        bot.register_next_step_handler(call.message, get_password)

def get_username(message):
    username = message.text
    if (Users.find_user(username) == True):
        bot.send_message(message.from_user.id,
                           "Пользователь найден. Введите пароль.")
        bot.register_next_step_handler(message, get_password)
        id = message.from_user.id
        dictionary[str(id)] = str(username)
    else:
        keyboard = types.InlineKeyboardMarkup()
        keyboard = types.InlineKeyboardMarkup() # наша клавиатура
        key_restart = types.InlineKeyboardButton(
            text='Ввести логин снова', callback_data='re_username')
        keyboard.add(key_restart)
        key_stop = types.InlineKeyboardButton(
            text='Прервать вход', callback_data='end')
        keyboard.add(key_stop)
        bot.send_message(message.from_user.id, text="Пользователь не найден.",
                           reply_markup=keyboard)

print(message.text)

def get_password(message):
    password = message.text
    id = message.from_user.id

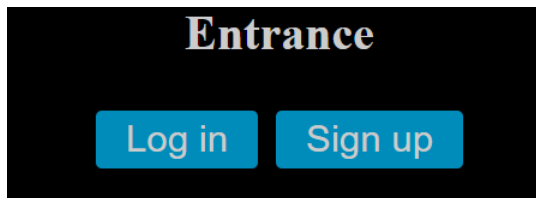
    if (Users.enter(dictionary.get(str(id)), password) == False):
        keyboard = types.InlineKeyboardMarkup()
        keyboard = types.InlineKeyboardMarkup() # наша клавиатура
        key_restart = types.InlineKeyboardButton(
            text='Ввести пароль снова', callback_data='re_password')
        keyboard.add(key_restart)
        key_stop = types.InlineKeyboardButton(
            text='Прервать вход', callback_data='end')
        keyboard.add(key_stop)
        bot.send_message(message.from_user.id, text="Пароль неверный.",
                           reply_markup=keyboard)
    else:
        bot.send_message(message.from_user.id, text="Вход осуществлен.")
        bot.send_message(message.from_user.id, text="Ваш список:")
        bot.send_message(message.from_user.id,
                           text=Cards.user_items(dictionary.get(str(id))))

bot.polling(none_stop=True, interval=0)

```

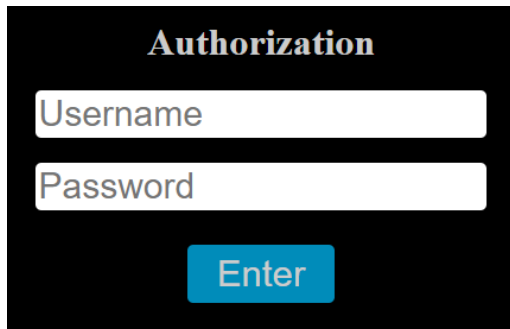
Взаимодействие с веб-сервисом

При переходе на сайт пользователь попадает на следующую страницу.



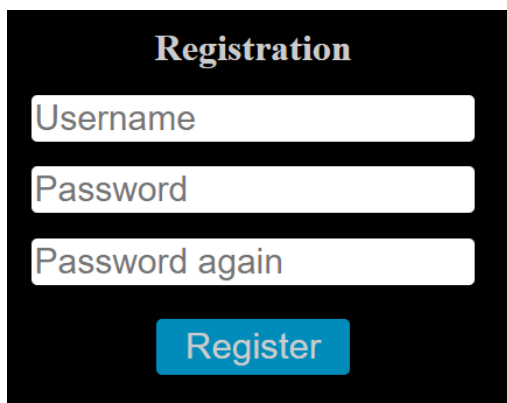
The image shows a black rectangular box with the word "Entrance" in a white serif font at the top center. Below the title, there are two blue rectangular buttons with white text. The left button is labeled "Log in" and the right button is labeled "Sign up".

При нажатии на кнопку “Log in” происходит переход на форму авторизации.



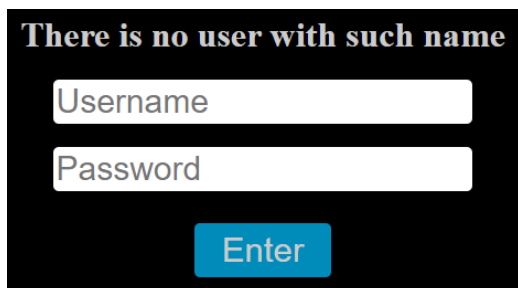
The image shows a black rectangular box with the word "Authorization" in a white serif font at the top center. Below the title, there are two white rectangular input fields. The first field is labeled "Username" and the second field is labeled "Password". Below the input fields, there is a blue rectangular button with white text labeled "Enter".

При нажатии на кнопку “Sign up” происходит переход на форму регистрации.



The image shows a black rectangular box with the word "Registration" in a white serif font at the top center. Below the title, there are three white rectangular input fields. The first field is labeled "Username", the second field is labeled "Password", and the third field is labeled "Password again". Below the input fields, there is a blue rectangular button with white text labeled "Register".

При вводе несуществующего логина в форму авторизации появляется следующее сообщение:



The image shows a black rectangular box with the text "There is no user with such name" in a white serif font at the top center. Below the message, there are two white rectangular input fields. The first field is labeled "Username" and the second field is labeled "Password". Below the input fields, there is a blue rectangular button with white text labeled "Enter".

При вводе неверного пароля в форму авторизации:



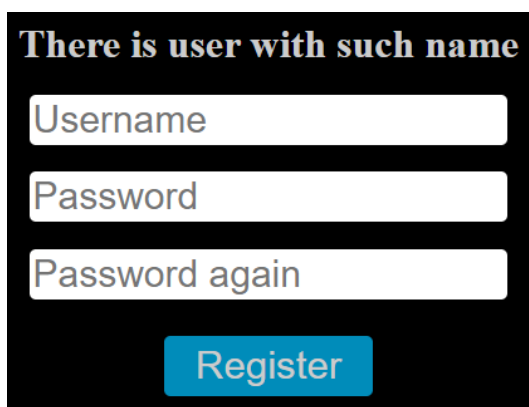
Wrong password

Username

Password

[Enter](#)

При вводе уже существующего логина в форму регистрации:



There is user with such name

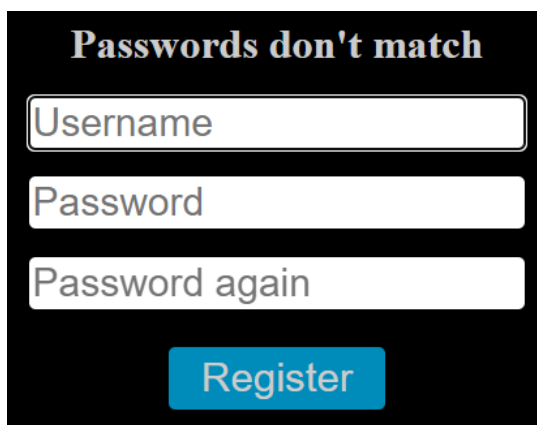
Username

Password

Password again

[Register](#)

При вводе несовпадающих паролей в форму регистрации:



Passwords don't match

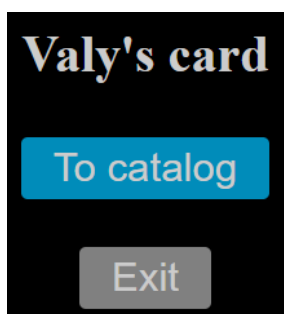
Username

Password

Password again

[Register](#)

При успешной регистрации происходит переход на страницу с пустой корзиной.



Valy's card

[To catalog](#)

[Exit](#)

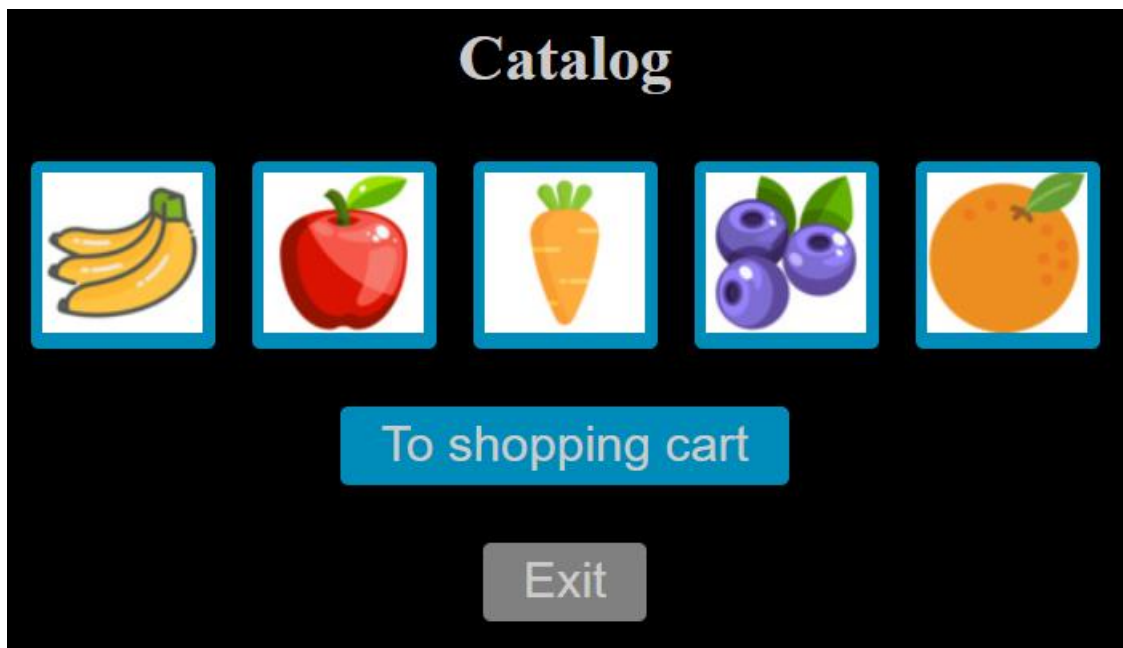
При успешной авторизации открывается страница, содержащая ранее составленный пользователем список.



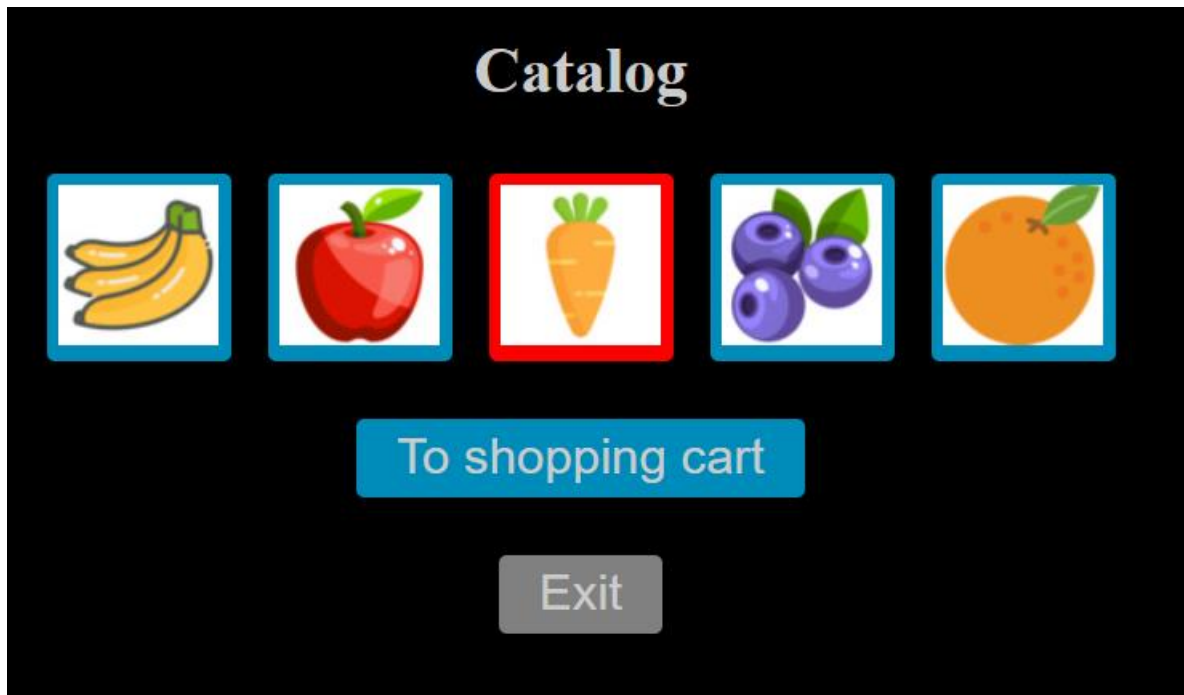
При нажатии на иконку товара он исчезает со страницы и удаляется из корзины.

Иллюстрация не представляется возможной

При нажатии на кнопку “To catalog” открывается страница-каталог с доступными для добавления в список позициями.



При нажатии на иконку товара она выделяется красным и добавляется в корзину. При повторном нажатии товар удаляется из корзины и снова выделяется синим.

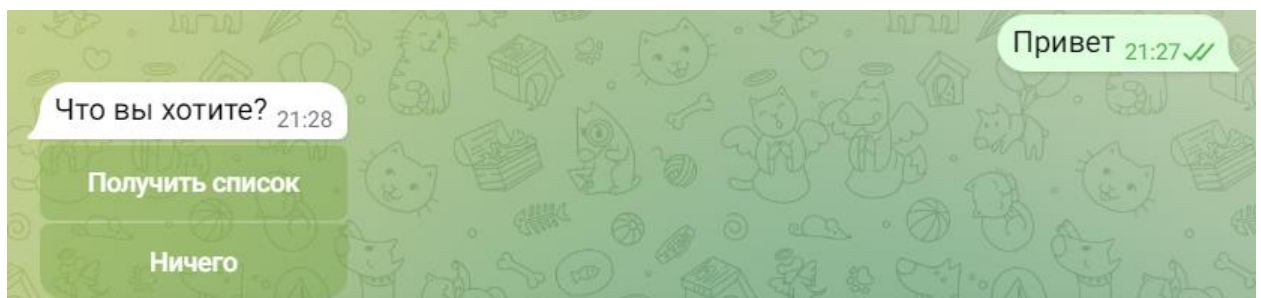


При нажатии на кнопку “To shopping cart” происходит переход на приведенную выше страницу со списком пользователя.

При нажатии на кнопку “Exit” страницы со списком пользователя или страницы-каталога осуществляется выход из аккаунта и открывается страница выбора регистрации или авторизации.

Взаимодействие со связанным Telegram ботом.

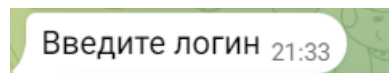
При вводе любого сообщения (например «Привет») пользователь получает следующий ответ:



При нажатии на кнопку «Ничего» диалог завершается таким сообщением от бота:



При нажатии на кнопку «Получить список» бот отправляет сообщение:

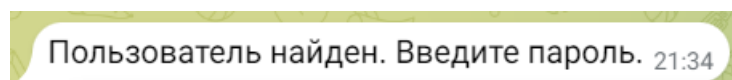


При вводе логина, которого нет в списке пользователей:

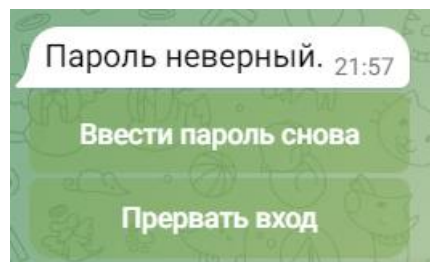


Кнопка «Ввести логин снова» начинает заново процедуру ввода, а «Прервать вход» завершает диалог приведенным ранее пожеланием хорошего дня.

При вводе существующего логина пользователь получает следующее сообщение:



Если ввести некорректный пароль, то бот отправит сообщение следующего содержания:



Здесь кнопки «Ввести пароль снова» и «Прервать вход» аналогичны по функционалу и названиям кнопкам, отправляемым при вводе несуществующего логина.

При вводе корректного пароля пользователь получит сообщение об успешной авторизации и перечисление позиций списка:

