Министерство науки и высшего образования Российской Федерации

САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО  
Факультет Безопасности информационных технологий

Дисциплина: Разработка систем аутентификации и криптографии

Отчет

по лабораторной работе №2

«Методы аутентификации»

|  |  |
| --- | --- |
|  | Выполнил  Магистрант учебной группы N42514c  Васильев Роман Александрович    Проверил:  Федров Иван Романович |

Санкт-Петербург  
2020 г.

**Задача**: реализация механизма аутентификации в клиент-серверном веб-приложении

*Требования к реализации:*

* необходимо реализовать метод аутентификации в клиент-серверном приложении согласно варианту
* клиент должен представлять собой веб-страницу с формой авторизации пользователя
* сервер должен включать в себя две части:
  + - таблица идентификаторов (данные о пользователях для аутентификации: логин/пароль/токен/итд в зависимости от метода аутентификации)
    - процесс с реализованной логикой метода аутентификации

Общий сценарий работы программы при сдаче ЛР должен выглядеть так:

1. Запускается “Клиент” (веб-страница) с формой авторизации, в которую вводятся данные для аутентификации
2. После ввода данные отправляются на сервер и проходят проверку согласно реализованному методу аутентификации
3. При успешной аутентификации выполняется редирект на другую веб-страницу (заглушку) с надписью “Hello, <Имя пользователя>”

**\* Программа не должна разрешать переход на страницу-заглушку без успешной аутентификации (напрямую по URL)**

**Вариант №5**

Реализовать аутентификацию по паролю с подтверждением по email. В таблице идентификаторов должны храниться: логин, email, пароль, хеш временного кода подтверждения (MD5). Таблица идентификаторов должна представлять собой таблицу в реляционной БД, данные должны передаваться через SQL-запросы. При аутентификации на сервере сравниваются пароли и на email пользователя отправляется сгенерированный на сервере временный код подтверждения. На клиенте после отправки данных с паролем должен произойти редирект на форму для ввода временного кода подтверждения. После отправки кода на сервере сравниваются хеш пришедшего кода и хеш кода из БД (MD5). При совпадении хешей аутентификация считается успешной и происходит редирект на страницу-заглушку.

**Выполнение работы**

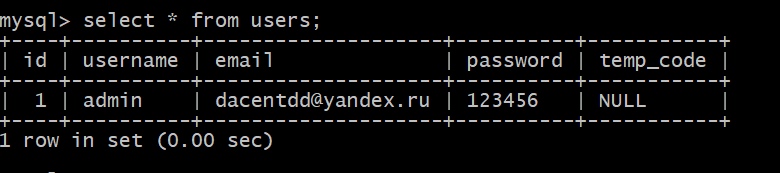
Для реализации был выбрал язык программирования Python 3.9 с веб-фреймворком Flask.

Исходный код представлен в Приложении и по ссылке:

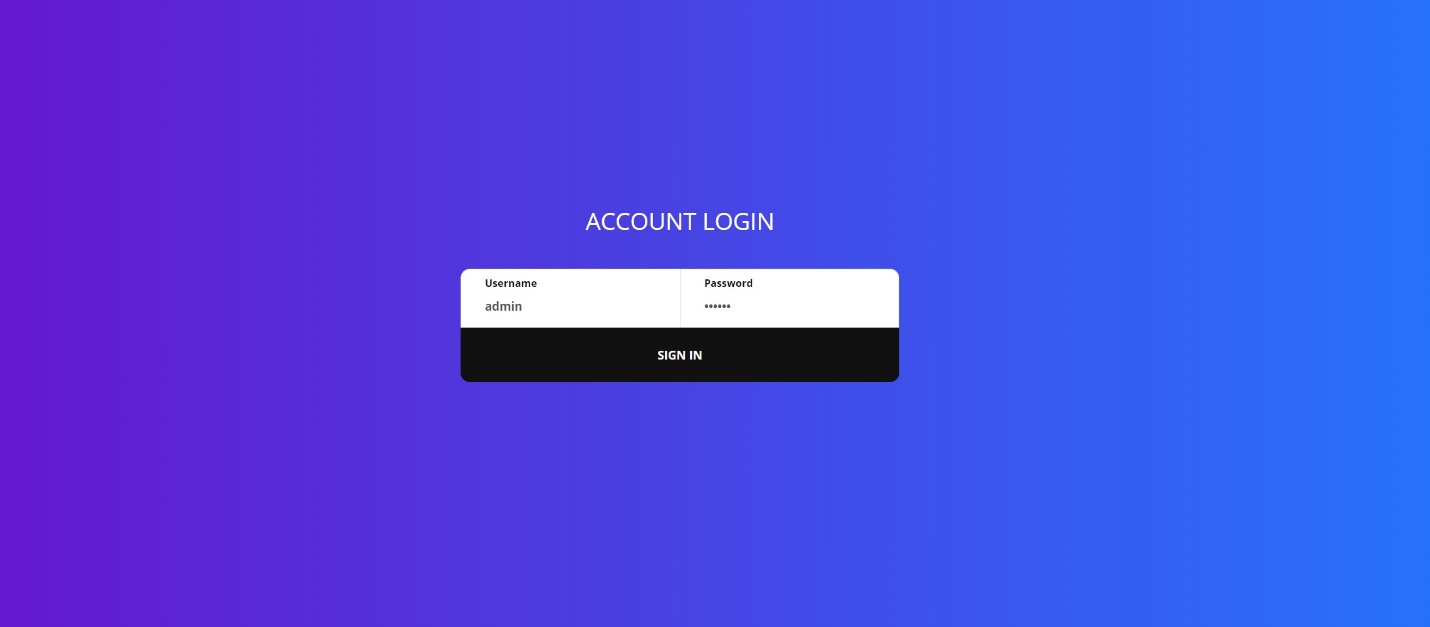
**https://github.com/DaCentDD/cryptography/tree/master/Auth**

Скриншоты, демонстрирующие работу веб-сервиса:

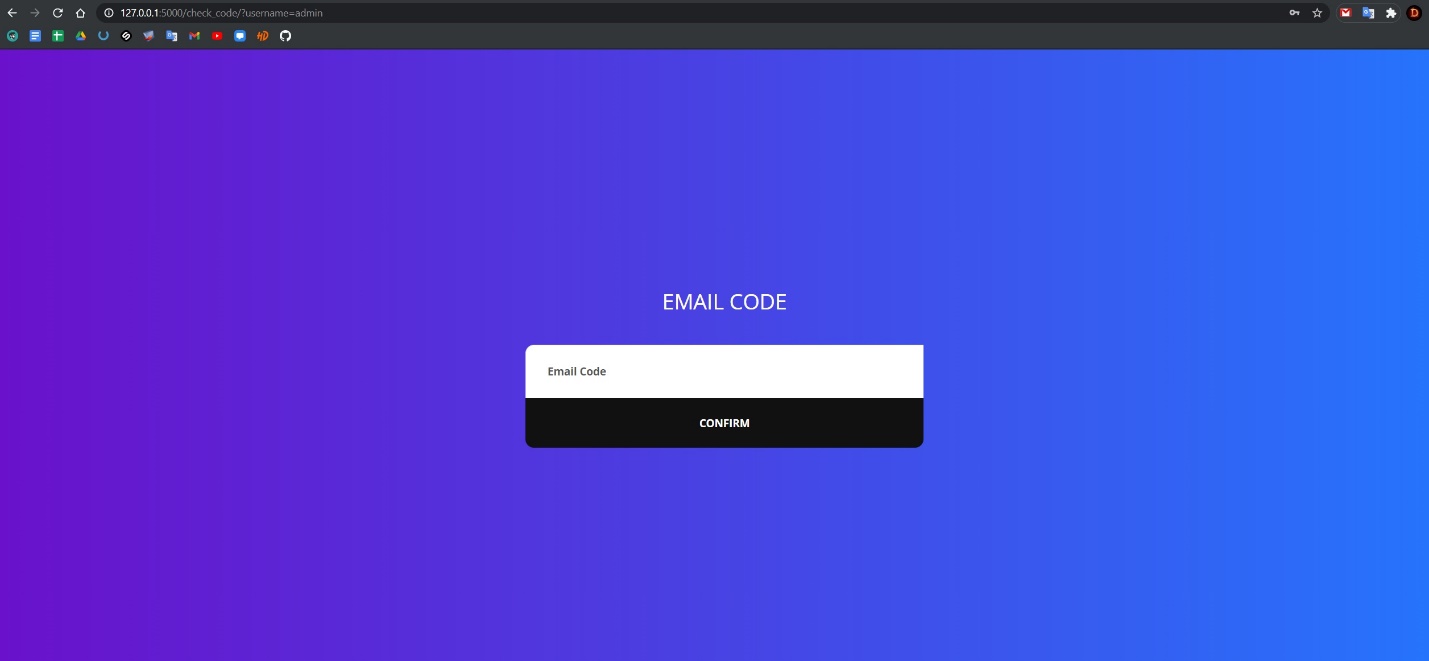
1) Таблица в базе данных MySQL с зарегистрированным пользователем admin.

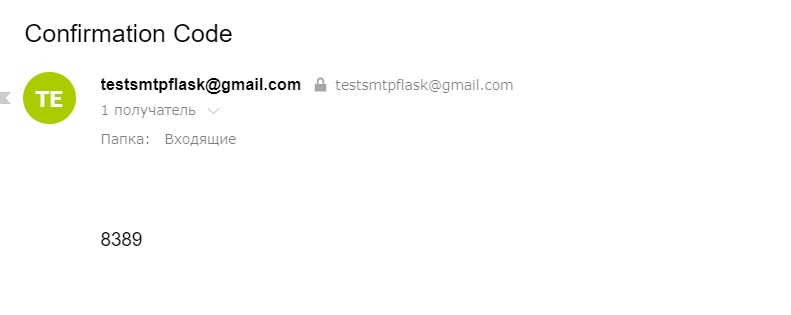


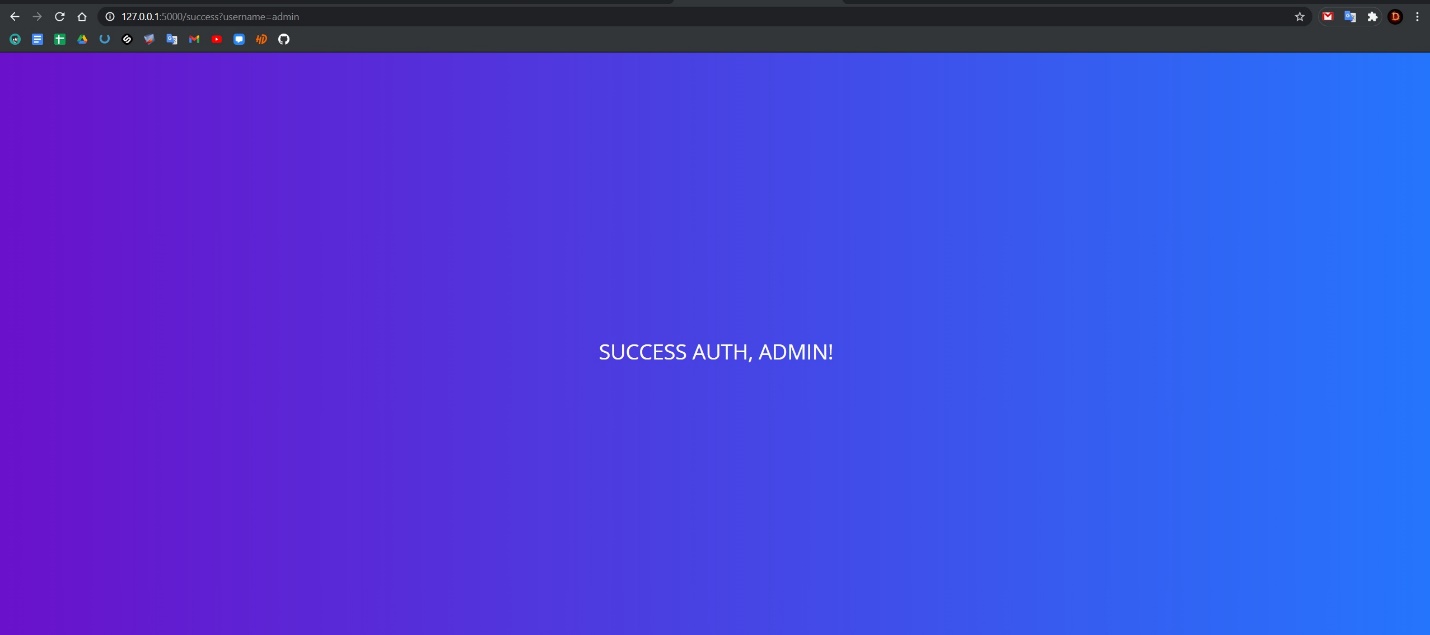
2) Главная страница.



3) Страница ожидания временного кода, отправленного на почту после успешной аутентификации по паролю.



4) Пришедший на почту пользователя код подтверждения.

5) Заглушка, открывающаяся в случае успешной аутентификации и авторизации.

**Приложение**

Файл **app.py**

import os  
import random  
  
from flask import Flask, render\_template, request, redirect, url\_for  
from hashlib import md5  
from flask\_sqlalchemy import SQLAlchemy  
from flask\_migrate import Migrate  
from flask\_login import LoginManager, UserMixin, login\_required, login\_user  
from flask\_mail import Mail, Message  
  
try:  
 from local\_settings import \* # Для секретной информации  
except ImportError:  
 pass  
  
app = Flask(\_\_name\_\_)  
app.config['SECRET\_KEY'] = os.urandom(24)  
app.config['SQLALCHEMY\_DATABASE\_URI'] = SQLALCHEMY\_DATABASE\_URI # Обязательно  
db = SQLAlchemy(app)  
migrate = Migrate(app, db)  
login\_manager = LoginManager(app)  
login\_manager.login\_view = 'index'  
app.config['MAIL\_SERVER'] = 'smtp.googlemail.com'  
app.config['MAIL\_PORT'] = 465  
app.config['MAIL\_USE\_TLS'] = False  
app.config['MAIL\_USE\_SSL'] = True  
app.config['MAIL\_USERNAME'] = 'testsmtpflask@gmail.com'  
app.config['MAIL\_DEFAULT\_SENDER'] = 'testsmtpflask@gmail.com'  
app.config['MAIL\_PASSWORD'] = MAIL\_PASSWORD # Обязательно  
mail = Mail(app)  
  
  
@login\_manager.user\_loader  
def load\_user(user\_id):  
 return db.session.query(Users).get(user\_id)  
  
  
class Users(db.Model, UserMixin): # Создаание таблицы в БД  
 \_\_tablename\_\_ = 'users'  
 id = db.Column(db.Integer(), primary\_key=True)  
 username = db.Column(db.String(50), nullable=False, unique=True)  
 email = db.Column(db.String(100), nullable=False)  
 password = db.Column(db.String(100), nullable=False)  
 temp\_code = db.Column(db.String())  
  
 def set\_temp\_code(self, code: str): # Занести захэшированный код подтвержденния в таблицу  
 self.temp\_code = md5(code.encode()).hexdigest()  
 db.session.commit()  
  
 def check\_temp\_code(self, user\_code: str): # Сравнить код в таблице с введенными пользователем  
 return True if self.temp\_code == md5(user\_code.encode()).hexdigest() else False  
  
 def clean\_temp\_code(self): # По окончании проверки удалить хэш кода  
 self.temp\_code = None  
 db.session.commit()  
  
  
@app.route('/', methods=['POST', 'GET'])  
def index():  
 if request.method == 'POST': # Запрос к данным формы  
 username = request.form.get('username')  
 password = request.form.get('pass')  
 user = db.session.query(Users).filter(Users.username == username).first()  
 if user.password == password: # Если пароль введен верно  
 return redirect(url\_for('check\_code', username=user.username))  
 return render\_template("index.html")  
  
  
@app.route('/success', methods=['GET'])  
@login\_required  
def success(): # В случае успеха авторизации перенаправление на страницу-заглушку  
 username = request.args.get('username')  
 return render\_template("success.html", username=username)  
  
  
@app.route('/check\_code/', methods=['POST', 'GET'])  
def check\_code(): # Отправка кода подтверждения на email и его проверка  
 if request.method == 'POST':  
 user\_code = request.form.get('email\_code')  
 username = request.args.get('username')  
 user = db.session.query(Users).filter(Users.username == username).first()  
 if user.check\_temp\_code(user\_code):  
 login\_user(user)  
 user.clean\_temp\_code()  
 return redirect(url\_for('success', username=username))  
 else:  
 user.clean\_temp\_code()  
 return redirect(url\_for('check\_code', username=username))  
 username = request.args.get('username')  
 if username is None:  
 return redirect(url\_for('index'))  
 user = db.session.query(Users).filter(Users.username == username).first()  
 temp\_code = str(random.randint(1111, 9999))  
 user.set\_temp\_code(temp\_code)  
 msg = Message("Confirmation Code", recipients=[user.email])  
 msg.body = temp\_code  
 mail.send(msg)  
 return render\_template("check\_code.html", username=username)  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 app.run()