**ФЕДЕРАЛЬНОЕ АГЕНТСТВО СВЯЗИ**

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

**«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ТЕЛЕКОММУНИКАЦИЙ ИМ. ПРОФ. М.А. БОНЧ-БРУЕВИЧА»**

**(СПбГУТ)**

**Кафедра безопасности информационных систем**

**ОТЧЕТ**

по лабораторной работе №3 на тему:

**«Разработка системы аутентификации и авторизации информационной системы»**

по дисциплине «Веб-технологии и защита информации»

**Исполнитель:**

студент 3 курса, группы ИСТ-831

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/Пономарев Е.И./

«\_\_\_»\_\_\_\_\_\_\_\_2020

**Проверил**

ст. пр. кафедры БИС

\_\_\_\_\_\_\_\_\_\_\_\_/Евстигнеев В.А./

«\_\_\_»\_\_\_\_\_\_\_\_2020

Санкт-Петербург

2020

**СОДЕРЖАНИЕ**

[Постановка задачи 3](#__RefHeading___Toc3137_499925716)

[Выбор методов 3](#__RefHeading___Toc3139_499925716)

[Решение задачи 3](#__RefHeading___Toc3141_499925716)

[Заключение 14](#__RefHeading___Toc3143_499925716)

# **Постановка задачи**

Разработать систему аутентификации и авторизации информационной системы «Интернет-магазин».

Так как в нашей системе предусмотрено создание личного кабинета, то необходимо создать модуль регистрации нового пользователя, предусматривающего поля имени, фамилии , пароля и email пользователя.

При аутентификации использовать email пользователя и его пароль

# **Выбор методов**

Языком серверной части является python , фреймворк — flask. Выбор на данные средства разработки выпал из-за прогрессивности языка,наличия большого количества библиотек, облегчающих разработку, а также быстроты разработки.

В качестве системы, реализующую авторизацию и аутентификацию , была выбрана библиотека flask-login.

В качестве рендеринга html-страницы фреймворком flask поставляется библиотека jinja2

# **Решение задачи**

Для начала реализуем всю необходимую серверную логику.

Блок кода, который отвечает за переход пользователя по адресу /login.

@app.route("/login",methods = ['GET','POST'])

def loginUser():

    if(flask\_login.current\_user.is\_authenticated): #Проверяем вошел ли уже пользователь

        return redirect(url\_for('userInfo')) #Перекидываем на страницу профиля

    if(request.method == 'POST'): # Если метод обращения к url POST

        email = request.form.get('email',type=str) # Получаем введенный email

        password = request.form.get('password',type=str) # Получаем введенный пароль

        userID = User.validateUserAndReturnUserID(email,password) #Получаем ID пользователя по введенному email и паролю

        if(userID != -1):# Если пользователь найден

            flask\_login.login\_user(load\_user(userID),remember=True)#Логиним пользователя в системе

            return redirect(url\_for('showBooks',page=1))# Перекидываем на страницу книг

        else:

            return render\_template('login.html',error = "Not found")#Выводим страницу логина с ошибкой

    else:

        return render\_template('login.html')# Если метод обращения не POST , то выводим html страницу логина

Валидация введенных email и пароля пользователя

    @staticmethod

    def validateUserAndReturnUserID(email,password):

        cursor = db.execute(

            'select \* from Покупатель where email \

             = "{}" and password\_hash = "{}"'.format(email,md5helper.ecnrypt(password))

            ) # Ищем пользователя в системе

        allRows = cursor.fetchall()

        if(len(allRows) == 0):

            return -1

        else:

            return allRows[0][0] # Возвращаем ID пользователя

Процесс регистрации нового пользователя,обработка перехода пользователя на /registration и вывод соответствующего шаблона html

@app.route('/registration',methods=['GET','POST'])

def registrationUser():

    if(flask\_login.current\_user.is\_authenticated):

        return redirect(url\_for('userInfo'))

    if(request.method == 'POST'):

        email = request.form.get('email',type=str) # Получаем введенный email пользователя

        pswd = request.form.get('pswd',type=str) # Получаем введенный пароль пользователя

        pswd2 = request.form.get('pswd2',type=str) # Получаем введенный 2 пароль пользователя

        first\_name = request.form.get('firstName',type=str) # Получаем введенныое имя пользователя

        last\_name = request.form.get('lastName',type=str) # Получаем введенную фамилию пользователя

        birthdate = request.form.get('birthDate') # Получаем введенную дата рождения пользователя

        if(pswd != pswd2):

            return render\_template('registration.html',error = 'Password mismatch') # Возвращаем html с ошибкой

        resultRegisterOperation = User.registerUser(email,pswd,last\_name,first\_name,birthdate) # Создаем пользователя

        if(resultRegisterOperation["status"] == 1):

            return render\_template('registration.html',error = 'User already exists') # Возвращаем html с ошибкой

        elif(resultRegisterOperation['status'] == 2):

            return render\_template('registration.html',error = 'Incorrect email') # Возвращаем html с ошибкой

        flask\_login.login\_user(load\_user(resultRegisterOperation["userID"]),remember=True) # Авторизируем пользователя

        return redirect(url\_for('userInfo'))

    else:

        return render\_template('registration.html')# Возвращаем html с формой

Процесс создания нового пользователя в базе данных

@staticmethod

    def registerUser(email,password,last\_name,first\_name,birthdate):

        result = {}

        cursor = db.execute('select \* from Покупатель where email="{}";'.format(email))

        if(cursor.fetchone() is not None):# Проверка,если пользователь с таким email существует

            result['status'] = 1

            result['message'] = 'User with same email already exists'

            cursor.close()

            return result

        if(not validationForm.validationEmail(email)):# Если неправильно введен email

            result['status'] = 2

            result['message'] = 'Incorrect email'

            cursor.close()

            return result

        cursor.close()

        cursor = db.execute(

            'insert into Покупатель("email","last\_name","first\_name","birthdate","password\_hash") \

                values("{}","{}","{}","{}","{}")'.format(

                    email,last\_name,first\_name,birthdate,md5helper.ecnrypt(password))

                ) # Создание нового пользователя

        db.commit()

        result["status"] = 0

        result["message"] = "OK"

        result["userID"] = cursor.lastrowid

        cursor.close()

        return result

Дальше следует создать формы регистрации и логина

Ниже приведен листинг html-верстки страницы логина

<!DOCTYPE html>

<html>

  <head><title>Войти</title></head>

<style type="text/css">

    @import url(https://fonts.googleapis.com/css?family=Roboto:300);

.login-page {

  width: 360px;

  padding: 8% 0 0;

  margin: auto;

}

.form {

  position: relative;

  z-index: 1;

  background: #FFFFFF;

  max-width: 360px;

  margin: 0 auto 100px;

  padding: 35px;

  text-align: center;

  box-shadow: 0 0 20px 0 rgba(0, 0, 0, 0.2), 0 5px 5px 0 rgba(0, 0, 0, 0.24);

}

.form input {

  font-family: "Roboto", sans-serif;

  outline: 0;

  background: #f2f2f2;

  width: 100%;

  border: 0;

  margin: 0 0 15px;

  padding: 15px;

  box-sizing: border-box;

  font-size: 14px;

}

.form button {

  font-family: "Roboto", sans-serif;

  text-transform: uppercase;

  outline: 0;

  background: #4CAF50;

  width: 100%;

  border: 0;

  padding: 15px;

  color: #FFFFFF;

  font-size: 14px;

  -webkit-transition: all 0.3 ease;

  transition: all 0.3 ease;

  cursor: pointer;

}

.form button:hover,.form button:active,.form button:focus {

  background: #43A047;

}

.form .message {

  margin: 15px 0 0;

  color: #b3b3b3;

  font-size: 12px;

}

.form .back {

    color: #b3b3b3;

    font-size: 13px;

    text-align: start;

}

.form .back a {

  color: #3ec9a6;

  text-decoration: none;

}

.form .error {

    margin: 1px 0 3px;

    color: #EF3B3A;

    font-size: 15px;

}

.form .message a {

  color: #4CAF50;

  text-decoration: none;

}

.container {

  position: relative;

  z-index: 1;

  max-width: 300px;

  margin: 0 auto;

}

.container:before, .container:after {

  content: "";

  display: block;

  clear: both;

}

.container .info {

  margin: 50px auto;

  text-align: center;

}

.container .info h1 {

  margin: 0 0 15px;

  padding: 0;

  font-size: 36px;

  font-weight: 300;

  color: #1a1a1a;

}

.container .info span {

  color: #4d4d4d;

  font-size: 12px;

}

.container .info span a {

  color: #000000;

  text-decoration: none;

}

.container .info span .fa {

  color: #EF3B3A;

}

body {

  background: #76b852;

  background: -webkit-linear-gradient(right, #76b852, #8DC26F);

  background: -moz-linear-gradient(right, #76b852, #8DC26F);

  background: -o-linear-gradient(right, #76b852, #8DC26F);

  background: linear-gradient(to left, #76b852, #8DC26F);

  font-family: "Roboto", sans-serif;

  -webkit-font-smoothing: antialiased;

  -moz-osx-font-smoothing: grayscale;

}

</style>

<body>

    <div class="login-page">

        <div class="form">

            <p class="back"><a href="{{ url\_for('showBooks',page=1) }}">&larr;На главную</a></p>

          <form action="{{ url\_for('loginUser')}}" class="login-form" method="POST">

            <input type="text" placeholder="Email" name="email"/>

            <input type="password" placeholder="Пароль" name="password"/>

            {% if error%}

                <p class="error">Error:{{ error }}</p>

            {% endif %}

            <button type="submit">Войти</button>

            <p class="message">Не зарегистрированы? <a href="{{ url\_for('registrationUser') }}">Создать аккаунт</a></p>

          </form>

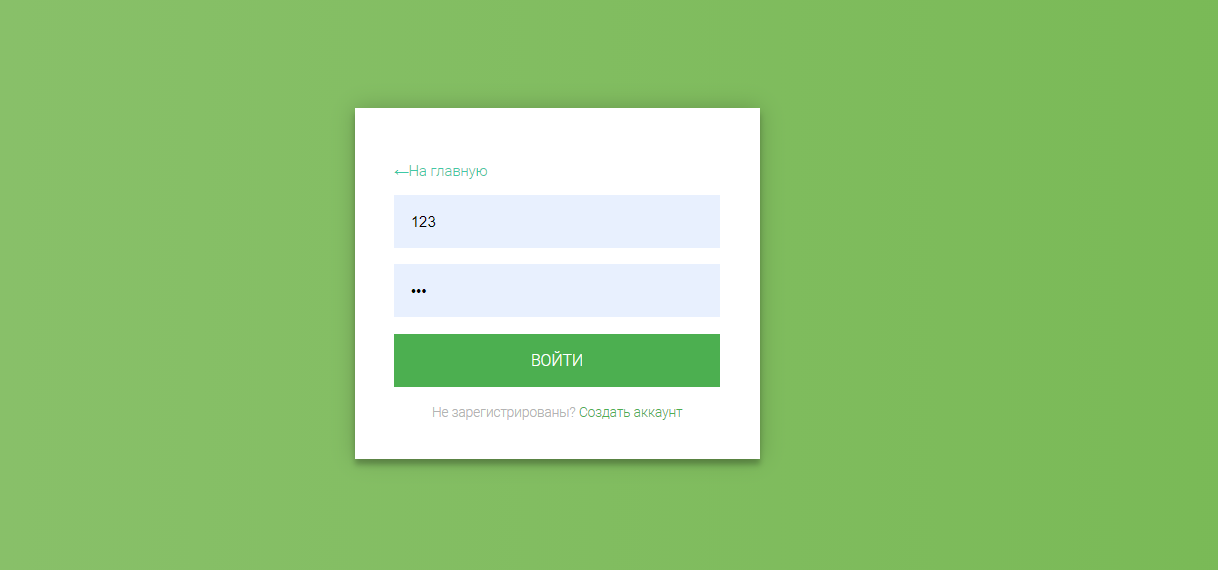
        </div>

      </div>

</body>

</html>

Визуальное отображение формы логина приведено на рисунке 5

Рисунок 5 — Форма логина

Ниже приведена html-верстка страницы регистрации, а на рисунке 6 показано ее визуальное отображение

<!DOCTYPE html>

<html>

  <head>

    <title>Регистрация</title>

  </head>

<style type="text/css">

    @import url(https://fonts.googleapis.com/css?family=Roboto:300);

.login-page {

  width: 360px;

  padding: 8% 0 0;

  margin: auto;

}

.form {

  position: relative;

  z-index: 1;

  background: #FFFFFF;

  max-width: 360px;

  margin: 0 auto 100px;

  padding: 45px;

  text-align: center;

  box-shadow: 0 0 20px 0 rgba(0, 0, 0, 0.2), 0 5px 5px 0 rgba(0, 0, 0, 0.24);

}

.form .back {

    color: #b3b3b3;

    font-size: 13px;

    text-align: start;

}

.form .back a {

  color: #3ec9a6;

  text-decoration: none;

}

.form input {

  font-family: "Roboto", sans-serif;

  outline: 0;

  background: #f2f2f2;

  width: 100%;

  border: 0;

  margin: 0 0 15px;

  padding: 15px;

  box-sizing: border-box;

  font-size: 14px;

}

.form button {

  font-family: "Roboto", sans-serif;

  text-transform: uppercase;

  outline: 0;

  background: #4CAF50;

  width: 100%;

  border: 0;

  padding: 15px;

  color: #FFFFFF;

  font-size: 14px;

  -webkit-transition: all 0.3 ease;

  transition: all 0.3 ease;

  cursor: pointer;

}

.form button:hover,.form button:active,.form button:focus {

  background: #43A047;

}

.form .message {

  margin: 15px 0 0;

  color: #b3b3b3;

  font-size: 12px;

}

.form .message a {

  color: #4CAF50;

  text-decoration: none;

}

.form .register-form {

  display: block;

}

.form .error {

    margin: 1px 0 3px;

    color: #EF3B3A;

    font-size: 15px;

}

.container {

  position: relative;

  z-index: 1;

  max-width: 300px;

  margin: 0 auto;

}

.container:before, .container:after {

  content: "";

  display: block;

  clear: both;

}

.container .info {

  margin: 50px auto;

  text-align: center;

}

.container .info h1 {

  margin: 0 0 15px;

  padding: 0;

  font-size: 36px;

  font-weight: 300;

  color: #1a1a1a;

}

.container .info span {

  color: #4d4d4d;

  font-size: 12px;

}

.container .info span a {

  color: #000000;

  text-decoration: none;

}

.container .info span .fa {

  color: #EF3B3A;

}

body {

  background: #76b852; /\* fallback for old browsers \*/

  background: -webkit-linear-gradient(right, #76b852, #8DC26F);

  background: -moz-linear-gradient(right, #76b852, #8DC26F);

  background: -o-linear-gradient(right, #76b852, #8DC26F);

  background: linear-gradient(to left, #76b852, #8DC26F);

  font-family: "Roboto", sans-serif;

  -webkit-font-smoothing: antialiased;

  -moz-osx-font-smoothing: grayscale;

}

</style>

<body>

    <div class="login-page">

        <div class="form">

            <p class="back"><a href="{{ url\_for('showBooks',page=1) }}">&larr;На главную</a></p>

          <form action="{{ url\_for('registrationUser') }}" class="register-form" method="POST">

            <input type="text" placeholder="Имя" name="firstName" required/>

            <input type="text" placeholder="Фамилия" name="lastName" required/>

            <input type="password" placeholder="Пароль" name="pswd" required/>

            <input type="password" placeholder="Повторите пароль" name="pswd2" required/>

            <input type="text" placeholder="Email" name="email" required/>

            Дата рождения:<input type="date" name="birthDate"/>

            {% if error%}

                <p class="error">Error:{{ error }}</p>

            {% endif %}

            <button type="submit">Создать</button>

            <p class="message">Уже зарегистрированы? <a href="{{ url\_for('loginUser') }}">Войти</a></p>

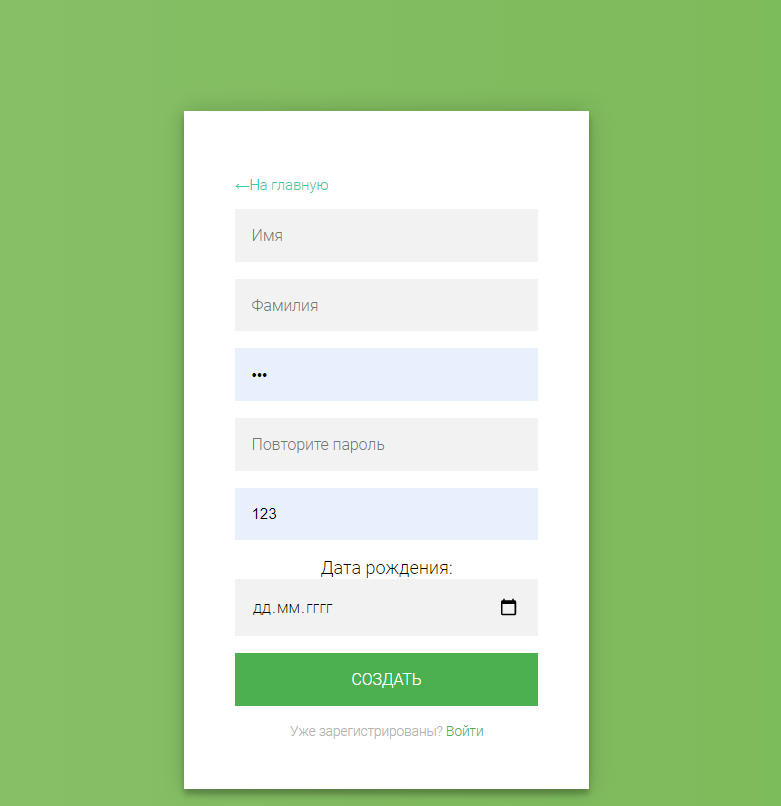
          </form>

        </div>

      </div>

</body>

</html>

Рисунок 6 — Форма регистрации

В дальнейшем проверка авторизации пользователя будет происходить либо соответствующим декоратором, либо получение объекта пользователя средствами flask-login.

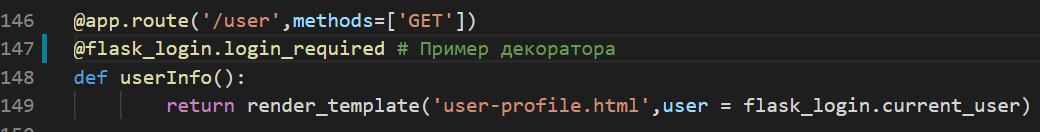
 Рисунок 1 — Пример использования декоратора

Рисунок 2 — Пример получения объекта пользователя

Для подробного исследования кода можно посетить репозиторий VCS:https://github.com/ponomaREG/viziProjectShop

# **Заключение**

В ходе выполненной работы были разработаны системы авторизации и аутентификации пользователей. Также стоит заметить, что flask-login взяло на себя львиную долю всей работы.