

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 N42514с
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 выбран язык программирования Python 3.9 с веб-фреймворком Flask.

Исходный код представлен в Приложении и по ссылке:

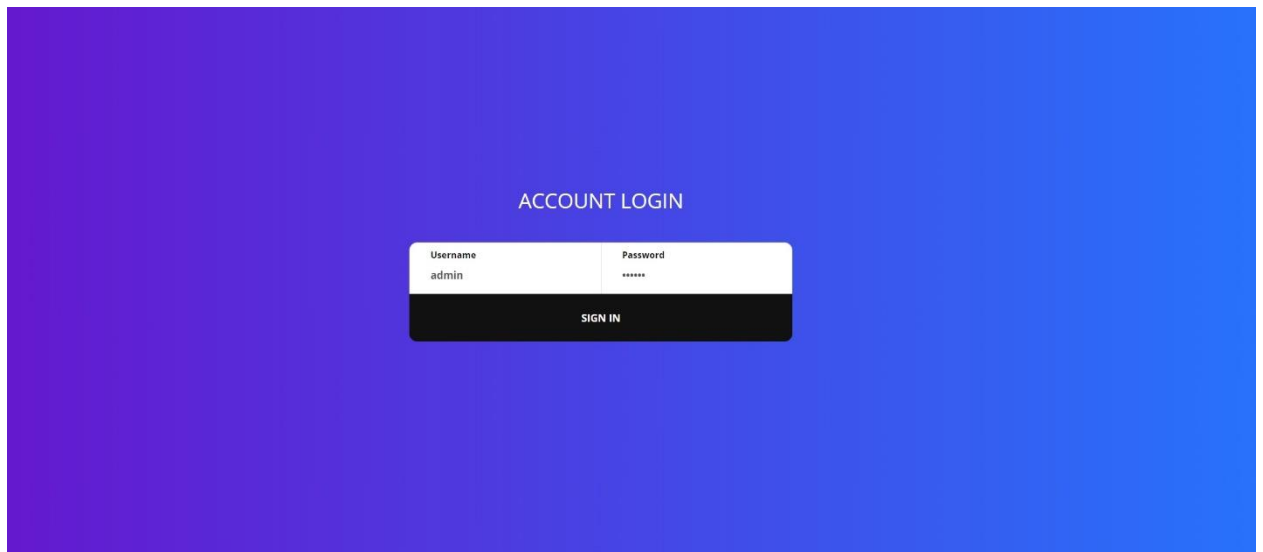
<https://github.com/DaCentDD/cryptography/tree/master/Auth>

Скриншоты, демонстрирующие работу веб-сервиса:

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

```
mysql> select * from users;  
+-----+-----+-----+-----+-----+  
| id | username | email | password | temp_code |  
+-----+-----+-----+-----+-----+  
| 1 | admin | dacentdd@yandex.ru | 123456 | NULL |  
+-----+-----+-----+-----+-----+  
1 row in set (0.00 sec)
```

2) Главная страница.

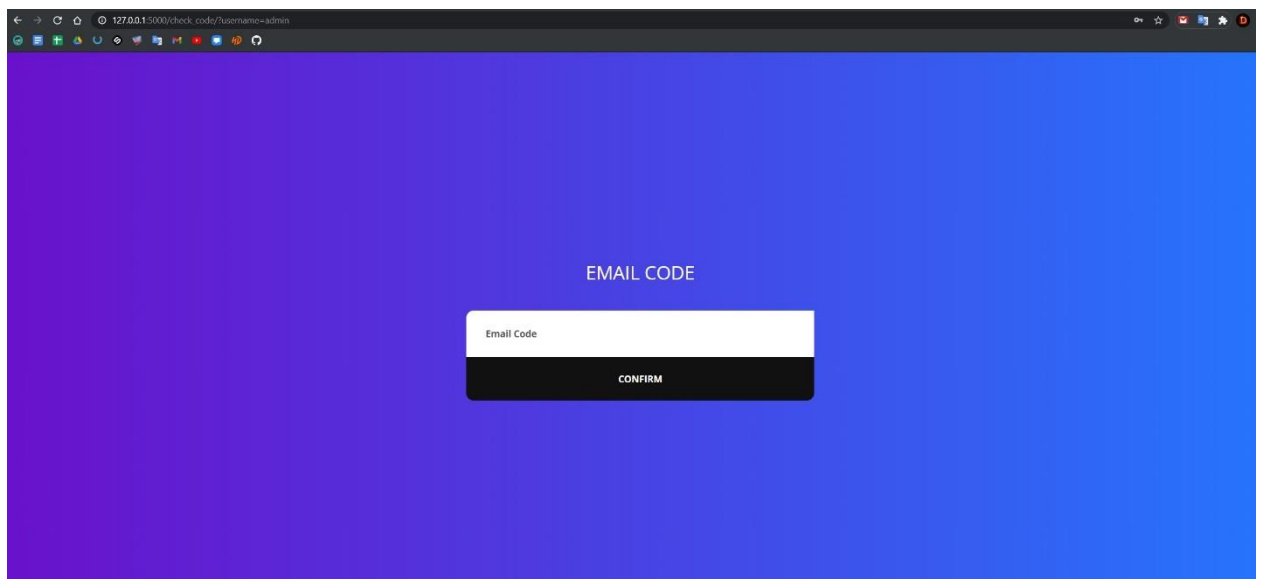


ACCOUNT LOGIN

Username	Password
admin	*****

SIGN IN

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

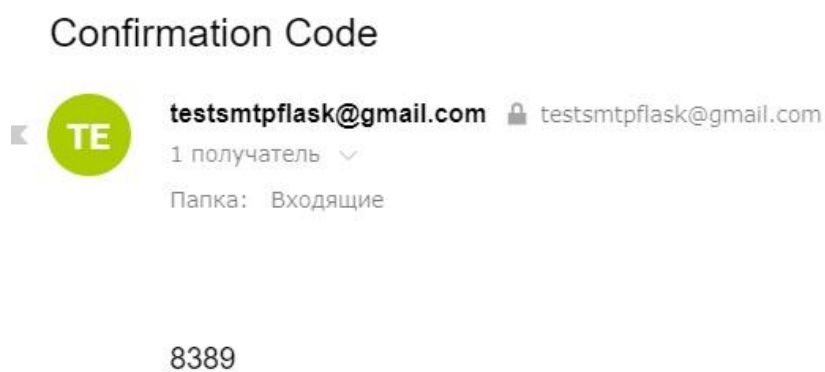


EMAIL CODE

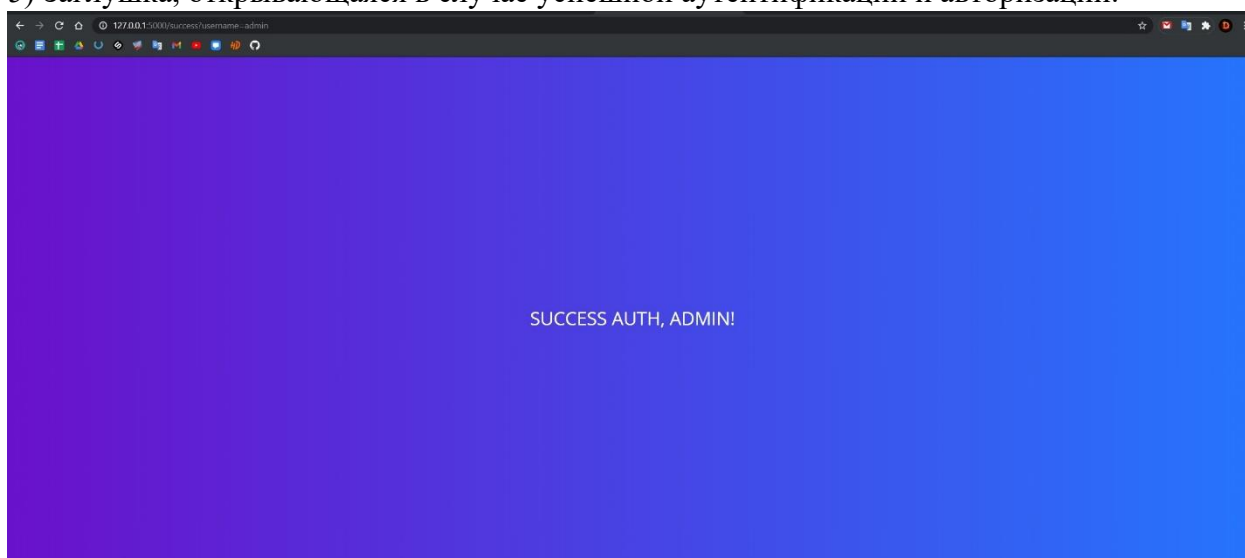
Email Code

CONFIRM

4) Пришедший на почту пользователя код подтверждения.



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



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): # Создание 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 захешированный код подтверждения 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()

```